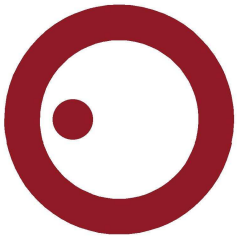


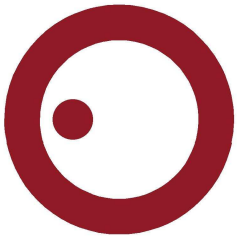
escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Engenharia Informática – Computação Móvel

DESENVOLVIMENTO DE APLICAÇÃO MÓVEL PARA INTEGRAÇÃO E MELHORIA DE PROCESSOS EM PLATAFORMA WEB

LOURENÇO DA SILVA LOURENÇO

Leiria, setembro de 2024



escola superior de tecnologia e gestão
instituto politécnico de leiria

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Engenharia Informática – Computação Móvel

DESENVOLVIMENTO DE APLICAÇÃO MÓVEL PARA INTEGRAÇÃO E MELHORIA DE PROCESSOS EM PLATAFORMA WEB

LOURENÇO DA SILVA LOURENÇO
Número: 2220649

Estágio realizado sob orientação da Professora Doutora Marisa da Silva Maximiano (marisa.maximiano@ipleiria.pt).

Leiria, setembro de 2024

AGRADECIMENTOS

Gostaria de agradecer a todas as pessoas e instituições que contribuíram para a realização deste trabalho.

Em primeiro lugar, um agradecimento especial à minha orientadora, Marisa Maximiano, pelo apoio, orientação e conhecimento que disponibilizou ao longo de todo o processo. A sua disponibilidade e incentivo foram essenciais para a concretização deste trabalho.

Agradeço também ao ValGrupo, pela oportunidade de realizar este estágio e por disponibilizar os recursos necessários para o desenvolvimento deste trabalho. A confiança depositada em mim foi essencial para o meu crescimento, tanto a nível profissional como pessoal.

Um reconhecimento à equipa de desenvolvimento com a qual colaborei, pelo seu apoio e espírito de equipa, que foram determinantes para o sucesso deste trabalho.

Não poderia deixar de agradecer à minha família e amigos, pelo constante apoio, compreensão e encorajamento ao longo desta jornada. Este apoio foi fundamental para a conclusão deste trabalho.

Obrigado!

RESUMO

Este relatório descreve o estágio realizado no ValGrupo, focado no desenvolvimento de soluções tecnológicas para melhorar a eficiência dos processos de auditoria, controlo de qualidade e gestão interna. O principal objetivo consistiu em atualizar o Portal de Gestão, desenvolvido em *Laravel* e *Vue*, e a criação de uma nova Aplicação Móvel, utilizando *Flutter*, para garantir a flexibilidade e eficiência no terreno, de acordo com as necessidades específicas do ValGrupo.

No contexto da empresa na qual se realizou o estágio, foi identificada a necessidade de otimizar o processo de auditorias, um processo anteriormente manual e com elevado consumo de papel. O processo de desenvolvimentos adotado seguia uma metodologia ágil, através do *Scrum*, para o desenvolvimento contínuo e incremental das funcionalidades requisitadas. Mais concretamente, a Aplicação Móvel foi criada de modo a suportar o seu uso *offline*, para que os colaboradores pudessem realizar auditorias em locais sem acesso à *Internet*, enquanto no Portal foram adicionadas funcionalidades para gerir e avaliar este processo. Foi também adicionada à API do Portal a funcionalidade de sincronização dos dados da Aplicação Móvel.

Os resultados gerais obtidos podem ser considerados positivos, com o desenvolvimento bem sucedido de ambas as plataformas, resultando numa melhoria significativa na eficiência das auditorias, na redução do consumo de papel e num sistema preparado para futuras expansões.

ABSTRACT

This report describes the internship carried out at ValGrupo, focused on developing technological solutions to improve the efficiency of internal audit and management processes. The main challenge was updating the Management Portal, developed in Laravel, and creating a new Mobile Application using Flutter to ensure flexibility and efficiency in the field.

The need to optimize the audit process was identified, as it was previously manual and involved high paper consumption. As a methodology, an agile approach was adopted, using Scrum, for the continuous and incremental development of the required functionalities. The Mobile Application was designed to support offline use, allowing employees to conduct audits in locations without Internet access, while additional functionalities were added to the Portal to manage and evaluate the audit process. The Portal's API was also enhanced to synchronize data from the Mobile Application.

The results achieved were positive, with the successful development of both platforms, leading to a significant improvement in audit efficiency, a reduction in paper consumption, and a system prepared for future expansions.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	Application Programming Interface.
APK	Android Application Package.
CORS	Cross-Origin Resource Sharing.
CRUD	Create, Read, Update, and Delete.
CSRF	Cross site request forgery.
CSS	Cascading Style Sheets.
HTML	Hyper Text Markup Language.
HTTP	Hypertext Transfer Protocol.
iOS	iPhone Operating System.
IoT	Internet of Things.
JSON	JavaScript Object Notation.
MPA	Multi Page Application.
MVC	Model–view–controller.
ORM	Object Relational Mapper.
PHP	Hypertext Preprocessor.
SPA	Single Page Application.
SSR	Server-Side Rendering.

URL Uniform Resource Locator.

1

INTRODUÇÃO

O presente trabalho foi realizado no âmbito da Unidade Curricular (UC) de Estágio do Mestrado em Engenharia Informática - Computação Móvel. O objetivo principal consistiu em desenvolver uma Aplicação Móvel de modo a melhorar a eficiência e a eficácia de vários processos realizados pelo ValGrupo [1], que consistem num grupo de empresas de diversas áreas. Este projeto interno do grupo ValGrupo surge da necessidade de atualizar e expandir as capacidades das ferramentas existentes, respondendo às novas exigências tecnológicas e operacionais.

A utilização de aplicações internas é uma prática comum em muitas empresas, especialmente em setores onde a mobilidade e a acessibilidade são cruciais. Estas aplicações permitem que os colaboradores da empresa realizem diversas tarefas de forma eficiente, independentemente da sua localização. No contexto do ValGrupo, as vantagens de uma aplicação interna incluem a possibilidade de mobilidade, permitindo aos colaboradores executar tarefas fora do escritório. Além disso, garantem acesso *offline*, essencial em locais onde o acesso à *internet* é limitado ou inexistente, permitindo que as operações continuem sem interrupções. Permitem também melhorar a eficiência, reduzindo a necessidade de papel e permitindo uma recolha e processamento de dados mais rápidos e precisos. Adicionalmente, aplicações otimizadas para dispositivos móveis podem funcionar eficientemente com menos consumo de energia, uma vantagem importante em áreas com acesso limitado a recursos energéticos.

Atualmente, o ValGrupo utiliza uma Aplicação Web interna (Portal Web de Gestão do ValGrupo) que permite a realização de diversas tarefas administrativas e operacionais. Esta aplicação é robusta e funcional, porém, a sua dependência de uma conexão constante à *internet* limita a sua eficácia em cenários onde a conectividade é instável ou inexistente. Além da Aplicação Web, existe também uma Aplicação Móvel simples que foi desenvolvida para suportar algumas funções básicas para motoristas que se encontram constantemente em deslocação. No entanto, esta Aplicação Móvel apresenta várias limitações em termos de funcionalidades, usabilidade e performance, não conseguindo atender completamente às necessidades dos utilizadores.

1.1 MOTIVAÇÃO

No contexto específico do ValGrupo, surge assim a necessidade de desenvolvimento de uma nova Aplicação Móvel, motivada pela identificação de várias limitações na aplicação atual e pelas oportunidades significativas de melhoria que uma nova solução pode oferecer. Além disso, esta nova Aplicação Móvel está integrada com o Portal Web do ValGrupo, que comunica diretamente com a aplicação, facilitando a sincronização de dados e a gestão centralizada de informações, promovendo uma maior eficiência nos processos de auditoria e operacionais.

1.1.1 *Limitações da Aplicação Móvel Existente*

As principais limitações da aplicação existente podem ser destacadas nos seguintes pontos:

- Usabilidade: A aplicação atual apresenta uma interface de utilizador pouco intuitiva, o que dificulta a navegação e a execução de tarefas, especialmente para novos utilizadores;
- Funcionalidades: Existem funcionalidades cruciais que não são suportadas pela aplicação atual, como a integração com novas ferramentas de auditoria e a capacidade de trabalhar *offline*;
- Tecnologia: A aplicação atual foi desenvolvida utilizando a *framework* Xamarin [2], que em breve deixará de ter suporte oficial. Esta descontinuação implica potenciais riscos de segurança e falta de atualizações, tornando a aplicação obsoleta e vulnerável.

1.1.2 *Vantagens da Nova Aplicação Móvel*

Dando continuidade às limitações identificadas na Aplicação Móvel atual, as vantagens da nova Aplicação Móvel destacam-se nos seguintes aspetos:

- Melhoria da Experiência do Utilizador: A nova aplicação será desenvolvida com foco na usabilidade, garantindo uma interface mais amigável e intuitiva;
- Novas Funcionalidades: Serão adicionadas novas funcionalidades que respondem diretamente às necessidades identificadas pelos utilizadores, como a

sincronização em tempo real e a capacidade de armazenamento local temporário para operações *offline*;

- Integração com Sistemas Existentes: A nova aplicação será projetada para integrar-se facilmente com os sistemas e ferramentas já em uso no ValGrupo, proporcionando uma experiência mais fluída e consistente;
- Reforçar da Segurança: Serão implementadas medidas de segurança mais robustas para proteger os dados sensíveis manipulados pela aplicação, utilizando padrões de segurança mais recentes.

Estas melhorias não só eliminam as limitações atuais, como também permitirão ao ValGrupo aproveitar novas tecnologias e melhorar significativamente a eficiência e eficácia dos seus processos. Esta modernização é essencial para manter a competitividade e a qualidade dos serviços prestados pelo grupo.

1.2 OBJETIVOS

Um dos objetivos iniciais do estágio foi o desenvolvimento de uma Aplicação Móvel para o controlo e realização de auditorias internas do grupo, de modo a prevenir futuras multas ou problemas com a entidade externa responsável pela regularização das regras e condições dos locais onde os animais são criados. Para tal foi necessário:

- Analisar a Aplicação Móvel existente;
- Analisar o Portal de Gestão existente;
- Identificar o software a usar para o desenvolvimento da nova Aplicação Móvel;
- Identificar o sistema de base de dados local para a nova Aplicação Móvel;
- Implementar as funcionalidades necessárias na Plataforma Web (Portal Web de Gestão do ValGrupo);
- Desenvolver a Aplicação Móvel atendendo às regras de boas práticas;
- Apoiar os restantes membros da equipa de desenvolvimento na implementação da Aplicação Móvel.

1.3 ESTRUTURA DO DOCUMENTO

No Capítulo 2, referente ao Enquadramento, é feita uma caracterização do ValGrupo, destacando o seu papel nos diversos setores em que atua. Além disso, o capítulo

explora brevemente os projetos tecnológicos existentes no ValGrupo, incluindo o Portal de Gestão e a Aplicação Móvel, que foram fundamentais no alavancar e desenvolvimento das novas funcionalidades.

A seguir, no Capítulo 3, sobre a Metodologia e Ferramentas, são abordadas as metodologias aplicadas e como foram adaptadas no contexto do ValGrupo. Também são discutidas as principais ferramentas utilizadas e caracterizados com mais detalhe os projetos existentes. Por fim neste capítulo são também referidos os requisitos identificados e é descrita a especificação tecnológica, tanto do Portal de Gestão, como da Aplicação Móvel.

No Capítulo 4, que acorda o desenvolvimento do Portal de Gestão, é apresentada a estrutura do Portal e as etapas do desenvolvimento das novas funcionalidades, na qual é analisada a criação de uma API para suporte à nova Aplicação Móvel e o desenvolvimento das funcionalidades identificadas.

De seguida, no Capitulo 5 é abordado o desenvolvimento da Aplicação Móvel, tendo em conta a Aplicação Móvel já existente, onde se detalha o processo de transição tecnológica e onde são descritas as várias funcionalidades e estrutura da nova aplicação, e como estas foram ajustadas para proporcionar uma *interface* intuitiva para o utilizador. Além disso, discute-se como foi executado o processo de testes.

Por fim, o documento termina com uma reflexão sobre o impacto geral do estágio, tanto a nível pessoal como profissional e onde são também propostas melhorias e sugestões para futuras expansões, tanto na Aplicação Móvel, como no Portal de Gestão.

2

ENQUADRAMENTO

Neste capítulo, é abordada a caracterização detalhada da Entidade de Acolhimento, nomeadamente a ValGrupo, que serviu de palco ao estágio que fundamenta este relatório de estágio de mestrado, assim como a descrição geral do projeto desenvolvido durante o período de estágio. Esta análise visa fornecer uma compreensão aprofundada do contexto organizacional e do ambiente operacional em que o projeto realizado no decorrer do estágio foi inserido, destacando a relevância da entidade e a sua contribuição para os objetivos do estágio. Além disso, será apresentada uma visão abrangente do projeto já existente que impactou as tarefas a desenvolver no decorrer do estágio.

2.1 CARACTERIZAÇÃO DA ENTIDADE DE ACOLHIMENTO

O ValGrupo, entidade responsável pelo estágio abordado neste documento, é um conglomerado composto por aproximadamente 27 empresas, com participação em outras 9. A empresa líder, ValSabor, denominada anteriormente por Carnes Valinho, fundada em 1989, foi adquirida em 2002 pelo atual gerente. O sucesso dessa primeira empresa impulsionou a aquisição e criação de novas empresas, culminando na formação do ValGrupo.

As atividades das empresas do grupo abrangem diversas áreas de negócios, como transporte, construção de pré-fabricados, serviços contabilísticos, comercialização, produção de rações e criação animal, abrangendo suínos, bovinos e aves, como é possível observar na Figura 1. O foco principal do grupo é a produção e comercialização de produtos relacionados à criação animal.

O ValGrupo mantém uma considerável quantidade de clientes, incluindo nomes como Intermarché, Sonae, Lidl, entre outros, e possui um acordo de exportação para a China, algo que não é muito comum neste setor em Portugal, uma vez que em fevereiro de 2023 existiam apenas 9 empresas com permissão para tal [3].

Atualmente, o grupo conta com mais de 1700 colaboradores, uma frota de aproximadamente 370 veículos, mais de 350 explorações de criação animal e 7 fábricas de

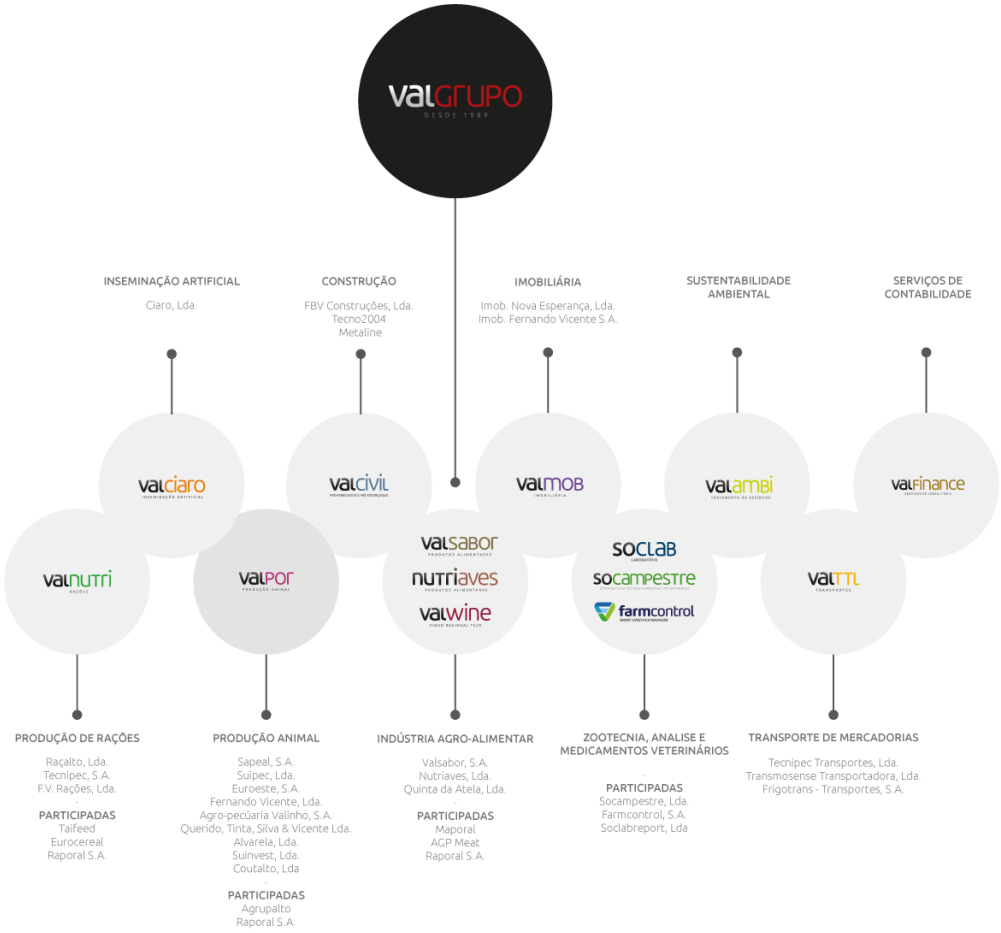


Figura 1: Organograma ValGrupo [1]

ração, bem como outros edifícios que suportam diversos modelos de negócios, como fábricas de construção de pré-fabricados, matadouros, entre outros. Essa estrutura robusta resulta num volume de negócios significativo, atingindo a marca de 1000 milhões de euros, Figura 2.

O ValGrupo encontra-se num período de expansão, com obras significativas em andamento. Essa iniciativa visa não apenas aumentar a capacidade de produção, mas também fortalecer a presença do grupo nos mercados em que atua e aumentar a quantidade de colaboradores. Esta expansão reflete o compromisso contínuo do ValGrupo com o crescimento sustentável e a adaptação às exigências do mercado bem como aos avanços da tecnologia.

Figura 2: Infograma ValGrupo [1]

A complexidade dos diversos projetos e empresas dentro do grupo impulsionou a criação de um departamento de Informática, dividido em duas vertentes, sendo estas o desenvolvimento de software e assistência técnica.

O trabalho desenvolvido no âmbito do trabalho de mestrado centrou-se na àrea de desenvolvimento, mas é bom destacar que a equipa de assistência técnica é responsável por reparar e instalar sistemas relacionados ao hardware, muitos dos quais estão diretamente ligados ao desenvolvimento de software, como câmeras, contadores de animais, entre outros, nos quais se incluem na Internet das Coisas (IoT).

No que diz respeito ao desenvolvimento de software, a equipa é composta por 8 programadores, todos do sexo masculino, com uma idade média de 27 anos. No entanto, há planos para expandir a equipa após a conclusão das obras de expansão.

Atendendo ao contexto atual, a empresa permitiu que o trabalho realizado em contexto de estágio ocorresse em regime híbrido, à semelhança do que é permitido a outros colaboradores da organização. O regime híbrido implicou a deslocação presencial às instalações da empresa dois dias da semana, realizando o restante trabalho de forma remota.

2.2 CARACTERIZAÇÃO GERAL DOS PROJETOS TECNOLÓGICOS EXISTENTES

O ValGrupo empreende vários projetos que englobam o desenvolvimento de software, todos alinhados com a sua atividade principal, sendo digno de destaque o Portal de Gestão e uma Aplicação Móvel, já em funcionamento.

O Portal de Gestão representa uma peça central para o grupo, estando o mesmo em constante desenvolvimento. O seu principal objetivo é unificar o grupo numa plataforma única, funcionando efetivamente como o "cérebro" do mesmo. Este portal permite visualizar estatísticas de diversos departamentos, desde finanças e gestão até ao suporte de funções mais operacionais, nomeadamente ao nível da contagem automatizada de animais. Além disso, estabelece conexões com equipamentos englobados na Internet das Coisas (IoT), possibilitando a execução de ações como pesagem de animais em balanças, entre outras funcionalidades vitais aos ecossistemas da área de atuação da empresa. Ressalva-se que o portal permite a integração com outro pacotes de software, nomeadamente com suporte a funções ligadas à componente de finanças, incluindo *Eticadata* [4] e *SAP* [5], dependendo das necessidades específicas de cada empresa do grupo.

