SPLIT- COMPANY RETROSPECTIVES AT SCALE

Mestrado em Eng.ª Informática – Computação Móvel

Nuno Miguel Hilário Caseiro

2202297

Leiria, Março de 2023

SPLIT- COMPANY RETROSPECTIVES AT SCALE

Mestrado em Eng.ª Informática – Computação Móvel

Nuno Miguel Hilário Caseiro

Trabalho de projeto realizado sob a orientação Professor Doutor Marco António de Oliveira Monteiro (marco.monteiro@ipleiria.pt) e sob supervisão do Eng.º Luís Soares (l.soares@kigroup.de) e Engº Gonçalo Dias (g.dias@kigroup.de)

Leiria, Março de 2023

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Agradecimentos

Agradeço ao orientador do projeto Professor Marco Monteiro pela disponibilidade, apoio e acompanhamento proporcionado durante o desenvolvimento do mesmo.

Pretendo prestar um agradecimento especial ao *Engineer Manager* Gonçalo Dias que orientou o projeto na empresa e, durante o desenvolvimento do mesmo, ofereceu total apoio, organizou todo o processo e geriu os requisitos da melhor forma possível, promovendo um ambiente de excelência o que permitiu cumprir os objetivos propostos com sucesso.

Quero agradecer à Anja Foerster, *designer* e *user-experience researcher*, por ajudar a elevar o processo de conceção do SPLIT a outro nível, transformando o projeto num produto real e robusto através do método executado na fase de conceção e *design* do produto. Foi fundamental para o seu sucesso.

Agradeço aos meus colegas de trabalho que participaram direta ou indiretamente na execução do projeto e que contribuíram para o estado atual do produto.

Os meus amigos foram sem dúvida um grande apoio e por isso quero agradecer-lhes, uma vez que sempre se disponibilizaram para receber as minhas preocupações e sempre me incentivaram a ultrapassar os desafios encontrados.

Um agradecimento especial pelo apoio incondicional da minha família e por me incentivarem a nunca desistir dos meus objetivos.

A todos vós, muito obrigado.

Resumo

As retrospetivas são cerimónias efetuadas em equipas de trabalho que têm o objetivo de definir e compreender os aspetos positivos, menos positivos e as ações a tomar para melhorá-los. Nestas sessões é importante que todos os colaboradores tenham oportunidade de expressar a sua opinião e que esta seja valorizada. Por norma, as retrospetivas têm o apoio de um quadro *kanban* onde se introduzem as opiniões de cada membro e que são representadas por cartões.

Em equipas grandes surge a dificuldade de dar oportunidade a todos os participantes em expressar a sua opinião. Como tal, é fundamental definir um mecanismo que resolva este problema. Com esse objetivo, pode-se dividir uma equipa grande em subequipas e cada uma destas deve realizar a sua retrospetiva, possibilitando que todos os membros demonstrem a sua opinião. Posteriormente, os resultados obtidos nas retrospetivas de cada subequipa devem ser unificados e gerados itens de ação que pretendem resolver os aspetos que correram menos bem.

O projeto SPLIT surge com o objetivo de resolver o problema referido, otimizar e automatizar o processo de retrospetivas em qualquer tipo de equipa, principalmente com elevado número de elementos. A solução pretende dividir as equipas noutras mais pequenas, criar automaticamente os respetivos quadros *kanban* e oferecer um conjunto de automações dentro destes, possibilitar agendamentos e a integração com o *slack*, o que permite de forma a automática enviar mensagens sobre todas as fases e ações relevantes durante o processo de retrospetivas.

Na fase anterior ao desenvolvimento, decorreu a fase de *product design* e conceção do produto onde foi executado um processo de pesquisa que serviu para compreender como é possível otimizar o processo e que requisitos são essenciais para a plataforma colmatar as necessidades dos utilizadores e da empresa, tendo o foco na execução de retrospetivas em equipas grandes. Este processo terminou com o levantamento de requisitos, a execução de um *design system* e de todo o *design* da aplicação.

A plataforma a nível tecnológico é composta essencialmente por dois módulos. O *backend* foi construído utilizando a *framework* NestJS e encapsula as regras de negócio, acede à base de dados e providencia uma REST API e um *socket gateway* com o propósito

de servir a aplicação web. Esta foi desenvolvida com recurso a uma *framework* de *react* denominada de Next.js. A aplicação web providencia uma interface, focada na experiência de utilização, que permite visualizar e interagir com a plataforma.

O sistema foi avaliado com testes manuais, alguns unitários e de usabilidade que contribuíram para a correção de erros e melhor perceção de possíveis melhorias.

Palavras-chave: retrospetivas, equipas, quadros, agendamentos, automatização, *product design*

Abstract

The retrospectives are ceremonies held in teams aiming to define and understand the positive and the less positive aspects and the actions to be taken to improve them. The retrospectives usually are supported by a Kanban board where each member's opinions can be entered and represented by cards.

In large teams, it is difficult to give all participants the opportunity to express their opinions. As such, defining a mechanism to solve this problem is fundamental. To this end, it's important that a large team be divided into sub-teams and each of these should perform its retrospective, allowing all the members to demonstrate their opinion. Afterwards, the results obtained from each sub-team should be unified and action items generated that aim to resolve the aspects that went less well.

The SPLIT project aims to solve this problem, optimize, and automate the retrospective process in any kind of team, mainly in large teams, by dividing them into smaller teams, automatically creating the respective Kanban boards and inside them providing some automation, enabling scheduling, and integrating it with slack, which automatically sends messages about all the relevant phases and actions during the retrospective process.

In the phase prior to development, the product design and product conception phase took place where a long research process was carried out to understand how the process can be optimized and which requirements are essential for the platform to meet the user's needs, focusing on the execution of retrospectives in large teams. This process ended with the requirements gathering, the execution of a design system and the entire application design.

At the technological level, the platform is essentially composed of two modules. The backend was built using the NestJS framework that encapsulates the business rules, accesses the database, and provides a REST API that serves the web application. This was developed using a react framework called Next.js. The web application provides an interface, focused on the user experience, that allows visualization and interaction with the platform.

The system was evaluated with manual tests, some unit tests and usability tests that contributed to error correction and a better perception of possible improvements.

Keywords: retrospectives, teams, boards, schedules, automation, product design

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

AD	Active directory
AWS	Amazon web services
BD	Base de dados
BSON	Binary JSON
CD	Continuous delivery
CI	Continuous integration
DDD	Domain-driven design
DNS	Domain name system
DTO	Data transfer object
E2E	End-to-end
EM	Engineer manager
ESTG	Escola Superior de Tecnologia e Gestão
GPDR	General Data Protection Regulation
HOC	High order component
IA	Inteligência artificial
IT	Information technology
JS	Javascript
JSON	Javascript object notation
MVP	Minimum value product
NLP	Natural language processing
ODM	Object document mapping
ORM	Object-relation mapping
PR	Pull request
SSR	Server-side rendering
SSO	Single-sign-on
SOC2	System and organization controls type 2
US	User story
WIP	Work in progress

1. Introdução

As retrospetivas segundo os autores de "*Learning in the large - an exploratory study of retrospectives in large-scale agile development*" [1], são momentos em que uma equipa de desenvolvimento se reúne com o intuito de efetuar o balanço do trabalho desenvolvido ao longo de um determinado intervalo de tempo e por isso é referido que têm elevada importância no desenvolvimento de um produto de *software*. No livro "*Agile Practice Guide | Project Management Institute*" [2], essa importância é reforçada, indicando que se trata da "prática mais importante quando se executa desenvolvimento agile" e que é "a cerimónia mais importante na metodologia Scrum". Esta importância deve-se ao facto de ser possível explorar e avaliar os resultados do trabalho numa equipa, em cada iteração ou fase de um projeto, com o intuito de "compreender, melhorar e adaptar o processo" [1]. Uma vez que as metodologias *agile* se focam no aperfeiçoamento do processo de desenvolvimento, através deste método é possível rever os pontos que correram bem, os pontos que correram menos bem e aqueles que podem ser melhorados. A partir destes criam-se itens de ação para os resolver, quer seja num projeto (pequeno ou grande) ou a nível de uma organização. O objetivo final é otimizar os resultados e o processo, evitando o desperdício.

Na xgeeks, empresa na qual foi implementado o trabalho de projeto e apresentado neste documento, as retrospetivas são efetuadas ao nível das equipas de projeto, mas também ao nível da empresa e a plataforma a ser apresentada neste documento tem o foco nestas últimas, contudo ambas podem ser executadas através deste produto. Com o intuito de permitir que todos os colaboradores consigam de forma assíncrona apresentar a sua opinião sem que esta seja desvalorizada, foi necessário implementar um processo que possibilite que mesmo em grupos com elevado número de pessoas, todas as opiniões sejam contabilizadas.

Com a ambição de resolver este problema, na xgeeks, foi definido e adotada uma estratégia que determina que as retrospetivas devem ter várias fases e os membros devem ser divididos em grupos de menor dimensão. Portanto, na primeira fase é efetuada a divisão dos membros da equipa em subequipas e para cada uma é eleito um responsável que trata de coordenar a retrospetiva do grupo. Isto é, necessita de criar um quadro *kanban* no formato de três colunas (o que correu bem, o que correu menos bem, como melhorar o que correu menos bem) numa ferramenta externa que é partilhado com os participantes. Estes colocam

os cartões que demonstram a opinião de cada um na respetiva coluna. O responsável de cada subequipa precisa também de agendar uma reunião para discutir os cartões criados. A divisão referida é efetuada mensalmente por um simples *script* alojado numa máquina virtual e na *cloud,* que divide em grupos os membros existentes no canal principal de retrospetivas do *slack* da empresa. Para cada equipa é criado um canal do *slack*, os membros são convidados para cada e é gerada uma mensagem no canal principal das retrospetivas onde é anunciada a divisão efetuada pelo script com a constituição das equipas. Adicionalmente, é criado outro canal apenas constituído pelos responsáveis selecionados para cada subequipa onde coordenam o momento de seguinte.

Na segunda fase, após a criação e discussão dos cartões em cada subequipa, um dos responsáveis deve criar na plataforma externa um novo quadro, designado como principal, que reúne os cartões criados por cada subequipa, no entanto, é necessário que todos sejam manualmente inseridos. O processo de cópia e agrupamento dos cartões semelhantes no quadro principal e discussão dos mesmos entre os responsáveis é efetuado numa segunda reunião. Após a reunião, dá-se início ao processo de votação e para tal é necessário partilhar no *slack* o *link* para o quadro principal e indicar que este se encontra para votação.

Findada a fase de votação é efetuada uma reunião entre os responsáveis e os *stakeholders,* que são os membros da empresa com responsabilidade de tomada de decisão, para definir os itens de ação que pretendem resolver os tópicos criados na coluna que corresponde ao que correu menos bem e por isso necessita de ser melhorado.

Apresentado este processo, é importante reforçar que existe dependência de ferramentas externas e é necessário desempenhar várias tarefas manuais, o que prejudica em parte a produtividade de alguns colaboradores e a eficiência do processo. Esta situação possibilita a introdução de uma oportunidade de melhoria com o intuito de otimizar este processo, automatizando todas as tarefas manuais, centralizar todas as ferramentas numa só plataforma com total controlo e acrescentar funcionalidades relevantes que promovem a diminuição de tempo gasto, desnecessariamente. Com isto, também se pretende promover a participação dos colaboradores em expressar a sua opinião permitindo assim melhorar o dia-a-dia na empresa. Adicionalmente, foi considerado que o projeto seria um ótimo ponto de partida para a introdução e adoção de novas tecnologias até então pouco ou nada usadas na empresa, em projetos com clientes, tais como a *framework* Next.js e NestJS mas também para a empresa se aproximar da comunidade ao definir este projeto como *open-source.* Este projeto

também tem o intuito de contribuir para a aprendizagem dos novos membros da empresa, isto é, pretende-se que seja um projeto que replique a forma de trabalhar em projetos com clientes, quer seja a nível de metodologias como de *frameworks* utilizadas o que possibilita a alocação de novos membros que necessitam de aprimorar os conhecimentos.

No seguimento da oportunidade de melhoria mencionada anteriormente, surge a criação do projeto descrito neste documento, o SPLIT, que tem o objetivo de unificar todas as ferramentas e funcionalidades numa só plataforma. Possibilitando principalmente, a criação de equipas e a criação de retrospetivas, agendadas ou não, e que seguem o processo anteriormente descrito, no entanto, com um conjunto de automações e novas funcionalidades. Nomeadamente a divisão de uma equipa e a criação automática de quadros para cada subequipa e para o quadro principal mantendo uma relação de parentalidade para ser possível automaticamente mover os cartões de um quadro de uma subequipa para o principal; manter e adicionar novas funcionalidades nos quadros; manter a criação dos canais do *slack* e adicionalmente notificar todas as ações e fases da retrospetiva através de mensagens; possibilitar a transição entre fases numa retrospetiva; centralizar toda a informação e apresentar uma interface apelativa com ótima experiencia de utilização; adicionalmente pretende-se agendar reuniões de forma automática.

Com o intuito de criar uma plataforma capaz de unificar todos os requisitos mencionados foi desenvolvido um sistema de *software* composto por três componentes principais:

1. *Backend* – é a base de todo o sistema, que encapsula regras de negócio e providencia uma interface que permite as aplicações clientes comunicarem com esta.
2. Base de dados – armazena a informação do sistema.
3. Aplicação web – providencia uma interface gráfica para os utilizadores interagirem com o sistema e criarem retrospetivas.

Adicionalmente foi criada uma infraestrutura com dois ambientes capaz de suportar o desenvolvimento, mas também disponibilizar a plataforma aos utilizadores finais. Com o intuito de automatizar a integração e a disponibilização foi criada uma *pipeline* de CI/CD

A empresa, uma vez que tem como objetivo manter proximidade com a comunidade, considerou que seria uma oportunidade de partilhar este projeto com outros interessados e por isso foi definido que seria *open-source* e o código fonte disponibilizado no *Github*

permitindo a colaboração de qualquer contribuidor. A infraestrutura e alojamento tem de ser disponibilizada e tratado por quem quiser usufruir do projeto.

Com o intuito de tornar o projeto num verdadeiro produto promoveu-se uma fase de conceção e design do produto que foi coordenada por um *designer* que também é *product researcher.* Nesta fase, prévia à implementação, foi efetuado um trabalho de pesquisa exaustivo com o intuito de compreender as necessidades dos utilizadores e como estes podem contribuir para sugerir melhorias no processo de execução de retrospetivas. Através deste trabalho foi possível definir o produto por completo. Esta fase foi finalizada com a execução total do *design* do produto, tendo em conta todos os aspetos necessários.

O presente documento, surge no âmbito da unidade curricular de Projeto do mestrado em Engenharia Informática – Computação Móvel do Instituto Politécnico de Leiria, no ano letivo 2021/2022 e descreve o trabalho realizado na empresa xgeeks Portugal [3], sediada em Leiria e integrante de um grupo de empresas denominado de KI Group[1]. Este projeto visou a aplicação e melhoria de competências associadas à área de formação, ao ensaio de práticas ajustadas no mundo do trabalho e a integração, uma vez que o processo de desenvolvimento deste projeto tentou aproximar-se ao máximo das metodologias de trabalho aplicadas em projetos com clientes. Um dos pontos principais foi a criação de hábitos e sentido de responsabilidade uma vez que o SPLIT foi iniciado com este projeto com a empresa e foi dada ao autor do mesmo total responsabilidade e propriedade sob o produto, e isto implica decisões tecnológicas e de implementação do mesmo, mas também ao nível de gestão de tarefas e coordenação da equipa que foi crescendo ao longo do desenvolvimento. Isto significa que perto do fim do projeto de empresa foi dada a oportunidade de geri-lo, permitindo trabalhar competências de gestão de projetos e essa gestão será continuada até ao fim do produto.

No que diz respeito à estrutura do documento, este é composto por 8 capítulos e estes por múltiplas secções que pretendem demonstrar detalhadamente o trabalho desenvolvido ao longo da implementação do projeto.

No capítulo **1 - Introdução**, - é descrito o que é uma retrospetiva, como estas são efetuadas na empresa, as ferramentas que são utilizadas para a execução das mesmas, os problemas que existem com o processo atual e como estes podem ser colmatados com o

[1] https://kigroup.de

projeto apresentado neste documento. São descritas de forma breve as funcionalidades principais e qual a constituição do sistema de software implementado. Finalmente são apresentados os objetivos técnicos e pessoais que o presente projeto pretendeu atingir.

No capítulo **2 - Trabalho relacionado** - é efetuado o levantamento de soluções existentes no mercado cujo funcionamento apresenta aspetos semelhantes ao sistema desenvolvido neste projeto. Além disso, é apresentada e efetuada uma comparação das soluções que é concluída com uma análise crítica e a motivação, que pretende apontar as mais valias diferenciadoras da solução do projeto apresentado neste documento.

No capítulo **3 - Planeamento do processo de desenvolvimento**, - tal como o nome indica, é apresentado o planeamento cronológico e flexível definido para as três fases do projeto, a fase de conceção e *design* do produto, a fase de desenvolvimento e a fase de testes.

No capítulo **4 - Metodologia de desenvolvimento**, - é efetuado um enquadramento teórico de conceitos relacionados com metodologias de desenvolvimento *agile* e é exposto detalhadamente como a metodologia definida foi utilizada ao longo do projeto e quais as ferramentas que serviram de suporte para a implementação da mesma.

No capítulo **5 - Pesquisa, conceção e desenho do produto**, - é apresentado o processo efetuado com a *designer* com o intuito de pensar e desenhar o produto, levantando e definindo os requisitos essenciais e necessários para a otimização do processo descrito anteriormente, idealizando e oferecendo a melhor experiencia de utilização com uma interface apelativa.

No capítulo **6 - Implementação**, - é descrita a arquitetura geral do sistema através dos diagramas que representam os diferentes níveis do modelo C4 e nos quais são apresentados os relacionamentos entre os vários componentes constituintes. São também apresentadas as tecnologias utilizadas na implementação do produto. Na respetiva secção é apresentada a base de dados, quais são as suas entidades e as relações entre elas. Na secção seguinte é apresentado um enquadramento teórico sobre padrões arquiteturais e de desenho de software e são descritos os detalhes de implementação do *backend* e como os padrões foram aplicados. Na próxima secção é apresentada a aplicação web e os seus detalhes de implementação. Na penúltima secção são apresentadas as funcionalidades e alguns detalhes de implementação. Este capítulo fecha com a apresentação da infraestrutura e *pipeline* de CI/CD.

No capítulo **7 - Testes e resultados**, - são descritos os testes unitários e de usabilidade efetuados ao sistema e a respetiva análise dos resultados. Neste capítulo é também descrito o estado atual do projeto, provando que é utilizado na empresa e que continua a ser desenvolvido.

No capítulo **8 - Conclusão e trabalho futuro**, - são apresentadas as conclusões da realização do projeto e são descritos os próximos passos do desenvolvimento do produto.

2. Trabalho relacionado

Neste capítulo são apresentadas algumas das plataformas existentes no mercado que se enquadram no âmbito da execução de retrospetivas. Na secção 2.1 pode-se verificar a análise efetuada a algumas ferramentas, antes da conceção do produto, de forma a compreender as lacunas, benefícios e funcionalidades fundamentais para uma plataforma que pretende auxiliar na execução de retrospetivas. Na secção 2.2 é apresentada uma comparação entre os projetos relacionados que também são comparadas com o projeto apresentado neste documento, com o intuito de perceber o que existe atualmente no mercado para otimizar o processo de retrospetivas. Na secção 2.3 é apresentada a motivação para execução deste projeto e as funcionalidades que distinguem o produto SPLIT.

2.1. Soluções Existentes no Mercado

Nesta secção são apresentadas e analisadas algumas soluções existentes no mercado no que diz respeito a aplicações que permitem efetuar retrospetivas, nomeadamente as ferramentas EasyRetro, MetroRetro, Reetro e Retrotool.io.

2.1.1. Easyretro

A plataforma EasyRetro [4], é uma das mais completas nas funcionalidades que proporciona aos utilizadores. Segundo os autores, este projeto iniciou-se em 2015 e é referenciado que é uma das ferramentas de retrospetivas mais utilizada no mundo. "*I created as an open-source project to help my team to do better retrospectives. The board got famous inside the company and then many other companies started using it as well.*", é uma frase dita pelo criador da plataforma [5] que permite compreender a necessidade e a utilidade deste tipo de ferramentas e como foi adotada por diversas entidades.

Esta aplicação permite autenticação normal ou por *single-sign-on* (SSO) através da google. Permite a criação de novos quadros *kanban*, como é possível verificar no exemplo da Figura 1. Esta apresenta um quadro com várias colunas e com cartões que podem ser votados e às quais podem ser adicionados comentários.

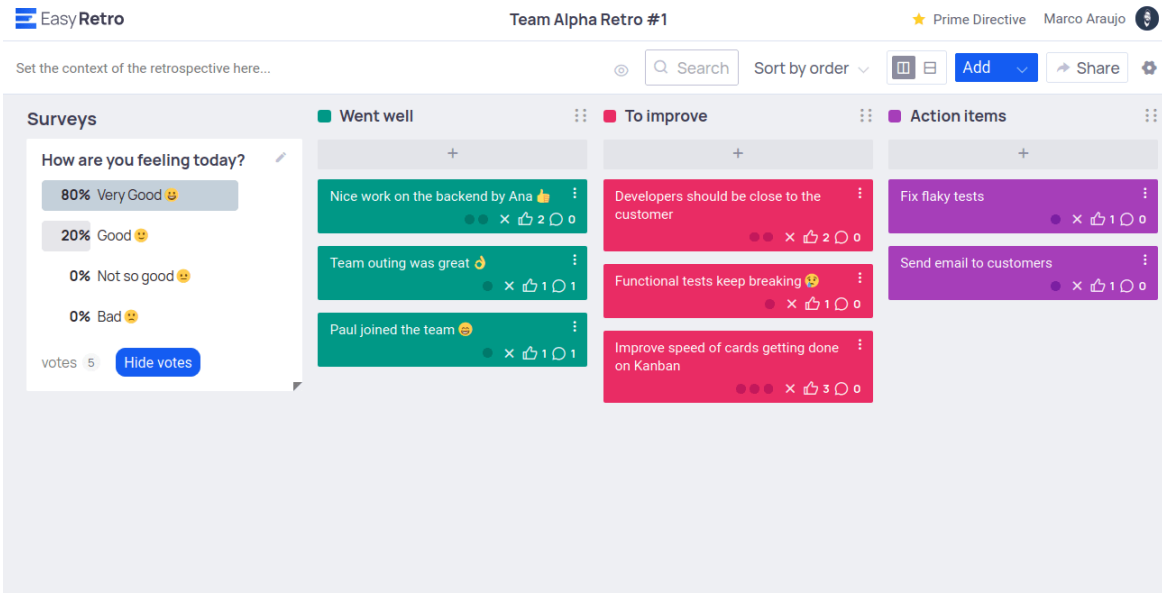


Figura 1 - Exemplo de quadro da plataforma EasyRetro [6]

Na criação dos quadros é possível escolher entre vários t*emplates* e selecionar um conjunto de configurações iniciais, tais como a possibilidade de esconder as próprias cartas e/ou os seus autores, limitar os votos, e definir o quadro como público ou privado. Adicionalmente, os utilizadores podem sugerir novos modelos para os quadros e partilhar com a comunidade. Os utilizadores podem personalizar os quadros definindo o número de colunas e os seus títulos, adicionar cartões e a estes adicionar comentários ou votos. Os utilizadores podem ordenar os cartões em cada coluna por votos com o intuito de conseguirem visualizar com clareza os cartões mais importantes, considerados pelos participantes.

A interação em tempo real é uma funcionalidade essencial neste tipo de aplicações e como tal, nesta plataforma, as alterações efetuadas num quadro, por um utilizador, são propagadas para todos os que o visualizam.

Uma vez que os quadros são do tipo *kanban,* é essencial que exista a possibilidade de mover os cartões e as colunas de posição e, como tal a funcionalidade *drag and drop* está presente na aplicação.

Nesta plataforma não é possível atribuir funções específicas aos utilizadores dentro de um quadro ou de uma equipa.

No que diz respeito à análise de dados é disponibilizada uma tabela resumo onde é possível verificar, por quadro, o número de cartões, votos, contribuidores e votantes. Em

relação a integrações, a plataforma permite integrar com outras aplicações como o slack[2], confluence[3], jira[4] e trello[5]. Não se verifica a possibilidade de agendamentos.

Um conjunto de ferramentas adicionais são proporcionadas aos utilizadores tais como um cronómetro possível de usar durante a reunião de retrospetiva, algumas questões de quebra de gelo e imagens com frases motivadoras. Com o intuito de contribuir para a educação dos utilizadores são facultadas algumas páginas de conteúdo informativo tais como um blog onde se verificam artigos que abordam temas relacionados com metodologias ágeis, questões colocadas em entrevistas, resolução de problemas técnicos, etc.

No que diz respeito a questões de segurança é indicado que existe um conjunto de acordos de confidencialidade que as entidades contratadas assinam e têm de respeitar. A gestão dos servidores é da responsabilidade da google *cloud* uma vez que o *backend* está baseado no firebase[6] e consumido como um serviço. Adicionalmente é indicado que as melhores práticas para manter a qualidade do código são praticadas, nomeadamente os testes unitários, de integração e *end-to-end* (E2E). Os testes de penetração são conduzidos de forma independente por uma agência externa uma vez por ano. A informação recolhida é encriptada. Com o intuito de proteger os dados face a perdas são efetuados *backups* através dos serviços da google. Os serviços *firebase* como são geridos pela google têm a responsabilidade de evitar ataques e assegurar alta disponibilidade tendo por isso um *downtime* irrisório. Finalmente é dito que o *backend* é protegido de abusos através de uma *app-check*[7] do *firebase* que deteta pedidos inválidos ou intrusos e utiliza a tecnologia reCAPTCHA v3.

A plataforma apresenta limitações no que diz respeito à utilização da mesma, isto significa que existem diversos tipos de planos, entre os quais um plano grátis que apenas permite a criação de três quadros públicos por mês; não permite a criação de equipas e evidentemente quadros associados a estas, e por fim não inclui a funcionalidade de análise de dados. Até à data de análise, a plataforma suporta 3 planos pagos [4],como se pode observar pela Tabela 1, sendo que cada um se diferencia pelo número de quadros que cada utilizador ou equipa pode criar. Apenas os planos pagos suportam a criação de equipas e

[2] https://slack.com/
[3] https://www.atlassian.com/software/confluence
[4] https://www.atlassian.com/software/jira
[5] https://trello.com/
[6] http://firebase.google.com/
[7] https://firebase.google.com/docs/app-check

cada um destes tem um número máximo de equipas que podem ser criadas. Todos os planos permitem a análise de dados e a autenticação por SSO.

Tabela 1 - Planos da plataforma EasyRetro [4]

Plano / Características	Grátis	Equipa	*Business*	*Large business*
Valor	0 por mês	27€ por mês	61€ por mês	95€ por mês
Quadros públicos por mês	3	5	15	30
Número de equipas	0	1	3	6
Inquéritos por quadro	1	ilimitado	ilimitado	ilimitado
Quadros ilimitados por equipa	Não	Sim	Sim	Sim
Membros ilimitados por equipa	Não	Sim	Sim	Sim

2.1.2. Metroretro

A aplicação MetroRetro [7] é uma plataforma que permite efetuar retrospetivas em quadros infinitos, isto é, uma área que não tem limitações onde pode ser inserido qualquer tipo de conteúdo através de *drag and drop*. O objetivo da plataforma é aproximar a realidade do digital, por isso, é verificada a interação em tempo real nos quadros sendo também possível observar o cursor do rato de cada utilizador. Esta plataforma é fundada por dois *developers* cujo objetivo foi tornar as atividades de colaboração remotas mais divertidas e fáceis como na "vida real". É dito que este projeto surgiu também para otimizar retrospetivas no projeto em que os autores estavam inseridos, de tal forma que atualmente é usado como uma ferramenta para organizar todo o trabalho diário.

A flexibilidade é um dos princípios chave deste projeto, por isso não se restringe ao tradicional quadro *kanban* e assim possibilita a execução de retrospetivas com múltiplas estruturas ao mesmo tempo. Existem diversos modelos de estrutura como se pode observar na Figura 2, no entanto estes podem ser totalmente personalizáveis.

Como existe esta liberdade, é possível criar páginas, desenhar, inserir cartões em diversos formatos, através de *drag and drop*, permite também efetuar zoom em todo o quadro e por isso obter perspetivas diferentes. É possível pré-agendar uma reunião e isto significa planear atividades para depois serem executadas. Não se verifica a possibilidade de

rastrear facilmente os itens que são criados para solucionar o que correu menos bem, uma vez que a ferramenta não é totalmente dedicada a retrospetivas.

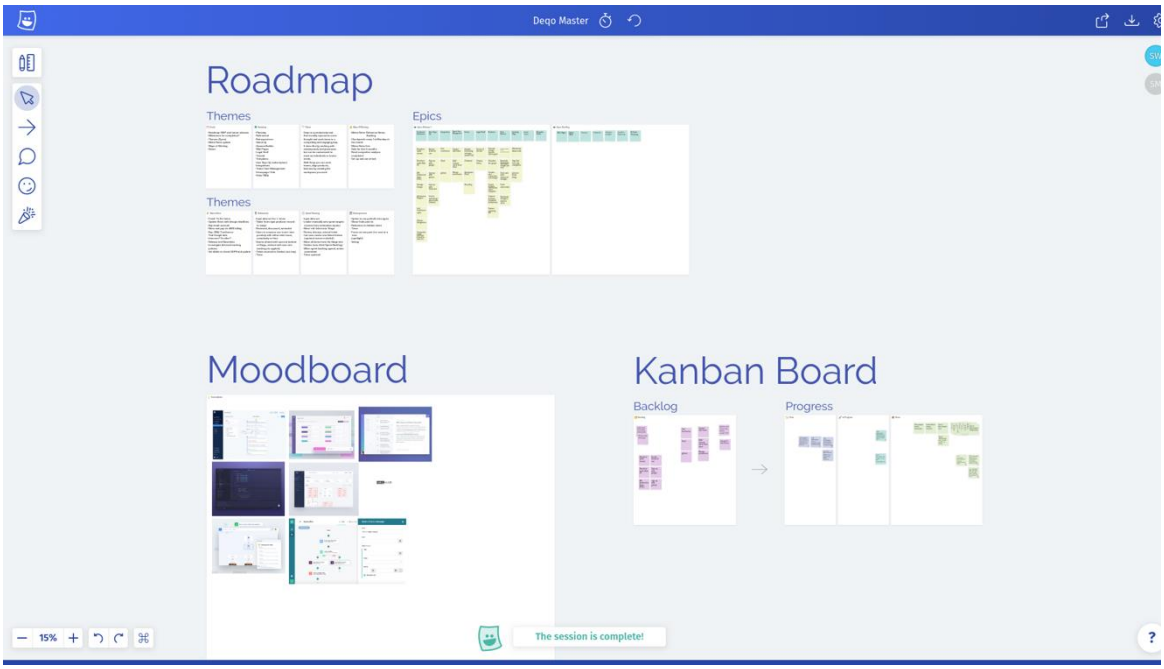


Figura 2 - Exemplo de quadro da plataforma metroretro [7]

É possível criar *workspaces* que representam equipas e podem ser adicionados membros a esta, verificando-se a possibilidade de atribuir administradores adicionais. Como mais-valia cada quadro pode ter um moderador ou organizador e como tal é possível definir o papel de cada membro num quadro com responsabilidades diferentes. Existe a possibilidade de incluir um cronómetro para temporizar eventuais etapas de uma retrospetiva e também é possível votar em cartões ou em grupos destes.

A plataforma apresenta um blog onde são reunidos artigos que permitem contribuir para aumentar a produtividade das equipas utilizando a plataforma, mas também são apresentadas as novidades no que diz respeito a funcionalidades ou correções que surgem ao longo do tempo. Adicionalmente existe uma secção de suporte onde são respondidas as questões mais frequentes, mas também é possível aceder a um canal de conversação em tempo real com a equipa de suporte.

Este projeto também apresenta vários planos de utilização como é possível observar, sendo que no "grátis" verifica-se a limitação no número de objetos inseridos e de utilizadores no mesmo quadro. Não permite criar vários *workspaces* e não é possível autenticar ou registar via SSO. Os planos pagos não apresentam as limitações anteriores. O primeiro denominado de "equipa", destina-se a organizações ou equipas com um elevado número de

membros que necessitam de autenticação por SSO e segurança melhorada mediante o pagamento de 6$ por cada membro em cada mês. O último plano destinado a singulares que pretendem partilha dos quadros com colaboradores externos sem necessidade de registo podem usufruir do mesmo mediante o pagamento de 25$ por mês.

Tabela 2 - Planos da plataforma MetroRetro

Plano / Características	Grátis	Equipa	Host
Valor	0$	$6/ por membro e por mês	$25/
Objetos	Limitados	Ilimitados	Ilimitados
Limitação de convidados	Total	Limitado	Ilimitado
Autenticação SSO	Não	Sim	Sim
Partilha de quadros privados	Não	Sim	Sim
Workspaces	1	Ilimitado	ilimitado

No que diz respeito a segurança é referido que todos os dados são alojados em base de dados localizados na união europeia, é usada encriptação através do padrão de criptografia AES-256 o que é uma boa prática para manter os dados em segurança.

Os *backups* como uma das mais importantes ferramentas de defesa num sistema, são armazenados e encriptados em discos físicos e duram no máximo 28 dias e todas as *passwords* fornecidas pelos utilizadores no registo são guardadas recorrendo a técnicas de *hashing* e *salting*

Os autores do projeto têm o foco na segurança e como tal é indicado que a infraestrutura é alojada na plataforma *digital ocean*[8], em Amesterdão, e toda a informação introduzida na plataforma pertence ao utilizador que criou a sua conta e tem o direito de modificar e apagar toda a informação introduzida. Finalmente, e com o intuito de monitorizar a aplicação, existe uma infraestrutura capaz de alertar os responsáveis da aplicação quando se verificam anomalias.

[8] https://www.digitalocean.com/

2.1.3. Reetro

A plataforma Reetro [8] é uma ferramenta direcionada para a execução de retrospetivas em equipas, de forma colaborativa oferecendo inúmeras funcionalidades e a maioria são disponibilizadas de forma gratuita.

A equipa autora do projeto é constituída por oito profissionais, em três localizações diferentes (Dinamarca, Finlândia e Países Baixos). O grande objetivo foi a criação de uma ferramenta que permita agilizar e otimizar o processo das retrospetivas, de forma a produzir valor, facilitando a vida de um *scrum master*, evitando o desperdício de tempo e dinheiro. É referenciado que o processo de retrospetivas pode ser otimizado se a ferramenta permitir a colaboração em tempo real de forma divertida e fácil, se o processo for totalmente automatizado e também se for possível integrar com outras ferramentas.

Nesta plataforma é permitida a criação de equipas e gerir os seus membros individualmente, bem como atribuir papéis como super administrador, administrador de equipa ou participante. O super administrador é automaticamente atribuído a quem cria a equipa e este pode atribuir o papel de administrador a outros. Normalmente estes são os *scrum master* ou gestores de projeto que gerem as reuniões. Outra das permissões é a possibilidade de adicionar ou remover utilizadores às equipas.

A criação de quadros é uma das funcionalidades principais e é possível selecionar um conjunto de configurações como o *layout*, o nome do quadro, esconder ou mostrar comentários que serão efetuados nos cartões, tornar o quadro público ou privado, criar cartões anónimos ou públicos, definir o número máximo de votos e permitir interações em tempo real. Quando o utilizador se regista é automaticamente criada uma equipa para si, uma vez que um dos requisitos para a criação de quadros é a associação destes a uma equipa. Em cada coluna podem ser adicionados novos cartões e esta pode ser personalizada sendo possível alterar o seu nome ou cor.

Os quadros exemplificados na Figura 3, podem ser clonados, apagados, bloqueados e isto impede a adição de novo conteúdo. Podem também ser exportados para a ferramenta *Jira*. Uma mais valia é a possibilidade de combinar quadros, isto é, passar os cartões de um quadro para o outro de forma automática. Tendo em conta as configurações mencionadas é evidente que é possível criar, editar, remover, personalizar cartões e comentários, votar, mover os cartões entre as colunas. É possível ordenar os cartões por número de votos e autor. Os itens de ação, ou seja, aqueles itens que pretendem resolver algo que tenha corrido menos

bem podem ser associados a determinados utilizadores querendo indicar que devem ser resolvidos por estes. No seguimento da funcionalidade referida existe uma secção onde é possível rastrear os itens de ação de todos os quadros e verificar o estado destes, melhorando assim a capacidade de verificar se o progresso e a resolução dos problemas registados estão a ser tratados.

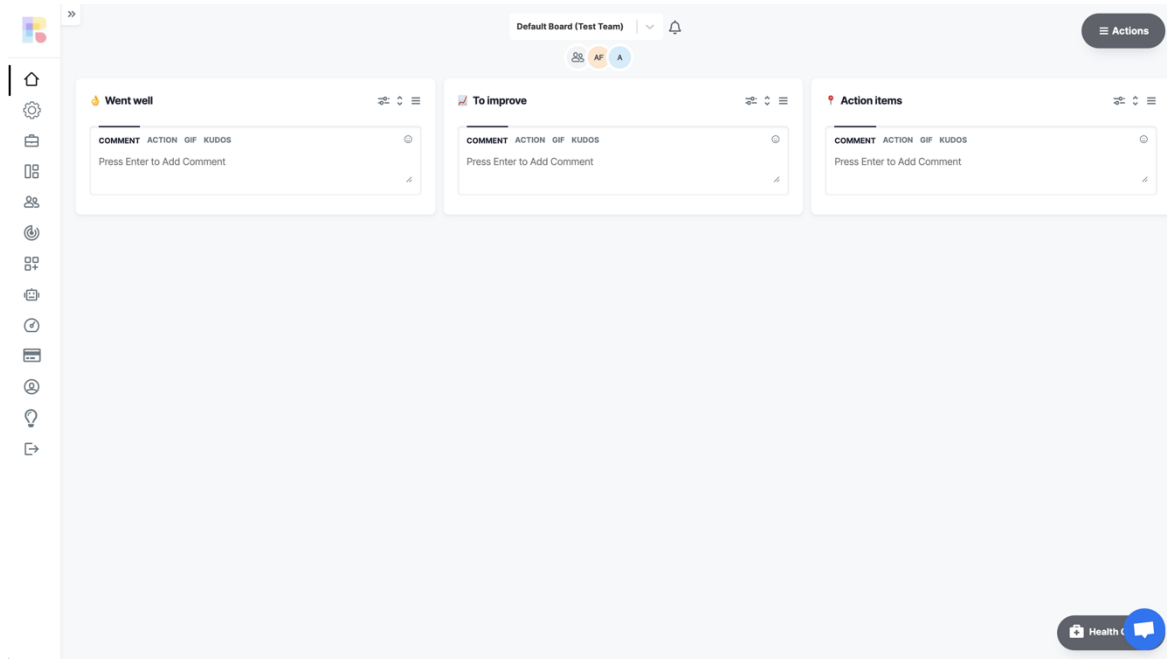


Figura 3 - Exemplo de quadro da plataforma Reetro [8]

Uma funcionalidade relevante passa pela capacidade de automatizar as retrospetivas na medida em que é possível agendar a criação de quadros segundo determinadas configurações.

É apresentada a funcionalidade de análise de resultados e recorrendo a gráficos é possível adquirir uma perspetiva geral de cada retrospetiva, isto é, verificar a média de comentários, de votos positivos e negativos, assim como a média do número itens de ação produzidos. Nesta plataforma não é apresentada nenhuma funcionalidade que visa a otimização de retrospetivas assíncronas com elevado número de elementos e de forma automatizada.

No que diz respeito à segurança, os autores do projeto indicam que a confiança no produto é fulcral e como tal é verificado um elevado nível de segurança seguindo um conjunto de princípios e regras fundamentais. A conformidade com o regulamento geral sobre a proteção de dados (GPDR), isto é: os membros responsáveis pela plataforma têm acesso ao mínimo de informação e apenas àquela necessária para efetuarem o seu trabalho.

A informação dos clientes é transferida através de protocolo HTTPS e a informação não relevante é tornada anónima.

Outra medida adotada, passa por seguir as boas práticas de segurança existentes nos controlos de sistema e organização do tipo dois [9] (SOC-2) que define os critérios de como deve ser gerida a informação do cliente tendo em conta a segurança, disponibilidade, integridade, confidencialidade e privacidade. A segurança é um especto considerado de extrema relevância e por isso reforçada. É referenciado que os servidores da aplicação são alojados na plataforma Heroku[9] e como tal é gerida pela Amazon Web Services[10], que está em continua manutenção das suas máquinas e atenta a alertas que possam surgir. Todos os *endpoints* da aplicação estão protegidos por *user tokens,* mais concretamente através de *JSON web tokens* (JWT). As aplicações são submetidas a testes de penetração e revisões regulares do código, com o intuito de minimizar o risco de intrusões. São garantidos backups regulares e um sistema de gestão de incidentes.

Até à data de análise aqui apresentada, na plataforma são evidenciadas duas versões pagas [8] como se pode observar na Tabela 3, a primeira – Pro – mediante pagamento mensal de 15$ por cada equipa. A segunda - empresa – não apresenta valor definido uma vez que depende do número de utilizadores. Ambas permitem integração com mecanismos de autenticação via SSO, a possibilidade de agendamento de retrospetivas sendo possível definir a recorrência, integração com outras plataformas (*jira* e *azure devops*), melhor suporte, mais informação para analisar, relatórios e quadros personalizados. É necessário subscrever o plano de empresa para garantir autenticação multi-fator, segurança melhorada, gestão de *backups* e suporte personalizado.

Tabela 3 - Planos da plataforma Reetro

Plano / Características	Grátis	Pro	Entreprise
Valor	0$	$15/ por mês e por equipa	Depende do total de utilizadores
Agendamentos	Não	Sim	Sim
Integrações	4	6	6
Autenticação SSO	Não	Sim	Sim
Relatorios	Não	Sim	Sim

[9] https://www.heroku.com/
[10] https://aws.amazon.com/pt/

Segurança melhorada	Não	Sim	Sim

2.1.4. Retrotool.io

A aplicação Retrotool [6] é uma ferramenta grátis que pretende destacar-se pela simplicidade na criação de retrospetivas. A empresa que desenvolveu a aplicação é denominada de u2i, sendo que conta com 60 colaboradores e que resolve problemas em diversos domínios como *backend*, *frontend*, *mobile*, UX *design* e DevOps. Referem que, inicialmente, a ferramenta foi criada como um projeto interno para efetuar as retrospetivas dos seus projetos, no entanto, após verificar o valor da mesma foi notada a necessidade de partilhá-lo com todo o mundo.

A plataforma não apresenta tantas funcionalidades como as anteriores, contudo é indicado que o foco é a sua simplicidade e como tal é possível criar quadros *kanban* ilimitados com três ou mais colunas em apenas três cliques. Sendo apenas necessário escolher entre um *template* ou criar um personalizado. Após escolha, o quadro e o link público associado são imediatamente gerados. Uma das funcionalidades que se destaca nos quadros é o facto de cada uma das colunas ter uma secção individual, privada e outra pública como se pode observar na Figura 4. Esta funcionalidade permite que os utilizadores tomem notas ou preparem conteúdo para posteriormente tornar público.

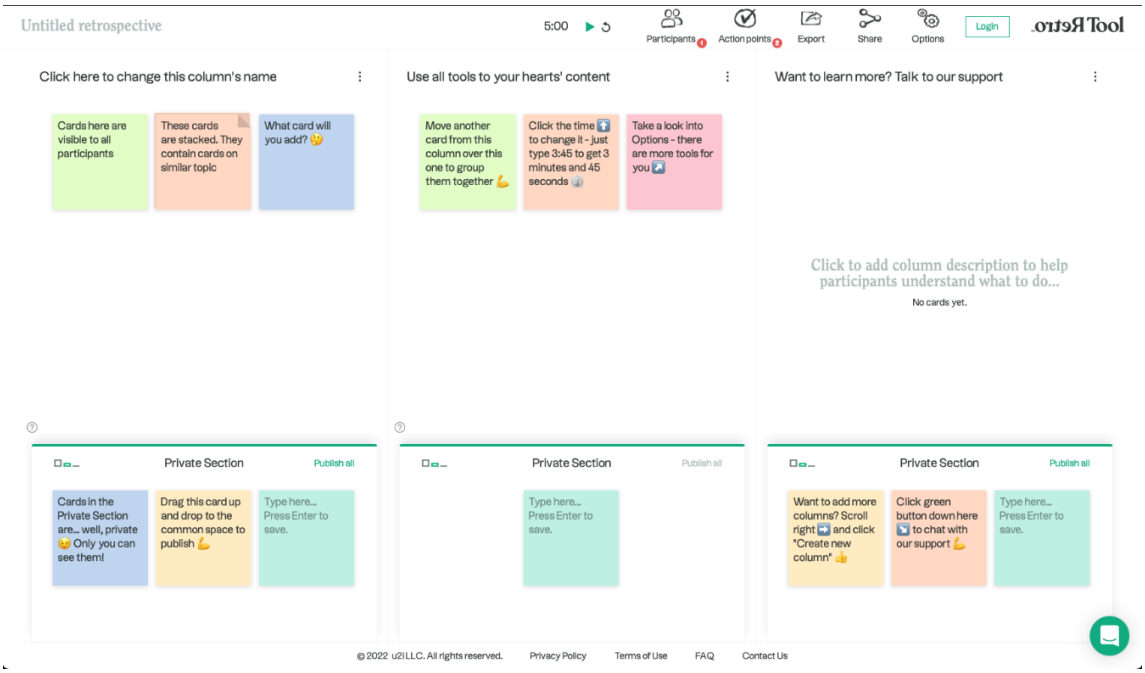


Figura 4 - Exemplo de quadro da plataforma Retrotool [8]

Os cartões adicionados podem ser personalizados, anónimos ou públicos. Tal como em todas as retrospetivas existe a possibilidade de votar, sendo também possível ordenar os cartões nas colunas por votos. É incluído um temporizador para controlar o tempo discutido sobre determinados tópicos. Como mais-valia a plataforma disponibiliza uma funcionalidade de demonstração onde qualquer utilizador pode aceder a uma retrospetiva exemplo e executar as funcionalidades principais.

A plataforma apresenta três planos, como é possível observar na Tabela 4, o anónimo que não requer autenticação e este é indicado para quem pretende iniciar rapidamente uma retrospetiva, não pretende ficar associado ao sistema e apenas deseja aprender como a ferramenta de retrospetivas funciona. No que diz respeito à segurança neste plano, os dados são encriptados seguindo os padrões da indústria.

O plano registado é também gratuito, no entanto, implica que os utilizadores se registem na plataforma. Permite realizar todas as operações que o plano anónimo, os quadros são guardados durante três meses, podem ser personalizados, um determinado utilizador pode assumir o papel de administrador dentro de um quadro e por isso pode habilitar ou desabilitar certas funcionalidades aos participantes com o *role* normal.

O plano *premium* é por isso pago mediante o valor de 20$ por mês. É verificado até à data de análise da plataforma [6], tem como principal funcionalidade extra a encriptação de todos os dados associados. Neste plano é possível criar quadros de retrospetivas privados e convidar pessoas para participar. Todas as retrospetivas são guardadas no *dashboard* por tempo ilimitado. É indicado que estes utilizadores têm prioridade no suporte.

Tabela 4 - Planos da plataforma Retrotool.io

Plano / Características	Anonimo	Registado	Premium
Valor	Grátis	Grátis	$20 por mês
Archivo	Não	Até 3 meses	Ilimitado
Customização	Não	Sim	Sim
Retrospetivas privadas	Não	Não	Sim
Permissões	Não	Sim	Sim
Prioridade no suporte	Não	Não	Sim

2.2. Comparação entre ferramentas

Esta secção tem o propósito de apresentar uma comparação entre as plataformas analisadas no que diz respeito às principais funcionalidades, mas também permitir verificar quais são aqueles essenciais numa aplicação deste âmbito. De relevância superior, a comparação serve para compreender se existe alguma que tem o foco na realização de retrospetivas em grandes equipas. Com base nessa análise apresentar de que forma o projeto apresentado neste documento pode afirmar-se no mercado apresentando funcionalidades que permitam efetuar retrospetivas em equipas ou organizações com elevado número de elementos através da divisão destas em subequipas.

A Tabela 5 pretende resumir a informação das secções anteriores. As linhas dizem respeito à funcionalidade e cada coluna da tabela refere-se a uma das ferramentas analisadas. A última coluna diz respeito à plataforma SPLIT e descrita neste documento com o intuito de realçar as mais valias em comparação com as analisadas.

Como nota prévia, é importante referir que as plataformas analisadas apresentam muitas semelhanças entre si, tanto a nível de funcionalidades como de configurações, uma vez que todas se destinam ao auxílio na execução de retrospetivas de forma virtual e através de um quadro.

Tabela 5 - Análise de Funcionalidades vs Plataformas. Legenda: MVP ou FV - Full version

Plataforma / Func.	EasyRetro	Reetro	Reetro	Retrotool	SPLIT
Tipo de quadro	Kanban	Infinito	Kanban	Kanban	Kanban
Cartões	✅	✅	✅	✅	✅
Agrupar cartões	✅	✅	✅	✅	✅
Comentários	✅	✅	✅	✅	✅
Votação	✅	✅	✅	✅	✅
Ordenação de cartões	✅	❌	✅	✅	✅
Exportação e integrações	✅	❌	✅	❌	✅
Tempo Real	✅	✅	✅	✅	✅
Equipas	✅	✅	✅	❌	✅
Papéis em equipas	✅	✅	✅	❌	✅
Papéis em quadros	✅	✅	✅	✅	✅
Rastrear itens de ação	✅	❌	✅	❌	MVP: ❌ ;FV: ✅
Análise de dados	✅	❌	✅	❌	MVP: ❌ ;FV: ✅
Integração com slack	✅	❌	❌	❌	✅
Agendamentos	❌	❌	✅	❌	✅
Cronómetro	❌	✅	✅	✅	✅
Divisão de equipas	❌	❌	❌	❌	✅
Juntar quadros	❌	❌	✅	❌	✅
Relações entre quadros	❌	❌	❌	❌	✅
Autenticação SSO	✅	✅	✅	✅	✅
Open-source	❌	❌	❌	❌	✅
Personalizar cartões	✅	✅	✅	✅	✅

Personalizar comentários	✓	✓	✓	✓	MVP: ✗ ;FV: ✓
Personalizar colunas	✓	✓	✓	✓	✓
Templates	✓	✓	✓	✓	MVP: ✗ ;FV: ✓
Anonimizar cartões	✓	✓	✓	✓	✓
Mostrar/ocultar cartões	✓	✗	✓	✗	✓
Limitação de votos	✓	✓	✓	✓	✓
Publico/Privado	✓	✗	✓	✓	✓
Totalmente grátis	✗	✗	✗	✗	✓
Partilha	✓	✓	✓	✓	✓

Analisando a tabela é possível constatar que a maioria das ferramentas para a execução de retrospetivas, apresentam um quadro do tipo *kanban*, no entanto, a plataforma MetroRetro dispõe de um quadro infinito e sem *template* definido. É também importante referir, como será visto no próximo capítulo, no que diz respeito ao desenvolvimento, o SPLIT foi dividido em duas grandes fases, o *minimum value product* (MVP) e a versão final. A primeira e algumas funcionalidades da segunda fase são apresentadas neste documento e as restantes são definidas como trabalho futuro. Por isso, na tabela e na coluna que diz respeito ao SPLIT, algumas funcionalidades poder-se-ão encontrar apenas numa das fases. De realçar que o planeamento, investigação e levantamento de requisitos foi efetuada tendo como premissa o produto completo.

As funcionalidades básicas tal como a gestão de cartões, comentários, votos, personalizações, anonimização de cartões, interações em tempo real, autenticação por SSO e partilha dos quadros através de um *link*, são disponíveis em todos as plataformas analisadas. A possibilidade de tornar o quadro público ou privado apenas, não é possível de definir no MetroRetro. Apenas uma das plataformas não conhece o conceito de equipa e todas estabelecem papéis específicos dos utilizadores dentro de um quadro, no entanto estes não são possíveis de alterar em todas as plataformas. De igual forma todas as que permitem

criar equipas também permitem definir papeis específicos para os membros constituintes do grupo.

No que diz respeito à possibilidade de rastrear os itens de ação e análise de dados apenas são disponibilizadas pela plataforma EasyRetro e Reetro.

Sendo o *slack* uma das principais ferramentas de comunicação utilizadas na xgeeks, merece um destaque especial e como tal é verificado que apenas o EasyRetro permite integração com o mesmo e assim enviar mensagens com base nas ações efetuadas na plataforma. De forma inversa é também possível realizar ações na plataforma através do *slack*.

Os agendamentos apenas são disponíveis na ferramenta Reetro. Todas as plataformas permitem integração e autenticação através de SSO.

Por fim, é possível afirmar que nenhuma das plataformas permite facilitar as retrospetivas em equipas com elevado número de elementos, ao verificar-se que não existe nenhuma funcionalidade de divisão de uma equipa e apenas uma das plataformas permite a transferência do conteúdo entre quadros e nenhuma apresenta relações entre eles.

2.3. Motivação

Apresentada a comparação das plataformas em estudo é importante realçar a motivação que promoveu a criação da ferramenta SPLIT e o porquê desta se diferenciar dos projetos existentes podendo afirmar-se como valiosa para outras empresas que verifiquem problemas semelhantes.

Assim, este projeto destaca-se pelas funcionalidades que permitem efetuar retrospetivas em equipas de larga escala otimizando o processo executado na xgeeks. Tendo em conta este pressuposto, a ferramenta permite dividir uma equipa grande em subequipas e automaticamente criar quadros, o principal e outro para cada uma das equipas. É também atribuído o papel de responsável a um membro participante e aleatório das subequipas com o intuito deste ser o moderador da retrospetiva e aquele que tem a função de dar por terminada a sessão e, por isso tem a permissão de passar o conteúdo do seu quadro para o principal. Aliado a esta funcionalidade existe uma relação de parentalidade entre subquadros e o principal, o que permite a transferência automática do conteúdo de um para o outro após o término da retrospetiva da subequipa. As operações relevantes tais como: divisão dos

membros em subequipas, nomeação ou alteração de responsável, ação de finalização de cada subretrospetiva, pode gerar uma mensagem no *slack* com o intuito de notificar os utilizadores da empresa. Finalmente, de enorme importância é a capacidade de efetuar agendamentos mensais que despoleta todo o processo de retrospetiva mencionado anteriormente. Também se pretende apresentar dados que permitam observar o desempenho das equipas em cada retrospetiva, mas também disponibilizar uma forma de rastrear o estado das ações que são definidas para melhorar os aspetos que correram menos bem.

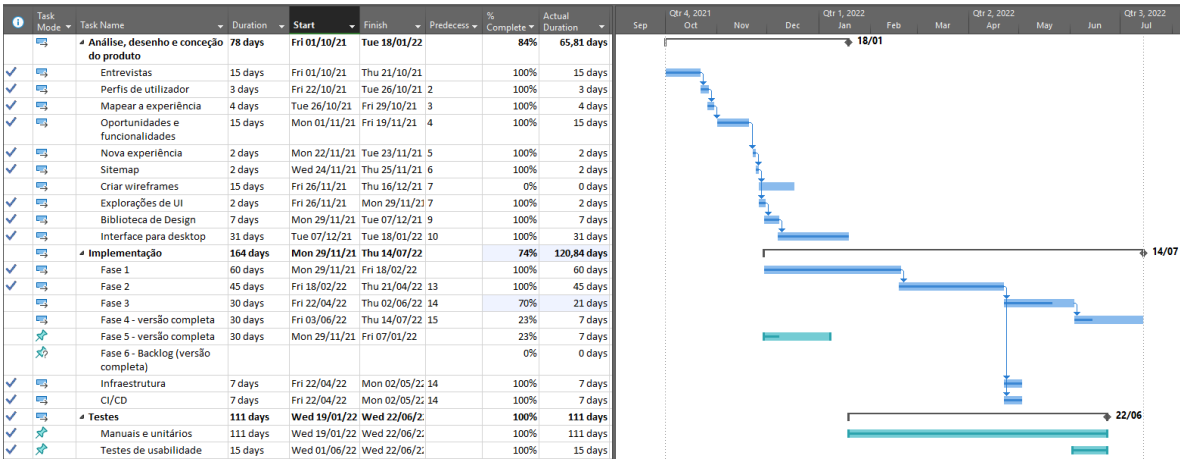
Com a realização deste projeto pretende-se apostar nos projetos *open-source* assim, o código fonte é acessível por qualquer empresa ou indivíduo que não pretende monetizar a aplicação ao abrigo da licença MIT, mas que o queira utilizar e alojar como bem entender.

3. Planeamento do processo de desenvolvimento

O planeamento e gestão apropriados são dois fatores chave para o sucesso no desenvolvimento do projeto. Neste capítulo, em específico na secção 3.1 é apresentado a representação temporal do plano do projeto. Na secção 3.2, são expostos o plano e a estratégia adotada para a definição do design e conceção do produto, pesquisa da melhor experiência e necessidades do utilizador. Na secção 3.3 é apresentado o planeamento do projeto no que diz respeito à sua implementação e na secção 3.4 o plano de testes.

3.1. Plano do Projeto

Nesta secção é apresentado o plano de alto nível do projeto, separado por três fases totalmente flexíveis e cada uma constituída por tarefas. O planeamento das fases é apresentado com maior detalhe nas próximas secções. Como se pode observar Figura 5, a primeira fase diz respeito à de pesquisa e conceção do produto que teve a duração de 78 dias, sendo que foi possível concluir 84% do planeado. A tarefa de criação de *wireframes* não foi possível realizar devido à falta de tempo, mas também porque foi considerado que não teria um valor justificável. A fase de implementação, com duração estimada de 164 dias, foi executada com 74% de sucesso. As funcionalidades definidas como principais e por isso críticas para colocar em produção (MVP) foram realizadas na fase 1, 2 e 3. Na fase três apenas as tarefas relacionadas a integração avançada do *slack* é que não foram efetuadas.



Task Mode	Task Name	Duration	Start	Finish	Predecess	% Complete	Actual Duration
	Análise, desenho e conceção do produto	**78 days**	**Fri 01/10/21**	**Tue 18/01/22**		**84%**	**65,81 days**
✓	Entrevistas	15 days	Fri 01/10/21	Thu 21/10/21		100%	15 days
✓	Perfis de utilizador	3 days	Fri 22/10/21	Tue 26/10/21	2	100%	3 days
✓	Mapear a experiência	4 days	Tue 26/10/21	Fri 29/10/21	3	100%	4 days
✓	Oportunidades e funcionalidades	15 days	Mon 01/11/21	Fri 19/11/21	4	100%	15 days
✓	Nova experiência	2 days	Mon 22/11/21	Tue 23/11/21	5	100%	2 days
✓	Sitemap	2 days	Wed 24/11/21	Thu 25/11/21	6	100%	2 days
	Criar wireframes	15 days	Fri 26/11/21	Thu 16/12/21	7	0%	0 days
✓	Explorações de UI	2 days	Fri 26/11/21	Mon 29/11/21	7	100%	2 days
✓	Biblioteca de Design	7 days	Mon 29/11/21	Tue 07/12/21	9	100%	7 days
✓	Interface para desktop	31 days	Tue 07/12/21	Tue 18/01/22	10	100%	31 days
	Implementação	**164 days**	**Mon 29/11/21**	**Thu 14/07/22**		**74%**	**120,84 days**
✓	Fase 1	60 days	Mon 29/11/21	Fri 18/02/22		100%	60 days
✓	Fase 2	45 days	Fri 18/02/22	Thu 21/04/22	13	100%	45 days
	Fase 3	30 days	Fri 22/04/22	Thu 02/06/22	14	70%	21 days
	Fase 4 - versão completa	30 days	Fri 03/06/22	Thu 14/07/22	15	23%	7 days
	Fase 5 - versão completa	30 days	Mon 29/11/21	Fri 07/01/22		23%	7 days
	Fase 6 - Backlog (versão completa)					0%	0 days
✓	Infraestrutura	7 days	Fri 22/04/22	Mon 02/05/22	14	100%	7 days
✓	CI/CD	7 days	Fri 22/04/22	Mon 02/05/22	14	100%	7 days
✓	**Testes**	**111 days**	**Wed 19/01/22**	**Wed 22/06/22**		**100%**	**111 days**
✓	Manuais e unitários	111 days	Wed 19/01/22	Wed 22/06/22		100%	111 days
✓	Testes de usabilidade	15 days	Wed 01/06/22	Wed 22/06/22		100%	15 days

Figura 5 - Plano do projeto

As fases 4, 5 e 6 são opcionais para este projeto uma vez que dizem respeito à versão final com funcionalidades menos relevantes para a primeira versão com valor do produto (MVP). Na fase 4 e 5 alcançou-se 23% do progresso total de cada uma. A fase de testes

decorreu durante o desenvolvimento do projeto e no final do mesmo. Ao longo do desenvolvimento foram efetuados testes unitários e manuais. Os testes de usabilidade levaram 15 dias a executar e analisar.

3.2. Fase de pesquisa e conceção do produto

Nesta secção é apresentado o plano elaborado para o desenho, definição e conceção do produto que implica a pesquisa, exploração, levantamento de requisitos, definição de funcionalidades, desenho da aplicação e busca da melhor experiência de utilização.

Tendo como foco a otimização do processo de retrospetivas realizadas na empresa, a equipa inicial constituída por três elementos cujos papéis são: *designer*, *developer* (autor deste projeto) e o *engineer manager* (EM), participou na definição e na execução do plano composto por onze pontos distintos, descritos de seguida e detalhados no próximo capítulo.

Passo 1 -Entrevistas e questionários

A primeira fase do plano pretende efetuar entrevistas a elementos voluntários das várias empresas do grupo cuja xgeeks é parte integrante e que participam regularmente em retrospetivas. As entrevistas têm como objetivo obter informação sobre as necessidades dos utilizadores, como estes efetuam as suas retrospetivas e os problemas que enfrentam durante a sua execução. Baseado nessas entrevistas e questionários é possível idealizar oportunidades e posteriormente funcionalidades para a ferramenta.

Tempo estimado: 1 a 2 semanas para entrevistar 5 - 10 pessoas incluindo a análise das mesmas.

2. Criação de perfis de utilizador

A partir das observações e entrevistas é importante perceber e categorizar por perfis os entrevistados. Ex: *scrum masters*, *developers*, *product owner*, etc.

Tempo estimado: 1-2 dias.

3. Mapear a experiência de cada utilizador

Mapeamento do processo dos utilizadores na participação em retrospetivas com a sua ferramenta de eleição para entender quais os possíveis pontos críticos, dificuldades e necessidade em cada etapa do processo.

Tempo estimado: 1-3 dias para mapear e definir os pontos críticos e necessidades para cada fase.

4. Procura de oportunidades e geração de funcionalidades

Através da análise de entrevistas e do mapeamento realizado no ponto anterior deve ser efetuado um mapa de oportunidades.

A partir deste, organizar um *workshop* com voluntários que participam em retrospetivas. Baseado nas oportunidades geradas no *workshop* são produzidas ideias e funcionalidades. Neste devem ser utilizadas questões "como nós poderíamos...?" para facilitar os participantes a pensar em funcionalidades e formas de superar os desafios.

Tempo estimado:

- 1-2 dias para oportunidades
- 1-2 dias para preparação do *workshop*
- 1 dia para o *workshop*
- 1-3 dias para análise de resultados

5. Nova experiência

Perceber de que forma a experiência com a ferramenta de retrospetivas pode ser melhorada através das sugestões referidas no ponto anterior. Para tal pode ser criado um exemplo melhorado do fluxo de utilização da ferramenta.

Tempo estimado: 1-2 dias

6. Sitemap

Criar um simples *sitemap* onde são apresentadas as funcionalidades que poderão ser apresentadas em cada página.

Tempo estimado: 1-2 dias.

7. Criar wireframes

Os *wifreframes* devem cobrir todo os *flows* da aplicação de forma a conseguir visualizar a aplicação como um todo, permitindo verificar se as funcionalidades idealizadas se enquadram corretamente no objetivo da ferramenta.

Tempo estimado: Depende dos *flows* e funcionalidades decididas para implementação.

8. Explorações de UI

Efetuar explorações a fim de encontrar uma interface moderna e com estrutura para suportar a ferramenta a desenvolver com as funcionalidades pretendidas. A partir desta é possível formar uma biblioteca de componentes.

Tempo estimado: 1-2 dias

9. Biblioteca de design

Definição da biblioteca de *design* como fonte da verdade para *designers* e *developers*, ou seja, implementação da biblioteca de componentes, cores e tipografia a serem utilizados na implementação da aplicação.

Tempo estimado: 2-4 dias

10. Criação da interface para desktop

Criação da interface a ser implementada na aplicação.

Tempo estimado: sem estimativa - depende das funcionalidades.

3.3. Fase de implementação

A presente secção tem como propósito apresentar o plano de implementação definido inicialmente como ponto de partida e com a consciência de possíveis alterações. Primeiramente foi definido que a implementação do projeto deve ter dois grandes marcos: o primeiro que diz respeito ao desenvolvimento do MVP e o segundo que diz respeito ao produto completo. Assim, um conjunto de funcionalidades essenciais são contempladas no primeiro marco. Todas as funcionalidades enquadradas nas respetivas fases são descritas no próximo capítulo, onde é apresentado o resultado da execução do processo de pesquisa e conceção do produto. O resultado de pesquisa incide sob o produto completo, contudo foi delineado que durante o trabalho de projeto na empresa o foco deveria ser o desenvolvimento da fase MVP e se possível, funcionalidades do produto completo.

Paralelamente à implementação das funcionalidades do projeto é importante construir um *pipeline* de *continuous integration* e *continuous delivery* e a respetiva infraestrutura. Isto é, criar um processo que automatize a integração e teste as funcionalidades desenvolvidas

com a disponibilização das aplicações com as novas funcionalidades, numa determinada infraestrutura na *cloud*.

3.4. Fase de testes

Nesta secção é apresentado o planeamento desenvolvido com o intuito de compreender o sucesso da implementação do produto apresentado neste documento, mas também assegurar a qualidade do código. Assim, definiram-se objetivos para compreender o nível de satisfação dos utilizadores e elaborou-se um plano de testes.

Neste plano pretende-se identificar os problemas de usabilidade, compreender se a solução desenvolvida corresponde ao modelo mental dos utilizadores alvo, e compreender se a solução desenvolvida satisfaz as necessidades do utilizador. Tendo em vista os objetivos, devem ser promovidos dois momentos de avaliação, o primeiro diz respeito à execução de testes manuais e unitários ao longo do desenvolvimento do projeto, e o segundo deve passar por convidar alguns membros da empresa a seguir um guia orientado, de forma a executar um conjunto de tarefas na aplicação, e posteriormente responder a questões de usabilidade. Pretende-se concluir a última fase ao fim de duas semanas.

4. Metodologia de desenvolvimento

Neste capítulo é apresentada uma breve análise da evolução dos processos de desenvolvimento de software e é descrita a metodologia utilizada ao longo da elaboração do projeto apresentado neste documento. Neste capítulo é documentado o enquadramento teórico de alguns conceitos relacionados com desenvolvimento *agile*. Assim, na secção 4.1 é apresentado o conceito de *lean thinking*. Na secção 4.2 é efetuada uma breve referência ao conceito *agile* e na secção 4.3 é apresentado um enquadramento teórico referente à metodologia *kanban*. Na secção 4.4 é exposta de que forma a metodologia *kanban* foi aplicada no projeto ao longo do decorrer do desenvolvimento. Aliado a este tópico são apresentadas as ferramentas essenciais e utilizadas na gestão do projeto que permitiram a implementação da metodologia.

Sem dúvida alguma que o processo de desenvolvimento influência a qualidade do software produzido, uma vez que, uma boa gestão aumenta a probabilidade de o produto apresente valor e que seja finalizado com sucesso e como tal é um ponto chave, muito estudado na literatura e em constante evolução. Assim, enormes esforços foram tomados para melhorar constantemente o processo de desenvolvimento de software. Com auxílio da linha temporal apresentada na Figura 6 é possível verificar que os processos de desenvolvimento de software evoluíram gradualmente ao longo do tempo e é possível categorizá-los em dois grupos: o desenvolvimento tradicional e o desenvolvimento ágil.

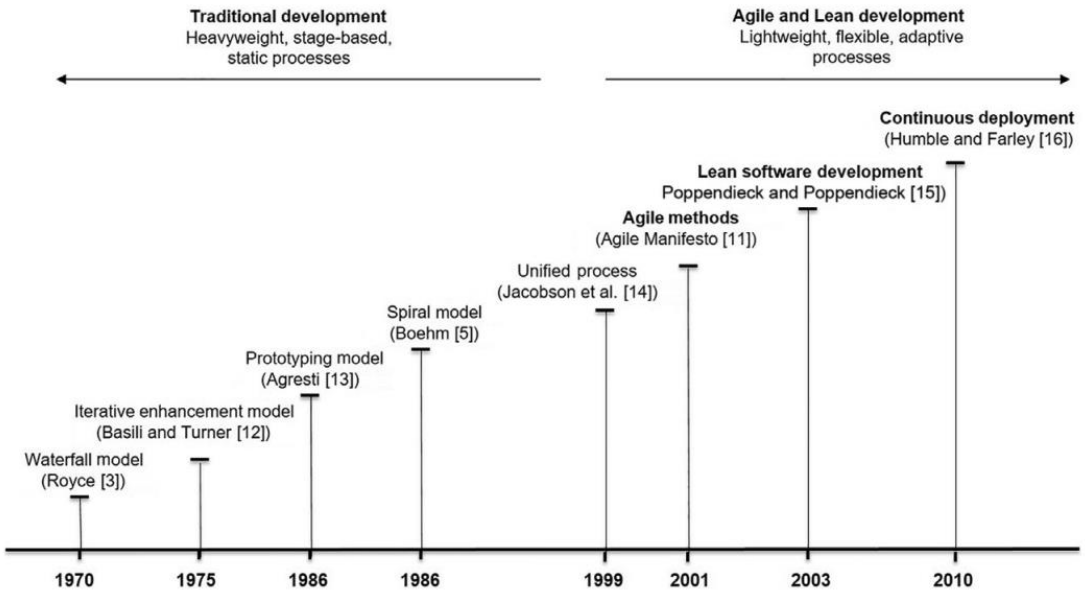


Figura 6 - Contributo para o desenvolvimento de software ao longo dos anos [10]

O primeiro - tradicional, diz respeito a processos que caíram em desuso uma vez que estes são mais rígidos, determinísticos, pesados e baseados em fases, assumindo que os requisitos são estáveis e o seu desenvolvimento é dividido em fases sequenciais. Com a necessidade de tornar o processo de desenvolvimento mais ágil e capaz de suportar alterações nos requisitos surgiu a necessidade de adotar métodos que permitam adaptar o desenvolvimento face a novas mudanças. Assim, surgiu o segundo grupo de processos mais ágeis, flexíveis e adaptativos.

Para melhor compreensão das metodologias ágeis é importante compreender como surgiram, o conceito, princípios e valores subjacentes. Nas próximas secções é efetuado um enquadramento teórico de conceitos ágeis e a metodologia aquela utilizada ao longo do desenvolvimento do projeto apresentado neste documento.

4.1.Lean thinking

Segundo os autores do artigo "*Advances in Using Agile and Lean Processes for Software Development*" [10], a forma de pensar *Lean* (*Lean thinking*), surgiu após a segunda guerra mundial, desenvolvida por Japoneses como um método de produção de manufatura cuja principal ideia é "fazer com pouco, produzindo idealmente as coisas certas, no momento certo, no lugar correto". Por outras palavras, a filosofia subjacente pretende maximizar o valor com o menor desperdício. Esta forma de pensar é assente em cinco princípios: identificar o valor, fluxo de valor, fluxo contínuo, *pull* e melhoria contínua.

Identificar o valor significa encontrar o problema que o cliente precisa de resolver e fazer do produto a solução. Especificamente, o produto deve ser a parte da solução pela qual o cliente pagará prontamente. Dessa forma, qualquer processo ou atividade que não agregue valor – ou seja, não apresente utilidade ou importância – ao produto final, é considerado desperdício e deve ser eliminado.

Uma vez definido o valor (objetivo final), o próximo passo é mapear o "fluxo de valor", ou seja, todas as etapas e processos envolvidos na produção do produto até à entrega do mesmo ao cliente. O objetivo deste princípio é, mais uma vez, identificar quais os passos que não criam valor e encontrar forma de eliminar esses momentos de desperdício.

Uma vez eliminado o desperdício, o princípio seguinte passa por assegurar que as etapas restantes funcionam sem interrupções. Para isto é importante dividir o processo de trabalho

em menores partes e também que seja possível visualizar o fluxo de trabalho por todos as partes integrantes no processo. Uma das técnicas que permite a execução deste princípio é o *kanban,* uma vez que facilita a visualização do que está a decorrer num determinado processo, facilitando a comunicação entre as equipas, e contribuindo para a sincronização relativamente ao que precisa de ser feito e quando.

O princípio *pull* no desenvolvimento de um sistema garante que o fluxo de trabalho seja contínuo, permanece estável e garante que as equipas consigam entregar o trabalho rapidamente e com menor esforço. Esta técnica garante que só se deve efetuar uma nova tarefa quando realmente for necessário.

A melhoria contínua ou perfeição é o princípio mais importante e diz respeito ao aperfeiçoamento do que é aprendido em cada ciclo, tendo sempre como base evitar desperdício.

Estes princípios têm como objetivo contribuir para melhorar a forma de pensar e o processo de desenvolvimento de um produto, em busca constante pelo seu aperfeiçoamento, tendo em vista a entrega de produtos com elevado valor e baixo desperdício.

4.2. Agile

O desenvolvimento de software *agile* refere-se a um grupo de metodologias baseadas em incrementos, onde os requisitos e as soluções evoluem através da colaboração entre equipas bem organizadas, cujos elementos apresentam capacidades distintas e que se complementam. Estas equipas são grupos de pessoas de áreas diferentes que ao trabalharem em grupo garantem um conjunto de capacidades que são essenciais para o sucesso de um projeto.

Este modo de trabalhar introduz um conceito de MVP e a ideia subjacente é desenvolver um produto incrementalmente que apresente as funcionalidades fulcrais para suportar um determinado negócio e assim promover vantagem competitiva. Para o desenvolvimento de um produto seguindo esta metodologia é importante manter contacto constante com o cliente de forma a compreender as suas necessidades e obter *feedback* das várias iterações de produto entregues.

As metodologias ágeis promovem um processo disciplinado de gestão de projetos assente em análises frequentes e adaptação; uma filosofia de liderança que encoraja a equipa

de trabalho, a organização pessoal e a responsabilidade de cada elemento; um conjunto de boas práticas de engenharia com o intuito de entregar rapidamente alta qualidade de software e uma abordagem de negócio que alinha o desenvolvimento com as necessidades do cliente e objetivos das empresas.

Assim o desenvolvimento *agile* refere-se a qualquer processo que vai ao encontro do *agile* manifesto. Este foi desenvolvido por um grupo de dezassete figuras de liderança na indústria de software, que reflete a sua experiência, as abordagens que funcionam e não funcionam no desenvolvimento de software [11]. Assim, promoveu mudanças na forma como a comunidade de engenharia de software desenvolve os produtos.

Como é descrito no "*Manifesto for Agile Software Development*" [12], assenta em quatro grandes valores:

- Indivíduos e interações mais que processos e ferramentas;
- Software funcional mais do que documentação;
- Colaboração com o cliente mais do que negociações de contratos;
- Responder a mudanças mais que seguir um plano.

