Desenvolvimento Web

Desenvolvimento e Manutenção de um Marketplace para a Indústria da Moda

Mestrado em Engenharia Informática - Computação Móvel

Xavier Santos Oliveira

Leiria, setembro de 2022

Desenvolvimento Web

Desenvolvimento e Manutenção de um Marketplace para a Indústria da Moda

Mestrado em Engenharia Informática - Computação Móvel

Xavier Santos Oliveira

Estágio realizado sob a orientação do Professor Doutor Alexandrino Gonçalves e sob supervisão do Sr. Engenheiro Luís Soares

Leiria, setembro de 2022

Originalidade e Direitos de Autor

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

Agradecimentos

Quero agradecer, em primeiro lugar, ao meu orientador, Professor Alexandrino José Marques Gonçalves, pela sua disponibilidade, acompanhamento e paciência ao longo do estágio e escrita do relatório.

Aos meus pais, pelo apoio incondicional, pelo encorajamento para concluir mais uma etapa importante na minha vida.

A todos os meus colegas e superiores na *xgeeks*, em especial ao Eng. Luís Soares pela mentoria durante o período de estágio.

Por fim, mas de igual importância, à Débora Antunes pelo apoio e paciência que teve para comigo e pela sua ajuda na revisão deste relatório.

Resumo

A *world wide web* encontra-se num desenvolvimento e crescimento constantes, com o surgimento de novas tecnologias e ferramentas. *Next.js*, *React.js*, *Typescript*, *Kubernetes* podem ser consideradas algumas das mais recentes e neste estágio tive a possibilidade de não só as aprender, como de aplicá-las num projeto real. Durante este estágio foi desenvolvido um *marketplace* para a indústria da moda desde o seu estado inicial, até à sua entrada em produção. O projeto destinou-se a um cliente de um grande grupo económico do Médio Oriente que procurava ser, neste domínio, uma referência na sua região. Este *marketplace* tinha, não só, como prioridade potenciar a experiência do utilizador neste mercado digital, desde a pesquisa dos produtos, processo de compra, até à entrega dos mesmos; como iria suportar inúmeras marcas luxo, algumas delas exclusivas. Outros dos objetivos eram o desenvolvimento de um portal escalável, a fim de alcançar mais de um milhão de produtos, e com suporte para dois idiomas de culturas substancialmente diferentes, como são o inglês e o árabe. Para tal, o portal foi desenvolvido nativamente para *desktop*, sendo totalmente responsivo, permitindo igualmente uma visualização adequada e eficaz em dispositivos móveis. Este foi um projeto desafiante, tanto na sua escala, como no período de tempo para o seu desenvolvimento, como nos requisitos muito particulares que tinha. Este *marketplace* está neste momento em produção e é já um caso de sucesso no Kuwait, tendo atingido mais de dois milhões de euros no valor de vendas.

Palavras-chave: Desenvolvimento Web, Árabe, *Frontend*, *Marketplace*.

Abstract

The world wide web has been in constant development and growth, with new technologies and tools continuously appearing. *Next.js*, *React.js*, *Typescript* or *Kubernetes* can be considered some of the most recent ones and in this internship, I had the opportunity of, not only learning them, as to use them in a real project. A marketplace for the fashion industry was developed in this internship, since its initial state until releasing it for production, for a client of a big Middle East economic group, who desires to be a reference in its region. This marketplace had as priority enhancing the user experience in this online market, since the search of products, checkout, until delivery of the items, as well as providing several luxury brands, some of them exclusive. Other important goals were the development of a scalable portal to reach more than a million products, and with support for languages of two quite different cultures, such as English and Arabic. To this end, the portal was natively developed for desktop equipment's, although being fully responsive since it offers an adequate and effective visualization on mobile devices. This was a challenging project, both in its scale and in the little time of development available, as with its quite specific requirements. Nowadays, this marketplace is in production, and is a successful business in Kuwait, having reached more than two million euros on its sales.

Keywords: Web Development, Arabic, Frontend, Marketplace.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

API	Application Programming Interface
AWS	Amazon Web Services
CD	Continuous Delivery
CI	Continuous Integration
CMS	Content Management System
CRM	Customer Relationship Management
CSS	Cascading Style Sheets
CTA	Click To Action
CV	Curriculum Vitae
DOM	Document Object Model
FAQ	Frequently Asked Questions
GMT	Greenwich Mean Time
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IP	Internet Protocol
ISR	Incremental Statis Regeneration
JSON	JavaScript Object Notation
KNET	K-Net Payment Gateway

LTR	Left To Right
MVP	Minimum Viable Product
PWA	Progressive Web App
QA	Quality Assurance
RGPD	Regulamento Geral de Proteção de Dados
RTL	Right To Left
SaaS	Software as a Service
SEO	Search Engine Optimization
SPA	Single Page App
SSG	Static Site Generation
SSR	Server-Side Render
SWR	Stale While Revalidate
TI	Tecnologias de Informação
UI	User Interface
URL	Uniform Resource Locators
UX	User Experience
VPN	Virtual Private Network
VSCode	Visual Studio Code
XP	eXtreme Programming

1 Introdução

Este documento surge do âmbito da unidade curricular de Estágio, do mestrado em Engenharia Informática – Computação Móvel do Instituto Politécnico de Leiria e descreve o trabalho realizado no decorrer de um estágio na empresa *xgeeks* Portugal [1], sediada em Leiria, com a duração de 9 meses (17 de agosto de 2020 a 17 de maio de 2021). Este estágio visou o desenvolvimento das competências associadas à área de formação da oferta formativa e tecnologias associadas.

Nos últimos anos tem havido uma tendência por parte dos consumidores de utilizar canais digitais online, por oposição a canais físicos offline, para realizar compras nas mais diversas áreas [2]. Por causa desta mudança de comportamento, o *e-commerce* tem vindo a crescer e a tornar-se lucrativo [3]. Este crescimento foi potenciado pela pandemia por COVID 19, na medida em que a criação de medidas que obrigavam os consumidores a permanecer em casa os condicionou a fazerem compras online [4]. As empresas na indústria da moda têm vindo a apostar cada vez mais na criação de *marketplaces* online onde o utilizador pode escolher, comparar e comprar o produto desejado com maior facilidade.

A tecnologia associada à *world wide web* tem vindo a adaptar-se à crescente inovação. Todos os utilizadores desta rede universal querem acesso à informação de forma rápida e prática. Todo este processo de crescimento e adaptação levou ao surgimento de diversas bibliotecas *JavaScript open-source* [5], em especial o *React.js* [6] que permite a construção de portais web com uma interface funcional e apelativa, de forma simples e eficiente. A par destas bibliotecas surgiram novas *frameworks* como *Next.js* [7], também *open-source*, que é compatível com *React.js* e oferece diversos benefícios, como um melhoramento de performance, melhor experiência para o desenvolver de *software*, a eficiência do *Search Engine Optimization* (SEO), entre outros.

1.1 Objetivos

O objetivo principal deste estágio foi incorporar uma empresa na área das Tecnologias de Informação (TI) e integrar um projeto com um cliente real para poder aplicar os conhecimentos e de competências teórico-práticas adquiridos ao longo do curso de licenciatura em Engenharia Informática, bem como do mestrado supramencionado. Mais especificamente, os objetivos deste estágio foram:

- Ser introduzido na área profissional das TI, desenvolvendo *software* para um cliente com necessidades reais;
- Obter formação por parte de engenheiros mais experientes e orientação para aprender novas e relevantes tecnologias para o mundo de trabalho;
- Desenvolver outros tipos de habilidades como, comunicação, trabalho em equipa, resolução de problemas, análise crítica e sentido de responsabilidade.

Dado que fui integrado num projeto com um cliente real, outras objetivos e desafios emergiram, nomeadamente:

- Familiarizar-me com as diferentes ferramentas, tecnologias e metodologias usadas no projeto;
- Implementar funcionalidades para o projeto do cliente, desenvolvendo código com qualidade, que siga as melhores práticas;
- Desenvolver todas as funcionalidades do projeto, de forma a estas seguirem os requisitos do cliente originário de uma cultura substancialmente diferente da ocidental, como o suporte a dois idiomas díspares na sua sintaxe;
- Apoiar o cliente nos seus processos internos, representando a empresa de acolhimento da melhor forma possível.

1.2 Estrutura do Relatório

O capítulo 2 (Enquadramento) contém a caracterização da entidade de acolhimento, do cliente e do projeto no qual estive integrado. O capítulo 3 (Metodologias e Ferramentas) apresenta uma visão geral das principais metodologias, tecnologias e ferramentas com as quais tive contacto durante o estágio.

O capítulo 4 descreve o Desenvolvimento do projeto, no qual se expõe o estado inicial do projeto, assim com a sua arquitetura. Este capítulo, descreve igualmente o desenvolvimento de diversas funcionalidades no qual estive envolvido, como as páginas estáticas, componentes desenvolvidas, a adaptação do conteúdo à orientação de diferentes idiomas, testes e monitorização. No final do Desenvolvimento também apresento alguns processos internos do cliente, no qual fui incluído, e o estado do projeto quando finalizei o estágio.

E por fim, a conclusão, no capítulo 5, sumariza o que foi alcançado durante o estágio e o trabalho futuro.

2 Enquadramento

Este capítulo destina-se a dar conhecer a empresa do qual realizei o estágio e também o projeto do cliente que fiquei alocado durante o mesmo. Por motivos de confidencialidade da empresa para a qual desenvolvi o trabalho realizado, a mesma será referida por *KuwaitShop* ao longo deste relatório.

2.1 Caracterização da Entidade de Acolhimento

A entidade de acolhimento do estágio foi a *xgeeks*, sedeada em Leiria, Portugal. A *xgeeks* é uma empresa jovem, fundada em dezembro de 2019, que tem apresentado um elevado e contínuo crescimento [1].

Esta faz parte da *KI group* [8], um grupo alemão que possui diversas empresas que apoiam os clientes em especialidades distintas. A *xgeeks* está focada em difundir o conhecimento de engenharia a cada um dos clientes, desenvolvendo *software* e sistemas adaptados para as suas necessidades. Esta empresa tem nos seus quadros engenheiros que trabalham com uma variedade de tecnologias e plataformas atuais, tais como, *React.js* [6], *Docker* [9], *Kubernetes* [10], *Amazon Web Services* (AWS) [11], *Azure* [12], *GO* [13] e *Django* [14].

A *xgeeks* lida com vários clientes de diversos setores, como *DHL* [15], que trabalha na logística de transporte, *Porsche* [16], *Cazoo* [17] e *Mercedes-Benz* [18] que operam na indústria automóvel, entre outros. Estes clientes estão sediados em diversos países europeus como a Alemanha e o Reino Unido.

A *xgeeks* é uma empresa que se tem destacado na cidade de Leiria pelo seu crescimento, tendo neste momento três escritórios nas cidades de Leiria, Lisboa e Viseu. Aquando do início do estágio, esta tinha 28 desenvolvedores de *software* e na data de escrita deste documento a empresa cresceu para 94 desenvolvedores de *software*, com uma distribuição de género 91% homens e 9 % mulheres, tendo uma média de idades de 30 anos.

Desde o início do estágio, a empresa disponibilizou todo o material que me permitisse desempenhar as minhas funções como desenvolvedor de *software*. Dos quais se incluem um computador portátil, rato, monitor e auscultadores.

Relativamente ao local de trabalho, a maior parte do estágio foi realizado remotamente, devido às restrições causadas pela pandemia da Covid-19, estando a opção de trabalhar de forma remota já disponível na empresa ainda antes de tal ser decretado pelo governo de Portugal.

2.1.1 Formação

A integração na empresa, foi sistematizada e pensada de forma a dar a conhecer os diversos serviços, plataformas, ferramentas disponíveis e usadas pela empresa, através de um quadro digital com tarefas, *Trello* [19] (Figura 1). Após este período de integração ao ambiente da empresa, que ocorreu na primeira semana de estágio, seguiu-se uma fase de desenvolvimento de capacidades vocacionadas a *frontend* e a *DevOps*, que ocorreu de 24 de agosto a 23 setembro.

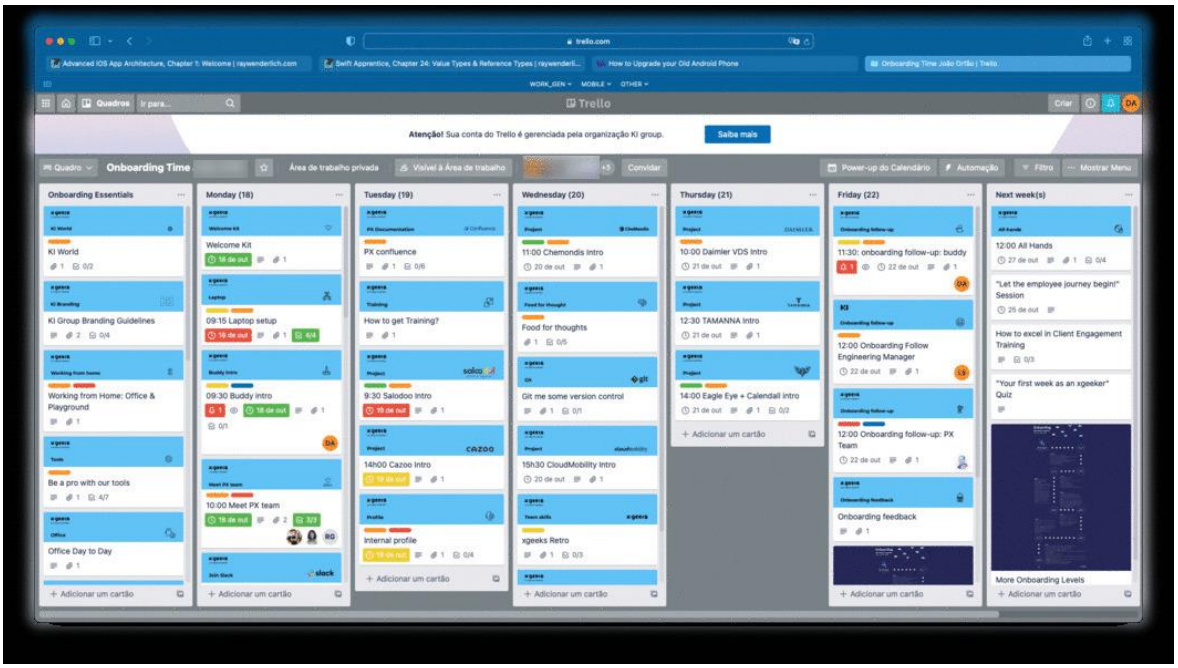


Figura 1 - Exemplo do calendário de *onboarding* no *Trello*

DevOps é uma mais-valia para qualquer engenheiro de *software*, já que abarca conceitos de *clusteres, nodes, deploy* de serviços/servidores que beneficia qualquer desenvolvedor de *software* quando os utiliza, para tal foram estudados e realizados cursos que incidiam no tema de *Kubernetes, Docker* e *Git* [20].

Como escolha pessoal, fui formado na área do *Frontend* onde veio a incidir a minha função no projeto que fiquei alocado. Assim sendo, os cursos visualizados foram essencialmente sobre

React.js, controlo de estados, *hooks*, persistência de informação (*Redux*), *TypeScript* [21] (uma vez que é uma linguagem em crescimento e utilizada em diversos projetos na empresa), *JavaScript* e conceitos avançados, entre outros como *Promises* (*Async/Await*).

2.2 Caracterização do Cliente

O Grupo *KuwaitShop* opera em múltiplos setores que incluem a moda, alimentação, saúde, beleza e entretenimento (Figura 2). Esta domina o mercado no Médio Oriente. O cliente visa a criação de uma empresa para operar no mercado online na área da moda, dado que que já controlam diversos centros comerciais com um elevado número de lojas físicas, acima das 4500, querendo alcançar um novo mercado que cada vez é mais relevante e lucrativo. O facto desta pretensão surgir numa época em que a sociedade estava a atravessar um período pandémico, com sucessivos confinamentos, só veio reforçar esta relevância.

Figura 2 - Empresas associadas ao cliente

2.3 Caracterização do Projeto

Este projeto procura destacar-se no Médio Oriente como um *marketplace* composto pelas mais prestigiadas marcas no mundo da moda, tendo como prioridade fornecer a melhor experiência ao cliente, acompanhando-o desde a pesquisa dos itens, processo de compra, até à receção da sua encomenda. Esta empresa foi criada em maio de 2020, com a meta de alcançar um *Minimum Viable Product* (MVP) até novembro desse mesmo ano e, para tal, pretendia o

desenvolvimento de um portal responsivo, para o crescente número de utilizadores com dispositivos móveis, e escalável para poder gerir os milhares de produtos à venda.

2.3.1 Desafios

Os maiores desafios que apareceram surgiram devido ao contexto cultural do cliente, uma vez que este tem como idioma nativo o árabe, logo, todo o conteúdo teria de ser traduzível. Uma particularidade deste idioma é ser lido da direita para a esquerda, requerendo então, não só a tradução do texto, mas também a alteração da orientação do mesmo tal como o seu ponto de início. O conteúdo gráfico também está incluído neste procedimento e teve de ser todo adaptado para tornar-se responsivo consoante o idioma escolhido pelo utilizador.

Surgiram diversas dificuldades associadas à localização geográfica do cliente e o seu público-alvo. Devido à localização da empresa estar sedeada no Kuwait, o fuso horário era diferente. Enquanto a equipa de engenharia estava em Portugal com o fuso horário *Greenwich Mean Time* (GMT), os restantes membros da empresa trabalhavam no fuso horário GMT +03:00, implicando que a colaboração com os colegas do Kuwait terminasse às 15h, hora portuguesa. Devido a questões culturais do cliente, a semana de trabalho começa no domingo, ou seja, eles trabalham de domingo até quinta-feira, sendo sexta-feira e sábado fim-de-semana para eles, reduzindo a possibilidade de discutir ideias ou ter reuniões com a equipa presente no Kuwait na sexta-feira.

Após a entrada em produção do *marketplace*, esta questão relacionada com o fim-de-semana tornou-se uma dificuldade acrescida, dado que o domingo corresponde ao dia em que os locais mais fazem compras e isto refletiu-se também nas visitas ao *marketplace*, sendo mais suscetível o surgimento de problemas, como de pagamento ou no acesso a produtos fora do horário de trabalho da equipa de engenharia, localizada em Portugal. Para estes casos foi criada uma equipa de gestão de incidentes que iria ser convocada para resolver o problema. Outra dificuldade emergiu com as particularidades de trabalhar remotamente durante as restrições causadas pela pandemia. Esta separação física dificultou a resolução de algumas inseguranças e questões inerentes ao primeiro contacto com o mundo de trabalho na área das Tecnologias de Informação (TI) e que seriam facilmente resolvidas num contexto pessoal.

3 Metodologias e Ferramentas

Para melhor entender como este *marketplace* funciona e foi desenvolvido, é importante salientar alguns conceitos chave. Neste capítulo serão descritas as metodologias, as tecnologias e ferramentas utilizadas, assim com alguns exemplos de *marketplaces* de sucesso usados como inspiração para o desenvolvimento da *KuwaitShop*.

3.1 Metodologia

As metodologias ágeis podem trazer sucesso a um projeto e são utilizadas desde a área tecnológica à indústria [22]. Apesar destas já existirem anteriormente, foi em 2001 que um grupo formado por *Kent Beck* e dezasseis colaboradores de renome assinaram o Manifesto para o Desenvolvimento Ágil de *Software* e o grupo foi batizado de aliança dos ágeis [23].

As metodologias ágeis defendem a satisfação do cliente, a entrega incremental prévia, equipas pequenas e altamente motivadas, artefactos de engenharia de *software* mínimos e, acima de tudo, simplicidade no desenvolvimento. Os princípios de desenvolvimento priorizam a entrega, mais do que a análise e o projeto, também priorizam a comunicação ativa e contínua entre desenvolvedores de *software* e clientes (Figura 3).

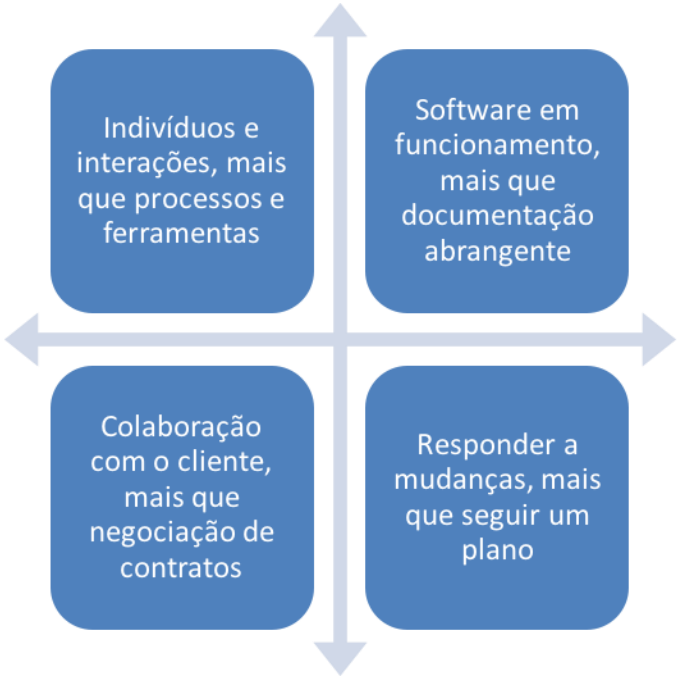


Figura 3 - Valores defendidos no manifesto das metodologias ágeis

3.1.1 Extreme Programming

Extreme Programming (XP) é uma metodologia ágil de desenvolvimento de *software* que visa uma produção de *software* com maior qualidade e uma maior capacidade de reação a alterações de requerimentos do cliente [24]. O XP é a mais específica das metodologias ágeis em relação às práticas de engenharia apropriadas para o desenvolvimento de *software*.

3.1.1.1 Practices

O núcleo do XP é o seu conjunto interconectado de práticas de desenvolvimento de *software*, esquematizadas na Figura 4. Embora seja possível realizar essas práticas isoladamente, muitas equipas descobriram que algumas práticas complementam outras e devem ser feitas em conjunto para eliminar completamente os riscos que muitas vezes as equipas enfrentam no desenvolvimento de *software*.

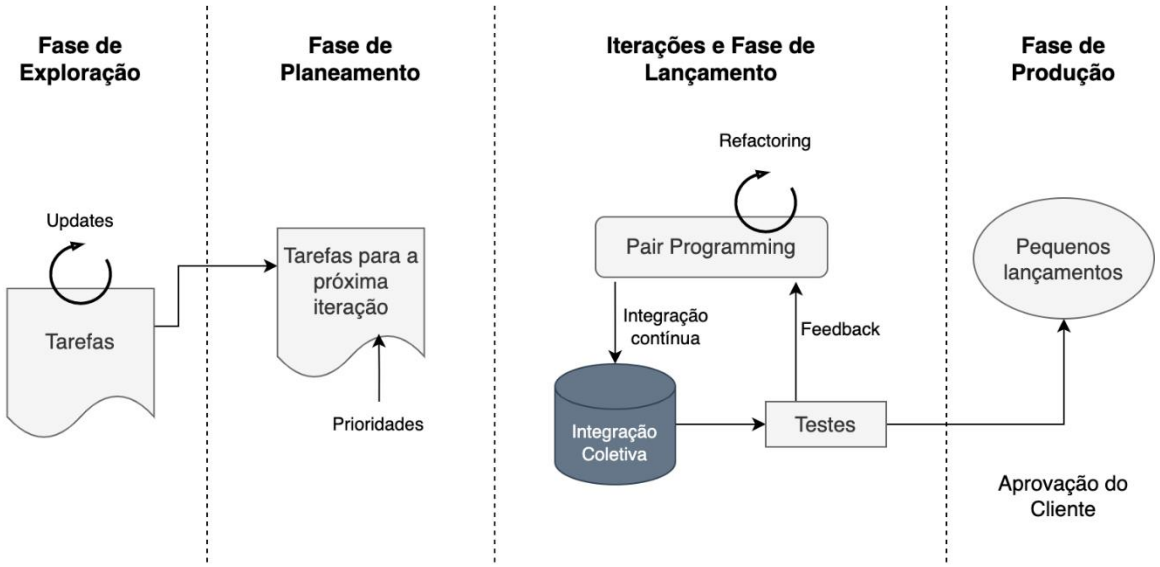


Figura 4 - Práticas usadas no XP

A fase de exploração inclui uma prática que corresponde à criação de tarefas a partir de um conjunto de requerimentos fornecidos pelo cliente para o *software*. Algumas destas são selecionadas na fase seguinte, pelo cliente e pelos desenvolvedores de *software* que se comprometem com uma lista de funcionalidades que irão estar incluídas no próximo lançamento de uma nova versão do *software*.

A fase de lançamento e as iterações incluem diversas práticas que têm como objetivo um desenvolvimento eficaz dessa lista de funcionalidades, das quais se destacam o *pair programming*, *refactoring* e testes.

O *pair programing*, uma forma colaborativa de trabalhar que envolve comunicação. Com esta prática dois desenvolvedores de *software* juntam-se não só para escrever código e concretizar a tarefa, mas também fazer o planeamento, a discussão de ideias e de diferentes abordagens para finalizar a tarefa. Tem um papel importante para compartilhar conhecimento entre a equipa, sendo especialmente eficaz na integração de juniores num novo projeto [25].

O *refactoring* é uma prática que envolve a reestruturação de um código já existente e que altera a sua estrutura interna, sem alterar o seu comportamento. Esta prática traz vantagens para projetos que estão a ser desenvolvidos durante longos períodos, pois pequenas alterações são muitas vezes aplicadas sobre outras, levando desnecessariamente a um aumento da complexidade. Desta forma, identifica-se e realizam-se otimizações que permitem uma boa manutenção do código [26].

Por fim, a prática de testar consiste no desenvolvimento de *software* acompanhada por um desenvolvimento sistemático de testes, com o objetivo de reduzir número de erros no *software*. Estes testes assentam na prática de pequenos lançamentos que não só permitem um desenvolvimento contínuo e incremental, mas também inclui o feedback do cliente antes do lançamento final do produto, facilitando o aprimorando o *software*/produto.

3.1.2 Scrum

A metodologia *Scrum* [27] é uma estrutura de processos, utilizada na gestão do desenvolvimento de produtos de *software* complexos, ajudando as equipas a criar soluções de uma forma iterativa. Como também é uma metodologia ágil tem algumas semelhanças com a metodologia XP, como a divisão dos requerimentos em pequenas unidades, implementar o projeto de forma incremental e priorizar as tarefas que trazem mais valor para o produto. No entanto estas duas metodologias diferem principalmente nas práticas da engenharia de *software*, sendo que a metodologia XP dá mais ênfase nas técnicas de programação usadas pela equipa, ao passo que a metodologia *Scrum* delega as decisões de desenvolvimento para os desenvolvedores de *software* sem ditar quais as técnicas que devem ser utilizadas [28].

A estrutura da metodologia *Scrum* inclui uma equipa composta por vários papéis que realizam uma série de funções, tendo ao seu dispor uma variedade de eventos e artefactos, bem como várias regras a seguir.