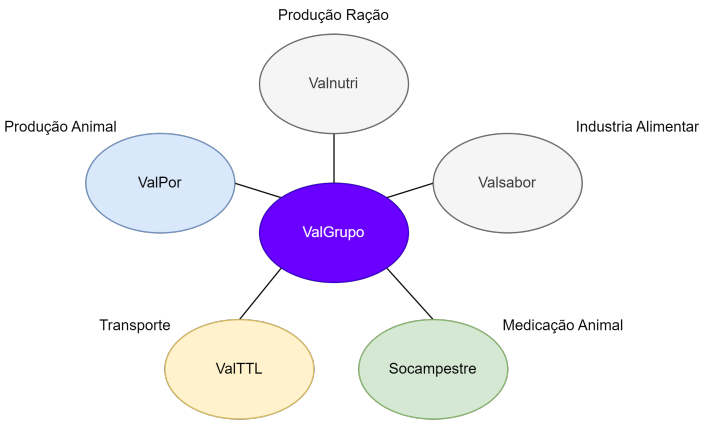


Figura 3: Organograma do Portal

Conforme referido, o Portal tem como objetivo apoiar as empresas pertencentes ao ValGrupo, sendo que estas operam em distintos setores, como demonstrado na Figura 3. No âmbito do trabalho descrito neste documento, algumas empresas merecem destaque devido ao volume e especificidade do desenvolvimento de software efetuado para as mesmas.

A ValPor corresponde a uma das empresas que integra o grupo sobre a qual recai um maior destaque nas funções operacionais. Através do Portal, é possível

gerir e monitorizar os animais nas várias explorações, bem como facilitar o processo de auditorias, sendo este um dos enfoques do trabalho desenvolvido no âmbito do estágio. No contexto da ValSabor, o Portal desempenha um papel crucial na visualização e consulta dos acessos nas portarias, melhorando a segurança e controlo de acesso. Por sua vez, a Socampestre beneficia do Portal para o controlo dos pedidos e encomendas de medicação animal, atendendo às necessidades específicas do sector agropecuário. No caso da ValNutri, o Portal oferece suporte à gestão e organização das rotas e dos locais de carga e descarga das rações. Finalmente, a ValTTL, embora não seja a maior empresa do grupo, distingue-se no sector de gestão de artigos diversos, como peças de máquinas e veículos, parafusos, e máquinas de construção civil, gerindo o maior volume de artigos dentro do grupo. O Portal permite a esta empresa gerir a frota de veículos, as requisições de veículos e os artigos de uso geral.

É importante salientar que, apesar destas empresas serem atualmente as principais beneficiárias das funcionalidades do Portal que integra as diversas componentes operacionais do grupo, existem também desenvolvimentos presentes neste Portal que permitem operações transversais às diversas empresas e que assim beneficiam outras empresas do grupo.

Inicialmente, o Portal estava exclusivamente implementado em *Laravel* [6], abrangendo tanto o *Backend* quanto o *Frontend*. No entanto, em termos tecnológicos, atualmente verifica-se uma significativa evolução em curso, com a aposta tecnológica envolvendo a migração progressiva do *Frontend* para *Vue* [7]. Essa transição visa facilitar futuras atualizações no Portal e ajustes que possam vir a ser necessários fazer, verificando-se que o processo de migração do Portal se encontra atualmente em mais de metade da sua conclusão. Apesar da migração do *Frontend* para *Vue*, em termos tecnológicos os responsáveis pela equipa de desenvolvimento optaram, nesta fase por manter o *Backend* em *Laravel*. Esta decisão prende-se com o facto de continuar a ser uma forte aposta no que toca ao desempenho, segurança e escalabilidade, oferecendo um vasto ecossistema de pacotes de software e uma comunidade de suporte, que contribui para que seja possível uma evolução constante das funções a suportar pelo sistema. Isto leva a que este Portal também funcione como suporte para a Aplicação Móvel desenvolvida no âmbito do trabalho de estágio.

Na Figura 4 é possível visualizar a *dashboard* do Portal de Gestão, onde à sua esquerda se encontra uma lista com os diferentes módulos e que podem conter sub-módulos, como é possível observar no menu das Consultas. Este formato de *dashboard* proporciona uma fácil escalabilidade e adição de novas funcionalidades. Cada colaborador na ValGrupo desempenha um papel específico dentro do grupo, e as opções disponíveis na lista são ajustadas de acordo com as permissões atribuídas

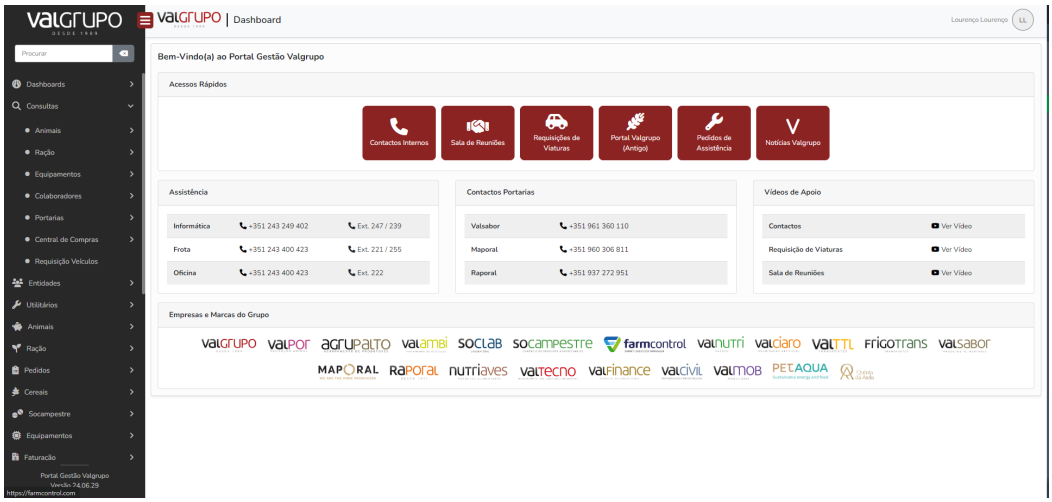


Figura 4: *Dashboard* do Portal de Gestão

a cada função, ou seja, um colaborador "comum" não terá tanta opção como um administrador.

Adicionalmente, o grupo conta também com a existência de uma Aplicação Móvel desenvolvida utilizando a *framework Xamarin* [2], que tem como objetivo o suporte na gestão do transporte de suínos e no transporte de rações. Essa aplicação estabelece uma conexão com o *Backend* do Portal de Gestão, sendo a lógica principal tratada no mesmo, permitindo a modificação de dados em tempo real. Na Figura 5 é possível observar a página inicial da Aplicação Móvel.

Figura 5: Aplicação Móvel Existente

O trabalho de estágio foi desenvolvido abrangendo as necessidades identificadas tanto no Portal de Gestão como na transição tecnológica da Aplicação Móvel, dois ecossistemas tecnológicos essenciais para a empresa. Estas plataformas desempenham um papel fundamental na operação e gestão das atividades da ValGrupo, sendo indispensáveis para a realização dos processos internos.

3

METODOLOGIA E FERRAMENTAS

Neste capítulo, será detalhada a abordagem metodológica adotada para o desenvolvimento da nova Aplicação Móvel, e alterações ao Portal de Gestão, bem como as ferramentas utilizadas. Começa por analisar as metodologias ágeis aplicadas, com foco no Scrum e Kanban, e a utilização do Jira [8] para a gestão de tarefas, e do Bitbucket [9] para o controlo de versões. De seguida é efetuada uma análise dos projetos existentes, identificando as limitações e áreas de melhoria. É também efetuado o levantamento de requisitos, incluindo a definição de perfis de utilizador e os requisitos específicos para o Portal e a Aplicação Móvel. Por fim, aborda a especificação tecnológica, descrevendo as tecnologias e arquiteturas escolhidas para o desenvolvimento da aplicação.

3.1 METODOLOGIA

Durante o decorrer do estágio, a metodologia de desenvolvimento adotada foi baseada nos princípios ágeis. Este tipo de abordagem é caracterizada pela sua flexibilidade e capacidade de adaptação a mudanças, uma vez que isso é algo muito recorrente no grupo. Posto isto, nesta secção serão abordadas as tecnologias e ferramentas utilizadas no decorrer do estágio que permitiram planear e executar as diversas tarefas apresentadas.

3.1.1 *Scrum*

A metodologia *Scrum* [10] é amplamente adotada no desenvolvimento de software em diversos projetos. Foi inicialmente introduzida na década de 1990 por Jeff Sutherland e Ken Schwaber na conferência OOPSLA (*Object-Oriented Programming, Systems, Languages and Applications*), nos Estados Unidos e, ao longo dos anos, passou por várias adaptações que fortaleceram ainda mais a sua eficácia. Essas adaptações permitiram que o *Scrum* se tornasse numa metodologia sólida e flexível, capaz de se ajustar às necessidades específicas de diferentes projetos e equipas. Esta

metodologia define um conjunto de orientações e princípios base que permitem às equipas de desenvolvimento adaptarem-na às suas necessidades específicas. Esta flexibilidade é um dos pontos fortes do *Scrum*, pois direciona a responsabilidade para a equipa, incentivando a adaptação e a personalização das práticas para melhor suportar diferentes tipos de projetos, mantendo sempre os princípios fundamentais da metodologia.

3.1.1.1 *Estrutura da Equipa*

Para que seja possível implementar esta metodologia é necessário possuir uma estrutura de equipa capaz de suportar a constante necessidade de suporte de novas funcionalidades, para isso numa metodologia *Scrum* existem 3 entidades fundamentais:

- ***Product Owner*:** Responsável por representar os interesses do cliente ou clientes, através do levantamento de requisitos que deverá escrever no *backlog*, e priorizar a lista de tarefas a serem executadas para o desenvolvimento do produto. Este papel geralmente é desempenhado apenas por uma pessoa.
- ***Scrum Master*:** É a peça central, é quem promove o entendimento e a adesão dos princípios da metodologia *Scrum*, tanto para a equipa de desenvolvimento como para a empresa, e é quem comunica entre as diferentes entidades. Por vezes é considerado o braço direito do *Product Owner* ajudando o mesmo a encontrar, e a melhorar as técnicas de desenvolvimento de software com o objetivo de tornar a equipa mais eficiente. Geralmente é um papel desempenhado por uma pessoa.
- **Equipa de Desenvolvimento:** Grupo multifuncional responsável por planear e implementar as tarefas que lhe foram atribuídas, tendo que indicar quando acaba essa execução colocando a tarefa como *"done"*. Durante o decorrer do estágio, integrei-me de forma ativa neste grupo, contribuindo para o planeamento e implementação das tarefas, garantindo que os objetivos fossem atingidos com sucesso.

3.1.1.2 *Abordagem iterativa e incremental*

Esta metodologia segue uma abordagem iterativa e incremental, que é o princípio central do desenvolvimento ágil, permitindo que o processo se ajuste e evolua conforme as necessidades do projeto e dos seus utilizadores. Esta abordagem divide o projeto em ciclos menores, conhecidos como *Sprints*, que duram tipicamente de 2 a 4

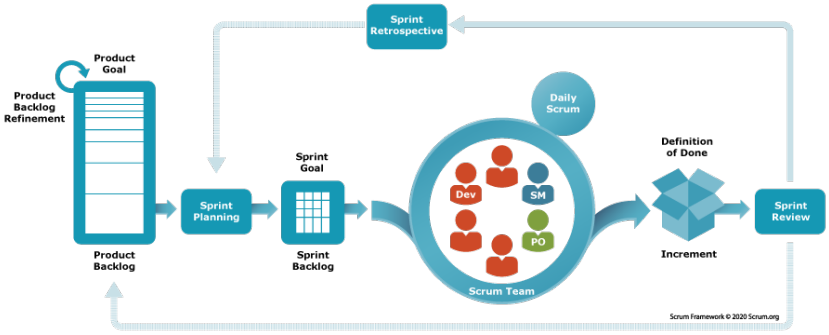


Figura 6: Estrutura *Scrum* [11]

semanas. Durante cada *Sprint*, a equipa passa por todas as fases do desenvolvimento de software, desde o planeamento até a sua documentação, culminando na entrega de um incremento ao projeto. Este por vezes pode ou não conter todas as funcionalidades finais definidas na *Sprint* [11].

A cada iteração, ou *Sprint*, o projeto em desenvolvimento beneficia da adição de novas funcionalidades, construídas com base no trabalho anterior, resultando num crescimento incremental do projeto. Desta maneira permite que a equipa de desenvolvimento se adapte rapidamente às mudanças no produto e também facilita o *feedback* contínuo dos *stakeholders*, sendo estes os utilizadores ou donos do projeto. Assim é possível assegurar que o produto final está alinhado com as expectativas e necessidades do mercado.

Outro ponto forte desta metodologia é a sua flexibilidade, que permite ajustes frequentes à medida que surgem novas informações ou alterações aos requisitos do projeto. Isto ajuda na gestão de riscos, permitindo que a equipa identifique e resolva problemas em fases iniciais, minimizando os seus impactos no desenvolvimento geral. Além disso, a visibilidade do progresso em cada *Sprint* aumenta a confiança entre a equipa de desenvolvimento, e a entrega contínua de incrementos de produto mantém a equipe motivada e focada.

3.1.2 *KanBan e Jira*

O *Kanban* é uma abordagem visual para gestão de projetos que tem como objetivo melhorar a eficiência do trabalho e a entrega contínua. Inicialmente criado pela *Toyota* [12], o *Kanban* baseia-se em princípios ágeis para otimizar processos e responder rapidamente às mudanças contínuas. É caracterizado por um quadro

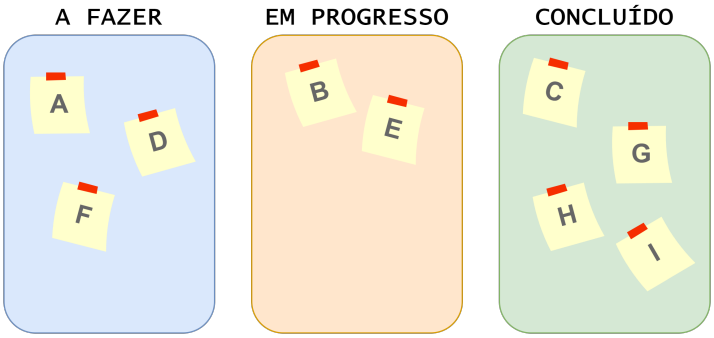


Figura 7: Estrutura *Kanban*

Kanban, dividido em colunas que representam diferentes fases do processo de trabalho (por exemplo, "A Fazer", "Em Progresso", "Concluído"), como é possível visualizar na Figura 7. As tarefas são representadas através de cartões que se movem entre as colunas conforme as tarefas avançam no processo. Isto proporciona uma visão mais clara do progresso das tarefas, ajuda a identificar problemas e dificuldades, e facilita a colaboração da equipa.

O *Jira* [8] é uma ferramenta desenvolvida pela *Atlassian* [13]. É uma plataforma completa para gestão de projetos e desenvolvimento de software. Uma das suas maiores características é suportar metodologias ágeis, como *Scrum* e *Kanban*. O *Jira* destaca-se pela sua capacidade de adaptação às necessidades variadas de equipas de desenvolvimento de software, permitindo a criação, atribuição e gestão de tarefas e *bugs* através de uma interface intuitiva e flexível. Cada tarefa dentro do *Jira*, conhecida como "*issue*", pode ser personalizada para se adequar ao contexto específico do projeto, e é possível adicionar diversos parâmetros ao *issue* incluindo tipo de tarefa, *status*, prioridade e descrições detalhadas.

Além de facilitar a gestão de projetos, o *Jira* disponibiliza funcionalidades avançadas de elaboração de relatórios e análise, apoiando as equipas na monitorização do seu progresso, avaliação do desempenho e identificação de áreas a aperfeiçoar nos seus projetos. Entre estas funcionalidades, destacam-se os épicos, que permitem organizar e monitorizar um grupo de tarefas, facilitando o planeamento, a priorização e o rastreamento do progresso em grandes projetos. Ao criar um "*issue*", é possível associá-lo a um épico, garantindo que todas as tarefas relacionadas com um objetivo maior estão agrupadas e facilmente rastreáveis. Os épicos ajudam a manter uma visão clara e organizada do trabalho. A plataforma é igualmente notável pela sua capacidade de se integrar com um vasto leque de ferramentas de desenvolvimento e controlo de versões, como *GitHub* [14], *GitLab* [15] e *Bitbucket* [9], bem como ferramentas de integração contínua e entrega contínua (CI/CD),

incluindo *Jenkins* [16]. Além disso, a integração com ferramentas de comunicação, como o *Slack* [17], facilita a interação e coordenação entre equipas. Estas integrações, aliadas a funcionalidades de automação, contribuem para acelerar e otimizar o fluxo de trabalho.

Em suma, o *Jira* é uma ferramenta muito completa que permite implementar um sistema *Scrum* e *Kanban*, e possui diversas integrações com o objetivo de melhorar o desenvolvimento de um projeto de software.

3.1.3 *BitBucket*

O *Bitbucket* [9] é uma plataforma de repositórios *Git* que oferece várias funcionalidades focadas nas equipas de desenvolvimento de software. Desenvolvido inicialmente em 2008, e posteriormente adquirido pela *Atlassian* em 2010, o *Bitbucket* destaca-se pela sua integração com outras ferramentas desta empresa, como o *Jira* (referido anteriormente), o *Bamboo*[1] (*continuous delivery*) e o *Trello*[2] (gestão global de projeto), oferecendo um ecossistema rico e coeso para gestão de projetos de software.

Além das funcionalidades básicas de repositório, como controle de versão e revisão de código, o *Bitbucket* também oferece *pipelines* de CI/CD (Integração Contínua e Entrega Contínua) integradas, permitindo às equipas automatizar os testes e a distribuição de software diretamente a partir de uma plataforma. Esta integração profunda com ferramentas de automação e gestão de projetos faz do *Bitbucket* uma escolha popular para empresas e projetos que procuram uma solução completa.

O *Bitbucket* suporta tanto repositórios privados, como públicos e oferece uma variedade de planos, incluindo, à data da escrita deste documento, opções gratuitas e pagas, ajustando-se assim às necessidades de diferentes tipos e tamanhos de equipas. A interface de utilizador é projetada para facilitar a colaboração entre os membros da equipa, com funcionalidades como *pull requests* e revisões de código integradas, melhorando a qualidade do software e a eficiência do processo de desenvolvimento.

Em resumo, o *Bitbucket* é uma plataforma robusta e versátil para gestão de repositórios *Git*, que se distingue pela sua integração com outras ferramentas da *Atlassian* e pelas suas funcionalidades avançadas de CI/CD, tornando-a uma opção sólida para equipas de desenvolvimento de todos os tamanhos.

1 https://www.atlassian.com/software/bamboo
2 https://trello.com/

3.1.4 *Aplicação Prática da Metodologia no Projeto*

Posto isto, no decorrer do estágio, durante o desenvolvimento dos diversos projetos foram adotadas as metodologias ágeis, *Scrum* e *Kanban*, contando com o apoio das ferramentas *Jira* e *Bitbucket* para a gestão de tarefas e controlo de versões, respetivamente.

Apesar da equipa de desenvolvimento na qual estive integrado não seguir à risca a metodologia *Scrum*, foi adotada a ideologia das reuniões diárias. Numa fase inicial do estágio existia também a reunião de *Sprint* de 3 em 3 semanas, na qual era referido o que tinha sido feito durante o *Sprint* atual e onde eram identificadas quais as tarefas a serem realizadas até à próxima *Sprint*. No entanto, a mesma foi deixando de existir, mas continuaram a existir os ditos *Sprints* na ótica de ir dividindo as tarefas em blocos menores. À semelhança da metodologia *Scrum* a equipa de desenvolvimento da empresa também integra um *Scrum Master*, que no nosso caso também é programador. No entanto, o *Product Owner* não foi estabelecido de maneira tradicional, pois não era ele o responsável por criar e gerir o *Backlog*, e sim o *Scrum Master* com a contribuição direta, por vezes da própria equipa de programadores, como foi no caso da definição das tarefas envolvidas na implementação da Aplicação Móvel, bem como do desenvolvimento de funcionalidades acrescida no Portal de Gestão.

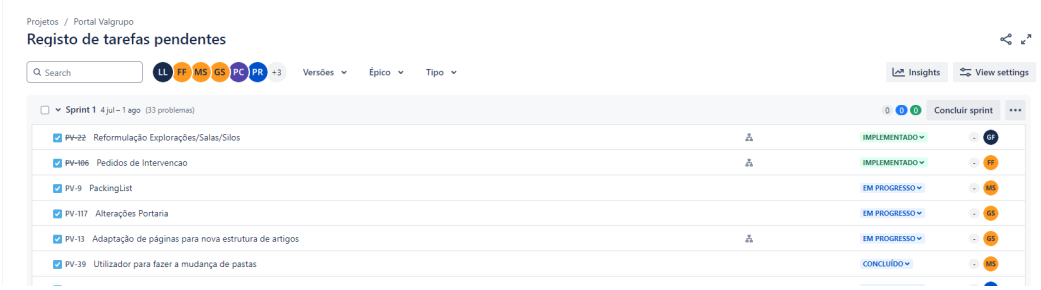


Figura 8: *Backlog* do Jira

Seguindo então a lógica anterior, o *Scrum Master* ou o próprio programador, numa fase inicial criam os *issues* relativos ao projeto/módulo em que vão trabalhar, sendo este processo suportado em termos de metodologia de desenvolvimento pelo registo no *backlog*, usando a ferramenta *Jira*, conforme demonstra a Figura 8. Nesse *issue* a pessoa associa-o a um épico, como referido na Secção 3.1.2, este épico no contexto da empresa corresponde ao projeto/módulo. Para facilitar a previsão do tempo necessário para implementar uma tarefa, anteriormente utilizava-se um sistema de categorização de dificuldade dos *issues* com base na sequência de Fibonacci.