Além dos valores chave mencionados, são doze os princípios que este tipo de pensamento impõe:

- Satisfazer os clientes através de entrega antecipada e contínua;
- Aceitar mudanças de requisitos mesmo no final do projeto;
- Entregar valor com frequência;
- As pessoas de áreas diferentes devem trabalhar em conjunto diariamente;
- Construir projetos em torno de pessoas motivadas;
- A melhor forma de comunicação é presencial, cara-a-cara;
- Software em funcionamento é a principal medida de progresso;
- Manter um ritmo de trabalho sustentável;
- Excelência continua aumenta a agilidade;
- Simplicidade é essencial;
- As equipas auto-organizadas geram mais valor;
- A reflexão e ajuste regular da forma de trabalhar aumenta a eficácia.

4.3. Kanban

Segundo os autores do artigo "*The state of the art of agile kanban method: challenges and opportunities*" [13], a palavra japonesa *kanban* que significa quadro visual, surgiu nos anos 50 e foi usada para definir um sistema de produção aplicado pela Toyota. No início do século 21, empresas chave de software rapidamente se aperceberam que esta forma de trabalhar poderia ser aplicada a software e alterar a forma como estes são produzidos e entregues. Apenas em 2007 passou a padronizar-se como uma metodologia aceite pela comunidade. Esta representa uma ferramenta eficaz no suporte para a produção de sistemas, bem como contribuí para promover melhorias no processo, sem sacrificar a produtividade. Assim, o grande objetivo é criar valor para o cliente minimizar o desperdício de tempo, recursos e consequentemente custos, tal como descrito na secção 4.1.

A metodologia *kanban* assenta em vários princípios básicos, descritos de seguida, e quando bem aplicados permitem otimizar o fluxo de trabalho de uma ou mais equipas num projeto.

- A **visualização do fluxo de trabalho** implica ter mecanismos de visualização e interação proporcionam uma visão geral do trabalho em desenvolvimento e assim sincronizar todos os elementos integrantes num projeto, simplificando a colaboração entre pessoas e entre diferentes equipas dos projetos;
- A ideia de **limitar o trabalho em progresso** (WIP) tenta assegurar que não será iniciada ou terminada uma tarefa que não poderá prosseguir para a próxima fase sem bloquear todo o fluxo de trabalho. A limitação da quantidade de trabalho em progresso pode ser associada a uma fase, pessoa ou tipo de trabalho. Por norma é o gestor de projeto que realiza esta tarefa;
- Uma vez que existe a necessidade de controlar e otimizar o fluxo de trabalho a todo o momento, esta ferramenta de trabalho permite efetuar a **medição do tempo de entrega das tarefas**. Assim, é possível medir o tempo de entrega das tarefas e através desta métrica alguns problemas podem ser identificados. Isto é, ao identificar as tarefas que demoram mais tempo é possível delegar a atenção para estas de forma a compreender e resolver o que impede de concluir a tarefa com maior velocidade, mantendo a qualidade e minimizando assim os engarrafamentos;
- As **políticas de processo devem ser explícitas**, isto é, todos os membros devem poder ter acesso e visualizar as regras impostas e podem ser definidas a nível de

tarefas, ao nível do quadro *kanban,* bem como entre fases de desenvolvimento. Um exemplo pode ser a limitação do trabalho em progresso; a obrigatoriedade de funcionalidades passarem e serem bem-sucedidas na execução de testes; ter 90% de cobertura pelos testes, etc.

- O método de *kanban* utiliza **modelos** para reconhecer oportunidades de melhoria. Este foca-se em modelos que levam a gerir o desperdício e controlar o fluxo de trabalho considerando a teoria das restrições [14] .

Além destes princípios, esta metodologia apresenta quatro práticas e estas são: começar com o que existe; desenvolver incrementalmente e aceitar mudanças ao longo da evolução do projeto; respeitar as funções de cada membro, processos e responsabilidades; encorajar a liderança durante o processo de desenvolvimento.

Também é referido no artigo que esta metodologia é diferente de outras porque não exige a definição de novas funções para os membros de uma equipa nem de cerimónias.

4.3.1. **O que é o Kanban board**?

Segundo os autores do artigo referido anteriormente [13], a metodologia *kanban* por norma tem como suporte visual um quadro que permite observar o fluxo de trabalho e monitorizar o progresso do projeto. Este contém as atividades e tarefas do processo de desenvolvimento. Através do quadro os *developers* podem observar as tarefas ainda não inicializadas, aquelas em desenvolvimento e as concluídas. Assim a equipa podem concentrar-se nas tarefas não concluídas e finalizá-las. Assim, os recursos e o tempo desperdiçado podem ser reduzidos na troca de tarefas. O quadro *kanban* é um dos aspetos chaves da metodologia uma vez que o processo de desenvolvimento pode ser rastreado visualmente por qualquer membro do projeto.

O quadro tem dois tipos, o simples e o detalhado. Por norma o gestor de projetos determina o tipo de quadro e o nível de detalhe tendo como base o tamanho do projeto, número de tarefas e número de membros da equipa. O quadro simples normalmente tem três colunas: “por fazer”, “em progresso”, “terminado”. Um quadro mais detalhado pode ter diferentes colunas, tais como "lista de tarefas" - *backlog*, “em desenvolvimento”, “em revisão", “revisto”, “testes”, “*deployment”* ou, “terminado”.

De acordo com o artigo existe uma grande tendência na adoção de um quadro *kanban* seja de forma isolada bem como em paralelo com outras metodologias, como *scrum,* que

utiliza um quadro como suporte para providenciar comunicação transparente entre todos os membros do projeto. Esta metodologia facilita o processo de desenvolvimento de um projeto de software uma vez que auxilia no acompanhamento do fluxo de trabalho.

4.4. Aplicação da metodologia no projeto

Na xgeeks, duas das mais conhecias metodologias que seguem os princípios *agile* são aplicados na maioria dos projetos, e são *scrum* e *kanban*. Estas também podem ser utilizadas em conjunto, mais conhecido como *scrumban*. A escolha da metodologia para cada projeto depende de diversos fatores tal como o cliente, o número de pessoas envolvidas, as equipas necessárias e o tipo de projeto. Aliado a este facto deve ser compreendido se existem condições para adotar uma determinada metodologia.

O projeto descrito neste documento é *open-source* e por isso foi criado um repositório no *github* onde qualquer contribuidor pode colaborar no desenvolvimento de funcionalidades ou oferecer suporte ao rever *pull requests*, testar, etc.

Os responsáveis do projeto são os membros da empresa que pretendem participar e, por isso, foram criadas equipas para cada *stack* (*frontend, backend, infrestrutura*) no *github*, como será detalhado nas próximas secções. Estas equipas são constituídas pelos membros da empresa, especializados em cada uma das áreas.

Assim, as equipas ao reconhecer a possibilidade de surgirem alterações e a qualquer momento, tanto dos membros integrantes do projeto, como de funcionalidades, foi definido que a metodologia *kanban* seria aquela a adotar porque melhor se adequa às características do projeto. É importante realçar que durante cerca de 5 meses, a equipa ativa do projeto foi constituída pelo autor deste documento e único *developer*, pelo *engineer manager* e pela *designer*, no entanto definiu-se que a qualquer momento poderiam ser adicionados novos elementos ao projeto.

O *github* proporciona um conjunto de funcionalidades que serve de ferramenta para a aplicação da metodologia, tal como um quadro que permite a qualquer contribuidor ou membro de uma equipa visualizar o fluxo de trabalho. Assim, aliado ao facto de ser *open-source,* não foram verificados pontos chave que contradigam a adoção desta abordagem.

Esta metodologia enquadra-se perfeitamente na conceção do SPLIT, uma vez que permite integração e entrega constante de incrementos, sem necessidade de definir rígidas

responsabilidades entre os membros da equipa de desenvolvimento, permitindo suportar mudanças que possam ocorrer em qualquer altura desde o início até ao fim do mesmo.

Nas próximas secções é apresentado detalhadamente de que forma o *github* foi útil para suportar a componente *open-source*, como foi utilizado para a aplicação da metodologia e como permitiu efetuar a gestão do projeto. Na última secção é apresentado o balanço da aplicação da metodologia, isto é, verificar de que forma os princípios básicos e as práticas comuns descritos no enquadramento teórico foram aplicados.

4.4.1. Github – open-source

Assumindo que o projeto depende de pelo menos três áreas, no *github* e exclusivamente para os membros responsáveis do projeto, foram criadas três equipas na página da organização da empresa. A equipa de *backend*, *frontend* e infraestrutura. A secção de discussões em cada equipa tem o propósito de registar, discutir e tomar decisões relativas a cada área. Os membros de cada equipa constituem o grupo de pessoas que pode efetuar revisões ao pull *requests* (PR).

Sendo um projeto *open-source* qualquer colaborador da empresa pode ser parte integrante de uma ou mais equipas. Com o intuito de facilitar e automatizar a integração nestas, na aplicação *slack* – principal ferramenta de comunicação da empresa, foi adicionado um atalho no canal do projeto e através do qual pode ser solicitado a adesão a uma ou mais equipas como é possível observar na próxima Figura 7.

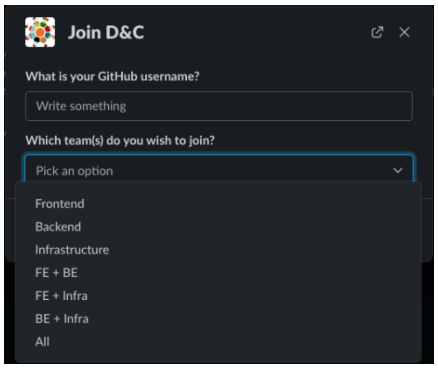


Figura 7 - Formulário de adesão às equipas da organização

Na figura apresentada anteriormente o colaborador da empresa pode utilizar o *bot* de forma a criar uma solicitação preenchendo o *username* do *github* e em que equipas pretende participar.

Primeiramente foi criado um repositório público, acessível a partir do seguinte url: https://github.com/xgeekshq/split e que segue o *template* da empresa para projetos *open-source*. Este apresenta a página inicial - *readme* - como é possível observar na Figura 8, na qual se verifica uma tabela de conteúdos com hiperligações que permitem aceder às diversas secções presentes na página: o código de conduta; um guia de como contribuir no projeto; a licença e a lista de contribuidores.

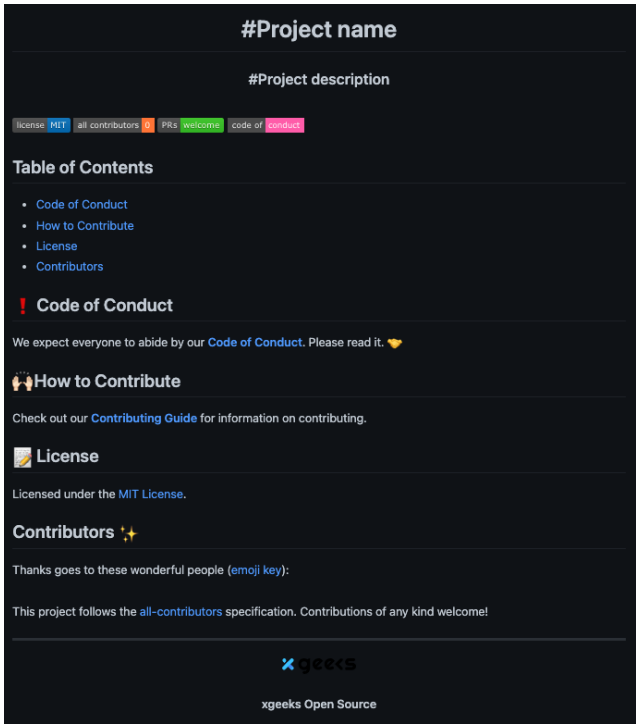


Figura 8 - ReadMe *template* de projetos open-source da xgeeks

No código de conduta, adaptado do pacto oficial de contribuidores *open-source*[11], é referenciado o juramento de garantir que os contribuidores e comunidade do projeto - mantenham uma boa conduta, não sendo permitido qualquer tipo de violência independentemente da raça, etnia, sexo, idade, etc. Neste ficheiro também é indicado o tipo de comportamento que é requerido para contribuir na criação de um ambiente positivo e aquele que deve ser evitado. Neste projeto os responsáveis assumem o compromisso de clarificar o comportamento aceitável e agir apropriadamente consoante cada uma das situações. É da responsabilidade destes remover, editar ou rejeitar *commits*, código, *wiki edits*, *issues*, ou outras contribuições que não vão ao encontro do código de conduta. É possível banir temporária ou permanentemente qualquer contribuidor que tenha comportamentos desapropriados.

[11] https://www.contributor-covenant.org/version/1/4/code-of-conduct

No que diz respeito ao guia de como contribuir é apresentado um ficheiro que indica como efetuar *pull requests*, como reportar um bug e como sugerir uma nova *feature* ou melhoramento. É indicado que o processo de revisão de código é efetuado pelos elementos principais da equipa responsável do projeto. O código e mensagens de *commits* efetuados devem seguir as guias orientadoras convencionais. Finalmente, é indicado que a abertura de *pull requests* deve seguir o *template* fornecido e o contribuidor deve associá-lo ao respetivo *issue* de forma a facilitar o trabalho dos revisores.

O projeto encontra-se sob licença do MIT e a informação referente foi retirada da página *open source initiavite*[12]. Esta permite que os utilizadores possam:

- Usar o código comercialmente;
- Efetuar modificações ao código original;
- Distribuir cópias ou modificações do código;
- Sub-licenciar o código, isto é, distribuir comercialmente versões modificadas;
- Os contribuidores ganham direitos de patente sobre o código.

Para terminar a descrição do modelo utilizado no repositório do projeto, este, também menciona em formato tabela todos os contribuidores. O processo de adição de novos contribuidores é parcialmente automatizado através do *bot all contributors*[13] instalado no repositório. Qualquer responsável pode adicionar um contribuidor ao projeto ao submeter um comentário num *pull-request* aberto e deve obedecer o seguinte formato: *@all-contributors please add @someUser for code, docs, test,* etc. O *bot* deteta através de compreensão de linguagem natural e ao constatar um novo comentário é criado automaticamente um PR para adicionar o utilizador à tabela de contribuidores.

No repositório criado a partir do *template* mencionado, definiu-se que este seria um repositório único para o projeto e como tal o código referente a todas as aplicações está presente no mesmo e separado por pastas (*frontend, backend, database*).

Tendo em conta o modelo apresentado, este foi adaptado de forma a incluir as especificidades do projeto tal como se pode observar na figura do Apêndice A - ReadMe.

[12] https://opensource.org/licenses/MIT
[13] https://www.allcontributors.org/

No readMe foram adicionadas três secções à tabela de conteúdos e estas são: "How to Run", "Requirements", "Usage".

Com o intuito de otimizar a organização da página inicial do repositório foram escritas duas páginas *wiki* para as secções "How to Run"[14] e “Requirements"[15] acessíveis ao clicar nas respetivas hiperligações. Todas as aplicações podem ser executadas em *containers*. A primeira secção - “How to run”, apresenta um guia de como correr o projeto localmente e através do *Docker*[16]. As bases de dados devem ser corridas em containers *container,* no entanto as restantes poderão ser executadas através da linha de comandos (forma tradicional).

4.4.2. Github - Organização de tarefas

Nesta subsecção é apresentado de que forma o *github* permitiu organizar o projeto no que diz respeito à adição de tarefas, concretização e monitorização das mesmas aplicando a metodologia *kanban* com o intuito de entregar valor incrementalmente.

Após o levantamento de requisitos, as resultantes *user storys* e respetivas funcionalidades descritas no próximo capítulo, são divididas em tarefas e adicionadas como *features* na aba *issues* do *github* e identificadas utilizando a *tag user story*. Assim, para cada US são geradas novas tarefas e estas seguem o *template* apresentado na Figura 9.

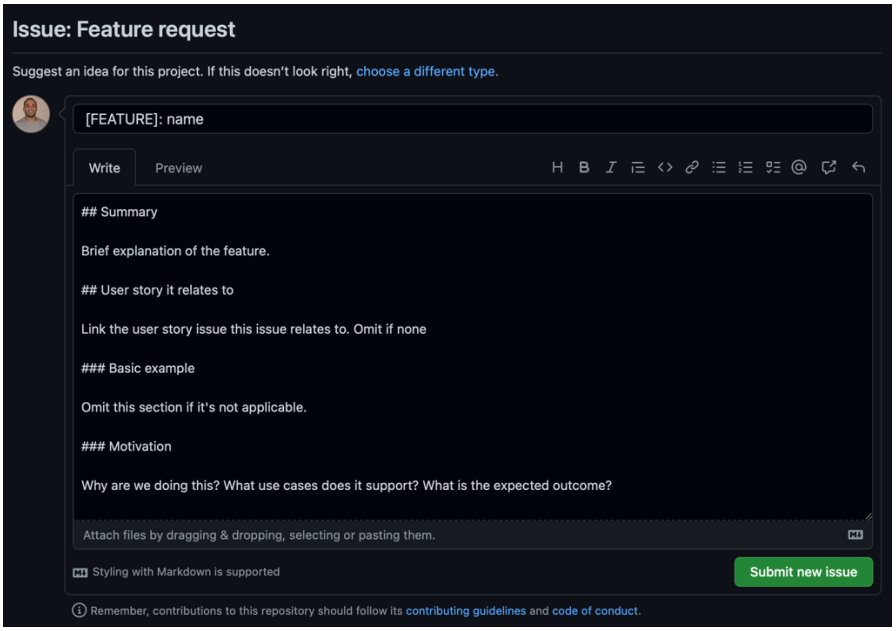


Figura 9 - Template de criação de uma issue/feature

[14] https://github.com/xgeekshq/split/wiki/How-to-run
[15] https://github.com/xgeekshq/split/wiki/Requirements
[16] https://www.docker.com/

Estas *issues* correspondem a pequenas funcionalidades/tarefas e uma vez finalizadas determinam a US como concluída. Para cada tarefa de menor dimensão é criada uma *issue* que também segue o *template* apresentado na Figura 9 e são-lhes atribuídas *tags* para auxiliar os contribuidores a compreenderem visualmente a que área a tarefa diz respeito e posteriormente para formar *change logs* que apresentam as mudanças por secções baseando-se nas respetivas *tags*. Como tal, para cada uma *issue* é pelo menos associada a *tag, backend, frontend, bug*, etc

Na secção *projects* encontram-se diversos quadros *kanban* referentes a cada uma das áreas do projeto, como tal, existe o quadro para infraestrutura, *backend*, *frontend*, *design* e *bugs*. Cada um dos quadros, como é possível observar na Figura 10, é composto por um conjunto de colunas e estas são: "Todo", "In progress", "Review in progress", "Review approved" e "Done".

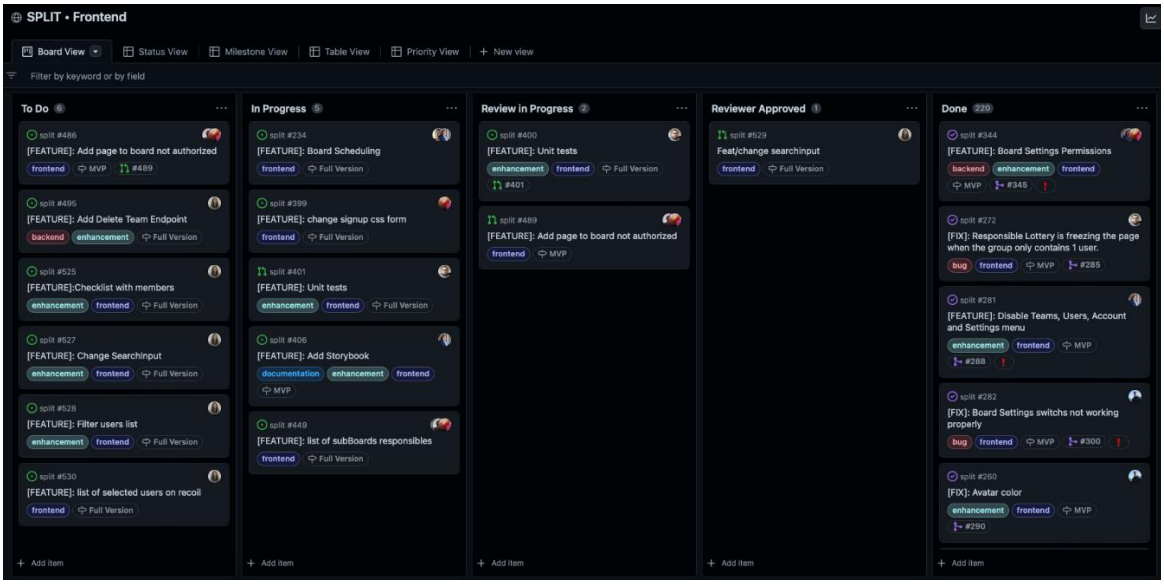


Figura 10 - Quadro kanban presente no github – frontend

Assim, na criação dos *issues* para cada funcionalidade ou bug, estes devem ser associados ao quadro respetivo e automaticamente são inseridas na coluna "todo" que corresponde ao clássico *backlog*. O processo de transição entre as colunas dos cartões que representam um *issue* ou *feature* do quadro é semiautomático. Embora seja possível automatizar totalmente através de *github actions*, o *developer* deve atribuir a tarefa ao próprio e mover o cartão para a coluna "in progress" quando a decide iniciar. Depois da tarefa concluída, um PR deve ser criado com o *template,* apresentado na Figura 11, e deve ser efetuado um pedido de revisão à respetiva equipa da área da qual a tarefa diz respeito. Assim, o cartão da tarefa é automaticamente transferido para a coluna "Review in progress".

Após aprovação do PR o cartão transfere-se automaticamente para a coluna "Review approved".

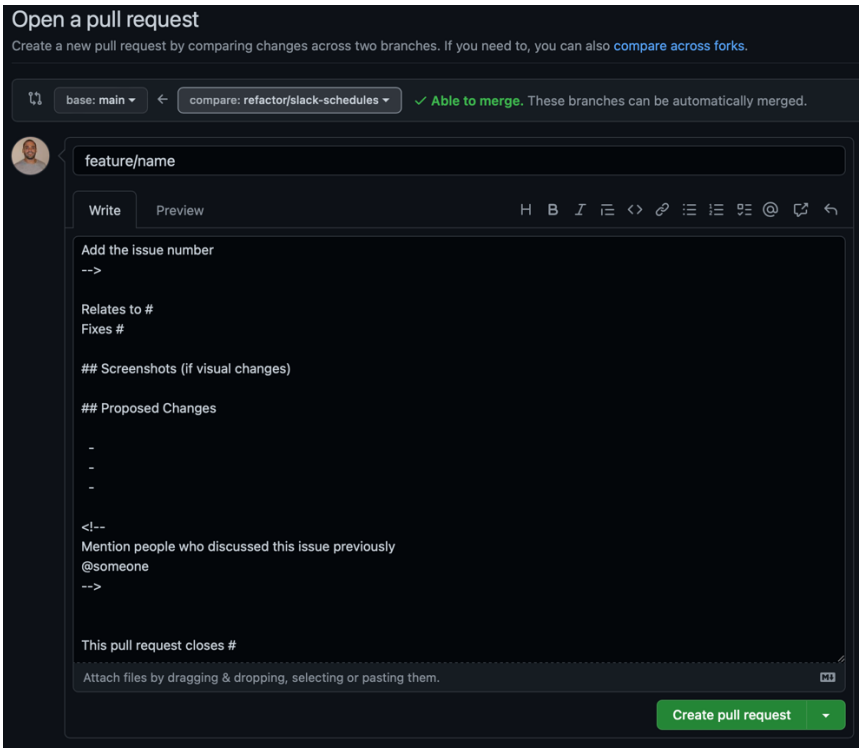


Figura 11 - Modelo de pull request

Desta forma, qualquer contribuidor pode aceder ao quadro da área que tem interesse ou à secção dos *issues* e selecionar qualquer uma das tarefas que pretende desenvolver. A equipa de desenvolvimento ou contribuidores através do quadro conseguem compreender e acompanhar o estado atual das tarefas e assim promover um sincronismo constante.

4.4.3. Balanço da aplicação da metodologia

Nesta secção é apresentado o balanço da aplicação da metodologia tendo em conta os princípios e práticas da metodologia descritos na secção 4.3. Isto é, perante os vários pontos mencionados, compreender de que forma foram aplicados durante o desenvolvimento do projeto. É importante referir que nem todas as práticas e princípios foram aplicados devido à característica *open-source* que o projeto apresenta.

Tendo em conta os princípios básicos da metodologia aplicados é possível afirmar que foi possível sincronizar os elementos integrantes do projeto simplificando a colaboração entre indivíduos, fornecendo a visão geral do trabalho em desenvolvimento através da visualização do fluxo de trabalho utilizando o quadro *kanban* do github.

Não foi necessário efetuar a medição do tempo de entrega das tarefas de forma concreta uma vez que não se verificaram problemas no fluxo de trabalho, contudo as tarefas cujo tempo de entrega se verificou superior ao normal mereceram especial atenção pela equipa de forma a compreender as dificuldades e auxiliar na conclusão da mesma evitandos engarrafamentos nas próximas tarefas.

As políticas de processo são explicitas, como referido anteriormente no *github* é apresentado o guia de como contribuir no projeto e o código de conduta. Nestes é possível verificar como deve ser o processo de contribuição desde a atribuição de uma tarefa até à conclusão e consequente *merge* da tarefa no *main branch*. Adicionalmente, como quem pode aprovar os PR são os *owners* do projeto e por isso membros da empresa, é indicado que os PRs devem ser aprovados se a execução dos testes for bem-sucedida.

Tendo em conta as práticas mencionadas na secção 4.3, seguiu-se o desenvolvimento incremental ao integrar valor por *features*; facilmente se aceitou mudanças ao longo da evolução do projeto uma vez que a mentalidade da equipa de desenvolvimento sempre foi nesse sentido e devido à natureza do projeto que implica a possível alteração de membros a qualquer momento; todos os elementos conseguiram respeitar as funções de cada membro, processos, responsabilidades e sempre foi encorajada a liderança por qualquer contribuidor da empresa se assim necessário.

5. Pesquisa, conceção e desenho do produto

O processo de *design* de produto é essencial para a conceção do mesmo e é nesta fase que são feitas um conjunto de tarefas de forma a garantir que o produto será bem definido, isto significa que é essencial compreender as necessidades dos utilizadores, explorar as ideias que surgem para o produto e materializá-las para perceber a sua viabilidade. Segundo os autores do artigo "*Design Thinking 101*" [15], *design thinking* é uma metodologia que proporciona uma abordagem baseada na solução para problemas complexos que são mal definidos ou desconhecidos e centra-se em servir as necessidades dos seres humanos. O processo de *design thinking* é compreendido em seis fases distintas: empatia, definição, idealização, prototipagem, testes e implementação como é possível observar na Figura 12.

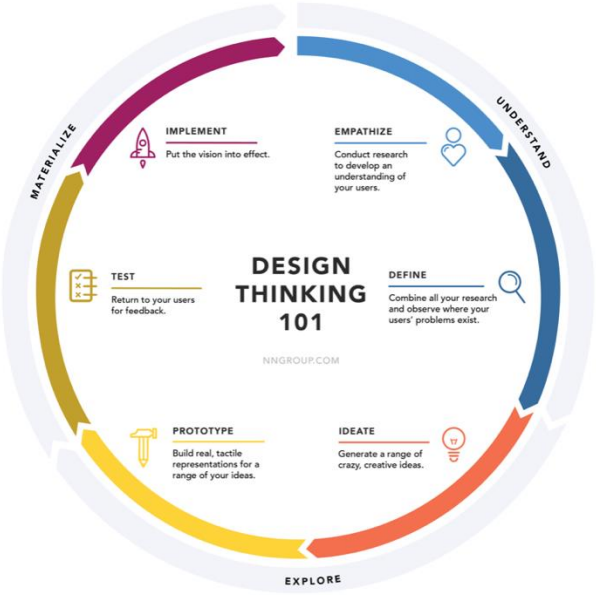


Figura 12 - Fases da metodologia *design thinking* [15]

Assim, a fase de **empatia** representa a pesquisa efetuada de forma a compreender as necessidades dos utilizadores, falando com estes, observando as suas ações, como pensam, o que querem, o que os motiva e as suas frustrações. O objetivo é obter informação de forma a criar empatia e compreender as perspetivas dos utilizadores. Na segunda fase, intitulada de **definição**, diz respeito à combinação de toda a pesquisa e observação na fase anterior e começo da extração de oportunidades. Nesta, as observações são organizadas e são procurados padrões das experiências dos utilizadores e através destes são encontrados pontos de interesse. A terceira fase designada de **idealização** refere-se à geração de ideias criativas que tentam resolver as necessidades dos utilizadores na fase anterior não existindo qualquer

limitação. A quarta e denominada de **prototipagem** implica a criação de representações reais das ideias geradas anteriormente. Nesta fase destina-se à compreensão de quais são as ideias incluídas no protótipo que podem ou não funcionar. Aqui pode começar-se a medir o peso e a viabilidade das ideias implementadas nos protótipos através do *feedback* dos utilizadores e partes interessadas. Podem ser criados *wireframes*, rascunhos e ou experimentações de *designs*. A quinta fase diz respeito aos **testes**, e nesta os protótipos são apresentados aos utilizadores reais de forma a verificar se permitem atingir os seus objetivos. A última fase e mais importante é a **implementação** (do *design*) onde a solução é materializada. Este ciclo de fases pode ser repetido se necessário com o intuito de aprimorar a solução.

Esta forma de pensar sobre o desenvolvimento do produto tem imensas vantagens, uma vez que é centrado no utilizador e por isso pretende resolver as necessidades reais dos utilizadores e testar as ideias geradas com utilizadores reais.

Finalizado o enquadramento, neste capítulo é apresentada a concretização do plano definido na secção Fase de pesquisa e conceção do produto, que seguiu a linha de pensamento apresentada anteriormente e que diz respeito a toda a pesquisa efetuada para a idealização e desenho do produto SPLIT e a partir desta, definir as funcionalidades, bem como o design e experiência de utilização. É importante referir que este processo foi liderado e maioritariamente executado pela *designer*, com o suporte do *developer* da equipa e autor deste documento.

5.1. Entrevistas

Nesta secção é apresentado o plano e guião das entrevistas e como estas foram realizadas. Previamente à execução das entrevistas foi necessário recrutar voluntários do grupo de empresas no qual faz parte a xgeeks e como tal foi possível reunir nove entrevistados. Esta fase pretende perceber qual é a experiência dos utilizadores usando ferramentas de retrospetivas em equipas.

As entrevistas foram compostas de três participantes, a *designer* cujo papel é de entrevistador, o *developer* e autor deste documento que é o responsável por tirar notas, o último e mais importante é o participante que tem o papel de entrevistado. Encontrados os voluntários, as sessões foram divididas em três por dia.

5.1.1. Guia de entrevistas

O guião utilizado para as entrevistas pode ser observado no Apêndice B – Guião da entrevista. Este permitiu auxiliar na condução da entrevista, de forma a procurar respostas a perguntas que permitem compreender as necessidades dos utilizadores. No início da entrevista, é efetuada uma introdução onde a entrevistadora e quem tira as notas se apresentam, indicam que o objetivo da sessão é aprender e compreender, a partir de cada um, sobre a experiência na execução de retrospetivas em equipas e que embora as perguntas sejam em grande número nunca será um teste. É questionado se a entrevista pode ser gravada apenas para uso interno e dá-se início à entrevista perguntando qual é a função do entrevistado na sua empresa e como é o seu dia-a-dia no trabalho.

Seguidamente são colocadas as questões mencionadas no apêndice anteriormente referido com o intuito de obter conhecimento referente ao processo de retrospetivas e ferramentas utilizadas pelos utilizadores; compreender a opinião dos entrevistados sobre o impacto do número de elementos de uma equipa na execução de retrospetivas; verificar se é importante a integração de outras ferramentas naquela considerada principal para realização de retrospetivas; compreender os problemas e verificar se a inteligência artificial pode contribuir para ajudar no processo; finalmente são realizadas outras questões gerais.

5.2. Recolha e tratamento dos dados

Uma vez definido o guia de entrevistas, realizou-se um *template* onde são colocadas as informações dos entrevistados, as notas e a partir destas, depois de analisadas, efetuou-se um mapa de experiência e extraíram-se possíveis oportunidades. Estas são frases ditas pelos entrevistados que permitem encontrar padrões entre os utilizadores e que por norma indicam alguma necessidade, algum problema ou pensamento.

O mapa de experiência é representado através de um esquema e refere-se ao processo das retrospetivas que o entrevistado descreveu e serve para compreender como o processo é executado por cada participante e como pode ser melhorado. Assim, na Figura 13 é possível observar o modelo definido e as secções descritas anteriormente. No Apêndice C - Relatórios de entrevistas, são apresentados os resultados de cada entrevista, omitindo a entidade dos participantes.

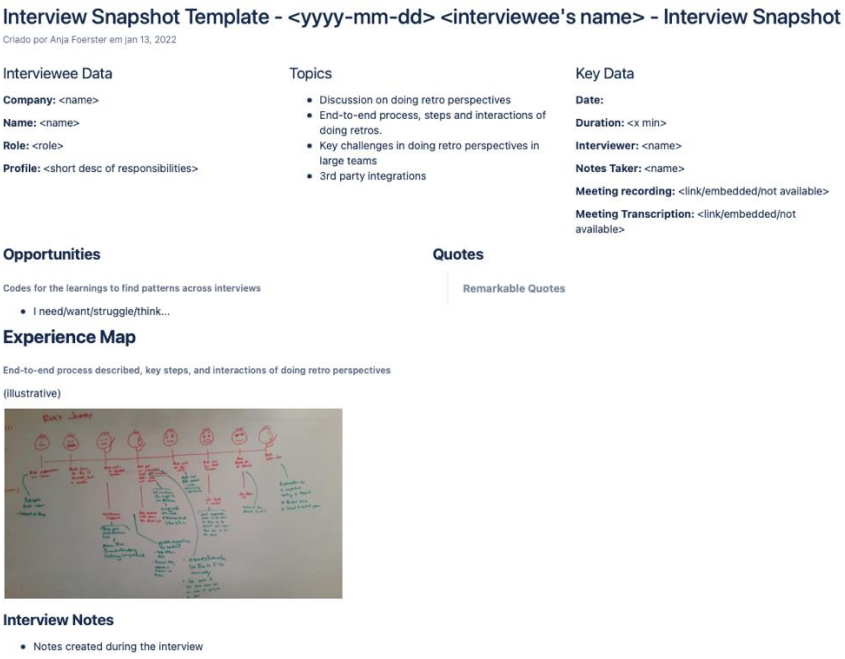


Figura 13 - Modelo exemplo para introdução dos dados extraídos das entrevistas

Uma vez finalizadas as entrevistas e as notas obtidas, o próximo passo implica a criação dos mapas da experiência do utilizador ao executar uma retrospetiva.

No Apêndice D – Mapeamento das experiências, retirado do ficheiro desenvolvido na ferramenta *figma* [16], podem ser observados os mapas de experiência extraídos de cada utilizador ao utilizar a sua ferramenta de retrospetiva. Na Figura 14 é apresentado a experiência obtida a partir da entrevista do utilizador 7 onde é representado sequencialmente o processo de retrospetivas desse participante. Ao visualizar a maioria dos mapeamentos é possível verificar alguns processos de retrospetivas do início até ao fim e é possível detetar alguns pontos comuns que na maioria incidem sobre a utilização de um quadro onde são inseridos cartões e que depois de votados são extraídos os itens de ação. Os pontos distintos dependem do número de elementos das equipas, portanto em equipas com poucos membros, toda a equipa é convidada para a retrospetiva; em equipas grandes estas são divididas em subequipas e é efetuada uma retrospetiva em cada uma das equipas divididas e posteriormente ocorre a convergência dos cartões criados num só quadro.

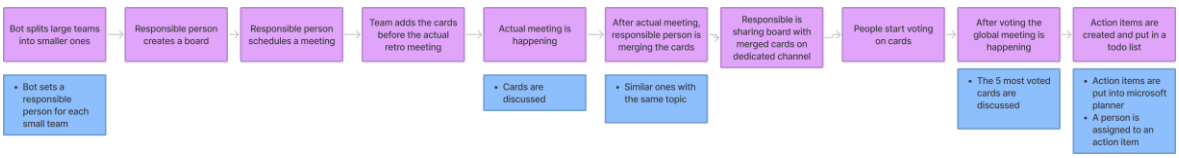


Figura 14 - Mapeamento da experiência do utilizador 7

Após a execução do mapeamento da experiência, a partir das entrevistas dos utilizadores, são extraídas as frases consideradas de maior relevância, ou seja, aquelas que podem sugerir oportunidades e ou funcionalidades para incluir no produto a desenvolver. Essas frases são organizadas e avaliadas de forma a perceber quais podem dar origem a reais oportunidades. A organização das frases foi efetuada através de uma tabela com o objetivo de compreender aquelas que são mencionadas mais vezes pelos utilizadores e com base nesse critério é determinada a confiança (baixa-media-alta) e o impacto das mesmas (baixo-médio-alto) com o intuito de filtrar as oportunidades por aquelas que maior interesse poderão ter para os utilizadores. No Apêndice E – *Airtable* de descobertas, é reunido o conhecimento extraído das entrevistas - as possíveis oportunidades - e ordenadas com base nos critérios mencionados anteriormente.

Verificando o apêndice referido anteriormente é possível concluir que vários utilizadores indicaram que necessitam de recorrer a uma ferramenta que permita efetuar o rastreamento adequado dos itens de ação extraídos das retrospetivas. A afirmação é considerada de alto impacto porque foi mencionado diversas vezes e porque foi realçado que existem dificuldades em compreender o estado em que se encontram as tarefas definidas com o intuito de melhorar o que correu menos bem. Os entrevistados indicam que as retrospetivas não devem ser efetuadas com um elevado número de pessoas e que por isso estas devem ser divididas em grupos mais pequenos. As retrospetivas com elevado número de pessoas inibe-as de falar e apresentarem a sua opinião. Estas observações são de elevado impacto porque a maioria indicou este problema, mas também porque é uma dificuldade que a empresa enfrenta e nenhuma ferramenta contribui na agilização do processo de retrospetivas em equipas grandes.

O próximo destaque indicado pela maioria dos utilizadores e atribuído como alto impacto diz respeito à possibilidade de votar no sistema uma vez que este processo permite representar a opinião dos participantes, realçando as questões que estes consideram mais relevantes.

É indicado pela maioria dos participantes que a ferramenta de retrospetivas deve ser visualmente apelativa, e apresentar funcionalidades que tornem o processo de retrospetivas divertido. Também é referido que deve ser colaborativa - proporcionando interações entre os utilizadores. Apesar de ser crucial não foi considerado de impacto alto uma vez que é algo

intrínseco a uma plataforma desta categoria, não otimiza o processo de retrospetivas nem solucionam o problema deste projeto.

No que diz respeito ao agrupamento de cartões que se referem ao mesmo tópico, é possível constatar que alguns participantes mencionaram que nas suas retrospetivas este é um processo manual e que deveria de ser automatizado com o intuito de evitar perdas de tempo desnecessárias. Este ponto foi considerado de alto impacto porque constatou-se que tem implicação direta no processo de execução de retrospetivas.

Considerado de impacto e confiança média, os utilizadores indicam que é importante que a ferramenta permita anonimização de conteúdo de forma a promover a introdução de opiniões e consequentemente cartões. É também indicado que as reuniões devem ter limitação de tempo bem como os vários assuntos a discutir, de forma a conseguir, controladamente, discutir todos os tópicos relevantes. Os utilizadores consideram relevante a possibilidade de identificar palavras-chave nos cartões ou que estes estejam associados a um assunto de forma a ser possível visualizar todos os cartões que dizem respeito ao mesmo tópico.

De forma a concluir a análise das observações dos entrevistados a partir da tabela mencionada, é possível verificar que aquelas frases que não foram mencionadas são de baixa confiança e impacto uma vez que são referidas por poucos utilizadores.

5.3. Oportunidades e levantamento de requisitos

Esta secção tem o propósito de apresentar a execução do processo, desenvolvido pela *designer*, que permitiu gerar um conjunto de requisitos que o sistema deve cumprir a partir a partir das possíveis oportunidades anteriormente referenciadas.

Findada a organização das observações dos entrevistados, o passo seguinte passa por mapear as oportunidades com maior confiança. Isto significa que as frases analisadas anteriormente e que apresentam impacto relevante devem ser agrupadas. Com isto é possível gerar uma árvore solução das oportunidades e com base nos resultados, criar questões para serem utilizadas no *workshop* a realizar no passo seguinte. O *workshop* tem o intuito de procurar soluções para resolver os problemas encontrados através de novas funcionalidades a implementar na aplicação. Com esse intuito, são geradas perguntas que começam por: “como poderíamos...?”.

Para cada tema considerado de elevado impacto: itens de ação, votação, suporte visual, combinação de cartões e tamanho de equipa, todas as possíveis oportunidades são colocadas em duas colunas: a primeira diz respeito às frases retiradas diretamente das entrevistas e a segunda são aquelas subentendidas a partir de outras frases extraídas, como se pode observar no Apêndice F – Mapa de oportunidades e retirado do ficheiro desenvolvido na ferramenta *figma* [17]. Seguidamente, estas são reformuladas e posteriormente, as similares são combinadas numa oportunidade genérica. Após este processo é gerada a árvore solução de oportunidades. Na Figura 15 é verificado do resultado deste processo e pode ser observado no ficheiro anteriormente referenciado.

Neste processo e como se pode observar próxima figura, começa-se por identificar que o objetivo de negócio é dar a conhecer o nome da empresa à comunidade e com isso encontrar potenciais colaboradores. O objetivo do produto é providenciar uma ferramenta fácil de utilizar e que permita efetuar retrospetivas em empresas de forma escalável.

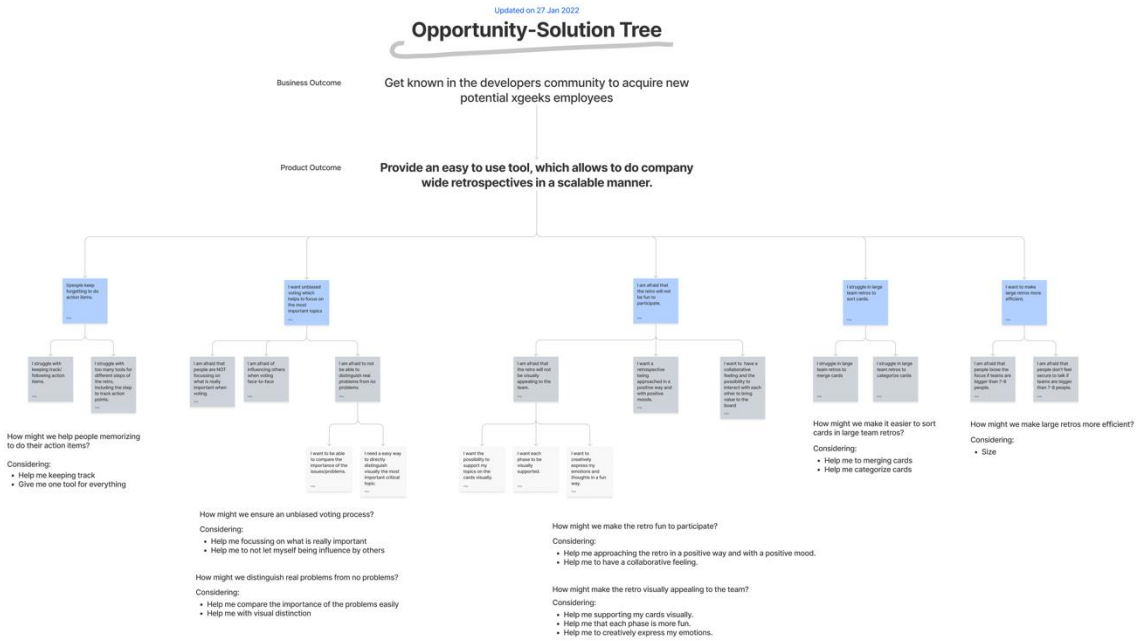


Figura 15 - Árvore solução de oportunidades

Observando a Figura 15, da esquerda para a direita é possível compreender que o primeiro problema mais abrangente diz respeito ao facto das pessoas se esquecerem de resolver os itens de ação. Derivado deste, e de forma mais concreta, há pessoas que têm problemas a rastrear ou seguir o estado dos itens de ação. Também é indicado que são usadas diversas ferramentas para as várias fases das retrospetivas, incluindo para acompanhar os pontos de ação e que por isso é difícil rastrear o que está a ser feito para resolver os

problemas. A partir destas oportunidades a questão originada para ser apresentada no *workshop* de forma a procurar solução deste problema é: "Como poderíamos ajudar os utilizadores a relembrar dos seus itens de ação?", tendo em conta também como ajudar no rastreamento e proporcionar uma ferramenta única para todos os passos do processo.

O foco seguinte diz respeito ao tópico de votação no qual é indicado que os utilizadores gostariam de ter uma votação eficaz e imparcial de forma a dar relevância nos assuntos mais importantes. Este tópico surge no seguimento de algumas pessoas sentirem que certos membros não se focam no que realmente é importante no ato do voto; estas sentem também que não pretendem influenciar a votação quando a retrospetiva decorre presencialmente; finalmente sentem receio de não conseguir distinguir os problemas reais dos menos importantes. Esta afirmação surge no seguimento de outras que indicam que os utilizadores gostariam de conseguir comparar o nível de importância dos problemas encontrados e assim distinguir visualmente os relevantes dos menos relevantes. Terminada esta análise foi possível formar duas perguntas que pretendem encontrar soluções para estes problemas. A primeira "Como poderemos assegurar que os votos são imparciais?", e esta questão deve ter em consideração como se pode ajudar os utilizadores que se foquem no que realmente é importante e como não ser influenciado por outros. A segunda questão, "Como poderemos distinguir os problemas reais daqueles menos importantes?" tendo em consideração como se pode definir e distinguir a importância dos problemas visualmente.

O tópico seguinte refere-se aos elementos visuais onde é dito que existe receio das retrospetivas não serem divertidas ao participar. Isto surge no seguimento das opiniões extraídas dos entrevistados onde dizem que têm receio que a retrospetiva não seja visualmente apelativa à equipa. Foi possível formar esta assunção com base noutras afirmações dos utilizadores, tal como, a necessidade de existir suporte visual quando são dadas as opiniões, a necessidade de existir suporte visual para cada fase de uma retrospetiva e também quando é dito que necessitam de forma criativa e divertida de expressar emoções e ideias de forma divertida. Ainda neste tópico foi possível formar a ideia que os utilizadores pretendem uma retrospetiva que seja abordada positivamente por todos os intervenientes. Para finalizar este tópico, foi possível afirmar que os utilizadores pretendem sentir um ambiente colaborativo com a possibilidade de interação entre eles. Neste tópico e através das assunções formadas foi possível gerar duas questões: "Como poderemos tornar uma retrospetiva divertida de participar?", tendo em consideração como se poderá tornar esse momento divertido, positivo e capaz de oferecer uma sensação de colaboração. A segunda

questão: “Como poderemos tornar uma retrospetiva visualmente apelativa?”, tendo em consideração como apresentar visualmente as opiniões dos participantes, como tornar cada fase do processo divertida e como permitir que os utilizadores expressem as suas emoções de forna criativa.

Verificando o tópico seguinte foi possível constatar que as pessoas tendem a ter dificuldades em ordenar, num quadro de retrospetiva, os cartões quando estes existem em elevado número. Esta assunção surge no seguimento de se ter apurado que os utilizadores têm dificuldades nas retrospetivas, com elevado número de membros, de agrupar e categorizar os cartões semelhantes. Tendo em conta esta assunção surge a questão: “Como poderemos facilitar a ordenação de cartões em retrospetivas com um elevado número de membros?” e esta questão deve ter em consideração os dois pontos mencionados anteriormente.

Finalmente, o último tópico considerado relevante refere-se à necessidade de realizar retrospetivas com um elevado número de membros de forma eficiente. É indicado que sentem que as pessoas poderão perder o foco se as equipas das retrospetivas tiverem um número de elementos superior a 7-8 e que os membros das equipas grandes poderão não se sentir à vontade para expressar a sua opinião. Assim, gerou-se a questão “Como poderemos tornar as retrospetivas em equipas grandes mais eficientes?”, considerando o tamanho das mesmas.

O passo seguinte à formulação de questões é a organização e execução de um *workshop* com a duração de 60 minutos, com o intuito de responder às perguntas formuladas através de funcionalidades que uma aplicação poderá ter de forma a colmatar os problemas encontrados, e para tal, foi enviado um convite para os membros da empresa se voluntariarem em participar.

Uma vez encontrados os participantes, a *designer* efetuou uma apresentação do *workshop*, possível de observar em Apêndice G – Apresentação do workshop e retirado do ficheiro desenvolvido na ferramenta *figma* [18], onde apresentou o propósito deste evento e como será realizado. Assim, é feito um enquadramento e apresentando de forma breve o percurso de pesquisa do produto até ao momento, e descrito anteriormente. Nesta apresentação é indicado que terá duas fases sendo que a primeira diz respeito à priorização dos problemas através de uma votação. Portanto, e como se pode observar na Figura 16, os

problemas principais retirados da árvore de solução anterior, são sujeitos a votação e os carimbos/*emojis* representam o voto de cada utilizador.

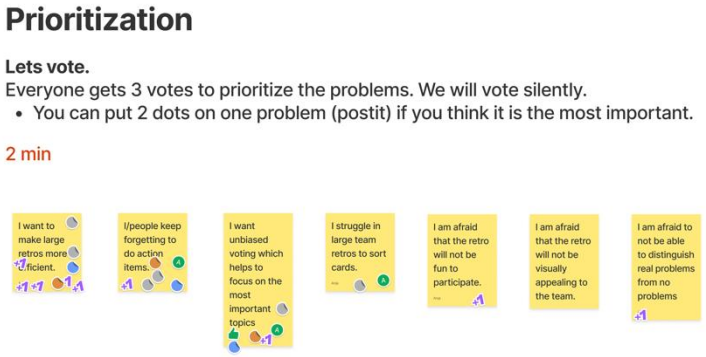


Figura 16 - Fase de priorização

Uma vez definidos os principais tópicos a abordar, a fase de idealização é a seguinte e esta é dividida em três passos que se repetem para cada questão formulada anteriormente. A primeira é a geração de ideias respondendo diretamente às questões "como poderemos...?" através de cartões que devem ser colocados no quadro e com o nome do participante. O passo seguinte implica que as ideias sejam lidas, debatidas e também agrupadas caso se verifiquem idênticas. Para finalizar, as ideias são votadas e avança-se para a próxima questão.

Apresentado o processo e após a votação nos tópicos mais relevantes, foi possível realizar, segundo a priorização definida, cinco das sete rondas devido à falta de tempo disponível e o resultado de cada uma destas pode ser observada no Apêndice H – Rondas do workshop e retirado do ficheiro desenvolvido na ferramenta *figma* [19].

Analisando as ideias geradas pelos participantes efetuou-se, através de uma matriz de decisão, uma análise esforço-valor em conjunto com a *designer*, EM e *developer* onde se enquadrou cada um dos cartões no respetivo quadrante. Assim, como se pode observar na Figura 17 retirado do ficheiro desenvolvido na ferramenta *figma* [20], da esquerda para a direita e de cima para baixo, o primeiro quadrante diz respeito a ideias de alto valor e alto esforço e como tal as ideias são consideradas menos prioritárias, no entanto poderão ser incluídas na aplicação como trabalho futuro.

Imediatamente à direita, o segundo quadrante representa as ideias que devem ser tidas em conta para incluir na aplicação numa primeira fase, uma vez que representam alto valor e baixo esforço.

Em baixo e à esquerda é representado o quadrante com as ideias menos relevantes uma vez que apresentam baixo valor e grande esforço. Por fim, no quadrante da direita, são colocadas as ideias possíveis de incluir na aplicação uma vez que, embora representem baixo valor, são consideradas de baixo esforço.

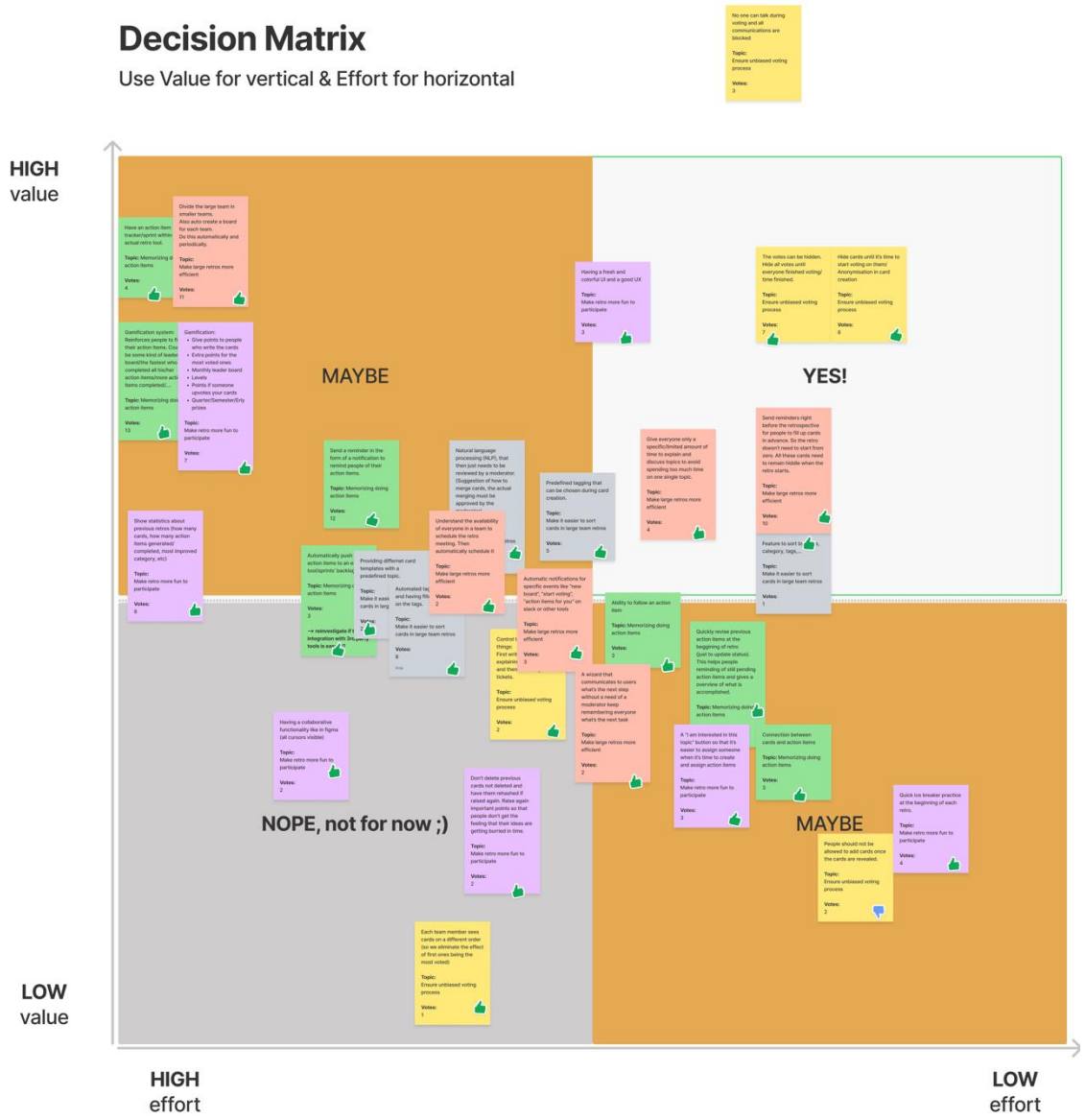


Figura 17 - Matriz de decisão

Uma vez concluído o processo de classificação das ideias e tendo em conta as ferramentas existentes que permitem realizar retrospetivas, foi possível definir cinco fases flexíveis e uma secção extra, sendo que cada uma consiste no desenvolvimento de um conjunto de funcionalidades. A extra diz respeito a funcionalidades que podem ser posteriormente desenvolvidas e pouco prioritárias. As fases são definidas como flexíveis porque poder-se-á dar o caso de se avançar com funcionalidades de fases diferentes da atual, se for justificável. Assim sendo, no Apêndice I – Funcionalidades, pretende-se apresentar

em formato tabela as funcionalidades definidas para cada uma das fases, no entanto, de seguida, é feito um breve resumo do que foi possível definir.

Na primeira, e considerada a fase mais importante para o arranque do projeto, foram definidas as funcionalidades fundamentais para uma primeira versão com valor e por isso passível de ser usufruído por parte dos utilizadores da empresa. Esta inclui autenticação e registo, um *dashboard* onde devem ser listados os quadros e onde deve ser possível realizar ações de gestão dos mesmos (criar, apagar, visualizar, filtrar, etc). O foco nesta primeira iteração do produto incide sobre os quadros *kanban* que é essencial para a realização de uma retrospetiva remota. Assim, funcionalidades de criação, edição, listagem e remoção de cartões, comentários e votos, em conjunto com a possibilidade de mover, interagir em tempo real e agrupar cartões são aquelas com maior relevância. É possível afirmar que todas funcionalidades desta fase se encontram desenvolvidas.

Na segunda fase e referente à gestão de utilizadores e equipas devem ser desenvolvidas funcionalidades tais como criação, listagem, remoção, filtragem de utilizadores e atribuição de funções dos utilizadores dentro da plataforma e dentro de uma equipa. No que diz respeito à criação de quadros, deve existir a possibilidade de associar uma equipa a um quadro principal, subdividi-la em equipas de menor dimensão e criar o respetivo subquadro, existindo uma relação de parentalidade entre o principal e os restantes. Após criação de um quadro deste género, deve ser gerado automaticamente um "*job*" que mensalmente deve ser despoletado e que replica o processo efetuado. Nesta fase, todas as funcionalidades que são mencionadas na tabela do Apêndice I – Funcionalidades encontram-se realizadas.

A terceira fase diz respeito a integrações, cujo objetivo é integrar o *slack* na plataforma de forma a ser possível notificar os utilizadores de que o processo de retrospetivas se iniciou, apresentar a divisão dos membros por equipas e o respetivo responsável. Como funcionalidades avançadas pretende-se possibilitar a interação no quadro através de comandos do *slack* sem existir a necessidade de abrir a aplicação. Nesta fase encontram-se desenvolvidas as funcionalidades básicas do *slack* e algumas avançadas.

A fase quatro diz respeito a agendamentos. Assim, na criação de um quadro o utilizador deve poder definir se pretende que o quadro seja criado recorrentemente, com uma determinada frequência e hora. A opção de edição ou remoção de agendamentos deve existir tal como uma secção própria onde deve ser possível visualizar os próximos agendamentos

em formato de lista, mas também através de um quadro interativo. As funcionalidades desta fase ainda não se encontram todas desenvolvidas.

A quinta e última fase diz respeito a personalização e como tal deve ser possível associar um cartão a um item de ação, definir palavras-chave para os cartões, exportar quadros, agrupar cartões por palavra-chave ou, através de compreensão de linguagem natural, modificar a cor dos cartões nas colunas e também definir *templates* a utilizar na criação de quadros ou cartões. Algumas funcionalidades desta fase ainda não se encontram desenvolvidas.

Sem fase definida, foram adjudicadas um conjunto de outras funcionalidades que, embora não sejam prioritárias, poderão acrescentar valor ao produto. A apresentação de estatísticas ou criação de um sistema de recompensas podem ser duas das funcionalidades que poderão atrair e reter os utilizadores. A verificação da disponibilidade dos utilizadores para agendar uma reunião de retrospetiva pode ser outra funcionalidade em destaque uma vez que nem sempre é fácil encontrar uma data acessível para todos os participantes.

5.4. Story map e papéis de utilizador

Uma vez definidas as funcionalidades, é importante imaginar um possível fluxo de utilização da plataforma tendo em conta o processo definido para as retrospetivas realizadas na xgeeks. Assim, foi criado um mapa separado por fases e nestas, por papéis, com o intuito de verificar a interação do utilizador com a plataforma em cada uma das fases e como esta reage a determinados *inputs.* Nesta fase foram também definidos papéis para cada utilizador ao nível das equipas, ao nível dos quadros, mas também ao nível da plataforma.

Na Figura 18, é possível observar os papéis definidos e as permissões associadas a estes. Assim, um membro deve poder criar equipas, quadros ou ser administrador da plataforma. Um utilizador, dentro de uma equipa poderá ser administrador, *stakeholder* ou membro. Um *stakeholder* será um administrador, mas que não participa nos sub-quadros gerados numa retrospetiva dividida. Os administradores podem nomear outros *admins* ou *stakeholders* e também criar, apagar ou editar quadros dessa equipa.

Um administrador da plataforma deve poder gerir todo o conteúdo relacionado com equipas, quadros ou utilizadores e também definir outros membros como super-administradores.

Um membro deve poder também criar, editar ou apagar quadros pessoais. Se o utilizador for o criador do quadro deve poder adicionar ou remover utilizadores e atribuir-lhes determinadas funções. Assim, o administrador de um quadro poderá definir outros membros como responsáveis e por isso serão considerados de administradores.

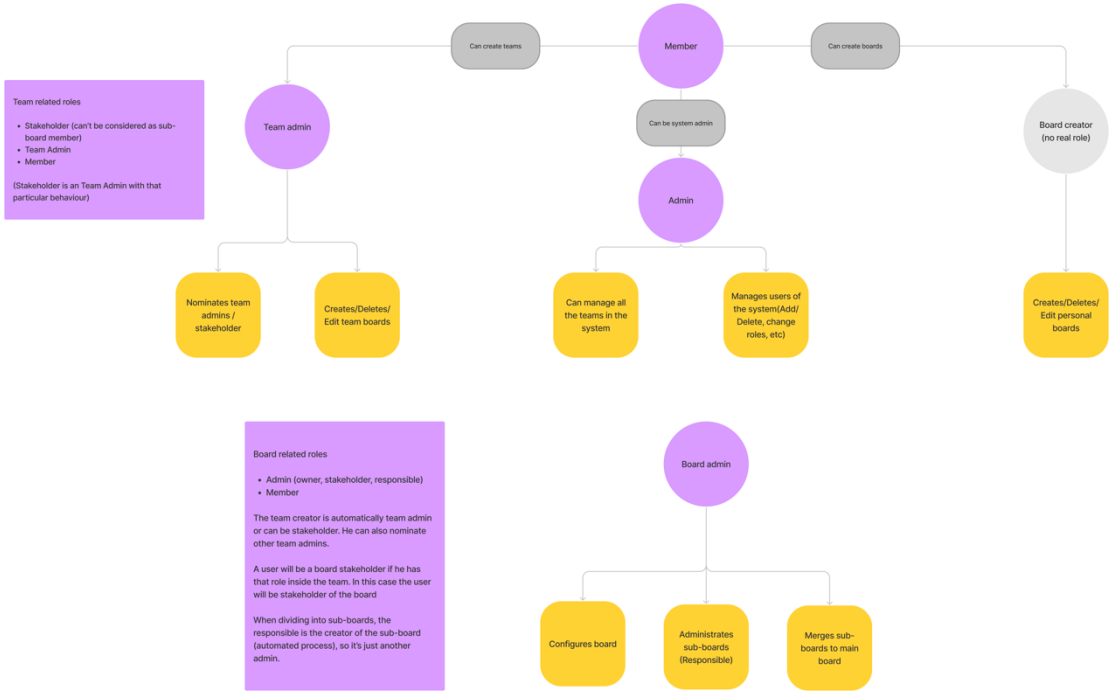


Figura 18 - Papeis dos utilizadores nas equipas, nos quadros e na plataforma

Quando se trata de quadros gerados ao dividir uma equipa, os quadros das subequipas devem ter um responsável que deve ser designado automática e aleatoriamente, que deve poder editar as configurações, selecionar um novo responsável ou passar automaticamente os cartões do subquadro para o principal. O quadro principal gerado do processo de divisão de uma equipa poderá ter *stakeholders* que são adicionados automaticamente se estes tiverem esse papel na equipa e tiverem permissões para editar as configurações de um quadro.

Um *story map* é um elemento do processo de *product design* que pretende apresentar um possível fluxo de utilização do produto, ainda não desenvolvido, para compreender se este é eficaz e pode ser implementado. Aquele criado para este projeto pode ser observado no Apêndice J – Story map e foi retirado do ficheiro desenvolvido na ferramenta *figma* [20] sendo que na primeira fase, denominada de registo, o administrador da equipa deve registar-se e criar uma equipa. Na fase de configurações deve enviar convites aos utilizadores para se registarem na plataforma e estes depois de se registarem devem ser adicionados à equipa.

Para criar uma retrospetiva o administrador deve ter a hipótese de escolher se pretende criar uma retrospetiva normal ou dividida. Se escolher normal deve poder adicionar manualmente os participantes ou selecionar uma equipa e todos os membros serão adicionados ao quadro criado. Se escolher dividida, uma equipa é dividida, o utilizador deve escolher o número de equipas ou o número de elementos por subequipa. Em ambas as opções deve ser possível definir um conjunto de configurações iniciais ou manter as predefinições, bem como agendar uma data para a execução da próxima retrospetiva com as mesmas configurações.

Após o utilizador tomar as decisões de criação do quadro, se tiver selecionado uma retrospetiva dividida, os utilizadores de uma equipa devem ser divididos em equipas mais pequenas, designadas de subequipas, deve ser selecionado um responsável para cada subequipa e o quadro principal deve ser criado tal como um para cada subequipa com as configurações selecionadas. Na retrospetiva normal, o quadro principal deverá ser criado com os utilizadores adicionados manualmente ou com os utilizadores da equipa selecionada usando as configurações definidas pelo administrador. Se o administrador pretender notificar os participantes via *slack* estes devem receber os convites para as retrospetivas através de uma notificação.

Se a retrospetiva for dividida, a plataforma deverá ter a responsabilidade de enviar uma mensagem para um canal existente, informando que ocorreu a divisão da equipa, criar um canal para cada subequipa e outro para todos os responsáveis, sendo que em cada um destes canais deve ser enviado o URL para o respetivo quadro.

Na fase de preparação o responsável de cada quadro de equipa deve poder efetuar alterações nas configurações nomeadamente esconder os cartões criados. Se tiver sido criado uma retrospetiva dividida, deverão poder alterar o responsável e ao fazê-lo deve promover uma notificação no *slack*. O principal objetivo desta fase é a criação de cartões de forma assíncrona por parte dos utilizadores até decorrer a retrospetiva. Existindo a funcionalidade de agendamento e lembretes, devem poder ser enviadas notificações no *slack* para relembrar que a retrospetiva decorrerá dentro de X dias e por isso devem ser criados os cartões sobre o que correu bem e/ou menos bem.

Na fase da retrospetiva, os participantes devem juntar-se através de alguma plataforma de reuniões, o responsável poderá retira a opção de esconder os cartões e estes devem ser discutidos. Se existirem cartões cujo assunto é semelhante deverão ser agrupados.

No final da reunião e considerada como uma fase opcional, porque só deve existir em retrospetivas divididas, o responsável poderá indicar através de uma funcionalidade que a retrospetiva está concluída e como tal os cartões devem ser enviados automaticamente para o quadro principal e deve ser enviada uma notificação para o canal dos responsáveis dizendo que a equipa Y terminou a sua retrospetiva.

Após realização de uma retrospetiva normal o administrador/responsável deve poder iniciar a fase de votação quando desejar através de uma funcionalidade para o efeito. No caso de retrospetivas divididas, quando todos os responsáveis indicarem que a sua retrospetiva está concluída, estes devem reunir-se uma vez mais e agrupar os cartões que existem no quadro principal com assuntos semelhantes. De seguida podem dar início à fase de votação.

Para finalizar, na retrospetiva normal e na dividida, após a votação, os cartões poderão ser ordenados por votos, estes devem ser discutidos e criados itens de ação com o intuito de resolver o que correu menos bem e devem ser associados a um participante. A reunião para a discussão e criação de itens de ação nas retrospetivas divididas deve ser feita com os responsáveis das subequipas e *stakeholders,* enquanto a normal deve ser realizada com todos os membros da equipa.

5.5. Site map

A presente secção tem como objetivo expor o mapeamento do site, isto é, apresentar as páginas do produto e as funcionalidades que cada uma contém. Assim, e observando o Apêndice K – Sitemap retirado do ficheiro desenvolvido no *figma* [21], é possível verificar que a plataforma deve apresentar uma *landing page* onde o utilizador poderá registar-se, recuperar a *password* ou autenticar-se através de credenciais ou SSO. Uma vez autenticado, o utilizador deve ser redirecionado para o *dashboard* onde se deve poder verificar os quadros das retrospetivas dos últimos três meses em que o utilizador participou e/ou as retrospetivas agendadas.

Na página dos quadros deverão ser listados todos os quadros pessoais, de equipa e divididos. Também deve ser possível verificar as próximas retrospetivas. Na página de cada quadro deverão ser apresentadas as colunas com os cartões, comentários, votos e onde poder-se-á efetuar todas as operações apresentadas na fase 1 do Apêndice I – Funcionalidades.

Na primeira página de criação de um quadro pretende-se que seja possível optar por criar uma retrospetiva normal ou dividida e o utilizador deve ser redirecionado para a página seguinte de criação e configuração de um quadro através das funcionalidades descritas no apêndice referido anteriormente.

Na página dos utilizadores deverá ser possível observar a lista de utilizadores da plataforma e se o utilizador for super-administrador deve poder atribuir esse papel a outros ou apagá-los, mas também aceder aos detalhes dos utilizadores, verificar a que equipas pertence e definir o seu papel dentro de cada equipa.

Na página das equipas pretende-se que seja possível criar e visualizar a lista de equipas do utilizador. Na página de cada equipa deve poder-se observar a lista de membros e se o utilizador autenticado for administrador deve poder adicionar ou remover utilizadores e editar a função de cada utilizador na equipa.

Na página de definições devem ser definidas algumas configurações globais da plataforma tal como as configurações de integrações como o *slack*.

Na página do perfil do utilizador deve ser possível alterar a password, apagar a sua conta, mas também alterar as suas informações pessoais.

5.6. Explorações de interfaces e biblioteca de design

Uma vez definidas as páginas da plataforma, o passo seguinte seria a criação de *wireframes,* no entanto, o tempo disponível e o rácio esforço-valor não compensou a execução das mesmas e como tal seguiu-se para a realização de explorações de interface com o intuito de validar a base do *design* a ser desenvolvido. No Apêndice L – Explorações de interface, verificam-se as explorações de interface efetuadas pela *designer*. Nestas foram testadas diferentes estruturas para a barra lateral de navegação, isto é, duas dimensões no que diz respeito na altura e outra quando se encontra colapsada. Foi efetuado um estudo de cores para os cartões a serem inseridos nas colunas, nos possíveis botões das plataformas, mas também nos ícones representativos de alguma informação. Uma versão de modo escuro também foi testada. Feitas estas experimentações ficou definido que a barra lateral de navegação teria a altura da página, os botões de ação teriam o preto ou branco, os cartões poderão ter, de preferência, cores vivas, tal como os ícones.

Uma vez efetuadas algumas explorações definiu-se um *design system* ou biblioteca de *design,* visível no Apêndice M – Design system e retirado do ficheiro desenvolvido na ferramenta *figma* [22], onde são definidas todas regras para construção dos componentes e neste também são visíveis os componentes primitivos que são utilizados ao longo do *design* do produto.

Assim é possível encontrar:

- As cores a utilizar na plataforma;
- Estilos de caixas;
- Tipografia;
- Regras de desenho dos botões (grandes, médios e pequenos);
- Tipos e diferentes variantes de botões;
- Regras para desenho dos *inputs;*
- Tipos e diferentes variantes de *inputs;*
- Tipos e diferentes variantes de *textArea*;
- Tipos e variantes de *selects;*
- Tipos e variantes de *checkboxes*, *radio buttons* e *switches;*
- Todos os ícones;
- Tipos de *tooltips* e *popovers;*
- Aparência dos itens da *sidebar;*
- Tipos de *toasts* e alertas;
- Calendário;
- Filtros;
- Conjunto de ícones com as iniciais dos participantes;
- *Design* predefinido para a listagem de quadros na página *dashboard* e dos *boards;*
- Temporizador;
- *Design* predefinido para a listagem de agendamentos;
- *Design* das colunas, cartões e comentários;
- *Design* dos cartões de estatística do *dashboard.*

5.7.Desenho do produto para desktop

A presente secção serve para demonstrar o *design* desenvolvido pela *designer* que deve ser replicado ao longo da implementação da plataforma, utilizando os componentes definidos na secção anterior. Todo o design do produto foi efetuado na ferramenta *figma* tal como na maioria das tarefas anteriores, contudo esta foi bastante útil para o desenvolvimento porque permite extrair com detalhe as propriedades de CSS que possibilitam reproduzir com quase 100% de precisão o *design* desenvolvido. É importante notar que o *design* implementado teve como foco o tamanho de um ecrã *desktop* (1440x1024) e assim sendo a aplicação não tem, numa primeira fase, a responsividade para ecrãs mais pequenos como um *smartphone.* Assim, para cada página mencionada no *Site map* foram desenvolvidos diversos ecrãs separados por papéis de utilizador onde se verificam todos os detalhes que devem ser visíveis para cada funcionalidade tendo em conta as respetivas permissões. Na Figura 19 é possível encontrar um exemplo do *design* criado para a página de um quadro e que foi implementada na aplicação web.

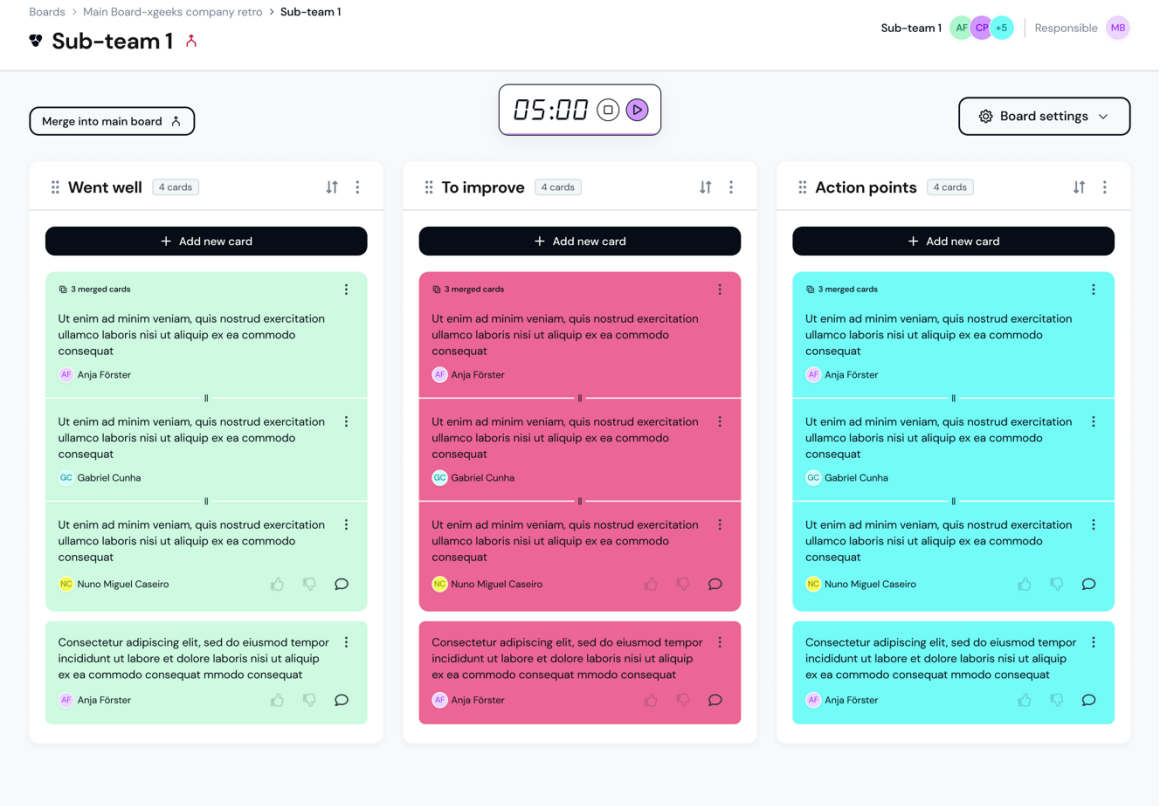


Figura 19 - Representação da página de um quadro

Assim, todas as funcionalidades, detalhes e especificidades das mesmas são demonstrados nos *designs* desenvolvidos. É possível observar no Apêndice N – Interface para desktop, com pouca ampliação, todos os *designs* desenvolvidos, extraídos do respetivo

ficheiro [23] e desenvolvidos na ferramenta *figma*. Na última referência é possível aceder e observar o *design* de todos os ecrãs e efetuar zoom sobre estes para compreender com pormenor como cada componente deve ser implementado. No próximo capítulo *6* são descritos os ecrãs e as respetivas funcionalidades tal como a sua implementação.