3.1.2.1 Eventos e Artefactos

Eventos programados são utilizados na metodologia *Scrum* para criar regularidade e minimizar a necessidade de reuniões não definidas. A principal unidade de tempo desta Metodologia é a *Sprint*. Esta é um evento de duração fixa que não pode ser reduzida ou aumentada, normalmente dura duas semanas a um mês, dependendo da equipa. Cada *Sprint* começa imediatamente a seguir à anterior, e todos os outros eventos ocorrem durante a *Sprint*. Os restantes eventos, não têm duração fixa e podem terminar sempre que o objetivo do mesmo for alcançado, garantindo que uma quantidade apropriada de tempo seja despendida sem permitir desperdício no processo.

Na Figura 5 está caracterizado o ciclo subjacente aos eventos constituintes de uma *Sprint*. A *Sprint* começa com a ***Sprint Planning***, um planeamento inicial de tarefas a serem concluídas no decurso da mesma. Durante a *Sprint* existe a ***Daily Scrum***, uma reunião diária para partilhar o estado das tarefas e fazer um levantamento de obstáculos para a conclusão destas, caso existam. No final da *Sprint* existe a ***Sprint Review*** na qual é feito o levantamento se o objetivo da *Sprint* foi atingido e se as tarefas foram terminadas e de seguida a ***Sprint Retrospective*** na qual se identificam problemas que ocorreram durante a *Sprint* e possíveis melhoramentos para as seguinte *Sprints*.

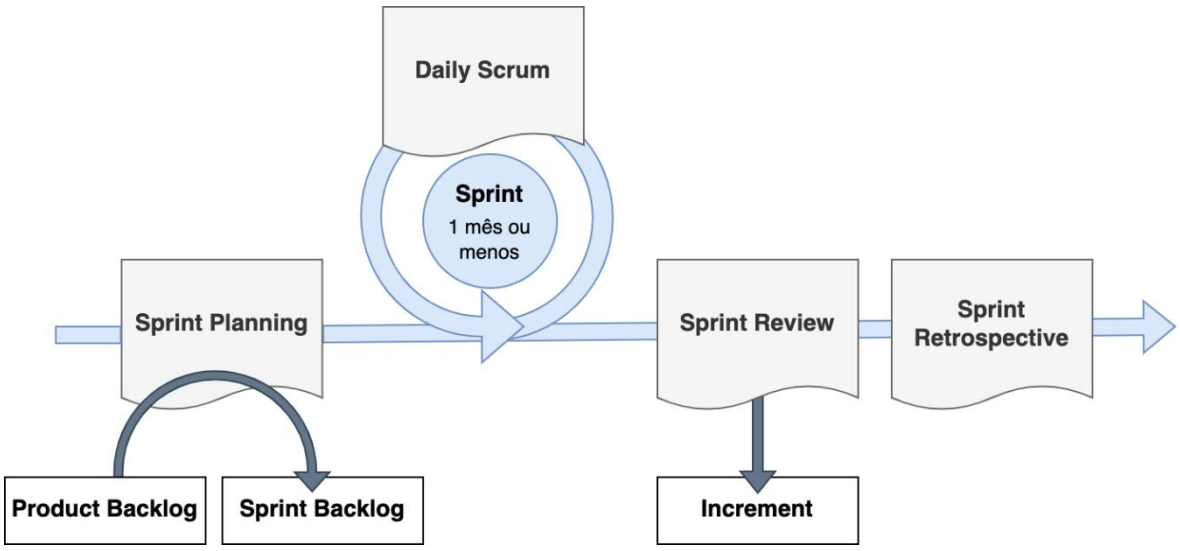


Figura 5 - Eventos e artefactos associados a uma *Sprint*

Os artefactos do *Scrum* são a informação que a equipa de *Scrum* usa para detalhar o produto a ser desenvolvido, as ações para o produzir e o trabalho resultante destas ações. Estes artefactos, definidos na metodologia *Scrum*, são projetados especificamente para maximizar a transparência das principais informações, para que todos tenham o mesmo entendimento do artefacto.

O ***Product Backlog*** corresponde a todas as tarefas necessárias para o produto, ou seja, as tarefas com as funcionalidades e os requerimentos do cliente para obter o Produto idealizado.

O ***Sprint Backlog*** contém as tarefas que foram delimitadas a serem desenvolvidas durante a *Sprint*, estas tarefas estão definidas no *Product Backlog*.

O ***Increment,*** são as funcionalidades concluídas e que são entregues ao cliente. Estas surgem aquando da conclusão das tarefas do *Sprint Backlog*. É necessário acordar entre a equipa de *Scrum* qual o critério para a conclusão de uma tarefa. Desta forma cada *Sprint* gera um *Increment* para aumentar o valor do produto final.

3.1.2.2 Os diferentes papéis da equipa de Scrum

As equipas de *Scrum* são equipas com um pequeno número de pessoas com os seguintes papéis:

- ***Product Owner***: Apenas uma pessoa. Tem conhecimento das necessidades do cliente e outras partes interessadas, comunicando ao resto da equipa o objetivo do projeto e os requisitos. Também é o responsável pela criação e comunicação das tarefas, podendo delegar este trabalho a outros membros da equipa;
- ***Scrum Master***: Apenas uma pessoa. É responsável por assegurar que a metodologia *Scrum* está a ser bem executada, planeando os eventos, auxiliando o *Product Owner* e facilitando a comunicação com o cliente e partes interessadas;
- **Desenvolvedores de *Software***: Os restantes membros da equipa. Desenvolvem o produto incrementalmente a cada *Sprint*, realizando diversas tarefas e adaptando o seu planeamento a um objetivo final de *Sprint*.

O modelo de equipa no *Scrum* foi desenvolvido para otimizar a flexibilidade, a criatividade e a produtividade. Estas equipas são interfuncionais, pois os membros têm todas as caraterísticas necessárias para criar valor para o produto a cada *Sprint* e auto gerenciadas porque decidem internamente quem faz o quê, quando e como.

3.2 Tecnologias

Neste subcapítulo estão identificadas as tecnologias que foram utilizadas durante o desenvolvimento do projeto ao longo do estágio.

3.2.1 TypeScript

TypeScript é uma extensão *open-source* da linguagem de programação *JavaScript* com o objetivo de melhorar o desenvolvimento de aplicações em larga escala [21]. O *TypeScript* visa resolver alguns dos problemas do *JavaScript* acrescentando tipos de dados como genéricos, módulos, classes, interfaces e *type system*. Esta é então uma linguagem compilada de tipos estáticos que gera código em *JavaScript* [29]. A adição de tipos de dados e a compilação do código permite ao *TypeScript* encontrar erros antes de correr o código. O *type system* permite inferir dados automaticamente sendo uma das principais razões para o seu uso.

3.2.2 React.js

O *React.js* é uma biblioteca de *JavaScript* que traz muitas vantagens no desenvolvimento da *User interface* (UI). O *React.js* é um projeto *open-source*, desenvolvido pelo Facebook [6].

A primeira vantagem desta biblioteca é o facto de ter uma pequena curva de aprendizagem. Esse é um dos motivos pelo qual é facilmente escolhida como tecnologia a usar num projeto, acabando mesmo por ser uma das mais usadas para desenvolvimento *web* [30]. O *React.js* constrói e usa o *Virtual Document Object Model* (DOM), que permite construir rapidamente uma aplicação *web*. Para o conseguir, o *Virtual* DOM compara os estados das componentes e só atualiza o DOM quando há alterações. Por fim, as componentes em *React.js* são construídas de forma a serem reutilizadas ao longo da aplicação. O *React.js* não é só utilizado para desenvolvimento *web*, pois existe a *framework React Native* que é utilizada no desenvolvimento de aplicações móveis [31].

3.2.3 Next.js

Next.js é uma *framework open-source* desenvolvida em *TypeScript* e *JavaScript*, que permite desenvolver aplicações *web* em *React.js*. Normalmente o *Next.js* gera previamente todas as páginas no lado do servidor, em vez de as gerar no lado do cliente quando há um

pedido. Esta estratégia permite ter uma melhor performance do SEO, melhor *User Experience* (UX) e usa menos recursos [32].

Esta *framework* procura fornecer a melhor experiência ao desenvolvedor de *software* e ter várias funcionalidades à sua disposição. Algumas destas funcionalidades são:

- Um sistema de rotas embutido que suporta rotas dinâmicas;
- *Code splitting* que efetua a separação de código em diferentes pacotes, útil para componentes como diálogos que raramente sejam usados;
- Fornece uma rápida atualização enquanto o desenvolvedor de *software* corre o projeto no ambiente de desenvolvimento;
- Fornece as imagens com o tamanho adaptado para o ecrã do dispositivo, melhorando assim a performance. Estas imagens só são carregadas quando o utilizador entra na janela de exibição do dispositivo, fazendo com que a página seja apresentada mais rapidamente ao utilizador.

3.2.4 Stale While Revalidate

Stale While Revalidate (SWR) é um *hook*[1] de *React.js* utilizado para a obtenção de dados dos pedidos [33]. Quando o cliente faz um pedido *HyperText Transfer Protocol* (HTTP), o SWR retorna dados que estão em cache apresentando uma resposta possivelmente desatualizada. Após retornar esses dados, faz um novo pedido, recebendo os dados atualizados para apresentar ao cliente. O SWR lida então com a cache, revalidação, rastreamento, intervalos entre pedidos, etc. O SWR traz diversas vantagens como uma UI mais rápida e uma melhor UX, suporta *Server-Side Render* (SSR) e *Static Site Generation* (SSG), retorna erros e o estado de carregamento, entre outras.

3.3 Ferramentas

Durante o desenvolvimento de software é comum usar ferramentas que permitem que este desenvolvimento seja mais rápido e/ou produtivo. Nas secções seguintes estão descritas algumas das principais ferramentas usadas durante o estágio.

[1] Os *hooks* são funções que nos permitem aceder e utilizar os diferentes estados, eventos e funcionalidade da aplicação *React.js*.

3.3.1 Visual Studio Code

O *Visual Studio* Code (VSCode) é um *Integrated Development Environment* (IDE) que não só permite visualizar o código de uma forma eficiente, como contém diversas integrações que facilitam o desenvolvimento de código, por exemplo, comandos de *Git* integrados, opções de *debugging*, etc [34]. Este IDE contém suporte para diversas linguagens como o *JavaScript*, *Python* [35], *Java* [36], entre outras e está disponível para *Windows*, *macOS* e *Linux*.

3.3.2 GitLab

O *GitLab* é uma ferramenta *web open-source* de colaboração e desenvolvimento de *software*. Esta ferramenta foi criada em 2011 e neste momento tem mais de 30 milhões de utilizadores registados e cerca de 1 milhão de utilizadores ativos. Uma das caraterísticas mais importantes desta ferramenta é que permite gerenciar diversos repositórios de *Git* [37].

O *Git* é um sistema de controlo de versões que permite rastrear as alterações feitas aos diversos ficheiros que compõem um projeto. Desta forma é possível coordenar o trabalho na equipa de forma que haja desenvolvimento de funcionalidades em simultâneo, sem que o trabalho feito numa tarefa afete outra tarefa.

O *GitLab* acrescenta ainda outras funcionalidades como documentação, o levantamento e rastreamento de problemas, entre outras. Esta ferramenta também facilita a automatização de tarefas, como correr testes, o gerenciamento do código base, através de revisões de código, e a monitorização e segurança de todo o repositório.

3.3.3 Jira

O *Jira* é uma ferramenta *web*, desenvolvida pela *Atlassian*, que auxilia a monitorização de tarefas e os seus estados, e facilita o gerenciamento de todas as atividades de um projeto [38]. O planeamento das tarefas é feito através da criação de tarefas que ajudam a especificar o desenvolvimento de uma determinada funcionalidade. Cada uma destas tarefas tem um estado associado que é representado, por exemplo, pela sua posição nas colunas de um quadro de planeamento. Alguns exemplos desses estados são “ToDo” (por começar), “In Progress” (em progresso), “In Review” (em revisão) e “Done” (completa). Existem ainda outros parâmetros associados às tarefas que permitem calcular métricas, como por exemplo, o esforço associado à execução de cada tarefa, o número de tarefas planeadas e concluídas num período

especificado, etc. Esta ferramenta é especialmente útil quando a metodologia adotada para o projeto é uma metodologia ágil como o *Scrum*.

3.3.4 Postman

O *Postman* é uma ferramenta de colaboração para o desenvolvimento de *Application Programming Interface* (API) [39]. Com esta ferramenta o utilizador pode facilmente criar, testar, partilhar e documentar APIs. Esta ferramenta permite então fazer pedidos HTTP e visualizar a sua resposta, facilitando a configuração dos cabeçalhos, construção do *Uniform Resource Locators* (URL), autenticação, configuração de parâmetros, entre outras caraterísticas. Esta aplicação está disponível para *Windows*, *macOS* e *Linux*.

3.3.5 Confluence

O *Confluence* é uma ferramenta de documentação *web*, desenvolvida pela *Atlassian* [40]. Com esta ferramenta é possível colaborar na criação de documentos e partilhar informação de uma forma eficiente. Estes documentos podem ser organizados por pastas e espaços, e estes têm diversas opções de formatação de texto e que podem incluir inúmeros elementos como figuras, tabelas, hiperligações.

Esta ferramenta contém ainda funcionalidades bastante úteis com a edição de documentos em tempo real, notificações de alterações de documentos, comentários e controlo de versões. Permite ainda a integração com outras plataformas externas.

3.3.6 Cypress

Cypress é uma ferramenta para criar testes de *frontend*, mais especificamente testes *end 2 end.* Esta ferramenta permite configurar projetos de testes, escrever testes, correr os testes e correr em modo *debug* esses mesmos testes. Muito utilizado por engenheiros de garantia de qualidade para testar aplicações *web* [41].

O *Cypress* destaca-se das outras ferramentas de testes *end 2 end*, pelas seguintes razões:

- Baixa curva de aprendizagem – pois este usa *JavaScript* e seletores de *Cascading Style Sheets* (CSS);
- Visualização do teste em tempo real – permite visualizar o teste a ser executado e interagir;

- Rápido – é bastante rápido em comparação com os seus competidores, pois os testes são escritos em *JavaScript* que é uma *framework* rápida a executar os testes.

3.3.7 Nightwatch

Nightwatch é uma ferramenta de testes *end 2 end* para aplicações *web* ou *websites*. É desenvolvido em *Node.js* e usa *Webdriver* para correr os testes no browser. O desenvolvimento de testes é fácil, uma vez que usa *JavaScript* e seletores CSS. Esta ferramenta é especialmente útil para testes a dispositivos móveis [42].

3.4 Marketplaces

Várias abordagens presentes neste projeto são inspiradas em outros *marketplaces* de sucesso. Um desses exemplos é o projeto *Airbnb* no qual os desenvolvedores criaram documentação em relação às melhores práticas de *JavaScript*, que se tornaram num ponto de referência para todos os projetos de desenvolvimento web em *JavaScript* com as suas 127 mil recomendações neste mesmo repositório [43].

Sendo o *Airbnb* um caso de sucesso, as tecnologias presentes no seu projeto são muitas vezes usadas para o desenvolvimento de novos projetos [44]. Este usa *React.js* pelas vantagens no desenvolvimento explicadas acima e *Hypernova* para realizar SSR de *JavaScript* [45]. Além disso, usa também *Redux* [46] para controlar os estados globais da aplicação e o *React Router* [47] para atingir o comportamento de uma *Single Page App* (SPA)[2] .

Outro exemplo de um *marketplace* com bons resultados é o *About You* [48]. Em 2017, este projeto atualizou e integrou novas ferramentas para melhorar o desenvolvimento e tornar o *website* mais estável [49]. Esta atualização envolveu diversas alterações entre elas a mudança para *TypeScript* para reforçar o uso de tipos, *React.js* pelas vantagens no desenvolvimento explicadas acima e *Redux* [46] para guardar e aceder aos diferentes estados da aplicação.

[2] O *Single Page App* é um design de *software* que reescreve os dados da página *web* corrente com novos dados provenientes do servidor sem requerer a atualização de toda a página.

A base das tecnologias usadas no projeto *KuwaitShop* é semelhante à usada nas aplicações do *Airbnb* e *About You*, o *React.js*. Por outro lado, nestes exemplos os problemas como SSR ou SPA são resolvidos com bibliotecas específicas, enquanto o projeto no qual trabalhei usou o *Next.js*, que já inclui a implementação dos conceitos de SSR e SPA na sua *framework*. A escolha do *Next.js* ainda não estava disponível quando as empresas descritas acima desenvolveram o seu produto, no entanto as bibliotecas utilizadas visavam resolver os mesmos problemas, que as funcionalidades do *Next.js* contém, acabando por utilizar os mesmos paradigmas.

4 Desenvolvimento

Neste capítulo será abordado e descrito todo o trabalho desenvolvido ao longo deste estágio.

4.1 Estado Inicial

Aquando da minha entrada na empresa, o projeto *KuwaitShop*, já se encontrava em desenvolvimento. Este começou a ser desenvolvido no dia 1 de julho de 2020. Já predispunha de uma metodologia escolhida até ao lançamento da aplicação e com uns eventos definidos. As escolhas das ferramentas já se encontravam definidas, assim como o uso do *Jira*, *Confluence*, *Gitlab* e *Contentful* [50].

As tecnologias associadas ao desenvolvimento de *frontend*, estavam igualmente definidas, nomeadamente o *TypeScript*, *React.js*, SWR. Toda a equipa já utilizava ferramentas como o VSCode para desenvolver o código e o *Postman* para testar pedidos.

Nesta fase, o projeto ainda se encontrava numa fase embrionária, contendo apenas as primeiras versões das páginas de listagem de produtos, da página de produtos e da autenticação.

4.1.1 Acolhimento no Projeto

A integração foi feita na equipa de desenvolvimento, orientado para o *frontend*, já composta anteriormente por quatro membros. Todos os membros da equipa se mostraram disponíveis para ajudar ou explicar qualquer dúvida com o projeto, criando um ambiente muito agradável e motivador entre a equipa.

O tempo de integração neste projeto começou no dia 23 de setembro de 2020 e foi estendido por um período de 3 dias. Este período de acolhimento permitiu não só conhecer a equipa e os eventos da equipa, assim como me deu uma formação relativamente ao cliente, ao produto que estava a ser desenvolvido e os seus requisitos. Também foi nesta fase que me foi dado acesso a todas as plataformas usadas no projeto, como o repositório do *Gitlab* e o projeto no *Jira* e em que me comecei a familiarizar com a base de código existente.

4.1.2 Planeamento

Os objetivos do projeto centram-se no desenvolvimento de uma aplicação *web* num curto espaço de tempo. Esta devia suportar um mercado online de muitas marcas e diferenciar-se da concorrência pela sua qualidade. Esta empresa procura destacar-se no Kuwait no sentido de, a médio e longo prazo, se expandir para os países vizinhos como Emirados Árabes Unidos e o Catar.

Para isto foi desenvolvido um plano onde se definiam as principais funcionalidades a ser incluídas no lançamento do MVP, que incluíam a autenticação, a concretização de transações e pagamentos, a lista de desejos e a página de listagem de produtos. Para que o processo de desenvolvimento fosse agilizado e a fim de conseguir entregar na data de lançamento do MVP, foi decidido que os erros que não fossem críticos ficariam no *backlog*, para serem posteriormente resolvidos nas duas últimas *Sprint*s.

O começo do desenvolvimento seguiu o princípio de *mobile-first*, isto advém do facto da utilização de dispositivos móveis ter crescido ao longo dos últimos anos com 56% do mercado superando os utilizadores de desktop (Figura 6), como tal a aplicação *web* é responsiva, ajustando-se aos ecrãs de diferentes dispositivos automaticamente, exibindo o conteúdo de forma a melhorar a UX [51].

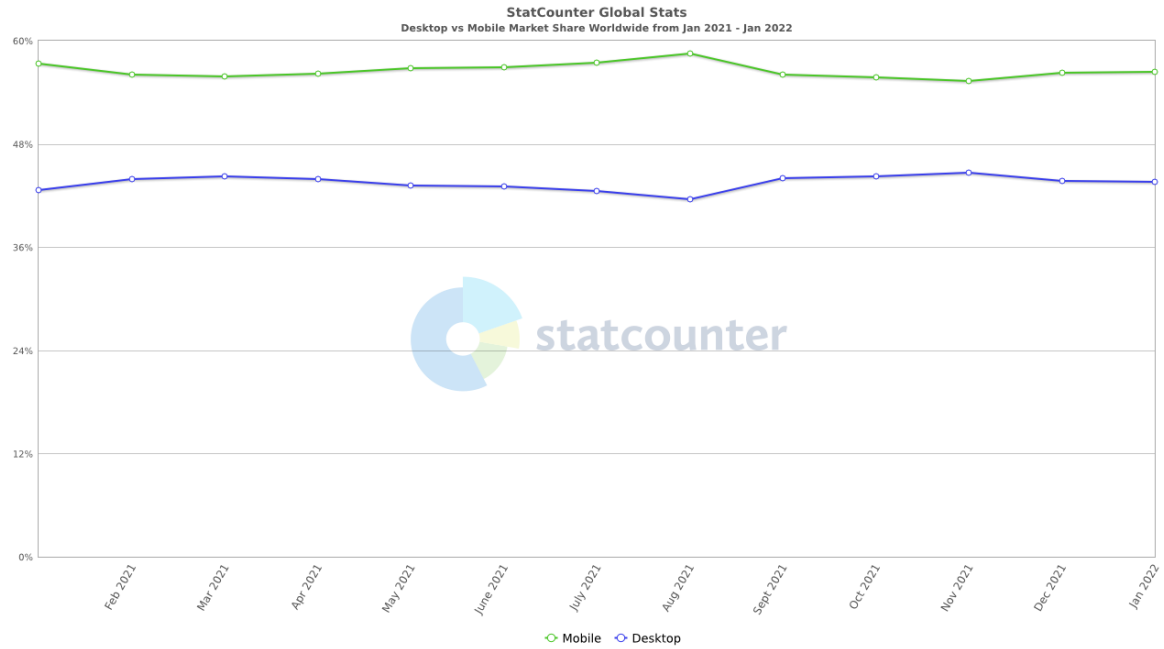


Figura 6 - Gráfico da quota do mercado por tipo de dispositivos (retirada do *Statcounter* [51])

Perante a existência de um estudo de mercado indicando que os utilizadores usariam maioritariamente dispositivos móveis, já estava planeada a adaptação para uma *Progressive Web App* (PWA), isto permitia que a aplicação funcione em modo offline através da memória cache dos dispositivos móveis e dos *service workers*, concedendo algumas caraterísticas das aplicações nativas a dispositivos móveis a este tipo de aplicações *web*.

Posteriormente uma aplicação nativa para dispositivos móveis também seria desenvolvida, para ter uma maior fluidez e funcionalidades nativas. A aplicação *web* não seria relegada, estando prevista a manutenção do site, correção de problemas e novas funcionalidades, como páginas exclusivas para cada marca.

4.1.2.1 Requisitos

No início do desenvolvimento do projeto *KuwaitShop* o cliente definiu os requisitos necessários para o *website*.

Como este era um *marketplace*, seria necessário haver uma página de listagem de produtos onde fosse possível aplicar filtros, como moda para homem ou mulher, para facilitar a procura dos itens e que contivesse paginação para não sobrecarregar o utilizador e a própria aplicação. Um outro tipo de página requerida pelo cliente era a página de *Stories*, que consistia numa listagem de artigos escritos pela equipa de Marketing que contém publicidade a um conjunto de produtos. Ao clicar em qualquer um destes artigos, o utilizador seria reencaminhado para a página da *Story* respetiva, onde poderia ler o artigo na integra.

No cabeçalho das páginas seriam incluídas outras funcionalidades como a área do utilizador, a lista de desejos, o *checkout* e a mudança de idioma. A área do utilizador permitia aceder os dados pessoais de um utilizador com a sessão iniciada ou permitia ao utilizador registar-se e autenticar-se no site. A lista de desejos é única para cada utilizador e permitia visualizar rapidamente os produtos favoritos adicionados pelo utilizador. O *checkout* permitia realizar o processo de compra que envolvia adicionar um produto ao saco de compras, inserir os dados de entrega da encomenda e finalizar a compra e estava integrado com uma forma de pagamento. A última funcionalidade do cabeçalho era um botão que permitia escolher a língua a ser aplicada no *website*, neste caso o utilizador poderia escolher entre inglês e árabe.

O rodapé das páginas incluía diversos botões que permitiam aceder de forma rápida a diversas páginas úteis. A página *About Us* fornecia informações sobre a empresa e a página

Contact Us disponibilizava uma forma de qualquer utilizador contactar um representante da empresa para ajudar a resolver algum problema. A página *Career* dava a conhecer as vagas de trabalho disponíveis na empresa. A página *Frequently Asked Questions* (FAQ) continha um conjunto de perguntas e respostas que informam os utilizadores de respostas a dúvidas frequentes. Por fim, a página *Terms and Conditions* continha as condições do contrato vinculativo entre a empresa e os seus utilizadores e a página *Privacy Policy* continha as políticas de privacidade de dados que eram necessárias para o cumprimento de diferentes legislações de privacidade.

A estrutura de navegação está relacionada com os requisitos, pois o objetivo é disponibilizar os acessos ao utilizador da forma mais simples possível. Na Figura 7 é possível visualizar a estrutura de navegação do website e de algumas funcionalidades que estariam disponíveis para o utilizador.

Por último, também era apresentada ao utilizador uma página inicial, que continha alguns produtos e *stories* previamente escolhidos pela equipa de Marketing.

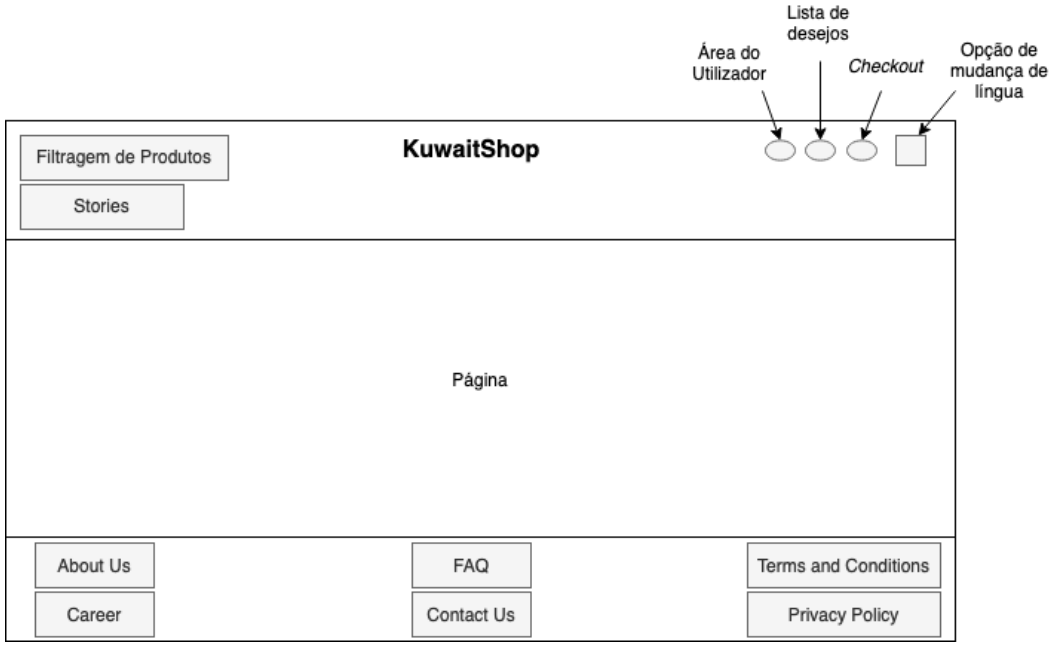


Figura 7 - Estrutura de navegação

4.1.2.2 Cronologia

Na Figura 8 encontram-se os principais marcos do processo de desenvolvimento do *marketplace*, durante o meu estágio.