Esse método permitia calcular, de forma aproximada, o tempo necessário para completar uma tarefa, onde 3 pontos equivaliam a aproximadamente 1 dia. No entanto, decidimos alterar essa abordagem e passámos a colocar a estimativa de tempo diretamente no próprio *issue*. Quando a tarefa era finalizada, registávamos o tempo efetivamente gasto. Isto permitiu um melhor controlo e monitorização de atrasos apesar de ser mais complicado efetuar uma previsão antecipada das mesmas. Após criar o *issue*, o mesmo é atribuído à pessoa responsável por implementá-lo.

Após a definição das tarefas o utilizador cria um *branch* principal no *BitBucket* através de *master* com o nome do projeto, e coloca o estado da tarefa como "Em Progresso" arrastando a tarefa no quadro para o estado pretendido, conforme visível na Figura 9. Posteriormente é criado outro *branch* através do *issue* que pretende implementar. Essa criação é feita pelo *Jira* que vai utilizar a integração com o *BitBucket* que permite que ao se escolher o repositório correto, e o *branch* principal como base, originar um *branch* onde o seu nome inicia com o código do *issue*, seguido da sua descrição. No entanto, esta nomenclatura pode ser alterada caso seja necessário. Essa nomenclatura permite no ato de revisão do código perceber logo a que tarefa este está relacionado.

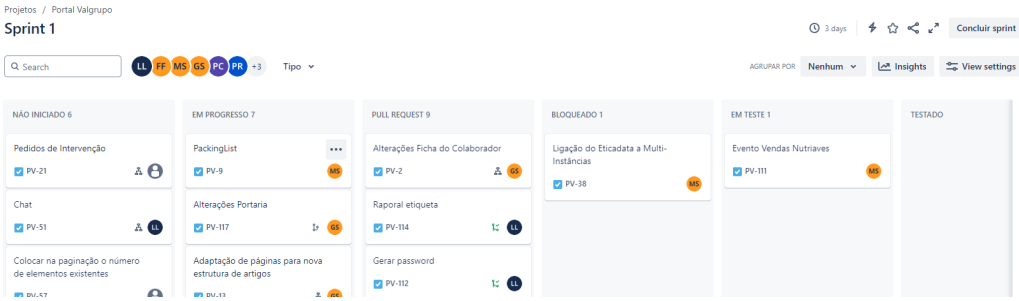


Figura 9: Quadro do Jira

Assim que a tarefa for implementada é feito um *commit* e um *push* para o repositório remoto, e de seguida é criado um *pull request*, no qual o objetivo é permitir o *merge* com o seu *branch* principal. Nesse *pull request* são selecionados todos os membros da equipa de desenvolvimento, incluindo o *Scrum Master*. Nesse momento é também alterado o estado da tarefa para *pull request*. Após ser efetuada a revisão pelo *Scrum Master*, e por pelo menos dois programadores, é possível efetuar o *merge*, sendo este feito pelo *Scrum Master*.

De modo a facilitar a visualização dos *Pull Requests*, visto que por vezes se tornavam excessivos, o *Scrum Master* seleciona cerca de 3 a 4 *Pull Requests* por dia, que ficam sinalizados e são comunicados ao grupo de desenvolvimento, para que esses sejam vistos e aprovados no próprio dia.

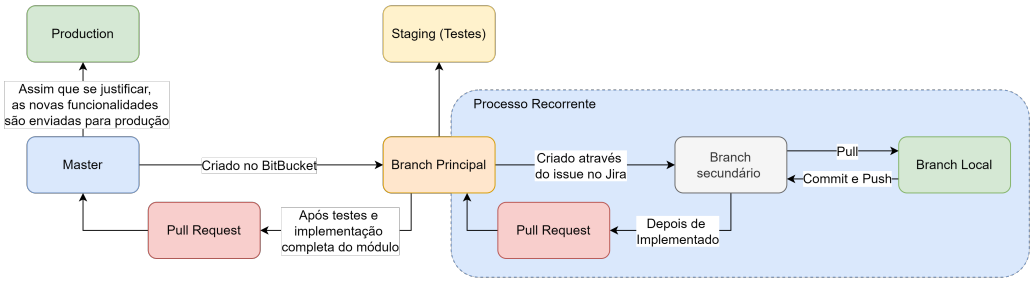


Figura 10: Fluxo de Trabalho

O *branch* principal vai sendo atualizado com o que está no *branch* do *master* periodicamente, de modo a manter este *branch* atualizado. Assim que este contenha toda a implementação (esteja finalizado), ou parte da implementação solicitada, é enviado para o *branch master* onde é revisto novamente de uma forma não tão exaustiva. Este *branch* principal também é usado por vezes para testes, uma vez que existem na empresa cerca de 6 serviços Web internos com as mesmas configurações de produção, nos quais é possível hospedar o mesmo. Noutras ocasiões é também usado o *branch* de *staging* para testes mais gerais, em que depois de terminados esses testes o código é efetivamente enviado para o ambiente final de produção, este fluxo de trabalho é possível ser observado na Figura 10.

O *deploy* das novas funcionalidades é feito de forma semi manual utilizando um *script* que efetua os comandos necessários para efetuar o *deployment*. Geralmente opta-se por fazer o *deploy* por volta das 19 horas, uma vez que corresponde à altura do dia com menos utilizadores ativos, isto porque para compilar o *Vue* é necessário reiniciar o serviço, o que pode causar problemas aos utilizadores no decorrer da operação.

3.2 CARATERIZAÇÃO DOS PROJETOS DE DESENVOLVIMENTO EXISTENTES

Como referido no Capítulo 2, o ValGrupo possui um projeto global que representa uma peça central, sendo esse denominado "Portal de Gestão", o qual vai ser apresentado em maior detalhe nesta secção. Para além da caracterização deste portal será abordada a Aplicação Móvel que inicialmente existia no ecossistema da empresa para dar suporte aos motoristas de animais e motoristas de rações.

Este projeto foi desenvolvido inicialmente por uma empresa externa, mas com as recorrentes necessidades do grupo, este optou pela criação de uma equipa de desenvolvimento dedicada e interna, que tem como objetivo dar suporte às fun-

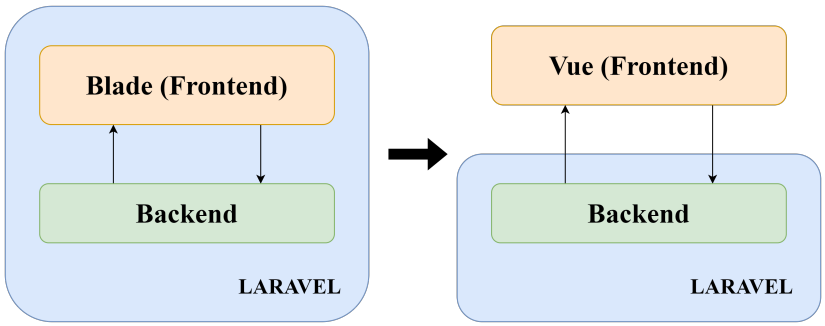


Figura 11: Transição do *Frontend*

cionalidades atuais do Portal, bem como ao desenvolvimento contínuo de novas funcionalidade que vão sendo identificadas como necessárias para o ecossistema das empresas que integram o grupo. O Portal, inicialmente encontrava-se desenvolvido na sua totalidade em *Laravel*, utilizando *Blade* para o seu *Frontend*. *Blade* corresponde ao motor de *templates* do *Laravel* [18] que combina código PHP com HTML e oferece recursos como *templates* e componentes reutilizáveis.

No entanto para permitir uma manutenção mais fácil e um aumento na fluidez da aplicação optou-se por fazer a transição do *Frontend* para *Vue.js*, aquando da transferência do projeto da empresa externa, para a equipa de desenvolvimento interna. Este processo de transferência das funcionalidades para outra plataforma tecnológica, nomeadamente a codificação do *Frontend* para *Vue*, em detrimento do *Laravel* ainda se encontra a decorrer, com o objetivo de transitar as páginas antigas desenvolvidas em *Blade* também para este novo formato, bem como implementar as novas funcionalidades já com esta ideologia.

Vue.js é uma *framework* progressiva para a construção de interfaces Web que permite criar aplicações *Single Page Application* (SPAs) [19]. Com esta *framework* é possível criar interfaces dinâmicas e reativas do lado do cliente, otimizando a experiência do utilizador através de atualizações de componentes da página, sem necessidade de recarregar a página por completo. Torna-se assim ideal para desenvolver aplicações Web modernas que requerem interatividade e desempenho.

Comparando com o *Blade*, que é um motor de PHP utilizado principalmente para gerar HTML no lado do servidor, *Vue.js* destaca-se na criação de SPAs, permitindo na mesma criar *Multi Page Applications* (MPAs) [19]. Enquanto o *Blade* apenas permite a criação de MPAs, onde cada alteração na página requer um carregamento da página do servidor. *Vue.js* destaca-se na construção de interfaces ricas e interativas dentro de uma única página, onde o conteúdo é dinamicamente atualizado conforme a interação do utilizador.

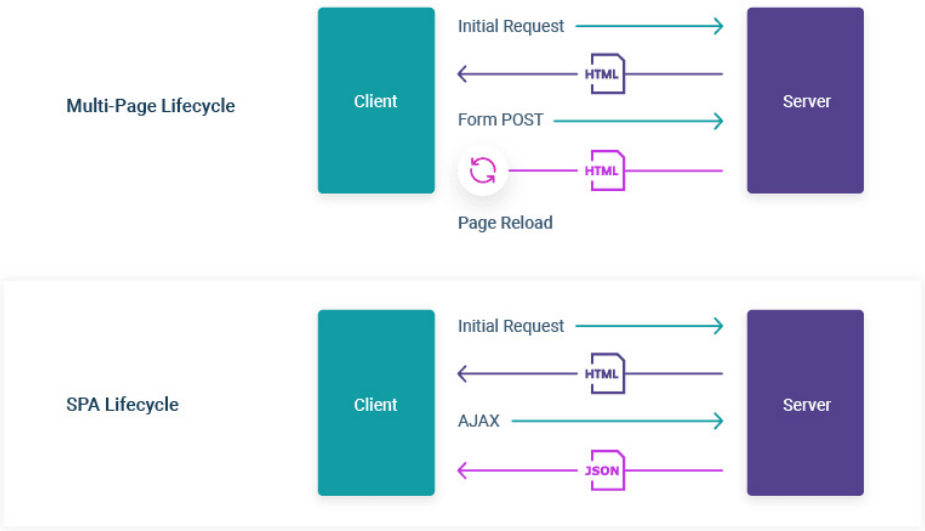


Figura 12: Comparação entre MPA e SPA [20]

A principal diferença entre SPAs e MPAs, é a abordagem para carregar e apresentar conteúdo ao utilizador. SPAs, favorecidas pelo *Vue.js*, oferecem uma experiência de utilizador fluida, com transições rápidas e sem o tradicional recarregamento da página. Isso é alcançado através da manipulação do *Document Object Model* (DOM) baseada em *JavaScript* e comunicação assíncrona com o servidor, geralmente via APIs *REST*[3] ou *GraphQL*[4].

Em contraste, MPAs, são aplicações mais alinhadas com a abordagem tradicional no desenvolvimento Web e bem suportadas pelo motor *Blade*. Carregam uma nova página do servidor a cada pedido, o que pode ser mais lento e resultar numa experiência menos fluida. No entanto, MPAs podem ser mais simples de implementar e naturalmente melhores para otimização de motores de busca (SEO), sem a necessidade de estratégias adicionais como o *Server-Side Rendering* (SSR), que as SPAs muitas vezes requerem.

Portanto, enquanto o *Blade* é uma escolha sólida para projetos que beneficiam de uma arquitetura MPA tradicional, *Vue.js* é particularmente poderoso para desenvolver SPAs reativas e interativas, proporcionando aos utilizadores finais uma experiência mais rápida e dinâmica, notando-se apenas um ligeiro atraso ao carregar a página pela primeira vez.

Em termos de Base de Dados a mesma encontrava-se em *MySql*[5] e optou-se por manter a mesma estrutura, alterando apenas o formato das colunas que se

3 https://www.ibm.com/topics/rest-apis
4 https://graphql.org/
5 https://www.mysql.com/

encontravam em Português para Inglês, caso no futuro seja necessário integrar novas pessoas na equipa que não dominem a língua portuguesa e visto que o Inglês tem vindo a tornar-se standard na indústria.

Para além do Portal de Gestão da empresa, foi-me também apresentada a Aplicação Móvel já existente, desenvolvida também pela mesma empresa externa. Esta encontrava-se desenvolvida em *C#* e *.NET*, utilizando a *framework Xamarin*.

A mesma possui um *login* e dois módulos desenvolvidos, sendo eles o transporte de rações e transporte de animais. Esta aplicação, na sua versão inicial, aquando do inicio das tarefas do estágio não possui base de dados local o que limita o seu uso apenas para locais com acesso a Internet, sendo a lógica toda aplicada do lado do *Backend*.

3.3 LEVANTAMENTO DE REQUISITOS

Pretende-se nesta secção abordar os diferentes perfis de utilizadores bem como os requisitos funcionais e não funcionais identificados para a Aplicação Móvel e para o Portal Web, ilustrando como estes foram concebidos para promover uma interação efetiva entre eles, com o objetivo de facilitar e melhorar o desempenho dos vários perfis de utilizadores.

3.3.1 *Perfis de Utilizador*

Uma vez que estes projetos são desenvolvidos de modo a dar suporte a um grupo de empresas em áreas diversificadas, existem diversos perfis de utilizadores a ter em consideração. Estes perfis são delineados, não apenas pelas suas permissões e acessos dentro do sistema, mas também pelas responsabilidades intrínsecas a cada cargo. No entanto, para os projetos descritos neste documento, e com os quais houve interação no decorrer do estágio, podemos destacar os seguintes perfis mais importantes no contexto do estágio:

- **Administrador**: Os utilizadores categorizados como Administradores detêm a capacidade de gerir e configurar as definições globais do sistema, assegurando o seu correto funcionamento. Este perfil está associado aos dirigentes do grupo e algumas pessoas do departamento informático, de modo a facilitar a correção de eventuais erros.

- **Gerente de Auditores**: A figura do Gerente de Auditores surge como um pilar fundamental na estruturação e condução das auditorias. Este perfil é encarregue de orquestrar o processo de auditoria, desde a sua fase de planeamento, até à análise dos resultados obtidos.
- **Auditores**: Por fim, o perfil dos Auditores é caracterizado pelos profissionais que executam as auditorias no terreno. Estes utilizadores são responsáveis pelo preenchimento de dados e informações diretamente nos locais designados, aplicando os critérios e procedimentos estabelecidos previamente pelo Gerente de Auditores.

Embora existam muitos outros perfis de utilizadores no sistema, estes são os mais relevantes para os projetos abordados neste documento.

3.3.2 *Funcionalidades a suportar pelo Portal de Gestão*

O Portal, apesar de assegurar todo um conjunto muito vasto de funcionalidades, no âmbito do trabalho de estágio estive envolvido na implementação de funcionalidade de suporte ao processo de auditorias. Posto isto, no Portal, são agendadas as auditorias, para depois na data da auditoria serem preenchidas através do dispositivo móvel, e posteriormente serem consultadas novamente no Portal. Para tal existe a necessidade de criar algumas funcionalidades no lado do Portal, sendo essas:

- Página para agendar auditorias e onde possa ser possível visualizar o agendamento das mesmas no formato de calendário;
- Página de consulta para marcar a "Ausência de Sede Prolongada" que corresponde a um valor cujo seu calculo é efetuado num software externo mas que necessita de ser associado à auditoria;
- Página de consulta de histórico para visualizar todas as auditorias realizadas numa determinada exploração;
- Na Página de consulta do histórico é necessário poder gerar um relatório em formato PDF contendo os dados referentes à auditoria.

3.3.3 *Funcionalidades a suportar pela Aplicação Móvel*

O objetivo da Aplicação Móvel que será desenvolvida para substituir a anterior, é numa vertente de complementar o Portal em tarefas onde um dispositivo móvel

possa facilitar, ou ate ser o único meio viável para a execução de determinadas funções. Posto isto foram identificados os seguintes requisitos:

- Página para autenticação no *Backend* com email e *password*;
- Possibilidade de utilizar a Aplicação Móvel sem acesso a Internet com base nos dados transferidos na ultima conexão com a Internet;
- Sincronizar os dados da Aplicação Móvel com os dados existentes no Portal;
- Permitir facilmente integrar novos módulos;
- Poder visualizar a versão instalada de modo a perceber se os problemas ou erros existentes podem advir do facto do utilizador possuir uma versão mais antiga da Aplicação Móvel;
- Sistema que permita detetar se existe uma nova versão da Aplicação Móvel, e caso exista, proceder a atualização da Aplicação Móvel de forma prática;
- Página com a lista de auditorias agrupadas por data, e na qual sejam apresentados os principais dados da auditoria;
- Página central de uma auditoria onde seja possível visualizar os dados da auditoria e navegar para as páginas destinadas ao preenchimento das informações relacionadas com essa auditoria;
- Página para responder a um questionário sobre a legislação em vigor. O questionário será criado no Portal e pode ser alterado a qualquer altura, mas deverá ser possível responder ao mesmo usando a Aplicação Móvel;
- Permitir anexar fotos da galeria ou tiradas no momento, caso no questionário seja selecionado "não conformidade";
- Página para responder a um questionário predefinido acerca das instalações e estado dos animais, com perguntas fixas que garantem a uniformidade e consistência das respostas;
- Página para listar os parques (local físico onde se encontram x animais) e poder visualizar o valor do peso médio referente aos questionários respondidos no ponto anterior;
- No caso de existir uma "não conformidade" criar automaticamente uma auditoria de verificação. Esta verificação é similar a uma auditoria, no entanto o intuito é apenas verificar se a "não conformidade" já foi retificada;
- Página para responder à auditoria de verificação.

3.4 ESPECIFICAÇÃO TECNOLÓGICA

Como referido no Secção 3.2, aquando da minha entrada no grupo já existia uma estrutura implementada para o portal que corresponde a uma plataforma Web, para o qual foi utilizado o *Laravel* para *Backend*, e o *Vue.js* para *Frontend* estando a base de dados de suporte ao mesmo em *MySQL*. As comunicações entre o *Frontend* e o *Backend* eram feitas utilizando uma API *REST* [21].

Para a Aplicação Móvel descrita neste documento, a qual corresponde ao objetivo principal das tarefas de estágio, foi considerado o desenvolvimento das novas funcionalidades, integrando-as na Aplicação Móvel já existente. No entanto, optou-se pela implementação de uma nova aplicação, a desenvolver totalmente de raiz na empresa. A transição tecnológica inerente à implementação da aplicação numa nova *framework* foi considerada, pois a *framework* já existente, *Xamarin* [2], estava a cair em desuso e perdeu o seu suporte dia 1 de maio de 2024 [22].

Em termos de linguagem de programação não se verificaram limitação na escolha, pelo que se optou pela análise de diferentes *frameworks* de modo a identificar qual poderia ser a transição tecnológica mais interessante de adotar. Considerando que o objetivo é permitir a utilização da aplicação nos sistemas *iOS* optou-se por considerar na análise *frameworks* com suporte multiplataforma, que permitissem compilar a aplicação, tanto para sistemas *android* como *iOS*.

Posto isto, numa fase inicial foram analisadas diferentes *frameworks* com base nos requisitos previamente identificados, destacando-se as seguintes: *React Native* [23], *Ionic Framework* [24] e *Flutter* [25]. O *Xamarin* embora sendo mencionado diversas vezes, e tendo em conta as razões referidas anteriormente, optou-se por não incluir nas *frameworks* a analisar.

O *React Native*[6], desenvolvido pelo *Facebook* (atualmente *Meta*), é uma *framework* que utiliza *JavaScript*. Diferenciando-se das abordagens baseadas em *Web Views*, o *React Native* converte componentes diretamente para código nativo, proporcionando um desempenho comparável ao de aplicações nativas. Esta *framework* destaca-se pela inclusão de uma ferramenta particularmente útil para programadores, o *Hot Reloading*. Esta funcionalidade permite visualizar alterações na aplicação em tempo real durante o desenvolvimento, logo após a alteração do código.

Contudo, uma das limitações do *React Native* é a eventual necessidade de desenvolver código específico para cada plataforma, dependendo das funcionalidades que

6 https://reactnative.dev/

se pretende implementar. Esta exigência pode aumentar a complexidade, e o tempo de desenvolvimento de projetos que visem múltiplas plataformas.