6. Implementação

Neste capítulo é apresentada a implementação do sistema composto por duas aplicações principais (*frontend* – aplicação web e *backend* - servidor). Na secção 6.1, é apresentada a arquitetura do sistema através da representação do nível 1 e 2 do modelo C4 [24]. Na secção 6.2 são apresentadas as tecnologias utilizadas ao longo do desenvolvimento do produto, com o intuito de proporcionar enquadramento teórico sobre as ferramentas e tecnologias utilizadas. Na secção 6.3 é apresentada a base de dados, as relações e as entidades criadas. Na secção 6.4 é apresentada e descrita a arquitetura do *backend*. Na secção 6.5 é apresentada a arquitetura do *frontend* e os detalhes mais relevantes da sua implementação. Na secção 6.6 são descritas as funcionalidades da aplicação, alguns detalhes da sua implementação e a sua representação visual na aplicação web. Na secção 6.7 é detalhada a infraestrutura que aloja as aplicações e também é apresentada a pipeline CI/CD.

6.1. Arquitetura de sistema

Nesta secção é apresentada a arquitetura do sistema através de dois diagramas baseados no modelo C4 e com diferentes níveis de granularidade. Assim, na Figura 20 é possível observar o nível 1 do modelo C4 onde se verificam três atores principais, o membro, o utilizador anónimo e o super-administrador. É importante referir que o membro é um utilizador normal da plataforma, pode interagir com esta, visualizar e gerir o seu conteúdo. O super-administrador pode fazer o mesmo que um membro, no entanto tem privilégios de gestão total da plataforma e por isso pode visualizar e gerir todo o conteúdo existente. O utilizador anónimo pode visualizar e interagir com um quadro, se este for público. O sistema SPLIT permite aos utilizadores gerir as suas equipas, criar retrospetivas e os quadros que auxiliam a sua execução.

Este sistema utiliza serviços externos para complementar as suas funcionalidades tal como a *Azure Active Directory* [25], que possibilita a autenticação no sistema SPLIT através do email da empresa. Caso o utilizador se registe com *username* e *password,* poderá em algum momento, necessitar de recuperar a sua *password,* como tal, o sistema SPLIT utiliza um serviço externo denominado de *SendGrid* [26]. Este permite enviar emails aos utilizadores com um *link* que pode ser utilizado para mudar a *password*. O terceiro serviço externo é aquele que permite enviar mensagens ou notificações a partir da API que o *slack* [27] disponibiliza. É possível de usufruir da API se existir uma *slack app* e se esta se

encontrar integrada num *workspace*, que no caso do SPLIT é o da xgeeks. A *slack app* ao receber as solicitações submetidas através da API, envia para os utilizadores os respetivos eventos como é detalhado nas próximas secções.

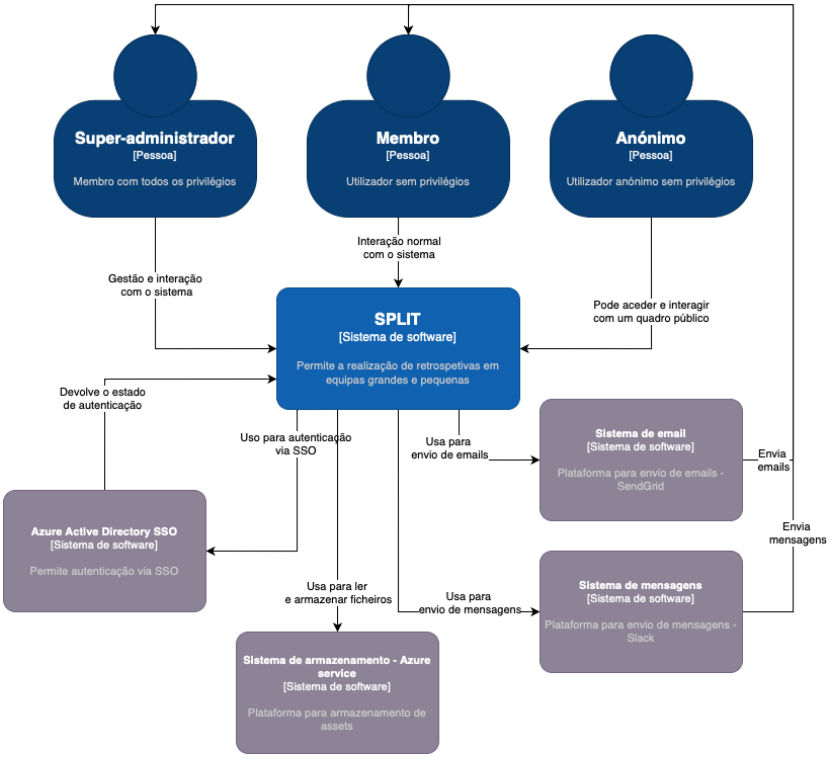


Figura 20 - Diagrama de arquitetura de sistema SPLIT - nível 1 do modelo C4

Na Figura 21 pode observar-se outro diagrama do sistema SPLIT que apresenta um nível superior de granularidade. Neste diagrama é possível observar os vários componentes (*containers*) que constituem o sistema, tal como, os papéis que os utilizadores podem ter dentro de uma equipa ou quadro. Assim, um utilizador pode interagir e visualizar a informação presente na plataforma, mas também pode participar numa equipa e/ou num quadro, podendo desempenhar papéis diferentes em cada um destes. Um utilizador se pertencer a uma equipa e se nesta for administrador ou *stakeholder* pode efetuar tarefas de gestão, nomeadamente, adicionar ou remover utilizadores, mas também definir o papel de cada um destes numa equipa. Como referido, a diferença do *stakeholder* para o *admin* reside no facto de estes não participarem num subquadro quando é criada uma retrospetiva dividida e, por isso, ambos têm privilégios de administração. Um utilizador registado ou anónimo se for participante de um quadro poderá ser responsável ou membro. O primeiro papel referido implica que o utilizador tenha privilégios dentro do quadro e isto significa que também pode geri-lo, sendo possível efetuar qualquer alteração no quadro, inclusive as suas configurações.

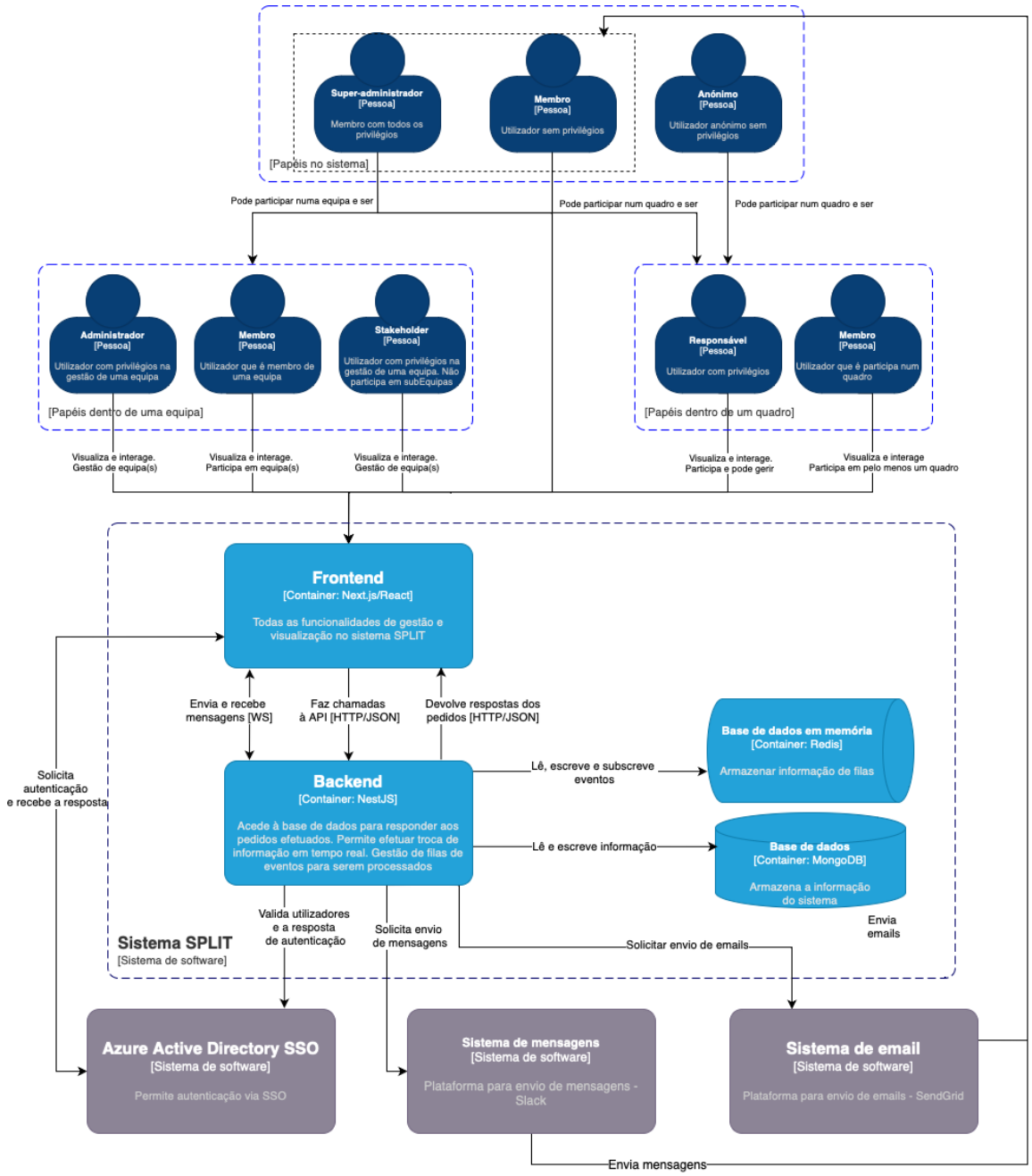


Figura 21 - Diagrama de arquitetura do sistema SPLIT - nível 2 do modelo C4

O sistema SPLIT, como é possível observar na Figura 21, é composto por vários módulos que comunicam entre si. Assim, o *backend*, desenvolvido com a *framework* NestJS, disponibiliza uma API REST e um *websocket gateway* que representam pontos de acesso a partir dos quais o *frontend* consegue interagir com o intuito de solicitar operações sobre os dados existentes na plataforma. A aplicação acede à **base de dados** - mongoDB - de forma a solicitar ou manipular os dados. Com o objetivo de efetuar tarefas assíncronas e em *background* para evitar constrangimentos de *performance* foi criada uma **base de dados em memória** - redis**.** Esta permite criar uma fila de processos/tarefas que são executados quando possível. Por isso, aquelas que demoram algum tempo a processar são anexados numa fila e

executadas, quando for possível, no *backend* por ordem de chegada. A aplicação do ***frontend**,* desenvolvida com a framework Next.js, comunica com o *backend* para troca de informação e é uma aplicação web (*single page application*) que disponibiliza uma interface gráfica e que possibilita a interação dos utilizadores com o sistema. A partir do sistema SPLIT, os utilizadores podem efetuar retrospetivas recorrendo a quadros *kanban* personalizáveis.

No que diz respeito à infraestrutura, é importante referir que todo o sistema se encontra disponibilizado na *cloud* através dos serviços da Microsoft *Azure* de forma duplicada, ou seja, todos os componentes são duplicados para cada ambiente, o de desenvolvimento e o de produção. Para correr localmente foi criado um ficheiro *docker-compose.yaml* e que ao ser executado promove a criação de vários *containers* em *docker* e cada um deles é responsável por executar uma das aplicações. No caso dos containers do *frontend* e *backend,* estes executam o *dockerfile* para instalação de dependências e execução da respetiva aplicação.

6.2. Tecnologias utilizadas

Esta secção tem o propósito de descrever as tecnologias e ferramentas utilizadas para o desenvolvimento do projeto.

6.2.1. Figma

A aplicação *figma* [28] é uma ferramenta web com funcionalidades colaborativas que permite o desenvolvimento do *design* de interfaces para aplicações de qualquer tipo de dispositivo. Através de ferramentas gráficas vetoriais, os *designers* conseguem criar desafiantes *layouts* otimizados para diferentes tamanhos de ecrã. Os utilizadores conseguem criar um sistema de *design* para as aplicações*,* declarando cores, tipografias, construindo componentes primitivos ou átomos que podem ser reutilizados facilmente em qualquer ecrã desenhado. Adicionalmente é possível incluir elementos interativos como *scroll*, funcionalidades de *hover* e animações, permitindo criar um *design* moderno, próximo da realidade e do produto final. Aliado a estes elementos é possível ainda visualizar o que foi desenvolvido através do modo *preview,* o que permite simular os dispositivos e tamanho dos mesmos, bem como a interação com estes. Esta funcionalidade com os elementos interativos referidos facilitam a obtenção de *feedback* antes da implementação. Outra das mais valias é a possibilidade de exportar algum código CSS, *kotlin* ou *swift* que facilita a implementação por parte dos *developers*, ajudando a replicar o *design* desenvolvido.

Sendo uma ferramenta colaborativa existe uma grande comunidade de *designers* e *developers* que produzem novos *plug-ins*, *templates* e *widgets* para consumo público, o que também facilita a integração e aprendizagem por parte de novos utilizadores.

No projeto SPLIT esta ferramenta foi utilizada para efetuar todo o processo de conceção do produto apresentado no capítulo anterior.

6.2.2. NestJS

A *framework* NestJS [29] foi desenvolvida para auxiliar a construção de forma eficiente e escalável aplicações *server-side,* baseadas em Node.js e permite com facilidade adotar o *domain-driven design* (DDD), *event sourcing*, *websockets* e arquitetura de microserviços. É referido na documentação que a linguagem suportada é *javascript* no entanto tem, também, total suporte para t*ypescript* e combina elementos de programação orientada a objetos, programação funcional e também programação reativa. Esta *framework* utiliza internamente e por defeito a *framework express*[17], no entanto também é possível configurá-la para utilizar *fastify*[18].

A *framework* apresenta uma estrutura modular que simplifica a divisão do projeto em blocos separados, sendo que, cada um, pode dizer respeito a uma determinada área de negócio e esta divisão facilita a integração de bibliotecas externas e a escalabilidade das aplicações.

Nesta ferramenta existem diversos conceitos estruturais tais como os controladores, *providers* e módulos, como pode ser observado na Figura 22. Os controladores são responsáveis por receber os pedidos dos clientes e depois de processados retornam as repostas. O mecanismo de *routing* faz a gestão de qual o controlador que deve receber cada um dos pedidos. Os *providers* são conceitos fundamentais da *framework* e qualquer classe básica pode ser considerado como tal, isto é, serviços, repositórios, *factories, helpers*, etc.

A principal ideia da ferramenta é possibilitar a injeção de classes como uma dependência, significando que estas podem criar relações entre si e a mesma instância pode ser usada noutras que necessitam do seu uso. Por de trás do pano as classes são instanciadas como *singletons*. Os serviços são, portanto, classes injetáveis e responsáveis por tratar da lógica de negócio. Os módulos são classes que fornecem os metadados necessários para

[17] https://expressjs.com
[18] https://www.fastify.io

injeção de *providers* e controladores na aplicação NestJS e também promovem a organização dos componentes, uma vez que cada módulo diz respeito a uma área de negócio e "agrupam" as classes correspondentes.

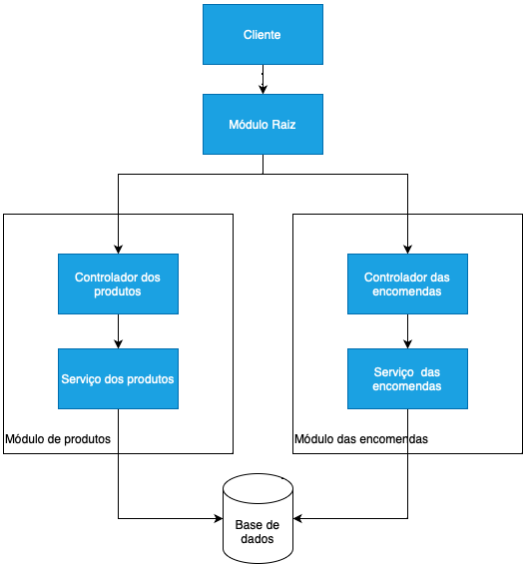


Figura 22 - Arquitetura base da framework NestJS

A framework permite a abstração de certos detalhes de implementação, pelo que, permite facilmente incluir ferramentas de *websockets*. Assim, dispõe de outro tipo de *provider* denominado de *gateway* que suporta a utilização de *packages* de *websockets* tal como o socket.io[19] e ws[20].

Uma fila no âmbito de software é um padrão de desenho que pretende ajudar a lidar com desafios de escala e performance que as aplicações enfrentam. Assim, estas ajudam a resolver problemas de picos de performance, uma vez que as tarefas podem ser colocadas numa fila e o processamento destas é realizado de forma assíncrona e de forma ordenada, tendo como critério a ordem de inserção na fila. Face a esta mais-valia, a framework NestJS providencia uma biblioteca, denominada de *bull*[21], que abstrai a implementação de um sistema baseado em filas e como único requisito é necessário instalar uma instância de Redis[22] que serve para armazenar a informação das filas. Seguidamente, a *framework* em conjunto com a biblioteca tratam da gestão das tarefas e execução das mesmas.

[19] https://github.com/socketio/socket.io
[20] https://github.com/websockets/ws
[21] https://github.com/nestjs/bull
[22] https://redis.io/

A *framework* permite a integração do padrão *event-based* através da biblioteca *event emitter*[23], o padrão facilita a comunicação entre serviços sem a necessidade de injeção das classes em módulos distintos. Esta *package* providencia a implementação de *observers* que permitem a subscrição e leitura de eventos que podem ocorrer na aplicação. Portanto, os eventos permitem desacoplar certos componentes da aplicação e despoletar ações noutras classes.

No projeto SPLIT esta foi a framework escolhida para implementação do *backend.* Desta forma, foi desenvolvida uma REST API e um *gateway* para comunicação através de *websockets* recorrendo à biblioteca *socket.io*. Esta permite que o *frontend* solicite informação, no entanto o *backend* também pode enviar informação se assim for necessário através da comunicação bidirecional que os *websockets* providenciam. Adicionalmente foi implementada uma *queue* com auxílio da base de dados *Redis*, onde são inseridas as operações para tarefas que necessitam maior desempenho, podem ser realizadas de forma assíncrona e de forma ordenada como por exemplo a criação de canais do *slack*, adição dos membros aos mesmos e envio das devidas mensagens. Alguns serviços podem comunicar entre eles através do mecanismo de eventos, não sendo necessária a injeção direta das classes nos serviços que pretendem tirar partido de outros. O padrão baseado em eventos foi utilizado com pouca exaustividade, no entanto aplicou-se na implementação e sincronismo do timer disponibilizado nos quadros e também na atualização das fases dos mesmos.

6.2.3. React

React é uma biblioteca *open-source* em *javascript* e desenvolvida pelo Facebook. Esta é declarativa, eficiente, flexível e permite construir interfaces através de pequenas peças chamadas componentes que, em conjunto, podem formar uma interface complexa. Para tal é importante compreender quatro conceitos fundamentais, os componentes, propriedades, estados e renderização com JSX.

JSX é uma extensão de *javascript* que permite utilizar elementos HTML no código. A Figura 23 demonstra como o código JSX é transformado em javascript. É possível verificar que o componente inserido com a *tag* <*MyButton*> é transformado em código *javascript* incluindo as propriedades definidas no elemento.

[23] https://github.com/nestjs/event-emitter

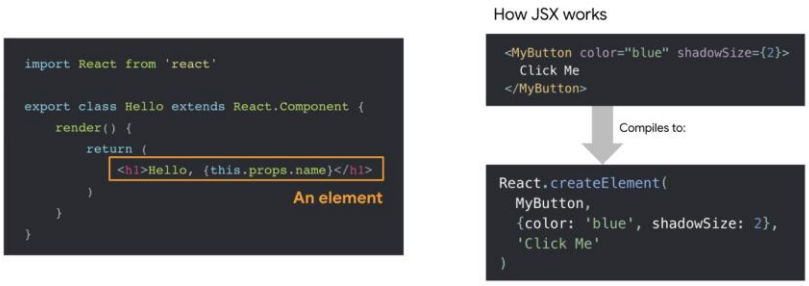


Figura 23 – Exemplo de JSX [30]

Os componentes em *react* são blocos de código que podem ser reutilizados em toda a aplicação, são representados por uma função em *javascript* que podem receber propriedades e devolvem elementos de *react* que descrevem o que deve ser visualizado no ecrã. Um conjunto de componentes forma uma página. Na Figura 24 é possível observar um exemplo de componente em *javascript* e *typescript.* Assim, é definida uma interface para o tipo de propriedades que o componente pode receber. Como tal é passado para a função o parâmetro nome e este pode ser incluído na renderização componente.

```
1. type IntroProps = {
2. name: string;
3. }
4. const Intro = ({name}: IntroProps) => {
5.        return <h1>Hello, my name is {name}</h1>;
6. }
7.
```

Figura 24 - Representação de código para um componente em react

Após verificar o bloco apresentado na Figura 24, pode referir-se que as propriedades são os argumentos que são passados para dentro de um componente/função e essa informação pode ser utilizada no componente.

Outro dos conceitos de *react* são os *hooks* que dizem respeito a funções *javascript* que podem ser reutilizadas em toda a aplicação, tendo como objetivo o de gerir informação ou efetuar operações. Um dos hooks é o *useState* que permite aceder e declarar um estado – um estado é como uma variável, mas que tem em conta o ciclo de vida do componente, isto é, não perde o valor se voltar a ser renderizado. Por outras palavras, os *states*, em *React,* são objetos que armazenam informação e que pode mudar ao longo da vida do componente e ao serem alterados promovem uma nova renderização, o que promove a atualização dos elementos visuais com a informação alterada.

No projeto SPLIT esta framework foi utilizada em conjunto com a próxima ferramenta apresentada e denominada de Next.js para a implementação do *frontend*.

6.2.4. Next.js

A *framework* Next.js [31], criada pela Vercel[24], é uma *framework* em Node.js que permite construir aplicações em *react*, escaláveis e com boa performance sem necessidade de configurações adicionais. É compatível com *typescript* e comparativamente com *react* trata de fornecer ferramentas e configurações importantes que teriam de ser tratadas pelo *developer*. A ferramenta fornece também um conjunto adicional de funcionalidades, nomeadamente a capacidade de ter *server-side app* facilmente acessível, programável e um conjunto de otimizações para a aplicação em desenvolvimento.

Assim, a *framework* apresenta o conceito de *pre-rendering,* que gera o HTML para cada página em adiantado, do lado do servidor, sempre que há um pedido, e por isso não é feito no lado do cliente. Tradicionalmente, o HTML é gerado do lado do cliente através do código *javascript* enviado. O Next.js assegura que o código *javascript* necessário para tornar a página totalmente interativa vai agarrado ao HTML gerado. O código *javascript* é corrido assim que a página é carregada. Este processo é chamado de *hydration.*

A *framework* suporta vários tipos de *rendering* tais como *server-side rendering* (SSR), *static generation* e *client-side rendering.* É também possível misturar os vários tipos numa aplicação, definindo um para cada página.

No *server-side rendering,* cada vez que o utilizador pede uma página, o servidor Next.js prepara-a efetuando os pedidos necessários à base de dados, construindo o HTML, anexando o código js e depois envia-o para o cliente. O browser posteriormente utiliza o HTML gerado, executa o restante código *javascript* e apresenta a página. Contudo, poderá não ser o ideal se o pedido da página for muito pesado, porque implica que o utilizador fique à espera da resposta sem visualizar qualquer conteúdo.

Com o *static generation,* o HTML é gerado em *build time*, isto é, quando o servidor Next.js efetua o *build*. Assim, as páginas estão prontas desde esse momento, são guardadas em *cache* e o HTML pode ser reutilizado quando os clientes solicitarem a mesma página, não existindo a necessidade de a preparar novamente, o que otimiza a performance, porque não é necessário processamento do lado do servidor em cada pedido. A desvantagem deve-se ao facto de não ser possível incluir pedidos dinâmicos, isto é, o conteúdo deve ser estático e os pedidos a realizar são durante o *build time.*

[24] https://vercel.com

O *client-side rendering* é útil quando o conteúdo das páginas precisa de atualização frequente e por isso as páginas são renderizadas apenas do lado do cliente.

Para definir qual o tipo de renderização para cada página é possível utilizar os seguintes métodos: *getServerSideProps*() ou *getStaticProps*(). O primeiro indica que a página deve ser renderizada no servidor em cada acesso à página e este prepara-a efetuando os pedidos necessários para aquisição dos dados. O segundo método indica que a página deve ser renderizada no servidor, mas apenas em *build time.*

Uma mais-valia em comparação com o *react* é o suporte de rotas a partir da pasta *pages.* Isto significa que qualquer ficheiro .js ou .tsx nessa pasta ou em subpastas corresponde a uma página e o caminho do ficheiro representa a rota correspondente. Adicionalmente também suporta a criação de rotas dinâmicas, ou seja, a rota da página é determinada pelo valor do parâmetro. Assim, um ficheiro na pasta "/pages/products/[id].js" terá a rota correspondente em "/products/" e através do mecanismo dinâmico é possível substituir o id por um valor e então a rota "/products/1234" apresenta a página do produto 1234.

Para finalizar é importante referir que a *framework* suporta divisão de código e isto é, o processo de dividir o *bundle* da aplicação em partes menores quando se efetua o *build* da mesma. A separação por norma faz-se por pontos de entrada na aplicação, assim ao aceder a uma página não é necessário código de outras. O *bundle* é o processo de resolver, agrupar todas as dependências e ficheiros em pacotes otimizados para o *browser,* com o intuito de reduzir o número de pedidos por ficheiros quando o utilizador visita uma página. Tendo isto em conta, a divisão de código ao separar o *bundle* em pacotes mais pequenos tem o objetivo de melhorar a performance da aplicação no carregamento inicial das páginas. Quando o utilizador navega na aplicação, os pacotes necessários e ainda não descarregados são recebidos e reutilizados se necessário.

No projeto SPLIT esta framework foi utilizada para a implementação do *frontend* e através da qual foi possível implementar o *design system* e o *design* da aplicação, oferecendo uma excelente experiência de utilização.

6.2.5. TypeScript

Typescript [32] é uma linguagem de programação construída sob *javascript* que inclui funcionalidades adicionais e que a complementam. Assim, fornece a possibilidade de atribuir tipos aos objetos, funções ou variáveis o que permite minimizar os erros durante o

desenvolvimento. O código *typescript* ao ser compilado é transformado em *javascript* através do *typescript compiler*.

No projeto SPLIT esta linguagem é utilizada em ambas as aplicações e todos os objetos, argumentos de funções, resultado das funções, as classes e as variáveis apresentam um tipo definido.

6.2.6. MongoDB

MongoDB [33] é uma base de dados *open-source*, não relacional e orientada a documentos. Assim, os dados são armazenados em objetos *javascript object notation* (JSON), mais concretamente *Binary* JSON (BSON), e por isso existe maior flexibilidade e facilidade quando se lida com estruturas de dados mais complexas. Portanto, uma base de dados *mongo* pode ter diversas *collections* e estas são compostas por documentos e subdocumentos. Esta suporta a criação de índices, *queries* complexas, bem como transações entre múltiplas coleções. Para efetuar transações é necessária uma réplica *set*.

Este tipo de base de dados por norma é utilizado em *big data* e em aplicações web que necessitam de atualizações em tempo real. É também popular devido à sua escalabilidade horizontal, o que permite lidar com elevadas quantidades de dados e muitos acessos.

Com o intuito de simplificar a modelação, acesso e escrita dos dados numa aplicação, a biblioteca *mongoose* é um *object document mapper* (ODM) que providencia uma solução para definir coleções numa aplicação em Node.js através de esquemas e facilita a criação de *queries.* Portanto, através de esquemas é possível definir os documentos numa coleção, as suas propriedades, os respetivos tipos e fornece uma forma acessível de interação com as coleções.

No projeto SPLIT esta é a base de dados eleita para suportar o projeto, devido à elevada *performance* adjacente a este tipo de base de dados, permitindo bons resultados na atualização dos dados em tempo real.

6.2.7. Redis

Redis (*remote dictionary server*) [34] é uma base de dados *open-source* que armazena os dados em memória. É frequentemente utilizada como mecanismo de cache, *message broker* ou *streaming* de dados em tempo real, ou seja, enviar de um local para outro uma grande e continua quantidade de informação. O Redis suporta uma grande variedade de

estruturas como *strings, hashes,* listas, *sets* e suporta avançadas funcionalidades como o sistema de mensagens *publish* e *subscribe*. Esta ferramenta é conhecida pela sua performance e por isso é usada em sistemas que necessitam de alta performance.

Assim, o Redis pode ser utilizado como um *message broker* para implementar filas de mensagens. Numa fila, a parte esquerda da mesma representa a cabeça e a parte da direita a cauda. Portanto as novas mensagens podem ser adicionadas do lado direito e removidas do lado esquerdo. Adicionalmente existem bibliotecas e *frameworks* como a *bull* que abstraem a implementação de filas em Redis e por isso facilita a utilização deste mecanismo.

No domínio das filas há três conceitos importantes, como se pode observar na Figura 25, os *producers* – que são *providers* responsáveis por adicionar *jobs*/tarefas às filas, os *consumers* – são as classes que executam o processamento, e por fim os *listeners* – são as classes que estão à escuta, observando o estado do trabalho como por exemplo, perceber quando é iniciada uma determinada tarefa, quando está a decorrer ou quando é finalizada. Os *consumers,* por norma, também são *listeners.* Para cada fila, existe um *producer* e o respetivo *consumer.*

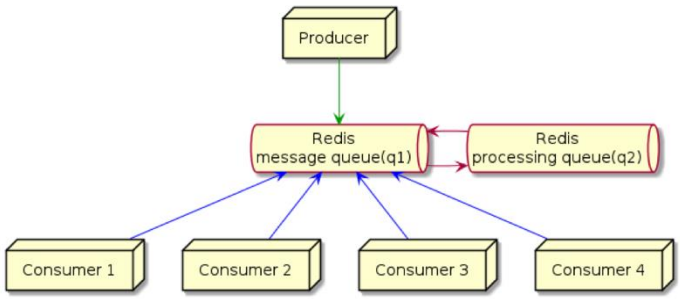


Figura 25 - Representação do funcionamento de uma fila utilizando redis

No projeto SPLIT, o Redis foi utilizado em conjunto com a biblioteca *bull* na *framework* NestJS para tirar proveito das filas, de forma a otimizar a performance do servidor em tarefas que requerem algum poder computacional.

6.2.8. Swagger

Swagger [35] é uma framework para descrever, consumir e visualizar uma API REST. É providenciada uma simples e poderosa representação dos recursos presentes na API e as operações disponíveis. Desta forma, os *endpoins* são apresentados, bem como os parâmetros e os respetivos tipos e também é possível verificar o tipo de resposta que é devolvida. Adicionalmente, se existir autenticação é possível providenciar um *bearer token* que pode ser utilizado para todos os pedidos.

No projeto SPLIT, esta *framework* foi utilizada para descrever a API bem como facilitar a sua exploração por parte de novos *developers.*

6.2.9. Github actions

As *Github Actions* [36] permitem que os *developers* automatizem certos trabalhos, através de scripts construídos e que são despoletados por certos eventos no *github*. Estes *scripts* são chamados de ações e podem ser escritos em várias linguagens. Podem ser despoletados quando existe um *push* para um repositório, na criação de um *pull request*, abrindo ou fechando *issues,* mas também quando se dá *merge* de um *branch* para outro. É possível automatizar os testes e o *building* das aplicações, efetuar o *deploy* para diferentes ambientes, automatizar as *releases* e enviar notificações.

O *github* providencia alguns scripts pré-construídos que podem ser utilizados para automatizar tarefas frequentes, como correr testes e dar *deploy* para várias plataformas como *Amazon Web Services* (AWS), *Azure* e *Google cloud*. No SPLIT as *actions* foram criadas e utilizadas para construir a *pipeline* de CI/CD

6.2.10. Sendgrid

SendGrid [37] é um serviço de email que permite enviar emails através da sua plataforma na *cloud.* Este providencia uma API que permite enviar e receber emails usando *webhooks*, SMTP ou HTTP. Também são fornecidas funcionalidades como análise de dados, email de teste e ferramentas para desenhar emails.

No projeto SPLIT, embora pouco utilizado uma vez que a autenticação é realizada através do email da empresa, este foi o serviço utilizado para testar a funcionalidade de envio de emails para recuperação de *passwords* e foi utilizado o plano grátis que apresenta a limitação de 100 emails por dia.

6.2.11. Slack e slack API

O *slack* é uma plataforma de comunicação que providencia um único local para troca de mensagens, ferramentas e ficheiros. Esta ajuda a manter as equipas organizadas, evita a sobrecarga de emails e melhora a comunicação entre os membros da equipa ou empresa. O *slack* permite integrações com inúmeras aplicações e serviços, o que permite que esta aplicação seja o centro de muita informação.

A *slack* API disponibilizada providencia um conjunto de interfaces que permite aos *developers* criar aplicações que podem interagir com a plataforma *slack*. Esta plataforma é o principal canal de comunicação da xgeeks e foi integrada no SPLIT. Portanto o *backend* tira partido da *slack api* para enviar mensagens para o *workspace* da empresa.

6.2.12. Azure cloud services

A *azure cloud services* [38] é uma plataforma de computação na nuvem, criada pela Microsoft, que providencia serviços de *cloud* aos utilizadores. Esta permite criar, disponibilizar, gerir aplicações e serviços, obtendo os benefícios de escalabilidade e poupança nos custos. Nesta plataforma são disponibilizados *clusters, container registries*, máquinas virtuais, base de dados, serviços de *machine learning*, de segurança, etc.

Neste projeto foram criados dois ambientes, um de desenvolvimento e outro de produção. Assim, foram criados quatros serviços aplicacionais concretamente os *azure app services*[25], um para cada aplicação em cada um dos ambientes. As aplicações correm nos serviços através de imagens de *Docker* que são armazenadas num *container registry*[26] criado na plataforma para o efeito. Foi utilizado o serviço de base de dados CosmosDB[27], no qual foram criadas duas instâncias de mongo, uma para cada ambiente. Adicionalmente, criaram-se duas instâncias da base dados *redis* e também um *storage container* para guardar ficheiros de imagens.

6.2.13. Stitches

Stitches [39], é uma biblioteca de estilização de componentes para *react* com bastante performance, na medida em que foi desenvolvida com esse foco. Esta permite facilmente criar *design systems,* ou seja, é possível criar componentes com variantes de estilos e que podem ser controlados através da respetiva propriedade e associada ao componente, esta ao ser alterada, afeta o visual do componente. Por exemplo, é possível definir um botão com uma variante de sucesso e este deve apresentar o *background* verde e o texto a preto, ou outra, de perigo que coloca a cor do *background* a vermelho e a cor do texto a branco. Assim, o *developer* apenas tem de reutilizar o componente e selecionar a variante que pretende. É também possível definir um conjunto de variáveis (*tokens*) que podem ser reutilizadas ao

[25] https://azure.microsoft.com/en-us/products/app-service/
[26] https://azure.microsoft.com/en-us/products/container-registry
[27] https://learn.microsoft.com/en-us/azure/cosmos-db/introduction

longo da aplicação e em todos os componentes, por exemplo é possível criar um objeto "sizes" e definir que "large" corresponde a X pixéis, "medium" a Y pixéis, etc.

Para melhor compreensão das variantes e do valor que a biblioteca oferece para o *developer* é disponibilizado o bloco de código seguinte.

```
1. const Button = styled('button', {
 2.   // base styles
 3.
 4.   variants: {
 5.     color: {
 6.       violet: {
 7.         backgroundColor: 'blueviolet',
 8.         color: 'white',
 9.         '&:hover': {
10.           backgroundColor: 'darkviolet',
11.         },
12.       },
13.       gray: {
14.         backgroundColor: 'gainsboro',
15.         '&:hover': {
16.           backgroundColor: 'lightgray',
17.         },
18.       },
19.     },
20.   },
21. });
22.
23. () => <Button color="violet">Button</Button>;
24.
```

Figura 26 - Representação do código JS e CSS para as variantes de um botão

Na Figura 26 pode-se observar que para a variante "color" existem dois valores, o "violet" e o "gray" e em cada um destes é definida a cor de fundo, do texto e a cor de fundo quando se faz *hover* sobre o componente. Na linha 23 é apresentado de que forma se pode utilizar a variante "color", e ao escolher um dos valores todas as propriedades de CSS definidas são aplicadas.

No SPLIT esta foi utilizada para a estilização de todos os componentes e implementação do *design system.*

6.2.14. Radix

Radix [40] é uma biblioteca de componentes de interface para construir aplicações e *design systems* de alta qualidade. Os componentes são focados na acessibilidade - seguindo os padrões de design da WAI-ARIA[28] - na customização, experiência de desenvolvimento e

[28] https://www.w3.org/TR/wai-aria-practices-1.2

performance. Estes não apresentam estilos, por isso o *developer* pode definir a aparência que desejar.

No projeto SPLIT foi utilizada como fonte de conhecimento para instalação e utilização de componentes complexos.

6.2.15. Nextauth

Nextauth [41] é uma biblioteca *open-source*, muito completa que serve de solução para a gestão total da autenticação numa aplicação em Next.js. Esta suporta a autenticação por email e password, mas também inúmeros *providers,* fornecendo boa segurança.

No projeto SPLIT esta é a ferramenta para total gestão da autenticação no *frontend*.

6.2.16. Jest

Jest [42] é uma biblioteca de *javascript* desenvolvida e mantida pelo Facebook e providencia uma simples forma de testar código *javascript,* sendo possível desenvolver testes unitários e de integração. A configuração da mesma é de baixa dificuldade e permite fazer *mock* do que for necessário para realizar os testes.

No projeto SPLIT é a ferramenta elegida para criação de testes tanto no *backend* como no *frontend.*

6.2.17. React hook form

React Hook Form [43] é uma biblioteca para *react* que providencia uma forma de lidar e validar a informação introduzida nos formulários da aplicação. O package providencia *hooks* que permitem gerir o estado do formulário e oferecer uma simples e intuitiva forma de validar os inputs. Para cada *input* é apenas necessário definir as regras e a biblioteca trata de realizar as verificações necessárias e disponibilizar os erros se for o caso.

No projeto SPLIT é a ferramenta utilizada para suportar a validação de todos os *inputs* dos formulários.

6.2.18. React beautiful dnd

React beautiful dnd [44] é uma biblioteca para *react* produzida pela atlassian[29] que providencia uma forma acessível de implementar a funcionalidade de *drag and drop* em

[29] https://www.atlassian.com/

aplicações. É desenhada para ser flexível e fácil de usar, enquanto disponibiliza um alto nível de customização e funcionalidades avançadas. Esta utiliza as últimas tecnologias web como HTML5 *Drag and Drop* API, para providenciar interações de *drag and drop* suaves e com elevada performance.

Esta ferramenta foi utilizada no projeto SPLIT para implementação das funcionalidades de *drag and drop* na página dos quadros.

6.2.19. React query

React query [45] é uma biblioteca para *react* que providencia uma forma de gerir e efetuar pedidos HTTP numa aplicação. É desenhada para simplificar o processo de *fetching*, *caching* e atualização da informação nos componentes de *react.*

A biblioteca ao efetuar os pedidos, armazena os dados das respostas em *cache,* associados a uma chave e os componentes que utilizam essa informação atualizam-se conforme os dados recebidos. É também providenciado suporte para *optimistic updates*, pedidos com paginação, permite lidar facilmente com erros, *retries* e por estes motivos permite construir aplicações robustas com confiança. Alem de ser possível efetuar pedidos HTTP também possibilita a utilização de GraphQL.

No projeto SPLIT esta biblioteca foi utilizada para suportar a solicitação de pedidos HTTP e gestão de cache dos dados obtidos.

6.2.20. Storybook

Storybook [46] é uma ferramenta que auxilia a construção, documentação e visualização de componentes de interface de forma isolada. Pode considerar-se que o *storybook* permite reunir e apresentar os componentes que uma aplicação tem implementado e como estes podem ser utilizados. Os *developers* para cada componente podem criar uma "story" que apresenta o componente com os seus diferentes estados e variações. Portanto, em cada "story" é possível visualizar e alterar os estados ou variações existentes no componente e observar de imediato como este se comporta visualmente.

No projeto SPLIT o *storybook* serviu para documentar os componentes criados.

6.3. Base de dados

No que diz respeito à base de dados optou-se por uma base de dados não relacional (no-sql) tendo em conta as características do projeto e aquelas adjacentes a este tipo de base de dados: por não se tratar de informação crítica, não é necessário garantir que as transações sejam 100% confiáveis, ou seja, garantir a atomicidade, consistência, isolamento e durabilidade; não existe a necessidade de fazer *queries* muito complexas à base de dados; é possível definir documentos dentro de documentos o que minimiza o número de relações necessárias estabelecer dentro de um quadro, aumentando assim a performance – significa que um quadro poderá ter todos os objetos (de entidades diferentes) que necessita dentro do mesmo, evitando ou minimizando as relações, e por consequência não é necessário efetuar a população (*populate*) das relações quando se efetuam pedidos à base de dados (BD); Uma vez que um documento pode conter *arrays* de outras entidades, a ordenação é preservada, o que é um ponto a favor comparando com *sql* que implicaria a construção de uma coluna extra para estabelecer a ordem dos elementos quando devolvidos – no SPLIT certos elementos devem vir ordenados por posição no quadro como é o caso dos cartões nas colunas. Uma vez que este tipo de base de dados tem elevada performance [47] para consultas pouco complexas e acessos frequentes, foi considerado um aspeto chave porque poderão existir atualizações muito frequentes sobre os dados dos quadros, e existe a necessidade de propagar essa informação em tempo real.

Assim, determinou-se que o MongoDB é a base de dados a utilizar no projeto. Para desenvolvimento local, o *developer* deve ter duas instâncias de *mongoDB* a correr através de *docker* e deve definir uma *replica set* para conseguir utilizar transações em coleções ou entre coleções. Esta última significa que, ao efetuar uma operação de atualização num documento de uma coleção, e esta é dependente do sucesso de outra operação, numa segunda coleção, é possível efetuar o *commit* se ambas forem bem-sucedidas, ou abortar a transação se alguma falhar.

Uma vez definidos os requisitos no capítulo anterior, foi necessário definir as entidades ou *schemas* e estruturar possíveis relações entre elas, como tal, desenvolveu-se o diagrama apresentado na Figura 27. Através da biblioteca *mongoose* foi possível mapear, no *backend,* o diagrama efetuado e estabelecer relações entre documentos através da chave estrangeira. Como referido a base de dados é não relacional, no entanto o *mongoose* permite estabelecer relações entre entidades.

Portanto definiu-se a entidade *user, board, board_user, column, card, card_item, comment, schedules, resetPassword, team* e *team_user*. Verificando a Figura 27, é possivel constatar que algumas entidades apresentam o título com cor de fundo e esta representação pretende indicar quais são as entidades que representam coleções. Enquanto aquelas que não têm cor de fundo são entidades inseridas dentro de outras. Para representar no diagrama que uma entidade se encontra contida noutra, utilizou-se a seta simples enquanto a outra seta pretende representar uma relação entre documentos.

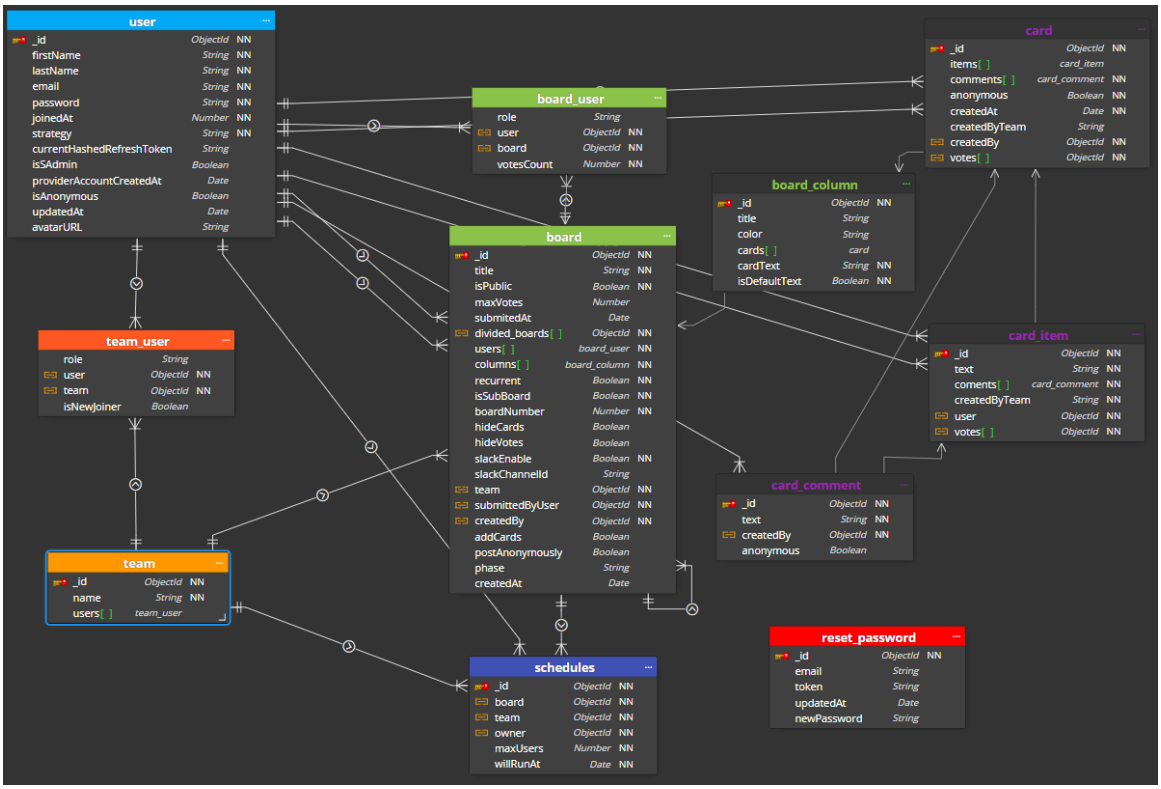


Figura 27 - Diagrama representativo da base de dados

Analisando a figura, verifica-se que um utilizador pode pertencer a uma equipa ou a um quadro e como tal estas entidades podem ter associados vários utilizadores. As entidades *team_user* e *board_user* guardam a chave estrangeira da equipa ou quadro, respetivamente, e a chave do utilizador, sendo assim possivel simular uma tabela *pivot* de uma base de dados relacional numa relação de muitos para muitos *user-board* ou *user-team*. O *array* de *users* das equipas ou dos quadros é uma propriedade virtual que é adicionada quando se efetua o pedido aos dados.

A coleção *board* é constituída por vários documentos que representam os quadros. Cada documento tem um *array* de colunas, estas têm um *array* de cartões e estes um conjunto de itens, comentários e votos. Os votos são representados por um *array* de ids de utilizadores.

Um quadro pode ter associado outros quadros e como tal o *array* de *divided_boards* guarda os ids dos subquadros. A um quadro pode ser associada uma equipa quando se pretende que os participantes do quadro sejam os membros da equipa.

No que diz respeito aos cartões de um quadro, é possivel agrupá-los e separá-los. Com o intuito de evitar perda de dados, surge o conceito de *card_item.* Portanto, um cartão tem sempre um *card_item* quando é criado e todos os novos comentários e votos são adicionados ao item. Caso um cartão seja adicionado a outro, no cartão destino é adicionado o *card_item* do cartão origem, formando assim um grupo (2 *card_items*). Quando é formado um grupo e adicionados novos comentários e votos, estes são adicionados nas respetivas propriedades do grupo. Se o grupo for desfeito, o *card_item* que restar ficará com os dados que foram anteriormente adicionados ao grupo. Quando um cartão é removido do grupo, o *card_item* é extraido do mesmo e é criado um *card* com esse *card_item* e assim mantém-se o registo dos dados.

No que diz respeito aos cartões de um quadro, é possivel agrupá-los e separá-los. Com o intuito de evitar perda de dados, surge o conceito de *card_item.* Portanto, um cartão tem sempre um *card_item* quando é criado e todos os novos comentários e votos são adicionados ao item. Caso um cartão seja adicionado a outro, no cartão destino é adicionado o *card_item* do cartão origem, formando assim um grupo (2 *card_items*). Quando é formado um grupo e adicionados novos comentários e votos, estes são adicionados nas respetivas propriedades do grupo. Se o grupo for desfeito, o *card_item* que restar ficará com os dados que foram anteriormente adicionados ao grupo. Quando um cartão é removido do grupo, o *card_item* é extraido do mesmo e é criado um *card* com esse *card_item* e assim mantém-se o registo dos dados.

Na coleção de *schedules* é guardada a informação de agentamentos. Portanto, quando se cria uma retrospetiva dividida é gerado um quadro, os subquadros e o respetivo *cron job* para despoletar o processo numa determinada data. Nos documentos *schedule* é armazenando o quadro origem, a equipa associada, o máximo de utilizadores que uma subequipa poderá ter e a data de quando o novo *job* irá correr.

Para finalizar, a coleção *reset_password* tem o propósito de armazenar a informação do *token* que permite alterar a password de um utilizador e a respetiva data de expiração.

Como é possível verificar, existem diversas relações entre as entidades e por isso a utilização de uma base de dados não relacional poderá não ter sido a melhor abordagem, uma vez que obriga a simular relações. A escolha deste tipo de BD deveu-se principalmente ao facto de que, quando se efetua o pedido do quadro, o respetivo documento devolve um único objeto sem necessidade de popular relações, contudo dificulta algumas operações de atualização dos dados. Outro dos motivos foi a pouca discussão na definição da melhor abordagem para a escolha da base de dados entre a equipa de desenvolvimento e outros colaboradores seniores. Alguns destes colaboradores sugeriram MongoDB e não existiu oposição nesta decisão. Como trabalho futuro poder-se-á efetuar a migração para uma relacional e otimizar as relações entre as entidades.

6.4. Backend

A presente secção tem o propósito de expor alguns conceitos arquiteturais utilizados no desenvolvimento da aplicação de *backend,* bem como descrever a sua implementação, tendo como foco os requisitos definidos no capítulo anterior e a experiência definida na secção 5.4.

6.4.1. Conceitos e padrões

Antes de concretizar a arquitetura implementada para esta aplicação é importante referir que o *backend* seguiu alguns princípios e padrões de desenho tal como a *Clean Arquitecture, Domain-driven Design* e SOLID.

A *clean arquitecture,* segundo o autor do livro *Clean Architecture: A Craftsman's Guide to Software Structure and Design* [48], tem como principal objetivo fornecer aos *developers* uma proposta de organização do código, separando as responsabilidades e dividindo o software em camadas. As camadas são interligadas através de interfaces e a regra principal da arquitetura define que as camadas interiores não conhecem as exteriores e estas apenas conhecem o que é disponibilizado nas interfaces.

Como se pode observar na Figura 28, a camada interior diz respeito às **entidades** e estas compreendem-se por objetos e métodos ou estruturas de dados. Esta camada não deve sofrer alterações se forem efetuadas modificações noutras camadas externas. A camada dos **casos de uso** contém a lógica de negócio da aplicação. Esta é responsável por aceder ou manipular os dados da camada mais interna. A camada seguinte e denominada de ***interface adapters*** refere-se a um conjunto de adaptadores que convertem a informação proveniente dos casos de uso para o formato mais conveniente, de forma a disponibilizar para o exterior da

aplicação. Inversamente, esta camada é responsável por transformar e receber os dados do exterior para o formato mais conveniente para os casos de uso e entidades. Os controladores e *gateways* são exemplos de adaptadores presentes nesta camada. A camada mais externa, a camada de **infraestrutura**, diz respeito a todas as ferramentas e bibliotecas que são usadas para construir a lógica de negócio como por exemplo o *object-relational mapping* (ORM), *object data modeling* (ODM), serviço de emails, base de dados e web *frameworks*.

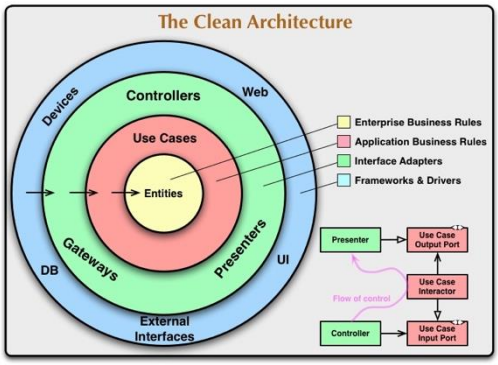


Figura 28 - Diagrama representativo das camadas *clean architecture* [48]

Os princípios SOLID [49] têm elevada sinergia com a arquitetura descrita anteriormente. Desta forma, são cinco os princípios subjacentes e cada um é representado por uma letra do acrónimo. Assim, o princípio ***s**ingle responsability* indica que cada classe só deve existir para implementar uma funcionalidade. O princípio ***o**pen/close* indica que uma classe deve ser aberta para extensão, mas fechada para alteração, ou seja, é possível adicionar funcionalidades, estendendo a classe, evitando modificações no ficheiro original. O princípio de substituição de **L**iskov indica que uma classe estendida de outra pode ser substituída pela sua classe base, ou seja, se uma classe S herda de uma classe T, então a classe S deve-se comportar igual à classe T. O princípio da segregação de **i**nterfaces indica que as interfaces devem apenas conter as declarações necessárias. O princípio da inversão de **d**ependências indica que se deve depender de abstrações e não de implementações, ou seja, as classes devem depender de interfaces ou de classes abstratas em vez de classes concretas.

O *domain-driven design* representa um conjunto de práticas e princípios que podem ser aplicados no desenvolvimento de software, tendo como foco a compreensão e modelação do domínio de negócio. O domínio é a área em que o *software* incide e pretende contribuir para solucionar problemas. É uma boa prática a separação de responsabilidades e devem ser isoladas de outras de forma a contribuir para uma boa manutenção do produto. Assim, um

domínio pode ser separado em subdomínios e na prática pode ser representado através de pastas e subpastas que representam pequenas peças de software.

A abordagem anteriormente referida sugere a utilização de alguns padrões como os repositórios e eventos. O primeiro permite separar a camada de acesso aos dados da camada de regras de negócio de uma aplicação. Desta forma, toda a lógica relacionada com o acesso aos dados é encapsulada e, por isso, ao alterar a lógica de negócio não implica alterações no código de acesso aos dados.

O padrão baseado em eventos é outra solução que permite comunicação entre classes sem necessidade delas se conhecerem. Assim, um serviço pode notificar outro em que determinado evento ocorreu, e o serviço que foi notificado trata de efetuar alguma coisa ao receber essa notificação.

6.4.2. Implementação

A aplicação de *backend* foi desenvolvida com recurso à *framework* NestJS e com a linguagem *typescript*. Esta tem o propósito de servir a aplicação *frontend*, gerir a informação da plataforma, executando operações e manipulando os dados na base de dados. A aplicação providencia uma REST API, que é documentada através da ferramenta *swagger* e possibilita as aplicações externas comunicarem com o sistema. É também implementado um serviço de *sockets* para estabelecer comunicação bidirecional e em tempo real com a aplicação web. Foi implementado um sistema de filas que permite executar tarefas assíncrona e ordenadamente fora do processo principal, o que contribuiu para otimizar a performance da aplicação. Adicionalmente, existe um mecanismo de agendamentos que trata de criar e executar os *cron jobs*.

A documentação definida com a ferramenta *swagger* e acessível a partir do *URL* do *backend*, concatenando a *string* "/docs", permite visualizar e tirar partido de todos os *endpoints* disponibilizados na aplicação. Para cada *endpoint* é apresentado o tipo do método que deve ser escolhido (GET, PUT, DELETE, etc), uma resposta exemplo, o*s* dados e os seus tipos que devem ser enviados no *body,* mas também são exemplificadas as mensagens de erros que podem ser devolvidas e o código de erro associado.

A aplicação foi desenvolvida seguindo o padrão arquitetural *clean arquitecture,* alguns princípios do *domain-driven design,* aplicando alguns SOLID *principles* e outros padrões como o *repository* e *event driven.* É importante referir que nem todos foram seguidos à risca

e por isso alguns padrões não foram aplicados na sua totalidade, contudo serviram de orientação para a definição, estruturação e organização do código. Embora este seja um projeto com alguma dimensão e complexidade sentiu-se que certos conceitos não trariam benefício como os agregadores que são sugeridos pelos princípios de DDD, uma vez que o custo comparando com o valor adjacente a esses conceitos não justificou a sua adoção.

O *domain-driven design* serviu de guia para estruturar e organizar o projeto com o intuito de ser escalável e para isolar responsabilidades. Portanto, serviu essencialmente para determinar o domínio principal e os subdomínios, a aplicação de *use-cases*, do *repository* e do *event driven pattern*. O principal domínio é a execução de retrospetivas e este pode ser dividido em subdomínios. Como tal, para cada subdomínio é definido o módulo correspondente e que apresenta uma determinada estrutura, permitindo separar responsabilidades e estruturar a aplicação em camadas, de forma a seguir a *clean architecture*.

A Figura 29 pretende demonstrar a estrutura de pastas definida para o *backend* que ajuda a compreender a separação de responsabilidades e domínios da aplicação.

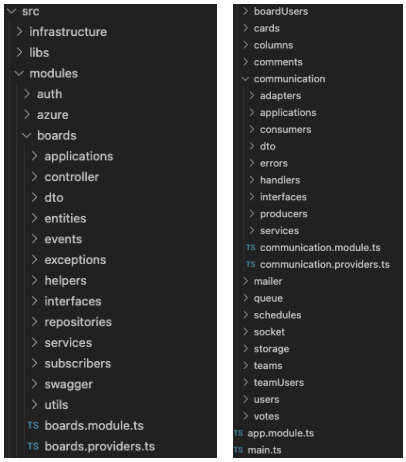


Figura 29 - Representação da estrutura de pastas do backend

Assim, é possível verificar que existem três pastas principais: infraestrutura, *libs* e módulos. A primeira diz respeito às configurações iniciais, tais como a injeção e validação das variáveis de ambiente, a definição do módulo responsável pela gestão dos *jwt tokens*, a definição do módulo da base de dados, onde é definido como a aplicação se deve ligar a esta, mas também são definidos os módulos de *mongoose* para construir as coleções no arranque da aplicação (se necessário). A pasta *libs* dispõe de todas as classes que podem ser reutilizadas na aplicação como constantes, *dtos*, *enumerators*, interfaces, modelos base, validadores de informação, etc. Finalmente, a pasta módulos define todos os subdomínios e

para cada um destes é associado um módulo que salvo algumas exceções, apresenta a seguinte estrutura: controladores, aplicações / casos de uso, dto, interfaces, repositórios, *entities* e serviços.

Seguindo os SOLID *principles* e aplicando o princípio da inversão de dependências e o princípio da segregação de interfaces, as classes de cada camada são injetadas mas apenas são disponibilizadas as interfaces correspondentes e elas só dispõem dos métodos que devem ser públicos para a camada exterior. O princípio da *single responsability* é aplicado na medida em que qualquer serviço, aplicação, *dto* ou interface diz respeito apenas a uma funcionalidade. O princípio *open-closed* também foi aplicado em algumas situações na medida em que certas classes não podem ser modificadas, mas sim estendidas como é o caso da classe genérica construída para implementação dos repositórios.

A Figura 30 representa a arquitetura do *backend* do SPLIT, correspondendo ao nível 3 do modelo C4, onde é possível visualizar as interações e troca de informação entre os componentes e sistemas.

Na figura é possível observar, da esquerda para a direita, três tipos de módulos distintos, dois deles que possibilitam a interação com o *frontend*, como o módulo do *socket,* responsável por lidar com a comunicação bidirecional e em tempo real; o segundo, que representa todos os módulos que efetuam operações sobre os dados, como por exemplo o módulo de autenticação por credenciais, o módulo de autenticação pela *azure*, o módulo de *storage* de *assets*, o módulo dos quadros, colunas, cartões, comentários, votos, agendamentos, utilizadores e equipas. Na imagem, à direita, encontra-se o módulo de comunicação que trata de adicionar as operações de comunicação às respetivas filas e executá-las ordenadamente. Isto significa que todas as operações de comunicação com o *slack* são lidadas por este módulo e para tal tira partido da base de dados Redis.

Seguindo o padrão *clean architecture,* na camada exterior encontram-se as bases de dados, as configurações, nomeadamente as variáveis de ambiente e todos os sistemas externos, como a comunicação com a *active directory* (AD)*,* o sistema de armazenamento de *assets,* o sistema de email e sistema de mensagens. As variáveis de ambiente servem para determinar algumas configurações da plataforma, nomeadamente a informação necessária para se ligar à base de dados mongo, à base de dados *redis*, à API do *slack*, o tempo de expiração do *access* e *refresh tokens.*

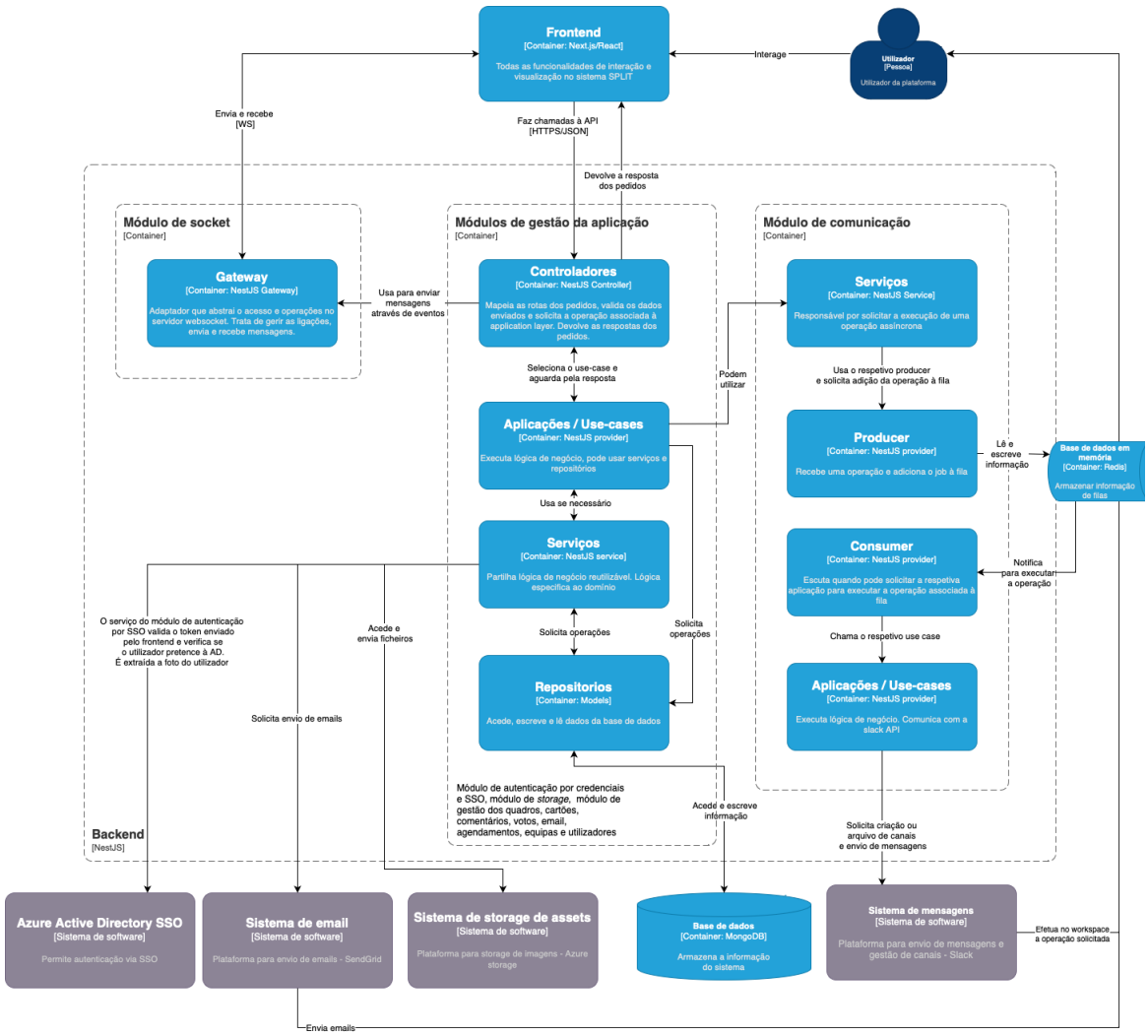


Figura 30 - Diagrama da arquitetura do *backend* - nível 3 do modelo C4

Como o projeto é *open-source* pretendeu-se tornar algumas funcionalidades opcionais, através de *feature flags*, que podem ser ativas ou não dependendo da necessidade de cada organização ou utilizador. Isto significa que certos módulos ou funcionalidades podem ou não ser disponibilizados com base nessa configuração inicial. Por exemplo, é possível definir se os módulos de autenticação por *azure* SSO, o serviço de emails ou o de comunicação devem ser importados e por isso disponibilizados ou não como funcionalidades. Todas as variáveis de ambiente são validadas no arranque da aplicação através da biblioteca *joi*[30].

O frontend pode comunicar por *websockets* através do *gateway* disponibilizado pelo módulo da esquerda ou através de pedidos HTTPS que são lidados pelos controladores disponibilizados pelos módulos representados na parte central do diagrama. A aplicação cliente consegue então comunicar com o *backend* através dos adaptadores disponibilizados

[30] https://joi.dev/

e representativos da segunda camada mais externa, segundo a *clean architecture,* e estes são os controladores e o *gateway*.

O módulo do *socket* possui apenas uma classe principal, o *gateway,* que é posicionada na camada dos adaptadores. Esta serve para gerir ligações, alocar os utilizadores que visualizam um quadro à respetiva *room* e serve para propagar as atualizações de um quadro para todos os utilizadores que o observam. Para os controladores de outros módulos poderem comunicar com este *gateway,* podem injetá-lo e utilizar a única instância desta classe. Como alternativa de comunicação entre classes, o *gateway* pode subscrever alguns eventos e que podem ser despoletados por outras classes.

Neste projeto e apenas em alguns locais, verificou-se uma quebra dos princípios de *clean architecure* que indica que as camadas internas não devem conhecer as externas. Em certos casos os serviços conhecem o *gateway* e esta é uma falha estrutural que pode ser resolvido através de eventos. Assim, identifica-se uma oportunidade de melhoria.

No que diz respeito aos módulos de gestão de dados da plataforma, estes são responsáveis por disponibilizar interfaces para gestão da informação da plataforma. Os módulos disponibilizam controladores que são um dos pontos de entrada e comunicação para as aplicações clientes. O acesso a determinadas rotas é protegido por *middlewares* denominados de *guards* que verificam se o utilizador envia algum mecanismo de autenticação, se este é valido, e se tem as permissões necessárias.

Como referido no capítulo anterior, definiu-se que um utilizador pode ser membro ou super-administrador da plataforma e pode participar numa ou mais equipas, tal como pode participar num ou mais quadros e poderá ter diferentes permissões em cada um dos domínios. Para gestão da autenticação e respetivos *tokens* é usada a biblioteca *passport* [31]. Através desta biblioteca é possível autenticar os utilizadores e devolver o *access* e *refresh tokens* criados nesse momento. Estes são *jwt tokens* que armazenam o tempo de expiração e o id do utilizador.

Em cada pedido os dados enviados no *body* são validados, em cada um dos *dtos,* utilizando validadores para cada atributo, sendo assim possível, de garantir e proteger que os dados enviados são do tipo correto. Se algum dado não corresponder ao devido tipo ou não apresentar as caraterísticas necessárias, é enviada uma exceção, impedindo que o pedido

[31] https://docs.nestjs.com/security/authentication

seja resolvido. Após a validação, o controlador transforma, se necessário, os dados recebidos e invoca a respetiva aplicação ou, por outras palavras, o caso de uso. Como os casos de uso são injetados mas apenas são disponibilizadas as interfaces nos controladores, estes apenas conhecem os métodos que são necessários e disponibilizados no caso de uso para serem chamados no método associado a cada *endpoint.*

Na camada seguinte, os *use-cases* e seguindo DDD à risca, devem ter a lógica de cada funcionalidade pedida pelo controlador, contudo foi adicionada uma subcamada que é chamada pelos casos de uso, camada essa que diz respeito aos serviços. Concretizando, os casos de uso, em vez de conterem a implementação da funcionalidade, chamam o respetivo serviço que trata da lógica de negócio e são estes que têm acesso aos repositórios. Assim, os serviços de cada funcionalidade são injetados no *use-case* mas apenas conhecem os métodos disponibilizados nas interfaces. Estas apenas disponibilizam os métodos do serviço que o caso de uso deve conhecer para cada funcionalidade. Os serviços podem ser partilhados entre os módulos o que não acontece com os casos de uso.

Como referido, os serviços executam a lógica de cada funcionalidade. Para tal, pode ser necessário aceder a métodos de outros serviços e utilizar repositórios. Os repositórios também são injetados e apenas disponibilizam os métodos que se encontram definidos nas suas interfaces. Um serviço nunca sabe como se deve aceder ou manipular os dados na base de dados uma vez que são os repositórios que têm essa responsabilidade. Assim, é possível desacoplar responsabilidades, ou seja, se for necessário trocar de base de dados, de ORM ou ODM não é necessário alterar a lógica de negócio.

Os repositórios utilizam a API da biblioteca *mongoose* para manipular os dados de cada coleção e providencia vários métodos definidos nas interfaces, que podem ser invocados pelos serviços de forma a solicitar acesso ou atualização dos dados. As entidades ou modelos de dados são construídos através de *mongoose schemas* onde se definem as propriedades, o tipo das mesmas e estes são importados no módulo da base de dados. Nestas classes são definidos todos os atributos, as relações entre documentos e a definição que determinado atributo poderá ter um conjunto de outros documentos de entidades diferentes.

Alguns serviços podem necessitar de chamar sistemas externos como é o caso do módulo de autenticação por *azure* que efetua chamadas à *active directory,* o módulo de armazenamento de *assets* que utiliza um serviço externo para guardar as fotografias dos

utilizadores, e o módulo de recuperação de *password* (em registos por credenciais) que utiliza um sistema de emails externo para enviar emails de recuperação.

Como referido, alguns serviços necessitam do resultado da execução de lógica de outros serviços para efetuar certas operações. Como tal, alguns dos módulos necessitam de enviar mensagens ou efetuar operações num *workspace* do *slack,* de forma a notificar os utilizadores de eventos na plataforma. Assim, o serviço de criação de uma retrospetiva/quadro dividido pode utilizar serviços do módulo de comunicação para notificar os utilizadores de que o processo de retrospetivas teve início e de que uma equipa foi dividida em subequipas, apresentando assim uma mensagem de texto num canal específico dos membros constituintes de cada subequipa. Neste processo, o serviço de comunicação é também responsável por criar os canais de *slack* para cada subequipa e outro para os responsáveis, sendo que, após a criação são inseridos os respetivos membros e é enviado o *link* do quadro para o canal correspondente. Isto é, para cada canal de uma subequipa é enviado o *link* do respetivo subquadro e para o canal dos responsáveis, e para o canal principal é enviado o *link* do quadro principal. Em cada subquadro é possível alterar o seu responsável e da subequipa associada, portanto, o serviço com a função de executar esta operação de atualização, solicita o envio de uma mensagem para o canal principal e outra para o canal da subequipa, notificando que o responsável foi alterado e este é adicionado no canal dos responsáveis.

A plataforma permite fundir os cartões de um subquadro para o principal, e o serviço responsável por fazê-lo, também, notifica o canal dos responsáveis de que uma determinada equipa se reuniu, deu por concluída a reunião e por isso os cartões passaram para o quadro principal. Quando todos os subquadros tiverem os seus cartões no quadro principal, outra notificação é enviada anunciando que pode dar-se início à fase seguinte, onde todos os responsáveis se juntam numa reunião para análise dos cartões e para juntar aqueles que dizem respeito ao mesmo tópico.

Adicionalmente, ao ser mudada a fase da retrospetiva na funcionalidade para o efeito, é despoletada uma mensagem para o canal principal das retrospetivas anunciando a nova fase. Assim, quando se inicia a fase de votação é enviada uma mensagem anunciando a fase atual e que devem proceder à votação dos cartões. De forma semelhante, ao ser finalizada a última reunião da retrospetiva, entre os responsáveis e *stakeholders,* na qual deu origem aos itens de ação, a retrospetiva é concluída e nesse momento a fase deve ser alterada para

"concluída". Alterando-a, os quadros ficam bloqueados e é enviada uma mensagem anunciando que o processo de retrospetiva terminou e são apresentados os itens de ação criados.

Após a criação dos quadros para as retrospetivas divididas e enquanto não é possível customizar, é agendada uma nova retrospetiva mensal que reutilizará as configurações definidas na criação dos quadros da retrospetiva anterior. Assim, é criado um *cron job* que, quando for despoletado, criará de novo o processo de retrospetiva divididas, gerando novamente as subequipas de forma aleatória, a partir da equipa anteriormente associada e são criados os respetivos quadros, bem como o quadro principal. Ao terminar este processo são enviadas novamente as notificações já descritas, através do *slack* e são arquivados os canais criados no processo anterior.

Para usufruir das funcionalidades de agendamentos foi instalado na framework a biblioteca *nestjs/schedule*[32] que permite realizar a gestão dos *cron jobs*. Observando o módulo de comunicação representado na parte direita da Figura 30, que serve para efetuar o processo de envio de mensagens descrito anteriormente, dispõe de uma estrutura um pouco diferente das restantes, devido à sua natureza e por isso, não contempla os controladores uma vez que este módulo é acedido a partir de outros. A gestão de filas é efetuada com auxílio da biblioteca *nestjs/bull*[33].

Quando um serviço que não pertencente ao módulo de comunicação pretende solicitar uma operação, invoca o respetivo serviço de comunicação para a executar. Nesse momento, o serviço do módulo de comunicação chama o *producer* que é responsável por adicionar a operação à fila associada, registando-a na base de dados em memória. O sistema *bull* notifica o *consumer* da fila quando uma determinada operação pode ser executada e, nesse momento, é chamado o respetivo *use-case* que contém toda a lógica da operação associada, de forma a ser executada. Como requisito para enviar mensagens ou efetuar operações no *slack* através da sua API que é disponibilizada, é necessário criar uma aplicação no respetivo *workspace*, atribuir as permissões e gerar um *token* que é usado sempre que o *backend* efetua um pedido à API externa do *slack*.