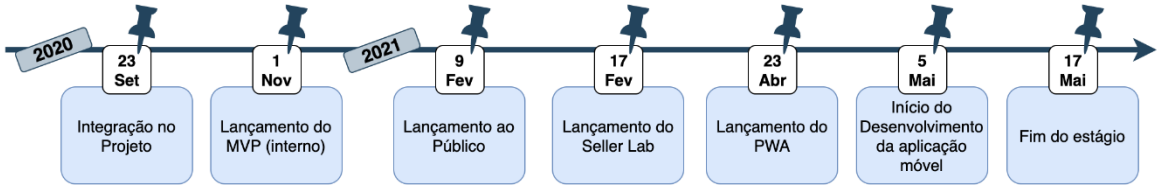


Figura 8 – Principais marcos do projeto

4.1.2.3 Fases de desenvolvimento

O lançamento da *KuwaitShop* estava marcado para o início de novembro e já existia um planeamento das principais tarefas a serem desenvolvidas, no entanto, estas funcionalidades estavam destinadas a serem resolvidas pelos membros da equipa com mais experiência, enquanto funcionalidades menos relevantes seriam destinadas para os novos membros da equipa.

Para isto foi desenvolvido um plano onde se definiam as principais funcionalidades a ser incluídas no lançamento do MVP, como foi explicado na secção dos requisitos. Para que o processo de desenvolvimento fosse agilizado e a fim de conseguir entregar na data de lançamento do MVP, foi decidido que os erros que não fossem críticos ficariam no *backlog*, para serem posteriormente resolvidos nas duas últimas *Sprint*s.

Até ao lançamento da aplicação *Web*, a metodologia estava orientada a agilizar o desenvolvimento e desbloquear algum problema na concretização de funcionalidades necessárias para o lançamento.

Após o lançamento do MVP interno foram encontrados alguns problemas, nomeadamente, no inventário e produtos incorretamente importados. Procedeu-se à correção destes e melhoramento do processo de integração de marcas e os seus produtos. O ritmo de desenvolvimento passou a ser alinhado mais com a metodologia *Scrum* e na procura de um desenvolvimento estável.

Após o lançamento para o público, manteve-se uma equipa para a manutenção da aplicação *Web*, na qual continuei inserido. Nesta fase foram igualmente criadas equipas para novas iniciativas como a criação do *Seller lab*, o desenvolvimento de uma PWA e o desenvolvimento de uma aplicação móvel nativa (Figura 8).

4.2 Arquitetura

O projeto *KuwaitShop* é composto por quatro módulos (Figura 9), cada um com a sua funcionalidade. Durante o meu estágio, trabalhei apenas no módulo *StoreFront*, que corresponde à plataforma *web* principal e contém como principais funcionalidades o login do utilizador, listagem de produtos e o *checkout*.

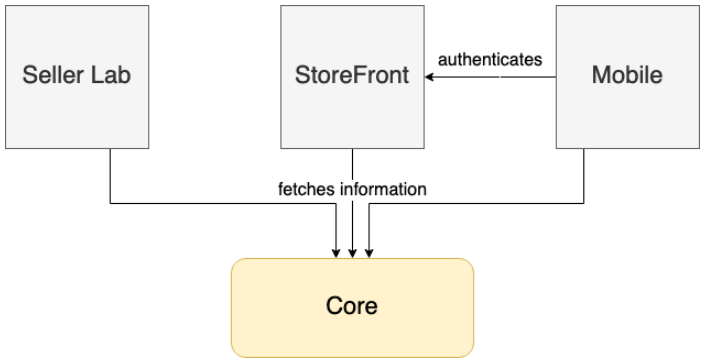


Figura 9 - Divisão do projeto em módulos

O módulo *Mobile* corresponde à aplicação móvel nativa que foi desenvolvida posteriormente para as plataformas *iOS* e *Android* como resposta à alta percentagem de utilizadores que acediam ao *marketplace* usando um *smartphone*, tirando melhor partido das funcionalidades nativas que não estão disponíveis na plataforma *web* que já tinha sido desenvolvida.

Além destes existiam ainda o módulo *Seller Lab*, que era responsável pela importação e exportação de produtos, a atualização de preços e o inventário das marcas presentes na plataforma, e o módulo *Core*, que contém a maior parte da lógica de negócio, desde a criação de promoções à listagem de produtos. Todas as aplicações como *Storefront, Seller lab* e Aplicação móvel conectam-se a este módulo para obter dados. O *Core* é responsável pelos produtos e a sua pesquisa, registos de compras e utilizadores.

Relativamente à infraestrutura do projeto, este foi desenvolvido em três ambientes diferentes, nomeadamente, *dev, staging* e *prod* e, para cada um destes, existe um *cluster* de *Kubernetes*, contendo 6 *namespaces* diferentes, representado pela Figura 10 [52]. Os *clusters* de *Kubernetes* são alojados pelos serviços do *Azure Cloud*.

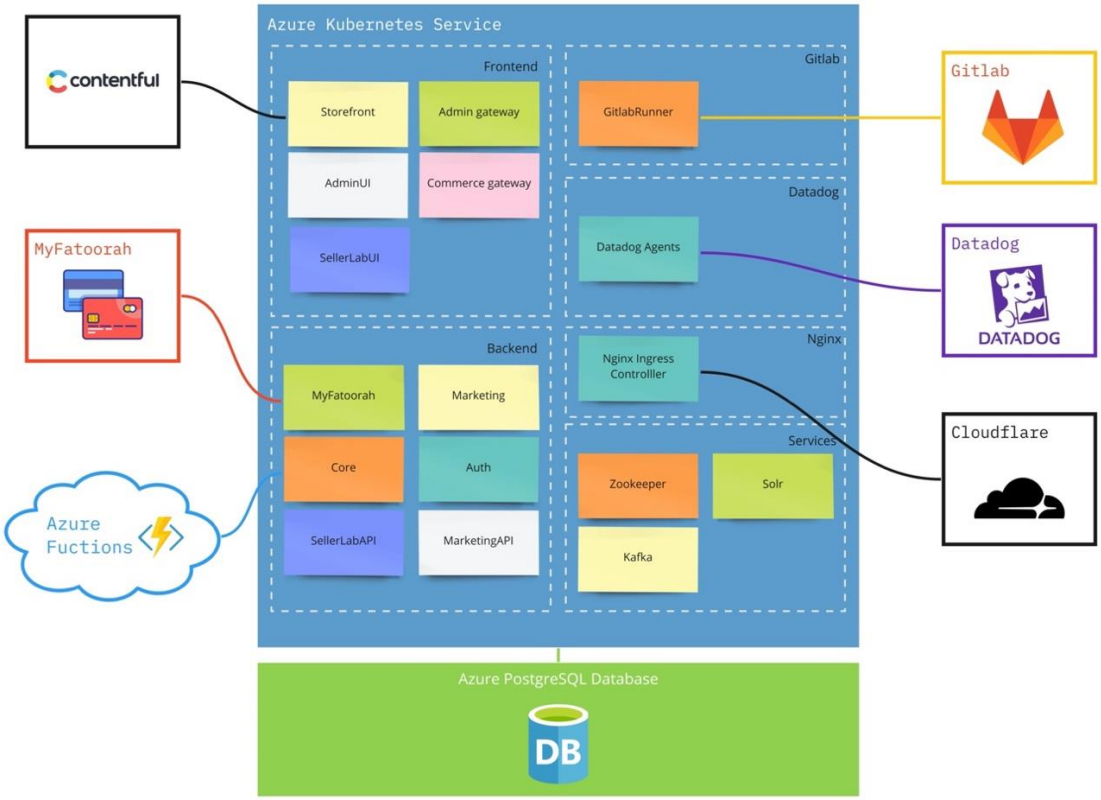


Figura 10 - Distribuição dos *namespaces* nos serviços de *Azure*

Kubernetes permite automatizar a implementação, o dimensionamento e o gerenciamento de aplicativos em ambientes virtuais isolados do sistema operativo (*containers*). No contexto de *Kubernetes*, os *namespaces* fornecem um mecanismo para agrupar e isolar *containers*, tendo cada um o seu âmbito e recursos exclusivos. Neste projeto cada *cluster* de Kubernetes contém os seguintes *namespaces*:

1. ***Frontend*** – Inclui todas as aplicações que interagem com o cliente e é onde correm os Gateways;
2. ***Backend*** – Contém a API, core, autenticação, marketing e os serviços de pagamento;
3. ***Services*** – Engloba os serviços secundários que suportam funções principais, como agentes de mensagens (*kafka* [53]), pesquisa de produtos (*solr* [54]) e configurações (*zookepper* [55]);
4. ***Datadog*** - Onde os agentes do *Datadog* [56] responsáveis por encaminhar as métricas e registos (*logs*) do sistema estão a ser executados;
5. ***Gitlab*** – Contém os agentes do *GitLab* que executam as tarefas de *Continuous Integration* (CI) / *Continuous Delivery* (CD);

6. ***Nginx*** - Executa o controlador de ingresso *nginx*.

Como o *StoreFront* corresponde ao website que interage com os utilizadores, este estava alojado no *namespace Frontend*.

4.2.1 Metodologias

Durante a realização deste projeto foram utilizadas as metodologias descritas no capítulo 3, no entanto, nenhuma delas foi seguida na totalidade, tendo-se optado por adaptá-las ao projeto e aos diversos momentos do desenvolvimento. Esta combinação de metodologias revelou-se uma mais-valia na organização e melhoria do método de desenvolvimento. Da mesma maneira que o constante envolvimento e partilha de opiniões da equipa no projeto, permitiu um enorme acréscimo na qualidade do resultado obtido.

Foram usadas várias práticas do XP. A fase de exploração envolveu a criação das tarefas com os requerimentos do cliente e na fase de planeamento, na qual participou o cliente, foram acordadas as tarefas com maior prioridade a serem desenvolvidas até à ao lançamento seguinte. Na fase de iteração foram aplicadas o *pair programming*, que foi utilizado para a integração na equipa e para ganhar contexto/conhecimento do projeto, o *refactoring*, utilizado em diversas situações em que o código presente não suportava a adição de uma nova funcionalidade ou aumentava a complexidade, e os testes unitários, foram também utilizados para aumentar a confiança na tarefa desenvolvida. A fase de produção, envolvia lançamentos pequenos e incrementais que foram complementados com o CI/CD. Após a finalização de uma tarefa, é feito um *merge* para o ambiente de *development*, posteriormente para *staging* e por último para *production*.

Neste projeto seguimos também a metodologia *Scrum*, com a utilização de diversos papéis desta metodologia. O *Product Manager* da equipa representou o papel de *Scrum Master* e de *Product Owner* e os restantes elementos de *backend* e de *frontend* corresponderam ao papel de desenvolvedores de *software*.

Os eventos de *Scrum* utilizados no desenvolvimento do projeto acabam por ser uma adaptação do seu original, ou seja, é aproveitado o conceito, porém são introduzidas algumas diferenças. As *Sprints* realizadas, na grande maioria das vezes, tiveram a duração de uma semana. Contudo esta duração de *Sprint* não foi fixa ao longo do processo de desenvolvimento, tendo sido estendida para duas semanas após o lançamento do MVP. O *Sprint Planning* ocorreu

sempre no dia anterior ao seu começo e focava-se no estabelecimento dos objetivos a realizar na *Sprint*, como indica a metodologia original. A *Sprint Retrospective* não foi realizada em todas as *Sprints*, no entanto existiu e ocorreu uma vez por mês e toda a equipa *Scrum* se encontrava presente. A *Daily Scrum* ocorreu todos os dias, ao longo do estágio. Os eventos de *Sprint Review*, estavam incorporados com o *Sprint Planning* onde era feita a análise do que foi entregue e o que iria continuar para a *Sprint* seguinte.

Adicionalmente, foi criada um encontro com todos os desenvolvedores de *software* de *frontend* de equipas diferentes, na qual se fazia um ponto de situação das tarefas que estavam a ser desenvolvidas, se partilhava conhecimento sobre novas tecnologias e ferramentas, se verificava se alguém estava com algum *blocker* já experienciado por outro desenvolvedor. Também foi criada uma reunião semanal com todos os trabalhadores do cliente, com o objetivo de introduzir os novos engenheiros contratados, informar todas as pessoas sobre o progresso e os resultados nas várias áreas como Produto, Engenharia, Marketing, Recursos Humanos e Operações, e por fim assegurar se existia algum obstáculo na concretização dos objetivos definidos.

Relativamente aos artefactos de *Scrum*, o *Product Backlog* (Figura 11) foi criado inicialmente, no entanto não foi finalizado uma vez que novas tarefas eram adicionadas pelo *Product Manager* à medida que outras tarefas eram concluídas, surgiam novos requisitos ou apareciam bugs durante a *Sprint*. Os outros artefactos também foram utilizados, sendo que o *Sprint Backlog* correspondia às funcionalidades que iram ser implementadas no decorrer da *Sprint*, e o *Increment* era entregue ao cliente ao fim de cada *Sprint*. Uma tarefa era considerada como terminada quando esta se encontrava em produção e corretamente testada.

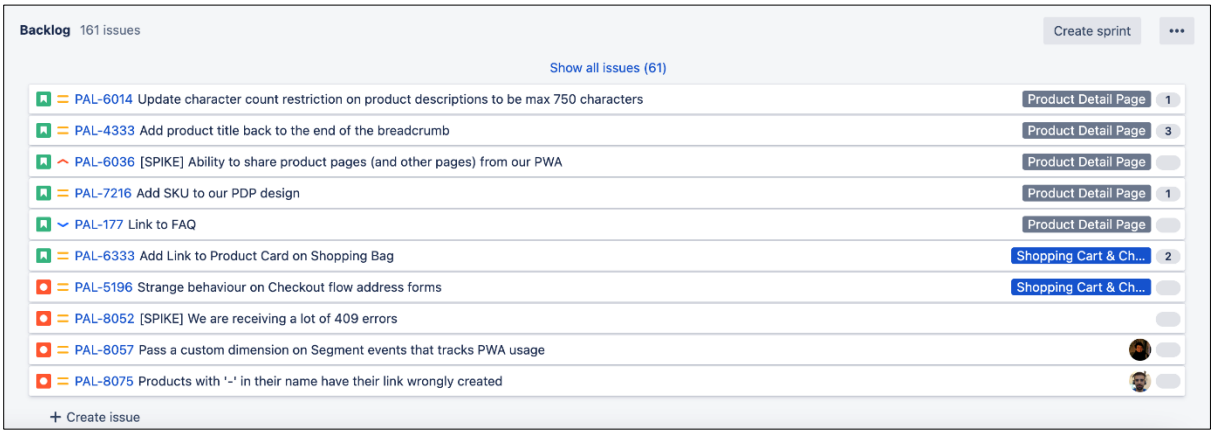


Figura 11 - Exemplo do *Backlog*

Os estados das tarefas da *Sprint* eram representados por um quadro composto por diversas colunas (Figura 12 e Figura 13):

- ***Reopened***: Corresponde às tarefas cuja implementação tenha sido testada em *develop* ou *staging* mas não passaram nos testes da equipa de *Quality Assurance* (QA);
- ***Todo***: As tarefas que estão no *Sprint Backlog* mas ainda não foram atribuídas;
- ***Blocked***: Indica que a realização da tarefa está de alguma forma impedida, por ter dependências de outra tarefa;
- ***In Progress***: Está a ser desenvolvido por um elemento da equipa;
- ***In Review***: A tarefa já se encontra realizada e um *merge request* está feito, faltando assim a revisão de pelo menos dois colegas de equipa, mantém-se neste estado mesmo quando à problemas em resolução encontrados no *merge request*;
- ***Dev***: Indica que a tarefa já foi implementada e encontra-se no ambiente de *develop*, pronta para ser testada;
- ***Translation***: Reporta que a tarefa está em desenvolvimento, no entanto não tem a sua tradução em árabe;
- ***Stage*:** Indica que a tarefa já se encontra no ambiente de *staging*, para ser novamente testada;
- ***Prod***: Indica que a tarefa se encontra em produção, onde se pode realizar o último teste;
- **Done**: Tarefa concluída e testada em produção.

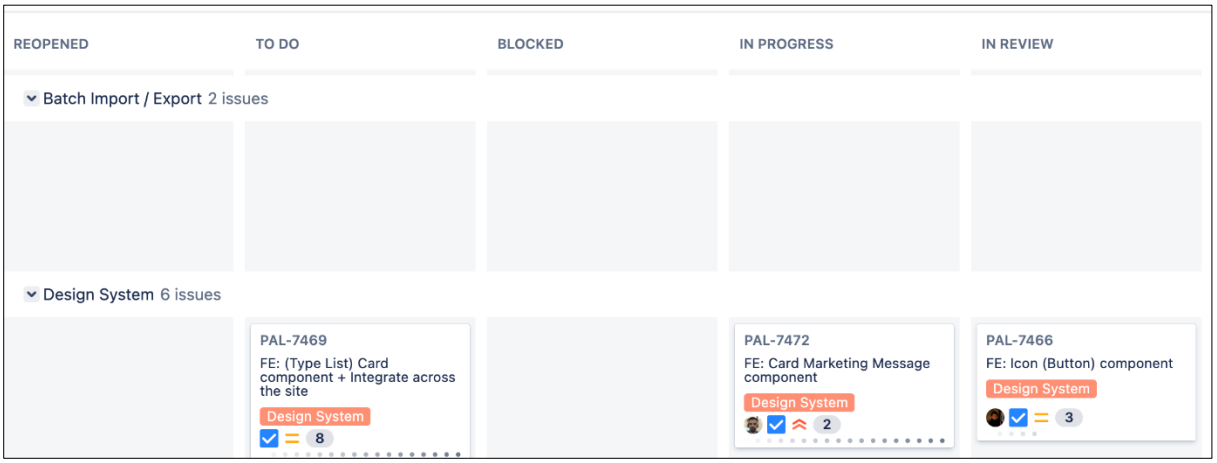


Figura 12 - Exemplo de um quadro de *Sprint*

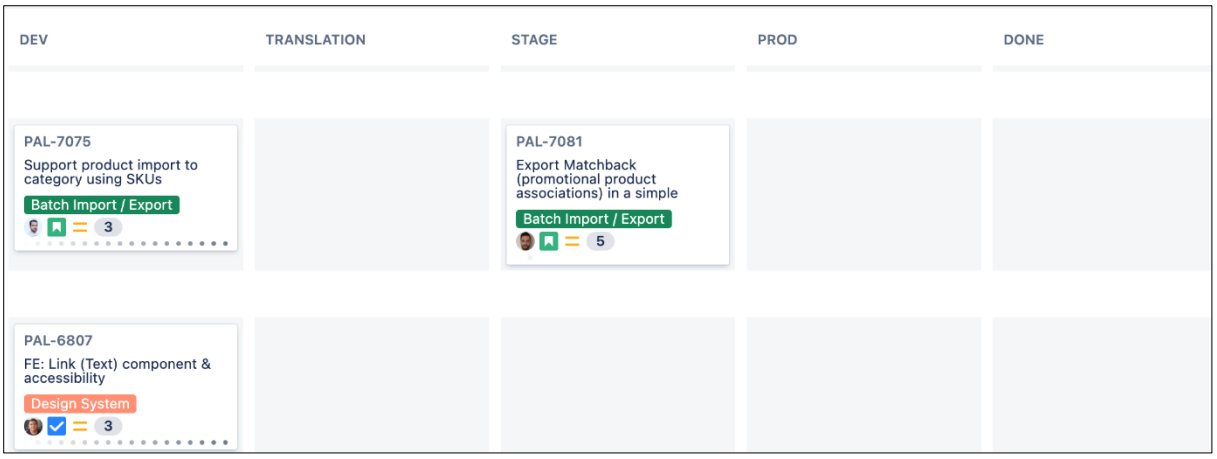


Figura 13 - Exemplo de um quadro de *Sprint* (continuação)

4.2.2 Estratégia de ramificação e controlo de versões

No projeto, é utilizada a ferramenta *Git* para controlo de versões. A estratégia utilizada foi *git feature branch workflow*, na qual o desenvolvimento é feito num ramo dedicado ao invés de usar o ramo principal. Para isso foram utilizados três ramos de desenvolvimento principais, *master*, *staging* e *develop* que correspondem aos respetivos ambientes. São também utilizadas convenções nos *commits* e nos ramos de desenvolvimento de novas funcionalidades.

Convenção de *commits*

Para criar um histórico de versionamento útil e legível, todos os desenvolvedores de *software* da equipa seguem uma convenção nas mensagens dos *commits* que segue a seguinte estrutura: tipo(escopo): tópico, como por exemplo, *feat(user-settings): add user preferences*.

- Tipo – O tipo de *commit* varia de acordo com o tipo da tarefa a ser realizada, tendo a seguinte lista de valores: *feat*, *fix*, *refactor*, *test* e *docs*;
- Escopo – O escopo é opcional e fornece mais contexto sobre a tarefa a ser realizada, exemplo '*user-settings*';
- Tópico – O tópico contém uma breve descrição informativa do *commit*, exemplo '*add user preferences*'.

Esta convenção foi aplicada pela biblioteca *husky* [57], esta ferramenta em conjunto com o *commitlint* [58] permite validar a mensagem do *commit.*

Convenção de ramos e estratégia

Como já referenciado, seguimos a estratégia de *git feature branch workflow*. Isto permite que múltiplos desenvolvedores de *software* estejam a trabalhar em tarefas separadas sem estarem a comprometer o código base comum a todos os desenvolvedores. Também assegura que o código presente em produção é estável, pois tendo vários ambientes de desenvolvimento como *develop* e *staging* permite testar as tarefas antes de chegar a *master*. Esta estratégia promove a colaboração entre a equipa através de *merge request* e *merge review*.

Convenção do nome de ramos

A convenção existente para nomear os ramos é composta por um prefixo que indica o tipo de ramo *feature*, *fix* e *hotfix*. Seguido pelo número da tarefa do *Jira*, isto permite facilmente associar as mudanças do ramo à tarefa, mantendo um bom histórico na navegação de *commits* nos ramos principais *develop*, *staging* e *master*, pois todos os *commits* são reduzidos para um quando são agregados para o ramo *develop*. Por fim termina com uma breve descrição da mudança com a alteração incorporada no ramo.

Exemplos do resultado da convenção de nomes de ramos feature/PAL-100-add-user-permissions; fix/PAL-101-product-variants; hotfix/PAL-102-correct-payment-methods.

A Tabela 1 apresenta diversas de regras:

- Não permite a implementação de código direto em *develop*, *staging* ou *master*, só pelo meio de *merge request*;
- Todo o código que é adicionado à base de código tem de ser revisto e aprovado por um ou mais membros da equipa de desenvolvimento, para isto a junção de *merge requests* requere de pelo menos uma aprovação;
- Só a equipa de QA e Infraestrutura têm permissões para realizar a agregação para os ambientes de *staging* e *master*.

Tabela 1 - Estratégia de Ramificação

Branches	Descrição	Target
<tipo-ramo>/<número do ticket><breve-descrição>	Feature branches contém o código da nova tarefa.	develop
develop	Isto representa o código a correr na estrutura de desenvolvimento.	staging
staging	Isto representa o código a correr na estrutura de QA.	master
master	Isto representa o código a correr em produção.	

Um exemplo da realização de uma tarefa com uma nova funcionalidade, passa por criar um ramo a partir do ramo de *develop*. Após o desenvolvimento do código e a realização de *commits* neste novo ramo, cria-se um *merge request* para *develop*. Este é revisto pela equipa de desenvolvimento e de seguida agregado para o ramo de *develop*. Uma vez que há uma mudança no ambiente de desenvolvimento, verifica-se se a funcionalidade opera corretamente num ambiente diferente do local. Por fim a equipa de QA ou Infraestrutura fica responsável pela agregação da funcionalidade para *staging* e consequentemente master.

4.3 Páginas estáticas

Existem seções do site que requerem ferramentas de *Content Management System* (CMS). Os sistemas CMS permitem que pessoas sem conhecimentos informáticos atualizem o conteúdo do site sem a necessidade de envolver a equipa de engenharia.

Como tal, foi feita uma análise ao mercado destas ferramentas para encontrar a que mais se adequa à necessidade do projeto em causa e tendo em conta que o tempo de desenvolvimento até ao lançamento do MVP era escasso. Muitas foram descartadas à partida por falharem em requerimentos cruciais como a simplicidade da integração e a customização dos modelos de conteúdo. No caso do *Ghost* [59] foi excluído por não permitir a customização de modelos de conteúdo, *Wordpress* [60] por os modelos de conteúdo serem difíceis de configurar, *Magnolia CMS* [61] devido a ser demasiado complexo para as nossas necessidades. Na Tabela 2 segue-se a análise aprofundada das ferramentas mais promissores para serem integradas neste projeto.

Tabela 2 - Análise de ferramentas CMS

	Directus	**Strapi**	**Contentful**
Tradução	Sim, mas tem de ser diretamente configurado, a modelagem dos dados tem de ser feita pelo utilizador	Existe alternativa para fazer a tradução	Sim
Suporte do *Workflow*	São contruídos na configuração do conteúdo	São contruídos na configuração do conteúdo	Sim
API / *Webhooks*	REST / GraphQL	REST / GraphQL	REST
Tecnologias	PHP	Node.js	-
SaaS / *Self Hosted*	Both	Both	SaaS
Modelagem de conteúdo do administrador	Sim	Sim	Sim
Pré-visualização	Não	Não	Sim
SEO *Support*	-	Suporta através da criação do modelo de conteúdos	Suporta através da criação do modelo de conteúdos
Headless	Sim	Sim	Sim
***Software* livre**	Sim	Sim	Não
Custo	Gratuito, pois é hospedado por nós	Gratuito, pois é hospedado por nós	Gratuito até 25 mil entradas de conteúdos
Vantagens	Além das funcionalidades de CMS também reproduz a base de dado SQL, permitindo uma modulação direta na UI	-	-
Desvantagens	A modelagem do conteúdo é mais complexa porque também gere as relações das bases de dados	Não suporta a tradução do conteúdo de forma direta	Os custos podem-se tornar-se dispendiosos se o projeto escalar É um SaaS Pode precisar de colocar em cache dados baseado na linguagem.

Os fatores que tiveram um maior peso na decisão foram a possibilidade de permitir traduções de diversas línguas já que à partida era um requisito, suportar a customização de SEO e metadados e o facto de poupar tempo e esforço por ser um *Software as a Service* (SaaS), pois assim não precisaríamos de implementar e manter o serviço, e permite começar a desenvolver rapidamente.

A empresa conta com uma equipa de editores de conteúdo (Marketing) para comercializar e editar produtos para vendas e marcas. Estes conteúdos podem ser encontrados na página inicial (*Home Page*), como nas páginas das *Stories*, entre outros.

O desenvolvimento de páginas estáticas foi propositadamente adiado para que engenheiros de *software* juniores as desenvolvessem. Nessa altura existiam outras prioridades, como o desenvolvimento da lista de desejos e o seguimento para o pagamento, sendo que as páginas estáticas, como a *Story* ou *About Us*, acabavam por não ser fundamentais para processo de *checkout* de um utilizador.