O *Ionic Framework*[7] usa como tecnologias de programação HTML, CSS e *JavaScript*. Destaca-se pela facilidade de integração com *frameworks* populares como *Angular*[8], *React*[9] e *Vue.js*[10], e oferece uma vasta gama de componentes visuais adaptáveis às diferentes plataformas. Uma das vantagens desta *framework* é oferecer uma maior rapidez no desenvolvimento, e a possibilidade de reutilizar conhecimentos de Web para criar aplicações móveis. No entanto, o desempenho pode não ser equivalente ao de aplicações nativas, especialmente em tarefas que exigem alto processamento.

O *Flutter*[11], lançado em 2017 e desenvolvido pela Google, é uma *framework* que tem experienciado um crescimento significativo, beneficiando-se do apoio de uma vasta comunidade que facilita a resolução de problemas. Esta *framework* adota a linguagem de programação *Dart* [26], introduzida em 2011 também pela Google. O *Dart* é reconhecido pela sua arquitetura baseada em *widgets*, comparáveis aos componentes de outras *frameworks*, o que facilita a modularização e reutilização de código.

Uma das principais vantagens do *Flutter* é a sua capacidade de compilar diretamente para código nativo, o que resulta num desempenho quase idêntico ao de aplicações nativas, com animações suaves e tempos de resposta rápidos. Tal como o *React Native*, o *Flutter* também suporta o *Hot Reload* [27], uma funcionalidade que agiliza significativamente o processo de desenvolvimento ao permitir que alterações no código sejam imediatamente visíveis na aplicação em execução.

Adicionalmente, o *Flutter* dispõe de um extenso repositório de *plugins* que facilitam a implementação de diversas funcionalidades, enriquecendo o ecossistema de desenvolvimento.

Uma desvantagem do *Flutter* é a necessidade de aprender *Dart*, no entanto, esta linguagem apresenta semelhanças com linguagens já conhecidas como *C#* e *Java*, o que ajuda na aprendizagem para programadores que possuam conhecimentos nessas linguagens.

Considerando todos estes aspetos, sumarizados na Tabela 1, optou-se pelo *Flutter* para o desenvolvimento da Aplicação Móvel. Esta escolha foi motivada, não só pelo

7 https://ionicframework.com/
8 https://angular.dev/
9 https://react.dev/
10 https://vuejs.org/
11 https://flutter.dev/

Tabela 1: Comparação entre React Native, Ionic Framework e Flutter

	React Native	**Ionic Framework**	**Flutter**
Linguagem	JavaScript/TypeScript	JavaScript/TypeScript	Dart
Plataformas	iOS, Android	iOS, Android, Web	iOS, Android, Web, Desktop
Performance	Alta (Performance Nativa)	Moderada (Webview)	Muito Alta (Compilado)
Popularidade	Muito Alta	Alta	Muito Alta
Comunidade	Muito Ativa	Ativa	Muito Ativa
Aprendizagem	Moderada	Fácil	Moderada
Suporte a Plugins	Amplo	Amplo	Amplo
Custo de Desenvolvimento	Médio-Alto	Baixo	Médio
Hot Reload	Sim	Não	Sim
UI Responsiva	Sim	Sim	Sim
Integração com APIs	Excelente	Boa	Excelente

crescimento e suporte contínuos proporcionados por uma empresa de renome como a Google, mas também pela robusta comunidade que já possui, e pela capacidade de criar aplicações com alto desempenho.

4

DESENVOLVIMENTO DO PORTAL - GESTÃO VALGRUPO

Dando seguimento à análise dos perfis de utilizadores e funcionalidades do sistema, o presente capítulo foca-se no desenvolvimento do Portal de Gestão para o ValGrupo. Este capítulo aborda detalhadamente os passos envolvidos na integração no projeto, e no desenvolvimento das funcionalidades do Portal, as tecnologias utilizadas, os desafios enfrentados e as soluções adotadas.

4.1 ANÁLISE DO PROJETO

Como referido anteriormente, aquando o inicio do estágio, o Portal já se encontrava em desenvolvimento, pelo que existiu a necessidade de adaptação à estrutura já existente. Deste modo, de seguida será descrita a estrutura do Projeto já existente.

4.1.1 *Estrutura de Ficheiros*

Em termos de *Backend* do Portal, é utilizada a *framework Laravel*, que segue a arquitetura MVC (Model-View-Controller)[28]. O *Laravel* já oferece uma estrutura de pastas organizada e intuitiva, sendo esta a seguida pela equipa de desenvolvimento, podendo-se destacar a seguinte estrutura:

- **App**: Contém a maior parte da lógica da aplicação;
  - **Http**: Pasta que contém Controladores, *Middlewares* e *Requests*.
    - ***Controllers***: Um controlador é uma classe que agrupa a lógica de processamento de pedidos HTTP e a geração de respostas, organizando e simplificando o código da aplicação. Os controladores respondem a pedidos específicos, executam operações como manipulação de dados, interações com o modelo e retornam vistas ou respostas JSON. Permitem uma melhor organização e reutilização do código, facilitando a manutenção e a extensão da aplicação. Por

exemplo, um controlador pode ter métodos para exibir uma lista de utilizadores, mostrar detalhes de um utilizador específico, criar novos utilizadores, atualizar informações existentes e apagar utilizadores. É o controlador que faz a ligação entre o Modelo e a *View*;

* ***Middleware***: Um *middleware* é um componente intermediário que processa pedidos HTTP antes de serem entregues aos controladores e também pode modificar as respostas antes de estas serem enviadas ao cliente. Serve para realizar tarefas como autenticação, autorização, proteção contra *Cross Site Request Forgery* (CSRF) [29], gestão de *Cross-Origin Resource Sharing* (CORS) e registo de pedidos. Essencialmente, um *middleware* inspeciona, modifica ou rejeita pedidos e respostas com base em determinadas condições, garantindo assim a segurança e o funcionamento adequado da aplicação. Por exemplo, um *middleware* pode verificar se um utilizador está autenticado antes de permitir o acesso a certas rotas;
* ***Requests***: Um *request* representa uma solicitação HTTP feita pelo cliente à aplicação, contendo informações como dados do formulário, parâmetros da URL, cabeçalhos e *cookies*. *Requests* são utilizados para validar e filtrar dados de entrada antes de serem processados pelos controladores, garantindo que a aplicação receba dados consistentes e seguros. Por exemplo, um *request* pode ser usado para validar os campos de um formulário de registo de utilizador, assegurando que os dados estão no formato correto antes de serem guardados na base de dados.

– ***Models***: Um modelo é uma classe que representa uma tabela na base de dados e encapsula a lógica de interação com essa tabela, como consultas, inserções, atualizações e eliminações de dados. Utilizando o Eloquent ORM (Object-Relational Mapping) [30], os *models* permitem uma abordagem orientada a objetos para trabalhar com dados, facilitando a criação, leitura, atualização e eliminação (operações CRUD) de registros na base de dados através de métodos intuitivos. Por exemplo, um modelo pode representar uma tabela de utilizadores e permitir operações como recuperar todos os utilizadores, encontrar um utilizador específico, ou adicionar um novo utilizador à base de dados;

- **Config**: Configurações da aplicação. Cada ficheiro dentro desta pasta corresponde a uma parte específica da aplicação, como por exemplo `database.php`

para configurações da base de dados, `mail.php` para configurações de email, etc.

- ***Database***: Gestão do base de dados.
  - ***Migrations***: Uma *migration* é uma classe que define a estrutura das tabelas na base de dados, permitindo criar, modificar e eliminar tabelas e colunas de forma programática e controlada. As *migrations* são utilizadas para gerir alterações no esquema da base de dados, garantindo que a estrutura esteja sincronizada entre diferentes ambientes de desenvolvimento e produção. Por exemplo, uma *migration* pode ser usada para criar uma tabela de utilizadores com colunas como id, nome, email e password, ou para adicionar uma nova coluna de perfil à tabela existente;
  - ***Seeders***: Um *seeder* é uma classe utilizada para popular a base de dados com dados de teste ou dados iniciais de forma programática. Os *seeders* permitem inserir registos automaticamente nas tabelas, facilitando a configuração de um ambiente de desenvolvimento com dados consistentes. Eles são especialmente úteis para preencher a base de dados com informações padrão ou amostras de dados necessários para o funcionamento da aplicação. Por exemplo, um *seeder* pode ser usado para criar um conjunto inicial de utilizadores, produtos ou categorias na base de dados.
- ***Routes***: Define todas as rotas da aplicação.
  - **web.php**: No ficheiro `web.php` são definidas as rotas que respondem a pedidos Web (HTTP) que não são API, especificando quais os controladores e as ações que devem ser executados quando determinados URLs são acedidos. Este ficheiro contém rotas que utilizam o *middleware* Web, fornecendo funcionalidades como sessões. Por exemplo, uma rota no `web.php` pode mapear um pedido GET ao URL /home para o método `index` de um controlador `HomeController`, definindo assim o comportamento da aplicação quando essa página é acedida. Para definir as páginas Web feitas em *Blade* é necessário inserir as rotas neste ficheiro;
  - **api.php**: O ficheiro `api.php`, à semelhança com o ficheiro `web.php` também serve para definir os controladores e as ações a executar. No entanto, este ficheiro é específico para as rotas que definem a API.

Embora existam muitas mais pastas e ficheiros na estrutura do *Laravel*, optou-se por mencionar mais detalhadamente apenas os mais importantes no contexto deste documento.

Apesar de utilizarmos a lógica MVC para o *Backend*, a parte da *View* é gerida pelo *Vue.js*. O *Vue.js* está integrado diretamente dentro do *Laravel*, permitindo que o *Laravel* entregue os ficheiros *Vue* ao *browser*. Este método elimina a necessidade de ter um servidor separado a correr o *Node.js* ou outro software, simplificando a arquitetura do projeto e reduzindo a sobrecarga de manutenção.

Para a integração entre *Laravel* e *Vue.js*, utilizamos pacotes como *laravel-mix*[1] , que facilita a compilação e otimização dos *assets* Vue. Isso garante uma integração fluida e eficiente entre o *Backend* e o *Frontend*.

No que diz respeito à estrutura do *Vue.js*, seguimos uma convenção própria para a criação de novas páginas. Cada nova página é criada dentro da pasta `pages`, que se encontra dentro do `Resources/Js` do *Laravel*, utilizando o nome do módulo em inglês. Dentro de cada módulo, a página principal é nomeada `main.vue`. Os componentes de filtros, que estão incluídos na página principal, são nomeados `filters.vue`. Além disso, existe um componente `edit.vue` que geralmente corresponde a um *modal* utilizado para criar e editar itens da página em questão, sendo também exibido dentro de `main.vue`. Caso exista a necessidade de criar componentes específicos daquele modulo é criada uma pasta denominada *components* no mesmo nível em que se encontra o `main.vue`. Isto tudo é possível observar na Figura 13.

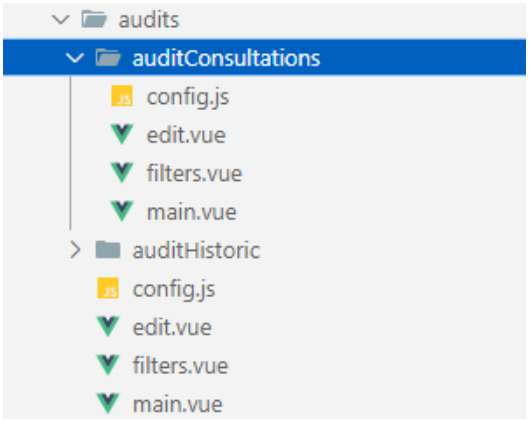


Figura 13: Estrutura de ficheiros

Os componentes genéricos, que são reutilizáveis em várias partes da aplicação, são armazenados na pasta `components/generic`. Esta organização permite uma reutilização eficiente e uma manutenção mais simples dos componentes comuns. Por exemplo, botões, *inputs*, *dialogs* de confirmação, *multiselects* e outros elementos comuns são colocados nesta pasta, evitando a duplicação de código e promovendo a consistência visual e funcional.

1 https://laravel-mix.com/

As rotas do *Vue.js* são definidas no ficheiro `index.js` dentro da pasta `router`. Este ficheiro centraliza todas as definições de rotas de *Frontend*, permitindo uma gestão coerente e organizada da navegação na aplicação. A utilização do *vue-router*[2] permite-nos criar uma experiência de navegação fluida e dinâmica, essencial para aplicações *Single Page Application* (SPA) [31].

Para a gestão de estado global do Portal, utiliza-se o *Pinia*[3], uma biblioteca que oferece uma forma eficiente e simples de gerir o estado da aplicação do lado do cliente. O *Pinia* facilita o acesso aos *storages* do *browser* (*local storage* e *session storage*) e a gestão dos *cookies*, o que permite uma boa partilha de dados entre componentes e manter informação associada ao *browser*.

4.1.2 *Página Principal*

Para entender melhor a estrutura e funcionalidade do portal, é importante destacar a consistência e coerência presente nas suas páginas. Esta abordagem facilita a utilização e adaptação por parte dos utilizadores, assegurando uma experiência de navegação intuitiva e eficiente.

As páginas do portal seguem uma estrutura semelhante, que pode ser observada na Figura 14.

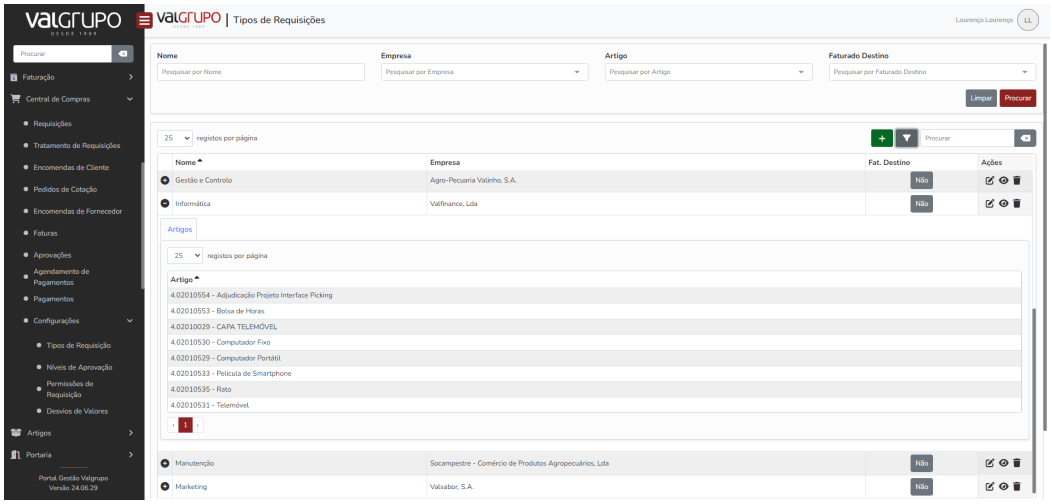


Figura 14: Exemplo de Página Principal do Portal de Gestão

À esquerda, encontra-se o menu lateral que exibe os diversos módulos disponíveis. Estes módulos variam conforme as permissões do utilizador autenticado. As opções

2 https://router.vuejs.org/
3 https://pinia.vuejs.org/

no menu principal podem ou não incluir sub menus, como é possível observar no caso da Central de Compras que possui uma opção para as "Requisições", "Tratamento de Requisições", entre outros.

Na parte superior da página, pode-se ver o título da secção atual, bem como o logótipo da empresa referente a secção atual, e um avatar com as iniciais do utilizador. Ao clicar no avatar, o utilizador tem acesso ao seu perfil e à opção de terminar a sessão.

Na zona do conteúdo, na parte superior, estão os filtros, que por omissão aparecem ocultos. Estes filtros permitem filtrar os dados exibidos na tabela abaixo.

Entre os filtros e a tabela, encontram-se as ações principais. Estas incluem a capacidade de alterar a quantidade de itens exibidos na tabela, abrir e fechar os filtros, pesquisar de forma geral na tabela, e abrir um modal para criar um novo registo na tabela.

A tabela é o elemento central da página, exibindo os dados filtrados. Possui uma coluna chamada "Ações" que permite realizar ações específicas para cada linha, como editar, visualizar detalhes, apagar, entre outros.

No lado esquerdo da tabela, em certos casos, é possível expandir a linha para mostrar informações adicionais relacionadas com essa entrada, o que é útil para fornecer contexto adicional sem sobrecarregar a interface principal.

Essa estrutura coerente não só facilita a navegação e utilização do portal, mas também garante que os utilizadores possam rapidamente adaptar-se e encontrar as funcionalidades necessárias. Às vezes, não é possível seguir esta estrutura à risca, mas procura-se sempre manter uma consistência próxima.

4.1.3 *Criar ou Editar*

Para proporcionar uma melhor compreensão sobre a estrutura das páginas no Portal, é importante destacar a forma como a criação e edição da informação é realizada.

Após a página principal, uma das ações mais cruciais no Portal é a criação e edição de dados. Conforme mencionado anteriormente, ao clicar nas opções de criar ou editar, um *modal* é exibido, permitindo inserir ou alterar as informações de um registo. Este *modal*, que pode variar em complexidade dependendo do tipo de registo que está a ser manipulado.

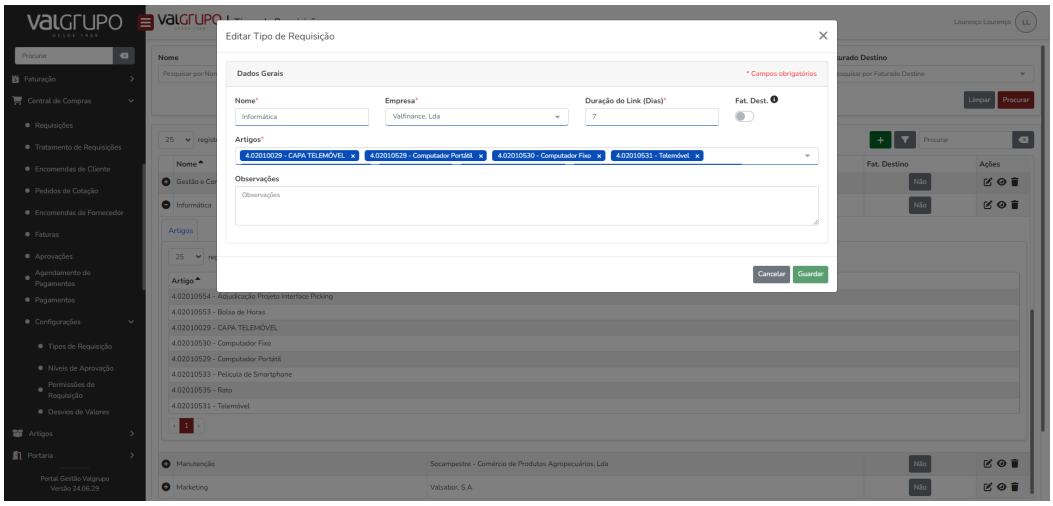


Figura 15: Exemplo do *Modal* de Criar ou Editar

Na Figura 15, é possível visualizar um exemplo deste *modal*. Observa-se que ele está dividido em várias secções para facilitar a inserção de informações. Os campos obrigatórios são destacados com um asterisco vermelho, como "Nome", "Empresa" e "Duração do Link (Dias)". Além disso, há a possibilidade de associar múltiplos artigos ao registo.

Outro aspeto importante é a inclusão de uma secção para observações, disponível em quase todos os módulos, permitindo que os utilizadores adicionem notas adicionais relevantes ao registo. A interface do *modal* também apresenta opções claras para cancelar ou guardar as alterações, garantindo uma experiência de utilizador eficiente e intuitiva.

4.1.4 *Visualização de Detalhes*

Outra funcionalidade importante a destacar é a capacidade de visualizar os detalhes de um registo. Quando um utilizador clica para ver mais detalhes na opção do "olho" na tabela, um *modal* é exibido, permitindo observar a informação do registo de forma mais detalhada. Este *modal* contém informações que podem não estar visíveis na tabela principal.

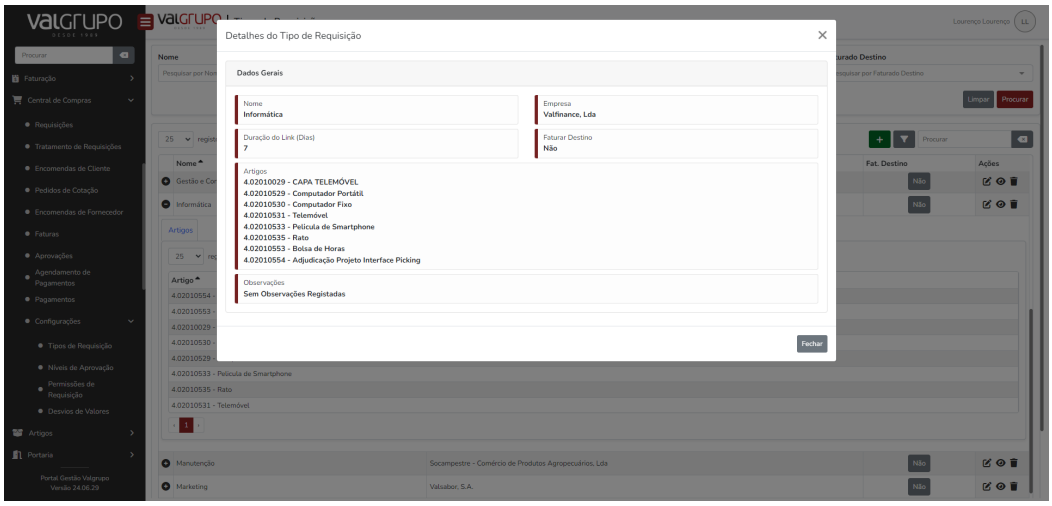


Figura 16: Exemplo do *Modal* de Visualização de Detalhes

Na Figura 16, é possível visualizar um exemplo deste *modal*. A estrutura do *modal* está organizada de maneira a facilitar a leitura e a compreensão das informações. No topo, encontram-se os "Dados Gerais" do registo, incluindo campos como "Nome", "Empresa", entre outros.