O padrão baseado em eventos foi utilizado para comunicação entre classes na funcionalidade de mudança de fase e na funcionalidade do timer. Este é um *countdown* que

[32] https://www.npmjs.com/package/@nestjs/schedule
[33] https://www.npmjs.com/package/@nestjs/bull

serve para ajudar a gerir a reunião. Como por exemplo, é possível definir algum tempo para a discussão de determinados tópicos e o tempo deve ser sincronizado entre todos os utilizadores que visualizem o mesmo quadro. No *timer* é possível definir o tempo, iniciar, parar, fazer *reset* ou pausar. O padrão permitiu evitar a injeção dos serviços do módulo dos quadros no módulo do *gateway* de *sockets* e vice-versa, o que permite a separação de responsabilidades e isto vai ao encontro dos princípios de *clean architecture* e DDD. Assim, o *gateway* ao receber uma mensagem promovida por uma ação do utilizador no *timer* do quadro em visualização, emite um evento e o *subscriber* associado à ação recebe-o. Este conhece o serviço do *timer* que implementa um conjunto de operações e é chamado o método associado à operação solicitada. Por sua vez, ao ser concluída a operação, o serviço emite outro evento para o *subscriber* que é responsável por propagar a alteração a todos os utilizadores que visualizam o quadro. Por isso, é injetado o *gateway* no *subscriber* que é responsável por lidar com a ação de propagar a informação para todos os utilizadores. O *subscriber* invoca o método correspondente do *gateway* para emitir a informação. O mecanismo *emit-subscribe* é aplicado de forma semelhante para a mudança de fases de uma retrospetiva e a respetiva propagação da informação.

6.5. Aplicação web

A aplicação web foi desenvolvida recorrendo a linguagens de programação, como *javascript* e *typescript* e através da framework Next.js, que tira proveito do *react* para a construção de uma *single-page application*. A escolha desta em prol de *react* deveu-se essencialmente à possibilidade de ter *server-side rendering,* o que proporciona uma melhor experiência de utilização, na medida em que a performance da aplicação é superior e por isso as páginas são carregadas com maior velocidade. Do ponto de vista de desenvolvimento a funcionalidade de *routing,* as configurações e ferramentas que são proporcionadas são, sem dúvida, uma mais-valia. Na Figura 31 é possível observar uma representação da arquitetura do *frontend* correspondente ao nível 3 do modelo C4. Antes de proceder à discrição do diagrama, é importante referir que, tal como no *backend*, foram definidas variáveis de ambiente que tornam certas funcionalidades opcionais e que, com base no seu estado, modificam a interface. Portanto, é possível definir se a interface deve apresentar a possibilidade de autenticação por credenciais e/ou por SSO. No que diz respeito a este último, é possível selecionar que *providers* de autenticação se pretende integrar, apresentando o botão de autenticação correspondente.

Na Figura 31 é observável que um utilizador da plataforma, independentemente do seu papel, pode visualizar e interagir com esta, no entanto, pode ter permissões diferentes. O utilizador ao aceder a um *url* da aplicação, a partir do browser, é requisitada a página correspondente. O Next.js sabe qual a página que foi solicitada a partir do URL através do seu mecanismo de *routing*. Acedendo ao URL é então promovida a solicitação da página ao *server side* do *Next* que é responsável por prepará-la. Nesta fase de preparação são efetuados os pedidos necessários ao *backend* pelo método *getServerSideProps()*. Uma vez recebida a resposta, os dados são enviados para o *react* que é responsável por construir o HTML da página tendo em conta os componentes de que é constituída e anexar o código *javascript* que permite tornar a página interativa quando for executado no browser. Após a preparação da página, esta é enviada para o *browser*.

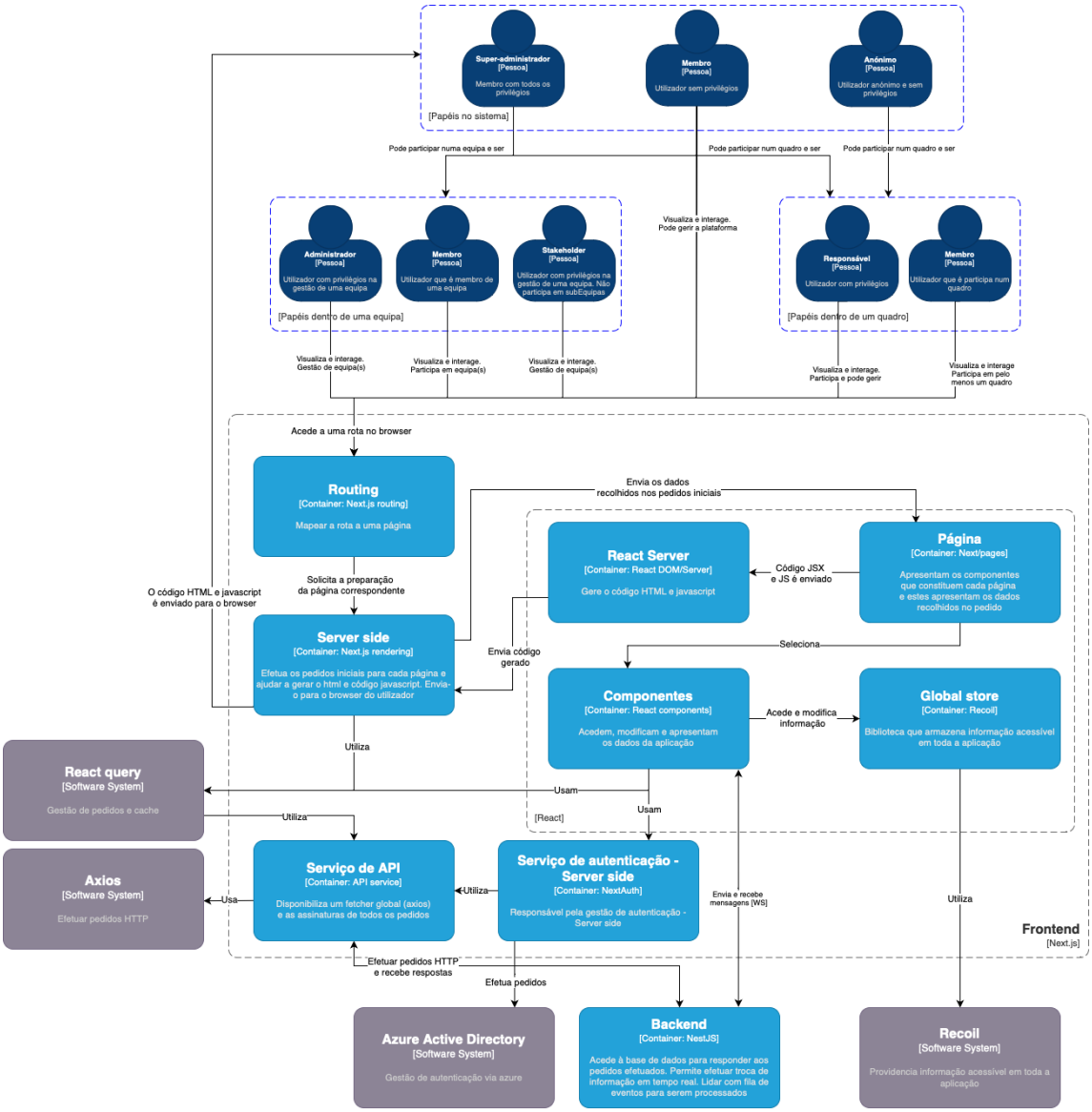


Figura 31 - Representação da arquitetura do *frontend*. Nível 3 do modelo C4

Os componentes que necessitam de efetuar operações de autenticação, como *sign in*, *sign out*, acesso ou validação da sessão, tiram partido do *hook* disponibilizado pela biblioteca *NextAuth* que comunica o respetivo serviço de gestão e que corre em *server-side*. Este serviço foi implementado com recurso à biblioteca *NextAuth* e tem a responsabilidade de gerir, na totalidade, a autenticação na plataforma tratando de efetuar os pedidos necessários ao *backend* ou aos *providers* disponíveis. O serviço, ao autenticar o utilizador, guarda a informação de forma segura no seu browser.

No que diz respeito à proteção de rotas, foi criado um *high order component* (HOC) que na execução do método *getServerSideProps*() da página, verifica se o utilizador se encontra autenticado, caso contrário é redirecionado para a *landing page*. Assim, é possível proteger as páginas de serem visualizadas por utilizadores não autenticados ou que não tenham permissões para tal.

No que diz respeito à troca de informação em tempo real, a página dos quadros quando é renderizada estabelece uma ligação ao *gateway* de *websockets* no *backend*. Para isolar e reutilizar o código no que diz respeito aos *sockets* foi desenvolvido um *custom hook* que trata de efetuar a ligação e contém os métodos necessários para ficar à escuta de eventos emitidos pelo *backend* ou para efetuar a emissão de eventos. Portanto, todos os utilizadores que entram num quadro, estabelecem uma ligação bidirecional com o servidor e se bem-sucedida enviam uma mensagem para o *gateway* com o id do quadro que estão a visualizar e o *backend* trata de adicionar os utilizadores na *room* identificada pelo id do quadro. Quando um dos utilizadores efetua uma alteração num quadro, é efetuado o pedido correspondente ao *backend* e no *frontend* ocorre um *update* otimístico, portanto efetua-se e verifica-se imediatamente a alteração na interface sem aguardar pela resposta do pedido. Se o pedido devolver um código de erro, as ações são revertidas. Tanto os *updates* otimísticos como a reversão das alterações são possíveis de efetuar facilmente através da biblioteca *react query*. Portanto, a alteração no quadro promovida pelo utilizador é imediatamente verificada na interface, é enviada a solicitação ao *backend* através de um pedido HTTPS no qual pode ser enviado um *dto* no *body*. Após a operação bem-sucedida, o *backend* além de enviar a resposta ao pedido, trata de propagar o *dto* através de *websockets*, para todos os utilizadores que se encontram na *room* com o id do quadro. No *frontend*, os clientes (exceto quem efetuou o pedido) recebem a informação propagada e a aplicação sabe como atualizar o quadro com base no *dto* recebido.

Contrariamente à implementação do *backend,* não foi utilizado nenhum padrão arquitetural de forma mais rigorosa, contudo utilizaram-se alguns conceitos de *domain-driven design* e SOLID *principles*. No que diz respeito à organização do código, em pastas, optou-se por uma estratégia em que cada pasta representa um domínio, como é possível observar na Figura 32. Esta estratégia foi sugerida por membros da empresa. Com o intuito de reutilizar lógica, foram criados *custom-hooks* e cada um destes apenas dispõe de lógica associada a um domínio específico.

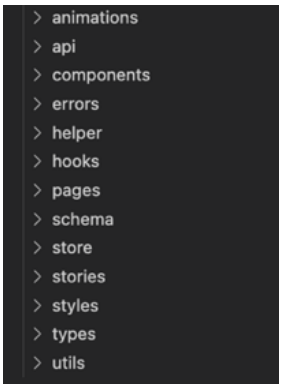


Figura 32 - Estrutura de pastas do *frontend*

Antes da construção das páginas e dos componentes complexos mapeou-se, através de código, o *design system* desenvolvido pela *designer* e referido na secção 5.6. Portanto, através das configurações globais da biblioteca *stitches*, definiram-se todos os tipos de letra, tamanhos, espaçamentos e cores que o projeto utiliza e definidos no *design system*. Esta biblioteca foi uma mais-valia uma vez que permitiu de forma acessível e organizada, definir todas as variantes dos componentes. A partir deste ponto foram criadas todas as primitivas (componentes básicos) que se encontram no Apêndice M – *Design* system e, para cada um destas definiram-se as suas variantes. Portanto, os componentes podem ser reutilizados e se é pretendido mudar o aspeto visual apenas é necessário mudar a propriedade variante para o valor desejado.

A partir destes componentes foi possível construir interfaces complexas que são apresentadas no Apêndice N – Interface para desktop. Neste é possível visualizar todos os ecrãs desenvolvidos durante a fase de desenho do produto como é referido na secção 5.7. Um dos exemplos de interfaces complexas é a página dos quadros onde se pode verificar um quadro interativo que apresenta funcionalidades de *drag-and-drop* que foram implementadas através da biblioteca *react-beautiful-dnd*, como será visto na próxima secção. No que diz respeito a formulários, todos são validados utilizando a biblioteca *react-hook-form* e, para cada um definiu-se um *schema* onde são registadas as regras para todos

os campos de cada formulário. Se alguma regra não se verificar a biblioteca torna o formulário inválido.

O *storybook* foi incluído nesta aplicação com o intuito de providenciar documentação e apresentar os componentes desenvolvidos na aplicação e as suas variantes. Na Figura 33, é possível visualizar do lado esquerdo alguns componentes implementados e a respetiva *story*. No conteúdo da *story* apresentada na figura é possível verificar as várias variantes de um botão e em baixo as propriedades que podem ser definidas no mesmo. Estas ao serem alteradas promovem automaticamente o ajuste visual do componente. A ferramenta é útil porque providencia um meio para os *developers* conseguirem, visualmente, compreender os componentes existentes na aplicação, os seus detalhes e como podem ser utilizados.

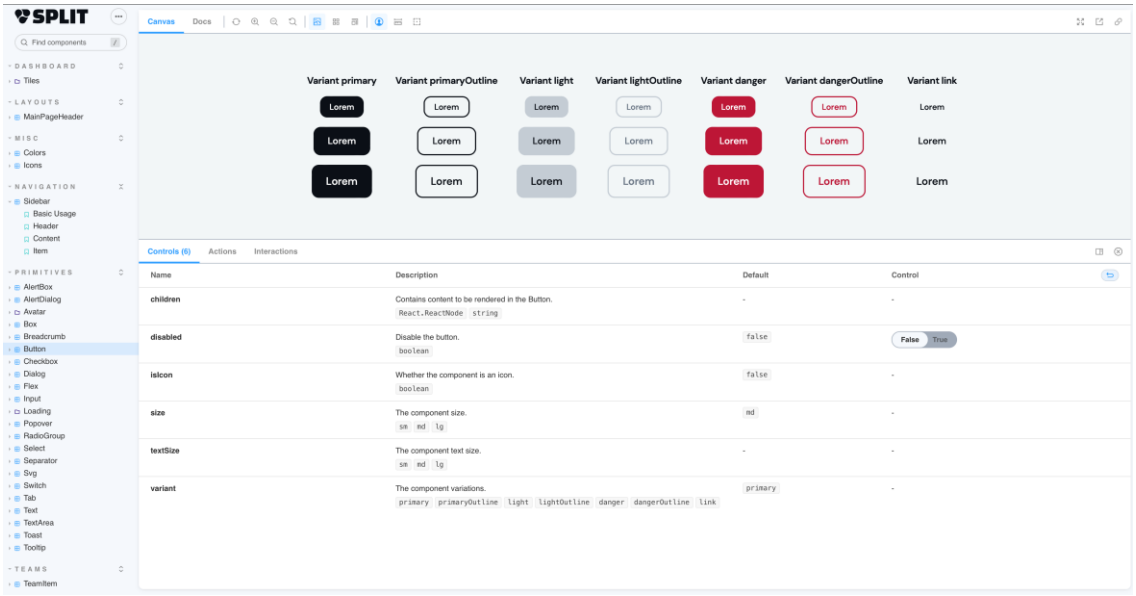


Figura 33 - Representação do storybook - componente botão

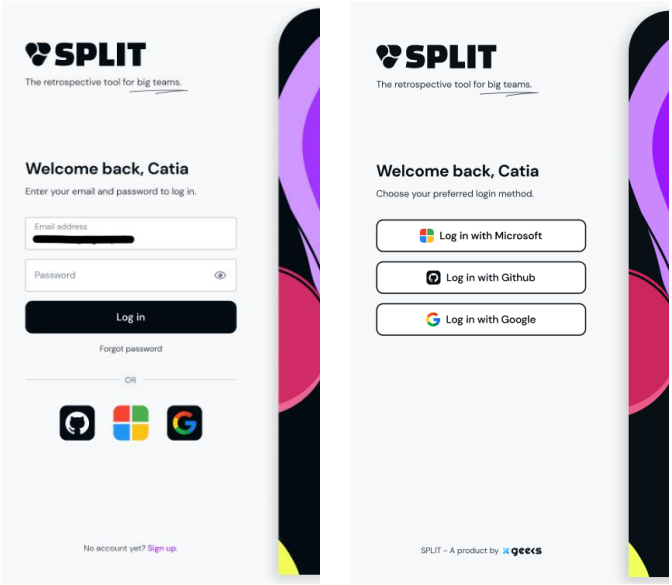
6.6. Páginas e funcionalidades

A presente secção tem o propósito de apresentar as funcionalidades implementadas no projeto, descrevendo-as, mas também realçando os aspetos relevantes da sua implementação.

6.6.1. Autenticação

Na aplicação SPLIT é possível efetuar autenticação de duas formas. Na *landing page,* o utilizador pode optar por efetuar o login através de credenciais (email e password) ou através de SSO como se pode observar na Figura 34.

O componente de autenticação apresenta diferentes aspetos com base nas variáveis de ambiente definidas, como se pode observar na Figura 34. Portanto, se a variável *NEXT_PUBLIC_LOGIN_SSO_ONLY* se encontrar a *false* é apresentado o ecrã da esquerda (a), e de forma inversa é apresentado o da direita (b). A apresentação dos ícones de autenticação do lado esquerdo ou os botões de diferentes *providers* no lado direito são condicionalmente apresentados com base nas variáveis de ambiente definidas. Para a utilização do SPLIT na xgeeks, apenas se encontra configurado a autenticação pela Microsoft.



(a) (b)

Figura 34 - Representação da funcionalidade de autenticação. (a) - login por credenciais e SSO; (b) – login por SSO.

No *frontend,* a biblioteca NextAuth, que lida com a autenticação, sabe como comunicar com o *backend* no caso de o *login* ser efetuado com credenciais, mas também pode saber como comunicar com os diferentes *providers* se estiver configurado para tal. Para a versão atual do projeto foi configurada a autenticação por credenciais e por Microsoft, no entanto qualquer organização ou utilizador que pretenda usar o projeto pode implementar outro *provider* de forma bastante acessível.

Na autenticação por credenciais, o *frontend,* em específico o serviço *server-side* do *NextAuth*, envia um pedido ao *backend* com as credenciais do utilizador e este verifica se os dados são validos e se o utilizador se encontra na base de dados e, em caso afirmativo, o *backend* devolve o *access* e *refresh tokens* gerado pelo *passport*.

No caso de autenticação pela Microsoft, o utilizador é redirecionado para o formulário do *provider* onde deve colocar e submeter os dados de autenticação. Se for bem-sucedido é devolvido, pelo *provider,* um *jwt token* com a informação do utilizador e que é enviado para o *backend*. O módulo de autenticação por *azure* trata do resto do processo, ou seja, o controlador ao receber o *token* através do pedido para o efeito, envia-o para o *use-case* que trata de efetuar o *login* e/ou registar um utilizador na plataforma. Portanto, o *jwt token* quando recebido pelo *backend*, é descodificado e o email extraído. De seguida, existe uma verificação na base de dados para saber se o utilizador com aquele email já se encontra registado, mas também é efetuado um pedido ao *Microsoft graph*[34] do *active directory* para validar o *token*. Se for válido e o utilizador não existir na base de dados significa que pode ser registado, seguidamente é efetuado o *login* e os *tokens* devolvidos. Caso o utilizador já exista, apenas é efetuado o login e são devolvidos para o cliente o *access* e *refresh token* gerados pelo *passport.*

Independentemente do método de autenticação, quando este é efetuada com sucesso, os *tokens* são recebidos pelo serviço de autenticação no *frontend* e é criada uma sessão para o utilizador, os dados da mesma são armazenados nos *cookies* do browser*,* e o utilizador é redirecionado para o *dashboard.*

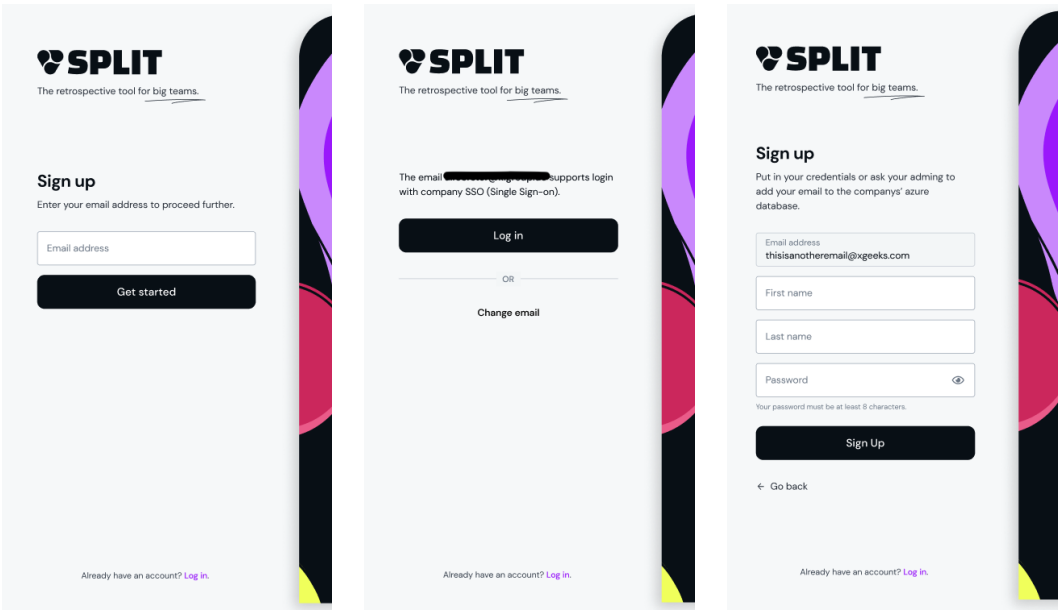
6.6.2. Registo por credenciais

Na aplicação SPLIT, se a autenticação por credenciais se encontrar ativa, é possível efetuar o registo a partir da interface. Como tal, a Figura 35 pretende apresentar os três ecrãs possíveis ao longo do processo de registo na aplicação.

A Figura 35 (a) corresponde ao ecrã onde o utilizador deve inserir o seu email para se registar. Ao proceder, se outro método de autenticação se encontrar configurado, o email é validado na medida em que se verifica, primeiramente, se aquele email pode ser utilizado para se autenticar por SSO. Em caso afirmativo, o utilizador é redirecionado para a Figura 35 (b) onde pode selecionar se pretende continuar com o processo de autenticação por SSO ou inserir outro email. Se o email do utilizador não for possível de autenticar por SSO e se já existir na base de dados é indicado que esse email se encontra registado. Caso o email não se encontre na base de dados, o utilizador é redirecionado para o ecrã representado pela Figura 35 (c), onde pode introduzir os seus dados e prosseguir com o registo inserindo

[34] https://learn.microsoft.com/en-us/graph/overview

algumas informações necessárias. Após a autenticação o utilizador é automaticamente autenticado e redirecionado para o *dashboard*



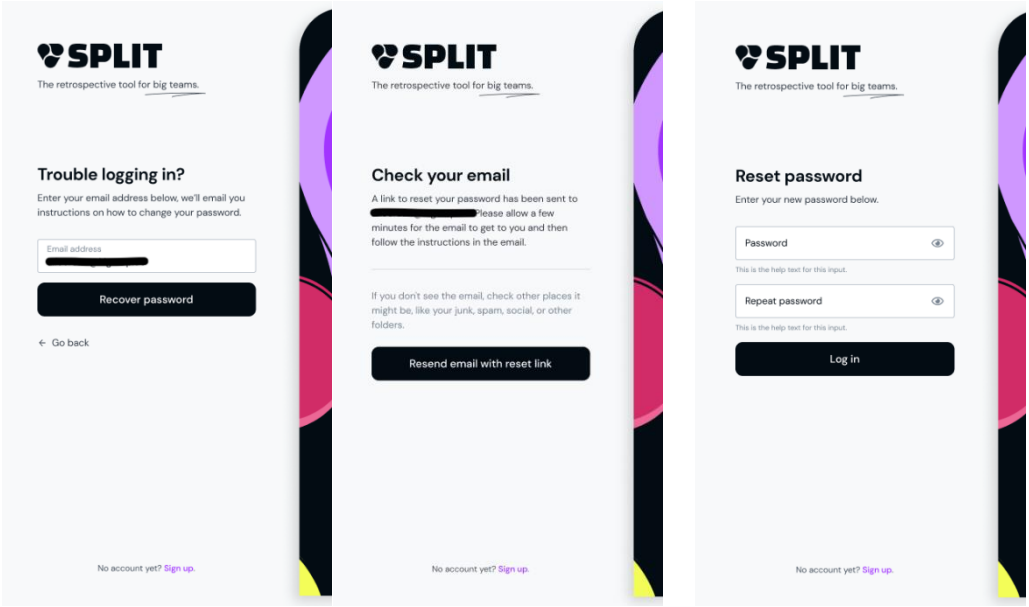
(a) (b) (c)

Figura 35 - Representação da funcionalidade de registo.
(a) – Inserção de email; (b) – Aviso de autenticação por SSO; (c) – Formulário de registo

6.6.3. Recuperação da *password*

Na sequência do registo é importante proporcionar ao utilizador alguns mecanismos de resolução de problemas. Assim, foi implementada a funcionalidade de recuperação de *password* e na Figura 36 é possível observar a representação visual do processo.

Na Figura 36 (a) é possível verificar o ecrã para inserção do email. Ao submeter é efetuado o pedido ao *backend* que trata de criar um documento na coleção "*ResetPassword",* onde é armazenada a informação de que foi efetuado um pedido para recuperar a *password*. Neste é guardado o email do utilizador, é gerado e guardado um *token* aleatório composto por 7 dígitos e também um *timestamp* de expiração do mesmo. Se o pedido de recuperação for bem-sucedido é pedido ao serviço de emails que envie um email para o utilizador e este é encaminhado para a Figura 36 (b). O utilizador, no email que recebe, dispõe de um *link* para a plataforma e este terá um *url parameter* onde consta o *token* gerado pelo *backend*. O utilizador ao aceder o *url* é reencaminhado para o ecrã representado na Figura 36 (c) e pode introduzir a nova password. Caso o *token* tenha expirado é negada a alteração da *password* e reencaminhado para o ecrã inicial.



(a) (b) (c)

Figura 36 - Representação da funcionalidade de recuperação da password.
(a) – Insereção de email (b) – Mensagem informativa (c) – Alteração da password

6.6.4. Dashboard

O utilizador ao autenticar-se é redirecionado para o *dashboard,* e todas as páginas possíveis de navegar a partir da barra de navegação seguem o mesmo *layout,* isto é, todas as páginas apresentam a barra de navegação e o espaço à direita diz respeito ao conteúdo da página. Na Figura 37 (a) é possível verificar a barra que permite navegar entre as páginas. É importante referir que as páginas *account* e *settings* ainda não se encontram disponíveis.

Na Figura 37, a secção (b) diz respeito a algumas estatísticas que são providenciadas pelo *backend* quando a página é carregada. Portanto, é possível visualizar em quantos quadros o utilizador participa, em quantas equipas e também o número de utilizadores registados na plataforma. No canto superior direito verifica-se um botão que serve para a criação de quadros que é detalhado na próxima secção.

Na zona (c) da figura anteriormente apresentada, pode encontrar-se a listagem ordenada por data de atualização, de todos os quadros em que o utilizador participa há menos de três meses. Com o intuito de otimizar a experiência do utilizador, os quadros não são todos devolvidos ao mesmo tempo na resposta ao pedido. Para isso, implementou-se no *backend* um mecanismo de paginação, portanto, o *browser* ao efetuar o pedido pode enviar como parâmetros, o número da página que quer receber, bem como o número de quadros que devem ser devolvidos por página.

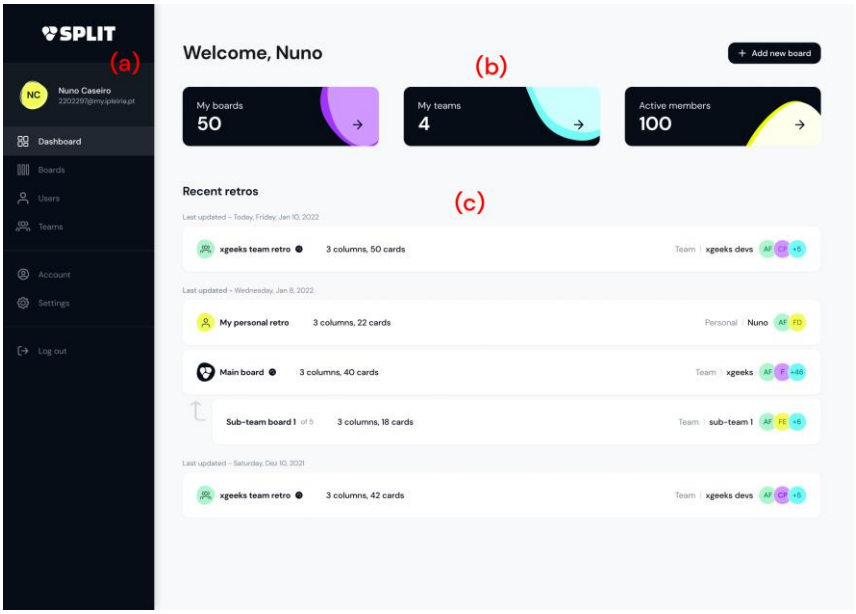


Figura 37 - Representação da página de *dashboard*

A biblioteca *react-query* apresenta uma funcionalidade para gerir e lidar com pedidos de paginação. Como tal, implementou-se na listagem de quadros um mecanismo de *lazy loading*. Desta forma, o utilizador efetua *scroll* pela lista e quando atingir uma distância perto do limite inferior é efetuado um pedido pela página seguinte, e uma vez recebida a resposta, os quadros são imediatamente apresentados na interface. Quando o pedido é efetuado, surge no final da lista, um ícone de *loading* representado por três pontos.

6.6.5. Criação de um quadro - retrospetiva regular

O utilizador quando pretende criar um quadro para uma retrospetiva deve carregar no botão que se encontra no canto superior direito da Figura 37 e é-lhe apresentado o ecrã observável na Figura 38 onde pode selecionar se pretende criar um quadro para uma retrospetiva regular ou dividida.

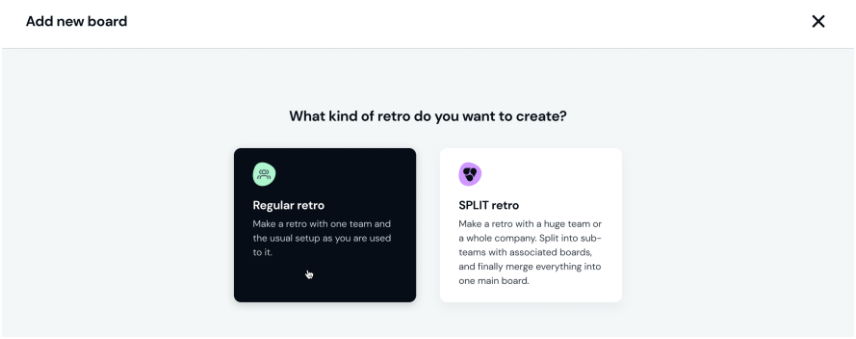


Figura 38 - Representação do ecrã para seleção do tipo de quadro

Selecionando a opção da esquerda, o utilizador é redirecionado para a página de criação do quadro para uma retrospetiva regular onde pode definir quem serão os participantes do quadro, mas também definir as configurações iniciais.

Observando a Figura 39, na *tab* de participantes, é possível constatar dois *radio buttons*. Selecionando o primeiro, do lado esquerdo, o utilizador pode escolher uma equipa em que participe e seja administrador, e assim, todos os membros serão participantes do quadro.

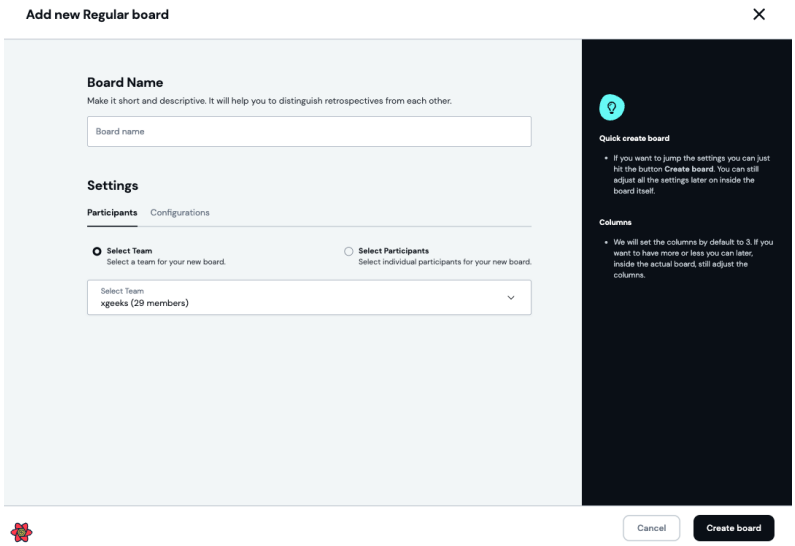


Figura 39 - Representação do ecrã de criação de um quadro para retrospetivas regulares – escolha de participantes.

Se o utilizador clicar no *radio button* da direita, tem a possibilidade de selecionar individualmente os participantes através de uma vista que surge do lado direito, composta por um *input* de pesquisa e por uma lista de utilizadores registados na plataforma. Para selecionar os utilizadores, estes têm uma *checkbox* antes do seu nome que serve para esse fim.

Ainda na secção das *settings*, é possível escolher outra *tab*, denominada de *configurations* onde o utilizador pode definir as preferências iniciais, como é observável na Figura 40.

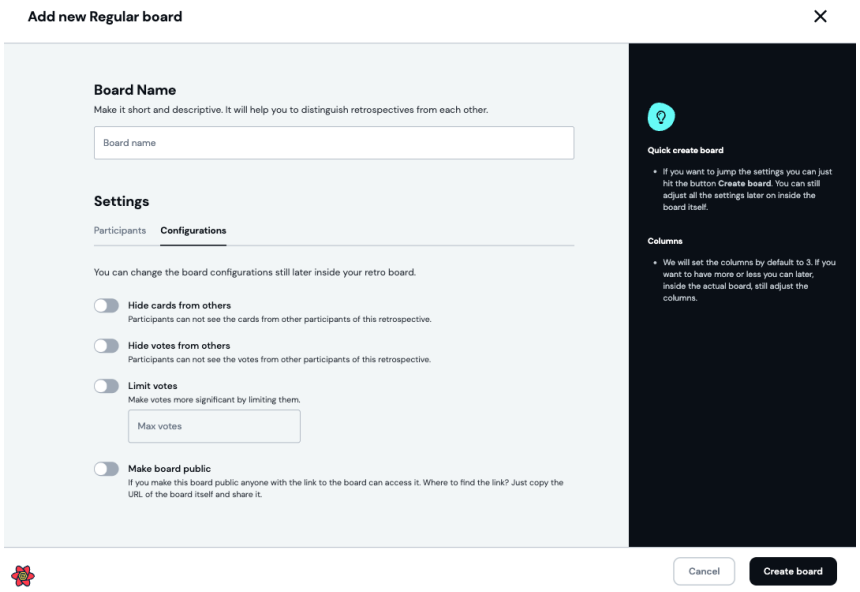


Figura 40 - Representação do ecrã de criação de um quadro para retrospetivas regulares – configurações.

Nesta, é possível verificar que o utilizador pode selecionar se pretende que os cartões ou os votos atribuídos a estes sejam escondidos para os outros utilizadores que não sejam os autores do conteúdo. Adicionalmente, o utilizador pode definir o limite de votos no quadro e por utilizador, mas também pode definir, se o quadro deve ser público ou privado. Se o quadro for público, o acesso é disponibilizado a todos os utilizadores que acedam através do URL do quadro. As primeiras três opções também podem ser definidas num quadro de retrospetivas divididas, como será referido na próxima secção

6.6.6. Criação de um quadro - retrospetiva dividida

Como referido anteriormente, o SPLIT tem como foco a execução de retrospetivas em equipas grandes, e por isso surge a estratégia de divisão da equipa em subequipas e a criação dos respetivos quadros e respetivos canais no *slack*.

Para dar início ao processo, o utilizador pode selecionar a opção da direita, apresentada na Figura 38 e, tal como nas retrospetivas regulares, o utilizador é reencaminhado para a página de criação do respetivo quadro e através da qual se pode definir o nome para o mesmo, assim como definir algumas configurações iniciais.

Como se pode observar na Figura 41, a grande diferença estrutural nesta página, comparando com a de criação de quadros para retrospetivas regulares, é a possível apresentação das subequipas geradas a partir da equipa selecionada e são apresentados os respetivos quadros onde também é possível selecionar aleatoriamente um novo responsável.

Assim, neste ecrã é possível escolher uma equipa em que o utilizador autenticado é administrador. Pode-se também visualizar os *stakeholders* da equipa e que por isso não participam nas retrospetivas das subequipas, pré-visualizar como será dividida a equipa com base nas definições de divisão e qual o responsável para cada subquadro. O utilizador pode definir aleatoriamente outro responsável, ao carregar no ícone com a varinha mágica. O responsável é escolhido tendo em conta a data de criação da conta na Microsoft ou na plataforma, isto significa que apenas os membros que se encontrem na empresa há mais de três meses poderão ser selecionados como responsáveis. Caso não existam elementos suficientes há menos de três meses os responsáveis são escolhidos aleatoriamente e sem restrições.

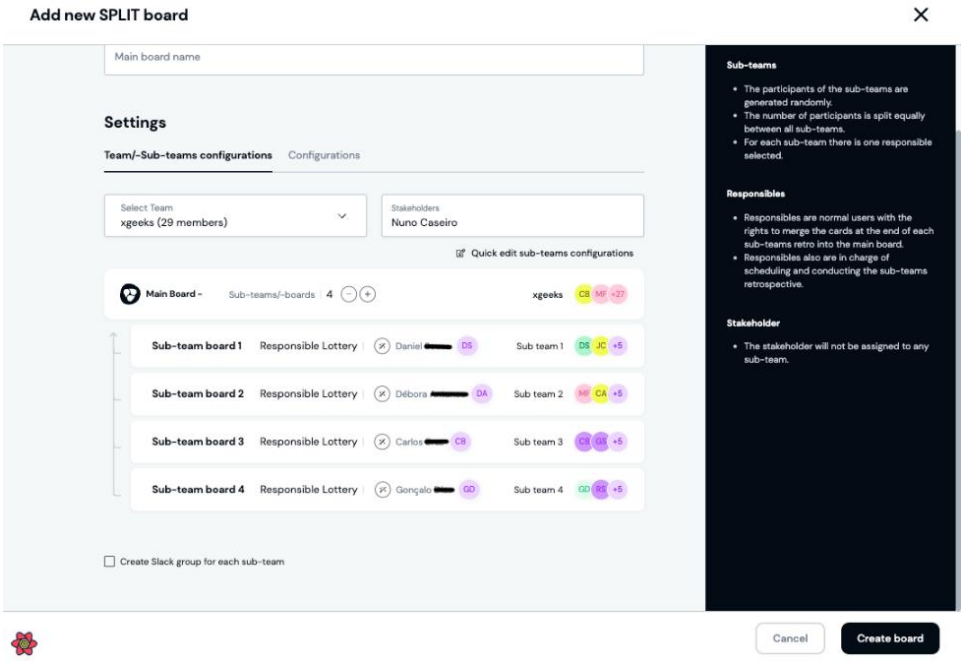


Figura 41 - Representação da funcionalidade de criação de retrospetivas divididos e os respetivos quadros

Adicionalmente e descrito de seguida, o utilizador pode ajustar o número de equipas ou o número máximo de membros por equipa, sendo que o valor mínimo para ambos os casos é de 2. Por isso é necessário que a equipa principal escolhida seja composta por pelo menos quatro membros.

Como a aplicação ainda não possui a funcionalidade completa de agendamentos, com o intuito de recriar o processo de retrospetivas que se efetua mensalmente na xgeeks, definiu-se que, apenas para a equipa com o nome da empresa é criado um *cron job* mensal no *backend.* O *job* ao ser despoletado, utiliza as configurações do mês anterior para recriar o processo de uma retrospetiva dividida. Assim, todos os meses este processo é recriado, a

equipa é dividida, são criados todos os quadros e canais do *slack* necessários, são selecionados novos responsáveis e são mantidas as configurações anteriores.

No que diz respeito à integração com o *slack*, uma vez que ainda não é possível definir a partir da interface, as configurações do *slack* por retrospetiva, definiu-se que, para a equipa cujo nome é *xgeeks,* se a *checkbox* apresentada no final da Figura 41 se encontrar selecionada, promove um conjunto de ações e notificações. Portanto, após criação dos quadros para uma retrospetiva dividida na plataforma, o *backend* adiciona um *job* na fila que é responsável por enviar uma mensagem no canal principal de retrospetivas. A mensagem indica que o processo de retrospetivas do mês atual teve início e também são apesentadas por texto as subequipas geradas, os membros constituintes tal como os responsáveis selecionados. Adicionalmente, é criado um canal no *slack* para cada subequipa e os membros de cada uma são convidados para o respetivo canal. Um canal para os responsáveis também é criado e os seus membros convidados. Em cada canal é enviada uma mensagem com o URL para o respetivo quadro. Regressando aos agendamentos, se o *cron job* mensal for despoletado, são utilizadas as configurações armazenadas para verificar se, no mês anterior, foram ativadas as notificações pelo *slack* e em caso afirmativo são também realizadas as operações descritas e enviadas as respetivas mensagens.

Ainda nesta página, o utilizador para poder aumentar o número de subequipas/subquadros pode utilizar os botões de “+” e “-”, sendo que ficam desabilitados quando alguma operação não é mais possível de efetuar. Como alternativa, foi definido um menu que permite introduzir valores numéricos para o número de equipas ou número máximo de elementos por equipas, como se pode observar na Figura 42. Nesta, o utilizador pode ajustar o número de equipas ou o número máximo de elementos por equipa, e ao modificar um dos *inputs* automaticamente ajusta o outro. Ao guardar as configurações, os ajustes são automaticamente visualizados no ecrã.

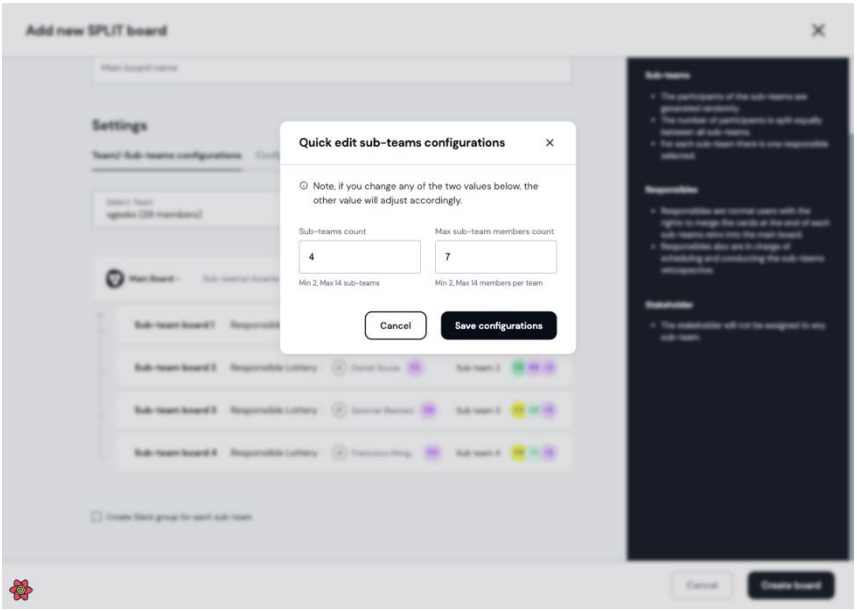


Figura 42 - Representação do menu para configuração das subequipas

6.6.7. Quadros de retrospetivas divididas e regulares e respetivas diferenças

Nesta secção são apresentadas as funcionalidades existentes em todos os tipos de quadros, mas também aquelas especificas de cada tipo. Pode-se considerar que existem três tipos de quadros, o quadro de uma retrospetiva regular e dois de uma retrospetiva dividida. Na dividida existe o quadro principal e o subquadro. A estrutura de um quadro é essencialmente definida por um conjunto de colunas que podem conter cartões individuais ou de grupo e cada um pode conter comentários e votos.

Visualizando a página de um quadro, como é possível visualizar na Figura 43, esta apresenta sempre um cabeçalho com algumas informações. No canto superior esquerdo pode encontrar-se um encadeamento de nomes de páginas com hiperligações que pretendem indicar onde o utilizador se encontra, e ao clicar neles pode navegar entre as páginas. Logo em baixo, é possível verificar o título do quadro. No lado direito do cabeçalho é possível encontrar os participantes separados por *roles* e ao clicar, dependendo do tipo de quadro, pode observar-se detalhadamente os utilizadores participantes.

Em qualquer quadro, os utilizadores com privilégios e por isso, os responsáveis, administradores de uma equipa ou criadores de um quadro podem observar e interagir com um botão do lado direito do mesmo para aceder e editar as configurações do quadro.

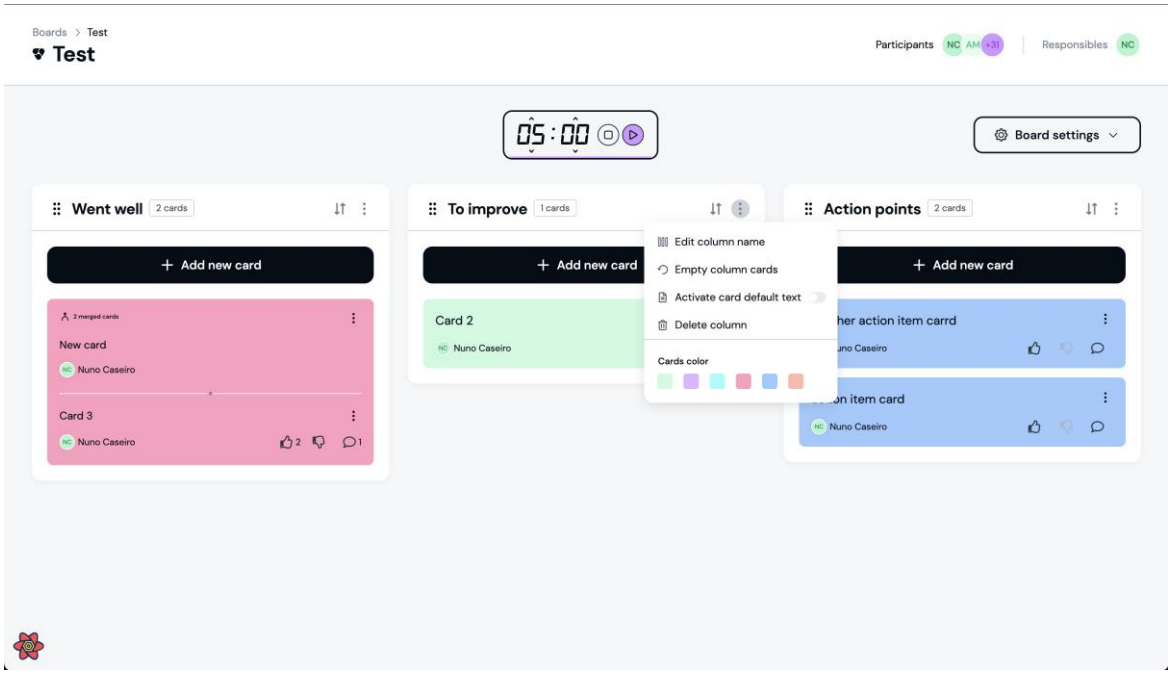


Figura 43 - Representação de um quadro de uma retrospetiva regular

Todos os quadros apresentam por omissão três colunas e a estas podem ser adicionados cartões que podem ser editados ou removidos. Em todos é possível mover os cartões entre colunas através de *drag and drop,* e esta funcionalidade foi possível implementar com recurso à biblioteca *react-beautiful-dnd*. O utilizador também pode juntar cartões e para tal pode arrastar um cartão para cima do outro. O cartão que está a ser arrastado quando se encontra em posição para ser junto a outro, apresenta um efeito de *blur,* dando assim a informação ao utilizador que este será agrupado. Inversamente, o utilizador dispõe da possibilidade de remover um cartão quando este se encontra num grupo. Em relação aos comentários, é possível adicioná-los a um cartão, editar ou removê-los. Existe também a possibilidade de votar nos cartões, mas também de remover os votos. Na adição de cartões e comentários é possível criá-los como anónimo e por isso não é apresentado o nome do autor. Em cada coluna existe a funcionalidade de ordenação por votos, ascendente ou descendente.

Em todos os quadros é possível observar um cronómetro que pode servir para temporizar algum tipo de tarefas, como discussão de ideias, tempo para adicionar cartões, votar, etc.

Na Figura 43 é possível verificar um exemplo de quadro de uma retrospetiva regular, onde podem ser observadas as funcionalidades descritas, mas também encontrar funcionalidades específicas deste tipo de quadros, como é descrito de seguida.

Neste tipo de quadros para retrospetivas regulares, exclusivamente, porque a estrutura pode ser modificada ao contrário de um quadro de uma retrospetiva dividida, é possível mover as colunas de posição através de *drag and drop,* adicionar (máximo de 4) e remover colunas (mínimo de 1). O utilizador pode também editar o nome de uma coluna, apagar todos os cartões, definir um *template* para o texto dos cartões e também pode alterar a cor dos mesmos.

Num quadro regular é possível adicionar e remover participantes e, para tal, ao clicar no canto superior direito onde se encontram os avatares com as iniciais dos nomes dos participantes, o utilizador é redirecionado para uma nova página onde são listados os utilizadores participantes e a partir da qual podem ser removidos ou adicionados.

Em relação às configurações de um quadro de uma retrospetiva regular, como se pode visualizar na Figura 44, é possível editar as configurações iniciais através do menu para o efeito e a partir do qual se pode definir algumas permissões e disponibilização de funcionalidades, mas também é possível remover ou adicionar colunas e definir a privacidade de um quadro.

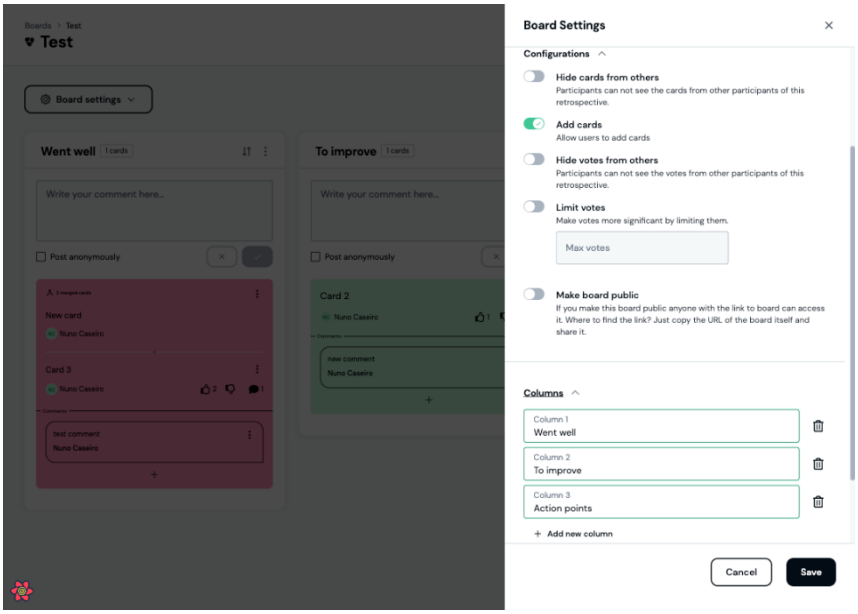


Figura 44 - Representação do menu de configurações de um quadro de uma retrospetiva regular

Os quadros de retrospetivas regulares podem ser públicos ou privados e isto significa que, no caso de o quadro ser público, os utilizadores não autenticados podem aceder ao quadro, no entanto antes de entrarem são reencaminhados para a *landing page* onde devem preencher o seu nome, como é possível visualizar na Figura 45.

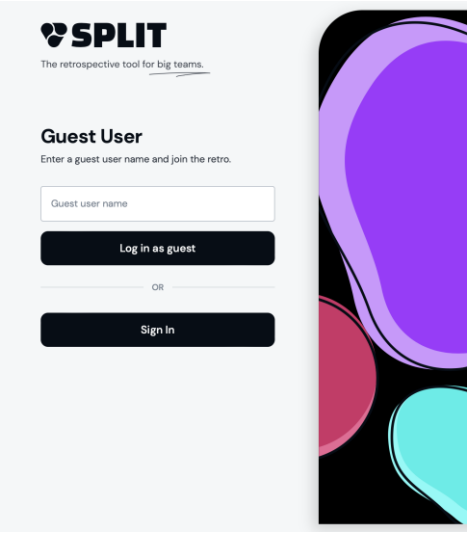


Figura 45 - Representação do ecrã de autenticação como anónimo

Um utilizador pode visualizar um quadro regular que seja público ao autenticar-se normalmente ou entrar como anónimo. Um utilizador anónimo ao tentar aceder a um quadro público é verificado se existe um *userId* armazenado nos *cookies*. Se não existir, o utilizador, após preencher o seu nome e submeter, promove um pedido ao *backend* para o registo na base de dados. Nesta é criado um utilizador anónimo e o utilizador é adicionado ao *array* de participantes do quadro, e por isso terá permissões para entrar e também possibilita o rastreamento do seu conteúdo. Após o registo, o *id* do utilizador e um *access token* é enviado na resposta e estes são guardados nos *cookies*. Ao aceder novamente, como um utilizador não autenticado, ao mesmo ou a outro quadro público, se for verificado que o *browser* do utilizador tem armazenado um *userId* nos *cookies*, considera-se que o utilizador já tentou entrar em algum quadro e por isso já se encontra registado e desta forma apenas é necessário verificar se entrou alguma vez naquele quadro que pretende visualizar. Caso ainda não o tenha visitado, o utilizador é adicionado aos participantes do quadro e consegue entrar no mesmo sem introduzir novamente o seu nome. Desta forma, um utilizador anónimo que já entrou num quadro pode alterar os seus cartões, votos ou comentários, mesmo que efetue *refresh* ou feche o *browser*. Caso o utilizador limpe os *cookies*, é considerado que este é outro utilizador e por isso o processo é reiniciado.

Prosseguindo pelas configurações, em todos os tipos de quadro é possível esconder ou mostrar os cartões e os votos dos utilizadores, definir se é possível ou não adicionar cartões e limitar os votos por utilizador.

Ainda no menu de configurações e num subquadro de uma retrospetiva dividida, o responsável ou um membro da respetiva equipa com permissões, pode alterar o responsável,

escolhendo outro participante a partir da opção para o efeito. Ao efetuar o pedido para realizar a alteração no *backend* e se for realizado com sucesso, é criado o *job* e é adicionado à respetiva fila de trabalhos para mudança de responsável, que tem a responsabilidade de enviar uma mensagem para o canal principal das retrospetivas no *slack*, para o canal de responsáveis e para o canal da respetiva subequipa, anunciando a mudança de responsável. Seguidamente, o novo responsável é adicionado ao canal dos responsáveis.

Um subquadro de uma retrospetiva dividida tem outras particularidades, com o intuito de replicar o processo de retrospetivas da xgeeks. Como tal, num subquadro não é possível votar, uma vez que esse momento é apenas efetuado no quadro principal após todos os cartões dos subquadros se encontrarem no quadro principal e quando terminar a reunião dos responsáveis.

A segunda particularidade diz respeito a um botão extra presente num subquadro. Após cada subequipa ter os seus cartões definidos, agrupados por assunto, compreendidos e a reunião terminada, os cartões devem ser movidos para o quadro principal e para tal existe um botão "*Merge into main board",* apenas visível para o responsável ou membro da equipa com permissões, como é possível observar na zona esquerda da Figura 46.

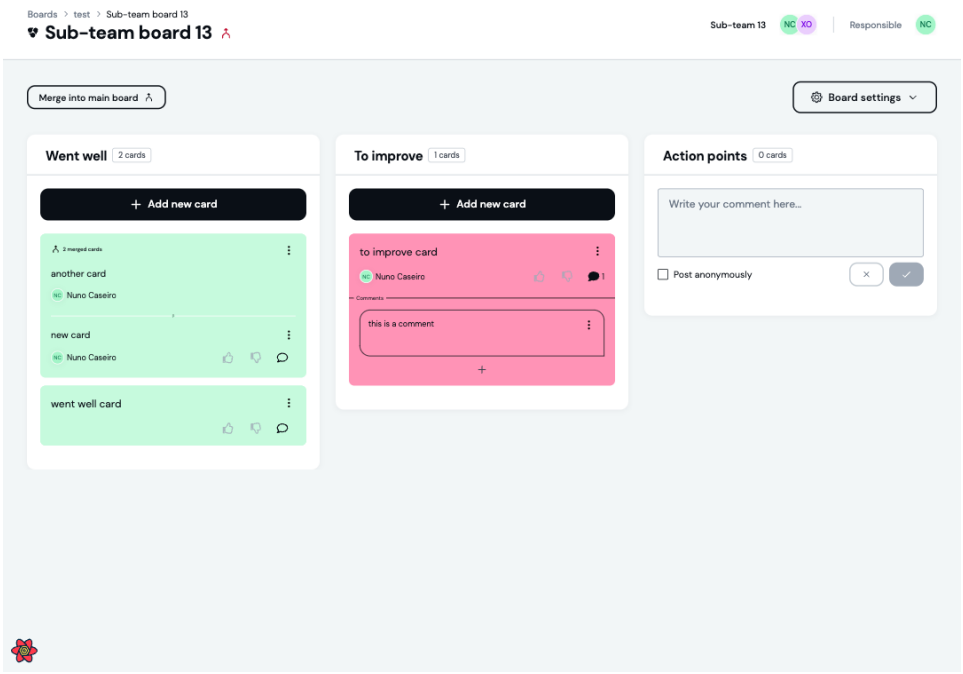


Figura 46 - Representação de um subquadro de uma retrospetiva dividida - quadro da equipa número 13

Após a reunião da subequipa, o responsável deve carregar no botão para efetuar o *merge* e ao fazê-lo é efetuado um pedido ao *backend,* que faz uma cópia dos cartões do subquadro

para o principal, e o subquadro passa a encontrar-se no estado submetido. Neste caso é considerado como bloqueado no *frontend* e por isso, não é possível efetuar qualquer alteração. Se a operação associada ao pedido for bem-sucedida, o *backend* cria um *job* que é adicionado à fila responsável por tratar de enviar mensagens quando se fundem os quadros. Assim sendo, este *job* trata de enviar uma mensagem no *slack,* no canal dos responsáveis, indicado que uma determinada subequipa já se reuniu e como tal efetuou o *merge* do quadro. No *merge* do último subquadro é enviada uma mensagem adicional indicando que todas as subequipas já se reuniram e como tal pode proceder-se à fase seguinte, que diz respeito à reunião dos responsáveis, onde vão agrupar os cartões das várias equipas que dizem respeito ao mesmo assunto e discussão de algum cartão se necessário. Finalizando a reunião, dá se inicio à fase de votação, na qual os utilizadores participantes podem aceder ao quadro principal e votar nos cartões mais relevantes. Após esta fase, decorre a reunião entre os *stakeholders* e responsáveis, na qual são determinados os itens de ação, com o intuito de solucionar o que correu menos bem.

Na Figura 47 é possível observar a representação de um quadro principal de uma retrospetiva dividida e no topo da página é possível verificar quais os quadros que foram fundidos no principal.

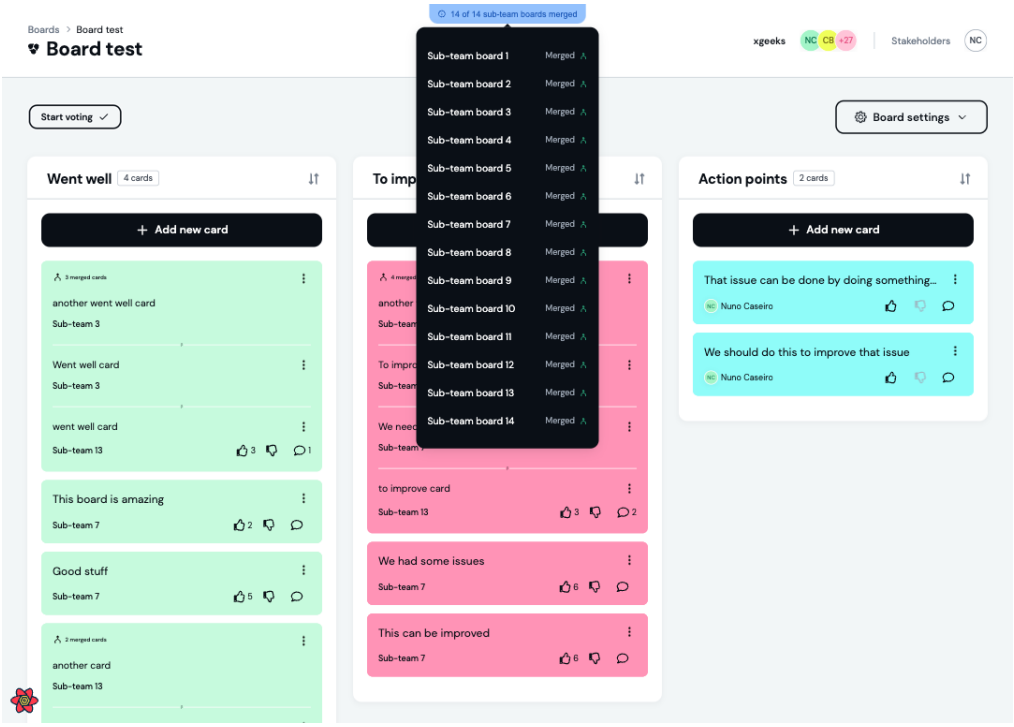


Figura 47 - Representação do quadro principal de uma retrospetiva dividida

Após a reunião entre os responsáveis, é possível iniciar a fase de votação ao pressionar no botão para o efeito "Start voting". O botão encontra-se do lado esquerdo da Figura 47 e

pode ser utilizado para efetuar um pedido ao *backend* para atualizar a fase do quadro e ao fazê-lo, também é criado um *job* que por sua vez é submetido na respetiva fila, responsável por tratar de trabalhos de envio de mensagens. Assim quando este *job* poder ser resolvido, é enviada uma mensagem no canal principal de retrospetivas do *slack*, onde é indicado que se deu início à fase de votação e é partilhado o *link* do quadro principal. Uma vez alterada a fase, na interface, em vez de se encontrar no topo da página a descrição de que todos os quadros se encontram fundidos é indicado o nome da fase: "*Voting phase*".

Após a fase de votação, decorre a reunião entre os responsáveis e *stakeholders,* onde se definem as ações a tomar para melhorar o que não correu tão bem, como é possível verificar na coluna de *action points* na Figura 47. Finalizada esta fase e com o intuito de bloquear a edição do mesmo, um dos responsáveis pode pressionar no botão para o efeito "*Submit board*", e que por sua vez, promove um pedido ao *backend* para atualização da fase e envio de uma mensagem no *slack,* anunciando que a retrospetiva terminou e são apresentados os *action points* gerados.

6.6.8. Página de gestão de quadros

A página dos quadros, representada na Figura 48, e acessível a partir do segundo botão na barra lateral de navegação, possibilita a visualização de todos os quadros em que o utilizador participa, mas também aqueles associados ao quadro principal e nesta listagem não existe restrição temporal como no *dashboard*.

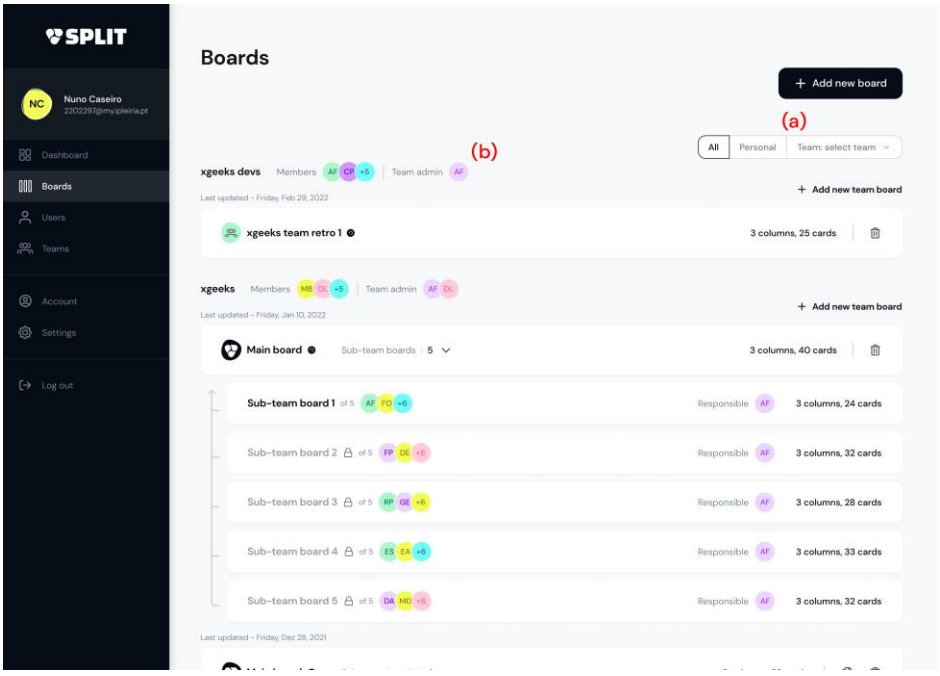


Figura 48 - Representação da página dos quadros

Na secção Figura 48 (a) é apresentada uma zona onde é possível filtrar a lista dos quadros. O utilizador pode selecionar todos os quadros, os pessoais, ou filtrar por equipa. Na secção (b) pode-se visualizar os quadros separados por equipas ou pessoais e ordenados por data descendente de atualização. Nos quadros divididos, o utilizador pode visualizar a lista dos subquadros associados, no entanto, se não tiver permissões para os visualizar estes encontram-se com acesso bloqueado. Na listagem dos quadros, tal como na página do *dashboard,* existe um mecanismo de *lazy loading* de tal forma que, à medida que o utilizador faz *scroll* pela secção, são pedidas as páginas seguintes, caso existam.

Finalmente, um utilizador com permissões pode eliminar um quadro através do botão representado por um caixote do lixo.

6.6.9. Gestão de utilizadores

Na página dos utilizadores representada na Figura 49, é possível visualizar os utilizadores registados e alguma informação relevante. Nesta secção também foi implementado o mecanismo de *lazy loading,* à medida que o utilizador efetua *scroll*. Um utilizador na plataforma pode ser super-administrador ou membro. A imagem é a vista do super-administrador e este pode definir outros utilizadores como super-administradores, pode eliminá-los da plataforma ou visualizar as equipas em que o membro participa. Para as visualizar, pode-se carregar no nome do utilizador ou no botão que apresenta um ícone representativo de edição. No ecrã de listagem das equipas em que o utilizador participa é possível alterar a *role* do mesmo.

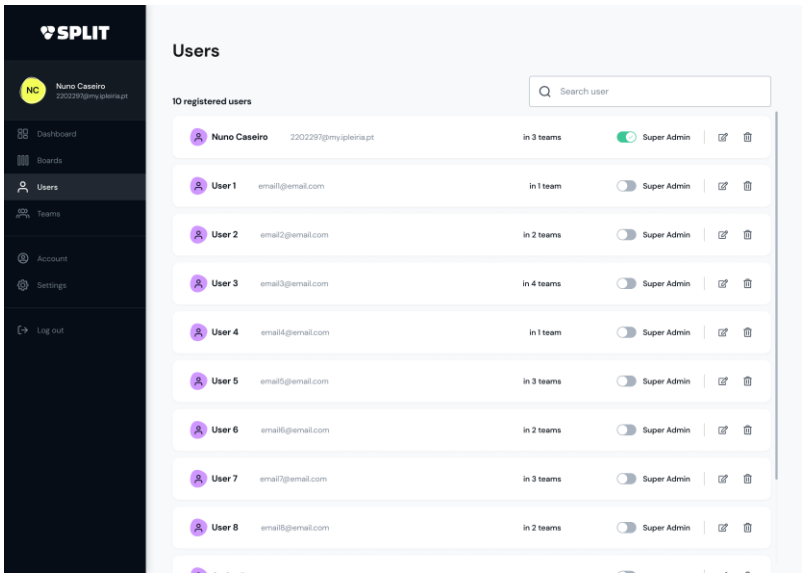


Figura 49 - Representação da página de gestão de utilizadores

Os dados do utilizador quando são apagados da plataforma não são removidos, porque a operação é feita através de um *soft delete,* para ser possível manter o histórico do utilizador e manter os seus dados acessíveis num quadro em que tenha participado, no entanto, é removido de todas as equipas em que participa.

Um membro sem permissões pode visualizar o nome e email dos utilizadores, em quantas equipas participa e os respetivos nomes.

6.6.10. Gestão de equipas

A página das equipas tem o propósito de proporcionar ao utilizador um modo de gerir ou visualizar as equipas em que o utilizador participa, como é possível visualizar na Figura 50. Como referido, um utilizador numa equipa pode ser membro, administrador ou *stakeholder* sendo que este último tem os mesmos privilégios que o anterior. Na página apresentada na figura seguinte, um administrador pode visualizar e apagar as suas equipas enquanto os membros apenas podem visualizá-las.

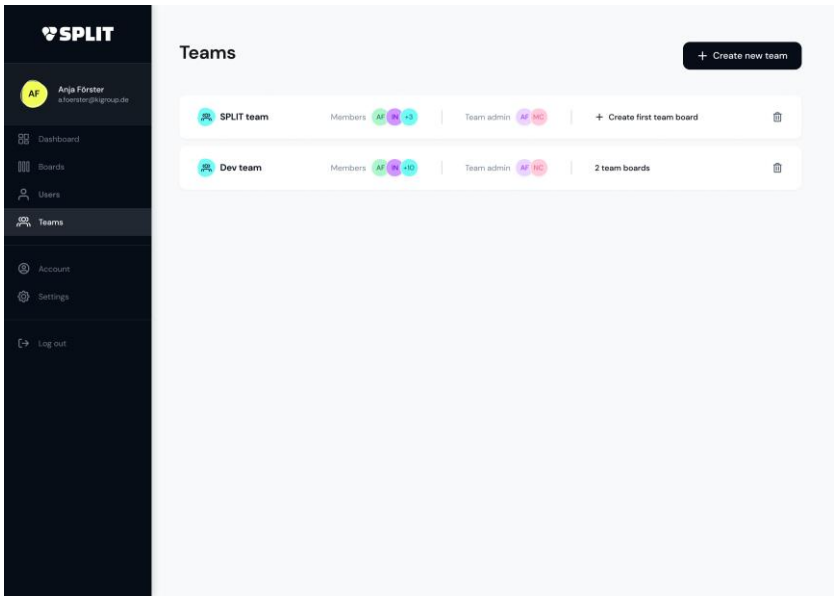


Figura 50 - Representação do ecrã de gestão de equipas

Na Figura 50, no canto superior direito, observa-se um botão que redireciona o utilizador para a página de criação de equipas.

A página de criação de equipas é representada pela Figura 51, e nesta é possível verificar que pode ser atribuído o nome a uma equipa e pode-se adicionar ou remover utilizadores pressionando o respetivo botão, que invoca uma vista lateral, a qual dispõe da listagem de utilizadores da plataforma que podem ser selecionados para pertencer à equipa.

Selecionando os utilizadores, estes surgem listados, como é possível observar na Figura 51. É importante referir que, para replicar o caso de uso da xgeeks, foi necessário adicionar a opção para definir se um utilizador ingressou na empresa há menos de 3 meses, e por isso são considerados *new joiners*. Então, uma vez que se deve ter conta se um utilizador ingressou na equipa ou na empresa há mais de três meses, surge a necessidade de se definir se o utilizador deve ser considerado como *new joiner* e assim, não pode ser selecionado como responsável de um subquadro numa retrospetiva dividida. Esta opção é automaticamente definida de duas formas: caso o utilizador tenha sido autenticado por SSO, a data de criação da conta (no *provider*) é guardada e serve para efeitos de contagem para a definição desta opção. Caso o utilizador, se registe por credenciais, a data de criação da conta na base de dados é aquela utilizada para efeitos de contagem. Após adição dos utilizadores na equipa, antes da criação na base de dados, o utilizador pode definir a *role* de cada utilizador e verificar o estado de *new joiner*

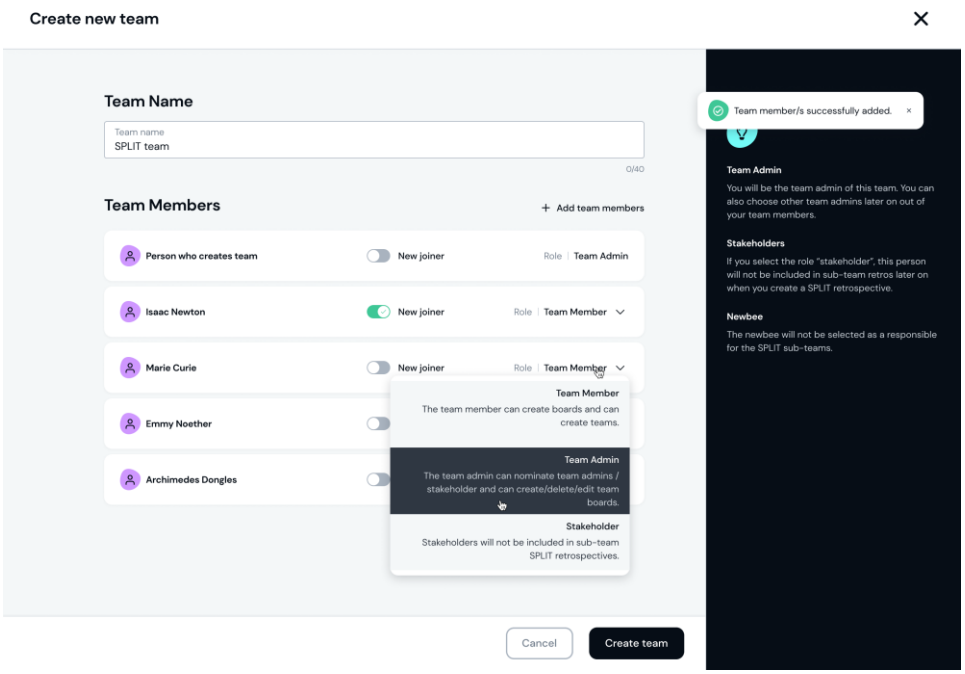


Figura 51 - Representação da página de criação de equipas

Uma vez criada a equipa, o utilizador é redirecionado para a página de detalhes da mesma, como é possível verificar a sua representação na Figura 52. Nesta página, também acessível a partir da página da listagem de equipas, um administrador ou *stakeholder* pode adicionar, remover ou editar a *role* de um utilizador na equipa. Um membro da equipa sem privilégios pode visualizar os participantes da mesma e as suas *roles*.