Estas páginas estáticas seguiram todas a mesma diretriz, com uma estrutura de página similar e uma lógica de desenvolvimento idêntica no que toca aos pedidos SSR realizados ao *Contentful*. A utilização do paradigma SSR melhorou o tempo de carregamento para a renderização da página inicial e garantiu um melhor SEO.

Foi no desenvolvimento destas páginas estáticas que tive o primeiro contacto com o *Contentful*, o que envolveu uma investigação à sua API para perceber os requisitos associados aos pedidos feitos a esta ferramenta. O primeiro passo é fazer a autenticação com o cliente do *Contentful* no projeto do *storefront* (projeto *frontend*) [62]. Esta autenticação deve ser incluída nos *headers* de todos os pedidos feitos ao *Contentful*, de forma a ter permissões para aceder aos dados.

Para facilitar este processo foi criada uma abstração de um *fetcher* associado ao *Contentful* (Figura 14). Este pode ser usado de forma global, para qualquer pedido ao *Contentful* e está preparado para fazer o pedido, tanto a uma entrada específica como a uma lista de entradas, independentemente do tipo do conteúdo que é pedido [63].

```
async function fetcher<T>(
  query: ContentfulRequestParameters
): Promise<ContentfulDataPage<T>>;
async function fetcher<T>(
  query: ContentfulRequestParameters,
  id: string
): Promise<T>;
async function fetcher<T>(
  query: ContentfulRequestParameters,
  id?: string
): Promise<T | ContentfulDataPage<T>> {
  const queryWithDefaults: ContentfulRequestParameters = {
    order: query.order || '-sys.createdAt',
    ...query,
  };

  try {
    if (id) {
      const { fields, sys } = await client.getEntry<T>(id, queryWithDefaults);
      return { ...fields, sys };
    }

    const { items, ...response } = await client.getEntries<T>(
      queryWithDefaults
    );
    return {
      ...response,
      items: items.map(({ fields, sys }) => {
        return { ...fields, sys };
      }),
    };
  } catch (ex) {
    Log.error(
      `Error fetching data from contentful (${id} ${JSON.stringify(
        query
      )}), due to:`,
      ex
    );
    throw new Error(ex);
  }
}
export default fetcher;
```

Figura 14 - *Fetcher* associado ao *Contentful*

Este *fetcher* também permite substituir as opções pré-definidas, como por exemplo a ordem pela qual os resultados estão organizados ou o limite de resultados do pedido. Este último parâmetro é especialmente útil para a paginação, por exemplo, e pode ser usado nos casos em que queremos mostrar apenas alguns resultados sendo que assim não se sobcarrega o utilizador e o pedido com um número maior de resultados do que é necessário.

Todo o conteúdo presente nas páginas estáticas suporta ambos os idiomas (inglês e árabe), sendo assim foi tirado proveito do *Contentful* para permitir o preenchimento do conteúdo nesses dois idiomas. A tradução de qualquer texto é inserida no *Contentful*, logo quando mandamos o pedido para obter a informação, este já vem como deve ser apresentado. Fica assim a cargo da equipa de marketing colocar a tradução dos textos dos botões, links, títulos, não sendo possível publicar sem tradução, pois é um requisito que foi definido no modelo de conteúdo.

4.3.1 Story

Uma *Story* permite à equipa de marketing criar pontos de discussão relevantes sobre a sazonalidade e novidades para envolver o utilizador, tentando destacar-se da concorrência. Uma *Story* é composta por produtos e conteúdo editado de forma atraente para os clientes.

Existem dois formatos que podem ser escolhidos para criar uma *Story*, ambos permitem a adição de parágrafos de informação, imagens e produtos, mas possuem uma estrutura diferente, especialmente na forma de apresentação dos produtos. No formato 1 (Figura 15) todos os produtos encontram-se agrupados na mesma listagem, e no formato 2 (Figura 16) apresenta-se vários subconjuntos de produtos ao longo da página. Como o conteúdo da *Story* tem uma estrutura diferente, foi separado em duas componentes, cada uma baseada no seu tipo no *Contentful*.

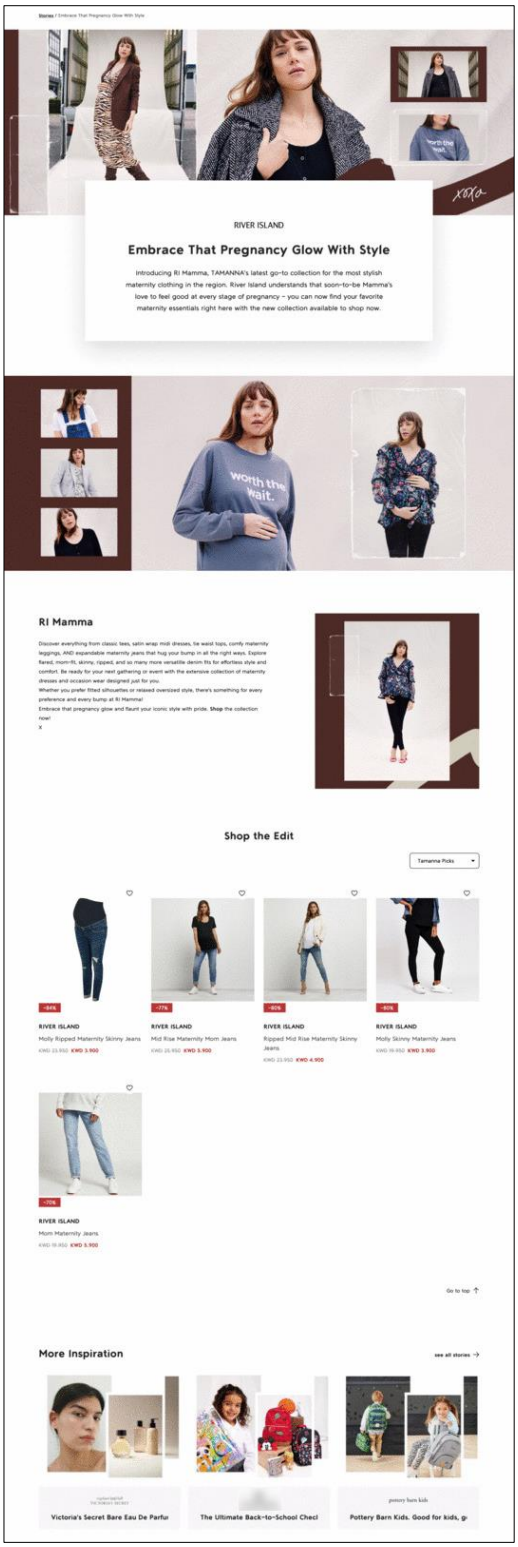


Figura 15 – Exemplo de uma página *Story* com o primeiro formato

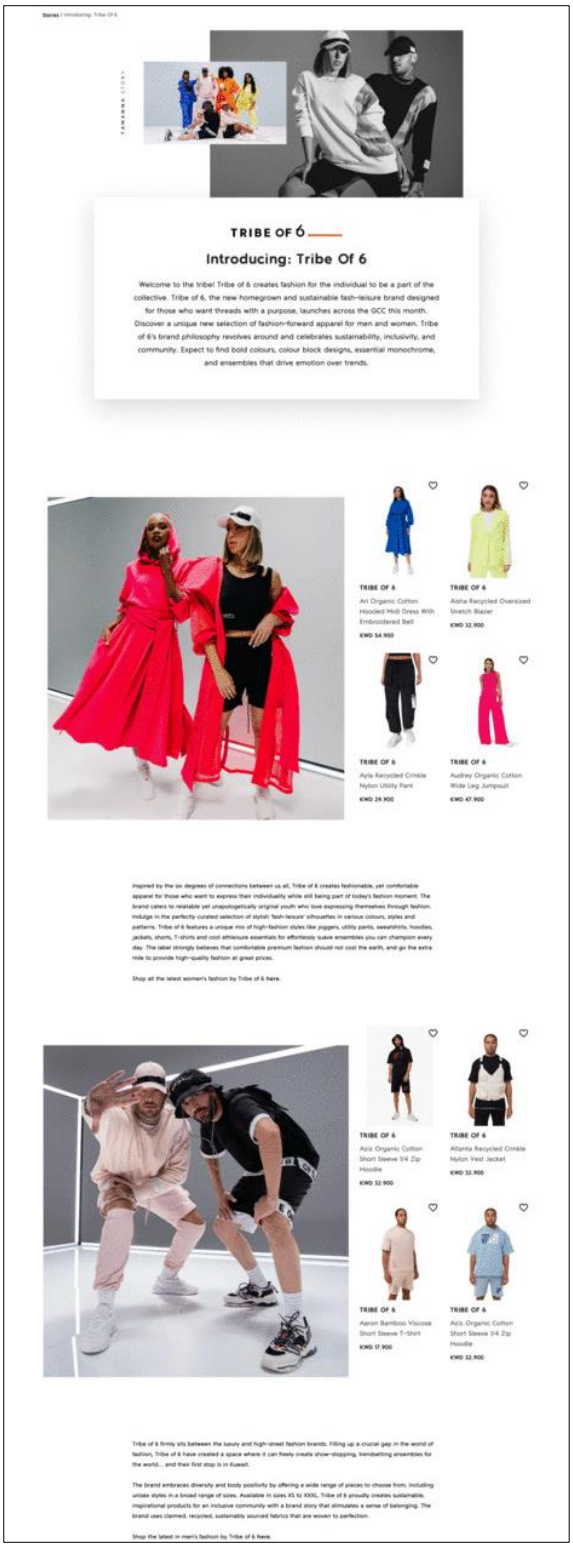


Figura 16 - Exemplo de uma página *Story* com o segundo formato

Quando um utilizador acede à página, é feito um pedido ao *Contentful* baseado no identificador presente no URL. Quando este pedido é enviado, a página não sabe qual é o tipo de formato que vai receber. Existe então um modelo de conteúdo que contém a informação comum aos dois modelos *Story Entry* e permite a inserção das informações específicas dos dois modelos, incluindo o conteúdo da página. Só na resposta é que é possível distinguir o tipo de formato baseado na informação que tem. Posto isto, no código referente a esta página, está presente os dois modelos, mas só um é que é aplicado.

O desenvolvimento desta página com o formato 1, no qual trabalhei, incluiu os seguintes critérios de aceitação:

- Composto por um *banner*, sumário, logótipo da marca;
- Uma componente de duas colunas, texto na primeira e imagens na segunda;
- Produtos relacionados/presentes na *Story*;
- Um carrossel no final da página contendo outras *stories*.

De forma a introduzir a *Story* foi requerido que cada uma contivesse um pequeno resumo e umas imagens ilustrativas. Foi tido em consideração o design da página, no entanto para páginas com um tamanho superior a 1200px foi decidido não esticar a imagem, atribuindo-lhe um tamanho máximo de largura com esse valor e aplicando preenchimento (*padding*) à volta dela de forma a centrar sempre toda a informação da página. Este módulo de introdução está presente nos dois modelos, como tal, foi retirado para estar presente na página e não nos componentes, como tal encontra-se presente no modelo de conteúdo *Story Entry*.

Posteriormente para melhorar o nosso SEO, em vez do URL de cada *Story* conter o seu identificador, foi substituído por um novo campo *slug*. Este *slug* corresponde ao título sem caracteres especiais, por exemplo *november-sale-our-biggest-of-the-year*.

Isto implicou um *refactor* no código, uma vez que o identificador estava presente no URL e a partir deste fazíamos o pedido ao *Contentful*. Por esta razão, foi preciso alterar o pedido de forma que este filtrasse toda a listagem de conteúdos *Story* pelo campo *slug*, e devolvesse o correto. Para garantir que só era retornado um resultado foi adicionado um limite de resultados ao pedido. No *Contentful* foi necessário adicionar este campo *slug* ao modelo da *Story Entry*, com caráter obrigatório e com uma validação a impedir a inserção de carateres especiais, para certificar que este *slug* é corretamente criado.

4.3.2 Stories

A página *Stories* contém uma listagem de *stories*, que permite o redireccionamento à *Story* completa, esta corresponde a um artigo/blog escrito pela equipa de Marketing que contém o conteúdo de produtos expostos de forma atraente para destacar esses mesmos produtos ou marcas. O conteúdo da página é maioritariamente constituído por conteúdo do CMS, e este é adicionado/escrito por parte da equipa de marketing. Durante o desenvolvimento destas páginas, e para efeitos de testes, foi necessário criar *stories* como *dummy data* para testar as diversas funcionalidades.

Esta tarefa continha os seguintes critérios de aceitação:

- Criação da página *stories*;
- Conter uma forma de paginação;
- Suportar a tradução de texto;
- Respeitar o design em dispositivos móveis;
- Conter um redireccionamento para outra página.

No que diz respeito à paginação, a escolha foi uma adaptação do *Infinite Scrolling*[3]. Neste caso o pedido de mais resultados é realizado com recurso a um botão "Load More". Desta forma, o conteúdo não é carregado automaticamente e necessita de uma ação do utilizador para ser feito o pedido para obter mais *stories*. Como é expectável que o utilizador queira ter acesso às *stories* recentes, não fazia sentido fazer uma paginação mais complexa que permitisse, por exemplo, aceder facilmente às *stories* mais antigas.

A Figura 17 apresenta o esquema da lógica do botão *Load More* que permitia a visualização de mais conteúdo. Primeiramente era realizado um pedido ao *Contentful* de um determinado número de itens e o valor do *skip* era incrementado. O *skip* registava o número de itens devolvidos, facilitando a identificação do conteúdo a apresentar nos próximos pedidos. O botão de *Load More* só era então visível se o valor de *skip* fosse menor que o número total de itens.

[3] O *Infinite Scrooling* é uma funcionalidade que permite carregar automaticamente mais conteúdo à medida que o utilizador faz *scroll down*.

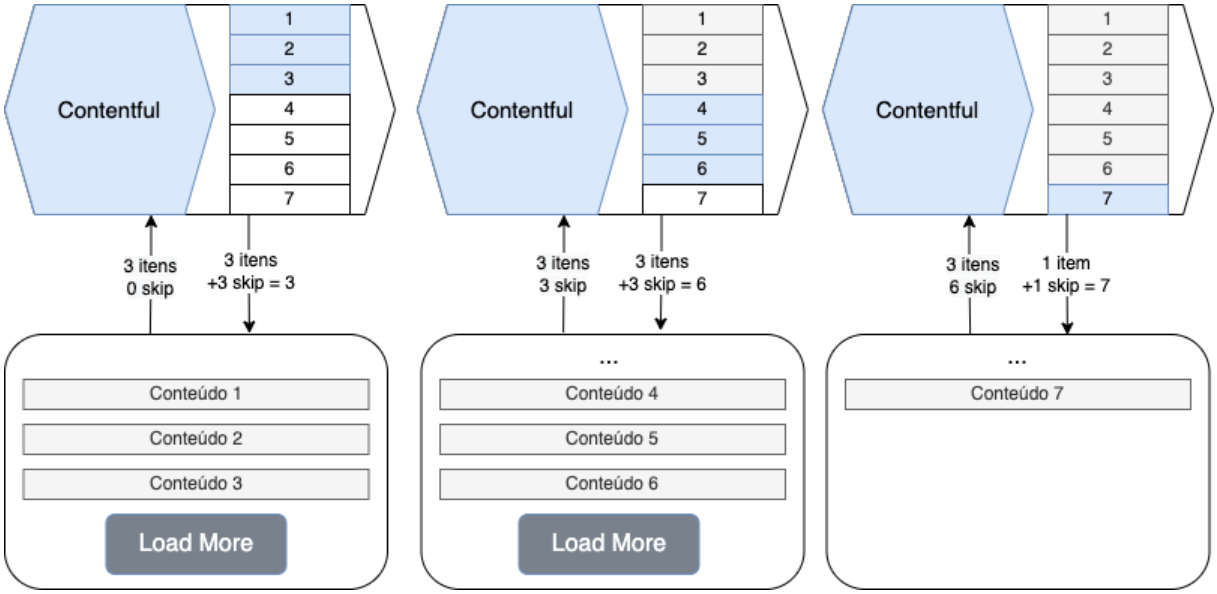


Figura 17 - Esquema da paginação aplicada no *Load More*

A página em modo dispositivo móvel, contém o mesmo conteúdo, no entanto a forma como a *Story* é exibida difere, para isto na componente relacionada com a exibição do conteúdo da *Story*, tem a orientação alterada consoante se é um dispositivo móvel ou ecrã maior.

Cada *Story* tem de conter o redireccionamento para uma página com a sua versão completa. Inicialmente como a página da *Story* não existia, foi utilizado outro valor só para testar a sua funcionalidade, posteriormente foi adicionada a rota correta mais o identificador da mesma, correspondendo ao URL correto.

Para permitir o preenchimento do conteúdo no CMS, foi necessário criar modelos no mesmo. Começando pelo *Story Entry* contendo toda a informação necessária, título, imagem da *Story*, logótipo da marca e sumário do conteúdo, dos quais todos são campos obrigatórios. Outros cuidados foram tidos em conta para melhorar a experiência do utilizador, como a adição de um *spinner* no botão "*Load more*".

Um requisito posteriormente solicitado, foi apresentar uma mensagem a indicar da ausência de *stories*, caso não haja nenhuma *Story*, ou caso haja algum problema com o CMS a página tem de estar preparada a não apresentar dados. Isto foi resolvido verificando a existência de resultados no primeiro pedido das *stories* ao CMS. Esta mensagem é adaptada para os dois idiomas do projeto.

Posteriormente, surgiu um novo requisito devido à similaridade dos termos "*Stories*" e "*Stores*". No qual foram pedidos os seguintes critérios de aceitação:

- A rota da página deixa de ser */stories* e passa a ser */inspiration*;
- Manter um código de estado HTTP 301[4] que redirecione */stories* para */inspiration*.

A alteração da página já requereu mudar o nome da pasta das *stories* no projeto e atualizar a rota da página *stories* e *Story*, normalmente usadas para botões *Click To Action* (CTA). Este redireccionamento pode ser facilmente implementado, adicionando à configuração do *Next.js* o redireccionamento da rota '/stories' para '/inspiration' e também '/stories/:*Story*Id' para '/inspiration/:*Story*Id'. Precavendo, para o caso de promoções com referência para *stories*, ou um URL em específico, que continue válido e com o correto redireccionamento. A Figura 18 e Figura 19 mostram o resultado inicial da página.

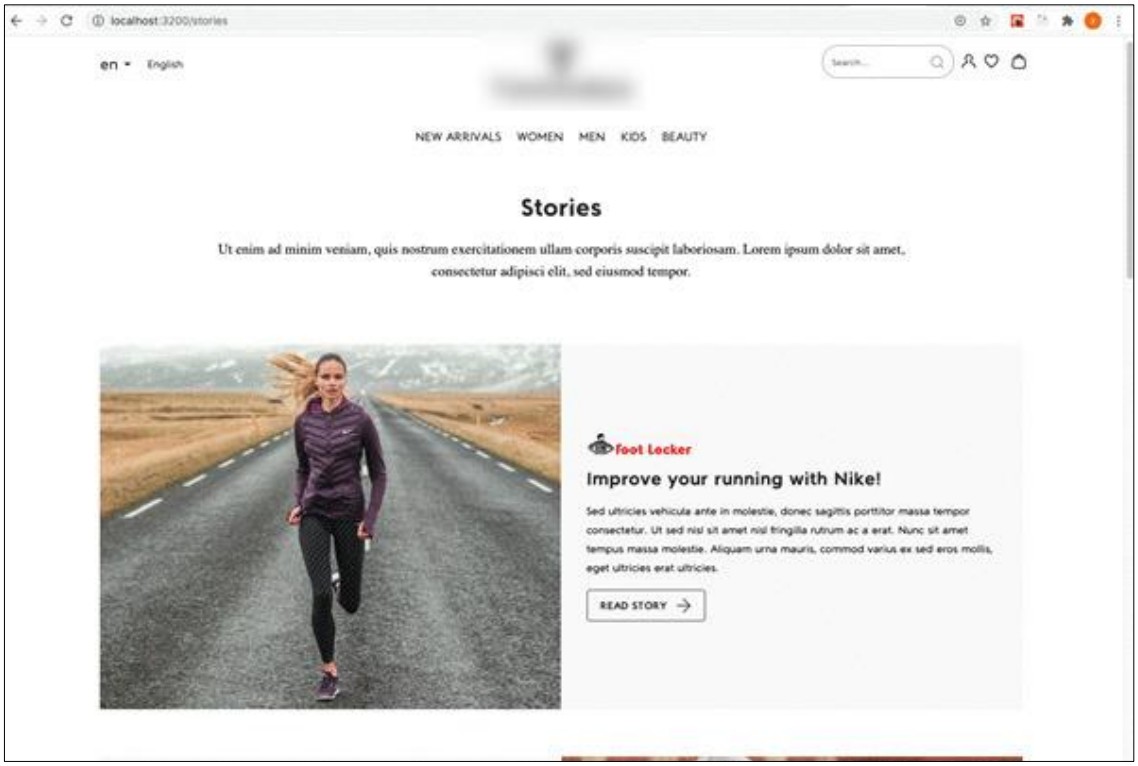


Figura 18 - Resultado inicial da página *stories*

[4] Os códigos de estado HTTP 300s indicam um redireccionamento, ou seja, que um novo recurso substituiu o solicitado. Mais especificamente, o código 301 indica que quando essa página é solicitada, o utilizador é redirecionado para outra.

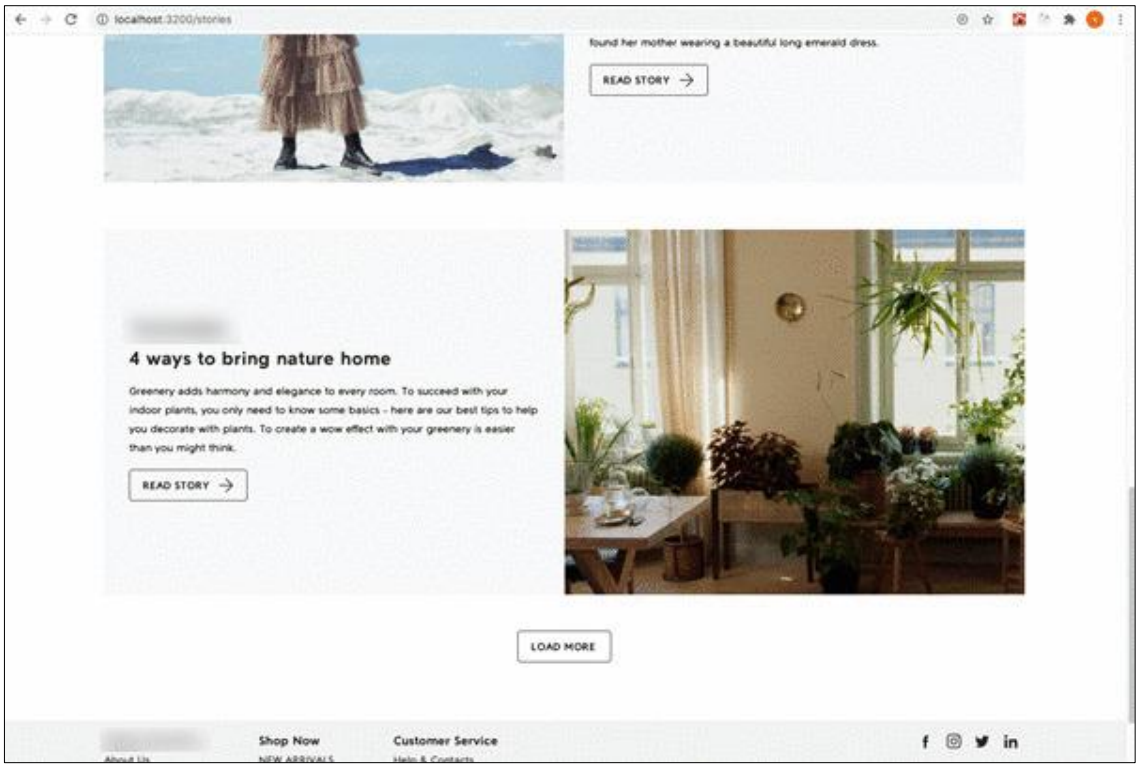


Figura 19 - Resultado inicial da página *stories* (continuação)

4.3.3 Página Contact us

Esta tarefa tinha o objetivo de criar um formulário para permitir que qualquer cliente pudesse entrar em contacto com a empresa, para levantarem questões e obter ajuda, como por exemplo, questões sobre devoluções ou problemas de pagamento. Para isto, o formulário necessitava de ter os seguintes campos: nome, email, tópico da questão, número de telemóvel e mensagem. Em adição a este formulário, também é disponibilizado um número de contacto (Figura 20). Todos os campos deste formulário eram de preenchimento obrigatório, por isso foi adicionada uma lógica de validação de preenchimento a cada um deles. Caso houvesse um preenchimento incorreto de algum campo, uma mensagem de erro aparecia para ajudar o utilizador a completar a sua submissão de forma correta.

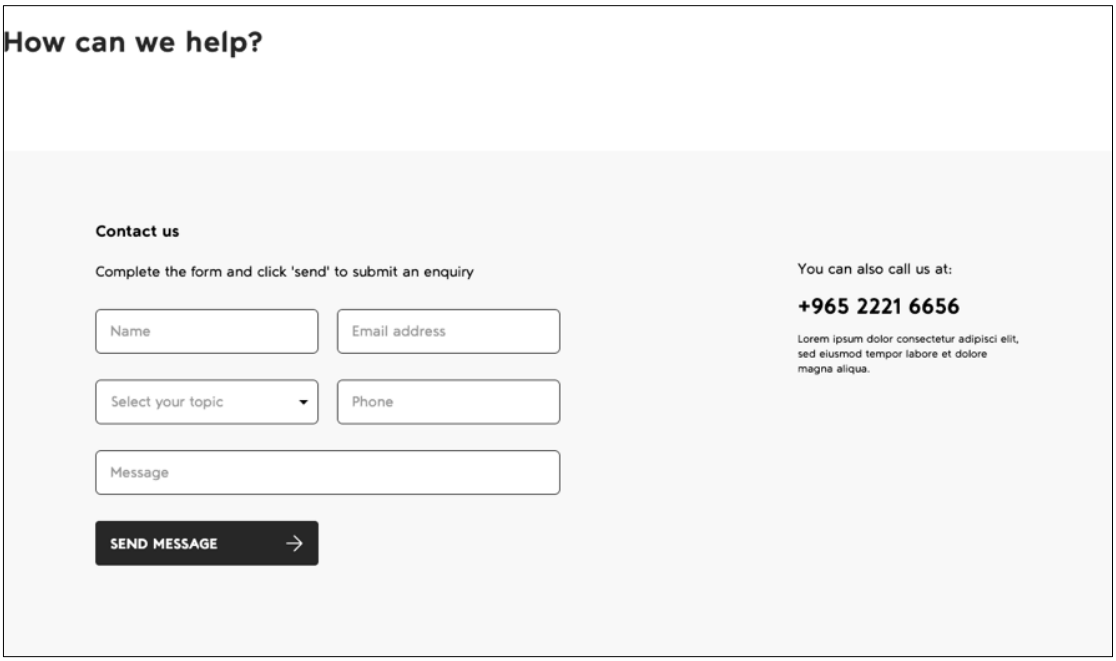


Figura 20 - Formulário *Contact us*

As componentes foram desenvolvidas de forma responsiva. O formulário apresentado na Figura 21 apresenta a estrutura associada a um dispositivo móvel, sendo a sua orientação somente vertical, contendo um campo do formulário por linha.