Além disso, o *modal* lista os artigos associados ao registo. Neste exemplo, podemos ver uma lista de artigos detalhados, com o nome e o código de cada um, proporcionando uma visão completa dos itens vinculados ao registo.

4.1.5 *Confirmação de edição de registo*

Quase todas as ações possuem validações de *Frontend* e de *Backend* e um *pop-up* de confirmação de modo a prevenir qualquer erro ao manipular um registo. Como é possível visualizar na Figura 17, ao tentar apagar um registo é solicitada a confirmação ao utilizador.

Figura 17: Exemplo de *pop-up* de Confirmação

4.2 AGENDAMENTO DE AUDITORIAS

Apesar de o Portal de Gestão já se encontrar em desenvolvimento, o mesmo não possuía qualquer tipo de funcionalidade relacionada com as auditorias, que inicialmente eram realizadas manualmente, utilizando suporte em papel, o que frequentemente se tornava pouco prático. Para integrar as auditorias no Portal, foi necessário desenvolver várias páginas específicas.

Para que as auditorias pudessem ser agendadas no Portal, conforme mencionado anteriormente na Secção 3.3, foi requisitada a implementação de visualização em estilo de calendário para mostrar todas as auditorias agendadas num determinado período temporal.

Na Figura 18, é possível visualizar o esquema preliminar da página, desenhado antes do desenvolvimento, para servir de orientação. Este esquema segue a mesma ideologia explicada na Secção 4.1 e por isso demonstra a necessidade de incluir uma secção de filtragem dos dados do calendário, permitindo aos utilizadores ajustar os dados exibidos, conforme necessário. Além disso, o esquema destaca a necessidade de uma funcionalidade para alterar a visualização do calendário entre os modos de dia, semana e mês. É igualmente importante fornecer uma funcionalidade de pesquisa para datas específicas e permitir a navegação entre as datas selecionadas, tanto para frente, quanto para trás. Adicionalmente, destaca-se a importância de incluir a funcionalidade de criação e agendamento de auditorias clicando no botão "Novo". Após a criação das auditorias, as mesmas serão exibidas no calendário, onde será possível ver mais detalhes e até mesmo editar uma auditoria.

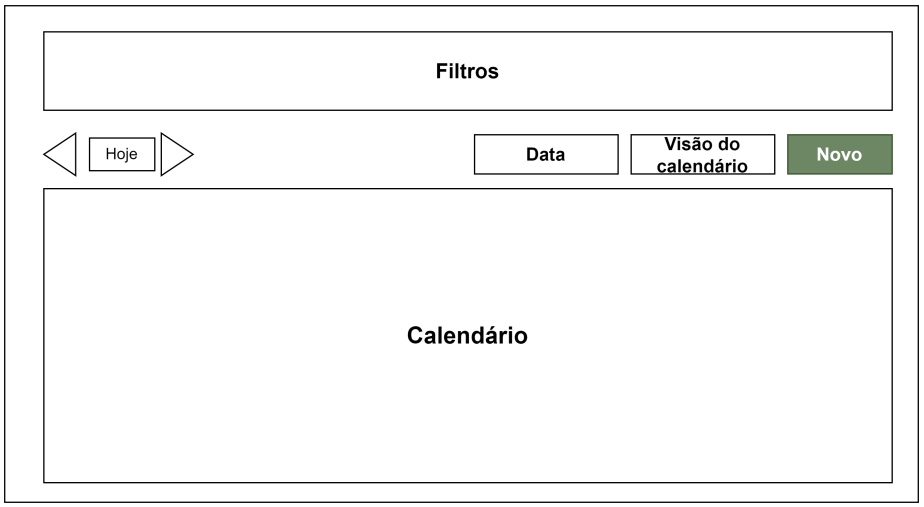


Figura 18: Esquema da página de Agendamento de Auditorias

Dado que ainda não existia nenhum componente de calendário no Portal, e não se justificava a criação de um componente próprio para esta funcionalidade, houve a necessidade de procurar um componente externo compatível com a versão 2 do Vue. Foram analisadas diversas opções, destacando-se as seguintes:

4.2.1 *Vuetify Calendar*

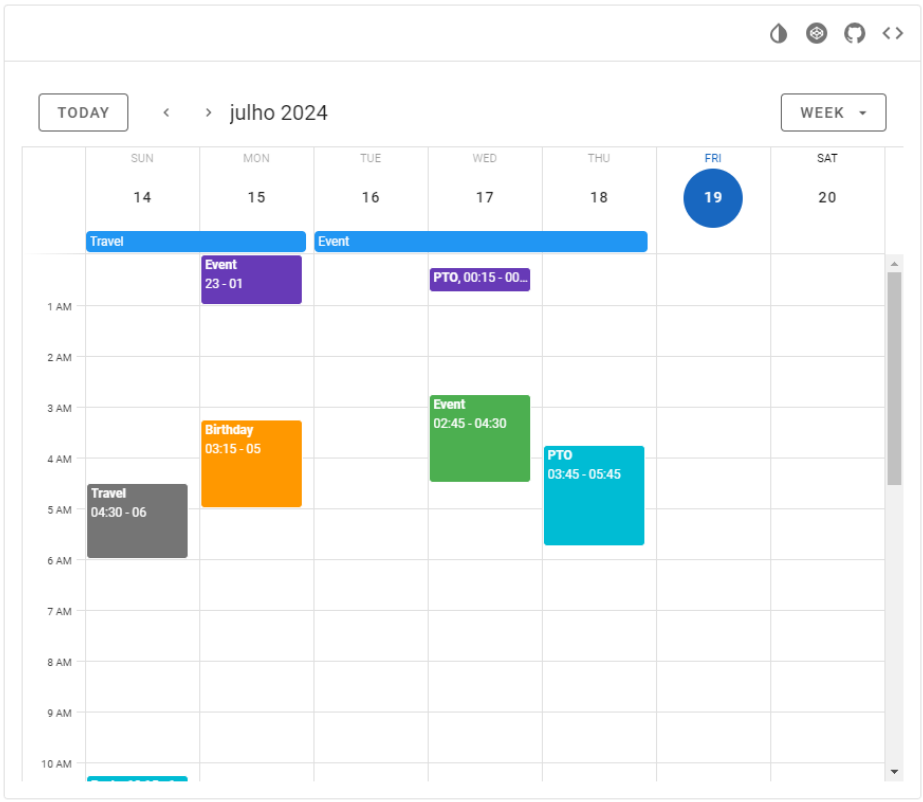


Figura 19: Calendário do Vuetify

O *Vuetify Calendar*[4] [32] é um componente de calendário oferecido pelo *framework* Vuetify[5], conhecido pela sua integração perfeita com *Vue.js* e pelo design elegante baseado em *Material Design*. Este componente oferece uma vasta gama de funcionalidades, incluindo a visualização diária, semanal e mensal, bem como a capacidade de arrastar e soltar eventos para alterar datas facilmente. No entanto, uma das suas desvantagens é a necessidade de utilizar todo o ecossistema Vuetify para tirar o máximo proveito do componente, o que pode adicionar complexidade ao projeto.

4 https://v2.vuetifyjs.com/en/components/calendars/
5 https://vuetifyjs.com/en/

4.2.2 *V-Calendar*

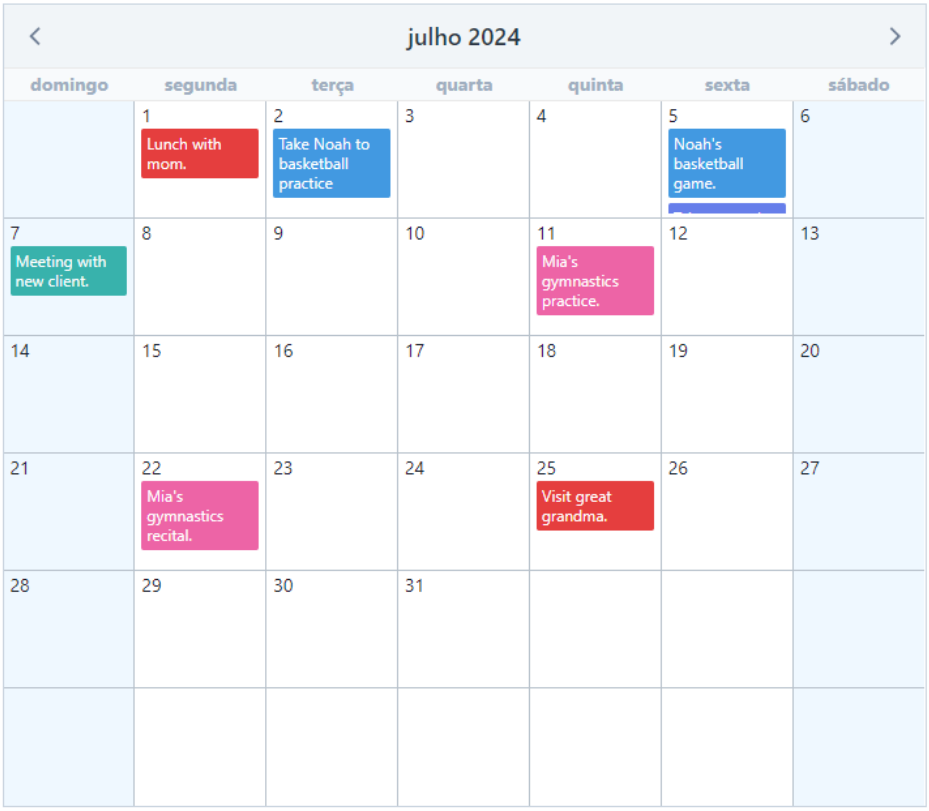


Figura 20: Calendário do V-Calendar

O *V-Calendar*[6] [33] é um componente robusto que oferece uma interface intuitiva e altamente personalizável. Este calendário é projetado para ser simples de usar, com opções para adicionar, editar e remover eventos diretamente na interface. Além disso, suporta múltiplas visualizações de calendário, o que permite uma flexibilidade significativa na apresentação dos dados das auditorias. A principal desvantagem é a sua limitada documentação e comunidade de suporte, o que pode dificultar a resolução de problemas e a implementação de funcionalidades avançadas.

6 https://v2–vcalendar.netlify.app/

4.2.3 *Tallent Calendar*

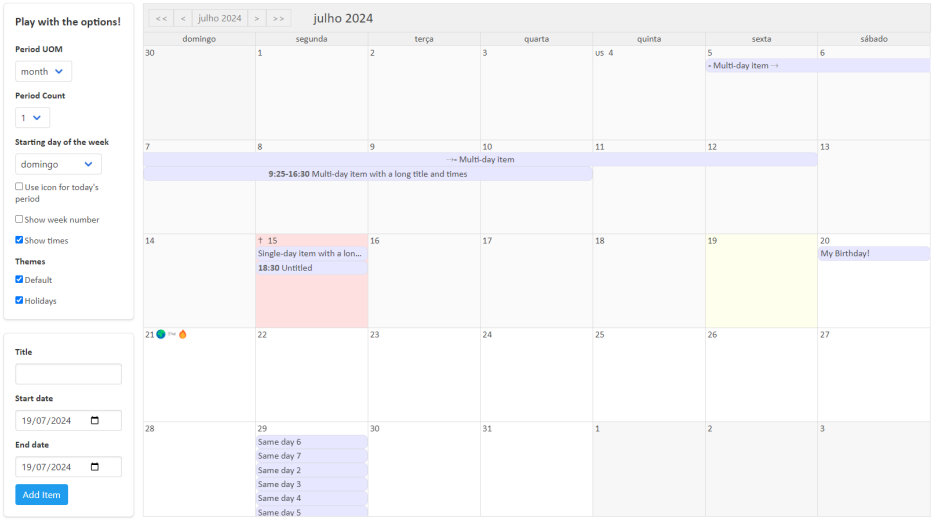


Figura 21: Calendário do Tallent

O *Tallent Calendar*[7] [34] destaca-se pela sua facilidade de integração e pelo conjunto abrangente de funcionalidades. Este componente permite a visualização e gestão de eventos com um design limpo e responsivo. Entre as suas características, encontram-se a capacidade de sincronizar com outras aplicações de calendário, notificações de eventos e suporte para diferentes fusos horários, tornando-o ideal para empresas que operam em múltiplas localizações. No entanto, a integração pode ser complexa, especialmente em projetos que já utilizam outras bibliotecas ou *frameworks*.

7 https://tallent.us/vue-simple-calendar/

4.2.4 *TOAST UI Calendar*

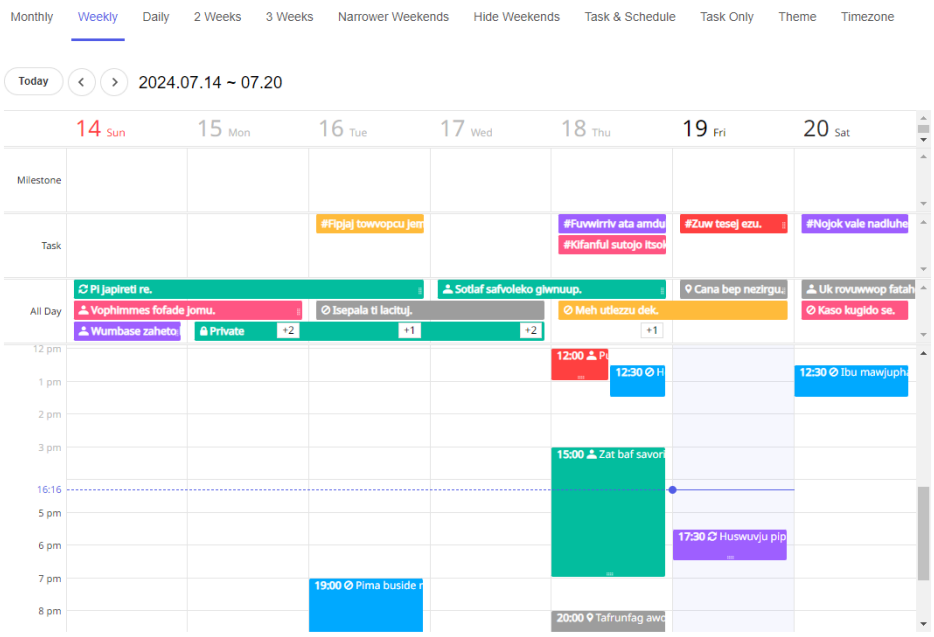


Figura 22: Calendário do TOAST UI

O *TOAST UI Calendar*[8] [35] é um componente altamente flexível e configurável, conhecido pela sua performance eficiente e pela ampla gama de funcionalidades. Este calendário permite a visualização em múltiplos formatos (diário, semanal, mensal) e oferece recursos avançados como filtros, *tags*, a capacidade de definir diferentes níveis de prioridade para os eventos e a capacidade de arrastar os eventos no calendário de forma prática, de modo a alterar as suas datas. Além disso, permite a atribuição de cores aos eventos, o que facilita a visualização e a organização das auditorias. A presença de *pop-ups* de pré-visualização ao clicar num evento também é uma funcionalidade útil (ver Figura 23), proporcionando um rápido acesso a informações detalhadas. Adicionalmente, o TOAST UI Calendar possui uma excelente documentação e uma comunidade ativa, o que facilita a sua implementação e a resolução de problemas.

8 https://ui.toast.com/tui-calendar

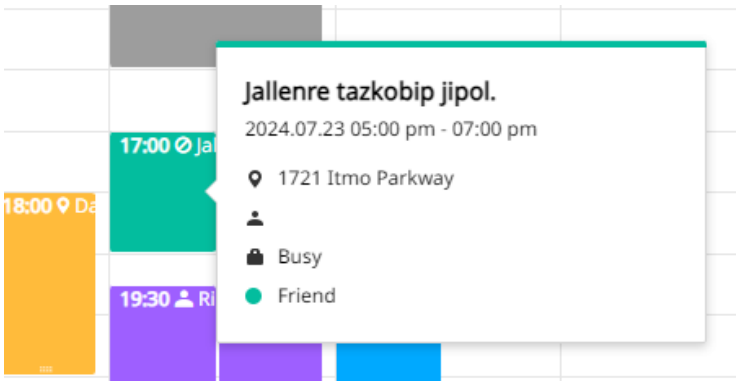


Figura 23: Pop-up do TOAST UI

Tendo em consideração a nossa análise optou-se por escolher o TOAST UI Calendar tendo sido esta motivada por várias razões. Em primeiro lugar, a sua flexibilidade e capacidade de configuração atenderam perfeitamente às necessidades do projeto, permitindo uma adaptação fácil às especificidades das auditorias. A ampla gama de funcionalidades, incluindo a possibilidade de visualizar o calendário em diferentes formatos e a gestão avançada de eventos, foi um fator decisivo. Além disso, a excelente documentação e a comunidade ativa proporcionaram um suporte contínuo durante o desenvolvimento. Estas vantagens, combinadas com a performance robusta do TOAST UI Calendar, tornaram-no a escolha ideal.

4.2.5 *Página Principal das Auditorias*

Uma vez escolhido o componente do calendário a ser usado, começou por se instalar o componente no Vue. Utilizando o comando:

```
npm install @toast-ui/calendar
```

Posto isto, foi desenvolvida a página principal que é possível visualizar na Figura 24. Os filtros pedidos pelos utilizadores corresponderam a: filtrar por empresa, por exploração, por pessoa, que é a pessoa responsável por realizar a auditoria, e filtrar por estado, de modo a descartar auditorias concluídas. Nestes filtros é possível selecionar mais que um item, como se vê na Figura 24, no filtro das empresas.

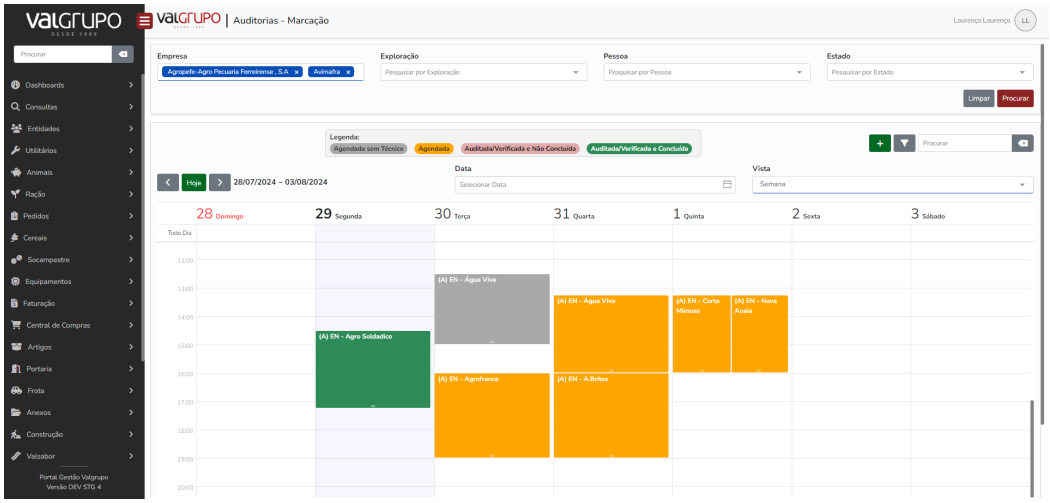


Figura 24: Página Principal das Auditorias

Como requisitado também é possível alterar a vista do calendário para analisar um mês completo (ver Figura 25), e para um dia específico, conforme apresenta a Figura 26.