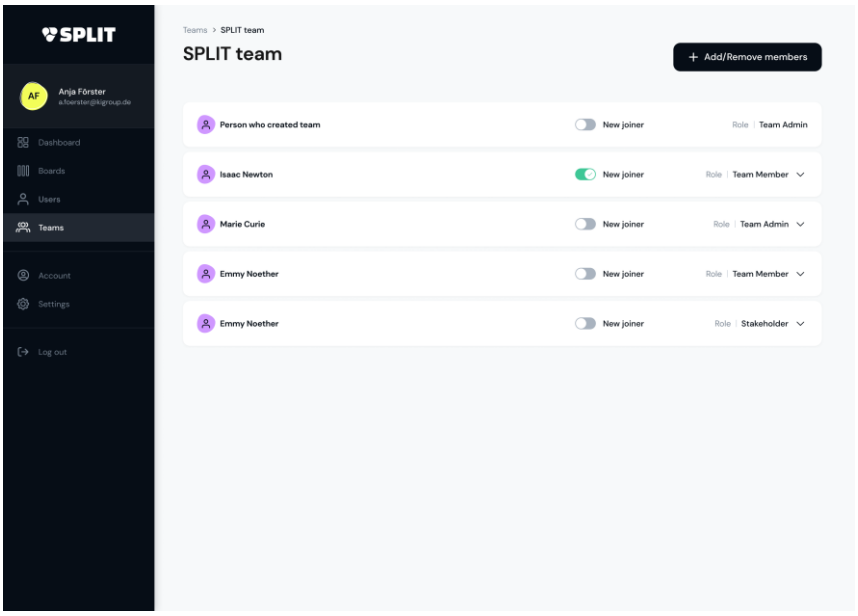


Figura 52 - Representação da página de detalhes de uma equipa

6.7.CI/CD e infraestrutura

A infraestrutura é essencial para a disponibilização da aplicação na internet e para tal é importante construir uma *pipeline* de *continuous integration and continuous delivery* (CI/CD) que automatiza o processo de integração, validação e disponibilização do código da aplicação na infraestrutura. A infraestrutura compreende-se por *software*, *hardware* e processos que suportam a disponibilização de aplicações.

Como é descrito na documentação da AWS [50] , *continuous integration* (CI) é uma prática de desenvolvimento de software onde os membros de uma equipa utilizam um sistema de controlo de versões e integram frequentemente o seu trabalho na mesma localização como por exemplo o *main branch* de um repositório. Antes da integração numa dada localização, cada modificação no código é verificada e a aplicação é construída com as novas alterações para ser validada, detetando possíveis erros e, por isso, pode-se dizer que a CI é focada na automatização do *build* da aplicação e execução dos testes. *Continuous delivery* é uma metodologia de desenvolvimento de software onde o processo de disponibilização da aplicação é efetuado de forma automática. Cada alteração é automaticamente testada, validada, a aplicação é construída e disponibilizada nos diferentes ambientes como desenvolvimento, *staging* e produção. Antes do último *push* para produção, por norma existe um mecanismo manual ou automático que valida se as alterações devem ser enviadas para produção.

Localmente, foi desenvolvido um ficheiro *docker compose*[35] e pode ser observado na Figura 53, que é composto por seis serviços, e estes compreendem-se pelo serviço do *backend*, *frontend*, a base de dados *mongo* e a sua réplica, a base de dados *redis,* mas também o azurite[36] que simula localmente o repositório de *assets* da Microsoft. Cada serviço permite construir o respetivo container em *docker* com o intuito de correr cada uma das aplicações. No serviço de *backend* e *frontend* é indicado a localização do *dockerfile* implementado, que dentro do respetivo container é executado e, portanto, as dependências são instaladas, é efetuado o *build*, a aplicação é iniciada e é exposto o respetivo *port*. Os *dockerfile* encontram-se na raiz da pasta de cada aplicação e podem ser observados no Apêndice O – Dockerfiles. As restantes aplicações não necessitam desse ficheiro.

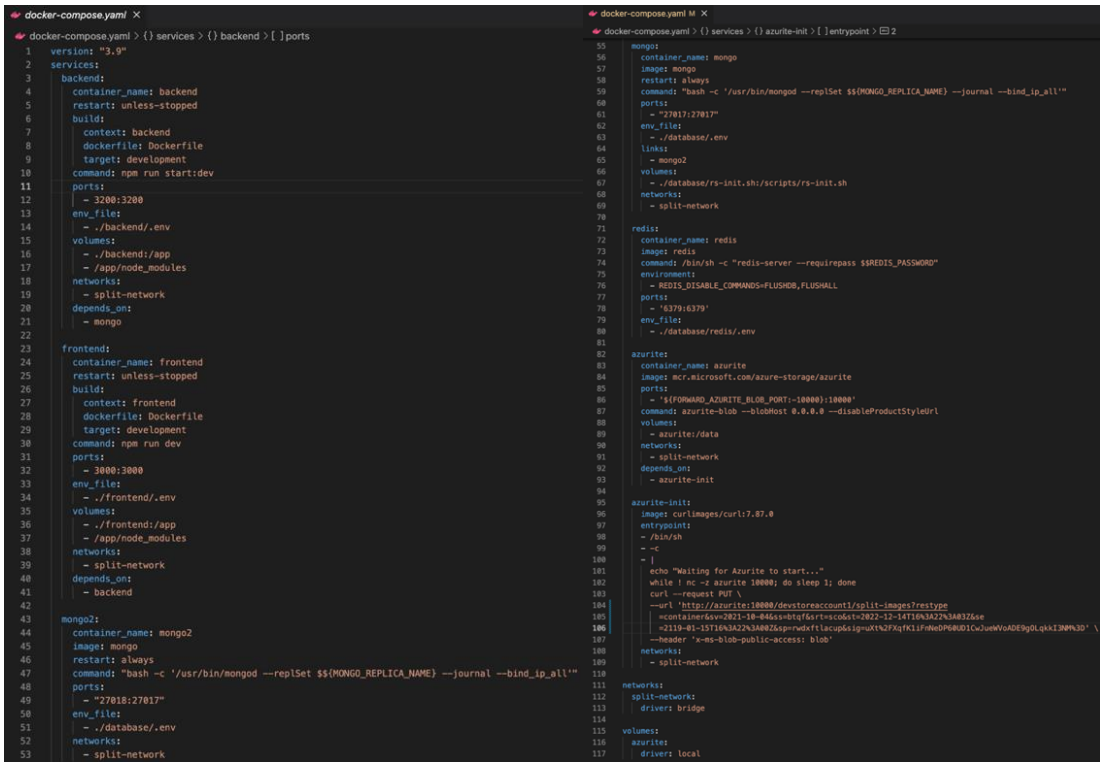


Figura 53 - Representação do ficheiro docker-compose.yaml

A infraestrutura do SPLIT foi implementada nos serviços *cloud* da Microsoft *azure* e todos os recursos criados podem ser observados na Figura 54. Foi definido que numa fase inicial que o projeto deveria ter apenas dois ambientes e, portanto, o ambiente de desenvolvimento e o ambiente de produção.

[35] https://docs.docker.com/compose/
[36] https://learn.microsoft.com/en-us/azure/storage/common/storage-use-azurite?tabs=visual-studio

Name	Type	Location	Resource Group
split-be-dev	App Service	West Europe	SPLIT
split-be	App Service	West Europe	SPLIT
split-dev	Azure Cache for Redis	West Europe	SPLIT
splitstorageaccount	Storage account		SPLIT
SPLIT	Resource group		SPLIT
split-db	Azure Cosmos DB for MongoDB account	West Europe	SPLIT
split-fe-dev	App Service	West Europe	SPLIT
KI-GRP - Internal Project - SPLIT	Subscription		
split	Container registry	West Europe	SPLIT
split	Azure Cache for Redis	West Europe	SPLIT
ASP-SPLIT-b124	App Service plan	West Europe	SPLIT
split-fe	App Service	West Europe	SPLIT
split-dev	App Service plan	West Europe	SPLIT

Figura 54 - Lista de recursos alojados na Microsoft Azure Cloud services

Primeiramente, foi definido um *resource group* denominado de SPLIT, que é um contentor que armazena os recursos relacionados com a solução e nele são adicionados os serviços que partilham o mesmo ciclo de vida.

Como tal, para cada ambiente (desenvolvimento e produção) foi criado uma *app service plan* que representa um conjunto de recursos de computação que podem ser utilizados por uma ou mais aplicações.

Associado a cada *service plan,* foram criados dois *app services,* um para o *frontend* e outro para o *backend* e neles foram definidas as variáveis de ambiente. Um *app service* é um serviço baseado em HTTP que permite alojar aplicações web, REST APIs, etc. É gerido automaticamente pela Microsoft, permite *autoscaling*, usufruir de *load balacing* e também é seguro.

As aplicações, como referido anteriormente, podem ser construídas a partir de *docker images*. Como tal, as apps *services* suportam a execução e disponibilização de aplicações a partir de imagens *docker*, e suportam *continuous deployment* através das *github actions,* o que é uma mais-valia para este projeto. Com o intuito de armazenar as imagens *docker* criadas para cada aplicação e a partir das quais são executadas no *app service*, criou-se um *container registry* que se compreende por um repositório de imagens *docker* que podem ser utilizadas pelos *app services.* Portanto, estes serviços utilizam as imagens armazenadas no *container registry* para executar e disponibilizar as respetivas aplicações.

A base de dados de cada aplicação foi criada com recurso ao *Azure Cosmos DB*. Este serviço é totalmente gerido pela Microsoft e oferece tempos de resposta baixos, escalabilidade instantânea e alta disponibilidade. Pode ser considerado como um cluster e permite criar várias bases de dados e como tal, foram criadas duas, uma para cada ambiente.

Para armazenar e disponibilizar ficheiros, foi criado uma *storage account* que permite criar contentores para os quais se pode enviar ficheiros e que são automaticamente disponibilizados através de um *link*. Este serviço tem o propósito de armazenar as imagens dos utilizadores que se autenticam por SSO.

Para finalizar os recursos criados e com o intuito de possibilitar a utilização de filas no *backend,* para cada ambiente, foram criadas duas instâncias (uma para cada ambiente) da base de dados Redis através do serviço *Azure Cache for Redis*.

No que diz respeito à *pipeline* de CI/CD, esta foi implementada utilizando as *github actions.* Esta é uma plataforma de CI/CD proporcionada pelo *Github* que permite automatizar o processo de construção, testes e disponibilização das aplicações. Para tal, é possível criar *workflows* constituídos por tarefas que podem efetuar o *build* das aplicações, testar o código em cada *pull request*, disponibilizar e armazenar novas imagens *docker* num *container registry*, etc.

No projeto SPLIT existe um mecanismo construído através da biblioteca *husky*[37], que em cada tentativa de *commit* é verificado se a mensagem apresenta o formato sugerido nos *conventional commits* e se não existem problemas de formatação no código sugeridos pelo *eslint* – responsável por verificar se o código segue as regras definidas. Se algum caso falhar, o *commit* não é efetuado até ser resolvido.

Em relação à construção da pipeline de CI&CD foram criados 4 *workflows* que se encontram na pasta *.github* do projeto e cada um tem a sua responsabilidade. Antes de os apresentar, é importante referir que foi definido que apenas é possível efetuar o *merge* de um *branch* para o *main* se todas as validações forem bem-sucedidas e se pelo menos um *reviewer* o considerar como aprovado.

O primeiro *workflow* trata de validar o título do *pull request* quando este é criado, verificando se segue as boas práticas sugeridas no *conventional commits*. Se for criado um *pull request* para *main,* outro *workflow* é despoletado e é composto por vários passos. Este é responsável por instalar as dependências, efetuar o *build* de cada aplicação, e executar os testes em cada uma destas, como é possível observar na Figura 55 . O relatório gerado na execução dos testes é associado ao *workflow* e pode ser visualizado.

[37] https://github.com/typicode/husky

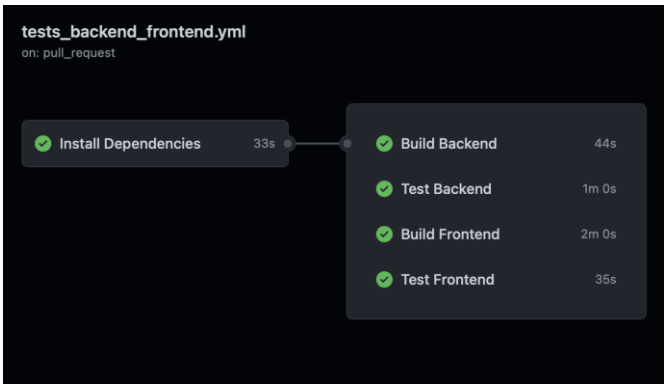


Figura 55 - Representação do *workflow* de *build* e execução de testes num *pull request*

Se os testes passarem com sucesso e se o *pull request* for aprovado e fundido no *main branch,* outro *workflow* é despoletado e é responsável por efetuar o *build* das aplicações, criar a imagem *docker* para cada uma – caso exista em cache, estas podem ser reutilizadas – é também responsável por efetuar o *push* das imagens das aplicações para o *container registry* e consequentemente atualizar os *app services* do ambiente de desenvolvimento, de forma a utilizarem e correrem as imagens recentemente enviadas e atualizadas. Após este passo, o *workflow* entra em estado pendente, porque também é responsável pela disponibilização no ambiente de produção, no entanto não o faz de imediato. É criada uma tarefa que precisa de aprovação manual por parte dos administradores do SPLIT, que podem dar permissões para a disponibilização em produção quando necessário. Depois de publicar em *dev* é executado outro passo, que cria uma *pre-release,* indicando quais são as *changes* incluídas na versão atual. Quando a tarefa manual é aprovada e por isso as imagens do *docker* podem ser reutilizadas pelo ambiente de produção, o *workflow* que se encontrava pendente continua e finaliza a sua execução, atualizando os *app services* do ambiente de produção para tirarem partido das imagens *docker* recentemente enviadas. O último passo é responsável por gerar uma *release* no *github* e publicar os *change log*s. O *change log* é composto por várias secções como "*what changed*", "*features*", "*bug fixes*" e "*documentation*". Cada secção é preenchida com as descrições dos *pull requests,* aos quais foi associado a respetiva *tag* da secção. Cada um dos passos descritos pode ser observado na Figura 56 e um exemplo de *change log* no Apêndice P – Exemplo de *change log* (v0.1.12).

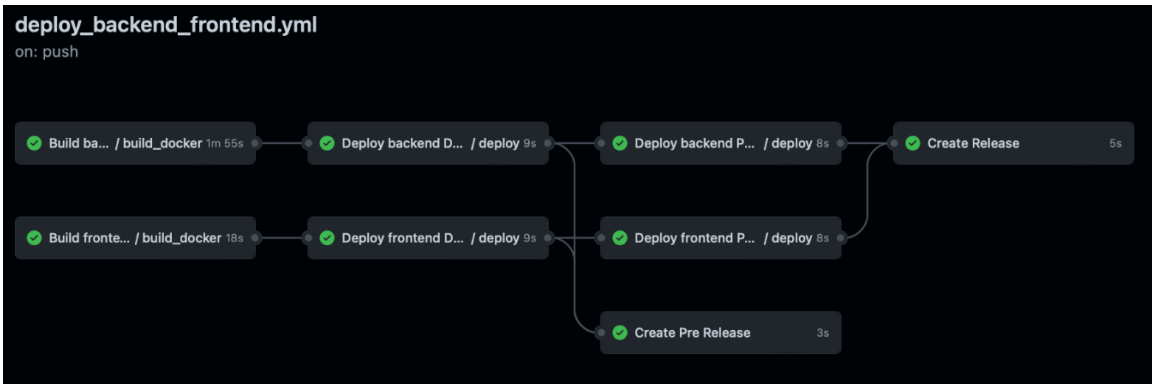


Figura 56 - Representação do workflow de disponibilização das aplicações nos respetivos ambientes

Finalmente, o último *workflow* apenas é despoletado quando alguma *release* é disponibilizada e este é responsável por atualizar as versões das aplicações, incrementando-as, nos ficheiros *package.json*.

Se algum *workflow* falhar é automaticamente enviado um email com os erros encontrados.

Ainda no âmbito de infraestrutura, importa referir que a *active directory* da Microsoft *azure,* que é responsável por gerir as autenticações via email da empresa, já se encontrava implementada pela equipa de *information technology* (IT) da xgeeks empresa, no entanto, foi necessário registar a aplicação SPLIT nesse serviço e indicar os *URLs* de *callback* das aplicações de *frontend* para permitir a autenticação dos utilizadores no SPLIT através do email da empresa. Adicionalmente, como o *backend* é responsável por validar o *token* enviado pelo *frontend*, necessita de efetuar pedidos á *Microsoft GRAPH API* e para tal foi necessário definir algumas permissões de leitura dos perfis dos utilizadores.

Finalmente, em relação aos domínios das aplicações, para definir que o URL do *frontend* deva incluir o sufixo *kigroup.de*, foi necessário criar dois subdomínios, um para o ambiente de *dev* e outro para o ambiente de produção. Após a criação dos mesmos foi necessário associar o endereço (IP) de cada aplicação (*app service*) ao respetivo subdomínio criado. O resultado é apresentado na Tabela 6 onde se verificam os URLs para cada aplicação. Esta é composta por três colunas, na primeira é indicado o ambiente, a segunda diz respeito à aplicação *frontend* e a última ao *backend.* Na aplicação *frontend* podem verificar-se os URLs da aplicação e do *storybook* para cada ambiente. Na aplicação do *backend* pode-se verificar os URLs da aplicação e do *swagger.*

Tabela 6 - Hiperligações relevantes para cada aplicação disponibilizada na respetiva infraestrutura

Ambiente	*Frontend*		*Backend*	
	App	Storybook	App	Swagger
Desenvolvimento	https://dev.split.kigroup.de	https://dev.split.kigroup.de/storybook	https://split-be-dev.azurewebsites.net	https://split-be-dev.azurewebsites.net/docs
Produção	https://split.kigroup.de	https://split.kigroup.de/storybook	https://split-be.azurewebsites.net	https://split-be.azurewebsites.net/docs

7. Testes e resultados

O presente capítulo tem o propósito de apresentar os testes efetuados e os resultados obtidos, mas também o estado atual do produto e de que forma está a ser utilizado na empresa. Portanto, na secção 7.1 são apresentados os testes unitários, manuais e de usabilidade efetuados e são descritos os resultados obtidos. Na secção 0 é descrito o estado atual do projeto e como este está a ser utilizado na empresa.

7.1. Testes

A secção de testes tem como objetivo apresentar os testes unitários, manuais e de usabilidade efetuados bem como os resultados obtidos.

7.1.1. Unitários e manuais

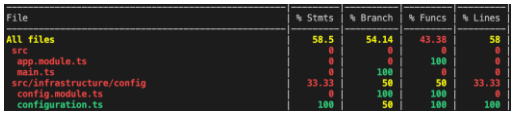
Durante o desenvolvimento das aplicações e da construção da infraestrutura foram efetuados testes manuais com o intuito de compreender se as funcionalidades se encontram bem desenvolvidas, através da comparação do resultado esperado com o obtido. Estes testes foram efetuados essencialmente através da impressão de resultados no ecrã. A mesma técnica serviu para efetuar *debug* nas aplicações, imprimindo a informação na consola do *browser* ou na linha de comandos.

Numa fase inicial, foram implementados alguns testes unitários no *backend* através da ferramenta *jest,* no entanto, a cobertura não foi a desejável, tendo sido obtido apenas 30% no *backend.* Estes valores devem-se ao facto de os testes não terem sido implementados desde início e à medida que as funcionalidades foram desenvolvidas. Com o intuito de aumentar a velocidade de desenvolvimento, não foi definido como obrigatoriedade a implementação de testes para as funcionalidades desenvolvidas.

Esta decisão teve certas implicações com o escalar do projeto, nomeadamente na dificuldade em efetuar o *tracking* das alterações efetuadas. Isto significa que se tornou difícil de verificar se as novas alterações no código afetaram as funcionalidades já desenvolvidas. Posteriormente, esse método foi revertido e, portanto, o desenvolvimento de testes unitários foi definido como alta prioridade. Deste modo a equipa dividiu-se, sendo que alguns elementos se focaram na execução de testes para as funcionalidades antigas (*backend* e

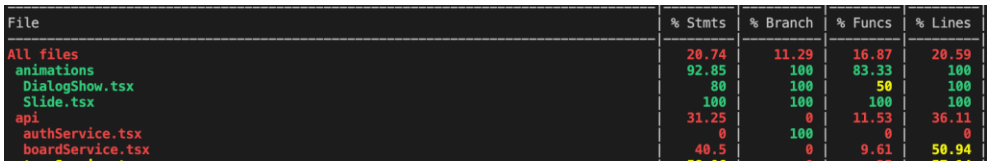
frontend), e outros elementos foram alocados ao desenvolvimento de novas funcionalidades, no entanto os testes devem acompanhar essas funcionalidades.

Após esta alteração verificou-se um incremento de cobertura de código, na medida em que o *backend* tem um conjunto de testes unitários que cobrem cerca de 59% do código e o *frontend* 20% como se pode observar na Figura 57.



File	% Stmts	% Branch	% Funcs	% Lines
All files	58.5	54.14	43.38	58
src	0	0	0	0
app.module.ts	0	0	100	0
main.ts	0	100	0	0
src/infrastructure/config	33.33	50	50	33.33
config.module.ts	0	100	100	0
configuration.ts	100	50	100	100

(a)



File	% Stmts	% Branch	% Funcs	% Lines
All files	20.74	11.29	16.87	20.59
animations	92.85	100	83.33	100
DialogShow.tsx	80	100	50	100
Slide.tsx	100	100	100	100
api	31.25	0	11.53	36.11
authService.tsx	0	100	0	0
boardService.tsx	40.5	0	9.61	50.94

(b)

Figura 57 - Representação dos resultados dos testes unitários ao *backend* (a) e *frontend* (b)

Numa fase inicial, os testes manuais foram suficientes para identificar e resolver problemas, no entanto, com o escalar do projeto, verificou-se que a falta de testes automáticos diminuiu a qualidade do código na medida em que, apenas após o *merge* para *main,* é que se verificaram alguns bugs desconhecidos, o que poderia ter sido minimizado se os testes tivessem realizados e executados. Esta situação pode-se definir como uma aprendizagem, uma vez que se verificou claramente, numa fase posterior, a utilidade na execução de testes.

Visualizando os resultados atuais no que diz respeito à cobertura atual do código pelos testes, é possível indicar que ainda existe um longo caminho para atingir um valor superior a 80% em ambas as aplicações, contudo é um trabalho em desenvolvimento e os objetivos serão cumpridos com o continuar do desenvolvimento do projeto.

7.1.2. Usabilidade

No sentido de testar a usabilidade do sistema e em especial a sua interface foi desenvolvido um formulário com tarefas a executar e questões associadas. O formulário construído é possível de observar no Apêndice Q – Formulário para testes de usabilidade e pretendeu-se que fosse preenchido de forma assíncrona por pelo menos 15 membros da empresa. O formulário é constituído por 9 secções e cada uma diz respeito a uma página da aplicação, no início das quais é solicitada a execução de algumas tarefas e de seguida são apresentadas questões obrigatórias de sim ou não e de classificação de 1 a 5. No final de

cada secção existe uma questão de resposta aberta onde os utilizadores podem fornecer os seus comentários livremente sobre a página e funcionalidades em análise.

Nesta secção do documento pretende-se apresentar, discutir e analisar as respostas obtidas no formulário. É de realçar que os participantes não tiveram qualquer contacto prévio com o sistema até à realização dos testes, o que possibilita, com fiabilidade, definir se a interface disponibilizada é fidedigna, intuitiva e fácil de utilizar. Infelizmente não foi possível reunir 15 respostas, no entanto reuniram-se 11 e todos os resultados obtidos são apresentados no Apêndice R – Resultados dos testes de usabilidade.

Pretendia-se que este processo fosse efetuado através de entrevistas e acompanhamento em tempo real na execução do guião de testes de usabilidade, no entanto devido à falta de tempo optou-se pela execução assíncrona.

No que diz respeito aos dispositivos utilizados para realização dos testes, todos os utilizadores usaram um computador com um *browser* instalado.

Na Tabela 7 é possível verificar algumas das respostas obtidas para cada questão disponibilizada no formulário e estas encontram-se separadas por secção. Observando a tabela da esquerda para a direita, a primeira coluna indica o número da questão, a segunda coluna pretende resumir o resultado das respostas sendo que, pode ser apresentada a média dos resultados para essa questão ou então apresentadas as opiniões dos utilizadores quando a pergunta solicita *feedback* escrito. A última coluna pretende apresentar a análise efetuada aos resultados obtidos para cada questão e/ou as ações tomadas para resolver algum *feedback* menos positivo, se assim se aplicar. Com o intuito de não tornar exaustiva a apresentação dos resultados, na tabela seguinte, apenas são apresentadas as questões mais relevantes nomeadamente a penúltima e ultima questão de cada secção. A penúltima pretende que seja feita uma avaliação geral à funcionalidade ou página em análise. A última questão pretende recolher comentários adicionais para cada secção. No Apêndice S – Análise completa dos resultados, pode-se observar a análise completa dos resultados obtidos.

Tabela 7 - Análise dos resultados dos testes de usabilidade

Q.	Resultados	Análise/Ações
Secção 1 – *Sign up / login flow*		
1	Média - 4.91.	Fluxo de registo fácil e intuitivo.

2	- Solicitam-se mais *providers* de autenticação - SSO facilita o processo de autenticação - No registo, o *feedback* dado ao utilizador poderia ser melhor.	- A disponibilização dos *providers* depende de quem aloja o SPLIT. - O *feedback* foi tido em conta, foi discutido pela equipa e originou um novo *issue* no github.
Secção 2 – Página do *dashboard*		
6	Média – 4.82.	- Pode-se assumir que a interface da página foi bem conseguida.
7	- O objetivo da informação relativa aos membros ativos não é claro. - Elogios à interface.	- Necessidade de melhorar a secção das estatísticas para clarificar o seu propósito.
Secção 3 – Gestão/Criação de equipas		
9	Média – 4.64.	- O processo de criação de equipas é bastante acessível de efetuar e a interface é intuitiva.
10	- As permissões de um administrador e de um *stakeholder* não são claras. - Alteração de *dropdown* para *checkboxes* para editar o papel dos utilizadores. - Bloco de informação à direita pode ser estranho para outras culturas	- Verificam-se opiniões cujo foco não são os aspetos fulcrais das operações de gestão de equipas. - É relevante melhorar a informação que diz respeito aos papéis dos utilizadores numa equipa.
Secção 4 – Página para criação de um quadro para uma retrospetiva dividida		
17	Média: 4.73.	- Pode-se assumir que a interface para a criação de retrospetivas divididas foi bem conseguida.
18	- A opção para escolher um novo participante não é intuitiva. - É indicado como aspeto negativo que só é possível adicionar ou remover equipas com o botão "+" e "–".	- Verifica-se uma oportunidade de melhoria para a seleção de novos responsáveis. - A localização do botão do menu para definir o número máximo de equipas ou elementos pode necessitar de melhorias. - O *feedback* será tido em conta nas novas versões da aplicação.

Secção 5 – Página de listagem dos quadros		
21	- Os quadros poderiam ser apresentados sempre expandidos.	- Poder-se-á integrar a sugestão, no entanto apresenta baixa prioridade.
Secção 6 - Página de um subquadro		
28	Média: 4.73	- Pode-se concluir que as operações num subquadro são bastante intuitivas, o que demonstra que a interface e as funcionalidades implementadas são facilmente utilizáveis.
29	- Não foi fácil de descobrir como efetuar o *merge* de cartões quando a opção para os esconder se encontra ativa. - É indicado que o caracter "\n" do texto do *template* para a coluna "to improve" é descartado quando se inicia a escrita.	- Ambas as opiniões foram tidas em conta e a última foi incluída no *backlog* e encontra-se resolvida.
Secção 7 -Página do quadro principal		
34	Média: 4.82	- Apesar de se poder considerar que a interface da página é bem conseguida, é possível verificar alguns comentários e melhorias nas respostas à próxima questão.
35	- Necessidade ou dificuldade em visualizar quais os subquadros que já foram *merged.* - É indicado que seria útil ter identificado os próprios cartões. - É referido que as cores dos botões e do texto dos votos não é adequada.	- É possível visualizar quais os quadros que foram *merged* contudo esta informação pode não ser intuitiva de observar no quadro principal. - Verifica-se uma oportunidade de melhoria no que diz respeito à apresentação da informação relativa aos quadros que já forem *merged.*
Secção 8 – Criação de um quadro para uma retrospetiva regular		

38	Média: 4.73.	- Interface de criação de um quadro de retrospetiva regular é intuitiva e fácil de compreender
39	- É sugerida a possibilidade de criar quadros para retrospetivas divididas onde são definidos manualmente os utilizadores de cada subequipa. - Foi indicado que alternar entre a *tab* de seleção de utilizadores e configurações descarta as opções selecionadas	- O problema do descarte da informação escolhida quando se alterna entre *tabs* foi imediatamente resolvida após análise dos resultados.
Secção 9 – Quadro regular		
46	Média: 4.9	- Conclui-se que a interface e as funcionalidades relativas ao quadro de uma retrospetiva regular foram bem-sucedidas.
47	- Não é claro o propósito do temporizador e seria útil incluir uma breve descrição em algum lugar da página. - É indicado que o temporizador poderia ter algum tipo de configuração para associar a eventos/fases da retrospetiva.	- Estas observações foram tidas em conta e seguiram para análise pelas partes responsáveis, nomeadamente a *designer,* com o intuito de as solucionar.

Analisando os resultados obtidos e efetuando a média das questões referentes à classificação geral de cada secção obtém-se o valor de 4.81. Assim, é possível concluir que, segundo as respostas dos participantes no inquérito, a plataforma e a respetiva interface foram bem conseguidas, sendo que todas as páginas foram consideradas bastante intuitivas, e por isso pode-se considerar que oferecem uma excelente experiência de utilização. As oportunidades de melhoria foram tidas em conta, algumas das sugestões já se encontram resolvidas e isso é possível comprovar ao verificar alguns comentários submetidos na segunda retrospetiva efetuada na empresa e possível de observar na imagem do quadro no Apêndice U – Resultados 2ª retrospetiva.

Adicionalmente e antes do lançamento do produto alguns membros da empresa voluntariam-se para testar a aplicação, sendo que na última sessão de testes, antes da disponibilização em produção, foi possível reunir cerca de 30 colaboradores. A sessão foi síncrona e simulou-se o processo de retrospetivas divididas que se efetua na xgeeks. Serviu para testar a usabilidade da aplicação, mas também para testar a sua performance (*stress test*) no ambiente de desenvolvimento. Através desta reuniram-se um conjunto de melhorias, nomeadamente no que diz respeito à interação em tempo real quando muitos utilizadores se encontravam a realizar operações em simultâneo e no mesmo quadro. Em determinado momento solicitou-se aos 30 colaboradores que efetuem operações ao mesmo tempo e num só quadro. Nesse momento verificaram-se problemas de performance observando-se picos de processador e memória no *service plan* onde estava alojada a aplicação, como se pode observar na Figura 58.

Figura 58 - Resultados de performance face ao *stress test* – ambiente de desenvolvimento

Na figura apresentada anteriormente é possível verificar um pico de quase 100% do CPU o que corresponde ao momento em que foram efetuados diversos pedidos e em simultâneo, como é possível observar pelos dois gráficos da direita que representam a entrada e saída de dados.

A sessão revelou-se bastante útil porque, entre outras oportunidades de melhoria, verificou-se que a abordagem para notificar todos os utilizadores das alterações do quadro não era a melhor, o que levou à sua reimplementação e como tal foi possível otimizar a performance. Aliado a esta otimização foi decidido aumentar os recursos para o ambiente de desenvolvimento e produção.

7.2. Utilização atual do produto e testemunhos

O SPLIT encontra-se em produção e é utilizado pela *xgeeks* para efetuar as retrospetivas. Para tal foi criada a equipa na plataforma com o nome da empresa e foi pedido a todos os colaboradores para se registarem. Os utilizadores foram adicionados à equipa da empresa e deu-se início à sua utilização como ferramenta para a execução de retrospetivas.

No dia 1 de fevereiro de 2023 foi criado o primeiro processo de retrospetivas através da plataforma e esta retrospetiva é relativa ao mês de janeiro. Foi definido que o quadro principal deve ter por defeito os cartões escondidos e um limite de 6 votos por pessoa. A equipa da xgeeks constituída, nesse momento, por 102 pessoas, sendo que 6 são considerados *stakeholders*, foi dividida. Foram selecionados os responsáveis e os membros de cada subquadro, fazendo um total de 12 equipas, como é possível observar na Figura 59.

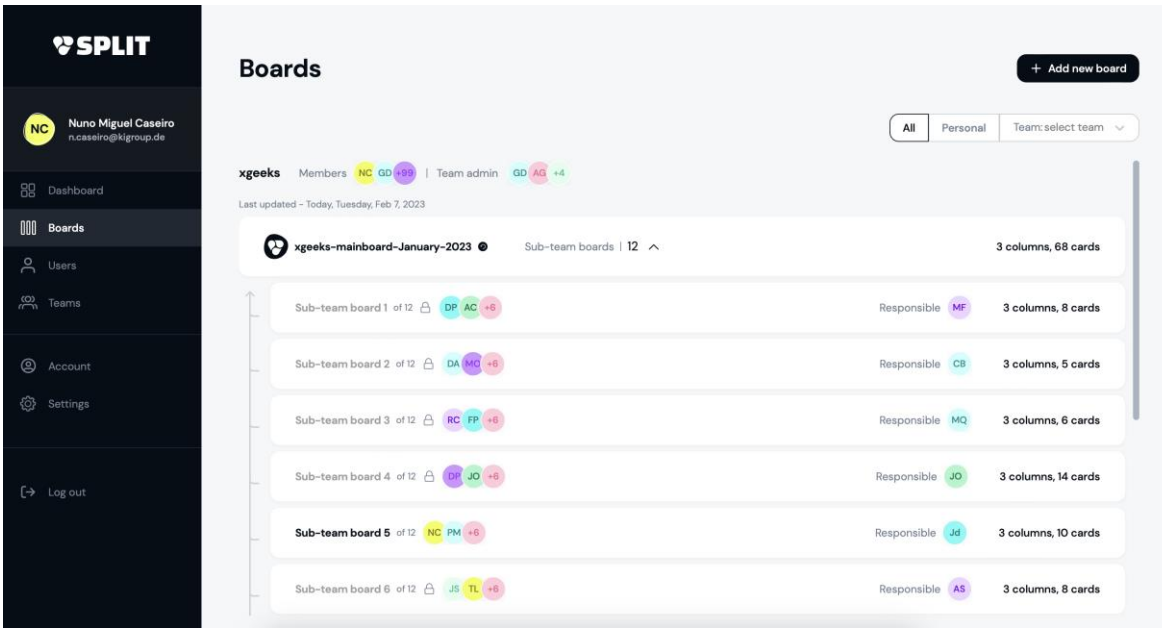


Figura 59 - Representação da retrospetiva de janeiro criada pela xgeeks

Aliado à criação dos quadros, foi gerado um *cron job,* agendado para o dia 21 de fevereiro, às 10 da manhã, para despoletar o processo de retrospetivas para o mês de fevereiro. Como a primeira retrospetiva foi gerada em fevereiro, mas relativa a janeiro, a data de agendamento automática seria dia 23 de março, portanto, esta foi alterada para o dia anteriormente referenciado, como é possível verificar na Figura 60.

Na figura é possível verificar o id do quadro principal gerado na primeira retrospetiva para serem copiadas as configurações na próxima retrospetiva, mas também é possível verificar o id da equipa xgeeks que serve para esta voltar a ser dividida. Na figura verifica-

se também o número de utilizadores que cada equipa deve conter, assim como a data de quando o processo será recriado.

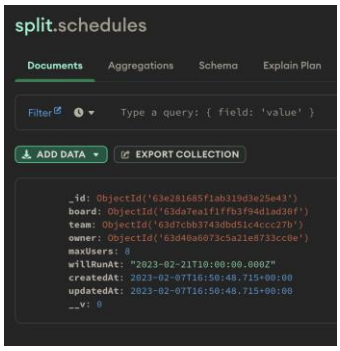


Figura 60 - Registo do agendamento criado na primeira retrospetiva do ano

Finalmente, foi enviada uma mensagem no canal principal das retrospetivas do *slack* da empresa e foram criados os canais para cada subequipa, outro para os responsáveis e os membros inseridos em cada um dos respetivos canais.

Na Figura 61 é possível observar a mensagem enviada no canal principal do *slack* da xgeeks. Esta mensagem foi gerada e enviada na criação da primeira retrospetiva, no dia 1 de fevereiro, e é possível observar a divisão da equipa em subequipas e a constituição de cada uma.

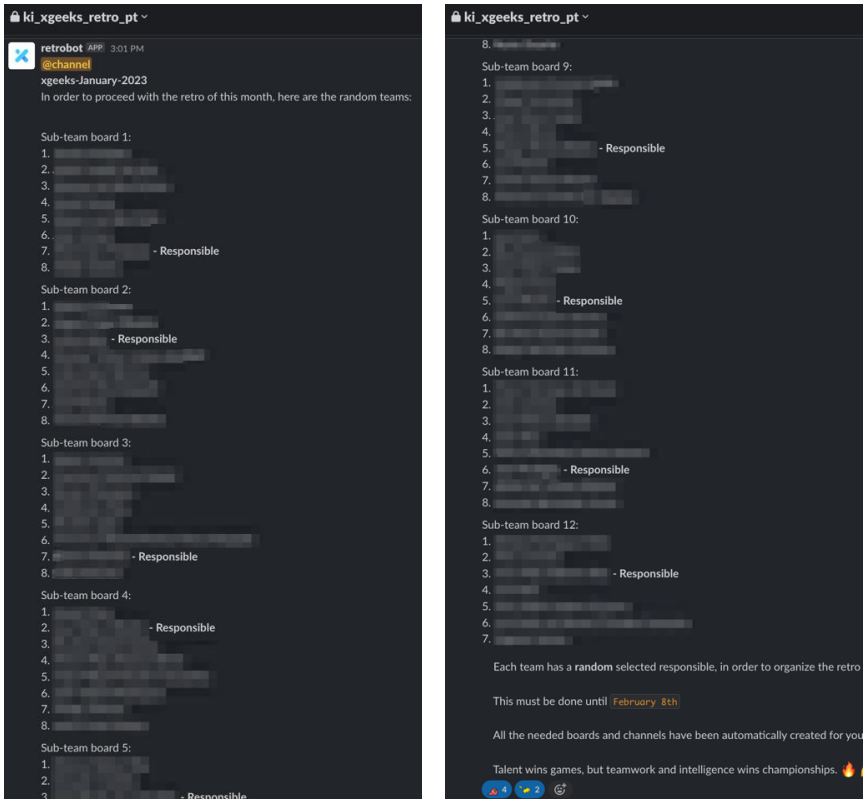


Figura 61 - Representação da mensagem enviada no slack com a constituição das sub-equipas

Finalizada a primeira retrospetiva com sucesso, foi possível constatar que foram criados 60 cartões na coluna "*Went well*", 7 cartões na coluna "*To improve*" e 6 na coluna "*Action points*", como é possível observar na Figura 62. Nesta é possível constatar o enorme sucesso da aplicação, uma vez que 31 cartões criados pelos colaboradores da empresa congratulam o lançamento e começo de utilização da plataforma.

-

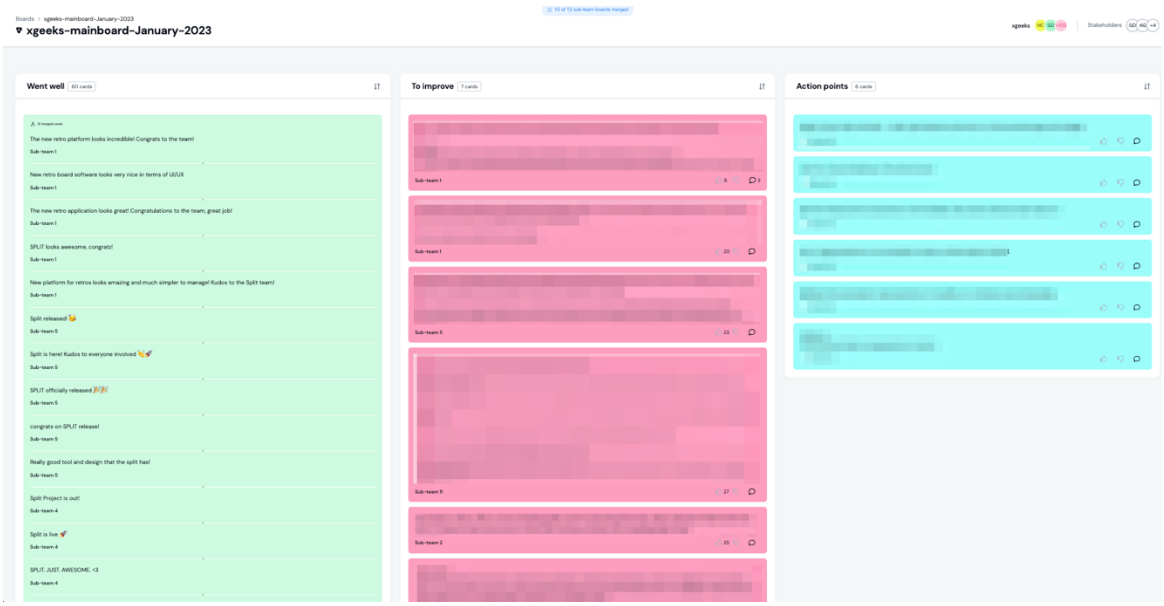


Figura 62 - Quadro principal da primeira retrospetiva

No dia 21 de fevereiro foi despoletado o segundo processo de retrospetivas da empresa usando o SPLIT, e esta é referente ao mês de fevereiro, na qual o processo anteriormente apresentado foi repetido, os resultados do processo são possíveis de observar no Apêndice U – Resultados 2ª retrospetiva. Por isso, novas subequipas foram criadas e os respetivos canais e quadros criados. Antes do processo ser criado, os novos colaboradores da empresa foram adicionados à equipa *xgeeks* e estes foram incluídos na retrospetiva seguinte. Neste processo foi criado um *cron job* para despoletar outro processo de criação de retrospetiva a ser despoletado no dia 24 de Março. Na imagem do quadro resultante da segunda retrospetiva não foram criados *action items* uma vez que não houve necessidade de o fazer tendo em conta as características dos comentários.

No dia 24 de Março foi despoletado o terceiro processo de retrospetivas.

O produto continua em desenvolvimento e a empresa aposta no projeto, também para alcançar visibilidade, e para servir de projeto introdutório para novos membros da empresa,

pouco familiarizados com as tecnologias. Visitando o repositório[38] é possível visualizar as *issues*, *commits* e *pull requests* recentemente criados. A Figura 63 demonstra a representação dos *commits* efetuados ao longo do tempo.

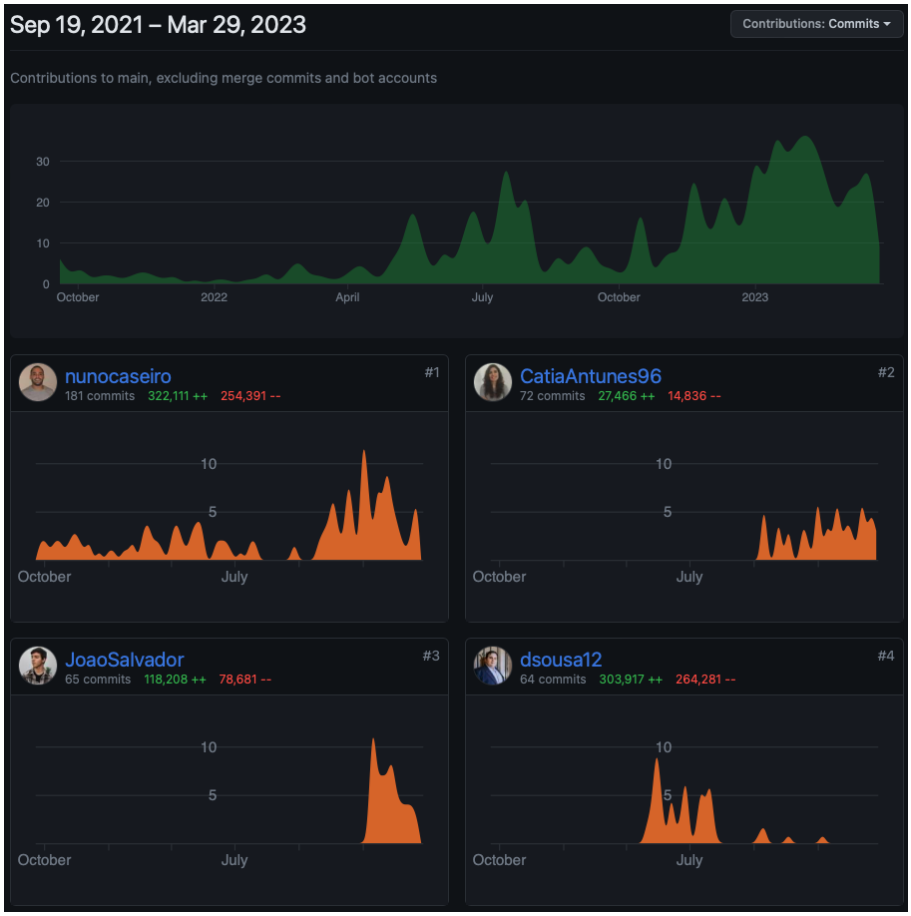


Figura 63 - Representação dos *commits* efetuados ao longo do tempo

Na Figura 63 é possível observar que foram efetuados *commits* desde setembro de 2021 até à data da extração da figura o que pretende representar a continuidade do projeto. Analisando o primeiro gráfico é possível verificar que a quantidade de *commits* aumentou ao longo do tempo uma vez que a equipa de desenvolvimento foi crescendo sendo que atualmente é constituída por 10 elementos, 6 *developers full time*, 2 *developers* que contribuem quando possível, 1 *developer* e autor deste documento que trata da gestão de projeto e continua a desenvolver quando existe disponibilidade tendo em conta a participação noutros projetos e finalmente o último elemento que se trata do EM responsável por garantir que toda a equipa se encontra funcional. Observando o gráfico referente ao autor deste documento, o primeiro do lado esquerdo, verifica-se que acompanhou o

[38] https://github.com/xgeekshq/split

desenvolvimento desde início até ao momento atual. Esta informação encontra-se acessível a qualquer pessoa que aceda à *tab insights* do repositório.

7.2.1. Testemunhos

Nesta subsecção pretende-se apresentar alguns testemunhos representativos do sucesso e utilização da plataforma. Durante o desenvolvimento do projeto, a equipa deparou-se com comentários positivos, na medida em que alguns elementos foram questionados por colaboradores da empresa ou ex-colaboradores sobre a data da disponibilização do produto, uma vez que pretendem utilizá-la como ferramenta para execução de retrospetivas.

Como é possível observar na Figura 62, são vários os comentários positivos face ao lançamento do produto, contudo esta não apresenta todos os comentários adicionados. Como tal, é importante realçar neste documento todas as opiniões dos colaboradores da empresa sobre a aplicação desenvolvida. Assim, no Apêndice T – Comentários sobre a plataforma, são apresentados todos os cartões com os comentários relativos à plataforma que foram criados na primeira retrospetiva e é possível constatar que 39 colaboradores votaram neste grupo de cartões, representando o cartão com maior votação, o que demonstra a intenção de reforçar a ideia de sucesso associada ao lançamento e utilização do produto. No Apêndice U – Resultados 2ª retrospetiva, verifica-se um cartão com 5 itens que mais uma vez congratulam a evolução da plataforma.

Com o intuito de apresentar uma opinião relevante, que analise o desempenho do autor deste documento, ao longo do desenvolvimento do projeto, e que reforce que este se encontra em utilização e contínuo desenvolvimento na empresa, foi pedido ao EM Gonçalo Dias, responsável pelo projeto, que indique seu *feedback* para ser incluído nesta secção. Este foi o seu *feedback*:

A xgeeks queria investir na criação de produtos internos que servissem de plataforma de treino e exploração de novas tecnologias por parte dos engenheiros, ao mesmo tempo criando solução com valor a nível interno da empresa e do grupo.

O projeto SPLIT engloba todas estas componentes assim como o facto de ser pensado para ser disponibilizado como produto open-source, uma vez que não existem soluções abertas com estas funcionalidades, e o valor acrescido é relevante tendo em conta que as ferramentas de retrospectivas são utilizadas de forma regular por todo o tipo de empresas.

O trabalho do Nuno neste projeto foi por isso preponderante para o seu sucesso, já que ele foi o maior responsável pelo seu desenvolvimento e posteriormente gestão do mesmo.

- *Começando pela decisão a nível de tecnologias a utilizar, o Nuno esteve envolvido na seleção de toda a stack tecnológica do projeto, seja Frontend, Backend, Base de Dados e arquitetura.*
- *Esteve também envolvido na fase de levantamento de requisitos e organização de workshops e entrevistas com utilizadores juntamente com a equipa de product design para prioritização de funcionalidades para o MVP e respetivo UI/UX.*
- *Foi gradualmente ficando responsável por gerir issues e o trabalho da equipa que foi sendo introduzida ao projeto, fazendo onboarding, mentoria e revisão de pull-requests.*

Como resultado final, foi lançada uma ferramenta interna que é usada neste momento diariamente por mais de 120 colaboradores, e que também será partilhada com o resto do grupo em Portugal e na Alemanha para gerirem as suas retrospectivas. Liderando agora também os esforços em colmatar tech debt e estabilizar a plataforma, prevê-se que no futuro próximo a aplicação passe também a ser usada e desenvolvida pela comunidade assim que a xgeeks começar a comunicação externa sobre o SPLIT.

Concluindo, o Nuno trabalhou muito acima das expectativas, num projeto de grande importância para a empresa e desenvolveu muito as suas capacidades a nível de engenharia, produto e gestão de projeto ao longo deste percurso.

8. Conclusão e trabalho futuro

As retrospetivas efetuadas em equipas são processos que podem consumir bastante tempo e, se não forem executadas corretamente pode-se correr o risco de não providenciar um meio para que todas as opiniões sejam tidas em conta. Este problema agrava-se quando as equipas apresentam um elevado número de membros, e como tal, é importante encontrar um mecanismo que permita que todos sintam que a sua opinião é bem recebida e tida em conta. Na xgeeks foi implementado um processo de divisão de equipas em subequipas com o intuito de reduzir o número de participantes em cada retrospetiva, permitindo dar relevância a cada opinião, no entanto, este mecanismo peca por automatizações e por dependência de ferramentas externas.

O produto SPLIT surge para colmatar as lacunas e dependências na execução das retrospetivas em equipas grandes. Previamente à implementação, foi executado um processo de *product design* e conceção do produto, com o intuito de compreender as necessidades dos utilizadores e com isso concretizar a ideia, levantar os requisitos e desenhar o produto e a sua interface. O SPLIT é uma plataforma que providencia uma interface na qual é possível criar e gerir equipas, mas também executar retrospetivas de forma assíncrona, através de quadros *kanban,* por norma constituídos por três colunas "*Went well*", "*To improve*" e "*Action points*". É possível criar cartões, movê-los, juntá-los ou separá-los. É possível votar nos cartões que se pretende dar relevância e também acrescentar comentários. A plataforma oferece suporte a dois tipos de retrospetivas, as regulares e as divididas, sendo que na primeira pode ser selecionada uma equipa, um conjunto de participantes ou nenhum utilizador adicional, promovendo assim um quadro pessoal. Nestas é simplesmente criado um quadro que serve de suporte para a execução de retrospetivas. A segunda replica o processo efetuado na xgeeks onde uma equipa é dividida em subequipas, é criado um quadro principal e outros para as subequipas. Cada subequipa deve efetuar a sua retrospetiva e quando terminar, os cartões gerados devem ser copiados, através da funcionalidade desenvolvida, para o quadro principal, que apresentará todos os cartões criados em cada uma das equipas divididas. Adicionalmente, a plataforma possibilita o agendamento de retrospetivas que faz despoletar um novo processo de retrospetivas numa determinada data e com uma certa recorrência. O SPLIT possibilita a integração com o *slack* da empresa, o que permite notificar os utilizadores de todas as ações relevantes, efetuadas durante o processo de uma retrospetiva.

Como o processo de retrospetivas divididas é longo e constituído por fases, é dada a possibilidade aos utilizadores responsáveis por coordenar uma retrospetiva, de mudar a fase atual. Assim sendo, após todos os quadros serem convergidos no principal e terminada a reunião entre os responsáveis, é possível mudar a fase da retrospetiva, passando da fase criação de cartões para a fase de votação. Ao transitar de fase, possibilita a votação nos cartões do quadro principal e adicionalmente, os utilizadores também são notificados, via slack, que a fase transitou e que devem votar nos cartões com os quais mais se identificam. Para finalizar o processo de retrospetiva, após a reunião final entre os responsáveis e o *stakeholders* e depois de gerarem os itens de ação para melhorar o que correu menos bem, é possível definir a fase "submetido" que implica o fecho do quadro e por isso não é possível efetuar alterações adicionais. Ao fazê-lo também é enviada uma mensagem no *slack* indicando que a retrospetiva do mês terminou e foram gerados determinados *action points.*

O sistema de software desenvolvido durante o projeto é composto essencialmente por três componentes. A aplicação web foi desenvolvida em *typescript*, através da *framework* Next.js e apresenta uma interface através da qual os utilizadores conseguem interagir com o sistema. O *backend* desenvolvido com auxílio da framework NestJS, encapsula a lógica de negócio e disponibiliza uma REST API e um *socket gateway* que possibilita que a aplicação cliente solicite o acesso ou alteração aos dados do sistema. Dados esses que são guardados numa base de dados não relacional *mongoDB* a que o *backend* acede e manipula. O sistema também utiliza uma base de dados em memória (Redis), para conseguir implementar e utilizar o sistema de filas para na execução de tarefas assíncronas que podem requerer algum processamento e que não devem limitar o processamento da aplicação.

O sistema é suportado por uma infraestrutura alojada na *Azure cloud services* onde foram implementados dois ambientes, um de desenvolvimento e outro de produção. Cada um é constituído por uma base de dados *mongo*, uma instância de *redis,* um *storage container* e dois app *services* que consomem e executam as imagens *docker* submetidas no *container registry,* também criado na infraestrutura. Através das *github actions* foi construído uma *pipeline* de CI/CD, sendo que, em todos o *pull requests* e *merges*, o código é validado e testado automaticamente através da execução dos testes unitários que foram criados. Após a validação bem-sucedida e se o *branch* for *merged* no principal, são construídas imagens *docker* para cada aplicação que são enviadas para o *container registry* e é pedido aos *app services* que as consumam, executem a última imagem enviada e assim disponibilizam a versão mais recente das aplicações.

As aplicações do sistema foram desenvolvidas de forma incremental, e ao longo do desenvolvimento foram executados testes manuais que permitiram identificar *bugs* e implementar correções e melhorias. Alguns testes unitários foram implementados no *backend* e *frontend* com o intuito de validar, com maior certeza, as funcionalidades desenvolvidas, obtendo cerca de 59% de cobertura do código no *backend* e 20% no *frontend*. Além destes, foram realizados testes de usabilidade no *frontend* com o intuito de compreender se a plataforma cumpre as necessidades dos utilizadores e se a interface proporciona uma boa experiência de utilização. Avaliando os resultados e *feedback* recebidos, é possível considerar que o produto apresenta uma interface intuitiva e capaz de servir as necessidades dos utilizadores. Nestes testes são propostas tarefas a serem executadas na aplicação e são disponibilizadas 47 questões que pretendem compreender se o utilizador conseguiu absorver tudo o que é apresentado e se conseguiu efetuar as tarefas propostas. Através destes testes foi possível concluir que a plataforma e a respetiva interface apresentam uma boa experiência de utilização, e, avaliando os resultados da execução do projeto, pode-se assumir que se colmataram os problemas da empresa previamente identificados ao automatizar o processo de execução de retrospetivas.

O projeto encontra-se em produção e continua em desenvolvimento, uma vez que não se encontram implementadas todas as funcionalidades descobertas na fase de conceção e *design* do produto. Na empresa decorreram três retrospetivas utilizando a plataforma, o que representa o sucesso da mesma na resolução dos problemas encontrados anteriormente. Adicionalmente, é possível constatar, observando a página do repositório do projeto no *github,* que esta conta com 40 estrelas, o que é representativo de alguma aproximação à comunidade.

Apresentado o resumo do projeto, importa referir que o desenvolvimento deste foi um enorme sucesso e que contribuiu para o crescimento a nível técnico, uma vez que permitiu contactar com imensas tecnologias até então desconhecidas e que providenciam enorme valor na participação no desenvolvimento de projetos do mundo real e com clientes. Importa referir que nenhuma das *frameworks* foi utilizada durante o percurso académico, o que acrescenta relevância na missão cumprida. É também possível definir o projeto como bem-sucedido, uma vez que foi implementaram-se todas as funcionalidades definidas para o MVP e outras definidas para a fase seguinte.

O projeto descrito neste documento serviu também como ponte para integrar projetos com clientes, uma vez que tentou-se replicar as metodologias e ambiente de trabalho utilizados em projetos de maior dimensão.

Após finalizar o trabalho de projeto, fiquei na empresa e continuei a dar suporte, mentoria e a gerir o projeto SPLIT, o que contribuiu para consolidar as capacidades de gestão de projetos. Adicionalmente, integrei um projeto de cliente com o foco no desenvolvimento de uma aplicação no sistema operativo iOS.

8.1. Trabalho futuro

Relativamente ao trabalho futuro, é possível definir que há um longo percurso pela frente, no sentido em que existem imensas funcionalidades, otimizações e testes que podem ser implementados no sentido de tornar a plataforma mais robusta. É com isto em mente que o projeto continua o seu desenvolvimento, com uma equipa exclusivamente alocada para o efeito, sejam eles, novos colaboradores da empresa ou antigos que pretendem participar.

No que diz respeito ao *design* do produto, pretende-se tornar a aplicação responsiva, ao ajustar o comportamento da interface, na medida em que esta se deve adaptar ao tamanho dos dispositivos, sendo que neste momento a interface torna-se difícil de utilizar em ecrãs de *smartphones*.

A nível de infraestrutura, esta foi implementada utilizando a interface disponibilizada no portal da *azure,* no entanto é importante definir a infraestrutura com código para documentar e facilitar a gestão da mesma, por isso pretende-se defini-la através de ficheiros *terraform*.

Relativamente aos padrões arquiteturais, se fizer sentido, no *backend* poder-se-á melhorar a aplicação do padrão *clean architecture* e *domain driven design*. Atualmente, um *refactor* encontra-se em desenvolvimento de modo a passar a lógica de negócio dos serviços para os *use-cases*. Em relação ao *frontend* é importante compreender se é necessário aplicar algum padrão, tendo o foco na escalabilidade do projeto.

Em relação aos testes, pretende-se obter em cada uma das aplicações a cobertura de código, no mínimo de 80%, e como tal, devem ser implementados testes unitários para esse efeito. A fim de testar a aplicação por inteira e testar a interface pretende-se efetuar testes E-2-E automatizados.

No que diz respeito à base de dados, apesar de boa performance, verificam-se algumas otimizações a fazer. Inicialmente, foi definido que uma base de dados não relacional seria o ideal para este projeto, uma vez que num só documento é possível armazenar e devolver toda a informação relativa a um quadro, no entanto, verificou-se que o número de relações entre as entidades tende em aumentar. Neste caso, seria uma mais-valia se fosse efetuada a migração para uma base de dados relacional.

No que diz respeito a funcionalidades, são inúmeras aquelas que se pretende adicionar e que foram definidas na secção 5.3. No entanto, é importante referir que algumas são consideradas prioritárias.

A funcionalidade de agendamentos é aquela que apresenta prioridade superior a qualquer outra, uma vez que foi definido o *design* para essa funcionalidade e é importante integrar na aplicação uma forma de agendar qualquer tipo de retrospetivas, definir a recorrência da mesma e também de que forma se quer ser notificado – sendo possível definir as configurações do *slack* para cada retrospetiva. Através da secção *upcoming* e do calendário definidos no *design* do produto será possível verificar as próximas retrospetivas e reagendá-las. Aliado a esta funcionalidade, pretende-se integrar o Microsoft teams e tirar partido da sua API, com o intuito de possibilitar o agendamento de reuniões de forma automática, com base na disponibilidade dos utilizadores.

Com prioridade também elevada define-se a funcionalidade que possibilite acompanhar os *actions points* definidos nas retrospetivas, isto é, acompanhar o estado das ações criadas em retrospetivas passadas que são associadas a alguém e que têm o intuito de resolver problemas que mencionados nas retrospetivas.

De seguida, seria interessante implementar a funcionalidade que permita extrair algum conhecimento das retrospetivas, isto é, proporcionar algumas estatísticas em cada retrospetiva e que permitam compreender, por exemplo, se os colaboradores se empenharam na execução da mesma, quais as equipas que criaram mais cartões, compreender se todos os *action points* foram executados com sucesso, etc.

Sem prioridades ainda definidas, mas com interesse superior, porque pode agilizar ainda mais o processo, é a funcionalidade que possibilita agrupar automaticamente os cartões que dizem respeito ao mesmo assunto, através de *natural language processing* (NLP).

Com pouca prioridade e com o intuito de cativar os utilizadores na execução de retrospetivas, poder-se-á implementar um sistema de gamificação que ofereça recompensas com base na performance de cada utilizador.

Bibliografia

[1] T. Dingsøyr, M. Mikalsen, A. Solem, and K. Vestues, 'Learning in the large - an exploratory study of retrospectives in large-scale agile development', *Lecture Notes in Business Information Processing*, vol. 314, pp. 191–198, 2018, doi: 10.1007/978-3-319-91602-6_13/TABLES/2.

[2] 'Agile Practice Guide | Project Management Institute'. https://www.pmi.org/pmbok-guide-standards/practice-guides/agile (accessed Dec. 25, 2022).

[3] 'xgeeks | Software development made in Portugal'. https://xgeeks.io/ (accessed Feb. 10, 2023).

[4] 'Online Retrospective Tool | Fun, Easy & 100% Free'. https://www.reetro.io/ (accessed Dec. 03, 2022).

[5] 'About us and how we started. | EasyRetro'. https://easyretro.io/pt/about/ (accessed Jan. 07, 2023).

[6] 'RetroTool | Your online retrospective made easy'. https://retrotool.io/ (accessed Dec. 03, 2022).

[7] 'Metro Retro - Early Access'. https://metroretro.io/early-access (accessed Dec. 03, 2022).

[8] 'Reetro.io'. https://www.reetro.app/ (accessed Dec. 03, 2022).

[9] 'What is SOC 2? A Beginners Guide to Compliance | Secureframe'. https://secureframe.com/hub/soc-2/what-is-soc-2 (accessed Jan. 08, 2023).

[10] P. Rodríguez, M. Mäntylä, M. Oivo, L. E. Lwakatare, P. Seppänen, and P. Kuvaja, 'Advances in Using Agile and Lean Processes for Software Development', *Advances in Computers*, vol. 113, pp. 135–224, Jan. 2019, doi: 10.1016/BS.ADCOM.2018.03.014.

[11] 'What is AGILE? - What is SCRUM? - Agile FAQ's | Cprime'. https://www.cprime.com/resources/what-is-agile-what-is-scrum/ (accessed Dec. 03, 2022).

[12] ‘Manifesto for Agile Software Development’. http://agilemanifesto.org/ (accessed Dec. 03, 2022).

[13] H. Alaidaros, M. Omar, and R. Romli, ‘The state of the art of agile kanban method: challenges and opportunities’, *Independent Journal of Management & Production*, vol. 12, no. 8, pp. 2535–2550, Dec. 2021, doi: 10.14807/IJMP.V12I8.1482.

[14] ‘Theory of Constraints (TOC) | Lean Production’. https://www.leanproduction.com/theory-of-constraints/ (accessed Mar. 22, 2023).

[15] ‘Design Thinking 101’. https://www.nngroup.com/articles/design-thinking/ (accessed Nov. 30, 2022).

[16] ‘Interview Experience Maps – Figma’. https://www.figma.com/file/IfUvUO8tJjOEE5R1ayoOrv/Interview-Experience-Maps?node-id=0%3A1&t=afweJGI81ZNHA2S6-0 (accessed Dec. 12, 2022).

[17] ‘Opportunity Mapping (Copy) – Figma’. https://www.figma.com/file/5maIXZSOMKbY6aG3j1a1no/Opportunity-Mapping-(Copy)?node-id=0%3A1&t=uZt4qK0KdnXR2z1i-0 (accessed Dec. 14, 2022).

[18] ‘Ideation workshop PRESENTATION (Copy) – Figma’. https://www.figma.com/file/2xoPv6DGnfV5pgSHhtetSe/Ideation-workshop-PRESENTATION-(Copy)?node-id=2%3A20&t=bz6EFTaidOkJLwgN-0 (accessed Dec. 14, 2022).

[19] ‘Ideation workshop WORKSHOP (Copy) – Figma’. https://www.figma.com/file/JZbyXvNRicjtdJK3rQnkRK/Ideation-workshop-WORKSHOP-(Copy)?node-id=0%3A1&t=Q7dgrSWRAF0xnfdP-0 (accessed Dec. 14, 2022).

[20] ‘Ideation Prioritization + Story Mapping – Figma’. https://www.figma.com/file/40pKKaLsqTFMkn8q9yyCzw/Ideation-Prioritization-%2B-Story-Mapping (accessed Mar. 30, 2023).

[21] ‘Sitemap – Figma’. https://www.figma.com/file/ehA5UOy8MHlakj46tiq9ak/Sitemap (accessed Mar. 30, 2023).

[22] ‘Design System Split – Figma’, 2021. https://www.figma.com/file/zlNIKl8GTE2jeU6jwzNhxH/Design-System-DC?node-id=2%3A15&t=yyUtEuZHkNXuLIZc-0 (accessed Dec. 24, 2022).

[23] ‘UI v2.0 – Figma’. https://www.figma.com/file/O7TOfs2OGc9ZehrMItgR6u/UI-v2.0?node-id=3%3A22 (accessed Feb. 28, 2023).

[24] ‘The C4 model for visualising software architecture’, 2021. https://c4model.com/ (accessed Dec. 29, 2021).

[25] ‘What is Azure Active Directory? - Microsoft Entra | Microsoft Learn’, 2021. https://learn.microsoft.com/en-GB/azure/active-directory/fundamentals/active-directory-whatis (accessed Dec. 30, 2022).

[26] ‘Email Delivery, API, Marketing Service | SendGrid’. https://sendgrid.com/ (accessed Dec. 30, 2022).

[27] ‘Where work happens | Slack’, 2021. https://slack.com/ (accessed Dec. 30, 2022).

[28] ‘Figma’. https://www.figma.com (accessed Jan. 02, 2023).

[29] ‘Documentation | NestJS - A progressive Node.js framework’, 2021. https://docs.nestjs.com/ (accessed Jan. 03, 2023).

[30] ‘Overview of React.js’. https://www.patterns.dev/posts/reactjs/ (accessed Jan. 14, 2023).

[31] ‘Next.js by Vercel - The React Framework’. https://nextjs.org/ (accessed Jan. 11, 2023).

[32] ‘TypeScript: JavaScript With Syntax For Types.’ https://www.typescriptlang.org/ (accessed Jan. 19, 2023).

[33] ‘MongoDB: The Developer Data Platform | MongoDB’. https://www.mongodb.com/ (accessed Jan. 19, 2023).

[34] ‘Redis’. https://redis.io/ (accessed Jan. 19, 2023).

[35] ‘API Documentation & Design Tools for Teams | Swagger’. https://swagger.io/ (accessed Jan. 19, 2023).

[36] ‘Features • GitHub Actions’. https://github.com/features/actions (accessed Jan. 19, 2023).

[37] ‘Twilio SendGrid Email’. https://sendgrid.com/go/email-brand-signup-sales-1?utm_source=google&utm_medium=cpc&utm_term=sendgrid&utm_campaign=SendGrid_G_S_Brand_ROE_Emerging&cq_plac=&cq_net=g&cq_pos=&cq_med=&cq_plt=gp&gclid=Cj0KCQiA8aOeBhCWARIsANRFrQEaLmd_1S_veQxUabljW8cyJ1yA9qjfme3Gvh_ZvdvU-vd679JveeoaAp8OEALw_wcB (accessed Jan. 19, 2023).

[38] ‘Cloud Computing Services | Microsoft Azure’. https://azure.microsoft.com/en-gb/ (accessed Jan. 19, 2023).

[39] ‘Stitches — CSS-in-JS with near-zero runtime’. https://stitches.dev/ (accessed Feb. 02, 2023).

[40] ‘Primitives – Radix UI’. https://www.radix-ui.com/ (accessed Feb. 02, 2023).

[41] ‘NextAuth.js’. https://next-auth.js.org/ (accessed Feb. 02, 2023).

[42] ‘Jest · Delightful JavaScript Testing’. https://jestjs.io/ (accessed Feb. 02, 2023).

[43] ‘Home | React Hook Form - Simple React forms validation’. https://react-hook-form.com/ (accessed Feb. 05, 2023).

[44] ‘atlassian/react-beautiful-dnd: Beautiful and accessible drag and drop for lists with React’. https://github.com/atlassian/react-beautiful-dnd (accessed Feb. 05, 2023).

[45] ‘Overview | TanStack Query Docs’. https://tanstack.com/query/v4/docs/react/overview (accessed Feb. 05, 2023).

[46] ‘Storybook: Frontend workshop for UI development’. https://storybook.js.org/ (accessed Feb. 05, 2023).

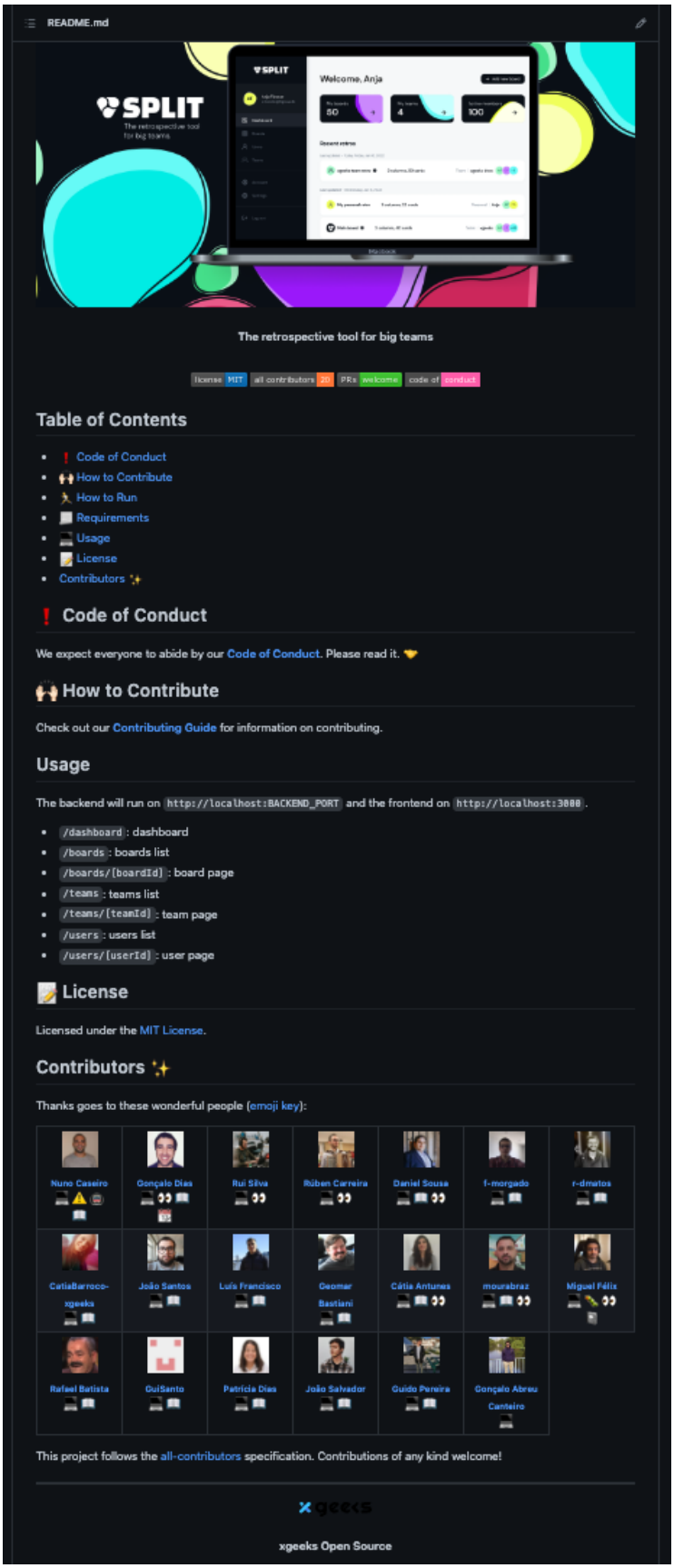
[47] ‘SQL vs. NoSQL Databases: What’s the Difference? | IBM’. https://www.ibm.com/cloud/blog/sql-vs-nosql (accessed Jan. 30, 2023).

[48] R. C. Martin, *Clean Architecture: A Craftsman’s Guide to Software Structure and Design*. Prentice Hall Press, 2017.

[49] ‘The SOLID Principles of Object-Oriented Programming Explained in Plain English’. https://www.freecodecamp.org/news/solid-principles-explained-in-plain-english/ (accessed Mar. 12, 2023).

[50] ‘Continuous delivery and continuous integration - AWS CodePipeline’. https://docs.aws.amazon.com/codepipeline/latest/userguide/concepts-continuous-delivery-integration.html (accessed Feb. 05, 2023).

Apêndice A - ReadMe



README.md

The retrospective tool for big teams

license MIT all contributors 20 PRs welcome code of conduct

Table of Contents

- ❗ Code of Conduct
- 🙌 How to Contribute
- 🏃 How to Run
- ⬜ Requirements
- 💻 Usage
- 📝 License
- Contributors ✨

❗ Code of Conduct

We expect everyone to abide by our Code of Conduct. Please read it. 🤝

🙌 How to Contribute

Check out our Contributing Guide for information on contributing.

Usage

The backend will run on `http://localhost:BACKEND_PORT` and the frontend on `http://localhost:3000`.

- `/dashboard` : dashboard
- `/boards` : boards list
- `/boards/[boardId]` : board page
- `/teams` : teams list
- `/teams/[teamId]` : team page
- `/users` : users list
- `/users/[userId]` : user page

📝 License

Licensed under the MIT License.

Contributors ✨

Thanks goes to these wonderful people (emoji key):

This project follows the all-contributors specification. Contributions of any kind welcome!

xgeeks

xgeeks Open Source

Apêndice B – Guião da entrevista

Processo de retrospetiva e ferramentas

Q: Alguma vez efetuou alguma retrospetivas remotamente?

- Se **nunca efetuou retrospetivas remotamente**:
  - Porquê?
  - Se precisasse de efetuar uma remotamente, o que seria mais importante para si?
  - Poderia descrever o processo do início até ao fim da última vez que fez uma retrospetiva?
    - Se não mencionarem o que fazem após a retrospetiva:
      - Qual é o resultado obtido de todo o processo? e, como coloca em prática os resultados para obter benefício?
- Se **sim**:
  - Poderia descrever o processo do início até ao fim da ultima vez que fez uma retrospetiva?
    - O que considera mais importante quando participa numa retrospetiva remotamente?
      - Se não mencionarem o que fazem após a retrospetiva:
        - Qual é o resultado obtidos de todo o processo? e, Como coloca em prática os resultados para obter benefício?
- Se **sim ou não**:
  - Com que frequência efetua retrospetivas?
  - Utiliza alguma ferramenta para efetuar retrospetivas?
    - Se **sim**:
      - Que ferramentas utiliza?
      - Quando utiliza essas ferramentas o que mais gosta ou menos gosta?
      - Quais são os fatores críticos para a adoção de uma ferramenta para efetuar retrospetivas?

- Quais são as funcionalidades mais importantes que uma ferramenta de retrospetivas deve providenciar?

- Se não:
  - Se existir, qual é a razão pela qual não utiliza estas ferramentas?
  - Que ferramentas uma ferramenta de retrospetivas deveria ter para começar a utilizar?

Equipas

Q: Qual é o tamanho das equipas nas retrospetivas que em que participa?