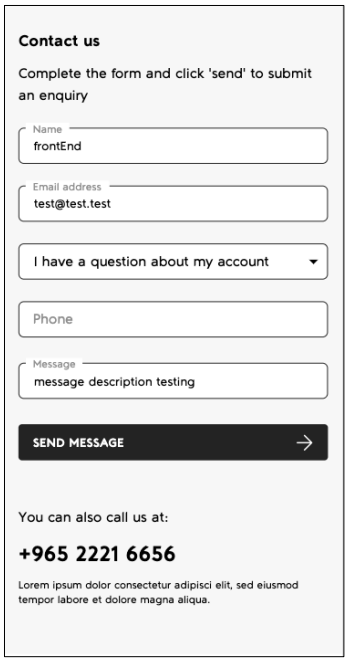


Figura 21 - Formulário *Contact us* para dipositivos móveis

Para gerir a informação enviada através destes formulários de forma automática, utilizou-se o *SalesForce* [64] que é uma plataforma de *Customer Relationship Management* (CRM)[5]. Quando uma submissão é bem-sucedida, o serviço que integra com o *SalesForce* faz um *redirect* de volta ao *marketplace*. Para estes casos, foi adicionado um parâmetro "sucess=true" ao *url* para diferenciar e avisar o utilizador que a submissão foi efetuada com sucesso. Perante a existência deste parâmetro é então exibida uma mensagem de confirmação no topo da página *Contact us* (Figura 22).

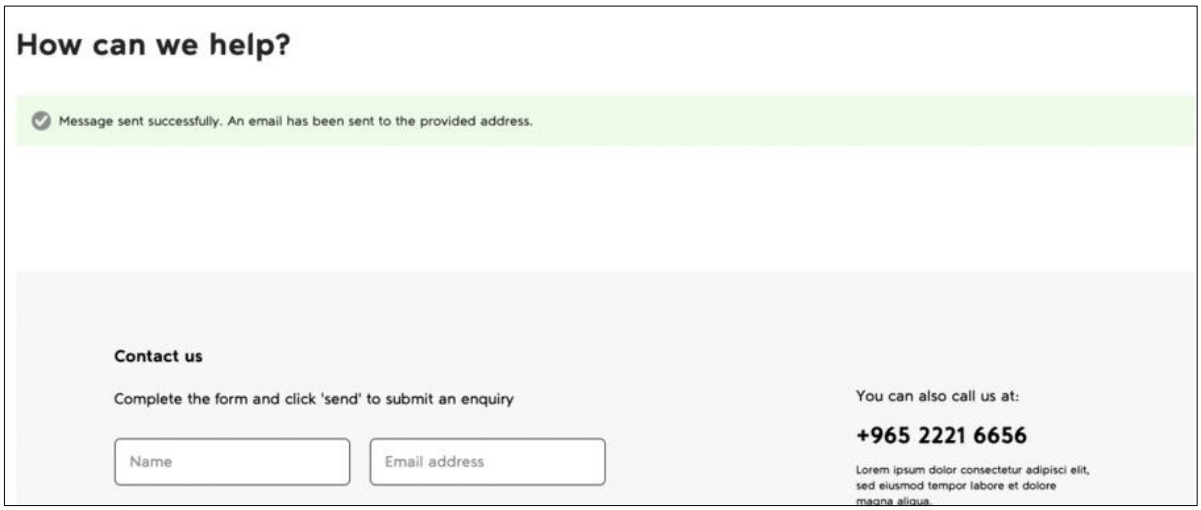


Figura 22- Mensagem de confirmação da submissão do formulário

Por fim foi adicionado um link no rodapé (Figura 23) que se encontra presente em todas as páginas que redireciona o utilizador para a página *Contact Us*. Desta forma, a possibilidade de visitar esta página está sempre disponível.

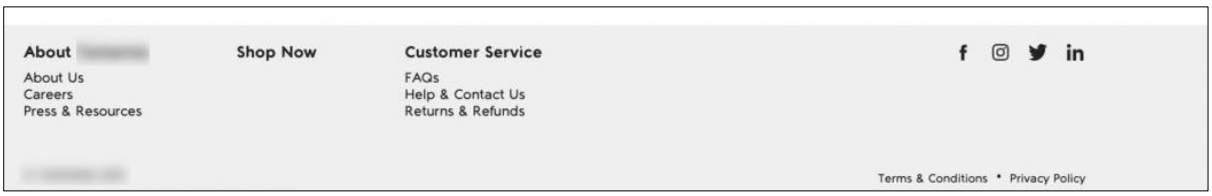


Figura 23 - Rodapé do site

[5] O CRM é a combinação de práticas, estratégias e tecnologias que são utilizadas para analisar e facilitar as interações com os utilizadores.

4.3.4 Página About Us

A finalidade desta tarefa era criar uma página para dar a conhecer aos utilizadores informações sobre a empresa. A página incluía informação sobre as origens da empresa, onde estão localizados os seus escritórios e os seus objetivos (Figura 24).

Figura 24 - Versão inicial da página *About Us*

Em relação ao preenchimento do conteúdo desta página, foi decidido que deveria ser dinâmico e facilmente alterado sem estar dependente de um desenvolvedor de *software*.
Para tal foi incorporada a informação do *Contentful* para preencher a página, tornando esta informação dinâmica e dando assim a responsabilidade pelo seu conteúdo à equipa de marketing.

No que diz respeito ao *Contentful*, foi criado um modelo constituído por um título (da página) e uma descrição, que poderia ter uma grande quantidade de caracteres. O texto destes campos podia ser formatado com *markdown*, ou seja, era possível adicionar títulos em itálico, negrito, ou outras formatações que respeitem a sintaxe de *markdown* em texto (Figura 25). No código foi preciso adicionar uma forma de aplicar a formatação ao *markdown* embutido no texto, pois de outra forma seria tudo interpretado como texto.

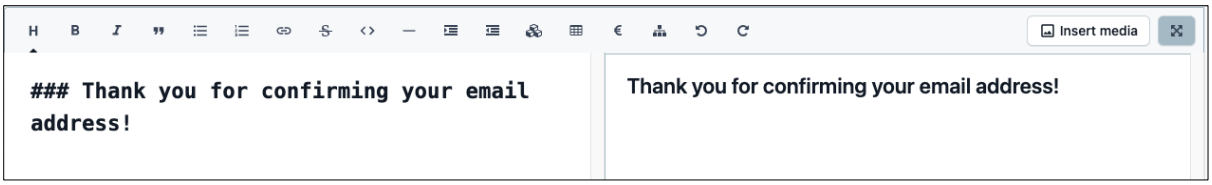


Figura 25 - Exemplo de *markdown* de um título

À semelhança da página anterior, foi igualmente adicionado um link ao rodapé que redireciona para a página *About Us* (Figura 23).

4.3.5 Página Terms and Conditions

A finalidade desta tarefa era criar uma página com os termos e as condições que permite estabelecer um contrato vinculativo entre a empresa e os seus clientes (Figura 26).

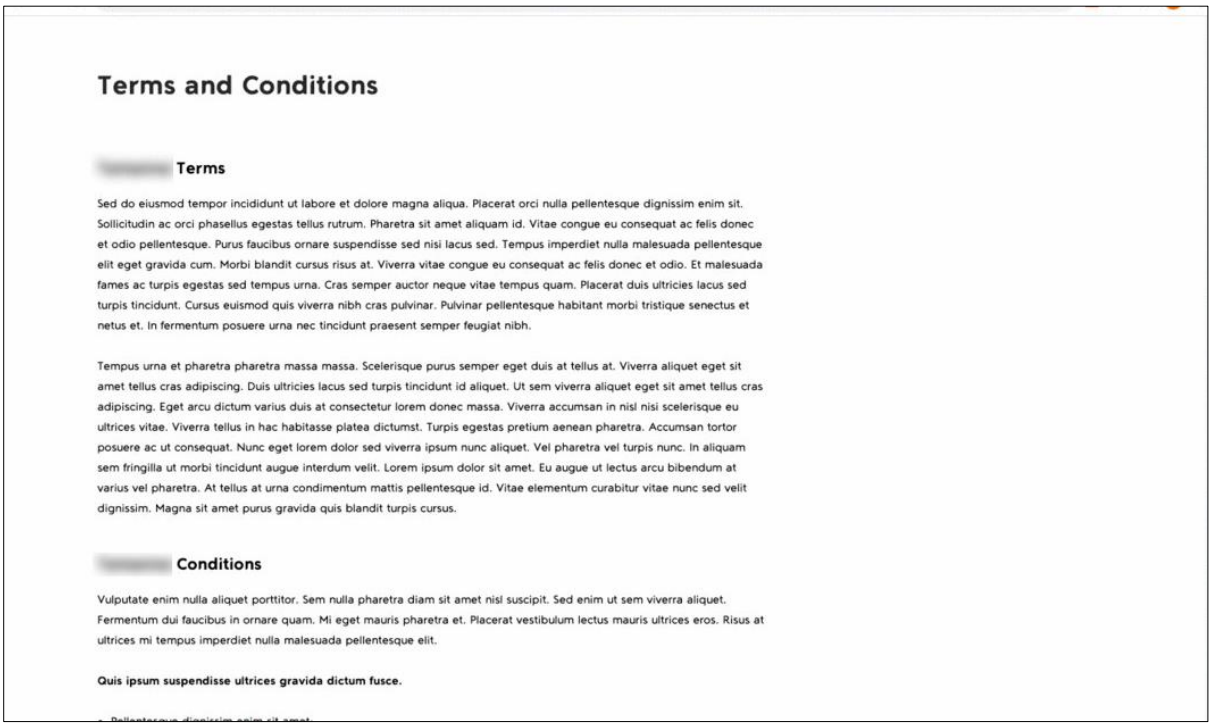


Figura 26 - Versão inicial da página *Terms and Conditions*

O preenchimento do conteúdo desta página, também é dinâmico utilizando a informação do *Contentful* para completar a página. O modelo utilizado para esta página foi o mesmo que o utilizado no *About Us*. Para isto foi adicionado um tipo para filtrar qual conteúdo corresponde a cada página, tendo esta página o tipo de "*Terms Conditions*" (Figura 27). A página contém texto em *markdown* e é assim constituído o seu conteúdo. Não foi necessária fazer uma adaptação para dispositivo móvel desta página.

```
const response = await contentfulFetcher<ContentfulPageContent>({
      content_type: CTYPE_LEGAL_AND_INFO_PAGES,
      locale,
      'fields.type': 'Terms Conditions',
    });
```

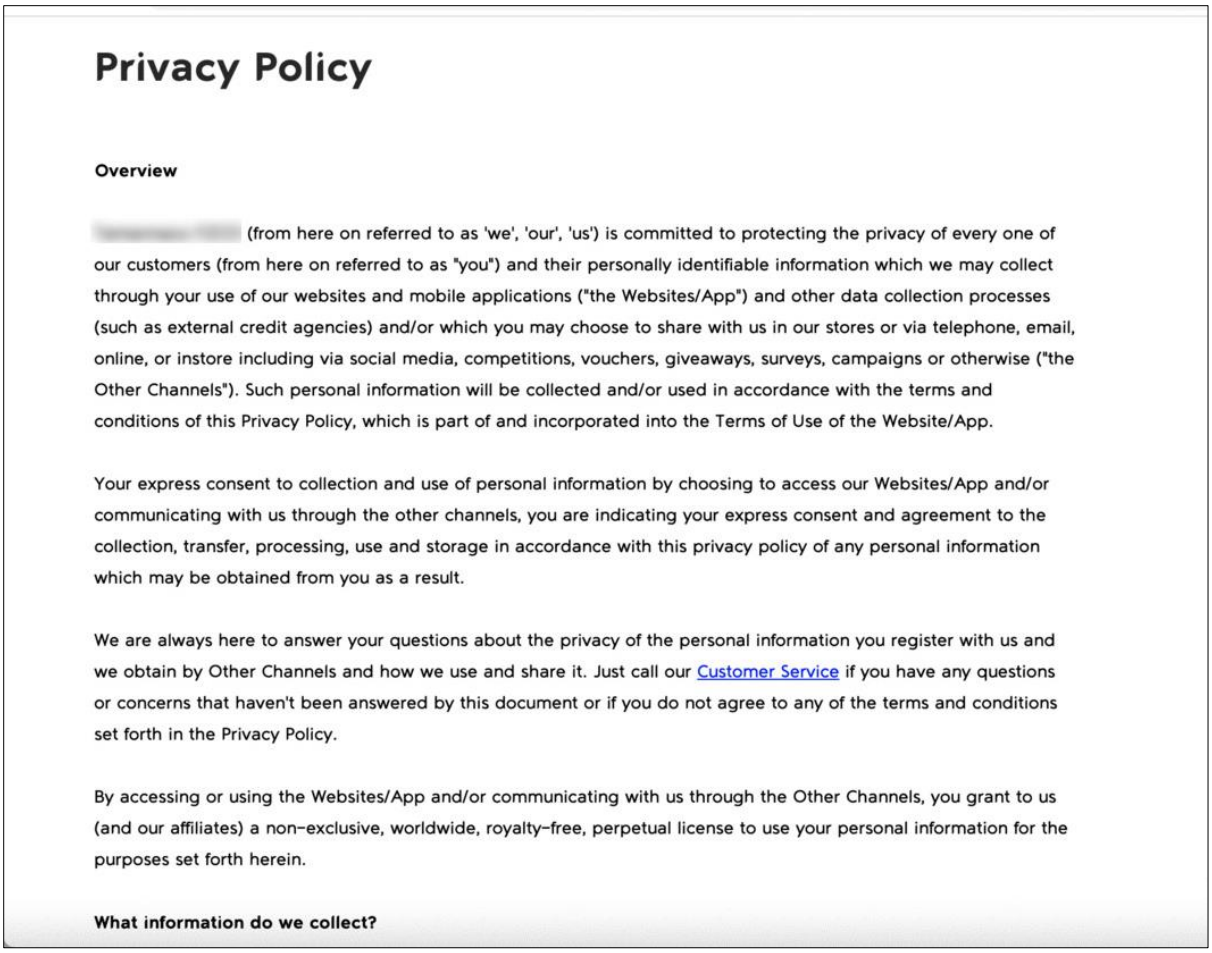
Figura 27 - Aplicação do filtro consoante o tipo *Terms Conditions* no modelo

À semelhança das páginas anteriores, também foi adicionado um link ao rodapé que redireciona para a página Termos e condições (Figura 23).

4.3.6 Página Privacy Policy

O objetivo desta tarefa era criar uma página com as políticas de privacidade de dados que são necessárias para o cumprimento de diferentes legislações de privacidade e as políticas de privacidade de dados, como o Regulamento Geral de Proteção de Dados (RGPD), contribuindo igualmente para definir as expectativas dos clientes em relação ao site. Portanto existe a necessidade de clarificar como a empresa usa e gere os dados do utilizador.

O preenchimento do conteúdo desta página (Figura 28) é semelhante ao das páginas anteriores e utiliza o mesmo modelo que o utilizado na *About Us* e *Terms and Conditions*, adicionado o tipo de "*Privacy Policy*" para filtrar qual conteúdo corresponde à página. Também não foi necessária fazer uma adaptação para dispositivo móvel desta página. O link que redireciona para a página Políticas de Privacidade foi também adicionado ao rodapé da página (Figura 23).



Privacy Policy

Overview

[redacted] (from here on referred to as 'we', 'our', 'us') is committed to protecting the privacy of every one of our customers (from here on referred to as "you") and their personally identifiable information which we may collect through your use of our websites and mobile applications ("the Websites/App") and other data collection processes (such as external credit agencies) and/or which you may choose to share with us in our stores or via telephone, email, online, or instore including via social media, competitions, vouchers, giveaways, surveys, campaigns or otherwise ("the Other Channels"). Such personal information will be collected and/or used in accordance with the terms and conditions of this Privacy Policy, which is part of and incorporated into the Terms of Use of the Website/App.

Your express consent to collection and use of personal information by choosing to access our Websites/App and/or communicating with us through the other channels, you are indicating your express consent and agreement to the collection, transfer, processing, use and storage in accordance with this privacy policy of any personal information which may be obtained from you as a result.

We are always here to answer your questions about the privacy of the personal information you register with us and we obtain by Other Channels and how we use and share it. Just call our Customer Service if you have any questions or concerns that haven't been answered by this document or if you do not agree to any of the terms and conditions set forth in the Privacy Policy.

By accessing or using the Websites/App and/or communicating with us through the Other Channels, you grant to us (and our affiliates) a non-exclusive, worldwide, royalty-free, perpetual license to use your personal information for the purposes set forth herein.

What information do we collect?

Figura 28 - Página *Privacy Policy*

4.3.7 Página Frequently Asked Questions

A finalidade desta tarefa era criar uma página FAQ com um conjunto de perguntas e respostas que informam os utilizadores de respostas a dúvidas frequentes e dá a conhecer informações sobre a empresa. Para preencher o conteúdo desta página, esta foi integrada com o *Contentful*, atribuindo assim a responsabilidade pelo seu conteúdo à equipa de marketing (Figura 29).

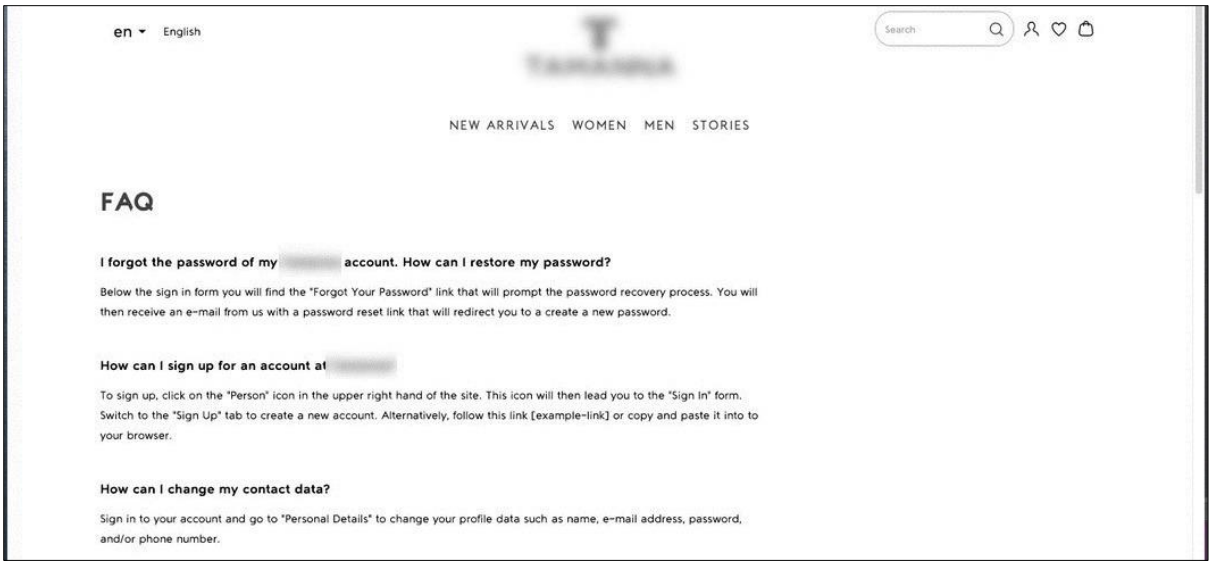


Figura 29 - Página FAQ

O modelo criado no *Contentful* foi diferente das páginas anteriores, este contém o título e um *array* de outro modelo Perguntas e Respostas, isto para permitir adicionar qualquer número de perguntas/respostas e aplicar um espaçamento entre cada entrada do modelo pergunta e resposta, mantendo assim o foco no conteúdo por parte da equipa de marketing.

Não foram necessárias muitas adaptações para dispositivos móveis, bastou apenas alterar o tamanho da largura disponibilizado para as diversas componentes da página (Figura 30).

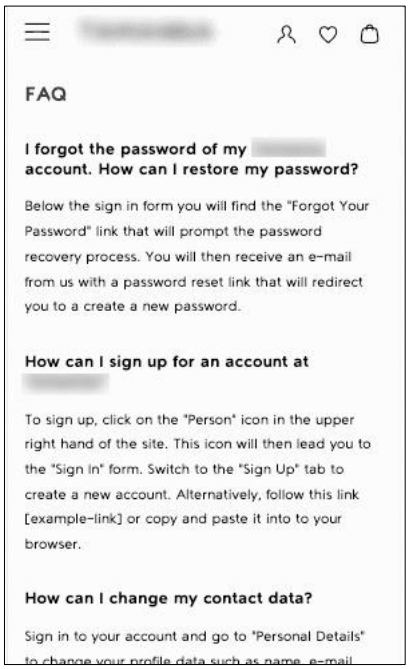


Figura 30 - Página FAQ em dispositivos móveis

Ao rodapé de todas as páginas também foi adicionado o link que redireciona os utilizadores para esta página (Figura 23).

4.3.8 Melhorias

Aquando do desenvolvimento das páginas abordadas neste capítulo, as mesmas seguiram o paradigma SSR, ou seja, a página era gerada no servidor cada vez que o cliente fazia um pedido (Figura 31).

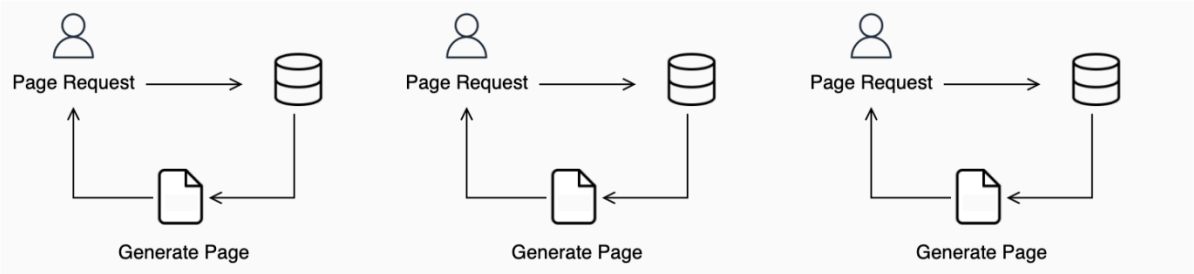


Figura 31 – Esquema do paradigma SSR

Estas foram posteriormente adaptadas para páginas estáticas SSG, como foram inicialmente idealizadas (Figura 32). O *Next.js* suporta a geração das páginas estáticas, sendo estas geradas no processo de construção do *marketplace* e guarda-as em cache para serem posteriormente apresentadas ao utilizador.

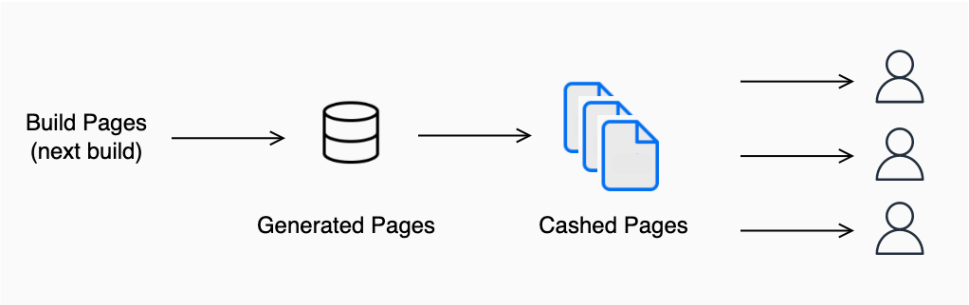


Figura 32 – Esquema do paradigma SSG

No caso das páginas das *Stories*, devido ao crescimento do seu número que, de forma proporcional, aumenta o tempo de construção da aplicação, recorreu-se ao *Incremental Static Regeneration* (ISR) [65]. Este utiliza SSG para páginas específicas, desta forma quando se adiciona uma nova *Story*, não há a necessidade de construir novamente toda a aplicação *web*.

Com este paradigma, algumas das páginas estáticas são geradas no processo de construção do *marketplace*, mas também permite que outras páginas estáticas sejam geradas sob pedido, ou seja, são geradas em tempo real à medida que são acedidas pelos utilizadores. Quando um utilizador acede a uma página que ainda não foi gerada, esta é construída no servidor seguindo o paradigma SSR (Figura 31), mas uma vez gerada é guardada em cache, servindo assim todos os utilizadores que tentem aceder à mesma página, durante um determinado tempo, até ser atualizada.

Este conceito de atualização da página já gerada corresponde à revalidação que é definida a cada página. Se definirmos o período de revalidação de 60 segundos, a página que iremos mostrar a todos os clientes é a mesma durante os 60 segundo após a sua geração. Desta forma reduz-se significativamente o tempo necessário para apresentar a página ao utilizador, pois esta já se encontra gerada (Figura 33).

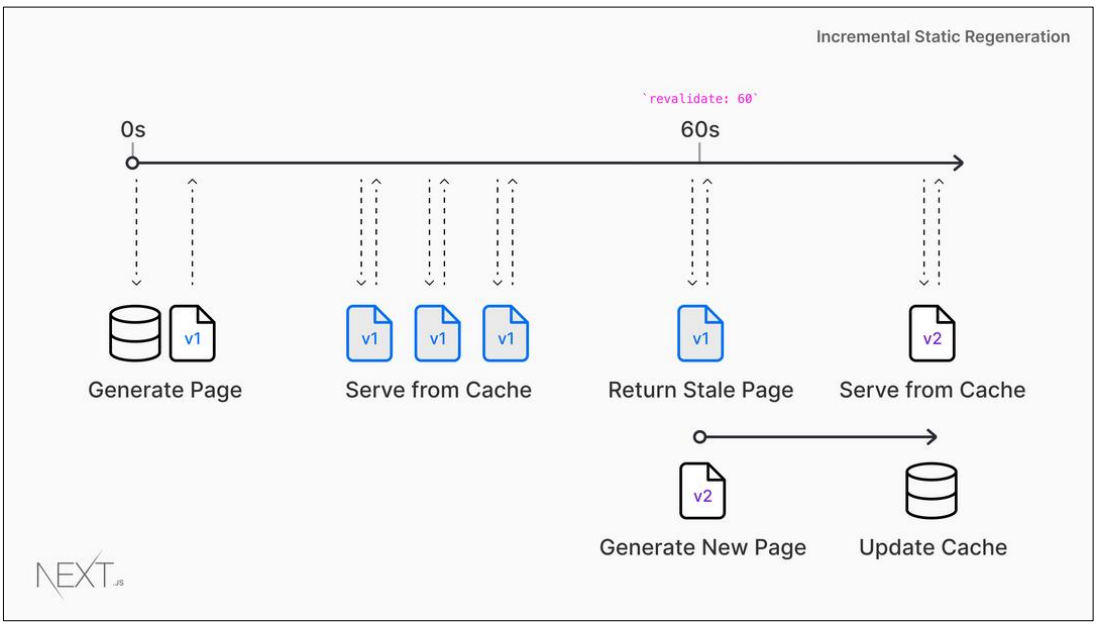


Figura 33 - Diagrama do fluxo de pedidos do modelo *ISR*

4.4 Componentes

Esta secção apresenta essencialmente tarefas não prioritárias. Uma vez que as funcionalidades principais do *website* ficaram completas no lançamento do MVP, o passo seguinte foi o desenvolvimento de tarefas de prioridade menor. As secções seguintes descrevem o desenvolvimento de algumas dessas tarefas.

4.4.1 Spinner

O site contém vários pedidos ao *backend* ou a outras API's externas como o *Contentful*, sendo estes pedidos assíncronos, levam algum tempo a carregar. Para melhorar a experiência do cliente é comum inserir um indicador de carregamento (*spinner*) enquanto os pedidos são feitos.