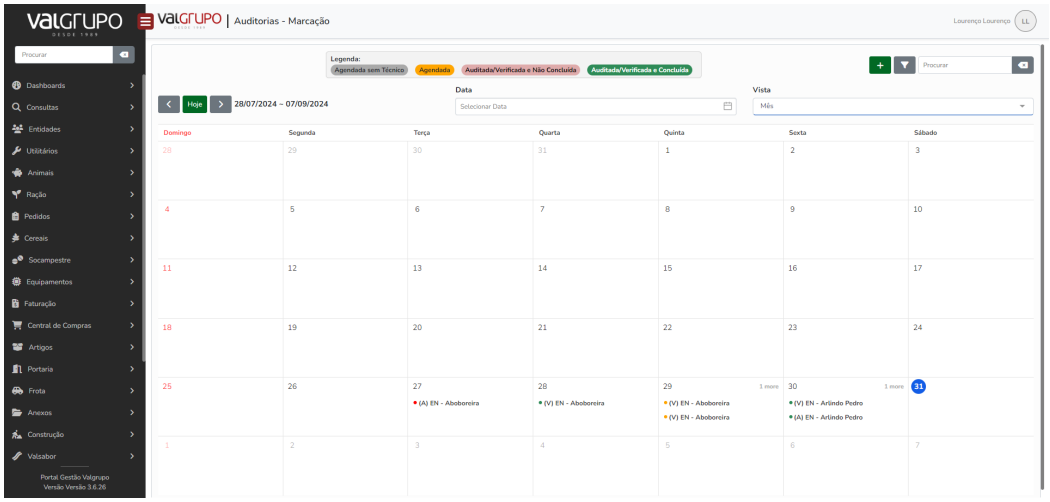


Figura 25: Calendário no formato de Mês

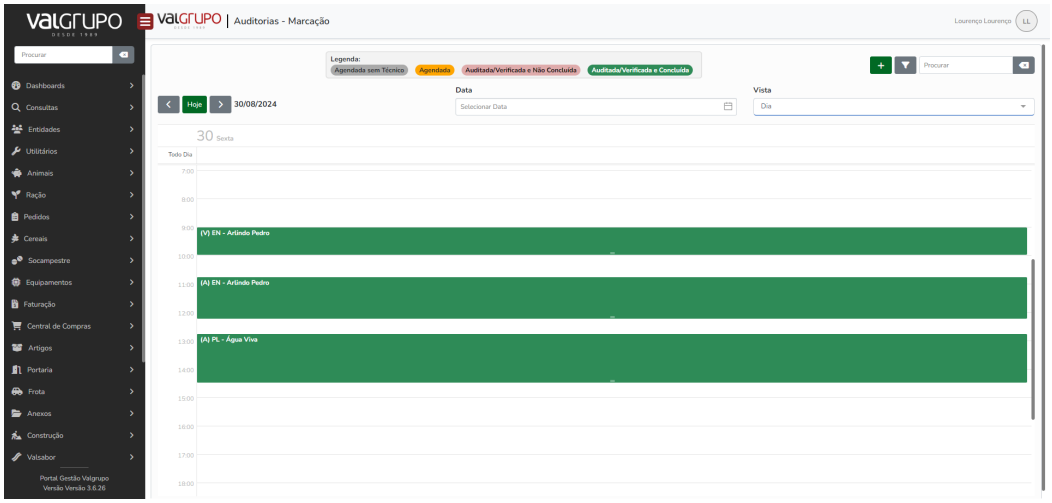


Figura 26: Calendário no formato de Dia

Uma vez que o componente do calendário, bem como o sistema de alteração de vista e de navegação do mesmo poderá vir a ser reutilizado para outros módulos, o mesmo foi implementado dentro do diretório `components/generic`, seguindo a estrutura de pastas anteriormente apresentada.

4.2.6 *Criar e Editar Auditorias*

Uma vez que é necessário criar e manter registos detalhados de auditorias, surgiu a necessidade de desenvolver um *modal* específico para criar e editar essas informações, Figura 27. Este *modal* é uma ferramenta crucial que permite aos utilizadores inserirem novos dados ou modificar registos existentes de auditorias.

De modo a prevenir alterações indesejadas, uma vez que podem existir mais pessoas a criar registos nesta página ou simplesmente com permissões para visualizar, a edição da auditoria só é possível ser realizada por quem a criou inicialmente, sendo que mesmo para a criação é necessário possuir uma permissão.

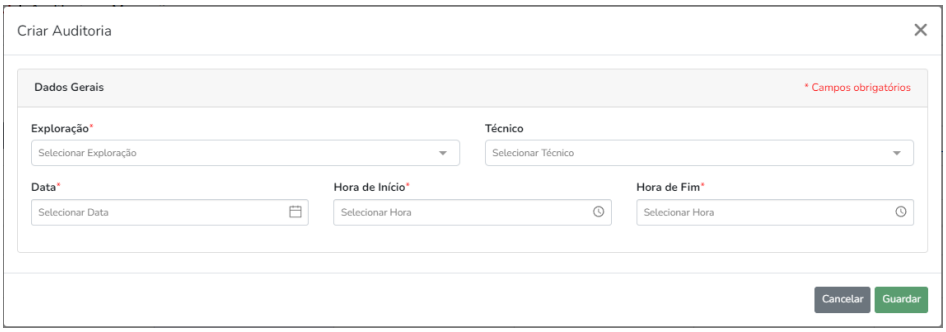


Figura 27: *Modal* de Criar e Editar Auditorias

Neste *modal* é obrigatório selecionar a exploração, sendo esta o local onde vai ser realizada a auditoria, bem como a data e a hora de inicio e de fim expectável. O Técnico é o responsável por realizar a auditoria no local físico e pode não ser selecionado logo no ato de criação. Caso isto aconteça, a auditoria fica no estado "Agendada Sem Técnico" e só pode ser auditada assim que um técnico for atribuído. Outro ponto a destacar é que este técnico é automaticamente preenchido ao selecionar a exploração, isto é feito com base no técnico responsável pela exploração selecionada.

Algo a denotar é que o utilizador pode também pressionar e arrastar no componente do calendário, de modo a auto preencher as datas e horas, como é possível observar na Figura 28.

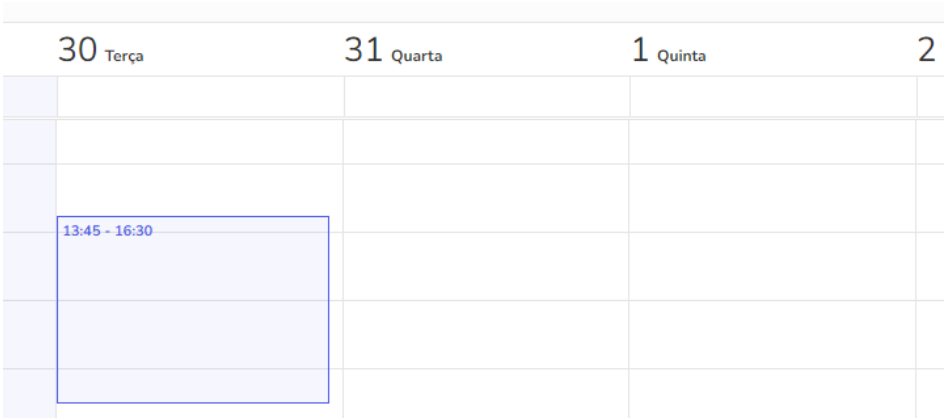


Figura 28: Criação de registo pelo Caléndario

4.2.7 *Detalhes da Auditoria*

Uma vez que uma auditoria não possui muita informação, não se justifica a criação de um *modal* específico para visualizar os detalhes da auditoria. Em vez disso, essa visualização pode ser realizada através da pré-visualização do calendário, conforme ilustrado na Figura 29.

Figura 29: Detalhes da Auditoria

Nessa pré-visualização é possível observar o nome da exploração que corresponde ao título, o local da auditoria, o responsável encarregue de realizar a auditoria, a empresa referente à exploração, e o estado da auditoria.

4.3 HISTÓRICO DE AUDITORIAS

A consulta do histórico das auditorias realizadas é uma funcionalidade importante para garantir a transparência e a rastreabilidade dos processos. Manter um registo detalhado das auditorias anteriores permite, não só a revisão das ações passadas, como também a identificação de padrões e a avaliação contínua da conformidade com as normas estabelecidas. Além disso, disponibilizar uma página dedicada a esta consulta facilita o acesso rápido e organizado à informação histórica, contribuindo para a tomada de decisões informadas e para a melhoria contínua dos processos auditados.

4.3.1 *Página Principal do Histórico de Auditorias*

Uma vez que esta página seguiu a estrutura padrão referida na Secção 4.1.2 não houve a necessidade de realizar um esquema novo, pelo que se optou por definir as colunas da tabela bem como a informação a mostrar dentro de cada linha e os filtros. Para as colunas definiu-se que era importante mostrar a lista de explorações com a sua empresa respetiva, o responsável da exploração, a data da última auditoria e o estado da última auditoria. Ao expandir a linha é também possível observar o histórico todo das auditorias realizadas na respetiva exploração. Em relação aos

filtros, foram aplicados os mesmo da página do agendamento. Tudo isto é possível observar na Figura 30.

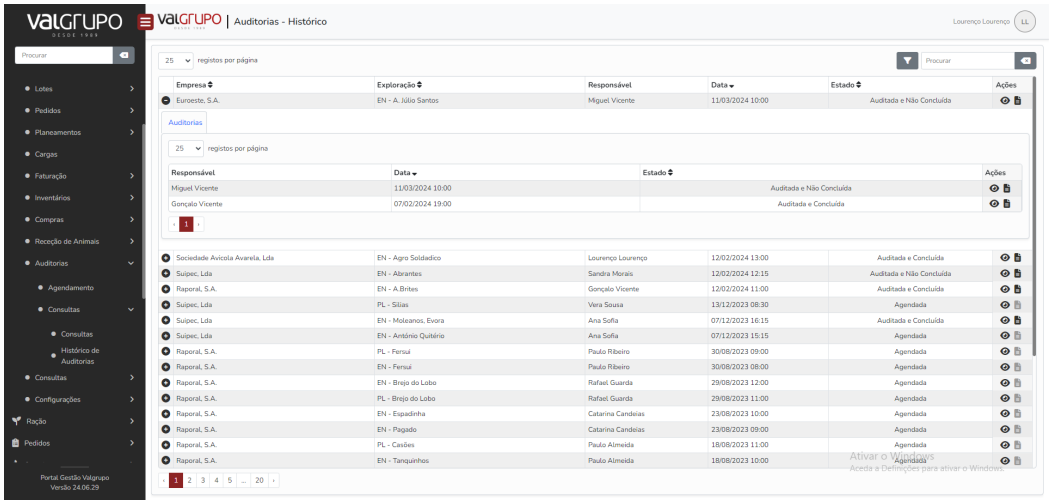


Figura 30: Histórico de Auditorias

4.3.2 *Detalhes do Histórico de Auditorias*

Ao premir o *icon* do "olho" é possível observar informação mais detalhada da auditoria, conforme apresentado na Figura 31.

Figura 31: Detalhes do histórico das Auditorias

4.3.3 *Relatório da Auditoria*

Apesar do objetivo deste projeto ser reduzir o consumo de papel, por vezes existe a necessidade de imprimir um relatório de uma determinada auditoria, ou até por

vezes pode ser apenas útil para enviar via email, por exemplo. Posto isso, verificou-se a necessidade de desenvolver um mecanismo para gerar um ficheiro PDF com as estatísticas mais relevantes.

Ao lado do *icon* do "olho" na Figura 30 é possível também observar um *icon* simbolizando um documento. Ao clicar no mesmo é possível gerar o PDF que é automaticamente transferido para o computador do utilizador.

Nesse documento é possível observar estatísticas relacionadas com a alimentação dos animais, o alojamento dos animais, a saúde dos animais, o comportamento do animal quando interagido com o ser humano e as não conformidades encontradas durante a auditoria, Figura 32.

valPOR
PRODUÇÃO ANIMAL

Auditoria Interna

Tipo:	Auditoria	**Responsável:**	
Empresa:	Suipec, Lda	**Data da Auditoria:**	
Exploração e Marca:	Moleanos, Evora-PTRG79M	**Estado:**	Auditada e Concluída

Resumo de Auditoria Interna Simplificada - Suínos de Engorda

Dados Gerais

Número de Animais Observados	Peso Médio (Kg)	Número de Parques Observados
140	90	10

Boa Alimentação

Parques com menos de 2 pontos de bebida	Limpeza de Bebedouros	Parques com mais de 10 animais por bebedouro	Média de Animais/ Bebedouros
10	100.00%	10	13.90

Bom Alojamento

Densidade	Fezes no corpo	Amontoamento
0.64	16.43% | 1.43%	0.00%

Boa Saúde

Diarreia	Lesões na Pele	Caudofagia	Bursites	Mortalidade
0.00%	2.14% | 0.00%	0.00%	7.14% | 6.43%	1.00%

Comportamento Apropriado

Medo de Humanos
0.00%

Figura 32: Relatório gerado da Auditoria em formato PDF

4.3.4 *Registo da Ausência de Sede Prolongada*

Após a realização de uma auditoria, pode surgir a necessidade de associar um parâmetro adicional denominado "Ausência de Sede Prolongada". Este parâmetro é calculado através de um software externo, denominado *INRAE Welfare Quality* [36], que utiliza uma fórmula específica para determinar o valor com base nos diversos dados recolhidos durante a auditoria. Para facilitar este processo, foi criada uma página que permite aceder a uma lista das explorações já auditadas, juntamente com os dados da última auditoria realizada, conforme ilustrado na Figura 33.

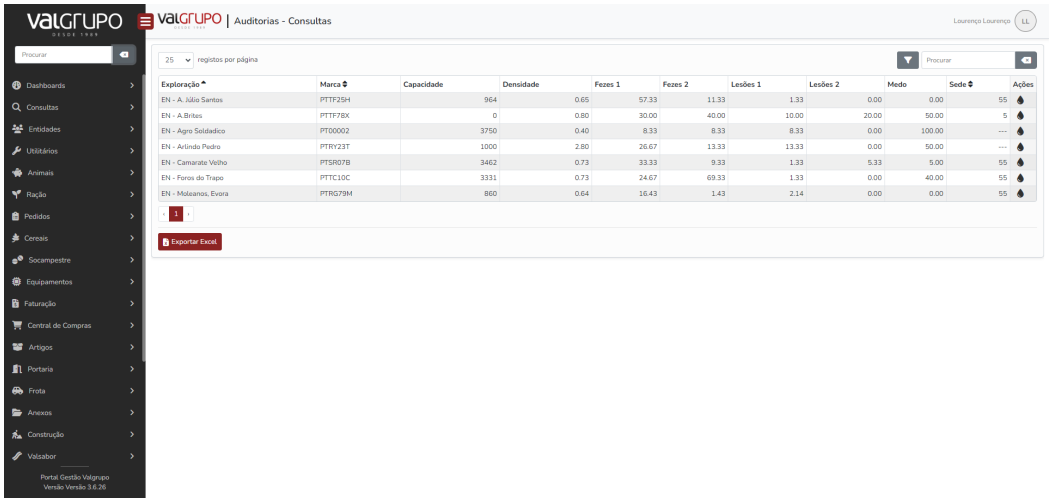


Figura 33: Página Principal do Registo da Ausência de Sede Prolongada

Ao clicar no ícone da "gota de água", é aberto um *pop-up* que permite inserir esse valor. Uma vez inserido, o valor é imediatamente refletido na tabela.

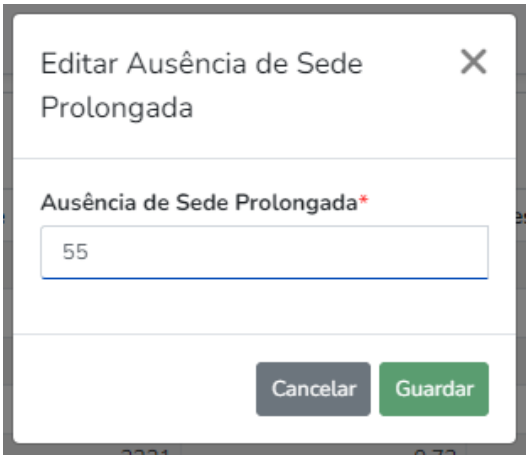
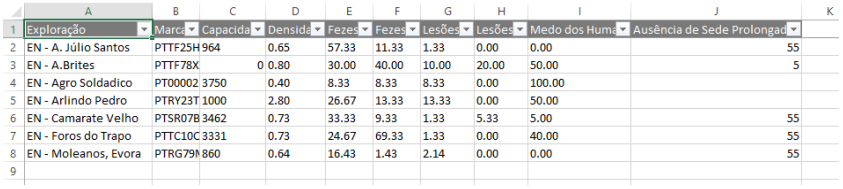


Figura 34: *Pop-up* do Registo da Ausência de Sede Prolongada

Nesta página, além de visualizar os dados, é possível exportar a tabela em formato Excel, facilitando a manipulação e análise dos dados fora da aplicação. Este ficheiro Excel pode ser posteriormente enviado por email para partilha com outros colaboradores, ou utilizado em outro software para análises mais avançadas, integração com sistemas externos ou criação de relatórios detalhados.



	A	B	C	D	E	F	G	H	I	J	K
1	Exploração	Marca	Capacida	Densida	Fezes	Fezes	Lesões	Lesões	Medo dos Huma	Ausência de Sede Prolongad	
2	EN - A. Júlio Santos	PTTF25H	964	0.65	57.33	11.33	1.33	0.00	0.00	55	
3	EN - A.Brites	PTTF78X	0	0.80	30.00	40.00	10.00	20.00	50.00	5	
4	EN - Agro Soldadico	PT00002	3750	0.40	8.33	8.33	8.33	0.00	100.00		
5	EN - Arlindo Pedro	PTRY23T	1000	2.80	26.67	13.33	13.33	0.00	50.00		
6	EN - Camarate Velho	PTSR07B	3462	0.73	33.33	9.33	1.33	5.33	5.00	55	
7	EN - Foros do Trapo	PTTC10C	3331	0.73	24.67	69.33	1.33	0.00	40.00	55	
8	EN - Moleanos, Evora	PTRG79N	860	0.64	16.43	1.43	2.14	0.00	0.00	55	
9											

Figura 35: Excel do Registo da Ausência de Sede Prolongada

4.3.5 *Criação dos questionários*

Uma auditoria é, em grande parte, baseada em questionários, e esses questionários podem necessitar de alterações frequentes devido a mudanças na Lei e regulamentação em vigor. Por isso, tornou-se essencial implementar um sistema que permitisse a atualização rápida e fácil desses questionários, garantindo que eles possam ser prontamente adaptados e respondidos posteriormente na Aplicação Móvel.

O Portal já dispõe de uma funcionalidade de criação de formulários, e como a nova Aplicação Móvel utilizará essa funcionalidade, uma vez que o preenchimento de auditorias será feito principalmente através de questionários, optou-se por aproveitá-la, ajustando-a para funcionar de forma eficiente na Aplicação Móvel. Na Figura 36, é possível visualizar a página que contém os formulários de diversos módulos, utilizados em diferentes locais no Portal.

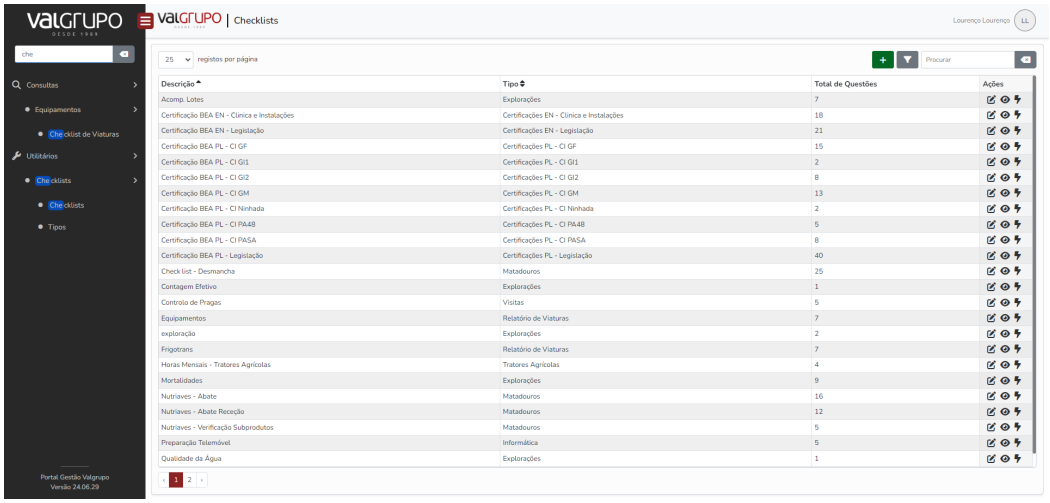


Figura 36: Lista dos questionários

Ao criar ou editar um formulário, Figura 37, é possível selecionar o tipo de questionário. Este tipo é utilizado para questões de filtragem e para obter os questionários referentes ao módulo pretendido, ou seja no caso das auditorias houve a necessidade de criar um tipo para cada questionário uma vez que posteriormente a Aplicação Móvel obterá os questionários com base nesse tipo.