- Se pudesse mudar alguma coisa, o que mudaria na ferramenta que utiliza, considerando o tamanho da equipa?
- O que pensa que pode tornar as retrospetivas mais simples quando se lida com equipas com um elevado número de membros?
- Qual poderá ser o limite de tamanho que uma equipa deverá ter de forma a realizar retrospetivas com sucesso?
  - Se **for mencionado um limite**:
    - Porque acha que deverá existir limite de membros?
    - Existe alguma coisa que pode ser feita para equipas numerosas participarem numa retrospetiva sem estabelecer limite no número de membros?
  - Se considerar que **não há limites**:
- Alguma vez experienciou dificuldades em agendar reuniões remotas para executar uma retrospetiva?
  - Se **sim**:
    - Pode explicar o processo e explicar as dificuldades inerentes?
    - Acha que poderá existir alguma forma de facilitar o agendamento?
  - Se **não**:
  - Pode explicar como funciona o processo de agendamento antes da reunião para executar a retrospetiva com a sua equipa?
  - Acha que poderá existir alguma forma de facilitar o agendamento?
    -

- Mudaria alguma coisa na forma como executa as retrospetivas (se possível pensando em equipas grandes)
    - Se **sim**:
        - O que faria de diferente? (Em relação à sua situação atual ou refernte a equipas grandes)

Limites de tempo

- Acha que as retrospetivas deverão ter uma duração limitada?
    - Como acha que a duração pode ser respeitada ou em caso negativo, porque acha que não deverá existir limite?

Integrações

Q: Considera que outra ferramenta que conhece ou utiliza diariamente poderia ser integrado de forma a facilitar a execução de retrospetivas?

Problemas durante as retrospetivas

Q: Pode falar um pouco sobre a última vez que teve problemas durante uma retrospetiva? Pode se referente a qualquer assunto.

Inteligência artificial

Q: Se existisse algum tipo de IA que ajudasse a realizar retrospetivas, o que gostaria que fosse ou de que forma poderia ajudá-lo?

Questões gerais

- Considera que as retrospetivas podem ser úteis, contribuir ou ajudar os seus colegas no dia à dia de trabalho?
- Alguma vez sentiu receio ou reticência em partilhar alguma coisa durante uma retrospetiva?
    - Se **sim:**
        - O que poderia ajudar a fazê-lo partilhar os seus pensamentos?
    - Se **não**:
        - Porque acha que se sentiu sempre confortável em partilhar os seus pensamentos?

- Considera que o que partilha é importante?
    - Se **sim ou não:**
        - O que o faz pensar dessa forma?

Apêndice C - Relatórios de entrevistas

Utilizador 1

Jan 18, Tuesday 2022 - [redacted] - Interview Snapshot

Interviewee Data

Company: <KI Group>

Name: <Birgül Kurt>

Role: <Product Owner in a Scrum Team>

Profile: <Started as Scrum Master, Project Lead, Product Owner>

Topics

- Discussion on doing retro perspectives
- End-to-end process, steps and interactions of doing retros.
- Key challenges in doing retro perspectives in large teams
- 3rd party integrations

Key Data

Date:

Duration: <30 min>

Interviewer: < Anja Foerster >

Notes Taker: < Nuno Miguel Caseiro >

Meeting recording: <Recording Link>

Meeting Transcription: <Transcription Link>

Opportunities

Codes for the learnings to find patterns across interviews

I need/want/struggle/think...

- I need to give everyone on a team the opportunity and the right environment to discuss about their issues and to express themselves. Especially in a remote environment.
- I need to feel the mood and the expressions of the people participating in retros.
- I want visuals to support each retro phase
- I want the rating to be fun an creative
- I need a tool which tracks the action items and reminds me of action points
- I think teams for retros should not be bigger than 7-8 people otherwise people loose the focus or don't feel that secure to talk.
- I want a feature which is automatically documenting what people say when considering doing retros with really big teams.

Quotes

Doing retros in large teams face to face can be hard. A strong moderator and a tool to support that structure and also something or someone to document the meeting is needed. An async process can work well for large teams remotely or face to face

There are scaled frameworks like scrum of scrums or SAFe with which you can do company wide retro spectives.

Experience Map

End-to-end process described, key steps, and interactions of doing retro perspectives

References

Scrum of Scrums : A technique to scale Scrum up to large groups (over a dozen people), consisting of dividing the groups into Agile teams of 5-10

SAFe: SAFe is a scaled framework that aims to scale up the agile handling and thinking of teams across the entire group. Similar to Scrum of Scrums.

Interview Notes

Process

- As a scrum master you have to prepare the retro properly.
- Deciding on the tool to use
- Preparing visuals for each phase of the retrospective.
- Inviting everyone. Sending all important links, like link to the tool,....
- Informing everyone about the agenda and about what we want to achieve.
- In the actual retro you do an intro where you also can check on the mood of your team
- Then based on the topic (transparency, performance,...) you are working on there are different tools to use (retromat is a cool one)
- Most common are what went well, What didn't go that well, What we would like to do
- Depending on what topic you are working on, there are sometimes 3 parts, sometimes 4,...
- After the actual retro is to decide who the action points are for.
- Then either putting them on an impediment backlog, or putting them in a board,...
- And as a team we need to have a look at them from time to time to see how its going with the action points
- Have a slot in retro to take a look at the action points from time to time.

- Most important thing in retros:
  - Giving everyone on a team the opportunity to discuss about their issues and how they are feeling about the current situation even if it is positive or negative. Especially in a remote environment.

- Biggest challenges in doing retrospectives remotely:
  - The right environment where all people can have the chance and the opportunity to express themselves like a face-to-face meeting.
  - Its harder for the moderator to understand people's moods and expressions and adapt the meeting due to the lack of that insights.
- How prepare a good retrospective?
  - Having the right collaboration tools with visual support
  - Preparing the visuals to be more interesting and whole team can work together and give their input
- Scheduling the retros via email and sending all the important links to the tool that will be used.
- Visuals must be something that supports each retro phase and there are some tools or objects like flipchars to draw some pictures or something that makes the retro more fun for everyone
- Rating different stuff can be fun and creative.
- After the retro and to track the action items:
  - A kanban board or the confluence page can be useful to visualize the current status of the action items
- How improve the process of tracking the action items?
  - Tool to track the development and self assign to the topics the team wants to work on.
  - Something that reminds the action points.
  - Having some slot in retros to analyse the status of the action items.

Teams

- Should be small like 7-8 people. If its too big you might loose the focus. And people don't feel like that secure to talk in big rounds compared to small rounds.
- When teams are larger, it is needed a strong moderator and a tool to support that structure and also something or someone to document the meeting.
- In larger teams it's possible to split into smaller ones (breakout rooms) and after that bring everything together
- Suggestion: scrum of scrums to scale the retrospectives (Is a scaled agile technique that offers a way to connect multiple teams who need to work together to deliver complex solutions)
- If the retro is well organized even in large teams can be a success. Transparency is the priority.
- The async process can work well if the participants write down their stickies and after that discuss them in a meeting. However this can be more harder if the retrospectives are face-to-face.

Tools

- Ai tool to document what people say
- Mentioned tools: Jira (kanban board) , Miro (visuals)

Utilizador 2

Jan 18, Tuesday 2022 - [redacted] - Interview Snapshot

Interviewee Data

Company: <name>

Name: <Tiago Sousa>

Role: <role>

Profile: <short desc of responsibilities>

Topics

- Discussion on doing retro perspectives
- End-to-end process, steps and interactions of doing retros.
- Key challenges in doing retro perspectives in large teams
- 3rd party integrations

Key Data

Date:

Duration: <30 min>

Interviewer: < Anja Foerster >

Notes Taker: < Nuno Miguel Caseiro >

Meeting recording: <Recording Link>

Meeting Transcription: <link/embedded/not available>

Opportunities

Codes for the learnings to find patterns across interviews

I need/want/struggle/think...

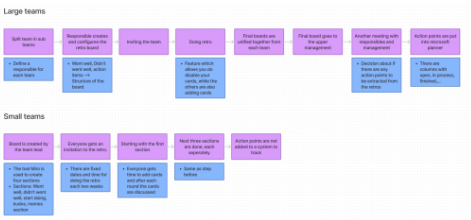
- I want a meme section to make retros more fun
- I struggle to find a time slot for doing the retro, where everyone is free.
- I need a tool when doing large team retros, which requires less manual work to order and merge the cards
- I want the cards to be automatically being categorized.
- I need a feature to track/subscribe the status of the action items after the retro.
- I need a tool wich integrates automatically the retro into peoples calendar. Maybe in combination with HR tools like personio
- I need anonymization, a voting system/add reactions, ordering the cards functionality
- I want gamification to make retros more fun

Quotes

> *I think the teams should be small. Around 6-10 people. Following the pizza concept.*

Experience Map

End-to-end process described, key steps, and interactions of doing retro perspectives



References

Miro: The online collaborative whiteboard platform to bring teams together .

Microsoft planner: Organize teamwork easily - with an intuitive, collaborative and visual task management

Interview Notes

Process

- Large teams:
  - Every month
  - Whole team is splitted into smaller ones and one member of each team is the responsible for schedule the meetings, create and configuring the retro boards.
  - Finishing the retrospectives the final board is unified from all the teams then goes to upper management.
  - The management staff has another meeting with the responsibles of each small team to decide if there are any action item that can be extracted from
  - A tool like microsoft planner is used to track the action items with a board with sections (on going... on process...)
- Small teams:
  - Every 2 weeks.
  - Sprint retrospectives where all people are together at the same meeting
  - Miro is used to create four sections (went well, didn't went well, start doing and kudos).
  - Meme section is very important

    - After the cards being added to the sections, they are discussed

- Teams should be small like 6-10 people, following the pizza concept
- The people in retros must have the responsibility to fill out the cards to help the company evolve but the number of cards added dropped when people work from home because there is nothing to say.
- Difficulties on scheduling calls:
    - Its hard to know the agenda of other people in a team and its hard to find the time slot that everyone is able to attend.
    - Suggestion: integrate person's calendar of each team into DC to check everyone's availability and suggest the time slot that fits better for everyone.

Tools

- Easy retro/fun retro is used in the company retrospectives to create a three-column board and also allows to toggle the visibility of the cards. They are not visible for everyone at the "adding" phase.
- Comparing the fun retro with miro, the first is more easy to use in larger teams because it requires less manual work to order and merge the cards
- How improve the tracking of items:
    - Subscribe feature to notify the users when the status of action items has changed
- In order to help with scheduling and planning the D&C could have some kind of HR tool like personio to track people and be able to automatically integrate the retro meeting into people's calendar.
- Critical things that a retro tool should have:
    - Anonymization (public cards or not)
    - Voting system for the cards or possibility to add some reaction
    - Ordering the cards by votes or reactions.
- Retros can be easier if they have some feature to notify people to fill the board with cards.
- Suggestion: add some gamification to the tool so the people want to vote and can get some points and after the month there are some ranking and prizes for the best.
    - Ex: earn points of adding cards and at the end of the month win something. Profile with levels and achievements.
- IA to merge cards but before of that the cards could be categorized.

Utilizador 3

Jan 19, Wednesday 2022 - [redacted] - Interview Snapshot

Interviewee Data

Company: <name>

Name: <Micael Rodrigues>

Role: <role>

Profile: <short desc of responsibilities>

Topics

- Discussion on doing retro perspectives
- End-to-end process, steps and interactions of doing retros.
- Key challenges in doing retro perspectives in large teams
- 3rd party integrations

Key Data

Date:

Duration: <30 min>

Interviewer: < Anja Foerster >

Notes Taker: < Nuno Miguel Caseiro >

Meeting recording: <Recording Link>

Meeting Transcription: <link/embedded/not available>

Opportunities

Codes for the learnings to find patterns across interviews

I need/want/struggle/think...

- I want to have the ability to choose from a template for the retro board
- I think that hiding/showing cards is good, because they don't influence other people when they are adding cards
- I struggle to understand the meanings of the cards sometimes
- I think that people hold back things, because they are scared to share it
- I think that some action points are just getting lost after doing the retro
- I want an AI to help with the organization and scheduling process before the actual retro (create the teams, schedule the meetings, create the board and share the link).

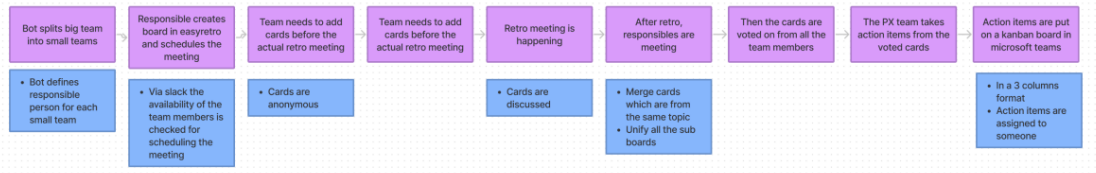
Quotes

Remarkable Quotes

Experience Map

End-to-end process described, key steps, and interactions of doing retro perspectives

(illustrative)



Interview Notes

Process

- In the company, a bot splits the whole team into a smaller ones every month.
- The responsible of each team creates a board in easyretro and find a time slot to schedule the meeting.
- Those who are not responsible only need to add cards before the meeting and this phase is anonymous.
- In preparation and via slack, the team members' availability is checked in order to schedule the meeting
- At the end of the meeting the cards are discussed
- After the retro, in xgeeks, all the team leaders take the feedback from each team and have other meeting to merge the cards that are about the same topic and unify all the sub boards.
- Then the cards are voted by all the members of the company and the px team took action items from that.
- The action items are on a kanban board of microsoft teams with 3 columns format and they are assigned to someone.
- The action points should be tracked and people should be notified about the status of them.
- Usually there are no difficulties in the scheduling process.

- In the project retros (sprint review) they are every two weeks.
- The action points are for the whole team.
- In the project retro the easy retro is used and the actions items stays there

- Difficulties:
  - Sometimes not everyone agrees on something.
  - Cards can have different meanings for different people. Sometimes it would be helpful if the owner of the cards could explain them.
  - Some teammates can withhold something because they may have some kind of fear.
- 1-1 between a team member and the responsible can be helpful to address some issues.
- Some times the action items get lost.

Teams

- Should be small (7-10 people).
- The retros can be scaled to more people depending on the size of the teams.
- If they are too large its hard to have time to everyone speak
- The time limit of each retro depends on the number of cards or topics to be discussed. Sometimes 20-30 min but it can go up to an hour. More than one hour might be too long.

Tools

- Teams and easy retro for scheduling.
- Having a ability to choose a template for the board is good.
- Hiding/showing cards is good in that type of tool because if they are hidden they don't influence other people when they are adding the cards.
- This type of tools must have same behavior/layout than others to easily promote the transition.
- IA/automation - could create the teams, schedule the meetings, create the board and share the link with all team members.

Utilizador 4

Jan 19, Wednesday 2022 - [redacted] - Interview Snapshot

Interviewee Data

Company: <name>

Name: <Adam Jacobson>

Role: <role>

Profile: <short desc of responsibilities>

Topics

- Discussion on doing retro perspectives
- End-to-end process, steps and interactions of doing retros.
- Key challenges in doing retro perspectives in large teams
- 3rd party integrations

Key Data

Date:

Duration: <30 min>

Interviewer: < Anja Foerster >

Notes Taker: < Nuno Miguel Caseiro >

Meeting recording: <Recording Link>

Meeting Transcription: <link/embedded/not available>

Opportunities

Codes for the learnings to find patterns across interviews

I need/want/struggle/think...

- I think limiting the votes is essentially for the people to choose what is really important
- I think the weakest part of the process is taking action items and then doing something with them on a regular basis
- I need the retro meetings to be time bounded, because meetings are very expensive
- I think if there are too many people they just don't talk
- I think retros is about the team talking to each other and forming a kind of bond

Quotes

During retros an issue might be the disagreement among engineers about some way to do something.

Teams are always a two pizza team because its become almost impossible to manage effective if they are too big.

Retros are also really about team cohesion, making people feel they're part of a group".

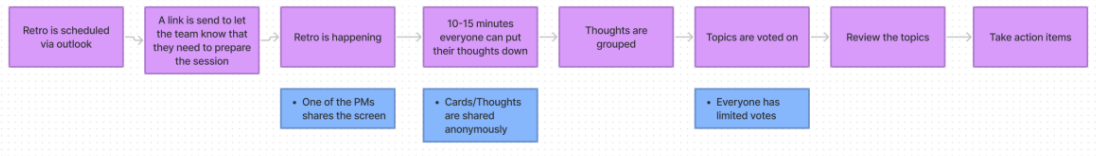
People should have an opportunity to speak up but they should not be forced to.

It's super common that people hold back their thoughts.

Experience Map

End-to-end process described, key steps, and interactions of doing retro perspectives

(illustrative)



Interview Notes

Process

- Before the retros a link is send to the team to give them a warning to prepare the session.
- The scheduling was via outlook, just a calendar invitation.
- Ideally the people shares their feedback at the start of the meeting but sometimes it doesn't happen. One of the pms shares the screen.
- An free online tool is used with 3 columns format (went well, things maybe didn't go well and action items).
- 10 or 15 minutes in the beginning, put thoughts down. then group and vote - 5 minutes. We would then have time blocked for reviewing everything. And then we'd have a kind of a block for action items.
- The cards are anonymous and the finite voting system provided by the tool is used as a proxy to rank the most critical topics to discuss. (5 votes and adjustable)
- Limiting the votes is essentially for the people to choose what is really important.
- There were no problems in scheduling the retros because they were at product team level so the team was used to meet on a regular basis.
- The retros were lead by engineers.
- The action points were taken but that was probably the weakest part of the process...taking action items and then doing something with them on a regular basis.
- Sometimes its hard to take time to work on internal things like being better as a team than it is to work on what the client wants.
- "I think that the biggest problem with retrospectives is, is the psychology"

- "I didn't see the problem being the tool. So the problem is for how the team works together"
- The action items were held in Jira - to confirm -
- Time bounding meetings (1h max) is a good idea because meetings are very expensive and people should not talk about off-topic issues.
- If there are many stuff to talk about and it's not enough time in a retro session it's possible to schedule more often or speak about the remained topics in the next session
- The top voted things should be the priority. The less voted things are discussed after.
- The action items from the last retro that were not discussed or dealt with properly are taken to the next retro.
- During retros an issue might be the disagreement among engineers about some way to do something.

Teams

- At most 10
- "I don't know if I would run retros like this sort of agile retrospective from a very large team"
- "From the development perspective the teams are always a two pizza team because its become almost impossible to manage effective if they are too big"
- Retros of retros if it's company-wide.
- The retrospective is not about gathering information but it is about the team talking to each other and forming a kind of bond.
- "The retrospective tool it's useful for identifying things, but it's also really about team cohesion, making people feel they're part of a group"
- If the number of people in a retrospective is too big "I would probably split up into 10 groups of 10 people. Everybody does a retro, and then we have a representative from each group that comes together and does a retro together" and aggregates everything up.
- If there are too many people they just don't talk.
- People should have a opportunity to speak up but they should not be forced to. They must feel they can talk and there are no negative consequences
- Its super common that people hold back their thoughts
- To encourage people who are quiet, introverted, or just feel more scared, it may be possible to have one-on-ones which is a kind of safe place.
- It's essential to have a good relationship with people and people should have the feeling that what they share is important. it's not the time to blame each other.

Tools

- Jira
- Gamification maybe motivates people (ex: language learning apps)

Utilizador 5

Jan 19, Wednesday 2022 - [redacted] - Interview Snapshot

Interviewee Data

Company: <name>

Name: <Pedro Cardoso>

Role: <role>

Profile: <short desc of responsibilities>

Topics

- Discussion on doing retro perspectives
- End-to-end process, steps and interactions of doing retros.
- Key challenges in doing retro perspectives in large teams
- 3rd party integrations

Key Data

Date:

Duration: <30 min>

Interviewer: < Anja Foerster >

Notes Taker: < Nuno Miguel Caseiro >

Meeting recording: <Recording Link>

Meeting Transcription: <Transcription Link>

Opportunities

Codes for the learnings to find patterns across interviews

I need/want/struggle/think...

- I think it's important that a retrospective is approached in a positive way and with positive moods.
- I want metaphors for the columns (happy column, happy moment, for us to learn - never in a negative way)
- I struggle with finding a way to track things (action points), because we have too many tools for different steps of the retro. It should only be one tool for everything.
- I sometimes struggle to find enough time to discuss everything that is important within one retro.
- I think the voting system is very relevant to show what are the real problems.
- I need the drag and drop experience and the feature to rank the issues/problems
- I want a feature that gives visual importance to the most voted /critical topics. (e.g. increase boarder size)
- I want visuals to support comments/cards
- I want a drawing feature
- I need the action item linked to the respective issues
- I want a feature to identify keywords

Quotes

"Retro should be a moment to learn", "we focus too much on the negative, but we need to remove completely the negative mood or a retro."

A feature that allows you to tap on the card and show it in fullscreen mode could be a good idea to force people to focus on the topic that is being discussed and to keep them from starting looking around

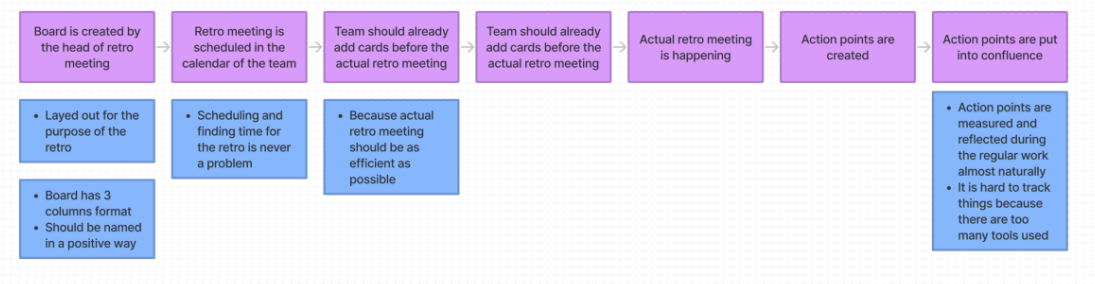
Teams with large number of people could be great because the votes get more relevant but on the other hand, for the discussion it's not good if everyone wants to talk.

Splitting the big team in smaller ones like we do in workshops.

Experience Map

End-to-end process described, key steps, and interactions of doing retro perspectives

(illustrative)



Interview Notes

Process

- A board is created and shared by the head of the retrospective meeting and it's laid out the purpose of that retrospective.
- Before the meeting, people should add some card because the meeting should be efficient as possible
- The meeting is scheduled in calendar and it is not a problem because people understands that it is necessary
- During the retrospective meeting, the points are already written in the board so we can just discuss about them.

- The last time he did a retro was during the discovery phase of a project to find out how the team was working and how they were communicating.
- The last two years all the retros were remote.
- The board has a 3 columns format (went well, what needs to be improved, action points)
- It's important to have the positive column, because if the people only think about the negative points it seems like everything went wrong and then it's always a negative feeling doing the retrospectives.
- "For me, it's quite important that a retrospective as very positive approach and moods and not be seen as something negative"
- The tracking process of the action points depends. The last time he done was very organic. People remembers and they talked about it in the following retros or in the daily routine.
- When working on a project those action points were placed in confluence to be more visible to all stakeholders.
- It's strange doing retro outside the confluence and then going into confluence to put action points on there.
- Most of the actions points are measured and reflected during the regular work almost naturally.
- Its hard to have a specific way to track things because we have so many tools. They tried to centralize in a single tool but confluence doesn't make retros the way we would like.
- The place where usually retros are done should be the same tool where we can also track things, have decision logs and document things.
- "Retro should be a moment to learn", "we focus too much on the negative, but we need to remove completely the negative mood or a retro."
- The time sometimes is short to discuss everything that it is important .

Tools

- Slack to communicate (share link and the topic about the retro is purposed)
- Fun retro was used to make retros and it's interesting because it allows to drag and drop topics inside the columns.
- The voting system is very relevant to show what are the real problems and important topics to discuss.
- The drag and drop experience and the feature to rank the issues/problems is important
- Confluence for action points
- One of the critical things to start adopting the tool could be the feature that gives visual importance to the most voted/critical topics. (e.g. increase boarder size)
- A feature that allows you to tap on the card and show it in fullscreen mode could be a good idea to force people to focus on the topic that is being discussed and to keep them from starting looking around.
- Metaphors for the columns (happy column, happy moment, this column for us to learn - but not in a negative)
- Visuals to support comments/cards: instead of sharing a link of an image, it should be possible to upload pictures (e.g: screenshot of something broken) to cards or to their comments, then an icon appears saying that there is an image. Clicking on it should show the uploaded picture.
- Drawing can be a feature.

- Suggestions: the action items are added to the respective column but it would be cool if we could know what issues they are connected to. Something like linking the action items to the things that need to be improved or went well (could be better). If the board is shared to someone who didn't attend the meeting, they visually understand what are the actions items extracted for each problem. And from each problem it could be possible to see the linked action items.
- Feature to identify keywords: button to show the most used words in each column. Then if we click on some word, they are highlighted in the cards - this is all about speed of getting more information.

Teams

- Size of last retro team: 5.
- Teams with large number of people could be great because the votes get more relevant but on the other hand, for the discussion it's not good if everyone wants to talk. To solve this there could be a timer mechanism and everyone who wants to speak has about 30 sec or 1 minute to do so.
- Other solution could be splitting the big team in smaller ones like we do in workshops. "So I will apply always this approach of breaking into teams, and then merging the work of everyone. So kind of clustering the problems and stuff like that."
- Usually one hour but depends on the frequency of the meeting. If the time gap between them is short they should be fast.

Utilizador 6

Jan 19, Wednesday 2022 - [redacted] - Interview Snapshot

Interviewee Data

Company: <KI Challengers>

Name: <Gabriel Cunha>

Role: <Product Manager, Head of Product>

Profile: <Align the vision and the strategy with all stakeholders. And aligning with all the different teams>

Topics

- Discussion on doing retro perspectives
- End-to-end process, steps and interactions of doing retros.
- Key challenges in doing retro perspectives in large teams
- 3rd party integrations

Key Data

Date:

Duration: <30 min>

Interviewer: < Anja Foerster >

Notes Taker: < Nuno Miguel Caseiro >

Meeting recording: <Recording Link>

Meeting Transcription: <Interview Transcription>

Opportunities

Codes for the learnings to find patterns across interviews

I need/want/struggle/think...

- I think people need a task management system to put their action items. Otherwise the action items get easily lost and will not be done.
- I think if you do voting face-to-face, others are influenced by others. Which is not good.
- I think a collaborative feeling and the possibility to interact with each other is a nice feature.
- I think voting via phone is a cool feature
- I want an AI to figure out why the action points were not done or why the work was not delivered
- I think for large/company wide retros, submitting the tickets and voting should be async. For discussing the topics the large amount of people will be splitted into breakout sessions to all discuss on the same priority topics and come up with action points.
- I think having some template for people to explain better their thoughts when they are writing a ticket is useful. If tickets are not understood, they can be marked to be explained.

Quotes

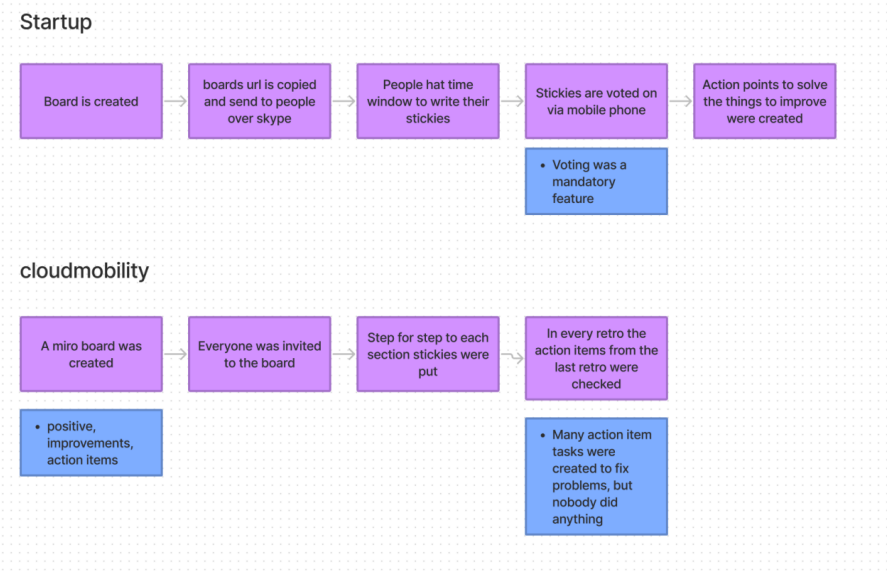
Suggestion - Amazon strategy for meetings: they are scheduled to have 30 minutes that are taken to people read something before the actual meeting start. In our retros these 30 minutes could be the time to read the added cards

In the book named "Team geek" by a former google manager its possible to learn that they structured their teams to follow three main principles: trust, respect and humility. This principles must be mandatory to team be effective.

Experience Map

End-to-end process described, key steps, and interactions of doing retro perspectives

(illustrative)



Interview Notes

Process

In Cloud mobility:

- Miro. The board was splitted in three sections: positive tickets, the improvements and action items.
- In every retro the action items from the last retro were checked to see what are the current status (if they are solved)
- A common thing in retros was that many tasks were created to fix the problems and nobody did anything. To solve this and to avoid losing items they were checked before the next retro starts. if they aren't solved they are taken into the next retro.
- Every sprint.

In a startup, running a product team:

- Retros in the end of every sprint (every 2 weeks)
- Sometimes the retros were remote because some people would not be in the office
- The board's url is copied and sent to people over skype.
- Then the people had a time window to write their stickers then in the meeting they are voted via mobile phone
- After the voting, they are discussed. Only one person is sharing the screen.
- Easyretrospectives - 2 columns (went well, to improve) - the action items are linked to the improve stickies. The voting system is a mandatory feature
- Could be a better way to track the action points, especially if they are assigned to someone and remind them they have things to do.
- The task management system for the develops was jira and nobody creates tickets for something like improving communication (non-technical). If people don't have a personal task management system to put their tasks on the action items, they can get lost
- the easyretrospectives tool were used and the votes were taken by phone.

Tools

- Easyretrospectives allows voting via mobile phones, which is nice.
- The annoying thing voting in face-to-face or with miro is that everyone sees the votes and are influenced by others. With phone, the votes are anonymous.
- The nice thing about miro is the collaborative feeling that you are actually in the same room because its possible to see people and interact with their cursor
- "I like the voting system via phone".
- "for me, writing the tickets is boring with the phone"
- AI to figure out why the action points were not done or why the work was not delivered.

Teams

- Cloud mobility: 7 people
- Startup: maybe 10

- This way to do the retros will not work with many people because it's impossible to have all the people explain their tickets in short period of time.
- For large number of people the retros should be async and there should be a period of time to submit tickets, other to reveal them and other to sanitize/review if they are misunderstood. The sanitization window is important to encourage people to write well because usually people just write "improve it". The tickets could also have some template for people explain better their thoughts when they are writing. If they are not understood, can be marked to be explained. After this phase comes up the voting phase (like 1 day). Then comes the meeting with the breakout rooms. The large team is divided into smaller groups with one person assigned to compile the discussion. All teams have the same priority tickets to discuss, and after half an hour they reconvene in the main room to talk about what was discussed in each small group.
- Suggestion - Amazon strategy for meetings: they are scheduled to have 30 minutes that are taken to people read something before the actual meeting start. In our retros these 30 minutes could be the time to read the added cards
- When the action items from the previous retros are discussed to understand why they didn't became solved the people usually say the real cause and it is never a personal attack. This is to say that people usually don't hold back something they want to say
- In the book named "Team geek" by a former google manager its possible to learn that they structured their teams to follow three main principles: trust, respect and humility. This principles must be mandatory to team be effective.

Utilizador 7

20, Thursday 2022 - [redacted] - Interview Snapshot

Interviewee Data

Company: <xgeeks>

Name: <Romeu Paz>

Role: <Operations Manager, Head of engineering>

Profile: <short desc of responsibilities>

Topics

- Discussion on doing retro perspectives
- End-to-end process, steps and interactions of doing retros.
- Key challenges in doing retro perspectives in large teams
- 3rd party integrations

Key Data

Date:

Duration: <30 min>

Interviewer: < Anja Foerster >

Notes Taker: < Nuno Miguel Caseiro >

Meeting recording: <Recording Link>

Meeting Transcription: <Interview Transcription>

Opportunities

Codes for the learnings to find patterns across interviews

I need/want/struggle/think...

- I think microsoft tasks is not the best tool to check up on action items. But it was the only one available
- Tracking on action items should be done like a development process. For example in jira.
- I need a tool which centralizes all our work. At the moment too many different tools are used to do different things.
- I struggle with choosing a responsible for managing the retro meeting. Newbies should not be chosen.
- I struggle with finding a way to export or save the retro.
- When doing company wide retros, we struggle with merging the cards from each team
- I need a feature which is linking the cards to a subject or related subject to easily merge similar topics.
- I need a possibility to reach anonymously out to a person who wrote a card where we don't understand the subject of what he /she wrote.
- I think anonymization for cards and for voting is important
- I need an AI for scheduling the meetings and for merging the cards.

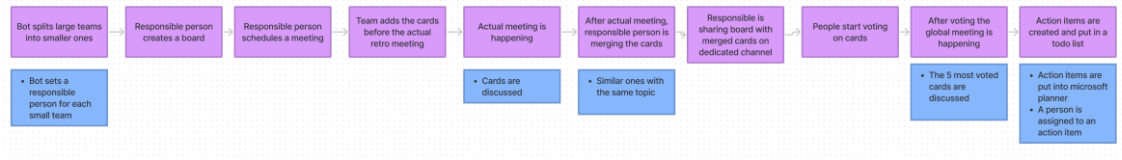
Quotes

> ***The biggest problem with retro is that people understand what retro is really for. The retro is very important for everyone and should not be seen as another meeting they have. They really need to focus in a real problems. Some issues can be solved with a short talk with someone.***

Experience Map

End-to-end process described, key steps, and interactions of doing retro perspectives

(illustrative)



Interview Notes

Process

- He doesn't participate in company retros until the last two steps.
- A bot splits randomly the large team into smaller ones and is set a responsible for each one of them
- The responsible person must create a board, schedule a meeting.
- After each team has their meeting, the responsibles join other meeting to merge cards that are similar or has at least the same objective.
- One of the responsibles should share the unified board on the dedicated channel to people start voting
- After voting there is the global retro review where all the responsibles, px and engineer teams join after the voting phase to sort the cards by number of votes
- Then the five most voted cards are discussed and are defined the action items in a todo list - tasks from microsoft
- Sometimes the action items can be explained in the all hands but actually we don't really do that because that meeting is already too long
- Usually the actions points are completed because there is a person assiged to that
- To measure the action items there are metrics for this but they are usually not checked
- The microsoft tasks is not the best and preferred tool but the one that was available

- The tracking should be done like an engineering process. Jira or something to management
- Should be integrated with the day to day things we have. Different tools are used to manager our work and we should have a centralized tool to handle that.
- The only fixed schedule is the global retro, every first Tuesday of every month. The retro week is before the global
- The biggest problem with retro is that people understand what retro is really for. The retro is very important for everyone and should not be seen as another meeting they have. They really need to focus in a real problems. Some issues can be solved with a short talk with someone.
- "The last retro step is a retro about the retro."

Tools

- One of the biggest problem is selecting people because we have a rule that people with at least three months in the company should not be responsible.
- We dont have a record of all retros.. i do a print and put it on confluence.. there is no archive or other way to save/export the retro

Teams

- The teams should not be small or too big. As the company continues to grow, the problem will be merging cards. We will have a lot of small teams trying to merge cards and it will be problematic. If there is a meeting with 15 people trying to merge cards, that will be difficult.
- Something that helps the merging process could be helpful. Something like linking the cards to a subject or related subject to easily merge subjects
- Sometimes the context is needed to understand what the card is talking about and these discussions are in the small teams, however, sometimes we don't get that context because they are anonymous. So we tried to create a template: card's subject and possible solution. And if they want to have more context put in the comments, but sometimes it doesn't work as expected.
- To solve this maybe it would be possible to anonymously reach out the person who wrote the card and get a response.
- People don't want to speak or have scared so the anonymisation is important. "I think anonymisation it's always better because of voting".
- AI? scheduling meetings and joining the cards are the two main things. Vacations - the bot dont know when people are in vacations so sometimes the chosen responsible may be on vacations. this should be fixed. Possible solution: having a second responsible to replace the main one if its needed.

Utilizador 8

20, Thursday 2022 - [redacted] - Interview Snapshot

Interviewee Data

Company: <xgeeks>

Name: <Fabio Ferreira>

Role: <developer>

Profile: <short desc of responsibilities>

Topics

- Discussion on doing retro perspectives
- End-to-end process, steps and interactions of doing retros.
- Key challenges in doing retro perspectives in large teams
- 3rd party integrations

Key Data

Date:

Duration: <30 min>

Interviewer: < Anja Foerster >

Notes Taker: < Nuno Miguel Caseiro >

Meeting recording: <Recording Link>

Meeting Transcription: <link/embedded/not available>

Opportunities

Codes for the learnings to find patterns across interviews

I need/want/struggle/think...

- I think there is no need to anonymize the cards when all people trust each other. They can then give feedback in a constructive way
- I need a retro tool which is flexible enough to change the structure of the board columns
- I need an AI which helps to remind people to stay in time when they deeply discuss a topic
- I want a tool which tracks/makes visible action items. Otherwise they get lot in limbo.
- I think in retros with many people it is necessary to merge the cards and vote on them
- I need an ice breaker at the beginning of retros. It helps to make people more relaxed and be more open.
- I think a time limit for each step of the retro helps to not pass 1 hour. Which I think is the time limit for the whole retro.

Quotes

Remarkable Quotes

Experience Map

End-to-end process described, key steps, and interactions of doing retro perspectives

(illustrative)

Interview Notes

Process

- He did remotely retrospectives inside and outside of the team's project
- The retros were done at the end of each sptring and there was a person in charge to schedule the meeting.
- This person in charge could be different in each retro. The next person was chosen by who hosted the last time. (10 people)
- Some people added their cards before the meeting others did it during the session
- They had a place to add card or topics - confluence page. This had 3 sections: went well, things didn't go well and action items.
- 5-10 minutes to write them and then all the topics were discussed. From the discussion, action items were extracted
- The action items were assigned to someone if it was meaningful. Some tasks could be for the whole team and for those tasks there was no need to be assigned to anyone, unless we want to reinforce some behaviour
- In the next retro we looked at previous retro actions to see if they are done.
- Sometimes the action tasks get lost into limbo
- It would be helpful if I can see/track the action items that are being solved and measure them (actual status). Because last time he did a retro they copied the action items from the last retro to the actual one and there is no way to track the progress of them.
- For the scheduling they decided a specific period in which the retros should be done (at the end of each sprint). Sometimes were hard to find a host because they are doing their job and no one wants to spend time on preparing and doing the retro.
- Having different hosts is a good idea because the team isn't stuck with the same ideas.

Tools

- Outlook for scheduling (an invitation to an event is sent)
- Documentation/board on confluence.
- In confluence it has limit of 10 people editing at the same time which is a pro and con. The same related content are not merged but is discussed.
- The cards/topics are not anonymous. No need if all people trust everyone and can give and receive feedback in a constructive way.
- Critical things to adopt a retro tool: ways to define which type of cards I want to have like went well, to improve... (add and edit columns) -> Flexibility in the structure of the board
- AI? An external "person" to advise the team that they are talking about something over too long.

Teams

- 10 people
- In large teams the similar topics must be merged and choose the most important ones to discuss. These topics could be chosen by the host or the most voted cards.
- To solve retros with many people at the same time the large team can be split in smaller groups. And each smaller group could have a person in charge who speaks at the end with others responsibles.
- The retros could have a fun topic to discuss.. to break the ice. They should be relaxed
- No issues during retros.
- Time limit? Some discussion might take few minutes but others might need more time. Max 1h.
- In order to optimize the retros, they can have a timeline for each step. e.g. 10 min to write, 30 to discuss and the remaining minutes for checking the action items

Utilizador 9

Jan 24, Monday 2022 - [redacted] - Interview Snapshot

Interviewee Data

Company: <name>

Name: <Jean Wichert>

Role: <role>

Profile: <short desc of responsibilities>

Topics

- Discussion on doing retro perspectives
- End-to-end process, steps and interactions of doing retros.
- Key challenges in doing retro perspectives in large teams
- 3rd party integrations

Key Data

Date:

Duration: <30 min>

Interviewer: < Anja Foerster >

Notes Taker: < Nuno Miguel Caseiro >

Meeting recording: <Recording Link>

Meeting Transcription: <Transcription>

Opportunities

Codes for the learnings to find patterns across interviews

I need/want/struggle/think...

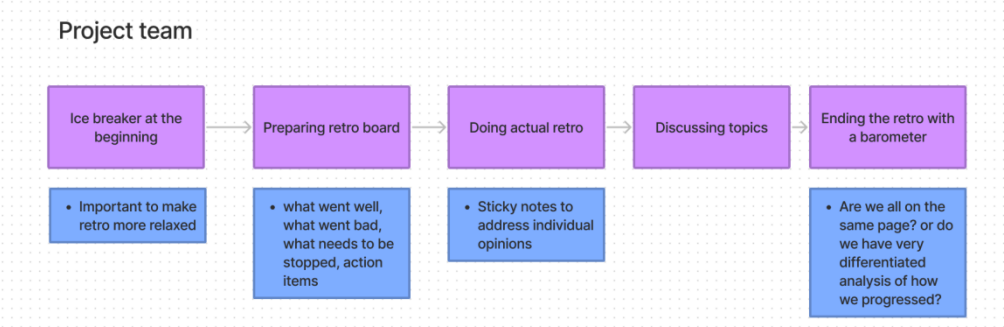
- I think an ice breaker before doing the actual retro helps to make it a non-stressful situation.
- I think a barometer [0- 100%] on how well the last sprint went, helps to figure out if everyone is on the same page.
- I think the most important thing in the retro with large teams is the voting process.
- After voting I want to see something like a heat map to distinguish the most important topics visually.
- I think when doing retros company wide, people should be splitted into teams around 10-15 people
- I think concentration can get very low if the retro is taking too long
- When doing retros company wide, people are struggling with the concept of scrum. They don't understand why a retro is important.
- I think a tool which lets people interact helps to bring value to the actual retro board.

Quotes

Experience Map

End-to-end process described, key steps, and interactions of doing retro perspectives

(illustrative)



Interview Notes

- SAFe allows multiple agile teams to work together on same goals. They started with a SAFe context with 7 teams. His team was composed 7 members mostly of them were developers but also a wingman and a ux designer. They were more focused on the backend development of the magento system.
- In the SAFe context more additional ceremonies were added to interact with other teams. This is done with a ART sync. This means that every week, all the product owners and scrum masters from each team, get together in one/two hour session and try to map out their team's progress. Their each swimlanes but also find out what kind of dependencies and impediments when collaborating with the other teams. Which means they need to work together and try to find out on which touch points they have to discuss and try to evaluate the work to do. when the art train starts - when you try to start the progressing or the progression of the work - you run into challenges and some obstacles when you're trying to fulfill you work because you're depending on others

Process

Project team:

- Inside each team they performed the typical scrum ceremonies - at the end of the sprint it is reviewed to evaluate their progress.
- After this they attend a one-hour retrospective where they identify different types of findings during the sprint. how was the work successful for you?; how did you progress the way you intended to? do you have the opinion that we create some value?
- They used a format to make an icebreaker at the beginning - they used giphy to express their mood, how they felt and it was pinned on a miro board and everyone had a chance to explain why they choose that gif that was pinned many times got a good laugh and it became a non-stressful situation.
- After that they could go into the things that went well, what went bad and what they need to stop - action item - and everyone could use a sticky note to address their individual opinion
- Then they discuss them because everyone had to explain why he selected this issue or topic on the note
- At the end of each retro a kind of barometer [0- 100%] was used to rate how good they achieved in the current sprint. They had stars to pin on the corresponding percentage to make a overall feeling. "are we all one the same page?" "or do we have very differentiated analysis of how we progressed?

Company wide:

- This happened mostly at the end of PI - program increment of five sprints, each 2 week long. - end of 10 weeks.
- Almost 100 people - it isn't so personal as the sprint format
- It's more on an organizational level to evaluate the PI - everybody is ok with methods we are using? what should we start, stop and revaluate?
- Zoom to videoconferences. Miro board.
- In some PIs (program increment) there is a preparation phase - meeting beforehand within each team to find out what they want to express during the PI and find out the topics that are most important. (development or organizational topic)
- PI's were executed in a two day workshop - objective: bring the work into a structure so that everybody knows what's gonna happen the next 10 weeks.
- The retrospective was at the end of the second day, where the stakeholders could address their most important topics or subjects to the audience. And everyone also tried to put a value or an importance tagged to each of the proposed recommendations (topics)

- So the scrum master of scrum masters could collect and try to structure the topics but also the RTE, the one that is actually responsible for the complete execution of the safe value stream and for moderating the whole discussion, try to emphasize on the topics that are relevant, also the quick wins and also revaluate the working methods that are being used in the current planning (A value stream is the set of actions that take place to add value for customers from the initial request through realization of value by the customers)
- All the people participate in the same miro board and the most important action from the participants is the voting. That recommendations /notes/cards are added to a list or a board and people must vote. From this it's possible to see the heat map.. the most relevant and important things.

- Participants in retro: RTE and scrum master of scrum masters and one additional help. around 10% were scrum masters.
- Each team has it's scrum master. In order to communicate with 100 people in a zoom session, we can split the whole group of people into a topic related breakout rooms so they can switch from the main room to a breakout room and have a conversation with maximum of 10 to 15 people.
- From the SAFe context there are guidelines that help process our thoughts into some information or context. That means that there are different boards inside the miro board. e.g. PI planning board: all teams in vertical and all iterations or sprints in horizontal -> map. Where everyone places their topics into their corresponding cell;
- Another sub board is the roam board where people can post there their problems, challenges they have and it's possible to see a dependency map. So if a topic from one team depends on a topic from other team, there is a line connecting them.
- There are templates where it is pre formatted and explains what is needed to be done there.

- After one day and half of total concentration and content producing,people start to become exhausted. At this point, the concentration levels are very low. The big challenge is how to mantain a level of concentration and participation at the end of such a planning. Another challenge is that many people were not familiar with this scrum methods. "What this is good for, and why does this, why does this make sense to use the continuous improvement cycle?". After the 3rd PI, people usually see the benefits.

Another way of doing retros in this large scene?

- This approach is good if you could insert more breaks or structure it in three days with high concentration.
- Face-to-face can be a different experience however in this cause a miro board will be difficult to use

Issues in doing this large retros?

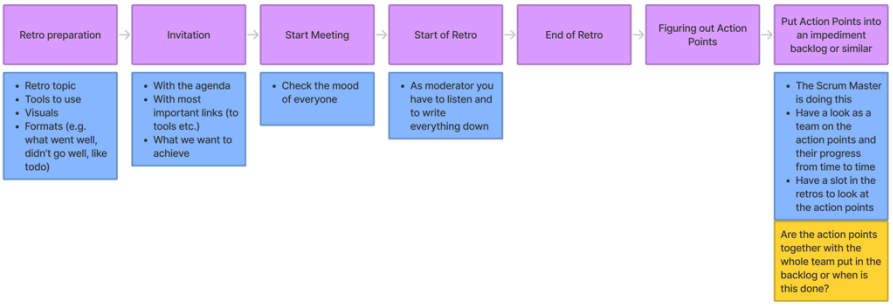
- No, they went pretty well. There was no feeling about chaos. Some moderators are needed to route the ideas and you always have a so called think tank or bucket. We put something there and concentrate on this.

Tools

- Miro has a high level of interactivity. It's not a front presentation, it's more interactive. The majority of people are using it actively and not just as reader but also contributing to or adding value to the board.

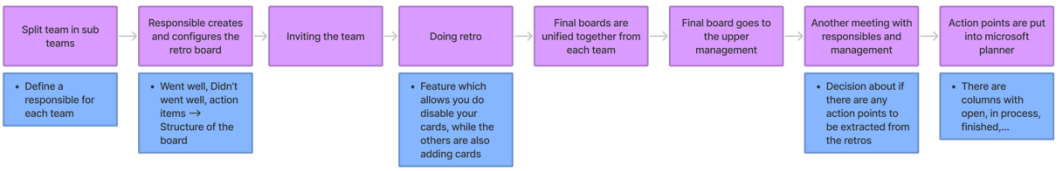
Apêndice D – Mapeamento das experiências

Utilizador 1

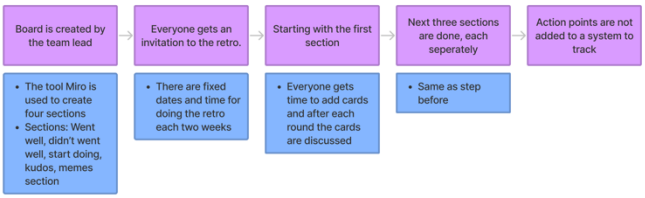


Utilizador 2

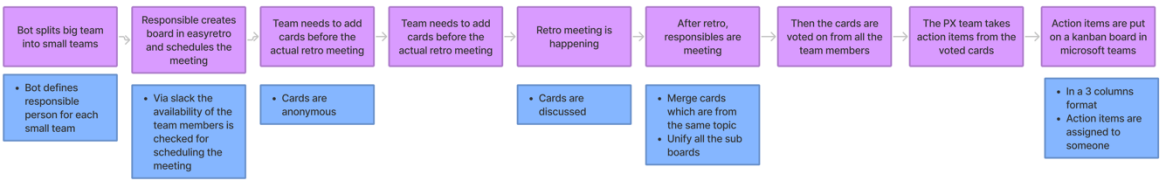
Large teams



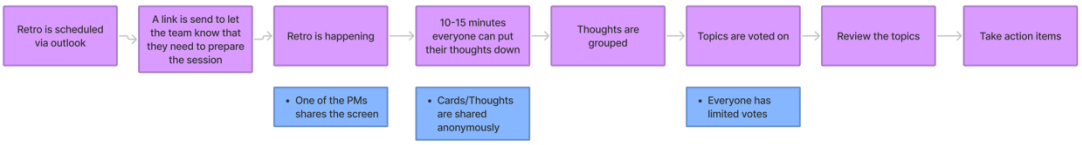
Small teams



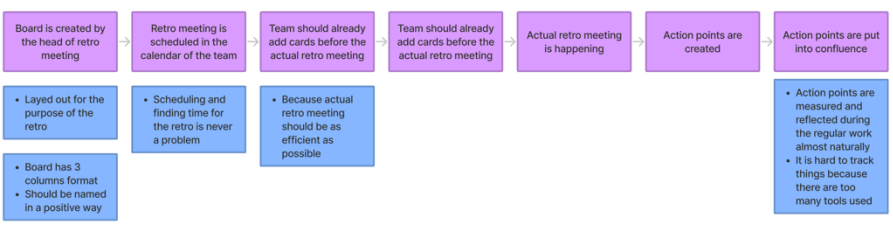
Utilizador 3



Utilizador 4



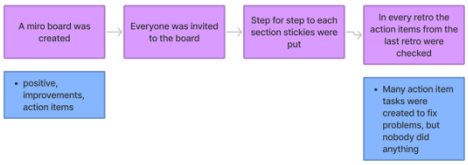
Utilizador 5



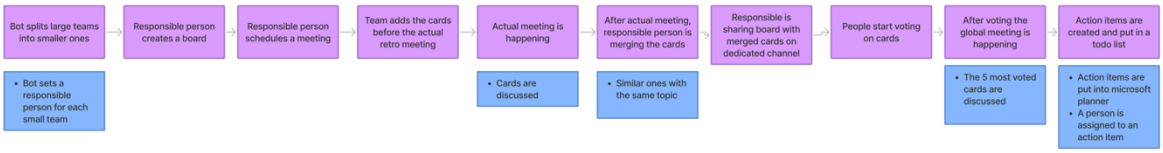
Utilizador 6

Startup

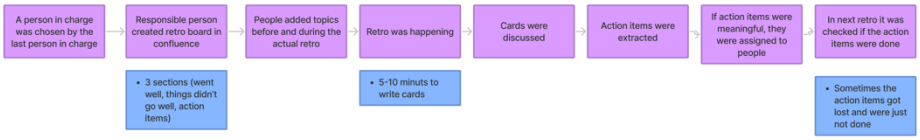
Company project



Utilizador 7

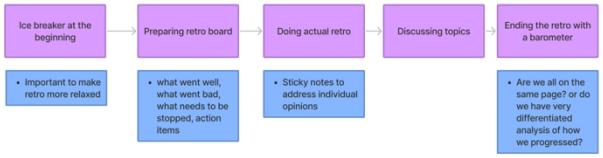


Utilizador 8



Utilizador 9

Project team



Apêndice E – Airtable de descobertas

Finding Description	Confidence	Impact	Themes	Evidence from
I need a tool which tracks the action items and reminds me of action points I need a feature to track/subscribe the status of the action items after the retro. Think the weakest part of the process is taking the action items and then doing something with them on a regular basis. I struggle with finding a way to track things (action points), because we have too many tools for different steps of the retro. It should only be one tool for everything. I think people need a task management system to put their action items. Otherwise the action items get easily lost and will not be done. I want an AI to figure out why the action points were not done or why the work was not delivered I think that some action points are just getting lost after doing the retro I think microsoft tasks is not the best tool to check up on action items. But it was the only one available	High (4-7)	High	Tool to track action items Subscribe to action items Working on action items Too many tools for different steps AI for action points tracking Action points get lost Tool to centralize work	Utilizador1 Utilizador2 Utilizador3 Utilizador4 Utilizador5 Utilizador6 Utilizador7 Utilizador8

I need a tool which centralizes all our work. At the moment too many different tools are used to do different things I want a tool which tracks/makes visible action items. Otherwise they get lot in limbo.				
I think teams for retros should not be bigger than 7-8 people otherwise people loose the focus or don't feel that secure to talk. I think if there are too many people they just don't talk I think people should be splitted to smaller teams into breakout sessions to all discuss on the same priority topics and come up with action points. I think when doing retros company wide, people should be splitted into teams around 10-15 people	High (4-7)	High	Team size 7-8 Smaller team size Breakout rooms	Utilizador1 Utilizador2 Utilizador4 Utilizador6 Utilizador9
I think limiting the votes is essentially for the people to choose what is really important I think the voting system is very relevant to show what are the real problems I need a voting system / and the feature to rank the issues/problems I think if you do voting face-to-face, others are influenced by others. Which is not good. I want a feature that gives visual importance to the most voted/critical topics	High (4-7)	High	Limiting votes Voting for identifying problems Influencing others when face-to-face-voting Visual importance	Utilizador3 Utilizador4 Utilizador5 Utilizador9

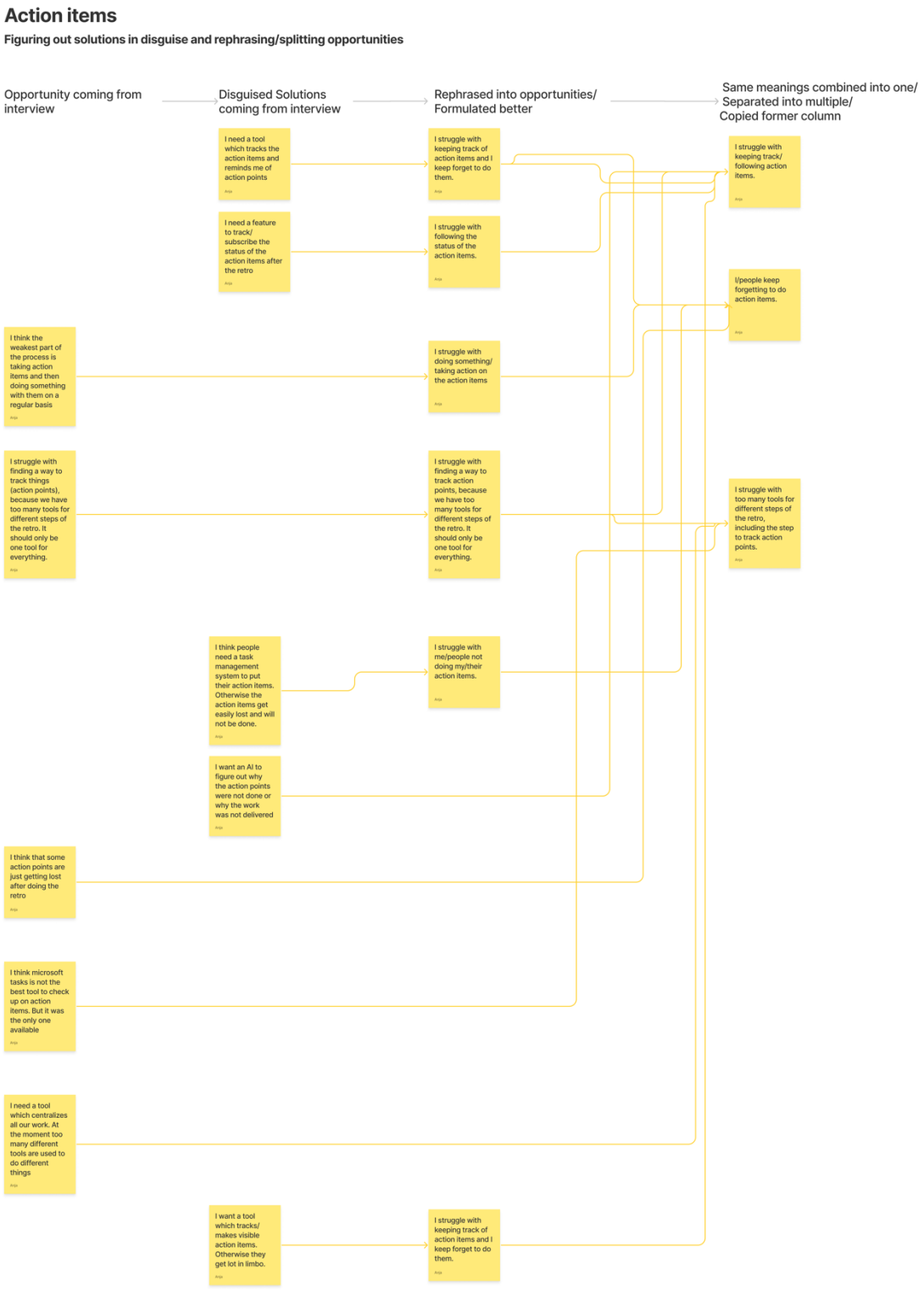
After voting I want to see something like a heat map to distinguish the most important topics visually. For large teams/companywide retros voting should be async			Async voting	
I want visuals to support each retro phase I want a meme section to make retros more fun I think it's important that a retrospective is approached in a positive way and with positive moods. I want visuals to support comments/cards I want a drawing feature I think a collaborative feeling and the possibility to interact with each other is a nice feature. I think a tool which lets people interact helps to bring value to the actual retro board I want metaphors for the columns I want gamification to make retros more fun	High (4-7)	Medium	Fun visualization Funny memes Positive approach Drawing feature Collaboration	Utilizador1 Utilizador2 Utilizador5 Utilizador6 Utilizador9
I need a tool when doing large team retros, which requires less manual work to order, categorize and merge the cards	Medium (2-3)	High	Automatically merging cards	Utilizador2 Utilizador7

When doing company wide retros, we struggle with merging the cards from each team I need a feature which is linking the cards to a subject or related subject to easily merge similar topics. I need an AI for merging the cards. I think in retros with many people it is necessary to merge the cards and vote on them			Automatically categorizing cards Struggle with merging cards Link cards to subject AI for merging cards	Utilizador8
I need anonymization of the cards I think that hiding/showing cards is good, because they don't influence other people when they are adding cards I think anonymization for cards and for voting is important	Medium (2-3)	Medium	Anonymization of cards	Utilizador2 Utilizador3 Utilizador7
I need the retro meetings to be time bounded, because meetings are very expensive I sometimes struggle to find enough time to discuss everything that is important within one retro. I need an AI which helps to remind people to stay in time when they deeply discuss a topic I think a time limit for each step of the retro helpsto not pass 1 hour. Which I think is the time limit for the whole retro.	Medium (2-3)	Medium	Time limit Not enough time to deeply discuss	Utilizador4 Utilizador5 Utilizador8

I want a feature to identify keywords I need a feature which is linking the cards to a subject or related subject to easily merge similar topics.	Medium (2-3)	Medium	Keyword identifier Linking cards to subjects	Utilizador5 Utilizador7
I need to give everyone on a team the opportunity and the right environment to discuss about their issues and to express themselves. Especially in a remote environment I need an ice breaker at the beginning of retros. It helps to make people more relaxed and be more open. I think an ice breaker before doing the actual retro helps to make it a non-stressful situation	Medium (2-3)	Low	Provide environment of trust Provide ice breaker at beginning	Utilizador1 Utilizador8 Utilizador9
I need to feel the mood and the expressionsof the people participating in retros I think retros is about the team talking to each other and forming a kind of bond	Medium (2-3)	Low	Feel mood of people Forming bonds	Utilizador1 Utilizador4
I struggle to find a time slot for doing the retro, where everyone is free. I need a tool wich integrates automatically the retro into peoples calendar. Maybe in combination with HR tools like personio I want an AI to help with the organization and scheduling process before the actual retro.	Medium (2-3)	Low	Scheduling meeting tool	Utilizador2 Utilizador3 Utilizador7

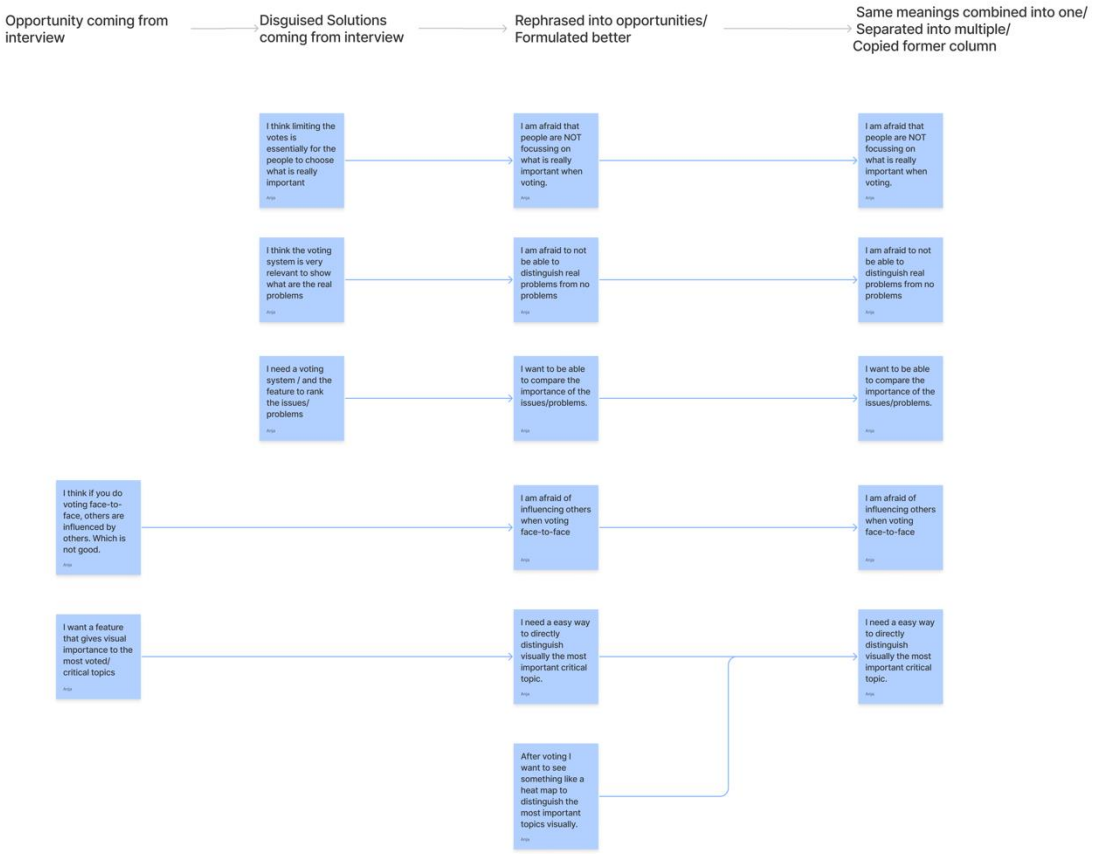
I need an AI for scheduling the meetings				
I struggle to understand the meanings of the cards sometimes. I need a possibility to reach anonymously out to a person who wrote a card where we don't understand the subject of what he/she wrote.	Medium (2-3)	Low	Meaning of cards Anonymously reaching out to people	Utilizador3 Utilizador7
I need the action item linked to the respective issues	Low (<2)	Medium	Link action items to respective issues	Utilizador5
I think for large/company wide retros submitting ticket should be async	Low (<2)	Medium	Async ticket submission	Utilizador6
I want the rating to be fun and creative	Low (<2)	Low	Creative/fun rating	Utilizador1
I want a feature which is automatically documenting what people say when considering doing retros with really big teams.	Low (<2)	Low	Automatic documenting	Utilizador1
I need the drag and drop experience	Low (<2)	Low	Drag and drop	Utilizador5
I think having some template for people to explain better their thoughts when they are writing a ticket is useful	Low (<2)	Low	Template for creating cards	Utilizador6
I want to have the ability to choose from a template for the retro board	Low (<2)	Low	Retro board templates	Utilizador3
I struggle with finding a way to export or save the retro.	Low (<2)	Low	Exporting/saving retro	Utilizador7

Apêndice F – Mapa de oportunidades



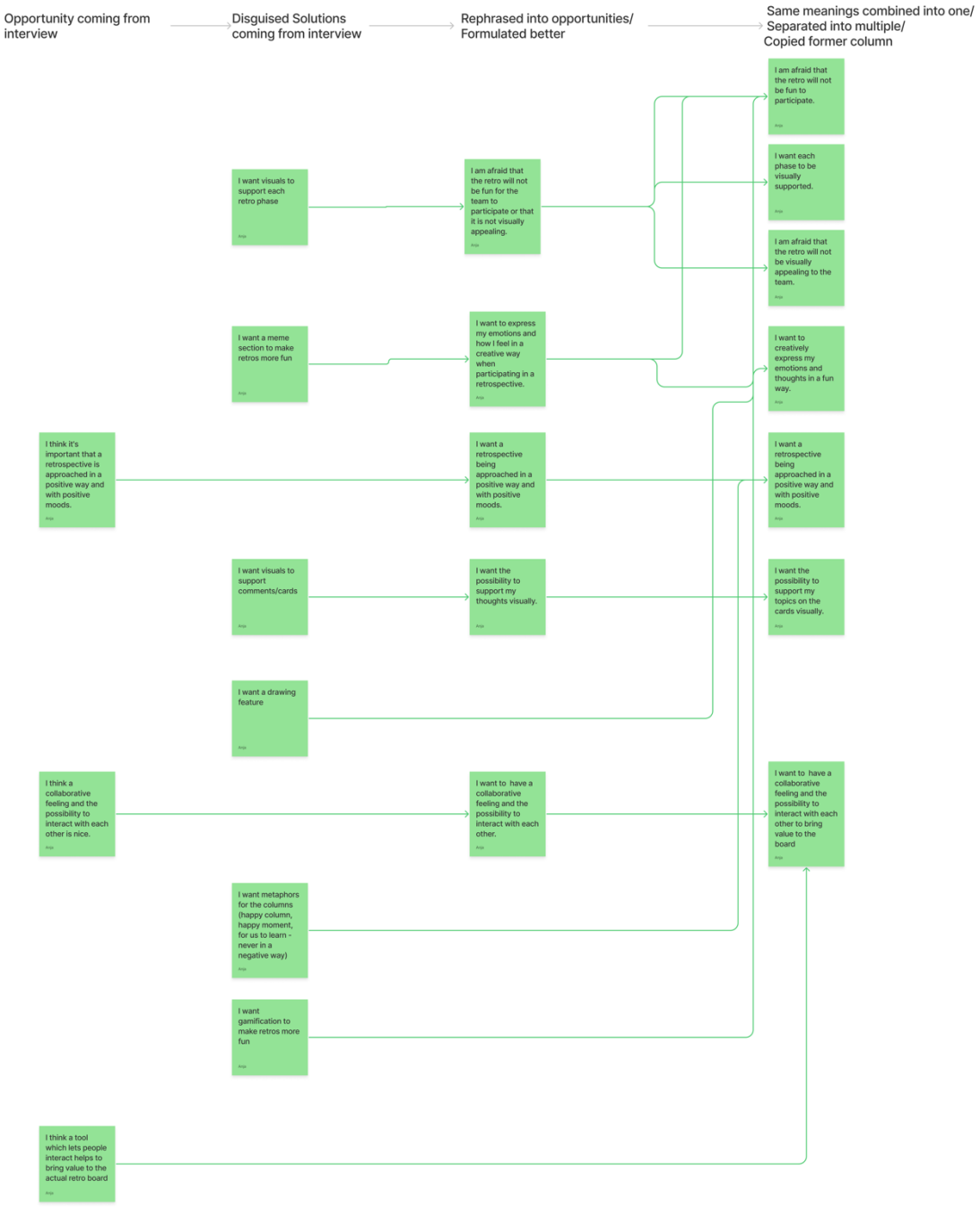
Voting

Figuring out solutions in disguise and rephrasing/splitting opportunities



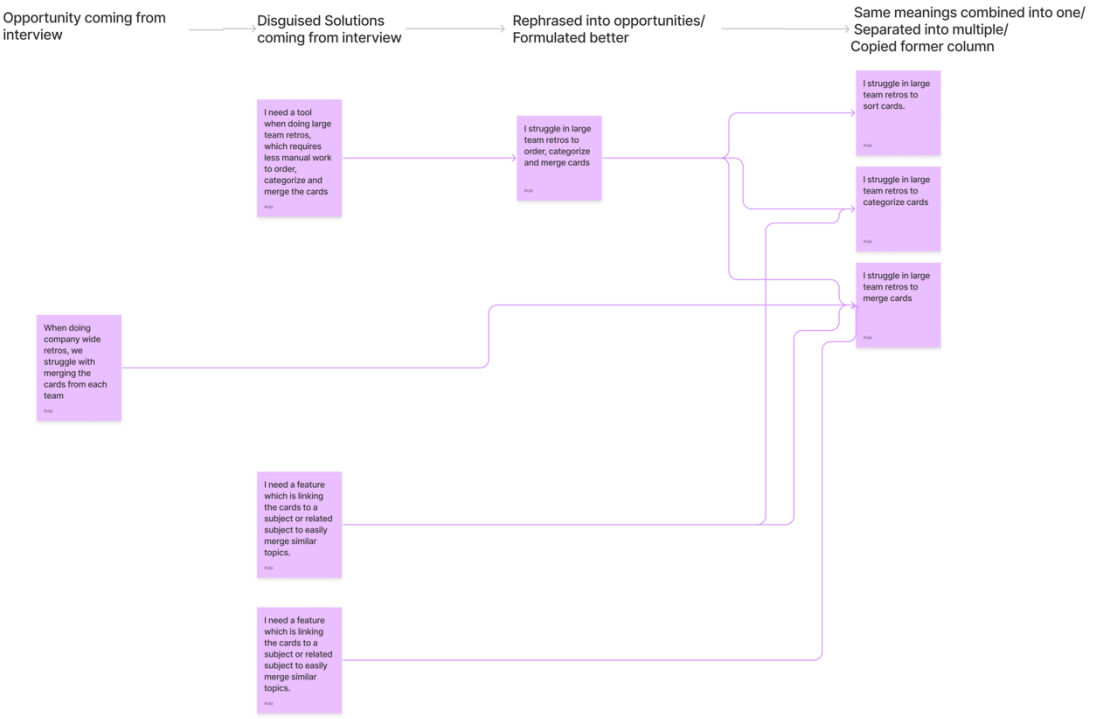
Visual support, making a fun retro

Figuring out solutions in disguise and rephrasing/splitting opportunities



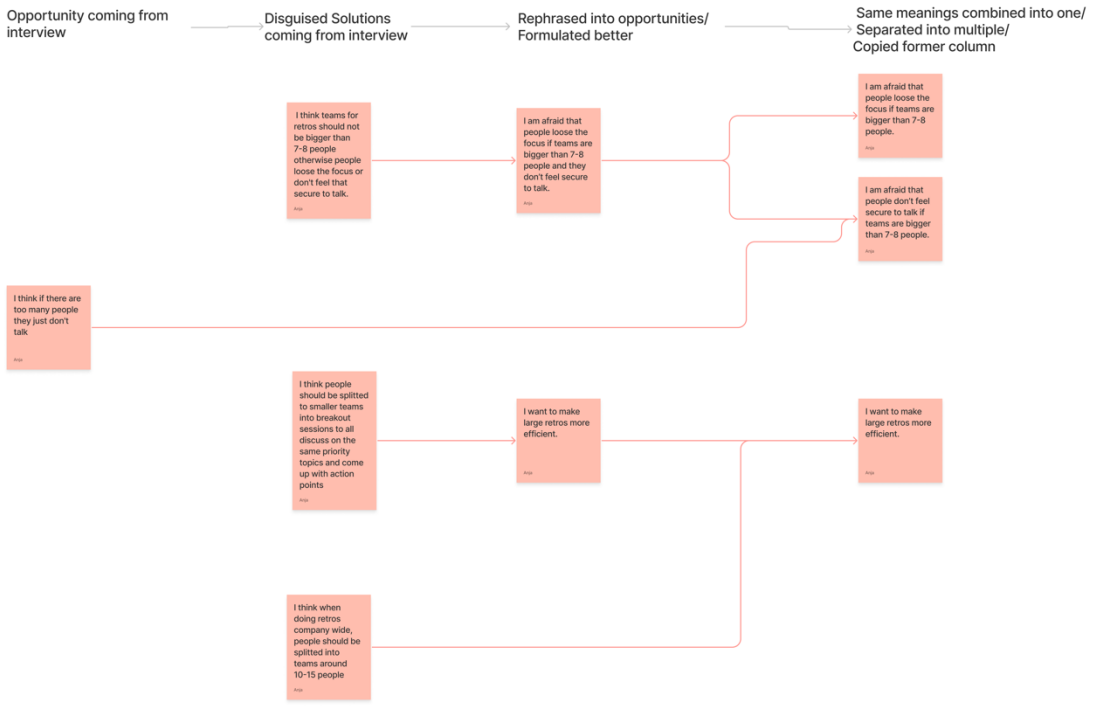
Merging cards

Figuring out solutions in disguise and rephrasing/splitting opportunities

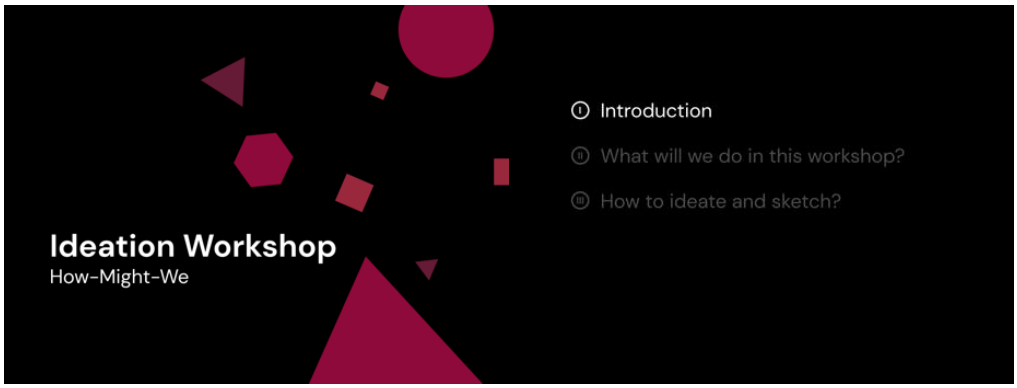


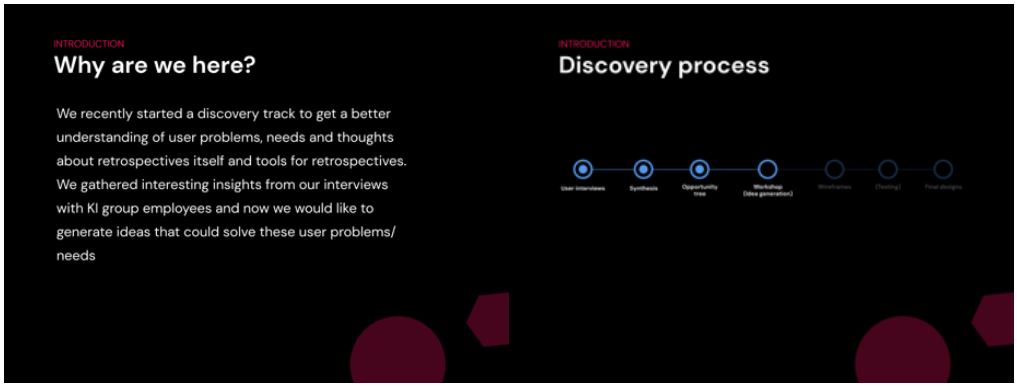
Team size

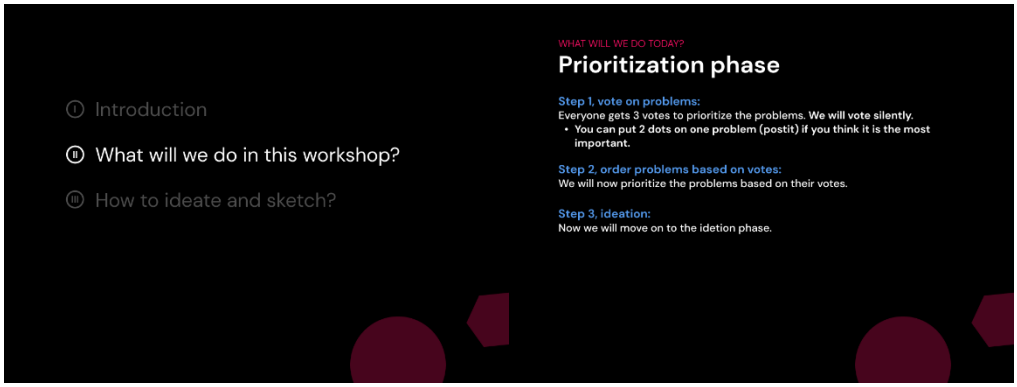
Figuring out solutions in disguise and rephrasing/splitting opportunities

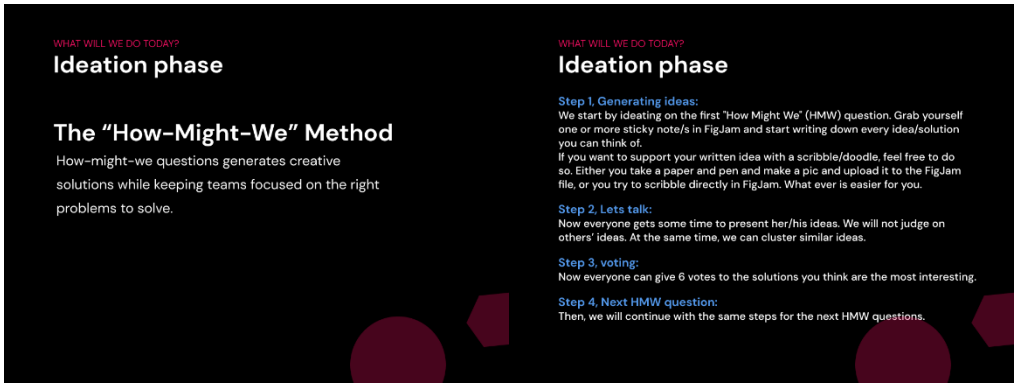


Apêndice G – Apresentação do workshop

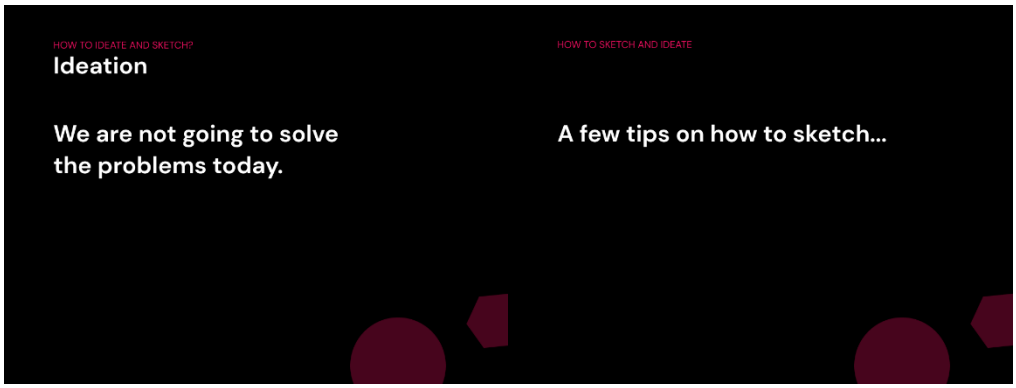


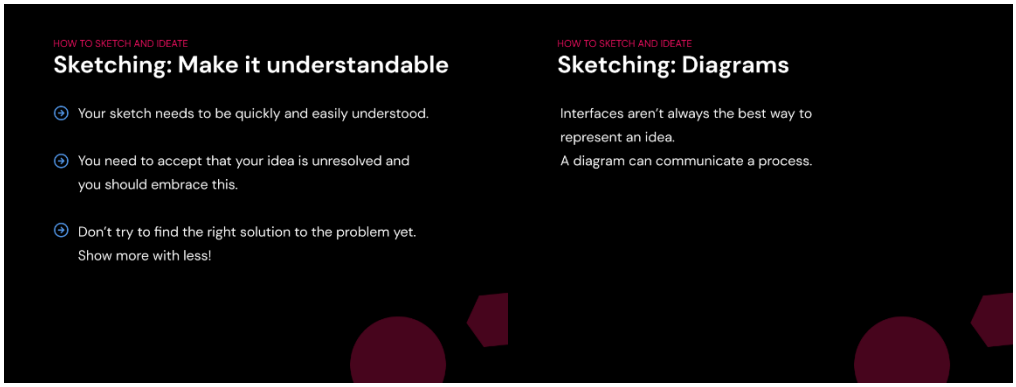




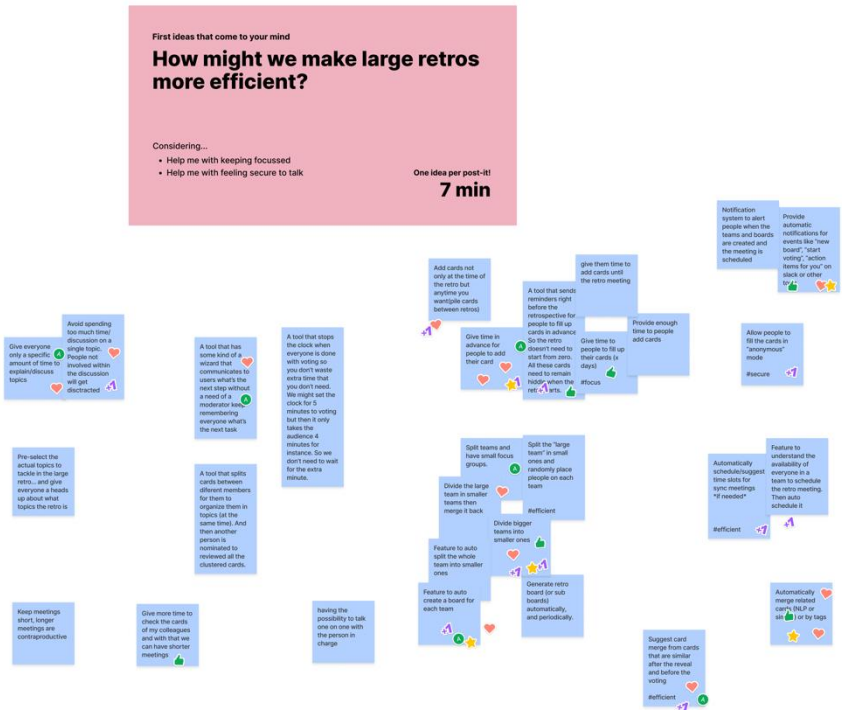


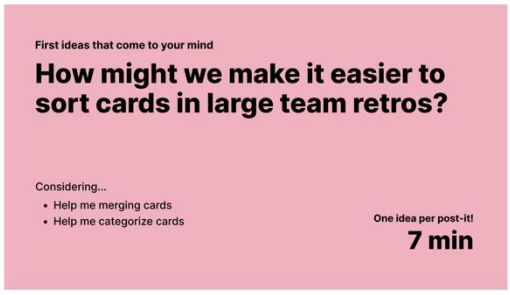

Introduction
What will we do in this workshop?
How to ideate and sketch?
HOW TO IDEATE AND SKETCH?
Ideation: What to do
One sticky note per idea
Produce as many ideas as possible. Its about quantitiy, not necessarily quality.
You can build on ideas of others, too
Encourage wild ideas
Stay within the time


HOW TO IDEATE AND SKETCH?
Ideation
We are not going to solve the problems today.
HOW TO SKETCH AND IDEATE
A few tips on how to sketch...