Para o desenvolvimento desta componente foi determinante a ajuda de um designer gráfico para a criação de uma imagem (gif). Para a sua integração foi utilizado o pacote *react-lottie* [66] que permite a manipulação de animações. O *spinner* teve igualmente de ser adaptado para um dispositivo móvel, alterando apenas o seu tamanho, consoante o tamanho do ecrã.

A segunda parte da execução desta tarefa correspondia à aplicação deste *spinner*, ou seja, procurar onde eram feitos pedidos assíncronos ao longo do site e aplicá-lo nessas páginas ou componentes. Este indicador foi então aplicado ao realizar algumas ações como o redireccionamento para a página principal após o login (Figura 34), a alteração de produtos no saco de compras, que também envolve a confirmação com o *backend* (Figura 35) e na finalização de uma compra (Figura 36), que depende da confirmação dada pelo *gateway* de pagamento, que é um intermediário virtual para as transações online. Todas as aplicações do *spinner* implicaram a alteração da lógica de forma a apresentar este indicador, quando o estado de um pedido assíncrono se encontra em progresso e retirá-lo quando o pedido é concluído.

Figura 34 - *Spinner* no redirecionamento após o Login

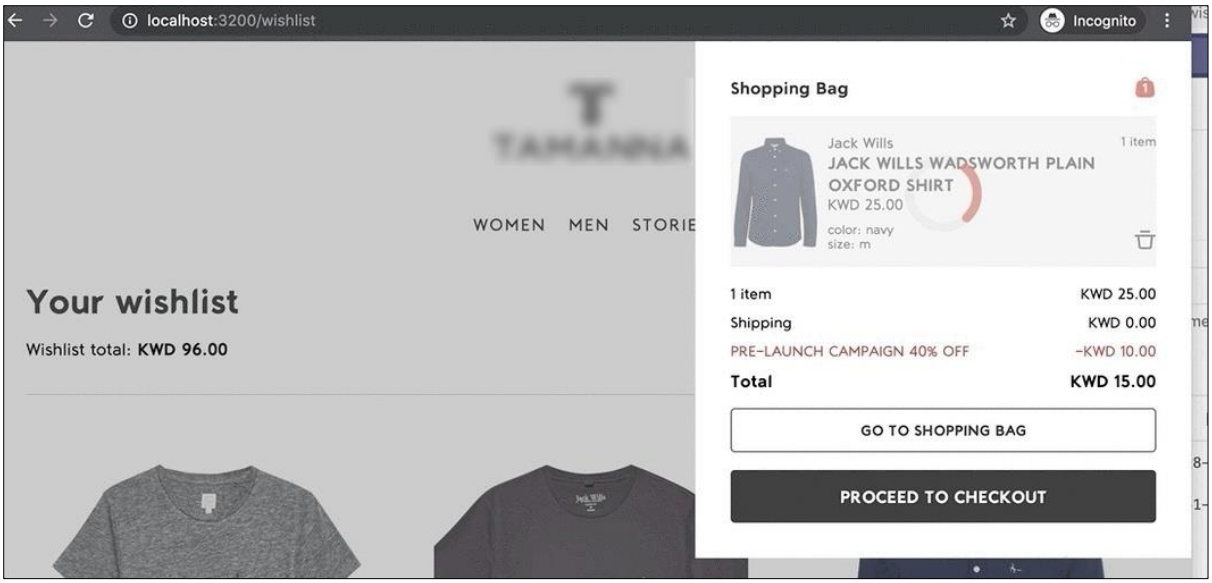


Figura 35 – *Spinner* refletindo a atualização do saco de compras

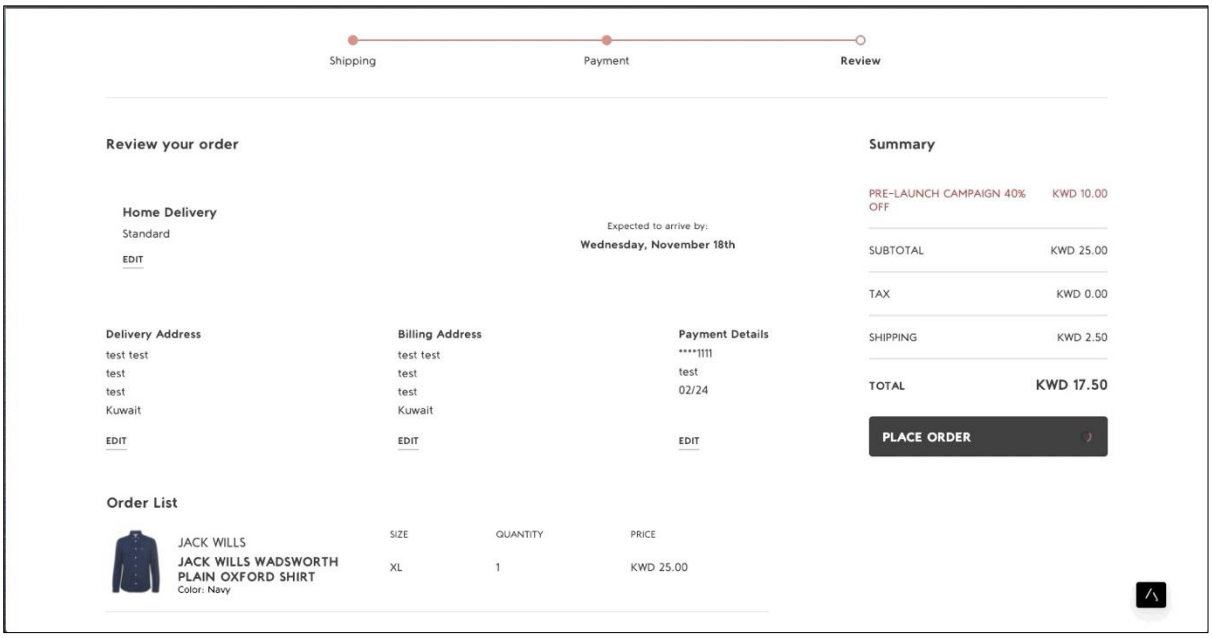


Figura 36 - Botão de compra com *spinner* para dar feedback ao utilizador

Para o caso da inserção do *spinner* em botões, foi então criado um botão generalizado que para além de suportar os comportamentos já existentes, como a adição de ícones, também incluía o estado de "carregamento" exibindo o *spinner* (Figura 37).

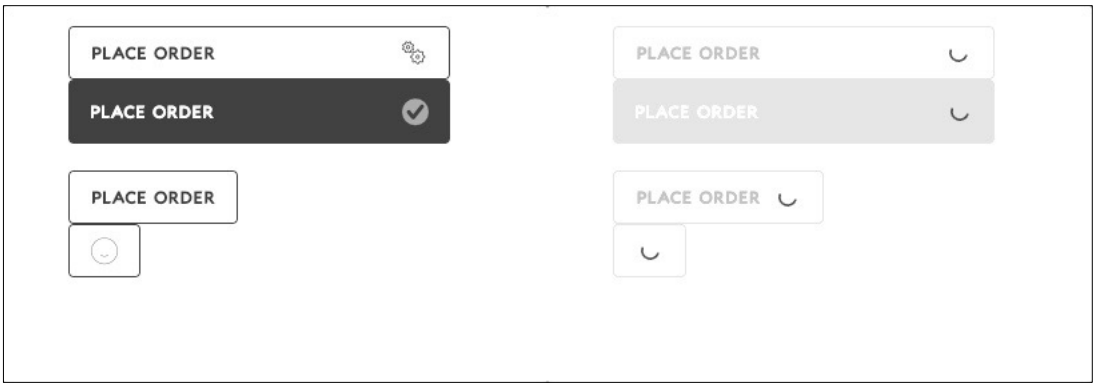


Figura 37 - Diferentes estados e tipos que o botão pode adotar

4.4.2 Módulo Shop by brand

Este módulo visa a dar uma visão global das várias marcas existentes no site e permitir de forma intuitiva redirecionar para uma página com os produtos da marca selecionada pelo utilizador. Este componente encontra-se presente nas páginas promocionais e foram desenvolvidas de forma que a equipa de Marketing conseguisse adicionar conteúdo através do *Contentful*. Este conteúdo está restrito a estruturas previamente implementadas, uma delas (*Promotion Brands*) é apresentada na Figura 38.

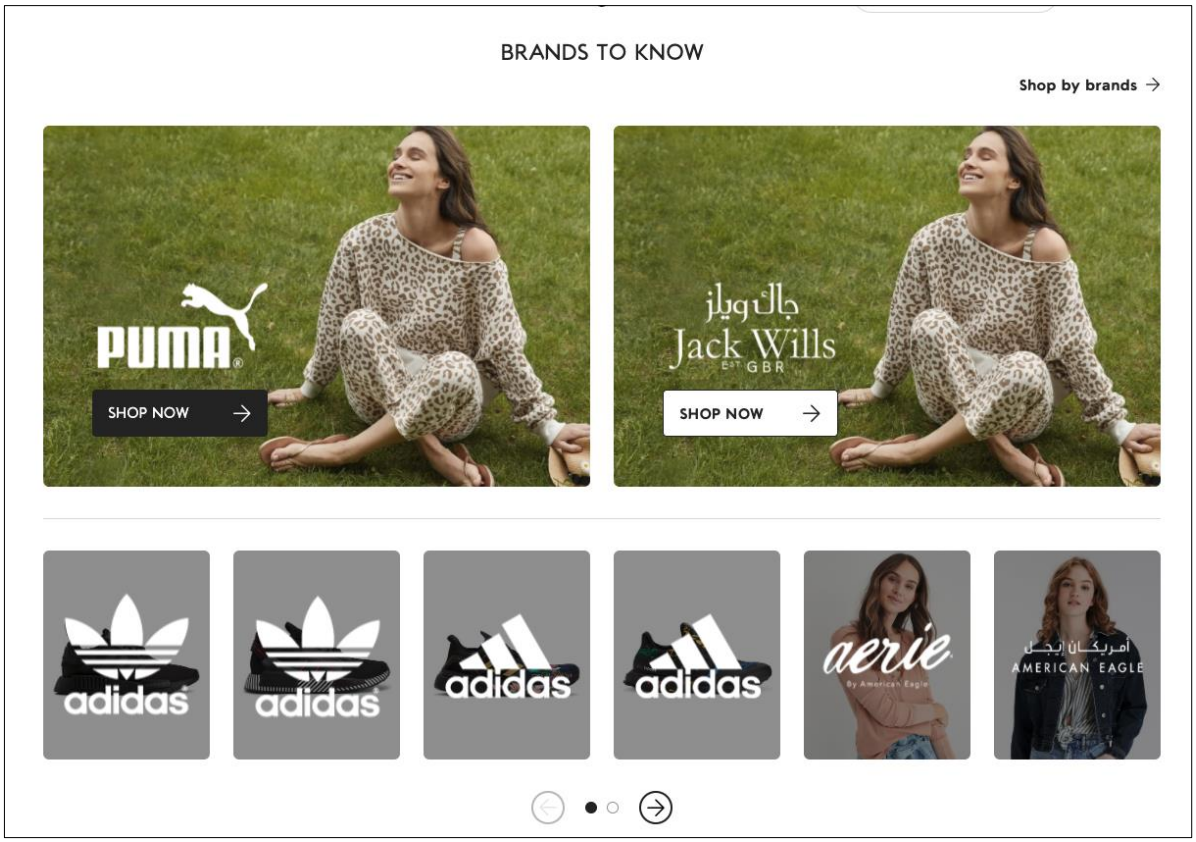


Figura 38 - Versão inicial da componente *Shop by Brand*

A estrutura *Promotion Brands* é composta por quatro elementos de caráter obrigatório: título, link, uma listagem que destaca duas marcas e outra listagem composta por um máximo de seis marcas. O título necessita de ter disponível as traduções para os dois idiomas. O link tem como objetivo redirecionar o utilizador para uma página que contém todas as marcas da categoria onde este se encontra, nomeadamente *Home, Woman, Men* ou *Kids*. A listagem que destaca as duas marcas recebe duas estruturas que contêm uma imagem, um logótipo, um link e texto para o botão CTA. Por fim, a listagem de marcas regulares recebe estruturas que contêm uma imagem, um logótipo e um link e, consoante o número de itens, ajusta o espaçamento e centra a posição do item (Figura 39). Todos estes dados são fornecidos através do *Contentful*.

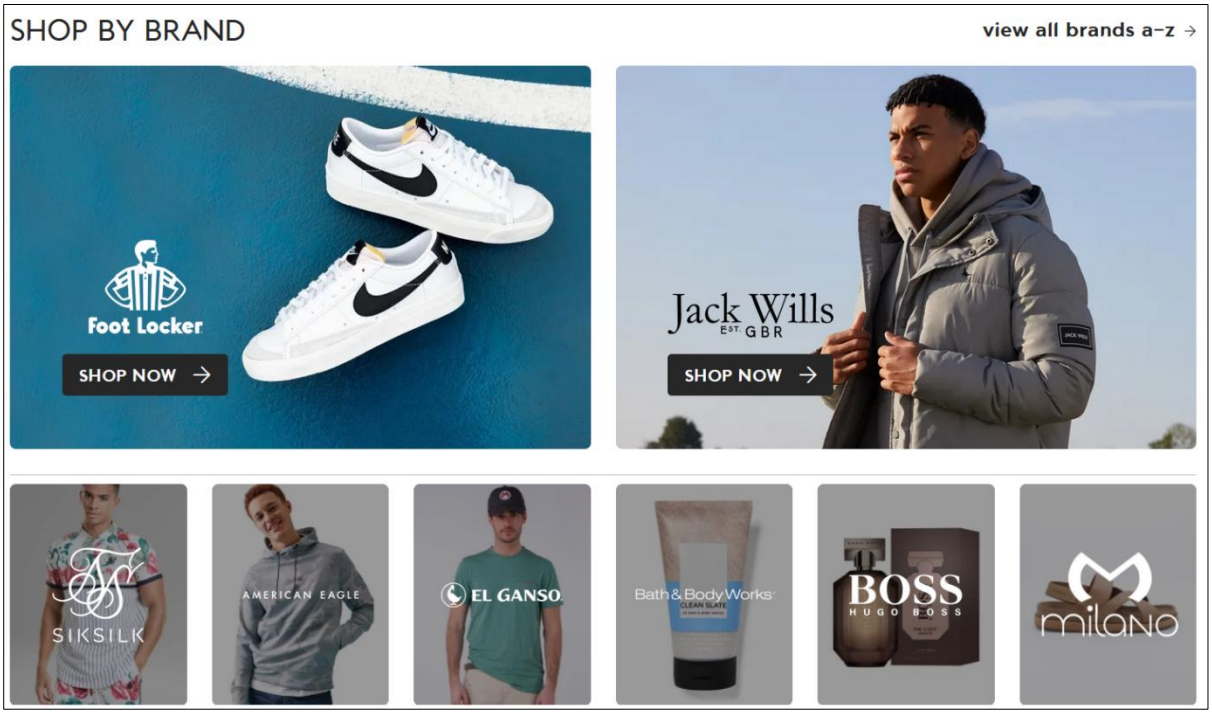


Figura 39 - Segunda versão da componente *Shop by Brand*

Uma das funcionalidades desenvolvidas neste contexto tinha os seguintes critérios de aceitação:

- Ser um carrossel de imagens de *brands*;
- Todas as imagens serem configuráveis e editadas no *Contentful*;
- Todos os links serem configurados no *Contentful*;
- Versão para computador ser composta por 6 colunas;
- Versão para dispositivos móveis ser composta por 2 colunas;
- Permitir que este carrossel possa ser reutilizado noutras páginas;

- Permitir imagens diferentes de acordo com o idioma;
- Permitir o controlo da cor do botão e do texto (tema) no *Contentful*.

O carrossel é uma estrutura que recebe uma lista de itens de um tipo genérico, dispondo-os numa listagem. Também permite navegar nessa listagem, caso o número de itens total seja maior do que o número de itens a visualizar de cada vez. Estes carrosséis tiveram de ser feitos de forma a se adaptarem ao idioma árabe, se necessário. Nesse caso a leitura é da direita para a esquerda e, como tal, a ordem dos itens é igualmente alterada, assim como os botões de navegação são invertidos.

Em dispositivos móveis esta componente tem um design e um comportamento diferente. O pedido feito à API do *Contentful* é feito sem distinção devolvendo a informação para preencher os dois carrosséis. Tendo em conta a restrição do tamanho do ecrã dos dispositivos móveis, adaptou-se a listagem de forma a esta conter um *scroll* horizontal. Desta forma, era possível inserir todas as marcas nos respetivos carrosséis.

O módulo *featured Brand* foi desenvolvido de forma a dar a escolher o estilo do botão "Shop Now". Tendo em conta que algumas imagens poderiam ser num tom escuro e seria difícil distinguir entre o botão e o *background* foi criada uma propriedade no seu modelo de forma a alterar o CSS associado à cor do botão e do seu texto. Este tinha estão o tema claro ou escuro de forma a ser adaptável à imagem de *background* (Figura 38).

4.4.3 Módulo Gift Note

Esta tarefa tinha como objetivo adicionar mais opções ao *checkout*, nomeadamente uma opção para a entrega de encomendas como presentes customizados (*Gift Note*). Ao escolher esta opção, o cliente recebia a encomenda numa embalagem diferente, juntamente com uma nota de agradecimento personalizada.

A versão inicial desta componente permitia escolher se o presente era embrulhado ou não, sendo a mensagem opcional e caso o utilizador escrevesse uma nota, a embalagem era garantidamente embrulhada (Figura 40). No entanto este formato revelou-se pouco intuitivo, levando à criação de uma nova versão, mais simples, na qual a *checkbox* fica selecionada sempre que é redigida uma nota, a compra é considerada como prenda, sendo por isso embrulhada. A *Gift Note* foi colocada no momento do *checkout* para, desta forma, estar presente em qualquer *flow* do utilizador, sem o sobrecarregar.

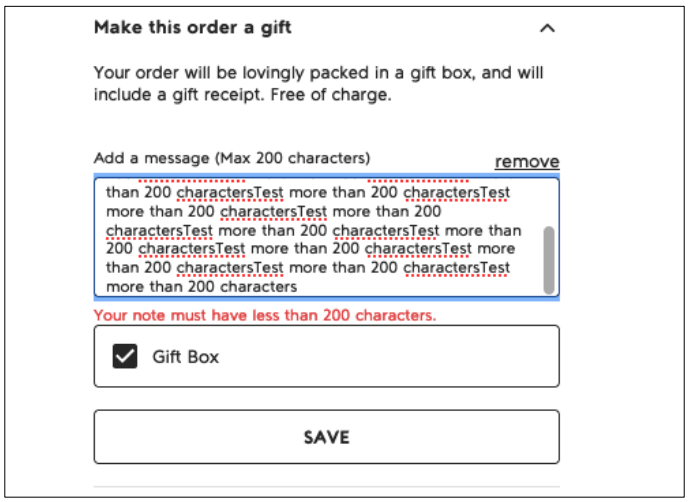


Figura 40 - Versão inicial do *Gift Note*

Tendo em conta a funcionalidade desta tarefa, os seus critérios de aceitação passavam por:

- Limitar a área de texto a 200 carateres;
- Conter um botão para apagar todo o texto;
- Suportar a tradução de texto.

Esta componente foi criada como um "acordeão" que expande para mostrar os detalhes relacionados com a prenda e permite a inserção de uma mensagem. Existe também uma mensagem de erro que é visível caso o limite de caracteres seja ultrapassado ao redigir nota. Ao redigir a mensagem é apresentado um botão que, ao ser selecionado, apaga todo o conteúdo da nota. A tradução do texto inserido também foi um critério de aceitação, como tal, o texto foi adicionado ao ficheiro *JavaScript Object Notation* (JSON) que contém todas as traduções em inglês relativas ao checkout e foi requisitada a sua tradução para árabe, através do *Lokalise* [67].

Num dispositivo móvel, o *Gift Note* tem o mesmo comprimento, no entanto o tamanho da letra e o espaçamento são alterados conforme as medidas pré-definidas para dispositivos móveis nestas duas propriedades.

Para que o utilizador soubesse da disponibilidade desta opção, adicionou-se também uma componente nas páginas de detalhes dos produtos (Figura 41). Isto implicou coordenar com o designer para ser possível adicionar um novo ícone à lista de fontes do projeto.

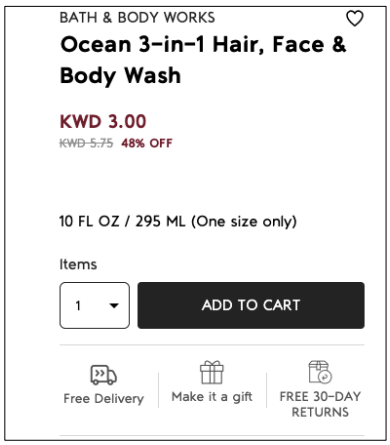


Figura 41 - Ícone do serviço novo disponível

No final foi adicionado um novo requerimento para garantir que o utilizador tinha feedback de quando esta opção está ativa. Para isso foi adicionado então um ícone (Figura 42) para mostrar ao utilizador que o saco de compras tinha o serviço de prenda associado.

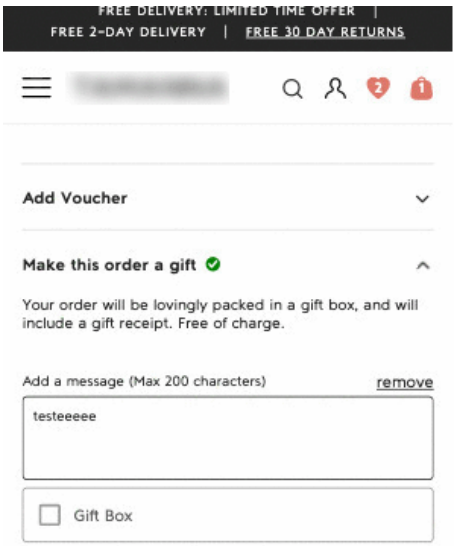


Figura 42 - *Gift Note* aplicado

Numa primeira fase, o registo do custo do embrulho não foi adicionado ao histórico da transação, quando aplicado. Isto acabou por não criar nenhuma inconsistência nos registos de compras do utilizador, pois este serviço for disponibilizado de forma gratuita. No entanto foi posteriormente necessário implementar o custo do embrulho no resumo da transação da compra, caso no futuro se quisesse cobrar esta funcionalidade (Figura 43).

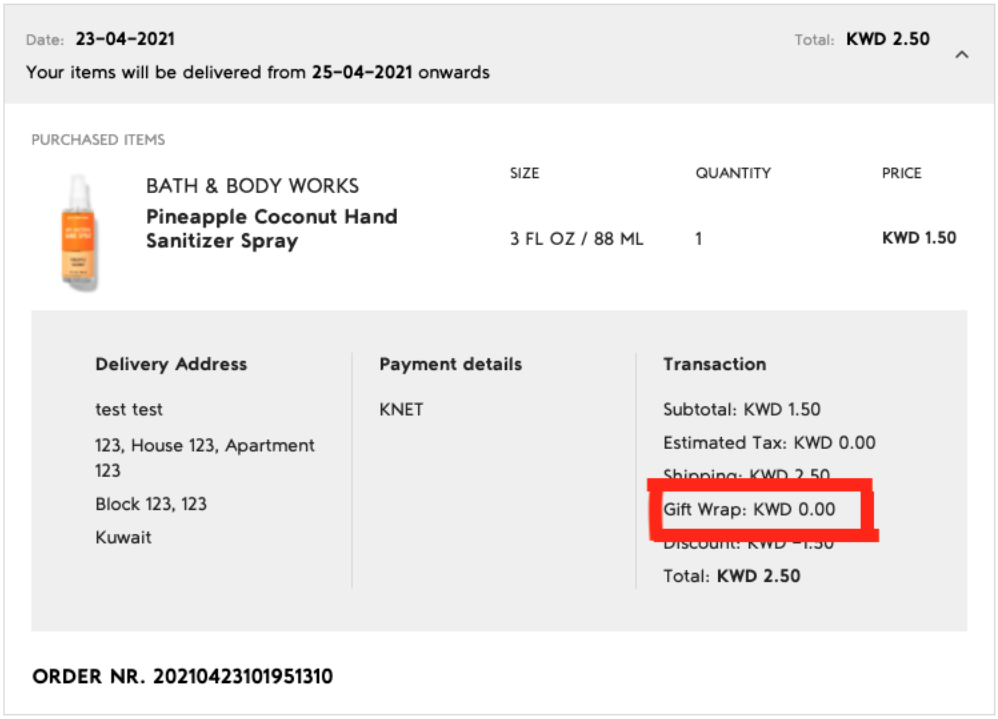


Figura 43 - Custo do serviço numa transação

4.5 Right To Left

O *marketplace* desenvolvido neste projeto está preparado para apresentar os mesmos conteúdos para os idiomas inglês e árabe. O idioma inglês utiliza a orientação tradicional no mundo ocidental, em que a leitura do conteúdo se realiza da esquerda para a direita – *Left To Right* (LTR) *script*, enquanto a leitura no idioma árabe é realizada direita para a esquerda – *Right To Left* (RTL) *script* (Figura 44).

Figura 44 - Página inicial no idioma árabe

O desenvolvimento dos conteúdos de forma a estes serem responsivos de acordo com idioma selecionado pelo utilizador cria diversos desafios, não só ao nível da tradução e apresentação do texto, mas também na adaptação dos conteúdos gráficos, que necessitam de seguir a mesma orientação [68].

O primeiro problema a ser resolvido, que é o mais simples, é a orientação do texto. Para isso basta uma simples mudança da direção das etiquetas *HyperText Markup Language* (HTML) para o formato RTL. No entanto, outros elementos, como uma imagem, não se adaptam corretamente com esta alteração, pois continuam a ser posicionados com base nas propriedades físicas da CSS, que incluem esquerda e direita, como o *margin-left* e o *padding-left* (Figura 45).

Figura 45 - Efeito da mudança de direção do documento

Para resolver o problema associado às propriedades físicas das CSS, utilizaram-se as propriedades lógicas que facilitam a adaptação da orientação dos elementos consoante a direção definida no documento [69] (Figura 46).

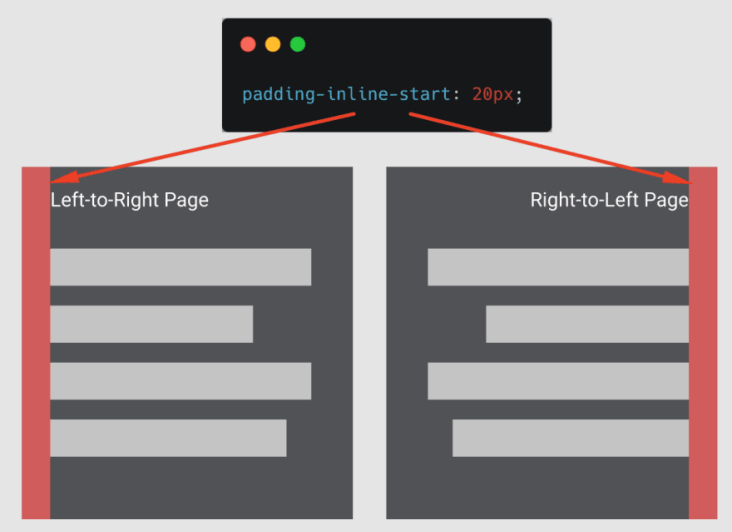


Figura 46 - *Padding-inline-start* aplicado a páginas LTR e RTL

Outro problema a considerar foi a adaptação dos ícones. Na Figura 47 é possível visualizar alguns exemplos dos três tipos de ícones utilizados neste projeto: bidirecionais, unidirecionais e associados a marcas.