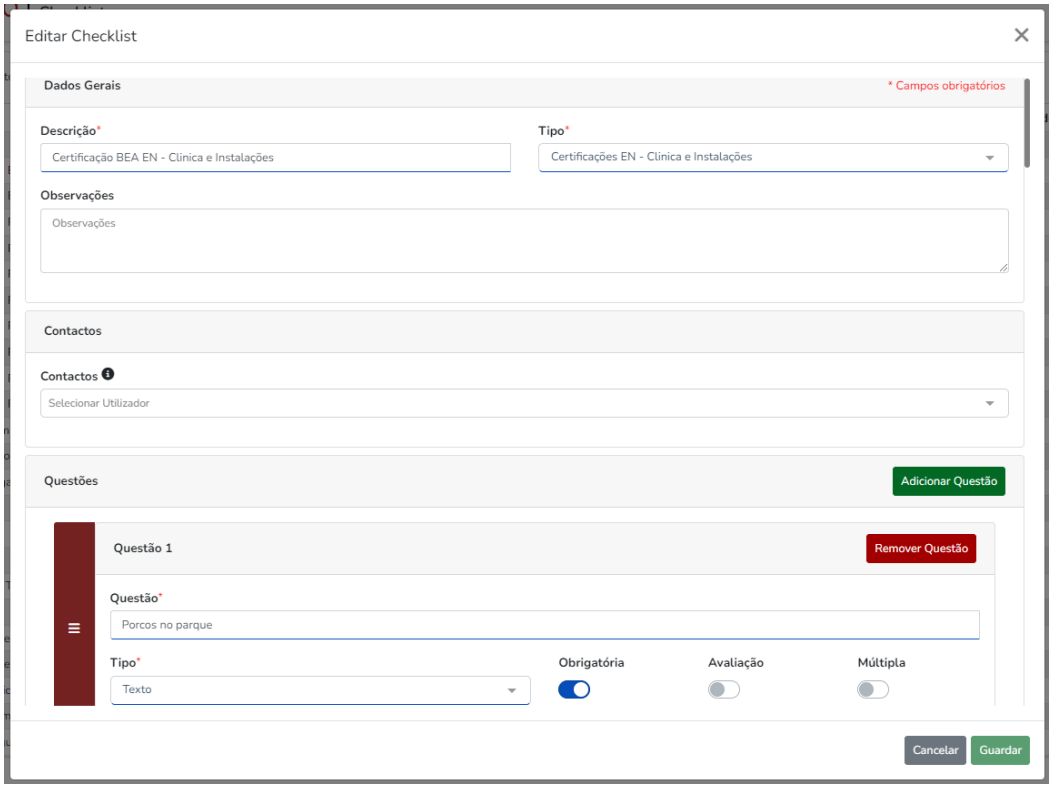


Figura 37: Editar dos formulário

Ao adicionar a questão é possível selecionar o seu tipo de questão, sendo as opções: "Texto" ou "Opção". Ao escolher "Opção" é necessário adicionar quais as opções que o utilizador pode selecionar, Figura 38. É possível também selecionar a questão como sendo de resposta obrigatória e/ou de escolha múltipla. É também possível selecionar a resposta como sendo de avaliação, que para o contexto deste projeto não é relevante.

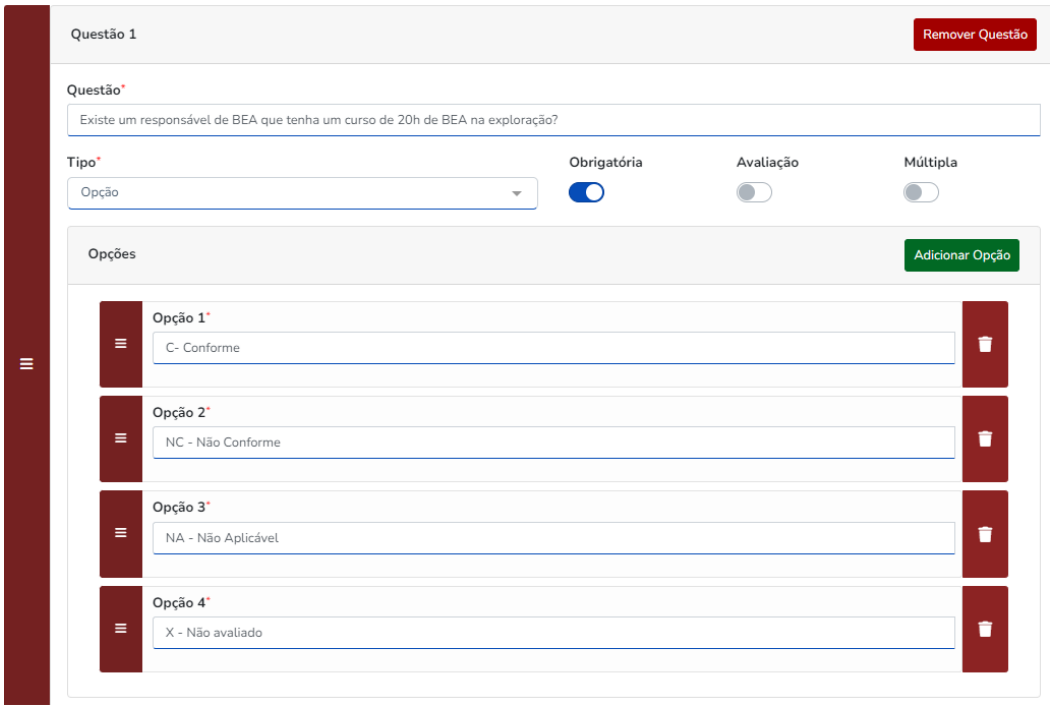


Figura 38: Opções na criação dos formulário

4.4 RESULTADOS

O desenvolvimento do Portal de Gestão trouxe diversos resultados práticos e relevantes para o projeto, cumprindo os objetivos estabelecidos e atendendo às necessidades identificadas. Um dos principais resultados foi a criação de uma API que serve de suporte à integração com a nova Aplicação Móvel, que será abordada no Capítulo seguinte, facilitando a comunicação entre eles, garantindo a sincronização de dados em tempo real.

Além disso, todas as funcionalidades requisitadas foram implementadas com sucesso. As principais funcionalidades incluem o agendamento, a gestão de auditorias e a geração de relatórios em formato PDF.

5

DESENVOLVIMENTO DA APLICAÇÃO MÓVEL - VALGRUPO

Tendo sido implementada a funcionalidade de agendamento de auditorias no Portal, começou-se o desenvolvimento da nova Aplicação Móvel.

Para garantir que a nova aplicação atendesse às expectativas dos utilizadores e melhorasse a experiência geral, decidiu-se iniciar o processo de desenvolvimento com uma análise detalhada da Aplicação Móvel anteriormente utilizada pelo ValGrupo. Esta análise foi essencial para identificar elementos que poderiam ser aproveitados, assim como avaliar quais as melhorias que poderiam ser implementadas. Paralelamente, identificou-se a necessidade de aprofundar os conhecimentos sobre a *framework Flutter*, o que levou à criação de uma *spike* (mini aplicação de testes). Esta *spike* serviu apenas como meio de explorar e confirmar se o *Flutter* poderia atender aos requisitos necessários, incluindo performance, responsividade, e facilidade de integração com os sistemas já existentes no ValGrupo.

A experiência adquirida durante o desenvolvimento da *spike* foi essencial, fornecendo uma visão prática das capacidades e limitações do *Flutter*. Com a confirmação de que a tecnologia era adequada para os objetivos do projeto, avançou-se para a fase de desenvolvimento da Aplicação Móvel oficial.

Este capítulo abordará detalhadamente a análise da Aplicação Móvel já existente e o processo de desenvolvimento da nova Aplicação Móvel, incluindo a arquitetura escolhida, as funcionalidades implementadas, e os desafios superados durante o desenvolvimento.

5.1 ANÁLISE DA APLICAÇÃO MÓVEL EXISTENTE

Durante a análise da Aplicação Móvel existente, foram avaliados os seus pontos fortes e fracos, bem como as necessidades específicas dos utilizadores. A nova aplicação foi desenvolvida levando em consideração esses conhecimentos, preservando aspetos familiares da interface anterior e incorporando melhorias para otimizar a usabilidade e a eficiência. Essa abordagem teve como objetivo principal evitar

que os utilizadores precisassem de se adaptar completamente a uma nova interface, mantendo a consistência e o conforto no uso da aplicação.

Posto isto, na Figura 39 é possível observar a página inicial após efetuado o *login*. Aqui é possível ver uma lista de "cartões" referentes aos transportes de animais a serem efetuados pelo utilizador com a sessão iniciada. De ressalvar que o administrador consegue visualizar todos esses transportes de animais, mesmo não estando associados a si.

Figura 39: Página dos Movimentos de Animais

Ao carregar num dos "cartões" é possível interagir com o planeamento em questão. Na Figura 40 é possível visualizar o processo de adicionar informação referente a esse movimento, dados como peso do veículo vazio, peso total, despesas, observações entre outros.

Figura 40: Página de Ações de Movimentos de Animais

Outro ponto importante da Aplicação Móvel existente a destacar é que esta aplicação não permite o seu funcionamento sem acesso à Internet, ou seja toda a sua informação, caso não seja selecionada a opção para guardar a informação, será perdida.

Esta análise foi realizada com o intuito de se perceber algumas das razões que levaram às decisões que foram tomadas para a implementação da nova Aplicação Móvel a desenvolver.

5.2 ARQUITETURA

Concluída a fase de análise da Aplicação Móvel existente começou-se a pensar na arquitetura e estrutura da nova aplicação pois optar-se pela definição de uma boa arquitetura é um passo muito importante no desenvolvimento de um software. Tendo isto em consideração é necessário definir uma estrutura sólida e coesa para permitir o desenvolvimento de um projeto escalável e organizado.

De seguida serão abordadas as escolhas efetuadas quanto à estrutura lógica e de ficheiros da aplicação bem como referir quais os modelos criados e as suas ligações.

5.2.1 *Estrutura Lógica*

A *framework Flutter* oferece ao programador a flexibilidade de estruturar a aplicação tanto em termos lógicos como na organização dos ficheiros. No entanto, essa liberdade pode, por vezes, dificultar a gestão do projeto, uma vez que não existem regras concretas.

Posto isto, optou-se por seguir uma arquitetura MVC (Model-View-Controller), Figura 41, uma abordagem que se alinha com a estrutura já seguida no desenvolvimento do Portal de Gestão.

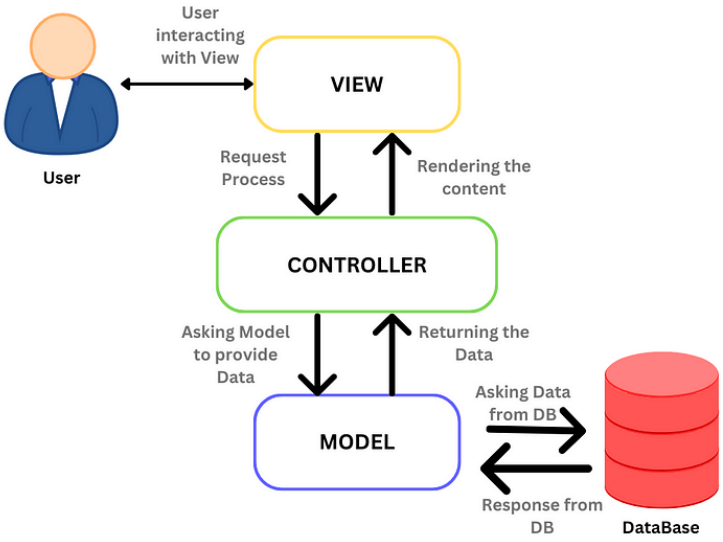


Figura 41: Arquitetura MVC (Model-View-Controller) [37]

A arquitetura MVC é um padrão de arquitetura de software que separa uma aplicação em três componentes principais: *Model*, *View* e *Controller*. Este padrão é amplamente utilizado no desenvolvimento de software devido à sua capacidade de organizar e modularizar o código, tornando-o mais fácil de manter e escalar.

- O *Model* (Modelo) representa os dados da aplicação. É responsável por gerir a forma como os dados são armazenados, recuperados e manipulados, sendo ele quem lida com as interações com a base de dados, como consultas e atualizações.

- A *View* (Visão) é responsável pela apresentação dos dados. É o componente que exibe a interface do utilizador e mostra os dados fornecidos pelo *Model*. No *Flutter*, uma *View* é frequentemente representada por *widgets*.
- O *Controller* (Controlador) atua como um ponto intermédio entre o *Model* e a *View*. O *Controller* recebe os dados do utilizador através da *View*, processa esses dados (por vezes alterando o *Model*), e retorna a informação adequada para a *View*. Por exemplo, o *Controller* pode lidar com eventos como o clique em um botão para adicionar um novo registo a tabela do *Model*, bem como validar se os dados estão corretos.

A adoção da arquitetura MVC traduz-se em diversos benefícios. A separação de preocupações, onde cada componente tem uma responsabilidade clara e distinta, facilita a manutenção e a escalabilidade do código. Além disso, a reutilização de código é facilitada, pois a lógica do negócio pode ser reutilizada em diferentes partes da aplicação ou em diferentes projetos, sem depender da interface do utilizador.

No desenvolvimento da Aplicação Móvel, a arquitetura MVC foi escolhida pelas suas vantagens na organização e modularização do código. Essa estrutura permitiu a criação de um sistema robusto, onde cada componente é desenvolvido e mantido de forma independente, facilitando futuras atualizações e melhorias.

5.2.2 *Estrutura de Ficheiros*

Em termos de estrutura de ficheiros, uma vez que o programador tem a possibilidade de estruturar à vontade, optou-se por, dentro do diretório do *Flutter* denominado `lib`, local onde se situa a lógica da aplicação implementada pelo programador, criar os seguintes diretórios:

- ***Models***: Local onde se encontram os modelos da aplicação. Estes modelos representam a estrutura dos dados que são manipulados pela aplicação;
- ***Controllers***: Diretório onde se encontram todos os controladores da aplicação. Os controladores são responsáveis pela lógica de negócio e pela intermediação entre os modelos e as *Views*;
- ***Pages***: Local onde se situa o código referente ao que o utilizador vai visualizar (no modelo MVC, estas correspondem às *Views*). Cada página representa um ecrã ou uma secção da aplicação com que o utilizador pode interagir. As páginas contêm o *layout* e os componentes visuais que compõem a interface do utilizador;

- ***Services***: Local onde se situa o código responsável por comunicar com o portal. Estes ficheiros possuem apenas pedidos HTTP e não efetuam nenhuma manipulação de dados. Os serviços são utilizados para isolar a lógica de comunicação com APIs externas, facilitando a manutenção e a reutilização do código. Por exemplo, um serviço pode conter funções para obter dados de um servidor ou enviar informações para ele;
- ***Styles***: Local onde se situam ficheiros de configuração dos estilos da aplicação, como cores, tamanhos de letras, tipos de letras, bordas arredondadas dos componentes, entre outros. Centralizar os estilos permite uma consistência visual na aplicação e facilita ajustes rápidos de design, uma vez que todas as alterações de estilo podem ser feitas num único sítio;
- ***Utils***: Diretório para ficheiros mais genéricos, por exemplo, um ficheiro responsável por comunicar com o `sharedPreferences`, onde persistem as informação do utilizador e as suas permissões. Outro exemplo de um ficheiro é o `validations`, que é o local onde se encontram funções de validações usadas em diversos locais da Aplicação Móvel, como validações de valor mínimo e máximo, entre outras. Os utilitários são funções ou classes que podem ser usadas em várias partes da aplicação, promovendo a reutilização do código;
- ***Widgets***: Diretório para componentes visuais mais genéricos, por exemplo, as barras de navegação (`navbars`) da aplicação;
- ***Globals***: Diretório para colocar entidades mais globais, como os componentes de inserção de texto, classes globais, diálogos genéricos e variáveis globais. Este diretório contém recursos que podem ser necessários em qualquer parte da aplicação, proporcionando um fácil acesso e evitando a duplicação de código;
- ***Isar***: Diretório com ficheiros para a comunicação com a base de dados local. A base de dados local é utilizada para armazenar dados que precisam estar disponíveis mesmo quando o dispositivo não está ligado à Internet. Este diretório contém a lógica para guardar, atualizar, apagar e consultar dados localmente, sem que exista manipulação.

No diretório `lib`, também se encontra o ficheiro `main.dart`, que é o ponto de entrada da aplicação *Flutter*. Este ficheiro contém a função `main()`, que é a primeira função executada quando a aplicação é iniciada. No ficheiro `main.dart`, é definido o *widget* raiz da aplicação, que configura a árvore de *widgets*, e inicia o ciclo de vida da aplicação. É neste ficheiro que geralmente se configura o tema da aplicação, as rotas iniciais, e outras configurações globais importantes.

Apesar de existirem mais pastas e ficheiros dentro do projeto, foram destacados aqui os mais importantes para o contexto deste documento, focando-se nos que possuem maior relevância para a estruturação e organização da aplicação desenvolvida durante o estágio.

5.2.3 *Modelos*

Compreendida a estrutura lógica e de ficheiros do projeto, deu-se início à criação dos seguintes modelos, essenciais para o funcionamento da Aplicação Móvel:

- ***Audit***: Este é o modelo principal de uma auditoria, onde são armazenadas todas as informações essenciais relacionadas com a auditoria em si. Inclui detalhes como a localização, a empresa detentora da exploração auditada, o estado atual da auditoria, entre outras informações relevantes que constituem o cabeçalho da auditoria. Este modelo é fundamental para a organização e rastreamento das auditorias realizadas;
- ***AuditClinicInstallation***: Serve como o cabeçalho das respostas ao questionário referente às instalações auditadas. Cada campo neste modelo contém informações específicas para cada parque ou instalação, proporcionando uma visão geral e organizada das respostas associadas a cada local auditado;
- ***AuditClinicInstallationAnswers***: Este modelo é utilizado para armazenar as respostas ao questionário das instalações. É aqui que cada resposta fornecida durante a auditoria é guardada;
- ***AuditVerification***: Criado especificamente para situações em que há uma não conformidade ao responder ao questionário da legislação. Este modelo regista e organiza as informações relacionadas com as verificações e correções necessárias. É essencial para assegurar que todas as não conformidades sejam devidamente identificadas e tratadas durante a auditoria;
- ***Checklist***: Modelo que contém as perguntas dos diversos questionário, que necessitam de ser respondidas durante a auditoria. Este modelo é essencial para garantir que as informações necessárias sejam acessíveis, mesmo quando não existe ligação à Internet. Armazena os dados provenientes do Portal, permitindo que os auditores possam continuar a responder ao questionário em modo *offline*;
- ***LegislationAnswers***: Este modelo armazena as respostas fornecidas ao questionário da legislação. Cada resposta associada às questões legais é guardada

aqui, facilitando a consulta e análise dos dados relativos ao cumprimento das legislações auditadas.

5.2.4 *Controladores*

De modo a permitir a interação entre a vista do utilizador e os modelos, seguindo arquitetura MVC, foram criados os seguintes controladores:

- **AuditController**: Este controlador é fundamental para garantir que toda a informação necessária esteja disponível ao abrir a página principal das auditorias. A sua principal função é obter os dados referentes à listagem de auditorias, verificações e perguntas dos diversos formulários. Para tal, o controlador efetuada pedidos à API do Portal de Gestão, recuperando os dados e armazenando-os localmente. Esta estratégia assegura que, mesmo sem acesso à Internet, a Aplicação Móvel já disponha dos dados necessários carregados. Além disso, é responsável por devolver a listagem de auditorias e verificações a serem realizadas para a página das auditorias, proporcionando uma experiência fluída e consistente ao utilizador. No final, quando o utilizador termina uma auditoria é ele o responsável por sincronizar os dados com o Portal, caso seja possível, uma vez que pode não existir comunicação com o Portal;
- **AuditLegislationController**: Este controlador é responsável pela gestão da página referente à legislação. É nele que são realizadas as requisições necessárias para carregar as perguntas do formulário de legislação, garantindo que os dados corretos estejam disponíveis para o utilizador. Ao finalizar o preenchimento do formulário, o controlador é responsável por guardar as respostas fornecidas localmente;
- **AuditClinicInstallationController**: Este controlador tem como foco a gestão da página das clínicas e instalações. Possui um funcionamento semelhante ao do `AuditLegislationController`. Nele são realizados os pedidos para obter as perguntas do formulário relacionado às clínicas e instalações e, após o seu preenchimento, guarda as respostas fornecidas;
- **AuditVerificationController**: Este controlador é responsável pela gestão da página de verificações. Carrega as informações necessárias para a verificação selecionada, garantindo que todos os dados relevantes estejam acessíveis ao

utilizador. Após a verificação, o controlador também é responsável por guardar as informações respondidas.

5.3 MODELO DE DADOS

A escolha do *Isar* [38] como o motor de base de dados local para a Aplicação Móvel foi cuidadosamente considerada, considerando diversos fatores como desempenho, escalabilidade, facilidade de uso e segurança. Ao longo desta secção, serão discutidos os motivos que justificaram a adoção do *Isar*, além de uma breve comparação com outras alternativas disponíveis no mercado.