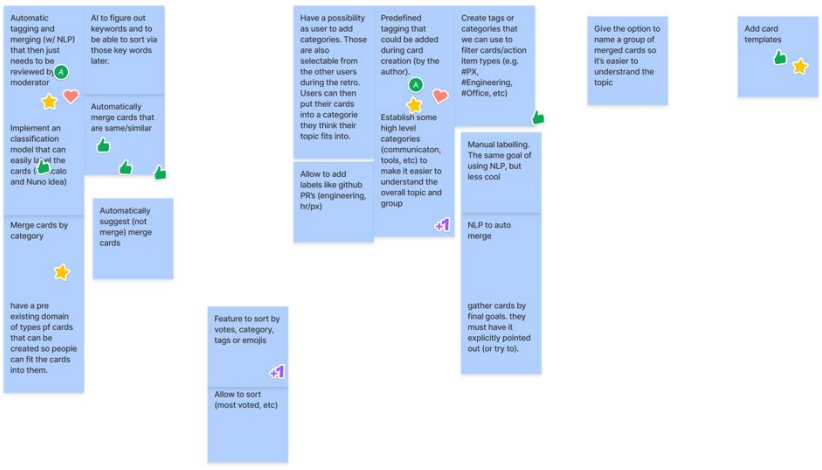

HOW TO SKETCH AND IDEATE
Sketching: Make it understandable
Your sketch needs to be quickly and easily understood.
You need to accept that your idea is unresolved and you should embrace this.
Don't try to find the right solution to the problem yet. Show more with less!
HOW TO SKETCH AND IDEATE
Sketching: Diagrams
Interfaces aren't always the best way to represent an idea.
A diagram can communicate a process.

Apêndice H – Rondas do workshop


First ideas that come to your mind
How might we make large retros more efficient?
Considering...
• Help me with keeping focussed
• Help me with feeling secure to talk
One idea per post-it!
7 min


First ideas that come to your mind
How might we make it easier to sort cards in large team retros?
Considering...
• Help me merging cards
• Help me categorize cards
One idea per post-it!
7 min

Anjaaaaaa


Automatic tagging and merging (w/ NLP) that then just needs to be reviewed by moderator
AI to figure out keywords and to be able to sort via those key words later.
Automatically merge cards that are same/similar
Implement an classification model that can easily b
cards (
calo
and Nuno idea)
Automatically suggest (not merge) merge cards
Merge cards by category
have a pre existing domain of types pf cards that can be created so people can fit the cards into them.
Have a possibility as user to add categories. Those are also selectable from the other users during the retro. Users can then put their cards into a categorie they think their topic fits into.
Allow to add labels like github PR's (engineering, hr/px)
Predefined tagging that could be added during card creation (by the author).
Establish some high level categories (communicaton, tools, etc) to make it easier to understand the overall topic and group
Create tags or categories that we can use to filter cards/action item types (e.g. #PX, #Engineering, #Office, etc)
Manual labelling. The same goal of using NLP, but less cool
NLP to auto merge
gather cards by final goals. they must have it explicitly pointed out (or try to).
Give the option to name a group of merged cards so it's easier to understand the topic
Add card templates
Feature to sort by votes, category, tags or emojis
Allow to sort (most voted, etc)

Apêndice I – Funcionalidades

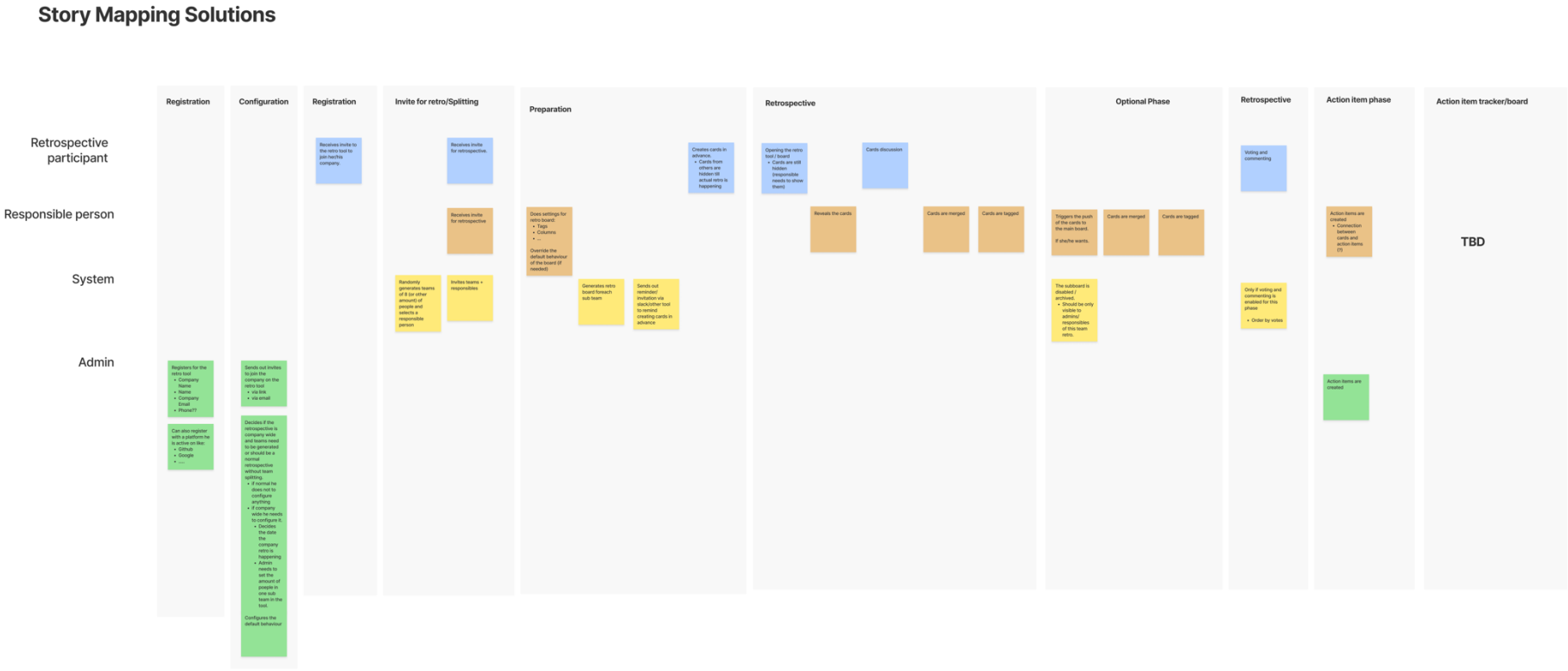
Tópico	Funcionalidade
Fase 1 - MVP	
1.1 Autenticação básica	Login
	Registo
	Logout
	Recuperar password
1.2 Dashboard	Criar quadros
	Listar quadros
	Apagar quadros
	Partilhar quadros
	Estatísticas
	Filtros
1.3 Quadro	Adicionar colunas
	Editar colunas
	Mover colunas
	Público / Privado
	Adicionar cartões
	Editar cartões
	Apagar cartões
	Juntar cartões
	Separar cartões
	Mover cartões entre colunas
	Listar cartões nas colunas
	Adicionar comentários (publico ou anónimo)
	Editar comentários
	Apagar comentários
	Listar comentários
	Adicionar votos
	Remover votos
	Limitar e configurar o número de votos

	Alterar nome do quadro
	Esconder os cartões
	Esconder os votos
	Ordenar os cartões por votos (ascendente e descendente)
	Interação em tempo real
	Adicionar / remover utilizadores
	Listar os participantes
	Temporizador
Fase 2 - Utilizadores e equipas	
2.1 Dashboard de utilizadores	Criar utilizadores
	Listar utilizadores
	Apagar utilizadores
	Definir super-administrador
	Pesquisar utilizador
	Detalhes de utilizador
2.2 Dashboard de equipas	Criar equipas
	Listar equipas
	Apagar equipas
	Adicionar utilizadores à equipa
	Remover utilizadores da equipa
	Pesquisar equipas
	Definir funções e permissões para cada utilizador numa equipa
2.3 Quadro de equipas	Criar quadro a partir de uma equipa
	Listar quadros por equipa
	Dividir equipa em sub-equipas e respetivo quadros
	Dividindo a equipa, definir o número de subequipas ou número de membros
	Dividindo a equipa, definir responsáveis de cada sub-equipa
	Listar os participantes

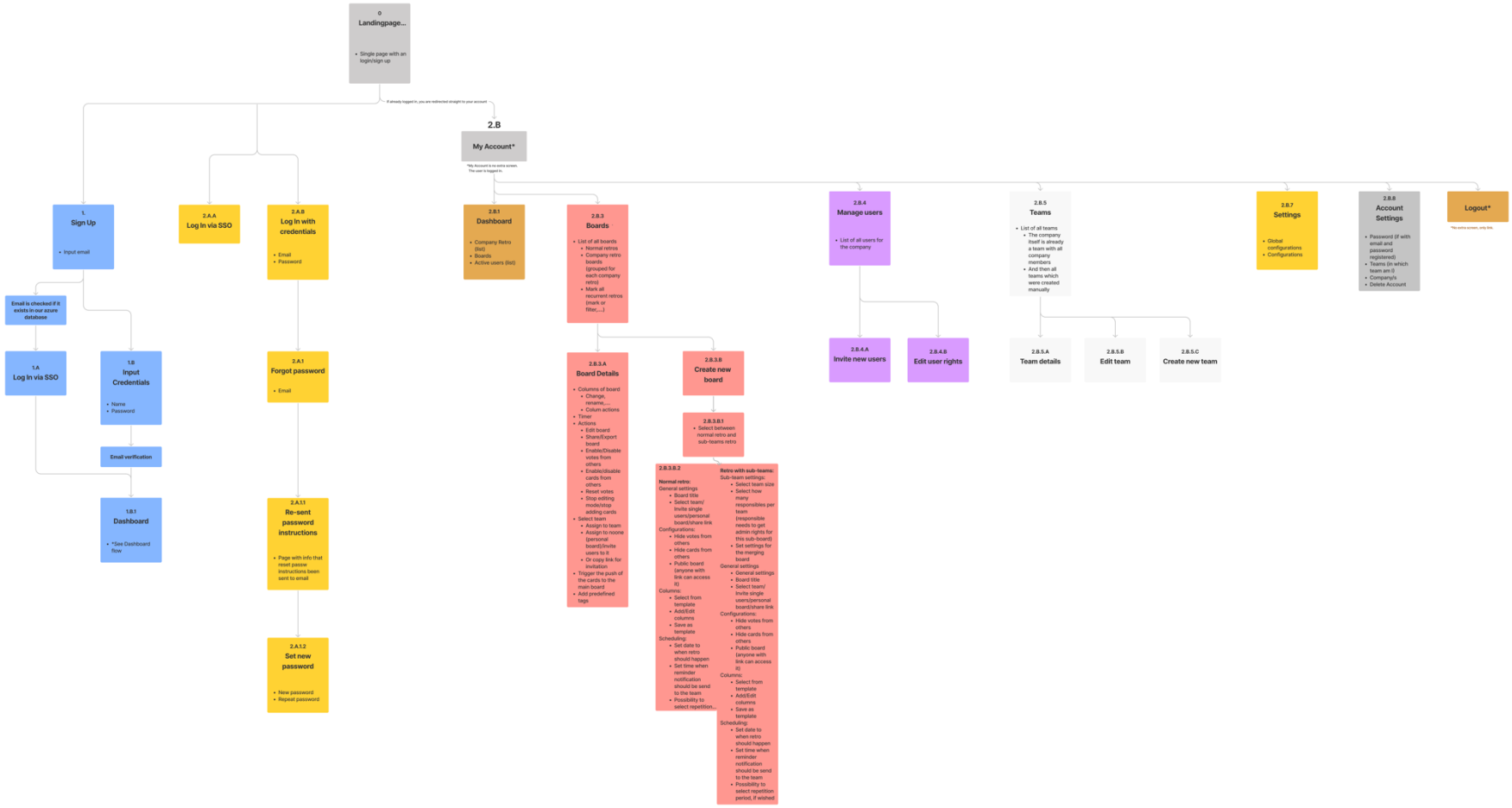
	Agendamentos mensais (divisão de uma equipa em sub-equipas, mensal e automaticamente, gerando os respetivos quadros)
	Bloquear um sub-quadro e copiar os seus cartões para o quadro principal
Fase 3 - Integrações	
3.1 Slack básico	Notificar via slack a divisão de equipas e apresentar os respetivos membros de cada uma
	Criar para cada subequipa um canal específico com os respetivos membros
	Arquivar os canais após efetuar a retrospetiva
3.2 Slack avançado	Efetuar operações sob os cartões de um quadro através do slack
	Copiar cartões de um quadro para o outro
	Publicar itens de ação no slack quando a retrospetiva termina
	Comandos de administrador (/)
	Enviar lembretes quando necessário
3.3 Outras integrações	Para ser discutido
Fase 4 - Agendamentos	
	Criar calendário interativo onde são apresentados os agendamentos do dia selecionado
	Listagem de agendamentos separados por dia
	Criar agendamentos definindo data e hora, recorrência e lembretes de antecedência
	Editar agendamentos
	Apagar agendamentos
Fase 5 - Personalização	
	Ligação entre cartões e itens de ação
	Definir tags para os cartões
	Providenciar templates para os quadros e cartões
	Exportar quadros

	Auto agrupamento de cartões utilizando processamento de linguagem natural
	Modificar a cor dos cartões nas colunas
	Gestão e o rastreamento dos itens de ação gerados em cada retrospetiva
	Possibilitar a revisão dos anteriores itens de ação no começo de cada retrospetiva se for gerada por agendamento
Backlog	
	Mostrar estatísticas de retrospetivas passadas
	Criar um sistema de *gaming* que promova a utilização da plataforma, cumprindo desafios e ganhando pontos
	Providenciar rápidos "ice breaker" no início de cada retrospetiva
	Compreender automaticamente a disponibilidade dos utilizadores para agendar uma retrospetiva
	Passar automaticamente os itens de ação não resolvidos para a nova retrospetiva
	Definir, controlar e visualizar nos quadros as fases para uma retrospetiva
	Permitir os utilizadores em demonstrar interesse em determinados itens de ação
	Providenciar um ajudante virtual que indique os próximos passos que o utilizador deve tomar

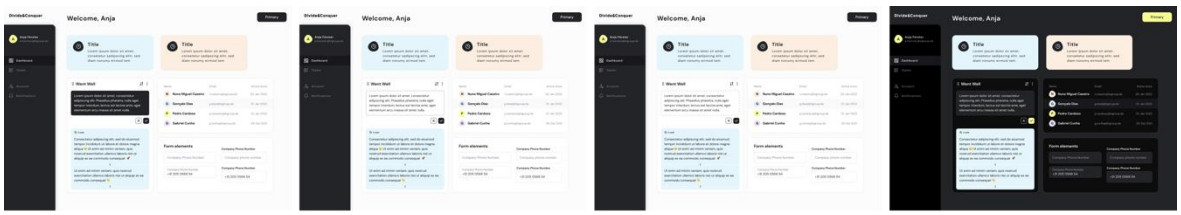
Apêndice J – Story map

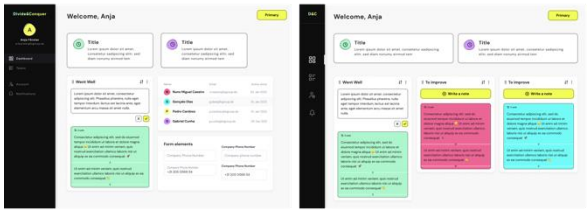


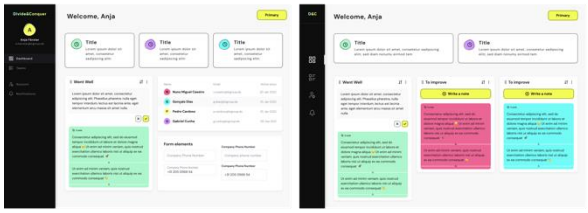
Apêndice K – Sitemap

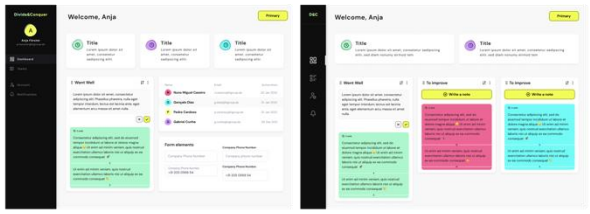


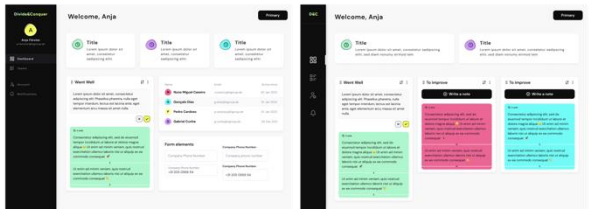
Apêndice L – Explorações de interface











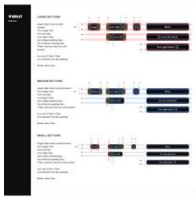
Apêndice M – Design system

Colors

Boxes

Typography

Button rules



Buttones

Form rules

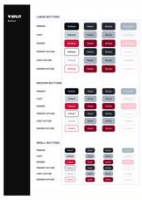
Inputs

TextArea

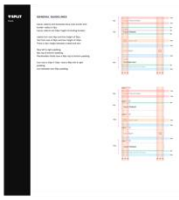
Select

Checkboxes/Radios/Switches

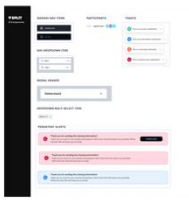
Icones



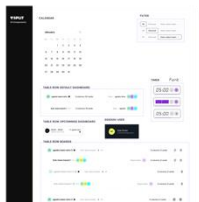
Tooltips/Popovers



Toasts/Alerts



Calendar/Filter/Raws/NavBar/Cards/Tiles

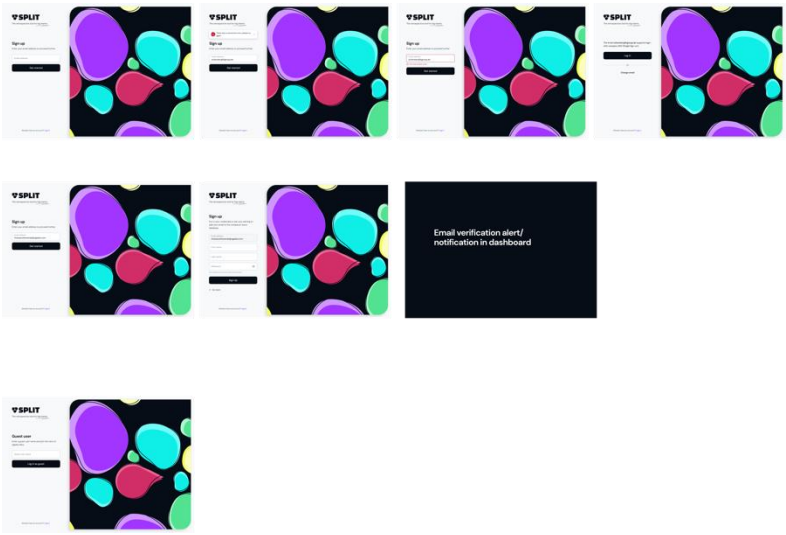


Columns

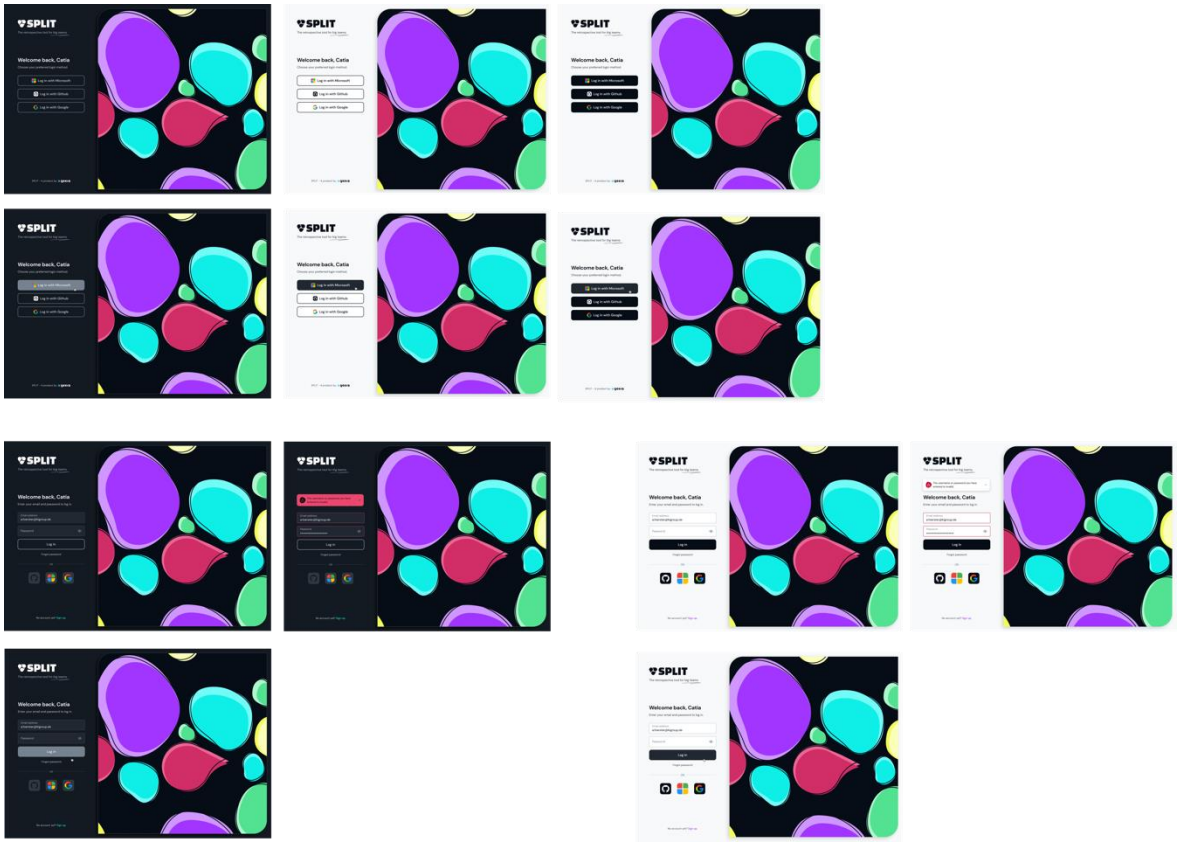
Misc

Apêndice N – Interface para desktop

Sign in

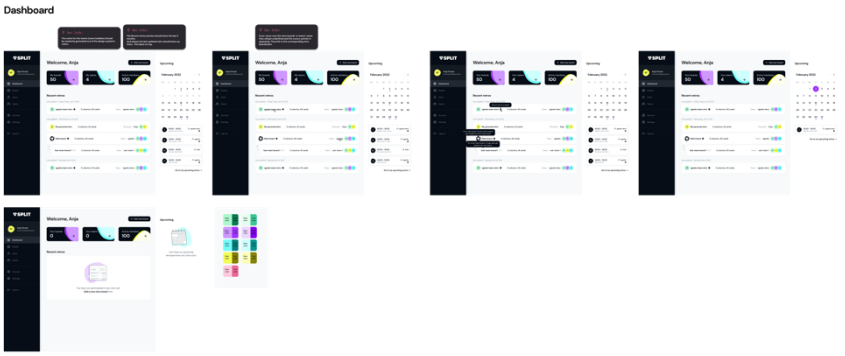


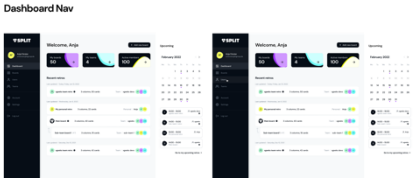
Sign up



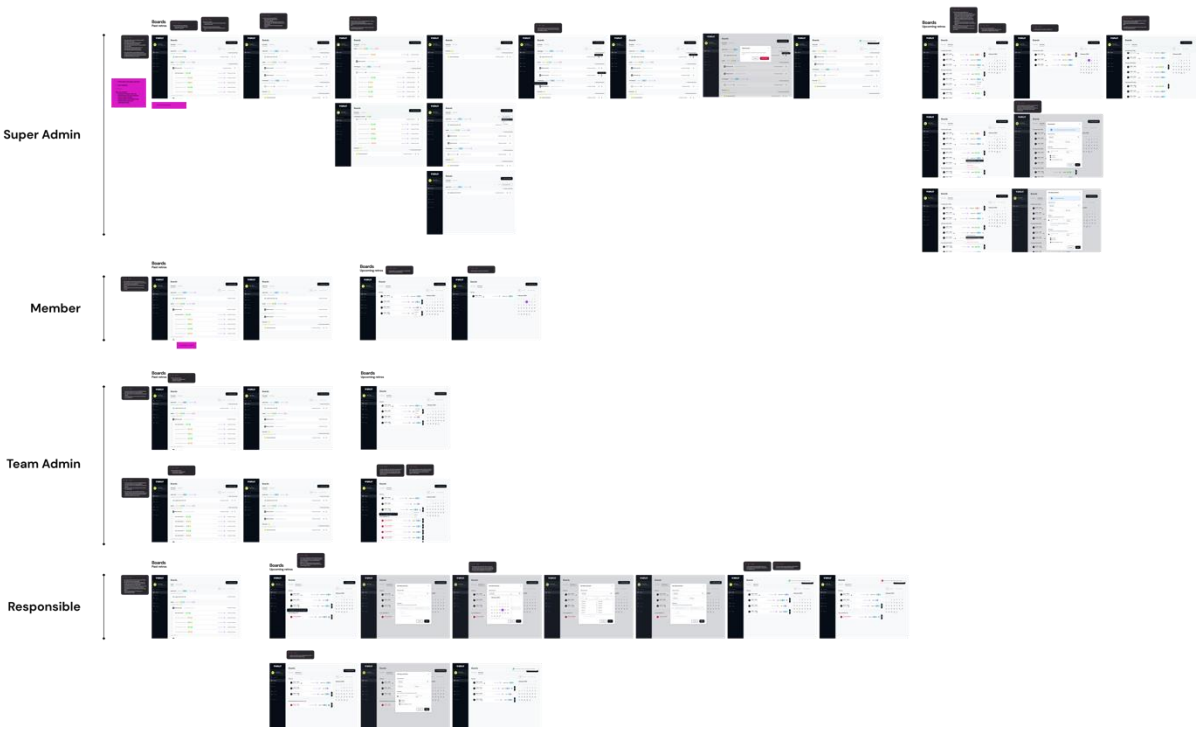
Dashboard

Member
Super Admin
Team Admin
Board Admin



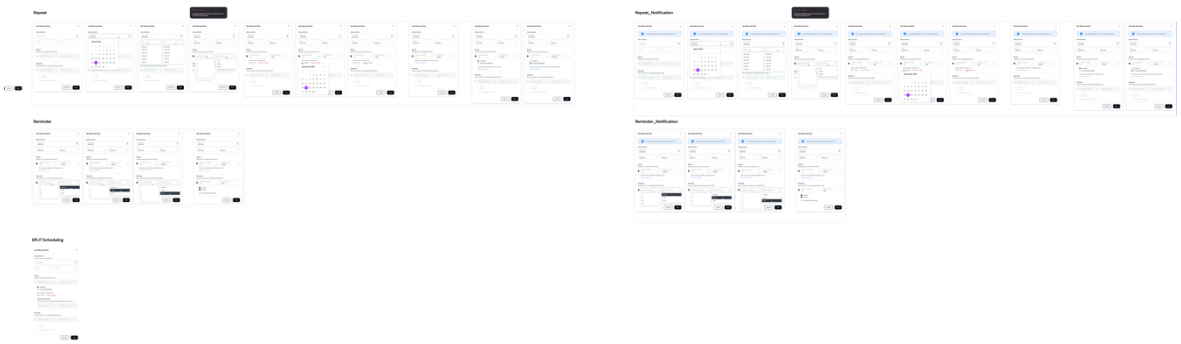


Boards page



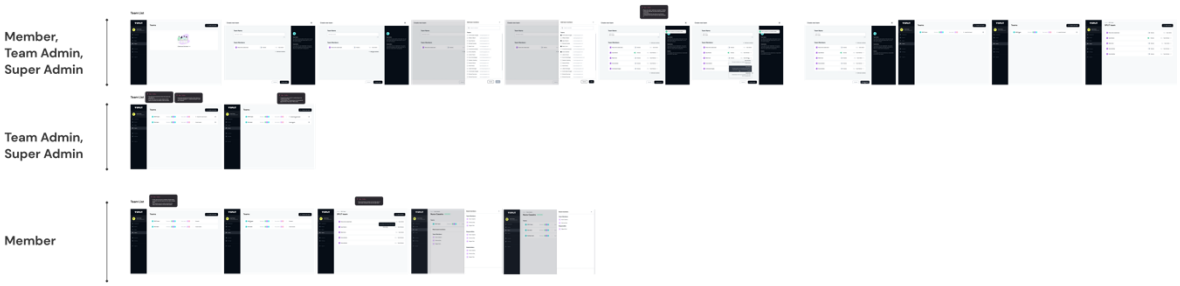
Recover password

Scheduling components



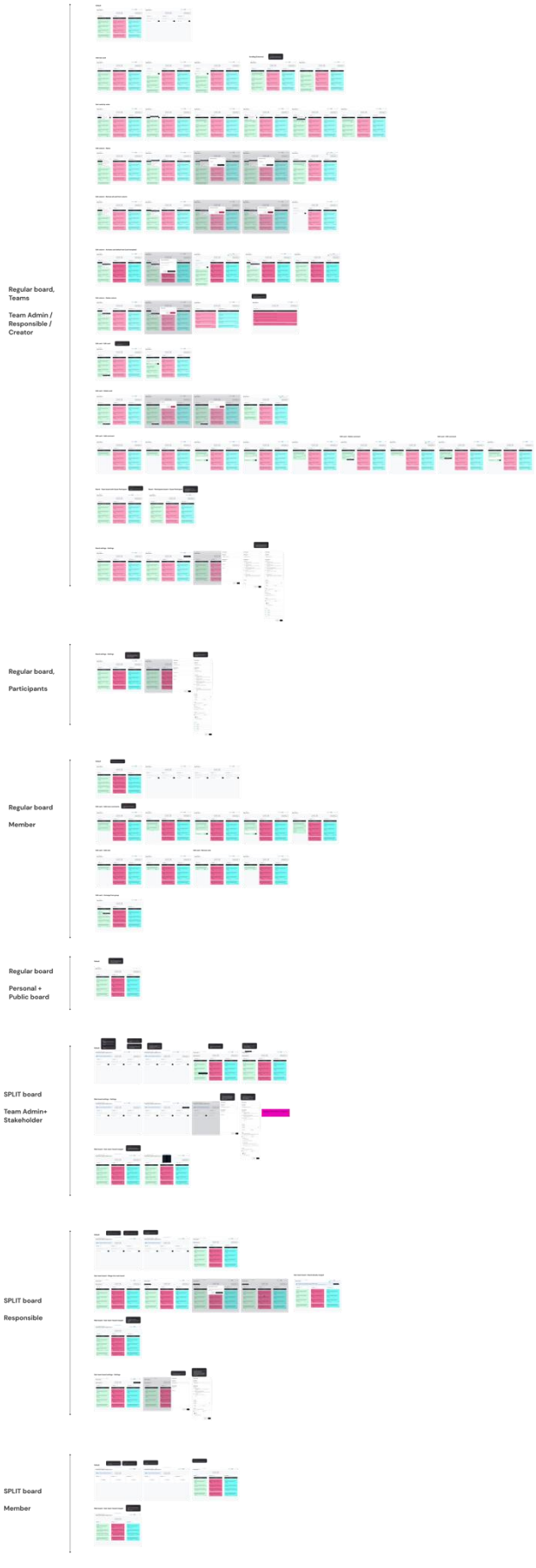
User management

Team management



Create new board

Board


Regular board,
Teams
Team Admin /
Responsible /
Creator
Regular board,
Participants
Regular board
Member
Regular board
Personal +
Public board
SPLIT board
Team Admin+
Stakeholder
SPLIT board
Responsible
SPLIT board
Member

Apêndice O – Dockerfiles

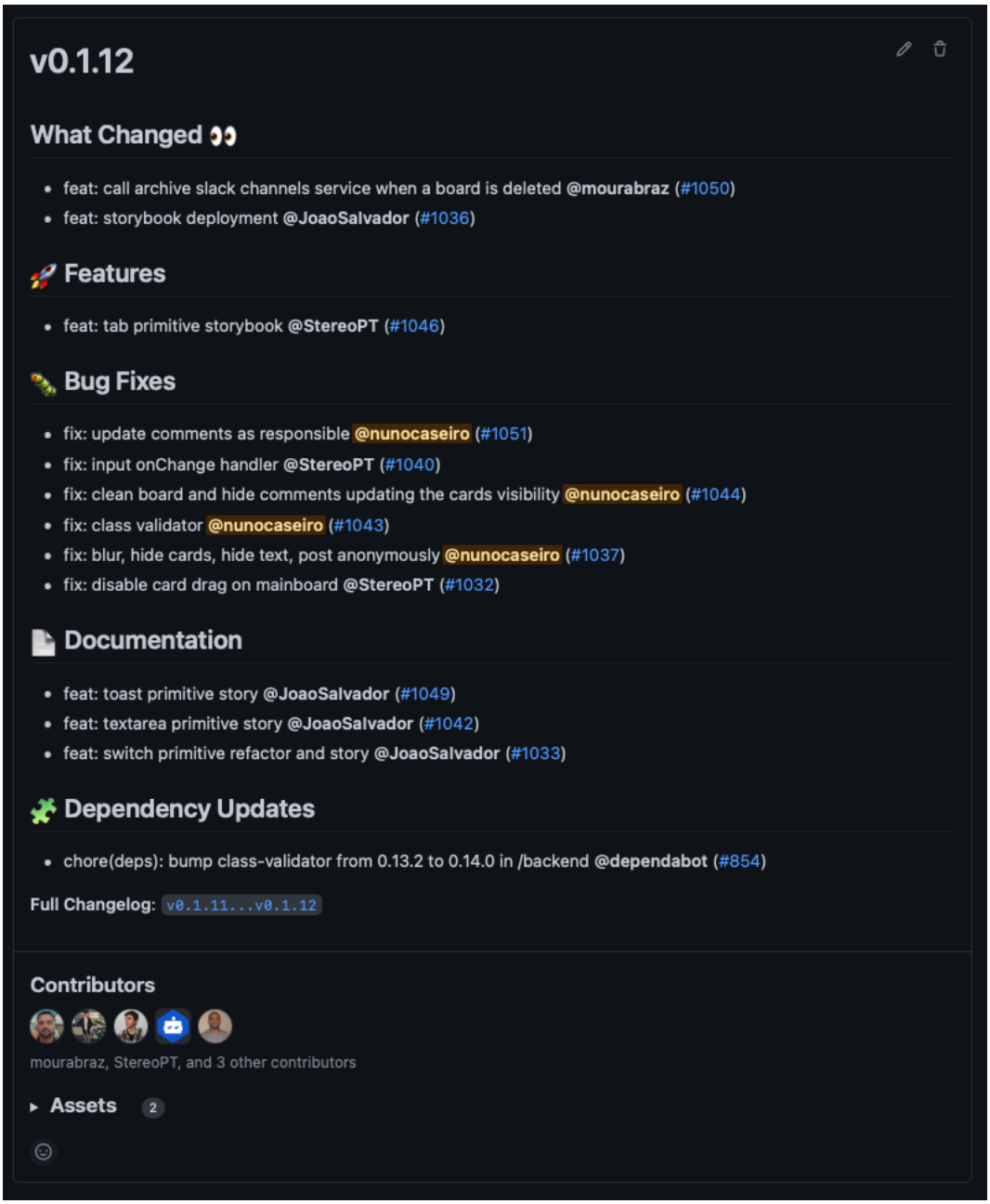
Frontend

```
ARG TAG=18.12.1-alpine

# Install dependencies only when needed
FROM node:$TAG AS development
# Check https://github.com/nodejs/docker-node/tree/b4117f9333da4138b03a546ec926ef50a31506c3#nodealpine
RUN apk add --no-cache libc6-compat
WORKDIR /app
COPY ./package*.json ./

RUN npm ci

COPY . .

# Rebuild the source code only when needed
FROM node:$TAG AS builder
ENV NODE_ENV=production
WORKDIR /app
COPY --from=development /app/node_modules ./node_modules
COPY . .

RUN npm run build
RUN npm run build-storybook
RUN npm prune --omit=dev

# Production image, copy all the files and run next
FROM node:$TAG AS production
WORKDIR /app

RUN addgroup -g 1001 -S nodejs
RUN adduser -S nextjs -u 1001

# You only need to copy next.config.js if you are NOT using the default configuration
COPY --from=builder --chown=nextjs:nodejs /app/.next ./.next

COPY --from=builder --chown=nextjs:nodejs /app/next.config.js ./
COPY --from=builder --chown=nextjs:nodejs /app/public ./public
COPY --from=builder --chown=nextjs:nodejs /app/package.json ./package.json
COPY --from=builder --chown=nextjs:nodejs /app/cypress.json ./
COPY --from=builder /app/node_modules ./node_modules

USER nextjs

EXPOSE 3000
ENV PORT 3000

CMD npm run start
```

Backend

```
ARG TAG=18.12.1-alpine

###################
# BUILD FOR LOCAL DEVELOPMENT
###################

FROM node:$TAG As development

# Create app directory
WORKDIR /app

# Copy application dependency manifests to the container image.
# A wildcard is used to ensure copying both package.json AND package-lock.json (when available).
# Copying this first prevents re-running yarn install on every code change.
COPY package*.json ./

RUN npm ci

# Bundle app source
COPY . .

###################
# BUILD FOR PRODUCTION
###################

FROM node:$TAG As build

WORKDIR /app

COPY package*.json ./

# In order to run `npm run build` we need access to the Nest CLI which is a dev dependency.
#In the previous development stage we ran `yarn` which installed all dependencies, so we can copy o
COPY --from=development /app/node_modules ./node_modules

COPY . .

# Run the build command which creates the production bundle
RUN npm run build

# Set NODE_ENV environment variable
ENV NODE_ENV production

# Running `npm` removes the existing node_modules directory and passing in --omit=dev ensures that
RUN npm prune --omit=dev

###################
# PRODUCTION
###################

FROM node:$TAG As production

# Copy the bundled code from the build stage to the production image
COPY --from=build /app/node_modules ./node_modules
COPY --from=build /app/dist ./dist
COPY --from=build /app/package.json ./package.json

EXPOSE 3200
ENV PORT 3200

# Start the server using the production build
CMD npm run start:prod
```

Apêndice P – Exemplo de change log (v0.1.12)



v0.1.12

What Changed 👀

- feat: call archive slack channels service when a board is deleted @mourabraz (#1050)
- feat: storybook deployment @JoaoSalvador (#1036)

🚀 Features

- feat: tab primitive storybook @StereoPT (#1046)

🍾 Bug Fixes

- fix: update comments as responsible @nunocaseiro (#1051)
- fix: input onChange handler @StereoPT (#1040)
- fix: clean board and hide comments updating the cards visibility @nunocaseiro (#1044)
- fix: class validator @nunocaseiro (#1043)
- fix: blur, hide cards, hide text, post anonymously @nunocaseiro (#1037)
- fix: disable card drag on mainboard @StereoPT (#1032)

Documentation

- feat: toast primitive story @JoaoSalvador (#1049)
- feat: textarea primitive story @JoaoSalvador (#1042)
- feat: switch primitive refactor and story @JoaoSalvador (#1033)

🧩 Dependency Updates

- chore(deps): bump class-validator from 0.13.2 to 0.14.0 in /backend @dependabot (#854)

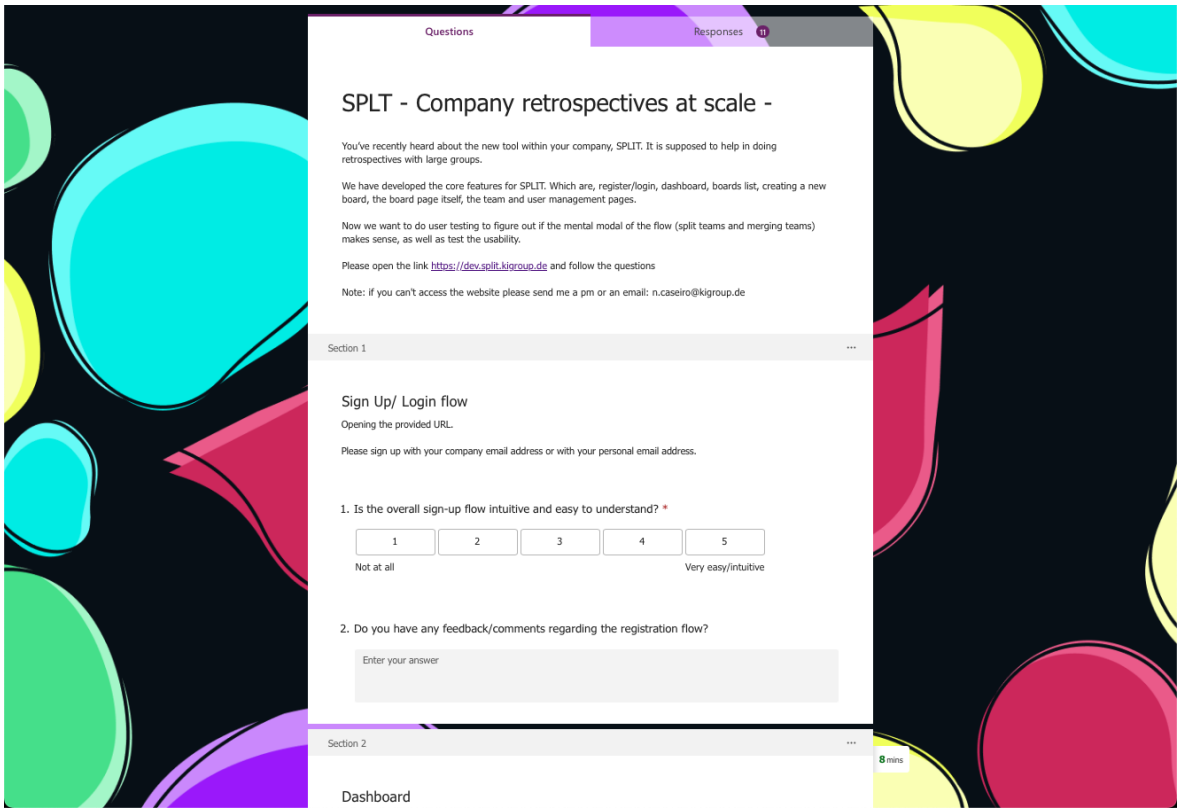
Full Changelog: `v0.1.11...v0.1.12`

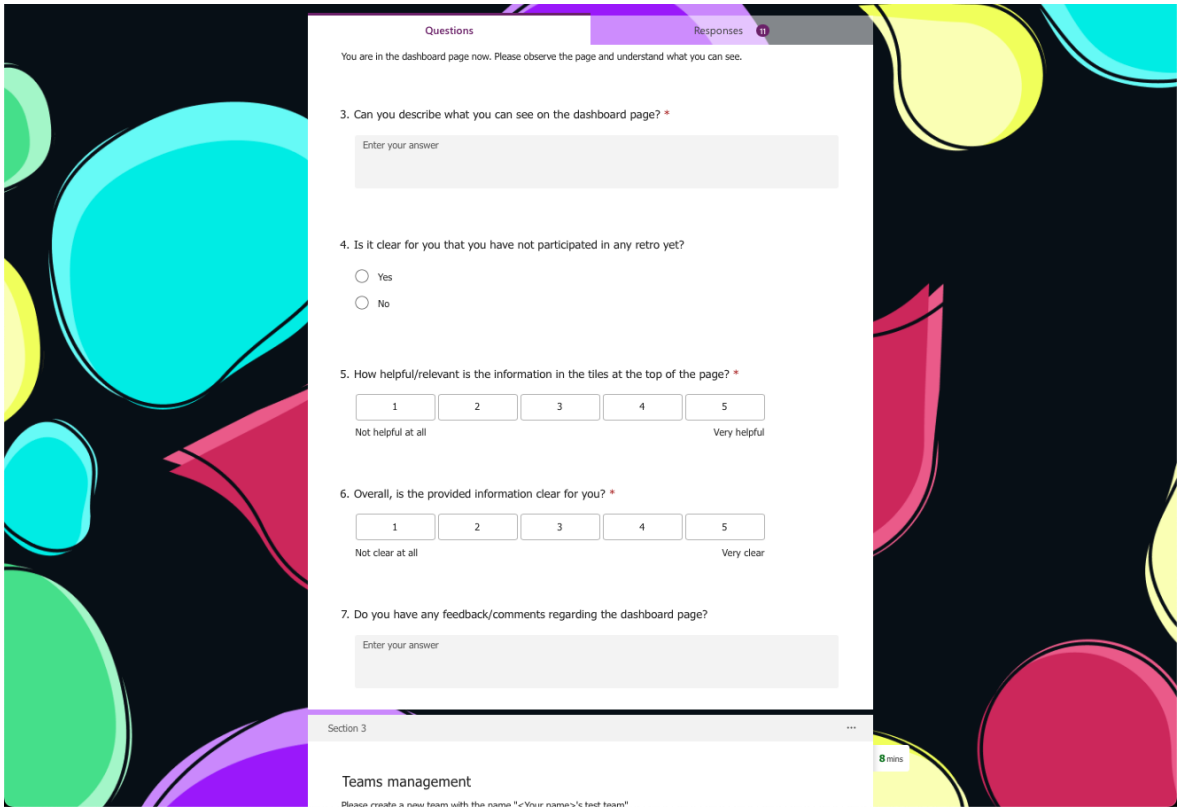
Contributors

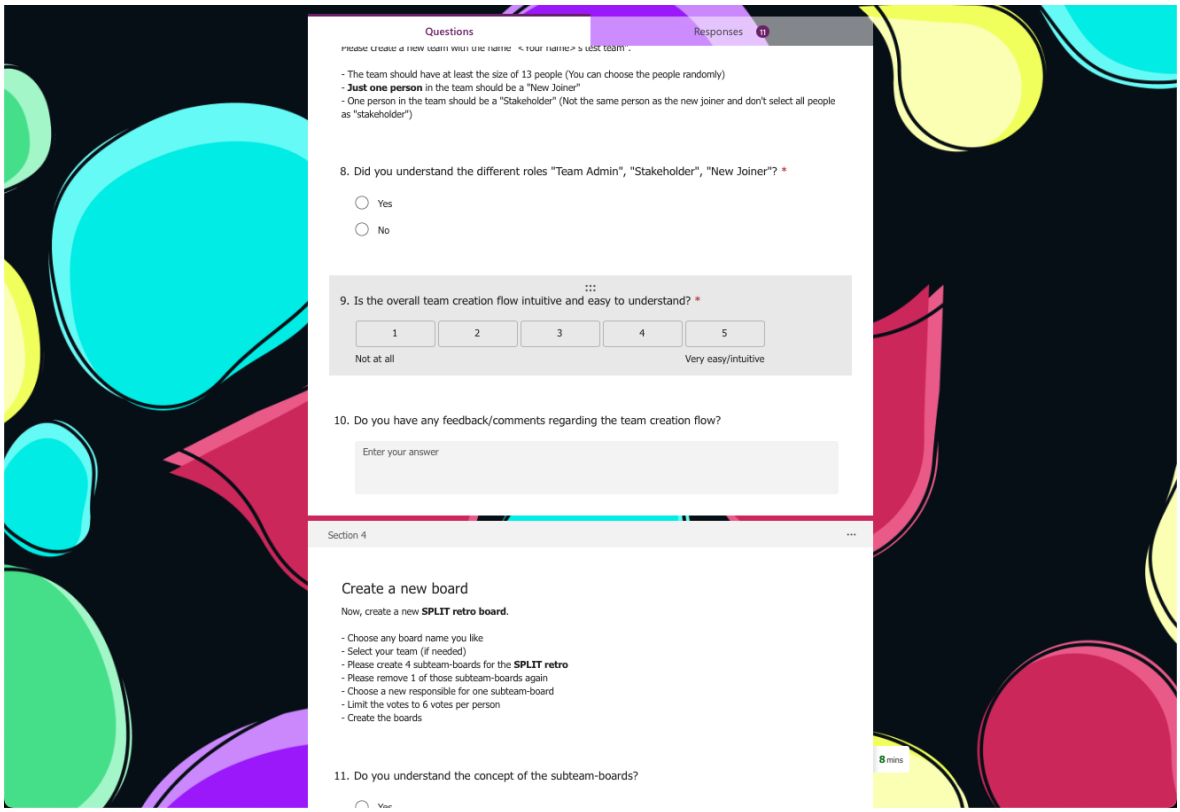
mourabraz, StereoPT, and 3 other contributors

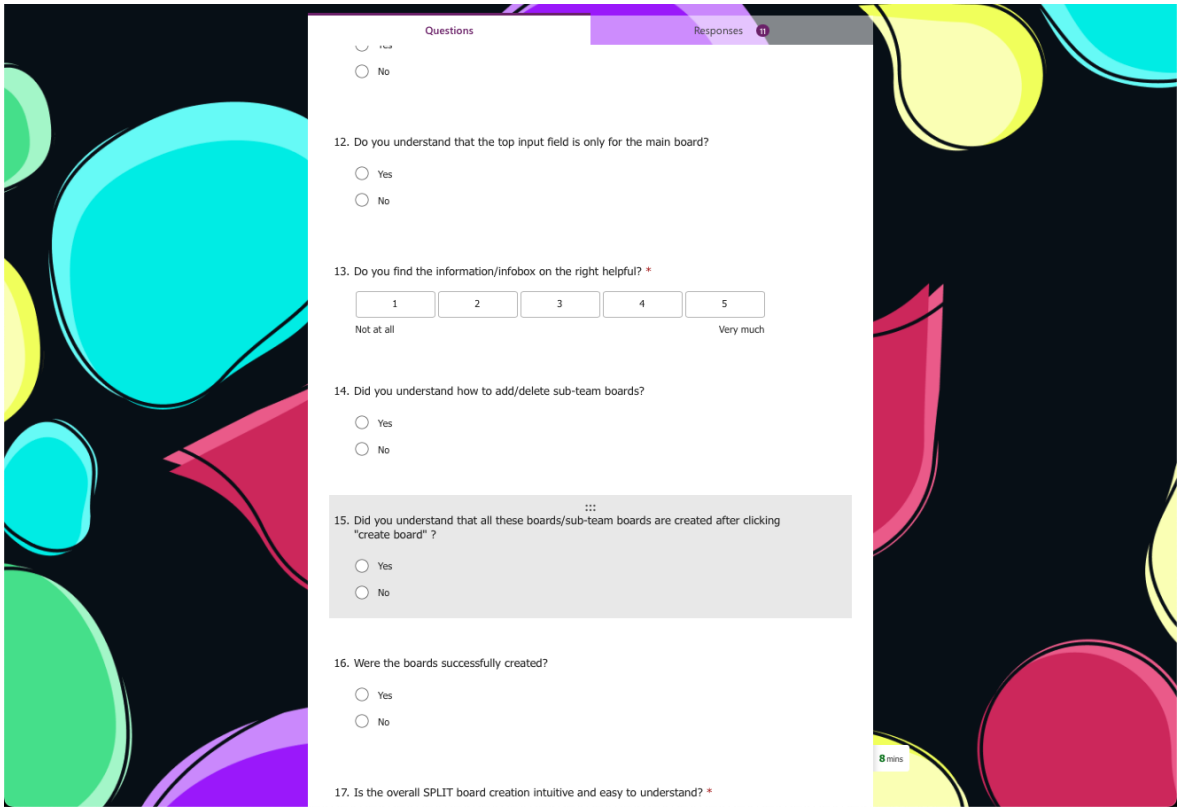
▸ **Assets** 2

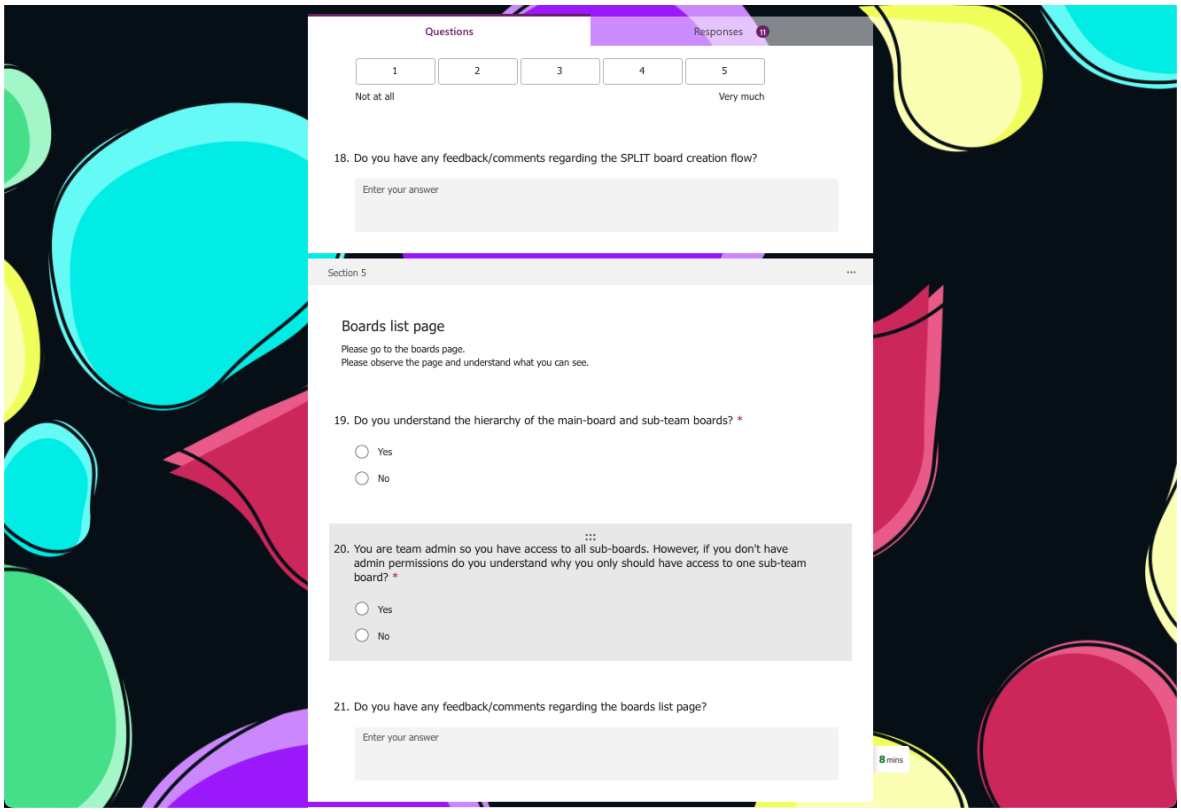
Apêndice Q – Formulário para testes de usabilidade

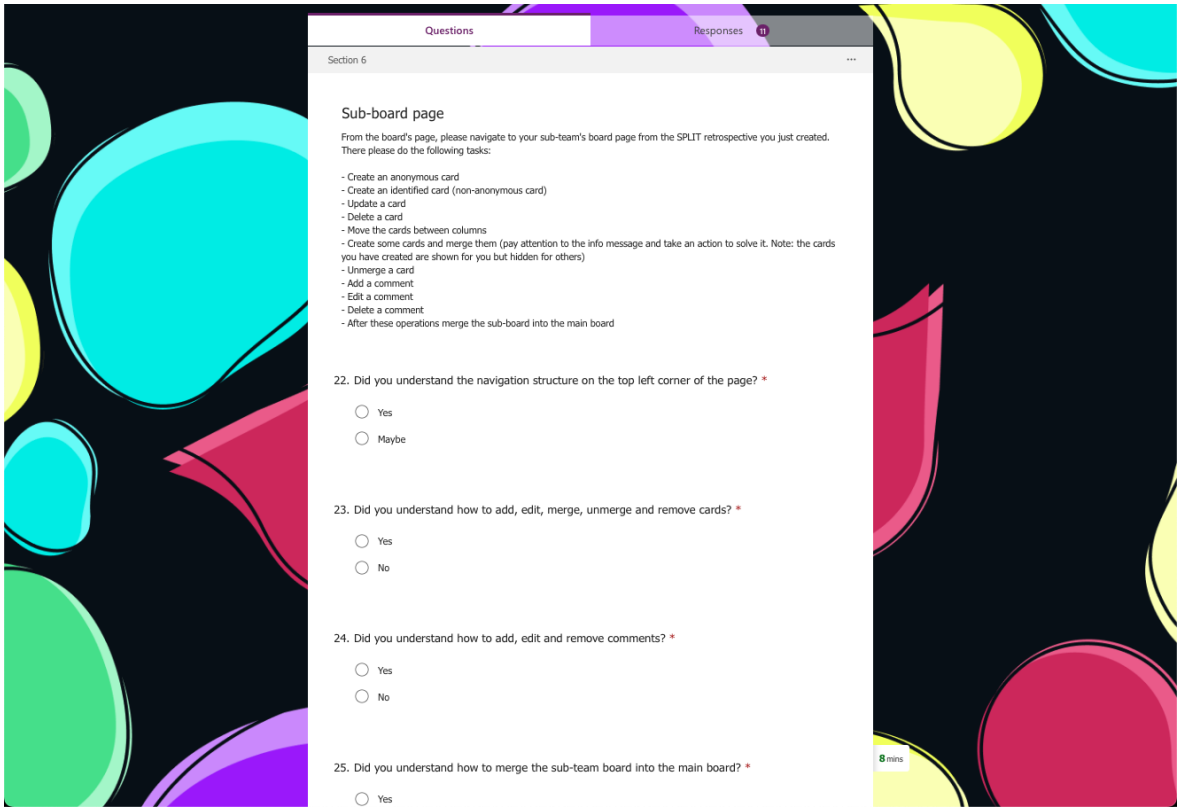


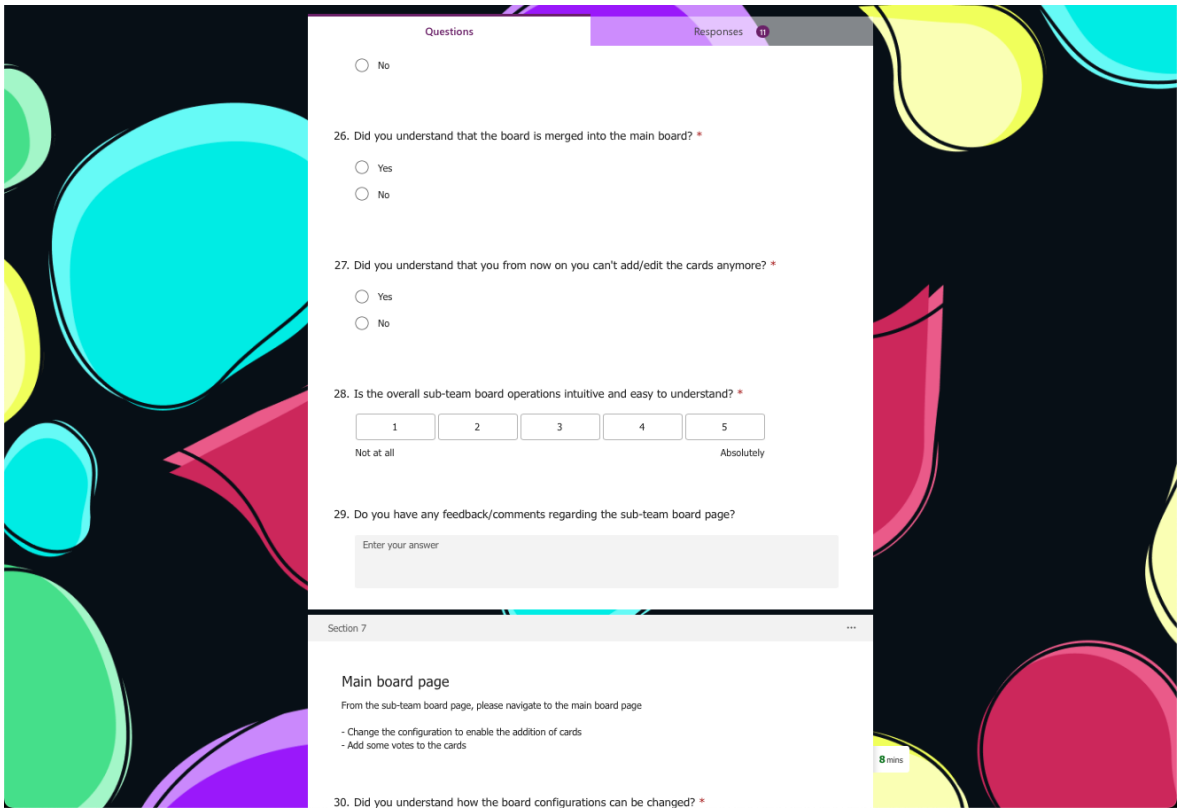


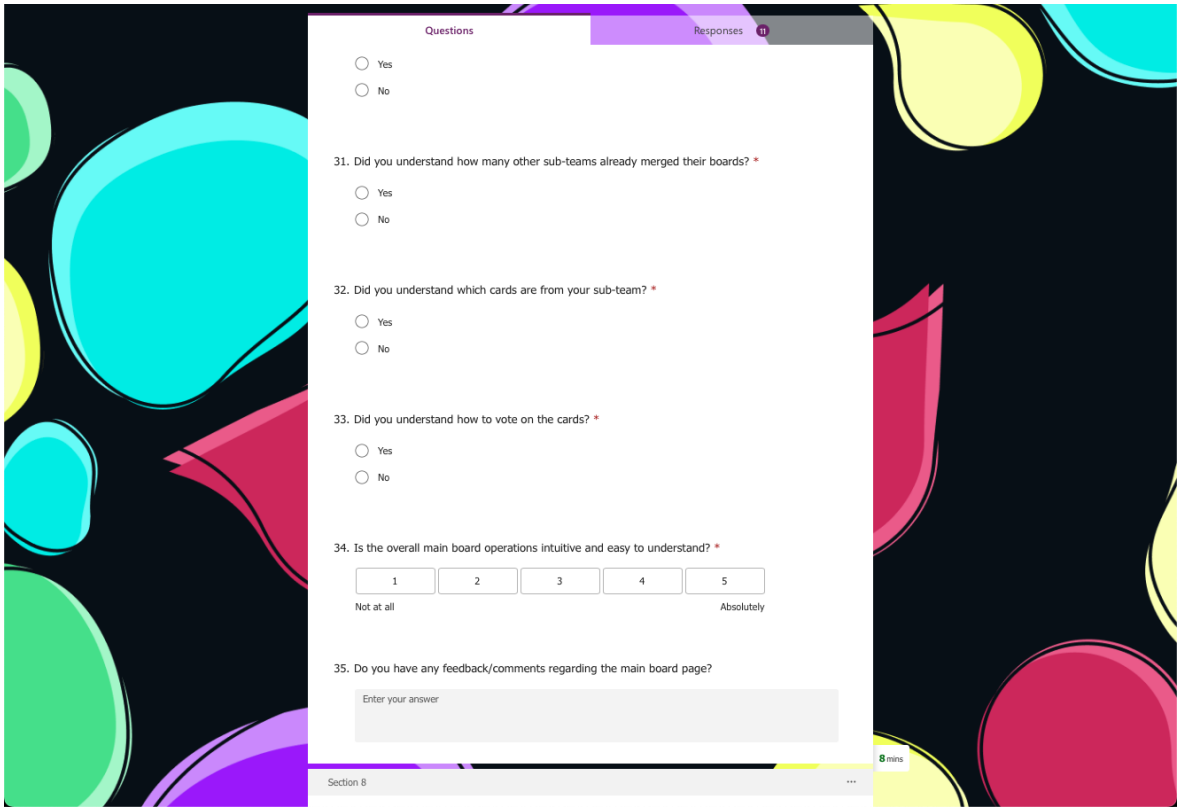

Questions
Responses 11
- The team should have at least the size of 13 people (You can choose the people randomly)
- Just one person in the team should be a "New Joiner"
- One person in the team should be a "Stakeholder" (Not the same person as the new joiner and don't select all people as "stakeholder")
8. Did you understand the different roles "Team Admin", "Stakeholder", "New Joiner"? *
Yes
No
9. Is the overall team creation flow intuitive and easy to understand? *
1 2 3 4 5
Not at all
Very easy/intuitive
10. Do you have any feedback/comments regarding the team creation flow?
Enter your answer
Section 4
Create a new board
Now, create a new SPLIT retro board.
- Choose any board name you like
- Select your team (if needed)
- Please create 4 subteam-boards for the SPLIT retro
- Please remove 1 of those subteam-boards again
- Choose a new responsible for one subteam-board
- Limit the votes to 6 votes per person
- Create the boards
11. Do you understand the concept of the subteam-boards?
8 mins