Figura 47 - Exemplos dos diferentes tipos de ícones

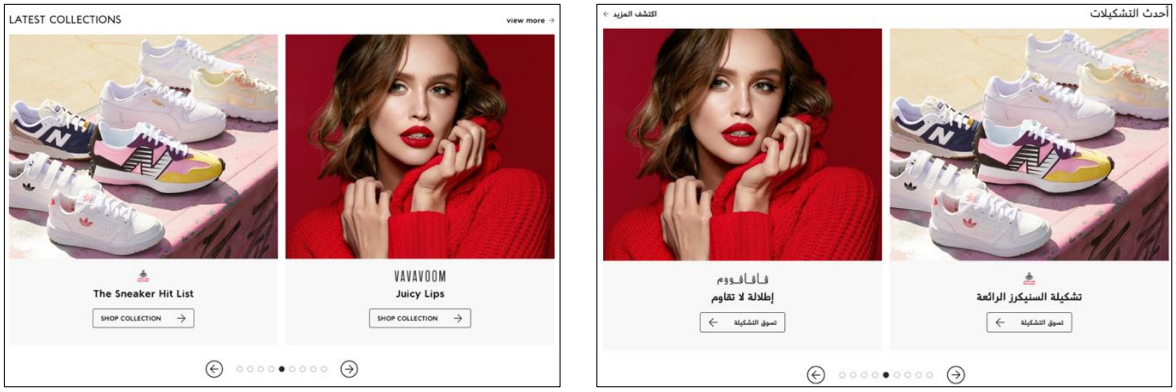
A direção dos ícones bidirecionais e dos associados a marcas não sofre alterações, sendo independente da direção do idioma. Por outro lado, a direção dos ícones unidirecionais é importante, pois o seu significado varia de acordo com a direção de leitura. Para implementar este comportamento diferenciativo dos ícones, foi criado um componente que os encapsula e

adiciona lógica que permite espelhar horizontalmente o ícone (Figura 48). Nesta lógica existe uma propriedade "*noFlip*" que permite definir se este ícone pode ou não ser espelhado. Todos os ícones bidirecionais e de marcas têm esta propriedade aplicada por isso a sua direção não é alterada. Os restantes (unidirecionais), como não têm esta propriedade aplicada, são espelhados consoante a direção do idioma.

```
import React from 'react';
import styled from 'styled-components';

const IconWrapper = styled('div')`
  html[dir='rtl'] &.flip-icon {
    transform: scaleX(-1);
  }
`;

const Icon = ({ name, onClick, noFlip }) => {
  return (
    <IconWrapper onClick={onClick} className={noFlip ? '' : 'flip-icon'}>
      <i className={`icon icon-${name}`} title={name} />
    </IconWrapper>
  );
};

export default Icon;
```

Figura 48 - Componente que encapsula os ícones

Alguns componentes também necessitam de uma adaptação ao idioma, como é o caso do carrossel (Figura 49-A). Neste projeto, esta componente foi importada de uma biblioteca *open-source* [70] e apesar de ser personalizável não continha suporte para RTL. Para adaptar o carrossel ao formato RTL é preciso alterar a ordem das imagens e modificar o comportamento dos outros elementos do carrossel. Para isso é necessário alterar a ordem da lista de imagens de forma que a última RTL corresponda à primeira LTR, atualizar comportamento despoletado ao pressionar os botões de próximo/anterior e alterar a lógica de desabilitar (*disable*) dos botões dependendo novamente da direção do documento (Figura 49-B).



A) B)

Figura 49 – *Carroussel*: A) Inglês; B) Árabe

Por último, apesar de a leitura em árabe ser feita na direção RTL, nem todo o texto é escrito desta forma, como é o caso da numeração. No início do desenvolvimento, quando se selecionava o idioma árabe todo o texto era apresentado da direita para a esquerda, o que levava à incorreta inserção de dados numéricos, afetando significativamente a experiência do utilizador quando este tinha de inserir, por exemplo, número de cartão de crédito, uma data, ou um número de telemóvel (Figura 50) [71]. Para solucionar este problema foi forçada a metodologia LTR quando se inserem dados numéricos, adicionando o *unicode* "\u200e" ao valor dos inputs que incluem valores numéricos [72].

A) B)

Figura 50 – A) Comportamento de números com direção RTL; B) Utilização do *unicode* LTR num documento orientado com RTL

4.6 Testes

O desenvolvimento de testes é de extrema importância, pois ajuda a garantir que o código desenvolvido está correto e contém menos erros, aumentando assim a confiança no lançamento de novas funcionalidades.

No desenvolvimento deste projeto, foram utilizados diferentes tipos de testes para assegurar a correta funcionalidade deste. No início do projeto a realização de testes não foi uma preocupação, dada a proximidade da data de entrega do MVP. Então os testes só foram desenvolvidos após o lançamento da aplicação *web*.

4.6.1 Testes Unitários

Os testes unitários isolam um componente da aplicação e certificam-se que este segue o comportamento expectável. Estes testes são vantajosos por causa da sua simplicidade e rapidez. Para serem mais eficientes é necessário isolar componentes a um nível granular.

No desenvolvimento dos testes, estes devem cobrir todos os diferentes resultados dessa componente, por exemplo, uma componente que quando recebe dados da API, mostra esses dados ou, em alternativa, uma mensagem da inexistência de dados, devem ser desenvolvidos testes para estes dois casos distintos.

Após o lançamento, como boa prática, cada funcionalidade desenvolvida tinha de ser acompanhada por testes unitários ou testes de integração. Esta prática foi mais tarde reforçada com uma regra que não permitia a inserção de novas funcionalidades em *develop* caso o *test coverage* do *merge request* descesse em relação ao valor existente em *develop*. Ao longo do tempo foram desenvolvidos testes para componentes previamente implementadas.

4.6.2 Testes de Integração

O objetivo dos testes de integração é verificar a funcionalidade, a fiabilidade e a execução de diferentes componentes quando integradas. Tipicamente usado para componentes mais complexas que requerem a interação com diferentes componentes, tais como, o *checkout* para uma compra que tem dependências consoante o utilizador está autenticado ou não, ou a lista de desejos que precisa de interagir com um produto para testar a sua adição/remoção de produtos na lista.

4.6.3 Testes *end 2 end*

Os testes *end 2 end* têm a finalidade de reproduzir o comportamento de um utilizador a interagir com a aplicação, tornando possível testar funcionalidades do início até ao fim. Estes testes são notoriamente instáveis, geralmente falham por motivos inesperados e imprevisíveis devido a peculiaridades do navegador, problemas do tempo de resposta da API, animações e

caixas de diálogo inesperadas. No entanto, oferecem uma maior confiança na decisão se a aplicação está ou não a funcionar corretamente. Os testes *end 2 end* exigem muita manutenção e são executados muito lentamente comparativamente aos outros tipos de testes.

Ao longo do projeto surgiu a necessidade de ter testes *end 2 end*, pois à medida que o lançamento se aproximava, a estabilidade e funcionalidade da aplicação *web* era de fundamental importância. Como o projeto é um portal de *e-commerce*, o percurso mais valioso e comum do utilizador, corresponde à procura de um produto, adicioná-lo ao saco de compras e prosseguir para o *checkout*. No entanto, também existem outros testes relevantes que simulam, a criação de uma conta e a verificação dos seus dados no perfil do utilizador e também um teste à funcionalidade de autenticação.

Os testes foram desenvolvidos com a *framework* de testes *Cypress*. Esta ferramenta foi escolhida porque tem uma baixa curva de aprendizagem e executa os testes de forma rápida. Tendo em conta que os testes *end 2 end* são os mais lentos, este é um fator muito positivo.

Após o desenvolvimento de testes *end 2 end*, estes foram adicionados à *pipeline* de lançamento nos ambientes de *development* e *staging* para garantir que nenhuma nova funcionalidade colocava em causa o processo de *checkout*.

Com as alterações de código, lançamento de novas funcionalidades e a importação de novas marcas, surgiram situações em que os testes *end 2 end* deixavam de estar funcionais, sendo necessário refazê-los.

Um desses casos foi uma mudança no processo do *checkout* que fez que alguns dos testes passassem a ser obsoletos. Nesse sentido, foi então corrigido o número de iterações necessárias para completar a compra, adicionando um novo método de pagamento – *K-Net Payment Gateway* (KNET) [73], que é bastante comum no Médio Oriente e, como tal, é crucial ser testado.

A importação de uma nova marca também era um processo instável que muitas vezes falhava e necessitava de ser repetida. Um novo requisito surgiu então da necessidade de testar que uma marca tinha sido corretamente importada. Para isso foi criado um teste que recebe um subconjunto de produtos da marca adicionada e realiza a compra de um exemplar de cada variante (tamanho e cor) para cada um dos produtos.

Utilizando os dados estatísticos referentes aos dispositivos de acesso à aplicação *web*, a quantidade de utilizadores de dispositivos móveis era, previsivelmente, o maior mercado deste *marketplace*, com valores acima dos 80%. Como o *Cypress*, ao testar dispositivos móveis, só simula o browser com um tamanho de ecrã reduzido, não sendo possível testar as particularidades dos dispositivos móveis, surgiu a necessidade de realizar os testes em dispositivos móveis reais. Posto isto, foi feita uma migração da *framework* de testes de *Cypress* para outra *framework*, o *Nightwatch*, que foi desenvolvida sob *Selenium* [74] e usa *WebDriver* [75], pois este já permite executar testes automáticos num emulador de dispositivos móveis, como *BrowserStack* [76], aumentando a fiabilidade dos testes para dispositivos móveis.

Dado que tanto o *Cypress* e o *Nightwatch* usam *JavaScript*, os testes anteriores foram maioritariamente reaproveitados, alterando somente alguns métodos característicos do *Nightwatch*. No entanto, foi necessário refazer alguns testes, nomeadamente a autenticação e registo do utilizador. Como foi desenvolvido outro projeto, *Seller lab*, que é responsável pela importação e exportação de produtos e a atualização de preços e inventário, todos os testes *end 2 end* referentes a estas funcionalidades ficaram da responsabilidade na nova equipa.

4.7 Monitorização

Os projetos que tencionam escalar requerem monitorização, como tal, apesar de não ser um requisito para o MVP, esta funcionalidade é relevante, principalmente a partir do momento em que o portal está em produção. A monitorização permite detetar falhas antes que estas afetem a maioria dos clientes, ajuda também na resolução de erros.

4.7.1 Incorporação na equipa de gestão de incidentes

Quando o *e-commerce* surgiu e começou a ter um número cada vez maior de clientes, passou a ser necessário ter uma equipa de gestão de incidentes, caso surgisse algum problema que comprometesse o normal funcionamento do site. Como tal, foram criadas três equipas especializadas: infraestrutura, *frontend* e *backend*. No âmbito deste estágio, fui integrado na equipa do *frontend*, que era a mais adequada para meus conhecimentos técnicos. Esta equipa tinha como responsabilidade analisar os problemas no fluxo do *checkout* e nas páginas cujo conteúdo era originário do *Contentful*, visto que foi inteiramente desenvolvido pela equipa do *frontend*.

Foi utilizado o *PagerDutty* [77] como aplicação para notificar e controlar o calendário das equipas de gestão de incidentes. Todas as semanas tinha de ser assegurado que estaria, pelo menos, um membro de cada uma das 3 equipas, responsável por estar de prevenção a problemas pontuais que pudessem surgir. A equipa de gestão de incidentes do *frontend* era constituída por três membros que seguiam um calendário rotativo semanal.

As três equipas existentes, eram divididas em dois perfis funcionais. A infraestrutura pertencia ao primeiro perfil e era a primeira a ser convocada caso surgisse um incidente grave. Após uma primeira análise esta recorria ou não às equipas de *backend* e *frontend*, que pertencem ao segundo perfil. A maioria dos incidentes eram resolvidos pela equipa do primeiro perfil.

4.7.1.1 O processo de gestão um incidente

Ao surgir um incidente, este é catalogado segundo um formato predefinido composto pelos seguintes campos: sumário, passos para reproduzir, resultado esperado, resultado real e URL. Estes campos permitem entender mais facilmente o problema e facilitam a atribuição do seu nível de severidade enumerados na Tabela 3.

Tabela 3 - Níveis de Severidade de incidentes

Severidade	Descrição	Exemplos	Tempo de Resposta	Tempo de Resolução
SEV-1 *	O sistema encontra-se num estado crítico e tem um impacto direto na capacidade de um utilizador fazer uma compra	• Não é possível adicionar produtos ao saco de compras. • A funcionalidade de registo não está funcional. • O site não é acessível a grande parte dos utilizadores.	15 minutos	4 horas
SEV-2 *	Incidentes que caso não resolvidos podem escalar para SEV-1	• Um pequeno grupo de utilizadores não consegue adicionar produtos ao saco de compras. • Um pequeno grupo de utilizadores não consegue fazer *Login*/*Register*. • O site não é acessível a uma pequena parte de utilizadores.	15 minutos	8 horas
SEV-3	Requer ação por parte da equipa de desenvolvimento, mas não afeta a capacidade de um utilizador fazer uma compra	• Não é possível adicionar/remover produtos à lista de desejos. • Não é possível aceder ao histórico de compras.	4 horas úteis	2 dias úteis

SEV-4	Um problema detetado que não afeta o uso normal de um utilizador, logo é tratado como um erro com baixa prioridade	• Texto ou uma imagem que está desalinhada num browser em específico.	1 dia útil	5 dias úteis

incidentes críticos

O próximo passo era a criação de um *ticket* na ferramenta *Jira* e de um canal de comunicação associado a esse *ticket*, no qual estavam presentes a equipa de gestão de incidentes e o membro que detetou o respetivo problema. Qualquer discussão do incidente ficava arquivada nesse canal, evitando a partilha de informação por mensagens privadas. Desta forma, todas as etapas da resolução do incidente ficavam disponíveis neste canal e podiam ser acedidas pela equipa responsável ou por qualquer engenheiro destacado para ajudar no problema.

Surgindo um incidente de nível de severidade 1 ou 2, verifica-se se este é um problema que é rapidamente resolvido e, caso não o seja, procura-se mitigar o problema de forma a não causar mais impacto (problemas). Para isso, algumas soluções são reverter alguma funcionalidade que continha uma *breaking change*, ou simplesmente colocar um certo produto como indisponível para venda, evitando assim a persistência do problema (Figura 51).

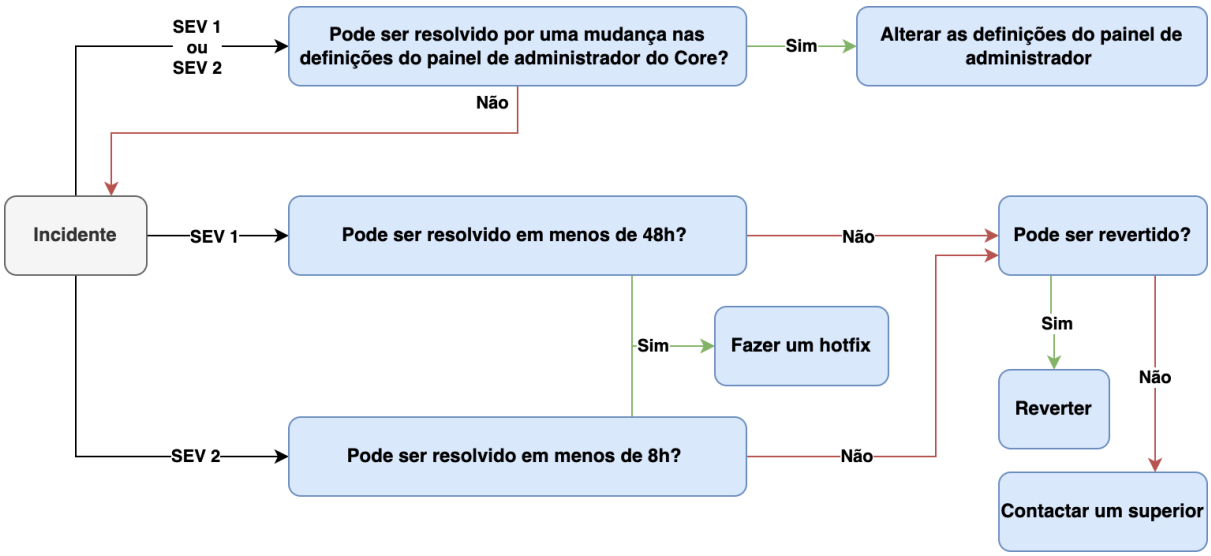


Figura 51 – Fluxo de um incidente de severidade 1 ou 2

Quando um incidente do tipo severidade 1 ou 2 é concluído, é requerido descrever e publicar um documento, o *postmortem*, na plataforma de documentação da empresa

(*Confluence*), para que fique disponível toda a informação relativa ao incidente para memória futura. O *postmortem* segue um modelo que contém os seguintes campos: o estado do incidente, o *ticket* que foi criado no *Jira*, uma linha temporal com os eventos que se sucederam ao longo do incidente, como por exemplo, a hora em que foi reportado, a hora em que foi detetada a origem do problema e a hora da sua resolução ou mitigação, as pessoas envolvidas na resolução do incidente, um tópico com a explicação do que gerou este incidente e também como esse problema foi mitigado ou resolvido. No caso de ter sido apenas mitigado, cria-se um plano de resolução, tipicamente associado a um novo *ticket* de resolução com alta prioridade. Este deixa de estar limitado pelas horas e pode ser resolvido em horário de trabalho.

4.7.1.2 Exemplos de Resoluções de Incidentes

Enquanto estive integrado na equipa de gestão de incidentes de *frontend* aconteceram alguns incidentes, dos quais vou destacar dois.

O primeiro incidente consistiu num conjunto de páginas que estavam a dar erro e a falhar na sua apresentação. Visto que estas páginas têm como fonte de informação o *Contentful*, verificou-se se algo novo teria sido adicionado nestas páginas. A causa do problema estava relacionada com um novo subtópico que foi adicionado a um Menu e este problema só surgia em páginas que incorporavam este subtópico no seu Menu. Este subtópico era do tipo link, mas não tinha nenhum URL associado porque a página deste subtópico ainda estava a ser desenvolvida. Como naquele momento o URL era uma propriedade obrigatória, isto impedia que a página fosse construída corretamente. De forma a mitigar o problema foi simplesmente inserido o URL da própria página neste subtópico, passando este URL a ser redundante, mas garantindo as funcionalidades dessas páginas e após o desenvolvimento da nova página, o URL era atualizado para o correto. Como plano de resolução foi criado um *ticket* que envolvia tornar opcional o campo URL dos subtópicos presentes no menu. Desta forma, prevenia-se que a má inserção de conteúdo por parte da equipa de Marketing impedisse a disponibilidade da página.

O outro incidente era um problema associado ao *frontend*, no qual alguns produtos não podiam ser adicionados ao saco de compras. Após uma análise aos produtos em causa, verificou-se que estes tinham dois aspetos em comum, a variante (corresponde à combinação da cor do produto e o tamanho) em causa estava fora de stock e ambos permitiam apenas a visualização de determinadas variantes. Verificou-se então que o produto tinha stock, mas não era nas variantes disponíveis no portal e o problema surgiu da incorreta importação dos

produtos. Para a resolução deste problema, foi realizada uma pesquisa na base de dados por produtos, onde se verificavam esses dois aspetos e estes foram colocados indisponíveis para desaparecem da pesquisa e das páginas de listagem de produtos.

4.7.2 Métricas

Para compreender melhor o comportamento do cliente no portal e melhorar a taxa de conversão (número de utilizadores que entram no *website* e efetuam uma compra), foi implementado um processo de rastreamento. Para isso usou-se o *Segment* no *frontend*, que rastreia eventos e recolhe dados relativos à interação do utilizador com o site [78]. Estes dados são posteriormente enviados para o *Google analytics*.

Um engenheiro ficou responsável por tratar os dados agregados na plataforma *Google analytics*. Este operava em conjunto com o *frontend*, para alinhar no modelo de dados para cada evento (Figura 52).

```
analytics.track('Product Clicked', {
  product_id: '507f1f77bcf86cd799439011',
  sku: '45790-32',
  category: null,
  name: 'Monopoly: 3rd Edition',
  brand: 'Hasbro',
  variant: null,
  price: 18.99,
  quantity: 1,
  coupon: null,
  currency: 'KWD',
  position: 3,
  url: 'https://www.example.com/product/path',
  image_url: 'https://www.example.com/product/path.jpg',
});
```

Figura 52 - Exemplo de um modelo de dados enviados num evento

No *frontend* foram vários os eventos rastreados, tais como: produtos visualizados, produtos pesquisados, produtos clicados, mas todos pertencem a três grupos distintos:

- ***identify*:** Serve para identificar o cliente. A informação é enviada após o cliente se autenticar;

- ***page*:** Indica em que página o utilizador se encontra. À medida que o utilizador muda de página este pedido é enviado;
- ***track*:** Identifica as ações do cliente, como por exemplo, a inserção de um produto na lista de desejos.

Alguns dos resultados da análise destas métricas resultaram na colocação dos artigos mais populares num espaço de destaque no *marketplace* e na descoberta de anomalias através do rastreamento da jornada do cliente. Um exemplo desta situação, foi o caso em que a quantidade de utilizadores que tentou criar conta era substancialmente maior do que o sucesso desta ação, o que revelou que de facto existiam erros a impossibilitar a criação de uma conta de utilizador na plataforma.

4.8 Processos Internos

Desde o início do projeto *KuwaitShop*, o objetivo era desenvolver o produto e ajudar a empresa a entrar neste mercado e a tornar-se independente, para quando se estabilizar, a *xgeeks* se retirar do projeto. Parte deste processo envolve ajudar a recrutar pessoas para o projeto *KuwaitShop*.

4.8.1 Incorporação na equipa de Recrutamento

O processo de recrutamento associado à engenharia consiste em 7 etapas, descritas na Tabela 4. Este processo recorre ao uso de uma aplicação, *Greenhouse* [79], que fornece um *software* de recrutamento, agrupando informação de diversas aplicações ligadas ao recrutamento num só portal. Este agiliza todo o processo e permite o levantamento de algumas métricas como a taxa de rejeição, a qualidade média dos candidatos, entre outras. Permite ainda armazenar várias informações que podem ser consultados por toda a equipa de recrutamento, como o *Curriculum Vitae* (CV), o *Linkedn* e as avaliações efetuadas ao candidato ao longo do processo de recrutamento.

Tabela 4 - Etapas do processo de recrutamento

Etapas	Descrição	Responsável
Recrutamento	Fase inicial, através de referências que existam e por ferramentas como *Linkedn* e *AmazingHire*	Recursos Humanos
Revisão da aplicação	A equipa dos recursos Humanos analisa o CV decide se o candidato passa à próxima fase	Recursos Humanos
Entrevista de triagem	Videochamada de 30 minutos, seguida do preenchimento de uma tabela de desempenho	Recursos Humanos
Entrevista técnica	Tem como objetivo avaliar os conhecimentos do candidato com base num conjunto de questões técnicas, sumarizando as respostas com o preenchimento de uma tabela de desempenho	Engenheiro de *software* especializado
Desafio técnico	É enviado um desafio ao candidato consoante a área onde se está a candidatar e, após estar completo, é feita a revisão à solução do candidato e preenchido uma tabela de desempenho	Engenheiro de *software* especializado
Entrevista final	Verifica se o candidato se enquadra ao projeto e se possui conhecimentos técnicos	Engenheiro chefe
Oferta	Discussão com as pessoas envolvidas no processo de recrutamento e é feita uma oferta ao candidato	Engenheiro chefe

Pertenci à equipa de recrutamento como engenheiro de *software* especializado em *frontend*, no qual realizei duas entrevistas técnicas. Nestas entrevistas estive acompanhado de um engenheiro sénior que me ajudou a integrar no processo e liderava a entrevista.

4.8.2 Mentor na integração de novos colaboradores

Durante os meses que trabalhei na *KuwaitShop* tive a possibilidade de ser mentor (*buddy*) para alguns engenheiros recém-contratados pela empresa.

A função de um *buddy* é apoiar a integração de um novo colega durante o processo de acolhimento, especialmente durante a primeira semana de integração, ajudando a familiarizar-se com o novo ambiente de trabalho. Nesse sentido, cabe ao mentor atuar como fonte de informação, sendo o primeiro ponto de contacto dentro da empresa, esclarecer qualquer tipo de dúvidas e orientar sobre a cultura e formas de trabalhar na *KuwaitShop*.

O processo de acolhimento também engloba uma vertente mais técnica na qual é explicada a funcionalidade dos diferentes repositórios, a forma de organização das pastas no projeto do

frontend, dar a conhecer os encontros existentes da equipa, o funcionamento do fluxo associado a um *ticket* no *Jira*, ajudar a configurar o projeto (*frontend*) localmente e configurar a VPN para aceder ao ambiente *development* e *staging*. Por fim, é também proposta uma série de exercícios para reforçar a utilização de várias tecnologias presentes no projeto. Estes são revistos num *merge request* para um ramo específico do novo colaborador, para ambientar o engenheiro ao uso do *Gitlab*.

Também no processo de *onboarding*, há a realização do primeiro *ticket* em *pair programming*, para colocar em prática os conceitos em cima descritos.

4.8.3 Documentação

Várias tarefas desenvolvidas consistiram em criar documentação com intuito de evitar criar silos de informação. Esta documentação foi criada no *Confluence* para ficar visível para toda a empresa e alguma desta informação também foi partilhada no *stack overflow* da empresa.

4.8.3.1 Hotfixes

O objetivo de documentar este processo é o de procurar instruir toda a equipa sobre a melhor forma de aplicar um *hotfix*. Isto surge do facto de a equipa de gestão de incidentes ser rotativa, podendo haver a necessidade de ser aplicado um *hotfix* por qualquer membro da equipa e também para evitar possíveis complicações que podem acontecer.

Na Figura 53 está retratado o problema que surgiu várias vezes após a aplicação incorreta de um *hotfix*. Quando este é aplicado em *master* e há uma continuação do desenvolvimento normal que resulta na adição de *commits* em *develop* e posteriormente em *staging*, a junção de *staging* para *master* acaba por ser bloqueada devido a conflito nas versões.