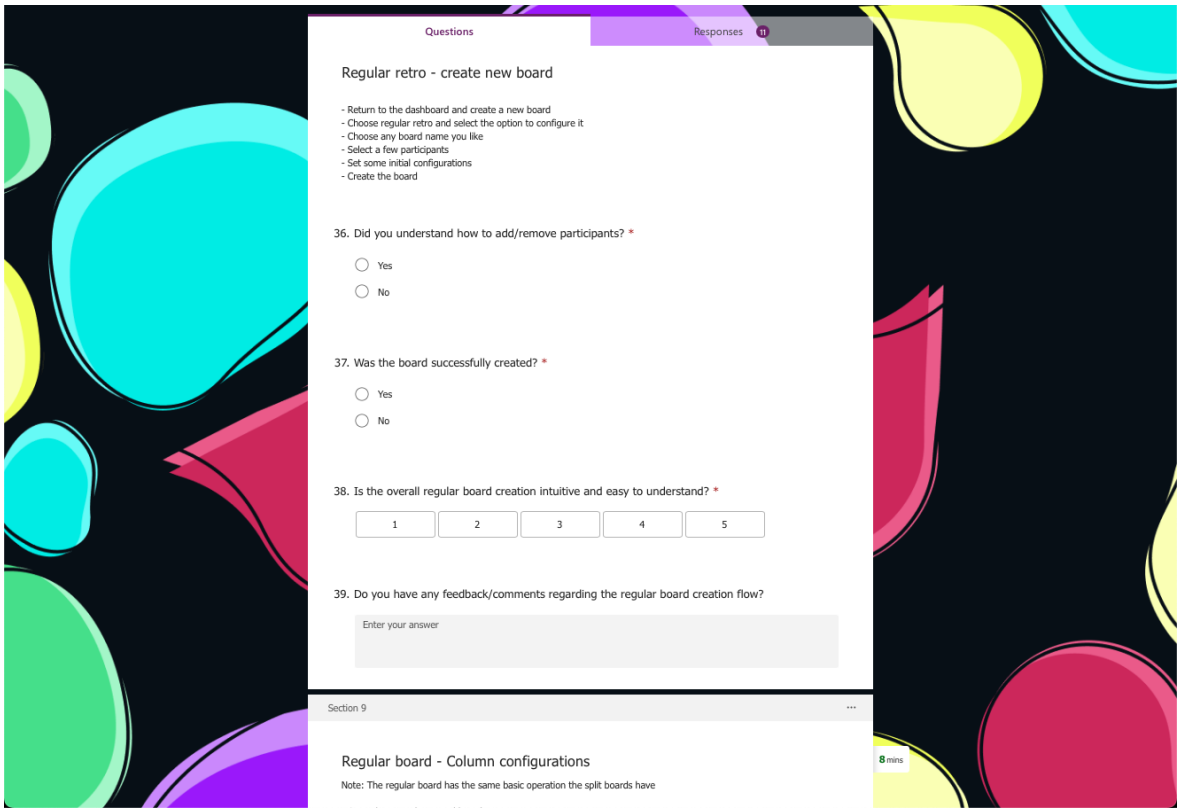

Questions
Responses 11
No
12. Do you understand that the top input field is only for the main board?
Yes
No
13. Do you find the information/infobox on the right helpful? *
1 2 3 4 5
Not at all
Very much
14. Did you understand how to add/delete sub-team boards?
Yes
No
15. Did you understand that all these boards/sub-team boards are created after clicking "create board" ?
Yes
No
16. Were the boards successfully created?
Yes
No
17. Is the overall SPLIT board creation intuitive and easy to understand? *
8 mins

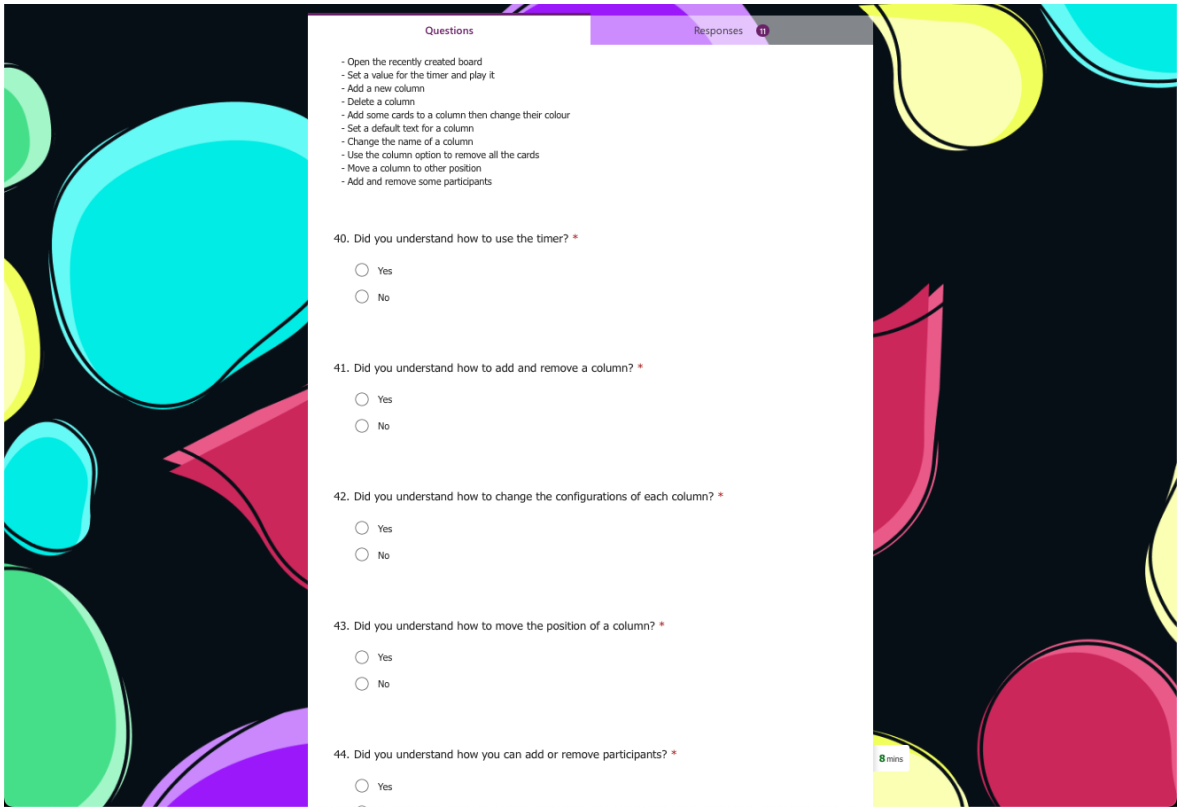

Questions
Responses 11
1
2
3
4
5
Not at all
Very much
18. Do you have any feedback/comments regarding the SPLIT board creation flow?
Enter your answer
Section 5
Boards list page
Please go to the boards page.
Please observe the page and understand what you can see.
19. Do you understand the hierarchy of the main-board and sub-team boards? *
Yes
No
20. You are team admin so you have access to all sub-boards. However, if you don't have admin permissions do you understand why you only should have access to one sub-team board? *
Yes
No
21. Do you have any feedback/comments regarding the boards list page?
Enter your answer
8 mins


Questions
Responses 11
Section 6
Sub-board page
From the board's page, please navigate to your sub-team's board page from the SPLIT retrospective you just created.
There please do the following tasks:
- Create an anonymous card
- Create an identified card (non-anonymous card)
- Update a card
- Delete a card
- Move the cards between columns
- Create some cards and merge them (pay attention to the info message and take an action to solve it. Note: the cards you have created are shown for you but hidden for others)
- Unmerge a card
- Add a comment
- Edit a comment
- Delete a comment
- After these operations merge the sub-board into the main board
22. Did you understand the navigation structure on the top left corner of the page? *
Yes
Maybe
23. Did you understand how to add, edit, merge, unmerge and remove cards? *
Yes
No
24. Did you understand how to add, edit and remove comments? *
Yes
No
25. Did you understand how to merge the sub-team board into the main board? *
Yes
8 mins


Questions
Responses 11
No
26. Did you understand that the board is merged into the main board? *
Yes
No
27. Did you understand that you from now on you can't add/edit the cards anymore? *
Yes
No
28. Is the overall sub-team board operations intuitive and easy to understand? *
1
2
3
4
5
Not at all
Absolutely
29. Do you have any feedback/comments regarding the sub-team board page?
Enter your answer
Section 7
Main board page
From the sub-team board page, please navigate to the main board page
- Change the configuration to enable the addition of cards
- Add some votes to the cards
8 mins
30. Did you understand how the board configurations can be changed? *


Questions
Responses 11
Yes
No
31. Did you understand how many other sub-teams already merged their boards? *
Yes
No
32. Did you understand which cards are from your sub-team? *
Yes
No
33. Did you understand how to vote on the cards? *
Yes
No
34. Is the overall main board operations intuitive and easy to understand? *
1
2
3
4
5
Not at all
Absolutely
35. Do you have any feedback/comments regarding the main board page?
Enter your answer
8 mins
Section 8


Questions
Responses 11
Regular retro - create new board
- Return to the dashboard and create a new board
- Choose regular retro and select the option to configure it
- Choose any board name you like
- Select a few participants
- Set some initial configurations
- Create the board
36. Did you understand how to add/remove participants? *
Yes
No
37. Was the board successfully created? *
Yes
No
38. Is the overall regular board creation intuitive and easy to understand? *
1
2
3
4
5
39. Do you have any feedback/comments regarding the regular board creation flow?
Enter your answer
Section 9
Regular board - Column configurations
Note: The regular board has the same basic operation the split boards have
8 mins


Questions
Responses 11
- Open the recently created board
- Set a value for the timer and play it
- Add a new column
- Delete a column
- Add some cards to a column then change their colour
- Set a default text for a column
- Change the name of a column
- Use the column option to remove all the cards
- Move a column to other position
- Add and remove some participants
40. Did you understand how to use the timer? *
Yes
No
41. Did you understand how to add and remove a column? *
Yes
No
42. Did you understand how to change the configurations of each column? *
Yes
No
43. Did you understand how to move the position of a column? *
Yes
No
44. Did you understand how you can add or remove participants? *
Yes
8 mins

No

45. Are the overall columns operations intuitive and easy to understand in a regular board? *

1	2	3	4	5

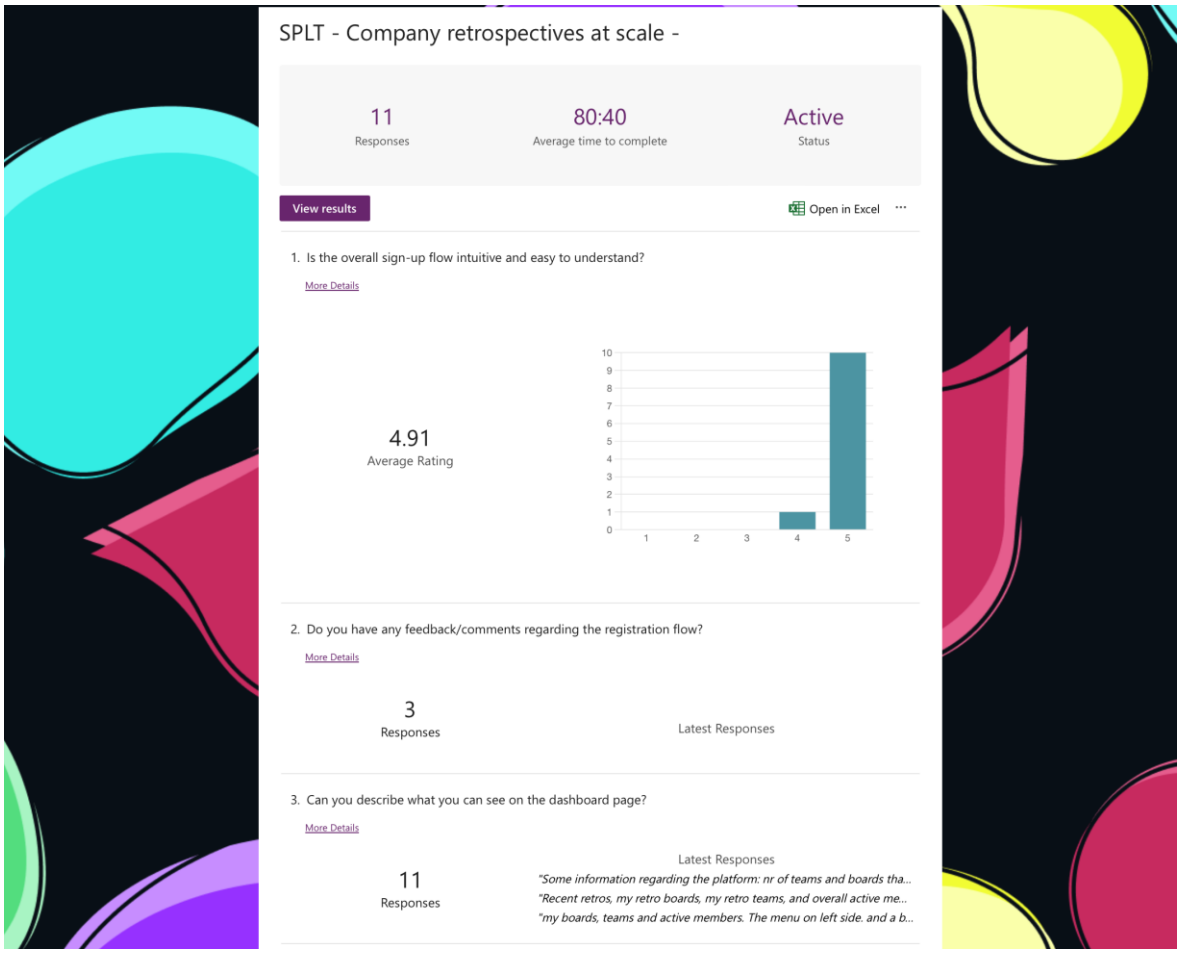
46. Is the overall regular board operations intuitive and easy to understand? *

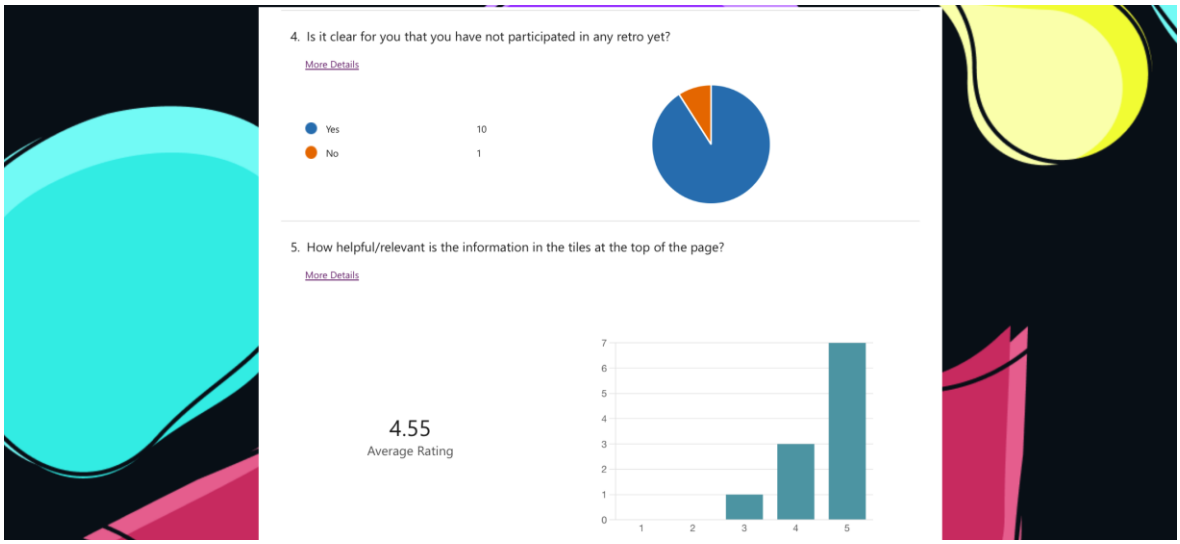
1	2	3	4	5

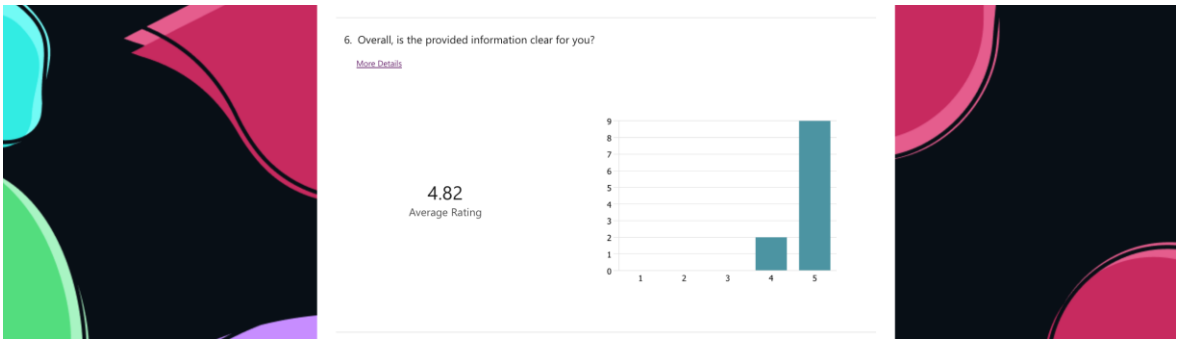
47. Do you have any feedback/comments regarding the regular board and the columns configurations?

Enter your answer

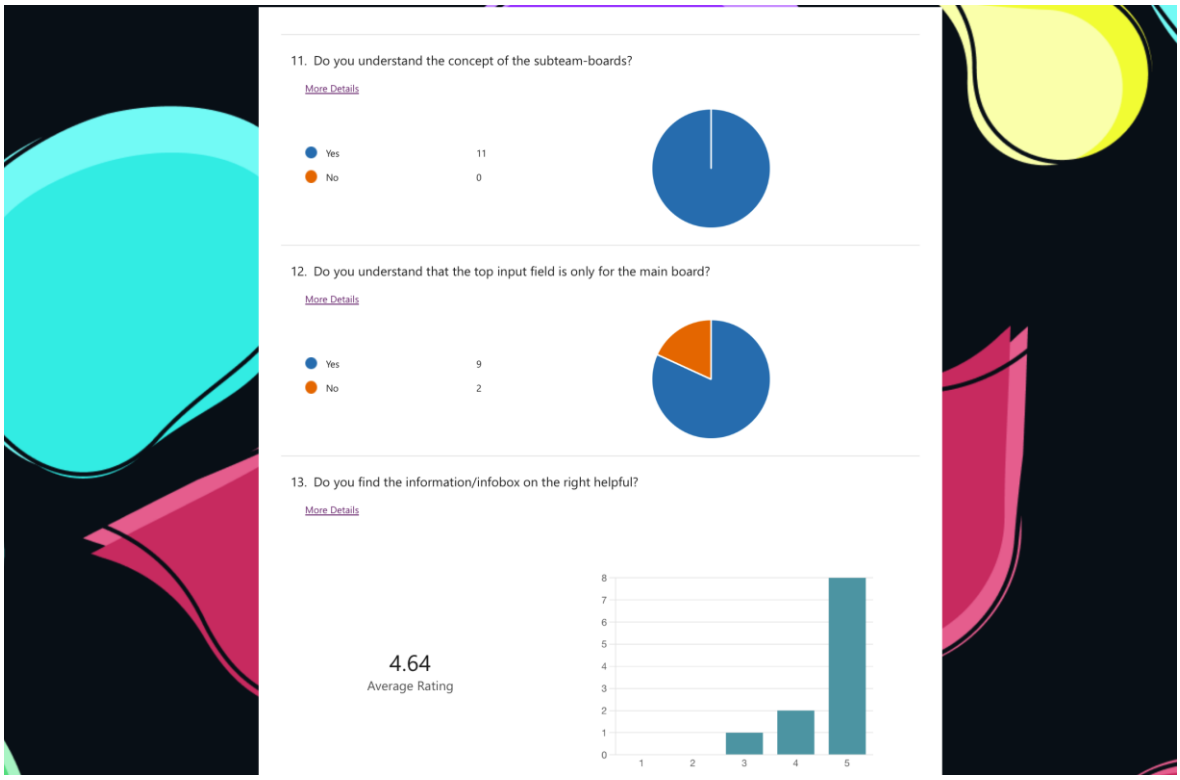
Apêndice R – Resultados dos testes de usabilidade

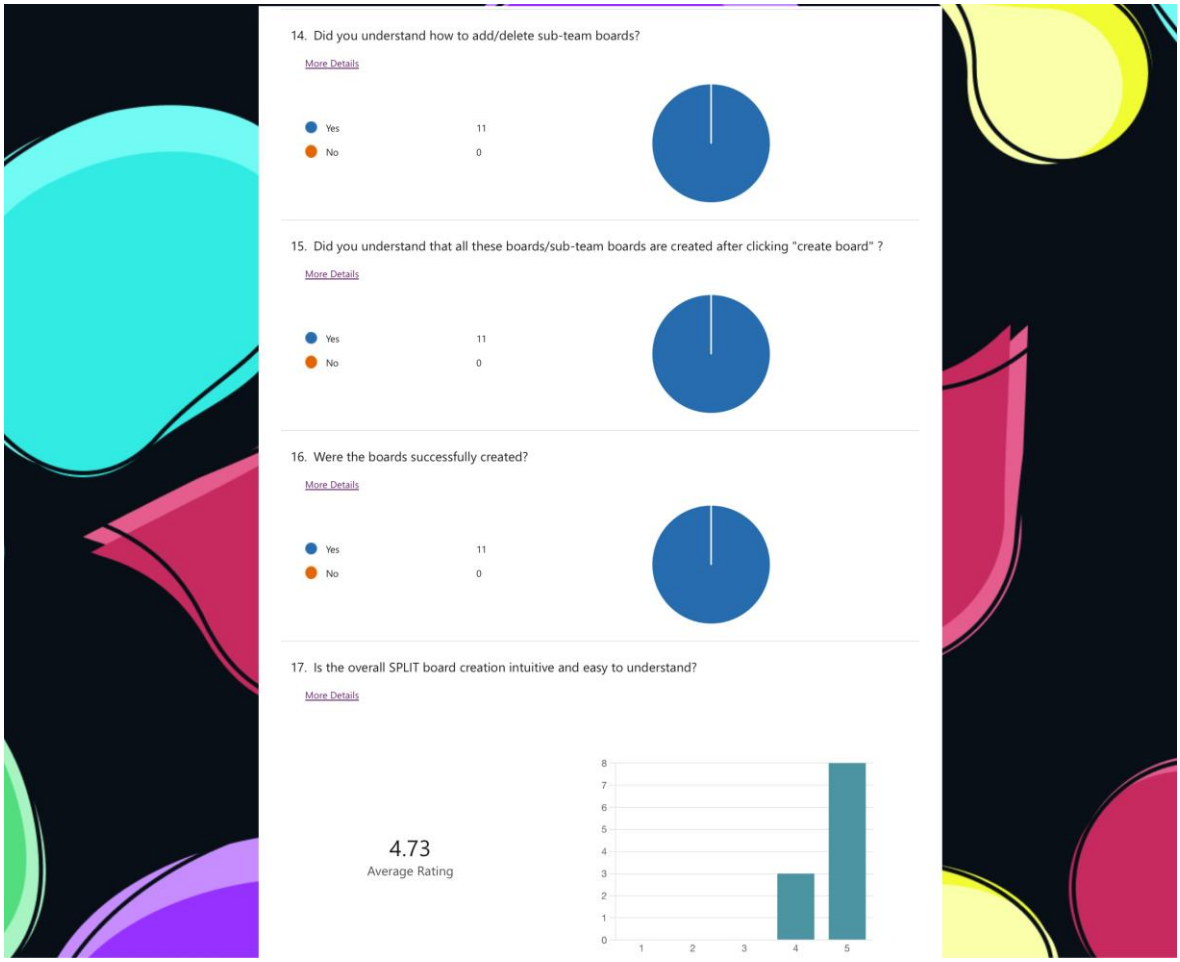


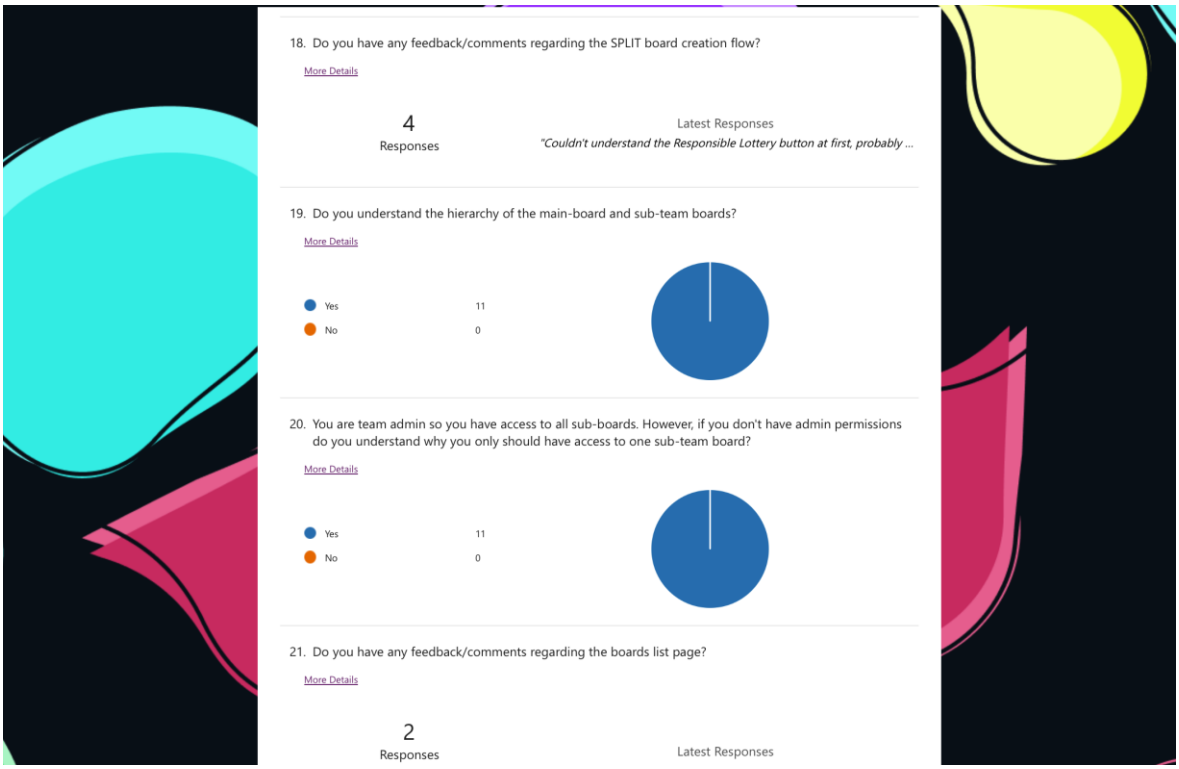


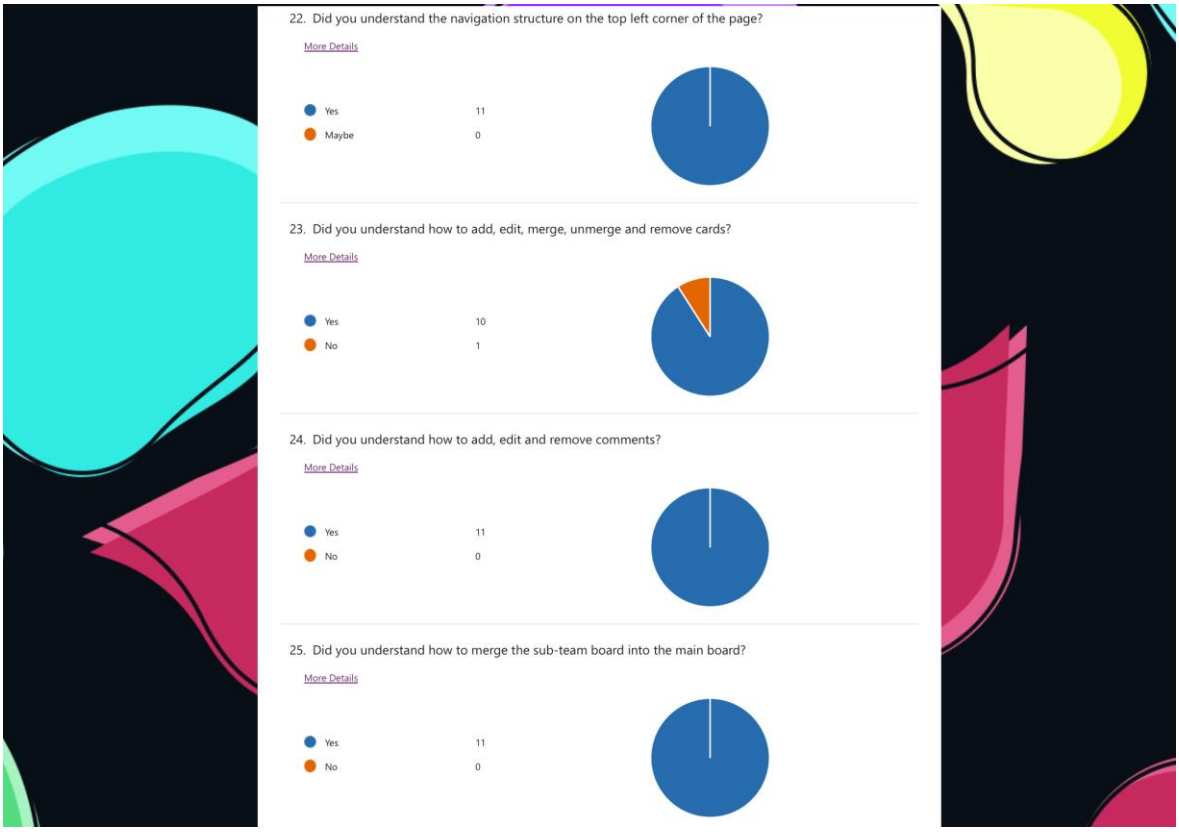

6. Overall, is the provided information clear for you?
More Details
4.82
Average Rating
9
8
7
6
5
4
3
2
1
0
1
2
3
4
5

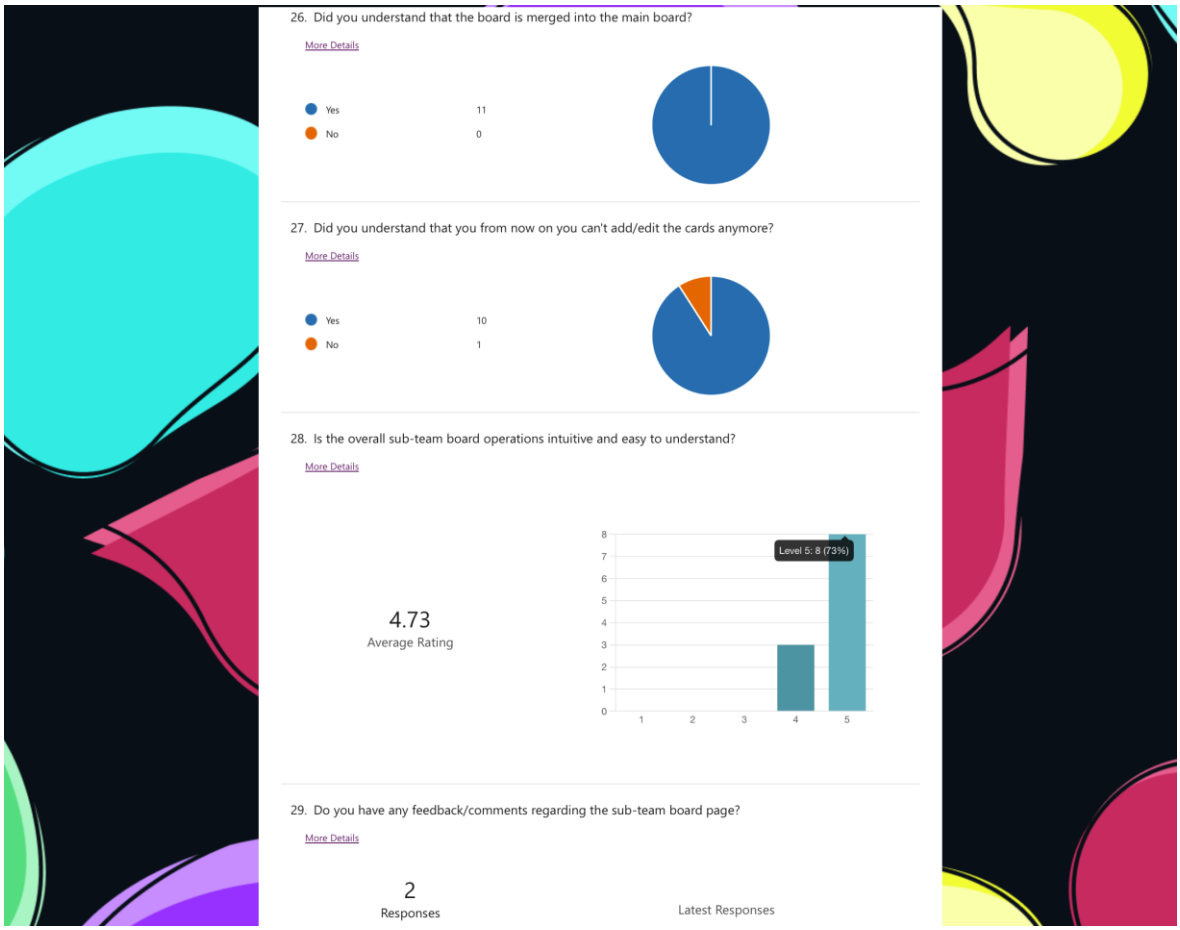

7. Do you have any feedback/comments regarding the dashboard page?
More Details
4
Responses
Latest Responses
8. Did you understand the different roles "Team Admin", "Stakeholder", "New Joiner"?
More Details
Yes
10
No
1
9. Is the overall team creation flow intuitive and easy to understand?
More Details
4.64
Average Rating
8
7
6
5
4
3
2
1
0
1
2
3
4
5
10. Do you have any feedback/comments regarding the team creation flow?
More Details
3
Responses
Latest Responses
"Sidebar describing roles (esp. new joiner) is on the right side instead o...

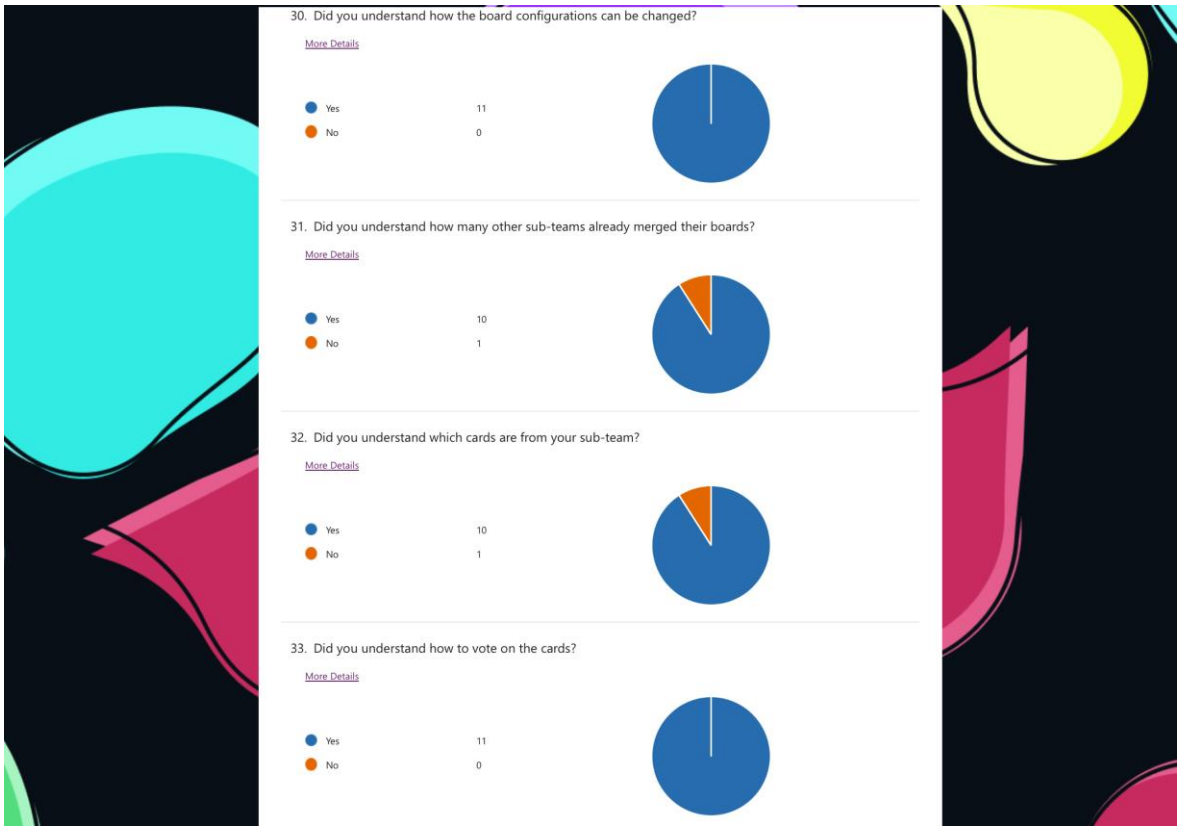

11. Do you understand the concept of the subteam-boards?
More Details
Yes 11
No 0
12. Do you understand that the top input field is only for the main board?
More Details
Yes 9
No 2
13. Do you find the information/infobox on the right helpful?
More Details
4.64
Average Rating
0 1 2 3 4 5 6 7 8
1 2 3 4 5


14. Did you understand how to add/delete sub-team boards?
More Details
Yes 11
No 0
15. Did you understand that all these boards/sub-team boards are created after clicking "create board" ?
More Details
Yes 11
No 0
16. Were the boards successfully created?
More Details
Yes 11
No 0
17. Is the overall SPLIT board creation intuitive and easy to understand?
More Details
4.73
Average Rating
0 1 2 3 4 5 6 7 8
1 2 3 4 5


18. Do you have any feedback/comments regarding the SPLIT board creation flow?
More Details
4
Responses
Latest Responses
"Couldn't understand the Responsible Lottery button at first, probably ...
19. Do you understand the hierarchy of the main-board and sub-team boards?
More Details
Yes 11
No 0
20. You are team admin so you have access to all sub-boards. However, if you don't have admin permissions do you understand why you only should have access to one sub-team board?
More Details
Yes 11
No 0
21. Do you have any feedback/comments regarding the boards list page?
More Details
2
Responses
Latest Responses


22. Did you understand the navigation structure on the top left corner of the page?
More Details
Yes 11
Maybe 0
23. Did you understand how to add, edit, merge, unmerge and remove cards?
More Details
Yes 10
No 1
24. Did you understand how to add, edit and remove comments?
More Details
Yes 11
No 0
25. Did you understand how to merge the sub-team board into the main board?
More Details
Yes 11
No 0

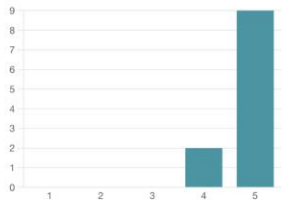

26. Did you understand that the board is merged into the main board?
More Details
Yes 11
No 0
27. Did you understand that you from now on you can't add/edit the cards anymore?
More Details
Yes 10
No 1
28. Is the overall sub-team board operations intuitive and easy to understand?
More Details
4.73
Average Rating
Level 5: 8 (73%)
29. Do you have any feedback/comments regarding the sub-team board page?
More Details
2
Responses
Latest Responses


30. Did you understand how the board configurations can be changed?
More Details
Yes 11
No 0
31. Did you understand how many other sub-teams already merged their boards?
More Details
Yes 10
No 1
32. Did you understand which cards are from your sub-team?
More Details
Yes 10
No 1
33. Did you understand how to vote on the cards?
More Details
Yes 11
No 0

34. Is the overall main board operations intuitive and easy to understand?

More Details

4.82
Average Rating



35. Do you have any feedback/comments regarding the main board page?

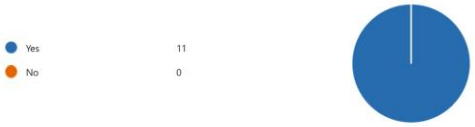
More Details

5
Responses

Latest Responses
"Took me a while to find place where it says how many other sub-team...

36. Did you understand how to add/remove participants?

More Details



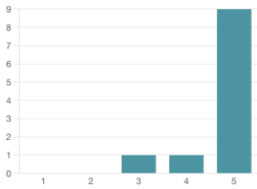
37. Was the board successfully created?

More Details

38. Is the overall regular board creation intuitive and easy to understand?

More Details

4.73
Average Rating



39. Do you have any feedback/comments regarding the regular board creation flow?

More Details

2
Responses

Latest Responses

40. Did you understand how to use the timer?

More Details

41. Did you understand how to add and remove a column?

More Details

42. Did you understand how to change the configurations of each column?

More Details

Yes	10
No	1

43. Did you understand how to move the position of a column?

More Details

Yes	11
No	0

44. Did you understand how you can add or remove participants?

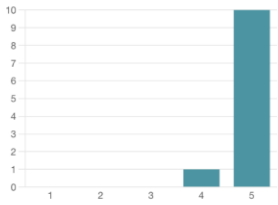
More Details

Yes	9
No	1

45. Are the overall columns operations intuitive and easy to understand in a regular board?

More Details

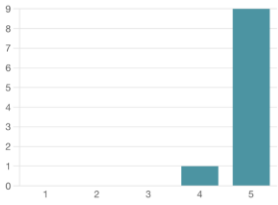
4.91
Average Rating



46. Is the overall regular board operations intuitive and easy to understand?

More Details

4.90
Average Rating



47. Do you have any feedback/comments regarding the regular board and the columns configurations?

More Details

2
Responses

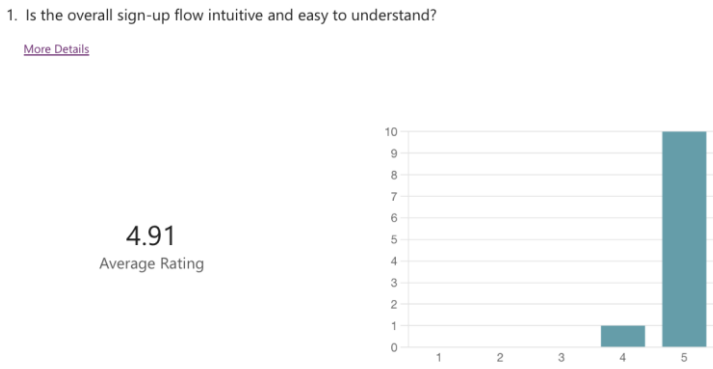
Latest Responses

Apêndice S – Análise completa dos resultados

Nesta secção do documento pretende-se apresentar e discutir todas as respostas obtidas no formulário. É de realçar que os participantes não tiveram qualquer contacto prévio com o sistema até à realização dos testes, o que possibilita, com fiabilidade, definir se a interface disponibilizada é fidedigna, intuitiva e fácil de utilizar.

No que diz respeito aos dispositivos utilizados para realização dos testes, todos os utilizadores usaram um computador com um *browser* instalado.

A secção 1 é constituída por duas questões e esta diz respeito ao registo dos utilizadores na plataforma. Na descrição da secção é pedido que se efetue o registo na plataforma com o email da empresa ou outro pessoal. Visualizando o resultado da primeira questão, observável na próxima figura é possível definir que o processo de registo é intuitivo e fácil de compreender, na medida em que a maioria o classificou com o valor 5.



A secção termina com a questão para introduzir *feedback* relativamente ao processo de registo e, como tal, verificam-se três respostas, as quais se podem observar na próxima figura. Nesta pode-se visualizar que um utilizador indica que pretende diferentes SSO *providers*, outro indica que a utilização de SSO é uma mais-valia na medida em que facilita o processo de registo. A última resposta indica algumas oportunidades de melhoria, nomeadamente no que diz respeito ao *feedback* que é dado ao utilizador enquanto efetua o seu registo, é dito que não proporciona a melhor experiência de utilização. Importa referir que a disponibilização de SSO *providers* depende da empresa que aloje e disponibilize o SPLIT. No que diz respeito ao *feedback* do último utilizador, este foi tido em conta e originou um novo *issue* a ser discutido pela equipa.

2. Do you have any feedback/comments regarding the registration flow?

3 Responses

ID ↑	Name	Responses
1	anonymous	Allow for different SSO providers
2	anonymous	It's very cool that we can sign-up or login with sso, it makes all the process easier and quicker.
3	anonymous	- Sign up button could be closer to the login form - After inputing my email and clicking Get Started it takes a while to move to the next step without any feedback to the user - Password field shows error feedback as soon as you start typing which is a bit distracting - After signing up, it shows the login form again before moving to the dashboard

A secção dois é constituída por 5 questões e na descrição é solicitado que se observe atentamente para a página do *dashboard*. Na questão número três do formulário é pedido que se descreva a página observada e a maioria consegue fazê-lo adequadamente, como é possível observar na próxima figura. É indicado que se observa a *sidebar*, algumas estatísticas com hiperligações para navegação, um botão para criar quadros e também as últimas retrospetivas em que o utilizador participou. Assim sendo, consegue-se afirmar que os elementos constituintes da página são facilmente identificados, tal como a sua utilidade.

3. Can you describe what you can see on the dashboard page?

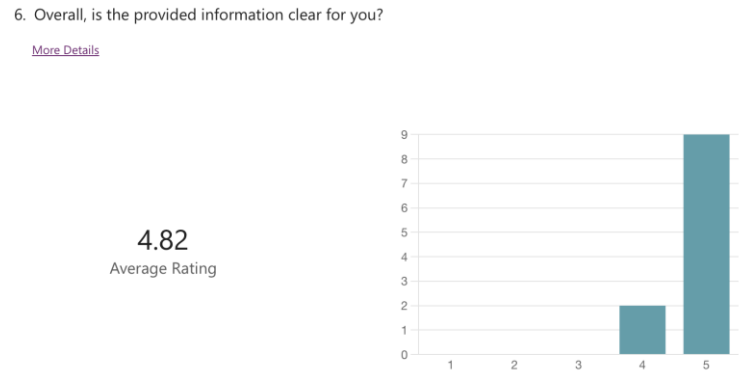
11 Responses

ID ↑	Name	Responses
1	anonymous	My boards; my teams; how many users are registered on split
2	anonymous	sidebar with options, statistics on top (active members is an hyperlink to the members list) with a "Add new board" button above them, recent retros with my sub-team and main boards as clickable links
3	anonymous	In the dashboard, I have a resume with the number of my boards, teams where I blong and active members. I also can see the latest retros where I participated.
4	anonymous	My boards, teams, active members, recent retros, and links to other pages
5	anonymous	On the left a menu, with logical separations and in the middle an easy way to find my boards, teams and active members.
6	anonymous	3 cards: Your boards, Your teams, Active members My Recent retros Add a new retro board call to action
7	anonymous	I can see 3 cards: boards, teams, and active members. I also see My current retros below (i don't have any now)
8	anonymous	I can see the boards that I am in, my teams, and all members. For each item of these items, I can navigate to see more details about each one. I also can see the recent retros that I participated. On the left side, I have some menu items, the same described before and account and settings menu items.
9	anonymous	my boards, teams and active members. The menu on left side. and a button to add a new board. Also, I see a list of recent retros but it's empty because I didn't use it
10	anonymous	Recent retros, my retro boards, my retro teams, and overall active members in SPLIT
11	anonymous	Some information regarding the platform: nr of teams and boards that I'm participating in, the nr of users in the platform

A questão número quatro pretende verificar se os utilizadores compreendem que após a criação da sua conta, estes ainda não participaram em qualquer retrospetiva. Analisando os resultados, pode-se verificar que à exceção de um utilizador, todos os outros compreendem através da interface que ainda não participaram numa retrospetiva.

A questão 5 pretende compreender se os utilizadores consideram que as estatísticas no topo da página são úteis e relevantes. Visualizando os resultados, pode-se assumir que a maioria considera como relevante, uma vez que o valor quatro e cinco são aqueles que a grande parte escolheu, contudo é importante refletir sobre este tópico para compreender se a informação apresentada fornece real utilidade ao utilizador.

Com o intuito de compreender de maneira geral se a informação apresentada no *dashboard* é clara, desenvolveu-se a pergunta número seis que apresenta resultados muito positivos, na medida em que a maioria escolheu o valor 5 e apenas dois participantes selecionaram o valor 4, como se pode observar na próxima figura, podendo assumir-se que a interface da página é bem conseguida.



Os comentários adicionados pelos participantes na questão número 7 são bastante gratificantes na medida em que reforçam o que é verificado na questão anterior, como se pode observar na próxima figura.

7. Do you have any feedback/comments regarding the dashboard page?

4 Responses

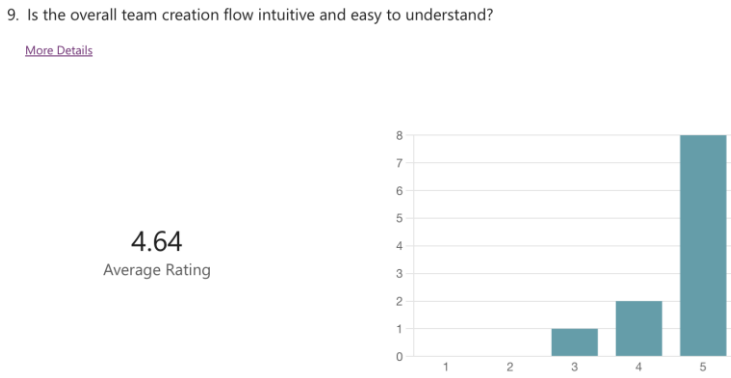
ID ↑	Name	Responses
1	anonymous	I don't find the active members information relevant and it is not clear how that information can be helpful to me in this context.
2	anonymous	just what was described previously, allow different SSO providers
3	anonymous	No, it's PERFECT
4	anonymous	the page structure is very nice and clean

A maioria das respostas são elogios à interface, uma das respostas não se enquadra com a questão efetuada e a última refere que a informação relativa aos membros ativos da plataforma não é relevante, e não é claro como esta pode ser útil, o que reforça a necessidade de repensar como otimizar a secção das estatísticas no *dashboard.*

Na secção três e constituída por três questões, é solicitado que se efetuem algumas operações na página de gestão de equipas, nomeadamente a criação de uma equipa e atribuição de papéis aos elementos constituintes.

A questão 8 pretende compreender se o participante entendeu os papéis que um utilizador pode ter numa equipa e quais são as suas diferenças. As respostas foram positivas, na medida em que apenas um participante indicou que não compreendeu.

Na questão 9 pretende-se avaliar se os utilizadores consideram que o processo de criação de equipas é intuitivo e fácil de utilizar. Analisando os resultados, possíveis de observar na próxima figura, verifica-se que a maioria define como muito fácil e apenas um participante avaliou com o número três. Estes resultados pretendem traduzir que, de forma geral, o processo é bastante acessível de efetuar e que a interface é intuitiva.



Para finalizar a secção relacionada com a gestão de equipas é proposto aos participantes que deixem a sua opinião através de comentários. Como tal verificam-se três opiniões diferentes, como é possível observar na próxima figura. Nesta, é indicado pelo utilizador 1 que não é muito claro do que um administrador da equipa pode realizar e que não é claro que um *stakeholder* é também um administrador. O segundo utilizador sugere a alteração da *dropdown* para *checkboxes,* para possibilitar a alteração do papel dos utilizadores numa equipa. O último utilizador indica que o painel informativo poderia encontrar-se noutra posição tendo em conta que, em certas culturas, pode ser estranho o posicionamento à direita.

As opiniões descritas vão ao encontro dos resultados obtidos na pergunta anterior. Verifica-se que as opiniões não se focam nos aspetos fulcrais das operações de gestão de equipas, no entanto, é relevante melhorar a informação proporcionada sobre os papéis dos utilizadores numa equipa.

10. Do you have any feedback/comments regarding the team creation flow?

3 Responses

ID ↑	Name	Responses
1	anonymous	It's not very clear what a team admin can do that other can't It's not clear that a stakeholder is also a team admin
2	anonymous	to change the role was a bit strange, I would suggest to you change the dropdown to some checkbox group, with the aim of displaying the 3 possible roles, since just there are 3 options
3	anonymous	Sidebar describing roles (esp. new joiner) is on the right side instead of the left side, which is not intuitive for some cultures

A secção 4 diz respeito à página e ao processo de criação de uma retrospetiva dividida e é composta por 8 questões. Nesta, é solicitada que se efetue a criação de uma retrospetiva para uma equipa e que esta seja dividida em três subequipas, que sejam criados os quadros respetivos e que se alterem algumas configurações iniciais.

A questão 11 tenciona compreender se todos os utilizadores compreendem o conceito de quadros de subequipas e, como tal, obteve-se 100% de respostas positivas.

A questão 12 tenciona verificar se é percetível que o *input* apresentado no topo da página de criação de retrospetivas, para definir o nome do quadro, diz respeito apenas ao quadro principal. A maioria indica que é percetível, contudo dois participantes indicam o contrário e por isso, é importante refletir para compreender se é possível melhorar este aspeto.

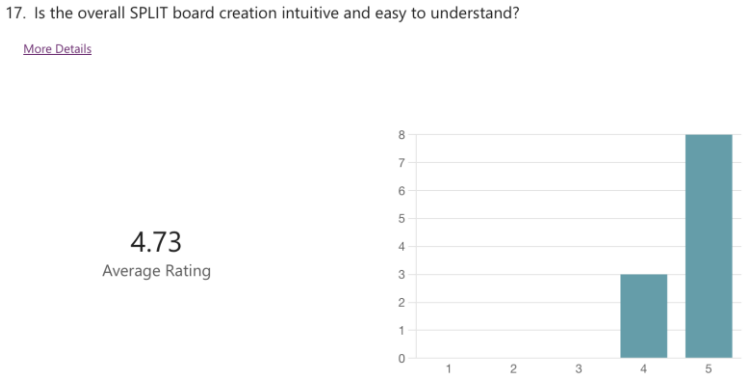
Em relação à questão 13, nesta é questionado se a informação apresentada à direita, na caixa de informação, é útil e obteve-se uma média de 4.64, o que significa que a maioria votou na opção 5 (muito útil), dois participantes votaram na 4 e um participante na três. Embora a maioria esteja em vantagem, não se verifica um consenso absoluto, o que pode indicar uma oportunidade de melhoria.

Na questão 14 pretende-se verificar se todos compreenderam como separar a equipa principal num número superior de subequipas ou, inversamente, apagar subequipas. Todos os participantes responderam que perceberam como adicionar ou apagar subequipas.

Na questão 15 pretende-se compreender se os participantes entendem que ao pressionar o botão para criar o quadro, todos os subquadros também são criados. As respostas foram 100% positivas.

Na questão 16 pretende-se verificar se o processo de criação de retrospetivas divididas e a construção dos respetivos quadros se realizou com sucesso, e pode afirmar-se que se obteve uma taxa positiva de 100%.

Na questão 17 propõe-se a classificação de 1 a 5, do quão é intuitivo e fácil de compreender o processo de criação de uma retrospetiva dividida, como é possível observar na próxima figura. Reforçando o resultado adjacente à questão anterior, verifica-se que a maioria classificou como muito intuitivo (5) e três elementos classificaram de intuitivo (4). Esta classificação permite assumir que o processo e a interface para a criação de retrospetivas divididas foi bem conseguida.



Na questão 18 foram efetuados quatro comentários que são apresentados na próxima figura. Nesta verifica-se que três participantes referiram que a escolha de um novo responsável não é intuitiva, pelo que foi difícil compreender que o botão com o ícone da varinha mágica existe para esse efeito.

18. Do you have any feedback/comments regarding the SPLIT board creation flow?

4 Responses

ID ↑	Name	Responses
1	anonymous	The button to select a new responsible is not that intuitive
2	anonymous	possibility to manually (or automated) select which users will belong to each sub-team
3	anonymous	maybe change the random responsible icon to an icon more explicit, I was a few seconds to try identify what was the icon
4	anonymous	Couldn't understand the Responsible Lottery button at first, probably don't use an icon -- use only text instead also tried to delete a board first without using +/- at the top, but only later realized that those two buttons are the only way -- perhaps add a side button to each subteam to delete

O último comentário também indica que os botões “+” e “–“ são a única forma de adicionar ou remover subequipas, o que não corresponde à realidade – existe o menu de configuração de número de equipas ou número máximo de elementos por equipa. Estes comentários representam oportunidades de melhoria a ter em conta para próximas versões da aplicação.

A secção 5, constituída por três questões, diz respeito à página da listagem de quadros onde é pedido para que esta seja observada atentamente. Na primeira questão da secção e, portanto, número 19, é questionado se é percetível a hierarquia apresentada do quadro principal e dos subquadros associados, a qual obteve 100% de respostas positivas e por isso, é possível assumir que é percetível, que existe uma relação de parentalidade entre os quadros das subequipas e o principal.

Na questão 20 é questionado se é percetível e compreensível que um utilizador apenas deva ter acesso ao subquadro da sua equipa, e as respostas obtidas confirmaram que 100% dos inquiridos compreenderam que os quadros das outras equipas não devem ser acessíveis para todos.

Em relação à questão 21, que pretende que sejam introduzidos comentários ou *feedback* sobre esta página, é referenciado, como é possível observar na próxima figura, que um administrador de equipa não deveria ter acesso a todos os quadros e que os subquadros deveriam ser apresentados de imediato na página, não sendo necessário expandi-los. Considera-se que poder-se-á integrar as sugestões, no entanto apresentam baixa prioridade.

21. Do you have any feedback/comments regarding the boards list page?

2 Responses

ID ↑	Name	Responses
1	anonymous	I don't think a team admin should have access to the sub-teams boards
2	anonymous	maybe the new boards could be presented with the sub-teams expanded

As próximas oito questões integram a secção 6, que diz respeito à página de um subquadro. Nesta é solicitado que se realize um conjunto de operações no quadro, tal como a criação, atualização e remoção de cartões e comentários. Pede-se que se mova os cartões entre as colunas, que se efetue o agrupamento de cartões e a separação dos mesmos. Finalmente, pede-se para editar as configurações do quadro e efetuar o *merge* para o quadro principal.

Na questão 22 é perguntado se a estrutura de navegação presente no canto superior esquerdo dá página é percetível, e todos os participantes responderam de forma positiva.

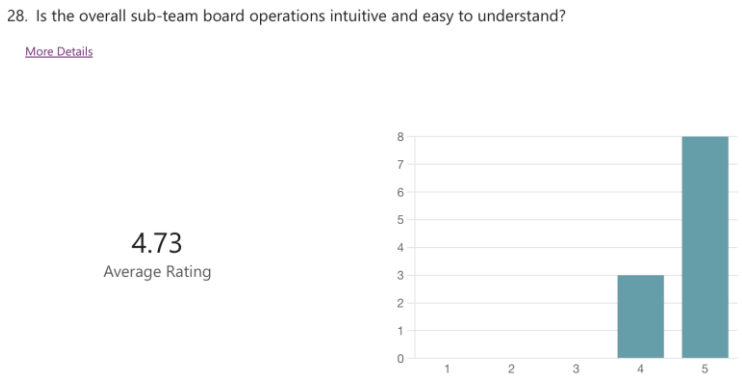
Na questão 23 pretende-se perceber se os utilizadores compreendem como adicionar, editar, juntar, remover e apagar cartões. Apenas um participante respondeu de forma negativa e os restantes de forma positiva.

A questão 24 pretende perceber se os utilizadores compreendem como adicionar, editar e remover comentários, a qual foi respondida com 100% de aprovação.

Na questão 25 pretende-se verificar se é percetível como efetuar o *merge* de um subquadro no principal, verificando-se 100% de respostas positivas.

Tal como na questão anterior, 100% dos participantes responderam à questão número 26 de forma positiva. Nesta pretende-se saber se a interface demonstra que é claro que um determinado subquadro foi *merged* para o quadro principal. Aliada a esta questão, na pergunta 27 pretende-se perceber se os utilizadores entendem que após um quadro ser *merged* não é possível efetuar operações no quadro, e esta foi respondida com apenas um resultado negativo.

A partir das respostas à questão 28, verifica-se que os participantes consideram muito intuitivo ou apenas intuitivo, uma vez que não existiram classificações inferiores ao número 4, sendo que, o 5 foi aquele que reuniu um maior número de votos, como se pode observar na próxima figura. Analisando os resultados, consegue-se definir que as operações num subquadro são bastante intuitivas, o que demonstra que a interface e as funcionalidades implementadas são facilmente utilizáveis.



Apenas dois utilizadores deixaram as suas opiniões sobre esta página, como se pode observar na próxima figura. Um dos comentários indica que não foi acessível descobrir como se pode dar *merge* de cartões quando a definição para os esconder se encontra ativa – Nota: não é possível agrupar cartões quando o quadro tem a opção ativa para esconder os cartões de outros utilizadores. Embora seja indicado que se deve desativar a opção seria importante adicionar alguma informação extra para indicar como aceder às configurações. Finalmente, é indicado que o texto do *template* definido para a coluna "to improve" quando apresenta o carácter "\n", este é descartado quando se inicia a escrita. Ambas as opiniões foram tidas em conta, a última foi incluída no *backlog* e encontra-se resolvida.

29. Do you have any feedback/comments regarding the sub-team board page?

2 Responses

ID ↑	Name	Responses
1	anonymous	I took a bit of time finding how to merge cards because they were hidden, maybe it would be better to give more information and instructions in the popup where it says that the cards are hidden.
2	anonymous	The To Improve template breaks the second line as soon as you start writing

A secção 7 diz respeito à página do quadro principal e é solicitado que se observe a página, é pedido que se altere as configurações e que se adicione votos nos cartões existentes. Esta secção é constituída por 6 questões.

Na pergunta 30 é questionado se o utilizador compreende como as configurações do quadro podem ser alteradas e 100% dos participantes respondeu de forma positiva.

Na questão 31 pretende-se verificar se é percetível quantos quadros de subequipas já foram *merged,* sendo que as respostas foram todas positivas à exceção de um participante.

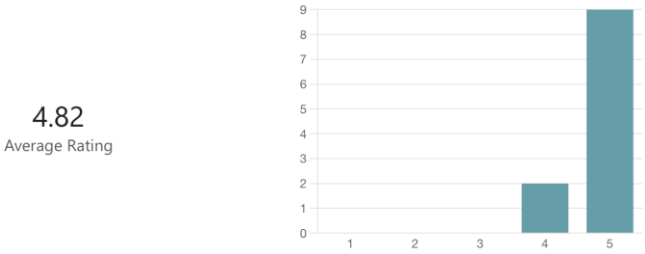
Com a pergunta 32 tenciona-se analisar se o utilizador consegue compreender quais são os cartões da sua subequipa. Tal como nas respostas à pergunta anterior, apenas um utilizador respondeu negativamente.

Todos os utilizadores compreendem como votar nos cartões, uma vez que não existiram respostas negativas para a pergunta número 33.

Na questão 34 propõe-se que os utilizadores indiquem se as operações no quadro principal são intuitivas e fáceis de perceber. Esta questão apenas obteve respostas positivas, isto significa que se reuniu 9 respostas para muito intuitivo (5) e apenas 2 para o valor 4. Estes resultados podem ser observados na próxima figura. Apesar de se poder considerar que a interface da página é bem conseguida, é possível verificar alguns comentários e melhorias nas respostas à próxima questão.

34. Is the overall main board operations intuitive and easy to understand?

More Details



Como referido na questão número 35, são apresentados alguns comentários e opiniões relativos à página em análise. Analisando a próxima figura, verifica-se que o utilizador 1 faz referência à necessidade de visualizar quais os subquadros que já foram *merged,* contudo essa informação existe no topo da página. O utilizador 2 indica que seria útil ter identificado os seus próprios cartões. O utilizador 3 provavelmente definiu-se como *stakeholder* na criação da equipa e por isso indica que não participa em qualquer subequipa. O utilizador 4 indica que as cores dos botões e do texto dos votos não é a adequada. O último utilizador indica que levou algum tempo a encontrar onde pode verificar quais são as equipas que já moveram os seus cartões para o quadro principal. Tendo estas opiniões em conta, verifica-se uma oportunidade de melhoria no que diz respeito à apresentação da informação relativa aos quadros que já forem *merged.*

35. Do you have any feedback/comments regarding the main board page?

5 Responses

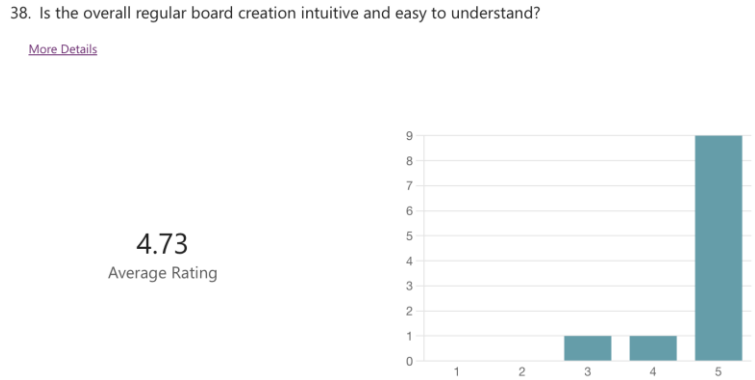
ID ↑	Name	Responses
1	anonymous	show on the boards page which sub-teams already merged their cards
2	anonymous	Maybe it would be useful to have some kind of indication of whether a certain card was written by me or by someone on my team.
3	anonymous	I didn't have a sub-team so it's clear which one was mine
4	anonymous	If I haven't careful, seems like I add one positive vote in all cards, because the color for positive with 0 votes is black, and the negative is grey
5	anonymous	Took me a while to find place where it says how many other sub-teams already merged their boards

Na secção 8 é solicitado que o participante crie um quadro de retrospetiva regular selecionando alguns participantes, definindo as configurações iniciais e que posteriormente responda às perguntas apresentadas.

Na pergunta 36 é questionado se os utilizadores compreenderam como adicionar e remover participantes. As respostas foram totalmente positivas.

Na questão 37 é perguntado se o quadro foi criado com sucesso e todos os participantes responderam positivamente.

Os resultados da questão 38 demonstram que o processo e interface de criação de um quadro de retrospetiva regular é intuitiva e fácil de compreender. Analisando os resultados apresentados na próxima figura, obteve-se uma média de 4.73 de *rating*, sendo que 9 participantes classificaram como muito intuitivo, um participante indica que é intuitivo (4) e outro que é neutro (3).



Relativamente aos comentários fornecidos nesta secção, na questão 39, possíveis de verificar na próxima figura, o utilizador 1 sugere a possibilidade de efetuar retrospetivas divididas em que os participantes de cada subequipa são escolhidos livremente, em vez de serem aleatoriamente selecionados. O utilizador 2 indica que após selecionar participantes, mudar de *tab* e voltar à seleção dos mesmos, os utilizadores selecionados são descartados. Este problema foi prontamente resolvido após a análise dos resultados deste formulário.

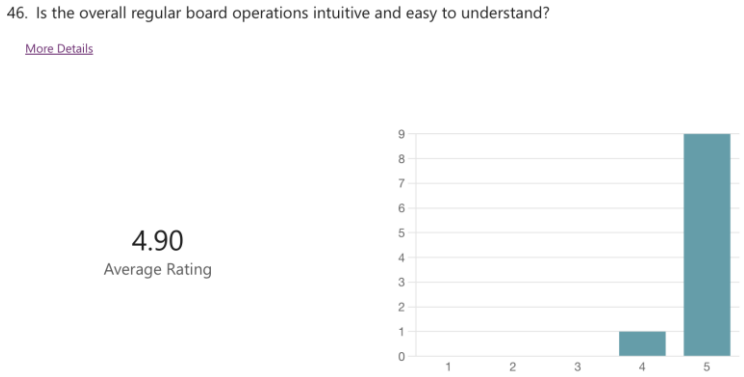
39. Do you have any feedback/comments regarding the regular board creation flow?

2 Responses

ID ↑	Name	Responses
1	anonymous	instead of allowing to specify which users will participate in a regular board, maybe also allow them to have a main board and sub-teams, basically a split retro but with specified teams instead of randomized
2	anonymous	When you select the participants and then go to the configurations, and back, the participants are gone

A secção 9 e última, diz respeito à página de um quadro de retrospetiva regular onde é solicitado que se interaja com o *timer*, que se adicione e apague colunas, que se adicione cartões e altere as cores dos mesmos, pré-defina o texto dos cartões de uma coluna, altere o nome de uma coluna, remova todas os cartões de uma coluna através da opção para o efeito, mova uma coluna de posição e finalmente que se adicione e remova participantes. Esta secção é composta por 7 questões.

A questão 40 pretende compreender se os utilizadores sabem como utilizar o



temporizador, cujos resultados foram totalmente positivos, na medida em que apenas se obteve respostas afirmativas.

Na questão 41 verifica-se que todos os utilizadores conseguiram adicionar e remover colunas.

A questão 42 trata de interrogar se os utilizadores entenderam como alterar as configurações de cada coluna, e todos indicaram que sim, à exceção de um utilizador.

Através dos resultados da questão 43, verificou-se que todos os utilizadores conseguiram mover a coluna de posição através de *drag and drop.*

A partir dos resultados da questão 44, verificou-se que apenas um utilizador não compreendeu como adicionar ou remover participantes a partir da página do quadro. Nesta questão verifica-se menos uma resposta, porque a questão não se encontrava realizada quando o primeiro inquirido respondeu às perguntas do formulário.

A questão 45 pretende perceber se as operações nas colunas são intuitivas e fáceis de compreender e os resultados foram bastante positivos, na medida em que todos consideraram muito intuitivo (5) e apenas um considerou intuitivo (4).

Analisando do ponto de vista geral, através dos resultados da questão 46, observáveis na próxima figura, verifica-se que apenas um participante votou no valor 4, sendo que os restantes consideraram muito intuitiva, o que leva a crer que a interface e as funcionalidades relativas ao quadro de uma retrospetiva regular foram bem-sucedidas.

Finalmente, foram reunidas duas opiniões sobre a página em análise, como é possível observar na próxima figura. Assim sendo, o primeiro utilizador indica que não é claro o propósito do temporizador e que seria útil incluir uma breve descrição em algum lugar da

página. De forma semelhante, não é claro para o utilizador 2, que um responsável num quadro de retrospetivas regular é um utilizador com privilégios. Este também indica que o temporizador poderia ter algum tipo de configuração e associação a eventos/fases da retrospetiva. Estas observações são tidas em conta e seguem para análise pelas partes responsáveis, nomeadamente a *designer,* com o intuito de as solucionar.

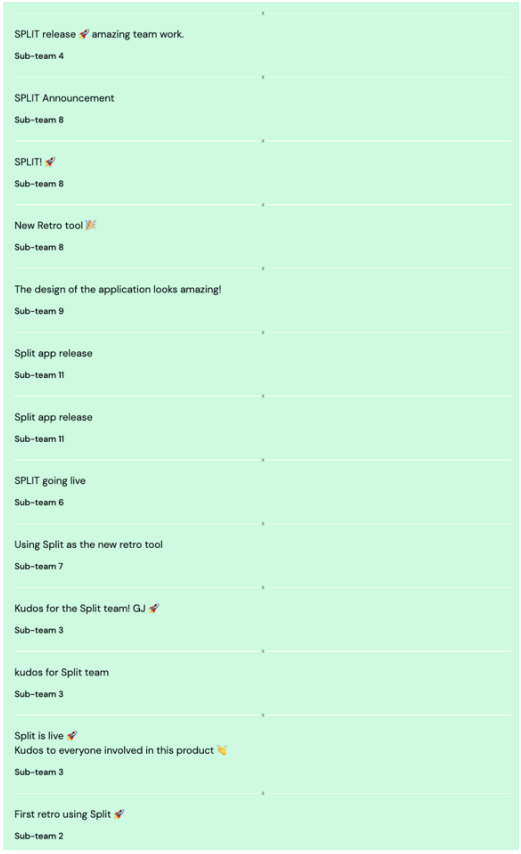
47. Do you have any feedback/comments regarding the regular board and the columns configurations?

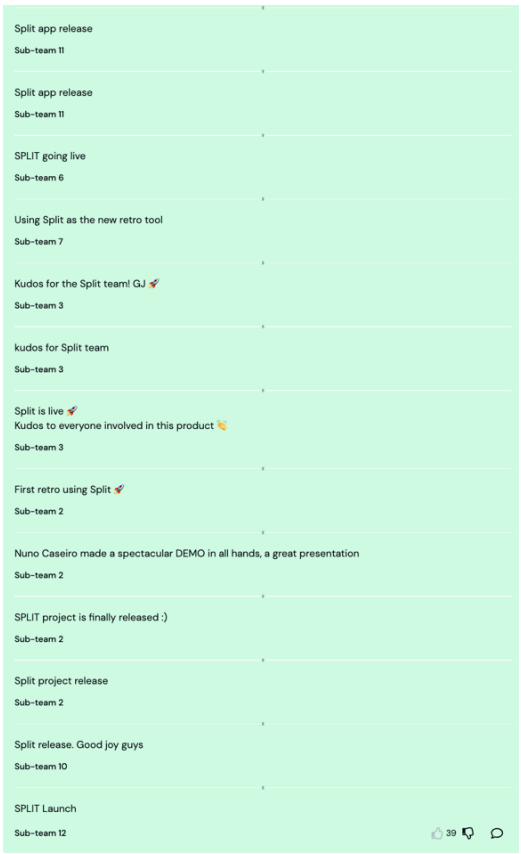
2 Responses

ID ↑	Name	Responses
1	anonymous	The reason why I'd like to use a timer is not clear. Maybe this could be a configuration and in the description it can have a small explanation.
2	anonymous	What's the purpose of the responsible toggle here? the timer could have a configuration for the description of the timed event

Analisando os resultados obtidos e efetuando a média das questões referentes à classificação geral de cada secção obtém-se o valor de 4.81, é possível concluir que, segundo as respostas dos participantes no inquérito, a plataforma e a respetiva interface foram bem conseguidas, sendo que todas as páginas foram consideradas bastante intuitivas, e por isso pode-se considerar que oferece uma excelente experiência de utilização. As oportunidades de melhoria foram tidas em conta, algumas das sugestões já se encontram.

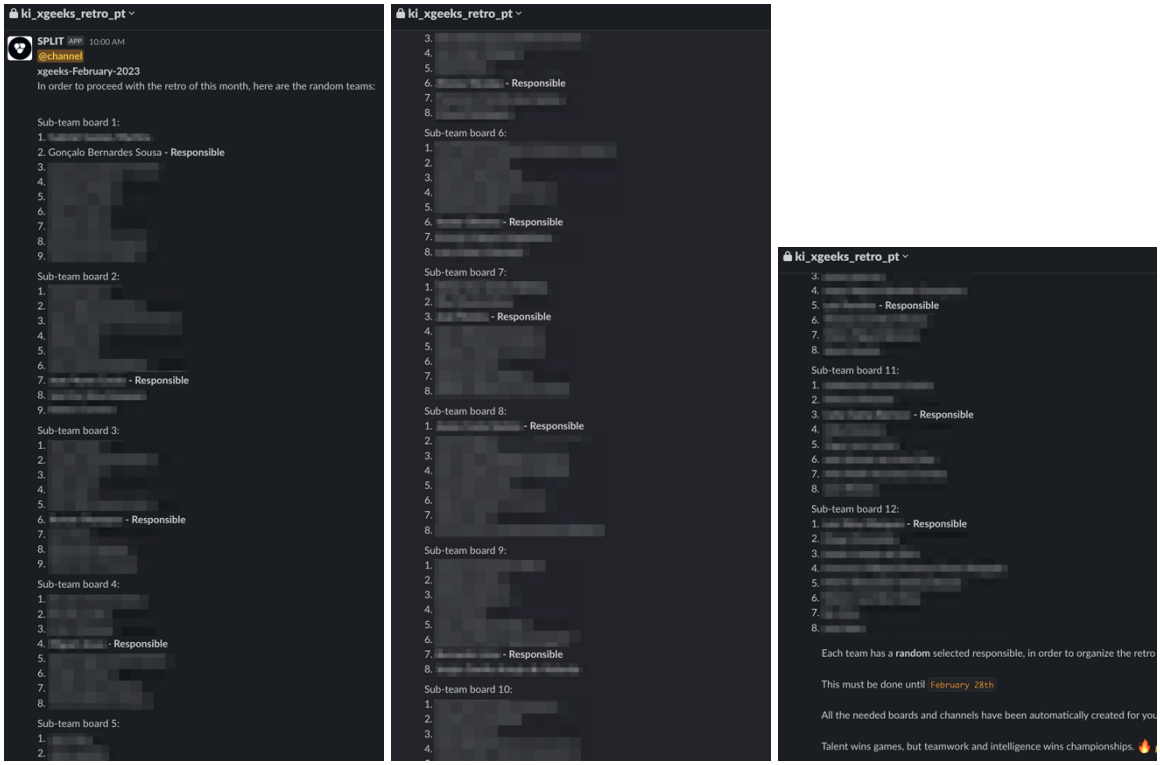
Apêndice T – Comentários sobre a plataforma





Apêndice U – Resultados 2ª retrospetiva

Mensagem no *slack*



Quadro principal