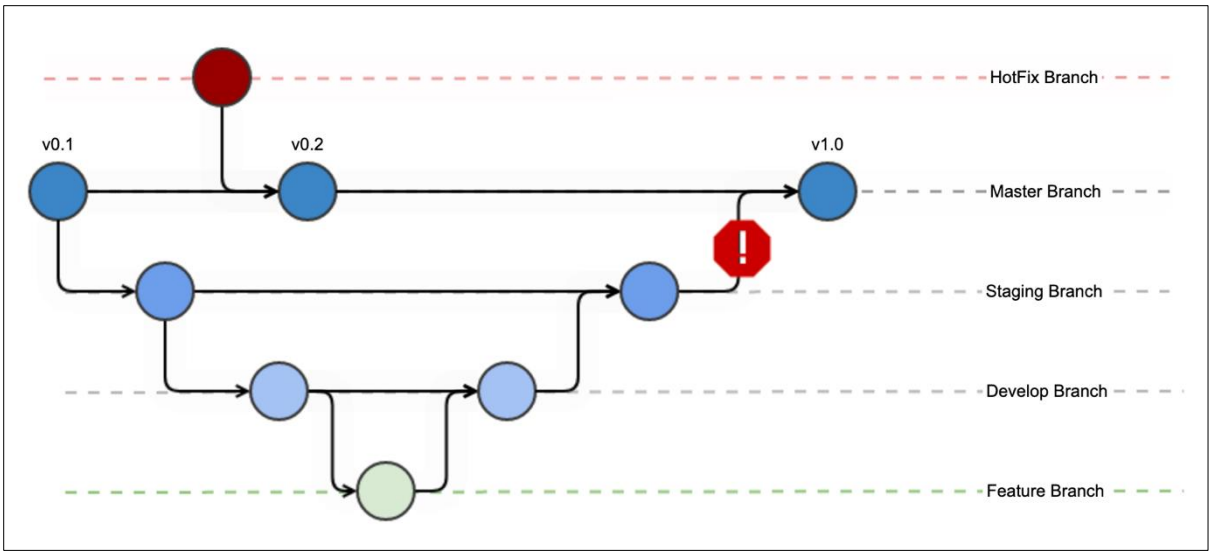


Figura 53 - Incorreta aplicação de um *Hotfix*

Para a correta aplicação de um *hotfix*, após a resolução do problema é necessário aplicar o *hotfix* no ramo correto e resolver os problemas de consistências nos diferentes ambientes *develop*, *staging* e *master*, por intermédio do comando *Git cherry-pick*[6] (Figura 54).

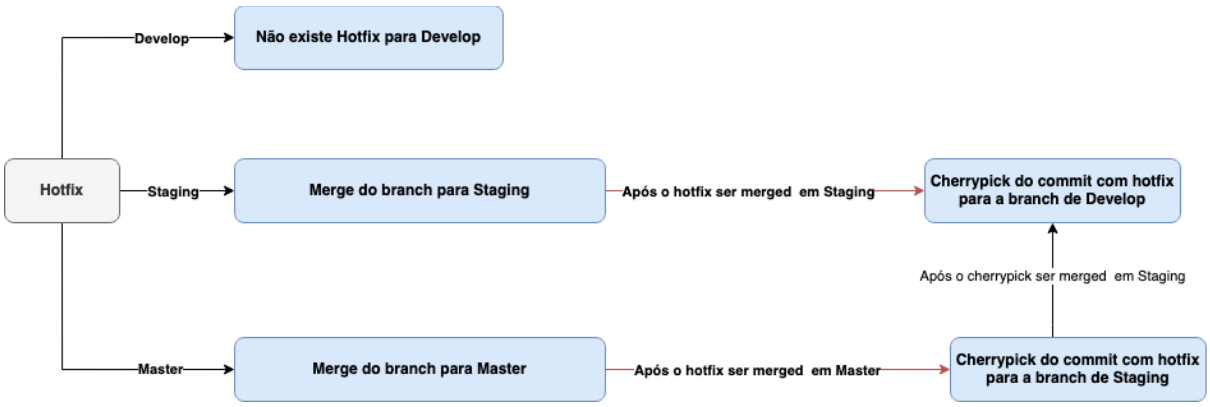


Figura 54 - Processo para a realização de um *Hotfix*

4.8.3.2 Revert

O uso do *revert* surge em dois casos, quando existe um incidente e quando uma tarefa não passa os critérios quando é testada pela equipa de QA. Num destes casos é necessária a aplicação do *revert* e esta pode acontecer a qualquer membro da equipa, logo, o processo precisa de ser documentado, consistindo nos seguintes passos:

[6] O *cherry-pick* é um comando de *Git* que permite selecionar um ou mais *commits* e introduzir essas mudanças num ramo à nossa escolha.

1. Revertendo o *commit* com o problema (Figura 55);
2. Solicitar a alguém da equipa de infraestrutura para validar e dar *merge* (por questões de permissões);
3. É necessário resolver os problemas de consistências entre os diferentes ambientes, através de um *cherry-pick* (Figura 56);

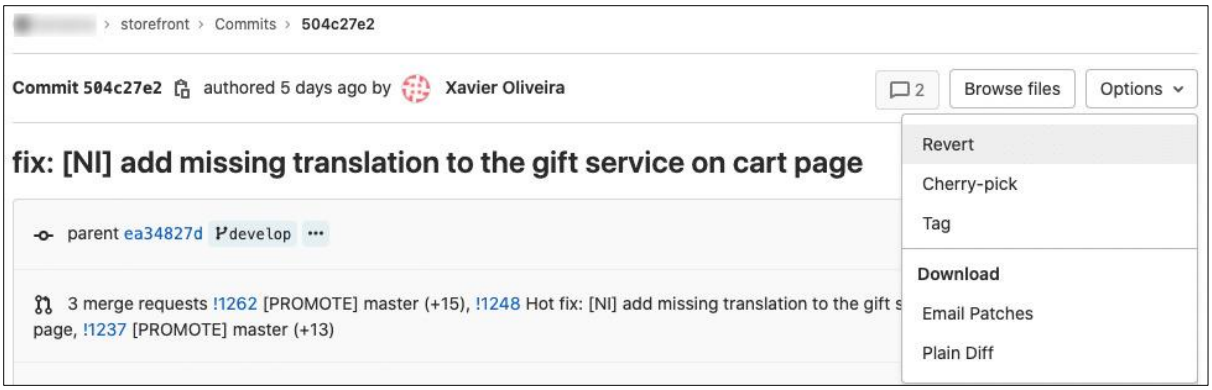


Figura 55 - Como reverter um *commit* através do *Gitlab*

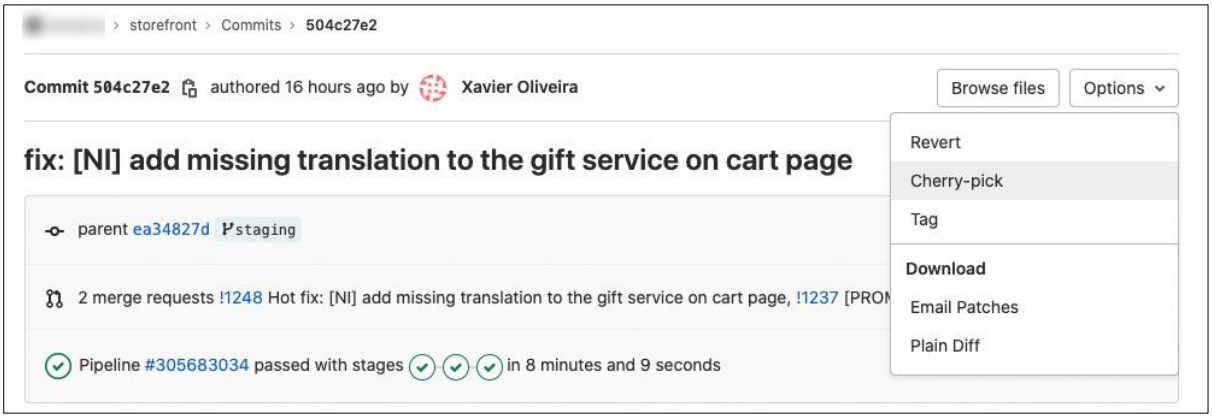


Figura 56 - Como fazer um *Cherrypick* de um *commit* através do *Gitlab*

4.8.3.3 Stack overflow da equipa

Não fazia sentido as questões mais técnicas relacionadas com o código ficarem documentadas no *confluence* para toda a empresa. Nesse sentido, foi criado um *stack overflow* para a equipa de engenheiros com intuito de documentar os problemas/dúvidas mais comuns que surgem no desenvolvimento de tarefas (Figura 57). São assim criadas questões que muitas vezes são respondidas pelo autor da mesma. Este *stack overflow* da equipa contém uma integração com o *Slack* [80], que é a ferramenta de comunicação da empresa, permitindo

publicar uma mensagem num canal com todos os engenheiros, quando uma pergunta ou resposta é adicionada.

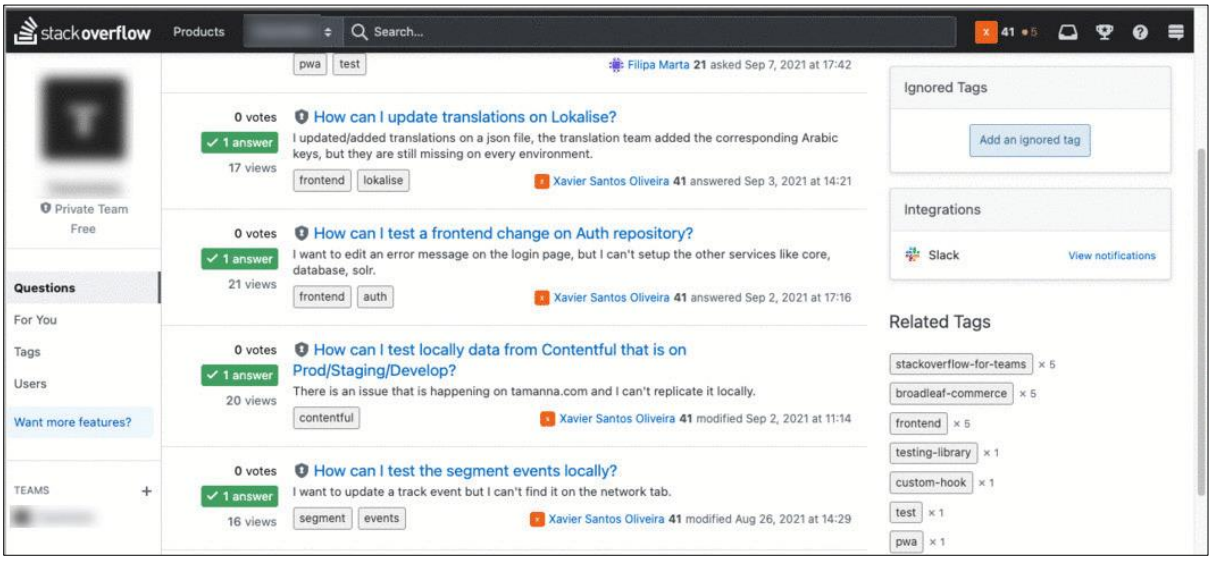


Figura 57 - *Stack overflow* da equipa

4.9 Resultados

Após a entrada em produção do *marketplace*, a empresa teve um registo crescente do número transações. Alguns dos resultados atingidos até junho de 2021 foram:

- 17 Marcas diferentes integradas e disponíveis no portal;
- Tráfego de 1000 visitas ao portal por meio de publicidade da Google;
- Tráfego de 1600 visitas ao portal por meio de publicidade nas redes socias;
- 36 transações por dia através de campanhas nas redes sociais;
- Retenção de clientes por volta dos 5%;
- 6500 utilizadores subscritos a emails de promoções da empresa.

Após o término do estágio, uma vez que me mantive a exercer as mesmas funções como desenvolvedor de *software* nesta empresa, foi atingido o primeiro milhão de euros no valor de vendas no mês de setembro de 2021 e até à data de escrita deste relatório já ultrapassou os dois milhões de euros.

5 Conclusão

O balanço deste estágio só pode ser positivo, pois serviu de porta de entrada para o mercado de trabalho. Primeiramente deu-me a possibilidade de ter contacto com novas tecnologias e ferramentas, como *Next.js*, *Cypress*, *Nightwatch*, SWR e *Contentful*. Tive a oportunidade de ter formação nas áreas da minha escolha, nomeadamente o *frontend* e *DevOps*, e reforçar o meu conhecimento em *Docker* e *Kubernetes* e aprender *React.js* e *TypeScript*. O projeto permitiu também aplicar os conhecimentos já adquiridos, especialmente em relação às metodologias, uma vez que pude aplicar a metodologia XP e *Scrum* no dia a dia num projeto real, mas também conhecimentos em *Git* e *JavaScript*.

Durante o meu estágio pude participar no desenvolvimento de um *marketplace* online para a indústria da moda, no qual integrei a equipa de *frontend* e desenvolvi funcionalidades como páginas estáticas e componentes para este *website*. Como este projeto teve o seu início relativamente aquando do começo do meu estágio, este acabou por ser uma grande oportunidade, pois tive a possibilidade de estar envolvido no desenvolvimento de um produto desde o estado inicial até este entrar em produção. No entanto, porque este projeto era de desenvolvimento rápido, acelerou a minha aprendizagem, não me permitindo aprofundar os conhecimentos nas tecnologias da forma que mais gostaria.

Foi possível atingir o primeiro objetivo do projeto, que era o lançamento do MVP em novembro. O período desde o lançamento do MVP até ao lançamento público permitiu-me fazer o desenvolvimento de outras tarefas e correção de erros, sem a pressão do desenvolvimento rápido e com a *webpage* disponível apenas para utilização interna da empresa *KuwaitShop*. Este período ajudou-me a aumentar a confiança no código que desenvolvia, permitindo-me arriscar e potenciando assim o meu crescimento como desenvolvedor de *software*.

O *marketplace* foi desenvolvido com a metodologia *mobile-first* sendo possível consultar o portal num computador e em dispositivos móveis, e de forma a suportar dois idiomas muito diferentes, quer no tipo de conteúdo textual, quer na direção de leitura. Estes requisitos, chamaram-me à atenção das diversas possibilidades, pormenores e casos de uso com os quais devo ter atenção durante o desenvolvimento de *software*. Aprendi que tenho de ter sempre em conta o público-alvo, neste caso, o idioma e o tipo de dispositivo, ou seja, preparar a lógica da

UI para as características dos diferentes requisitos e manter as várias vertentes do código abstraídas para um desenvolvimento menos problemático.

Adquiri ainda outras responsabilidades no projeto *KuwaitShop*, nomeadamente, fazer parte da equipa de gestão de incidentes, da equipa de recrutamento e mentor de novos colegas que se juntavam à equipa, o que permitiu auxiliar o cliente representando da melhor forma os valores da *xgeeks*. O facto de estar a trabalhar com um cliente sedeado noutro país, permitiu que eu desenvolvesse outras valências como o idioma inglês, escrito e falado, lidasse com pessoas de várias nacionalidades e adquirisse conhecimentos sobre uma cultura nova.

Como resultado do trabalho desenvolvido neste estágio, foi publicado e apresentado um artigo científico na *CISTI'2022 - 17th Iberian Conference on Information Systems and Technologies*. Este artigo descreve os desafios e soluções encontradas para as particularidades de desenvolver um *website* que suporta um idioma cuja orientação de leitura é RTL [68].

Por fim, saber que participei na construção de um projeto, que não só entrou em produção, como ao fim de alguns meses atingiu resultados bastante positivos, nomeadamente, mais de 6500 utilizadores registados no *marketplace*, com um excelente tráfego diário, fez-me sentir realizado. Pude ter contacto com tecnologias e ferramentas inovadoras e atuais que não conhecia e muitas delas me serão úteis durante todo o meu percurso profissional. Durante estes meses evolui como engenheiro de *software* não só nas minhas capacidades técnicas, mas também nas habilidades interpessoais.

5.1 Trabalho futuro

Desde a sua entrada em produção, o *marketplace* tem sido um sucesso, tendo já algum destaque no Kuwait. Havendo a expetativa que, num futuro muito próximo, obtenha idêntico sucesso nos países vizinhos. No entanto, ainda existem algumas áreas nas quais o portal pode ser melhorado.

Para além da contínua manutenção da aplicação *web* e a futura aplicação móvel que iria ser desenvolvida, outras funcionalidades podem ainda ser adicionadas. Uma delas passaria por dar destaque a marcas de luxo. Estas teriam a sua própria página com um design mais apelativo e customizável. Podiam ainda ser adicionadas as opções de *checkout* anónimo e avaliar os produtos após a compra, quando o utilizador está registado.

Apesar do site estar desenvolvido de acordo com os requisitos do cliente e ter diversas funcionalidades interessantes e apelativas, este tem em falta o suporte à acessibilidade, por isso, considero essa uma das melhorias mais importantes a realizar neste *marketplace*.

Bibliografia

[1] ‘xgeeks | Software development made in Portugal’. https://xgeeks.io/ (accessed Sep. 23, 2022).

[2] A. Gupta, B.-C. Su, and Z. Walter, ‘An Empirical Study of Consumer Switching from Traditional to Electronic Channel: A Purchase Decision Process Perspective’, *International Journal of Electronic Commerce*, vol. 8, no. 3, 2004.

[3] P. Dolfen *et al.*, ‘Assessing the Gains from E-Commerce’, *NBER Working Paper*, no. 25610, Feb. 2019, Accessed: Sep. 25, 2022. [Online]. Available: http://www.nber.org/papers/w25610

[4] ‘E-commerce, Trade and the COVID-19 Pandemic’, May 2020, doi: 10.30875/17F1E119-EN.

[5] ‘JavaScript’. https://www.javascript.com/ (accessed Sep. 23, 2022).

[6] ‘React – A JavaScript library for building user interfaces’. https://reactjs.org/ (accessed Sep. 23, 2022).

[7] ‘Next.js by Vercel - The React Framework’. https://nextjs.org/ (accessed Sep. 23, 2022).

[8] ‘KI group’. https://kigroup.de/ (accessed Sep. 23, 2022).

[9] ‘Docker’. https://www.docker.com/ (accessed Sep. 23, 2022).

[10] ‘Kubernetes’. https://kubernetes.io/ (accessed Sep. 23, 2022).

[11] ‘Serviços de computação em nuvem - Amazon Web Services (AWS)’. https://aws.amazon.com/pt/?nc2=h_lg (accessed Sep. 23, 2022).

[12] ‘Serviços de Computação na Cloud | Microsoft Azure’. https://azure.microsoft.com/pt-pt/ (accessed Sep. 23, 2022).

[13] ‘The Go Programming Language’. https://go.dev/ (accessed Sep. 23, 2022).

[14] ‘The web framework for perfectionists with deadlines | Django’. https://www.djangoproject.com/ (accessed Sep. 23, 2022).

[15] ‘Logística Global - Envios Internacionais | Página Inicial da DHL | Portugal’. https://www.dhl.com/pt-pt/home.html (accessed Sep. 23, 2022).

[16] ‘Porsche HOME - Porsche Portugal’. https://www.porsche.com/portugal/ (accessed Sep. 23, 2022).

[17] ‘Cazoo: The better way to buy your next car’. https://www.cazoo.com/global/ (accessed Sep. 23, 2022).

[18] ‘Veículos de Passageiros Mercedes-Benz’. https://www.mercedes-benz.pt/?group=all&subgroup=see-all&view=BODYTYPE (accessed Sep. 23, 2022).

[19] ‘Manage Your Team’s Projects From Anywhere | Trello’. https://trello.com/ (accessed Sep. 23, 2022).

[20] ‘Git’. https://git-scm.com/ (accessed Sep. 23, 2022).

[21] ‘TypeScript: JavaScript With Syntax For Types.’ https://www.typescriptlang.org/ (accessed Sep. 23, 2022).

[22] ‘Agile Alliance’. https://www.agilealliance.org/ (accessed Sep. 23, 2022).

[23] K. Beck *et al.*, ‘Manifesto for agile software development’, 2001.

[24] ‘Extreme Programming: A Gentle Introduction.’ http://www.extremeprogramming.org/ (accessed Sep. 23, 2022).

[25] B. Böckeler and N. Siessegger, ‘On Pair Programming’, Jan. 15, 2020. https://martinfowler.com/articles/on-pair-programming.html (accessed Sep. 23, 2022).

[26] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Boston, MA, USA: Addison-Wesley, 1999.

[27] K. Schwaber and J. Sutherland, ‘The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game’, 2020, Accessed: May 23, 2022. [Online]. Available: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

[28] C. Schults, ‘The Difference between Extreme Programming vs. Scrum’, May 14, 2021. https://www.coscreen.co/blog/extreme-programming-vs-scrum-difference/ (accessed Sep. 23, 2022).

[29] C. Nance, *TypeScript Essentials*. Packt Publishing Ltd, 2014.

[30] ‘Most used web frameworks among developers 2022 | Statista’. https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ (accessed Sep. 25, 2022).

[31] A. Boduch and R. Derks, *React and React Native: A complete hands-on guide to modern web and mobile development with React.js*, 3rd ed. Packt Publishing, 2020.

[32] ‘Next.js by Vercel - The React Framework’. https://nextjs.org/ (accessed Sep. 23, 2022).

[33] ‘React Hooks for Data Fetching – SWR’. https://swr.vercel.app/ (accessed Sep. 23, 2022).

[34] ‘Visual Studio Code - Code Editing. Redefined’. https://code.visualstudio.com/ (accessed Sep. 23, 2022).

[35] ‘Welcome to Python.org’. https://www.python.org/ (accessed Sep. 23, 2022).

[36] ‘Java | Oracle’. https://www.java.com/en/ (accessed Sep. 23, 2022).

[37] ‘The One DevOps Platform | GitLab’. https://about.gitlab.com/ (accessed Sep. 23, 2022).

[38] ‘Jira | Issue & Project Tracking Software | Atlassian’. https://www.atlassian.com/software/jira (accessed Sep. 23, 2022).

[39] ‘Postman API Platform’. https://www.postman.com/product/what-is-postman/ (accessed Sep. 23, 2022).

[40] ‘Confluence | Your Remote-Friendly Team Workspace | Atlassian’. https://www.atlassian.com/software/confluence (accessed Sep. 23, 2022).

[41] ‘Introduction to Cypress | Cypress Documentation’. https://docs.cypress.io/guides/core-concepts/introduction-to-cypress#What-you-ll-learn (accessed Sep. 23, 2022).

[42] ‘Nightwatch.js | Node.js powered End-to-End testing framework’. https://nightwatchjs.org/ (accessed Sep. 23, 2022).

[43] ‘GitHub - airbnb/javascript: JavaScript Style Guide’. https://github.com/airbnb/javascript (accessed Sep. 26, 2022).

[44] A. Neary, ‘Rearchitecting Airbnb’s Frontend.’ https://medium.com/airbnb-engineering/rearchitecting-airbnbs-frontend-5e213efc24d2 (accessed Sep. 26, 2022).

[45] ‘hypernova - npm’. https://www.npmjs.com/package/hypernova (accessed Sep. 26, 2022).

[46] ‘GitHub - lelandrichardson/redux-pack: Sensible promise handling and middleware for redux’. https://github.com/lelandrichardson/redux-pack (accessed Sep. 26, 2022).

[47] ‘Home v6.4.1 | React Router’. https://reactrouter.com/en/main (accessed Sep. 26, 2022).

[48] ‘ABOUT YOU’. https://corporate.aboutyou.de/en/ (accessed Sep. 26, 2022).

[49] ‘Frontend Stack at ABOUT YOU Desktop Unit’. https://aboutyou.tech/blog/frontend-stack-at-about-you-desktop-unit-21e68bc32f92/ (accessed Sep. 26, 2022).

[50] ‘API-first content platform to build digital experiences | Contentful’. https://www.contentful.com/ (accessed Sep. 23, 2022).

[51] ‘Desktop vs Mobile Market Share Worldwide | Statcounter Global Stats’. https://gs.statcounter.com/platform-market-share/desktop-mobile/worldwide/2021 (accessed Jan. 23, 2022).

[52] ‘Overview | Kubernetes’. https://kubernetes.io/docs/concepts/overview/ (accessed Sep. 23, 2022).

[53] ‘Apache Kafka’. https://kafka.apache.org/ (accessed Sep. 26, 2022).

[54] ‘Welcome to Apache Solr - Apache Solr’. https://solr.apache.org/ (accessed Sep. 26, 2022).

[55] ‘Apache ZooKeeper’. https://zookeeper.apache.org/ (accessed Sep. 26, 2022).

[56] ‘Cloud Monitoring as a Service | Datadog’. https://www.datadoghq.com/ (accessed Sep. 26, 2022).

[57] ‘Husky - Git hooks’. https://typicode.github.io/husky/#/ (accessed Sep. 23, 2022).

[58] ‘GitHub - conventional-changelog/commitlint: Lint commit messages’. https://github.com/conventional-changelog/commitlint (accessed Sep. 23, 2022).

[59] ‘Ghost: Turn your audience into a business’. https://ghost.org/ (accessed Sep. 26, 2022).

[60] ‘WordPress.com: Fast, Secure Managed WordPress Hosting’. https://wordpress.com/ (accessed Sep. 26, 2022).

[61] ‘The most flexible Headless CMS | Magnolia Headless CMS’. https://www.magnolia-cms.com/ (accessed Sep. 26, 2022).

[62] ‘Getting Started with Contentful and JavaScript | Contentful’. https://www.contentful.com/developers/docs/JavaScript/tutorials/using-js-cda-sdk/ (accessed Sep. 23, 2022).

[63] ‘Content Management API | Contentful’. https://www.contentful.com/developers/docs/references/content-management-api/ (accessed Sep. 23, 2022).

[64] ‘CRM Software & Cloud Computing Solutions - Salesforce EMEA’. https://www.salesforce.com/eu/ (accessed Sep. 27, 2022).

[65] ‘Incremental Static Regeneration – Vercel Docs’. https://vercel.com/docs/concepts/next.js/incremental-static-regeneration (accessed Sep. 23, 2022).

[66] ‘react-lottie - npm’. https://www.npmjs.com/package/react-lottie (accessed Sep. 23, 2022).

[67] ‘A Localization and Translation Software Tool | Lokalise’. https://lokalise.com/ (accessed Sep. 27, 2022).

[68] X. Oliveira and A. Gonçalves, ‘The Development of an Arabic Online Fashion Marketplace’, *Iberian Conference on Information Systems and Technologies, CISTI*, vol. 2022-June, 2022, doi: 10.23919/CISTI54924.2022.9820349.

[69] ‘CSS Logical Properties and Values’. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Logical_Properties (accessed Feb. 18, 2022).

[70] ‘react-responsive-carousel - npm’. https://www.npmjs.com/package/react-responsive-carousel (accessed Feb. 18, 2022).

[71] ‘Right-to-left Styling’. https://rtlstyling.com/ (accessed Feb. 18, 2022).

[72] ‘H34: Using a Unicode right-to-left mark (RLM) or left-to-right mark (LRM) to mix text direction inline | Techniques for WCAG 2.0’. https://www.w3.org/TR/WCAG20-TECHS/H34.html (accessed Feb. 18, 2022).

[73] ‘Payment Gateway – The Shared Electronic Banking Services’. https://www.knet.com.kw/services/payment-gateway/ (accessed Sep. 27, 2022).

[74] ‘Selenium’. https://www.selenium.dev/ (accessed Sep. 27, 2022).

[75] ‘WebDriver | Selenium’. https://www.selenium.dev/documentation/webdriver/ (accessed Sep. 27, 2022).

[76] ‘Automated App Testing On Real Mobile Devices | BrowserStack’. https://www.browserstack.com/app-automate (accessed Sep. 27, 2022).

[77] ‘PagerDuty | Real-Time Operations | Incident Response | On-Call | PagerDuty’. https://www.pagerduty.com/ (accessed Sep. 27, 2022).

[78] ‘Spec: V2 Ecommerce Events | Segment Documentation’. https://segment.com/docs/connections/spec/ecommerce/v2/ (accessed Sep. 23, 2022).

[79] ‘Greenhouse | Applicant tracking system and recruitment software’. https://www.greenhouse.io/uk (accessed Sep. 26, 2022).

[80] ‘Slack is your digital HQ | Slack’. https://slack.com/ (accessed Sep. 26, 2022).