5.3.1 *Principais fatores na escolha do Isar*

Durante o processo de seleção de base de dados, várias alternativas foram consideradas, incluindo o SQLite[1], Hive[2], e Moor[3]. Embora cada uma dessas opções apresente diversas vantagens, o *Isar* destacou-se por vários motivos:

- **Desempenho**: *Isar* é altamente otimizado para dispositivos móveis, oferecendo um desempenho significativamente superior em comparação com o SQLite e Moor, especialmente em operações de leitura e escrita massivas. Essa eficiência é crucial para uma aplicação que exige a manipulação de grandes volumes de dados, como é o caso da Aplicação Móvel da ValGrupo.
- **Simplicidade**: Com o *Isar*, a criação de modelos de dados é simples e direta, permitindo que as mudanças sejam aplicadas rapidamente. Isso contrasta com o SQLite, onde a modelação pode ser mais complexa e requer migrações mais elaboradas quando ocorrem alterações nos esquemas.
- **Suporte a Consultas Não-Relacionais**: Como uma base de dados não-relacional, *Isar* permite uma maior flexibilidade na forma como os dados são armazenados e recuperados. Esta característica facilita a escalabilidade do sistema, uma vez que a estrutura dos dados pode evoluir sem a necessidade de redesenhar complexas relações entre tabelas, como seria necessário em uma base de dados relacional.

1 https://www.sqlite.org/
2 https://docs.hivedb.dev/
3 https://pub.dev/packages/moor

- **Sincronização Facilitada**: A arquitetura do *Isar* é altamente adequada para cenários de sincronização *offline-online*. Isso é essencial para a aplicação da ValGrupo, onde os dados precisam ser armazenados localmente e sincronizados posteriormente com o Portal de Gestão, garantindo que o fluxo de trabalho do utilizador não seja interrompido, mesmo em condições de conectividade instável.

5.3.2 *Escalabilidade e Validação de Dados com Isar*

Visto que a escalabilidade e a validação de dados são fatores cruciais para o sucesso de uma aplicação móvel, a escolha do *Isar* como base de dados local foi fundamentada tendo em conta essas necessidades.

Uma das principais vantagens do *Isar* é sua natureza não relacional, que permite uma escalabilidade mais eficiente à medida que a quantidade de dados armazenados cresce. Diferente de base de dados relacionais tradicionais, o *Isar* não depende de tabelas rigidamente definidas com relações complexas, o que facilita o crescimento e a manutenção do sistema. Essa característica é particularmente importante em ambientes móveis, onde a capacidade de armazenamento e processamento é limitada e a flexibilidade para adaptar-se a novas exigências de negócio é crucial. Além disso, a simplicidade na modelação de dados não relacionais permite uma adaptação mais rápida a mudanças nos requisitos, suportando um crescimento contínuo sem a necessidade de grandes reestruturações.

No que diz respeito à validação de dados, o *Isar* oferece um maior controlo por meio da definição precisa dos tipos de dados diretamente nos modelos. Por exemplo, ao criar um campo do tipo *int*, o *Isar* assegura que apenas valores inteiros serão aceites, evitando assim a introdução de dados inconsistentes ou inválidos na base de dados. Essa validação intrínseca ao modelo adiciona uma camada de segurança e integridade, garantindo que os dados armazenados sejam sempre do tipo esperado.

Além dessa validação, outras validações mais específicas, como a verificação se um valor numérico está dentro de um determinado intervalo, ou se um texto tem um comprimento mínimo, são implementadas diretamente nos *inputs* de cada formulário da aplicação. Isso permite que os dados sejam validados ainda antes de serem submetidos para a base de dados, assegurando que somente informações corretas e dentro dos parâmetros definidos sejam processadas e armazenadas.

5.3.3 *Sincronização com o Portal*

A sincronização dos dados entre a Aplicação Móvel e o Portal é um componente crítico para garantir a integridade e a consistência das informações. O utilizador ao iniciar a aplicação, após a mesma verificar se a versão instalada é a mais recente, inicia automaticamente o processo de sincronização.

Primeiramente, todos os dados não sincronizados, armazenados na base de dados, são enviados para o Portal de Gestão através da invocação do *endpoint* de uma API própria. Isso inclui auditorias concluídas, respostas a questionários e outras interações do utilizador realizadas enquanto o dispositivo estava *offline*.

Em seguida, o Portal processa os dados e retorna em formato JSON uma lista de IDs das auditorias e informações que foram armazenadas com sucesso. A aplicação utiliza esses IDs para remover as entradas correspondentes da base de dados local, garantindo que dados ainda pendentes de sincronização, ou com erros permaneçam para tentativas futuras de sincronização.

Esse processo assegura que a base de dados local seja mantida limpa e sincronizada com o sistema central, minimizando conflitos e redundância de informação.

5.3.4 *Representação Visual dos Modelos*

O diagrama apresentado (ver Figura 42), ilustra a relação entre os diferentes modelos, referidos na Secção 5.2.3, utilizados na Aplicação Móvel. Este diagrama facilita a compreensão das relações entre os modelos e a forma como os dados são estruturados e interligados dentro da aplicação.

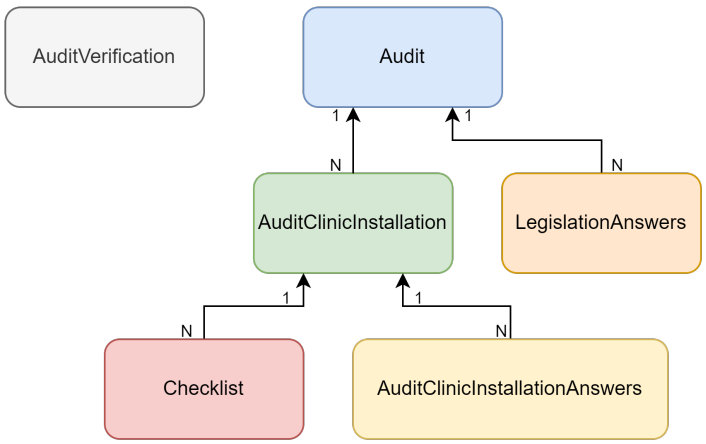


Figura 42: Diagrama de Modelos definidos na Aplicação Móvel

5.4 FUNCIONALIDADES DA APLICAÇÃO MÓVEL

Nesta secção, serão exploradas as principais características e funcionalidades, tanto visuais como não visuais, da Aplicação Móvel. O objetivo é proporcionar uma compreensão mais clara do estado funcional da aplicação, oferecendo uma breve explicação do funcionamento das páginas e destacando os detalhes mais relevantes.

5.4.1 *Layout Principal*

Um dos grandes objetivos desta Aplicação Móvel é suportar futuras adições de conteúdo. Deste modo, um dos focos principais da estrutura visual principal da aplicação foi permitir o crescimento e a facilidade de navegação. Em vez de se optar pelo atualmente popular menu na parte inferior da página com alguns botões, optou-se por posicionar as opções de navegação na lateral esquerda da aplicação, conforme ilustrado na Figura 43.

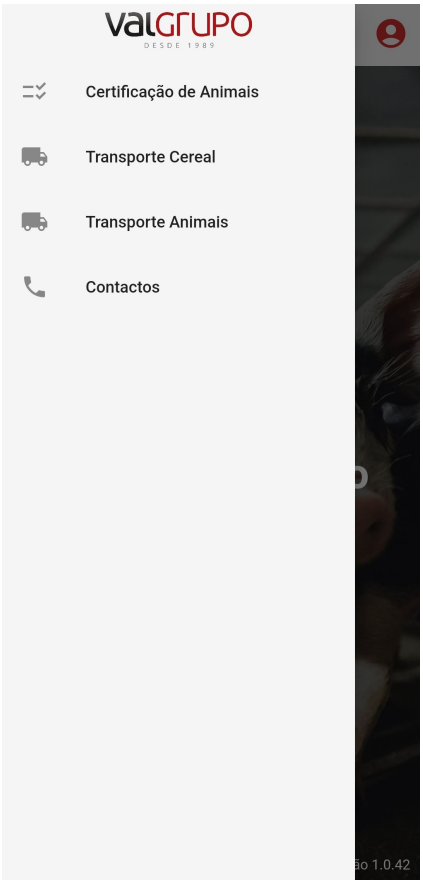


Figura 43: Navegação na lateral da Aplicação Móvel

Desta forma, no futuro, caso seja necessário adicionar mais páginas de acesso, basta incluir a nova página, escolher o ícone, definir a rota de destino e configurar as permissões de acesso numa lista, e a nova página será automaticamente adicionada à barra lateral. No entanto, é importante destacar que o utilizador final verá apenas as opções para as quais possui permissões de acesso. Poucos utilizadores terão tantas opções como o administrador, que terá acesso a todas as funcionalidades.

5.4.2 *Página de Login*

A página de *login* é a porta de entrada da Aplicação Móvel e foi projetada com o objetivo de ser simples e funcional. Ela consiste em apenas dois campos de entrada: um para o email e outro para a palavra-passe, Figura 44. Não foi necessária a inclusão de uma funcionalidade para criar novas contas de utilizador diretamente na Aplicação Móvel, dado que esta é uma ferramenta interna e é apenas utilizada por utilizadores previamente criados por utilizadores com permissão para tal.

Figura 44: Página de *Login* da Aplicação Móvel

Um ponto importante a destacar é a implementação de um mecanismo para contornar a falta de conectividade com a Internet. Quando não há acesso à Internet, o utilizador é redirecionado automaticamente para a página principal da aplicação, permitindo a utilização dos dados já disponíveis localmente. Posto isto, é importante notar que, neste modo *offline*, o utilizador terá acesso apenas aos dados e permissões que foram transferidos durante o último uso da aplicação com acesso à Internet, garantindo assim a continuidade do trabalho, mesmo sem uma conexão ativa.

Outro ponto importante foi a necessidade de reformular o processo de *login* no Portal, uma vez que, na Aplicação Móvel anterior, o *token* de autenticação era armazenado diretamente no código da aplicação, comprometendo a segurança. Para resolver esta questão, foi implementado um novo sistema de *login* para a Aplicação Móvel utilizando API *tokens*, geridos através do sistema Sanctum [39] do *Laravel*. Esta abordagem oferece uma solução mais segura e eficiente, permitindo um controle mais rigoroso sobre os utilizadores que possuem acesso ao Portal. Além disso, facilita a revogação de acesso, bastando apagar o *token* associado ao utilizador em questão.

5.4.3 *Página Principal dos Auditores*

De modo a que o utilizador consiga ver as auditorias que estão planeadas para si, foi criada uma página que possui em formato de lista de blocos as diferentes auditorias ou verificações planeadas para o utilizador com a sessão iniciada na Aplicação Móvel. De realçar que o administrador consegue visualizar as auditorias de todos os técnicos.

Figura 45: Página Principal dos Auditores da Aplicação Móvel

Esta lista está agrupada por data de agendamento da auditoria, e em cada bloco é possível observar as informações mais importantes referentes à auditoria, como a exploração, empresa detentora da exploração, tipo de auditoria e estado. Ao clicar no bloco, o utilizador é redirecionado para a página onde pode efetuar as diversas funções referentes à auditoria selecionada.

5.4.4 *Página com as Ações afetas aos Auditores*

Uma vez selecionada a auditoria que deseja ser efetuada pelo utilizador, caso o tipo de auditoria não seja uma "verificação", este é direcionado para a página de

ações da respetiva auditoria. Uma vez que existem dois tipos de explorações, sendo elas "Engordas"(EN) ou "Porcas e Leitões"(PL), e as auditorias para cada tipo de exploração é diferente, houve a necessidade de implementar de modo diferente as respetivas páginas. Na Figura 46, é possível observar esta página, tanto para as explorações de Engordas e as explorações de Porcas e Leitões. Neste caso a única diferença é que no caso da página referente às Engordas possuir um contador até 10 que é referente à quantidade de locais onde estão os animais para os quais se pretende introduzir a informação.

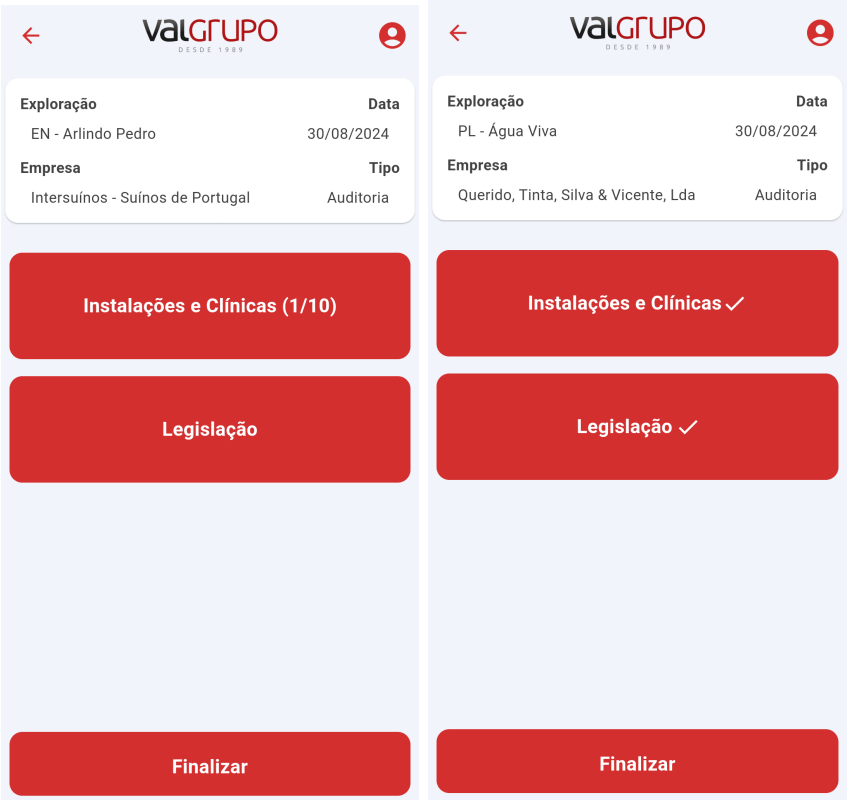


Figura 46: Página Ações dos Auditores da Aplicação Móvel

O símbolo do certo só aparece assim que seja respondido o formulário, de modo a facilitar uma visão geral da informação que já foi tratada. No caso em que possui o contador, a própria conotação já é um fator que permite visualizar o estado do processo e neste caso pode não existir a necessidade de efetuar a análise a todos os 10 locais, ou seja, por exemplo, o auditor pode finalizar uma auditoria analisando apenas 1 local.

5.4.5 *Página das clínicas e instalações*

A gestão eficaz das clínicas e instalações é um elemento crucial para garantir o cumprimento das normas e a manutenção da qualidade no ambiente auditado. Desta forma quando o utilizador seleciona a opção clínicas e instalações, dependendo do tipo de exploração, redirecionado para diferentes páginas.

No caso das explorações de Engordas será uma página que possui um campo para colocar a percentagem de mortalidade em média daquela exploração, e uma lista de 10 itens que representam os 10 locais onde se localizam os porcos, Figura 47.

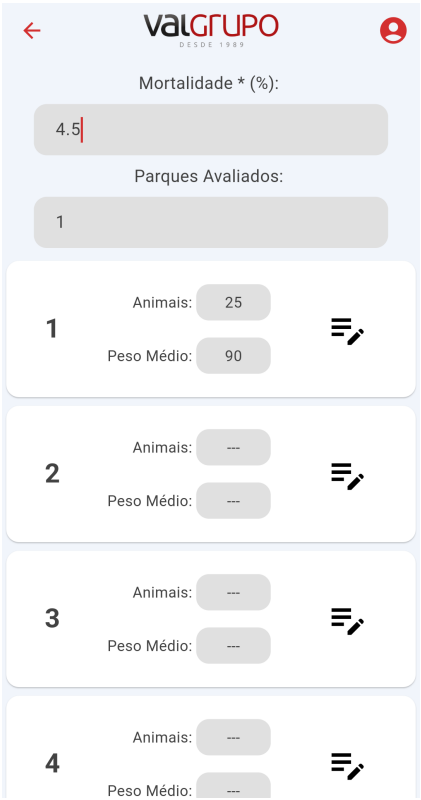


Figura 47: Página clínicas e instalações de exploração de engorda da Aplicação Móvel

Ao premir o *icon*, dentro desse item é possível responder a um questionário referente a esse local, Figura 48. Ao preencher essa informação e clicar em guardar, na página anteriormente referida, visualizada na Figura 47, aparecerá a informação referente ao peso médio e a quantidade de animais registadas.

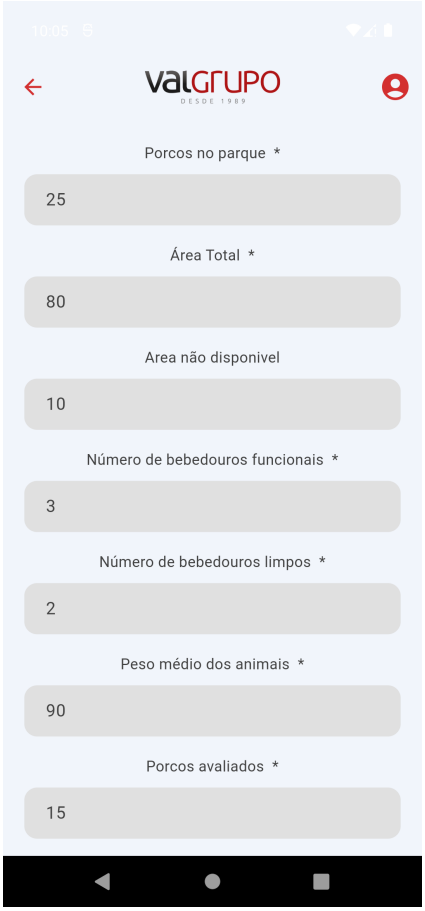


Figura 48: Página do questionário das clínicas e instalações da Aplicação Móvel

Uma vez que as instalações de Porcos de Engorda e de Porcas e Leitões são diferente, houve a necessidade de criar uma página extra. Nesta página o utilizador seleciona em que fase é que os leitões ou as porcas se encontram, Figura 49.

Figura 49: Página clínicas e instalações de exploração de porcas e leitões da Aplicação Móvel

Ao selecionar uma das opções, aparecer uma lista de itens, sendo a quantidade destes itens definida pela opção escolhida, de acordo com o apresentado na Figura 50. Está página acaba por funcionar da mesma maneira que a página referente às explorações de Engordas, e o utilizador ao responder ao questionário este é assinalado como estando "Respondido".

Figura 50: Lista de Parques a responder

5.4.6 *Página do Questionário da Legislação*

Na página das Ações dos Auditores, ao selecionar a opção legislação o utilizador é redirecionado para a um formulário, conforme visível na Figura 51. Este formulário apresenta diferentes perguntas, dependendo se se refere a explorações do tipo "Porcas e Leitões" ou se de "Engordas". No entanto, a lógica de funcionamento interno é semelhante.

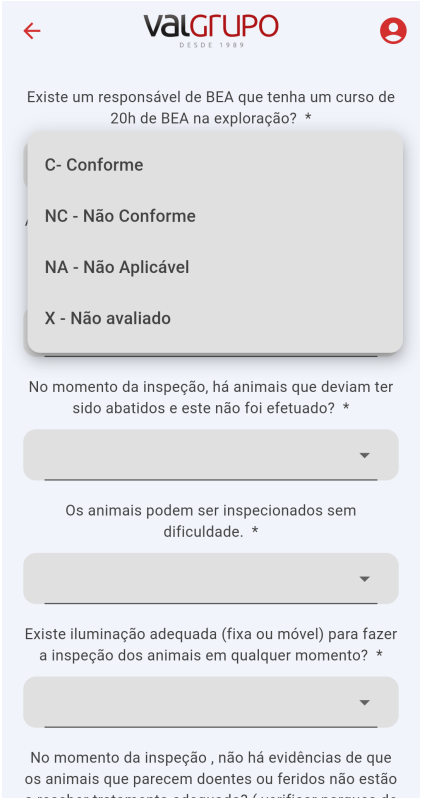


Figura 51: Formulário da Legislação da Aplicação Móvel

Este formulário é composto por diversas questões com diferentes opções, e quando o utilizador seleciona a opção "NC - Não Conforme", um *pop-up* é exibido, conforme ilustrado na Figura 52. Neste *pop-up*, o utilizador pode adicionar uma foto que demonstre a não conformidade, inserir uma observação para fornecer mais detalhes, e definir uma data limite para a resolução do problema. Esta data é posteriormente utilizada no Portal para gerar automaticamente uma auditoria do tipo "Verificação", agendada para o dia anterior à data selecionada.

Figura 52: Pop up da Não conformidade

5.4.7 *Página Verificação de Não Conformidades*

Quando o utilizador seleciona uma auditoria do tipo "Verificação", é redirecionado para uma página diferente da página das Ações dos Auditores, mencionada na Secção 5.4.4. Nesta página, ilustrada na Figura 53, o utilizador pode visualizar a data de resolução, a pergunta que foi identificada como uma não conformidade, a observação feita pelo auditor no momento da deteção da não conformidade, e as fotos anexadas como evidência. O utilizador tem então a opção de "Aprovar" ou "Rejeitar" a auditoria, com base na avaliação de se o problema foi ou não resolvido.

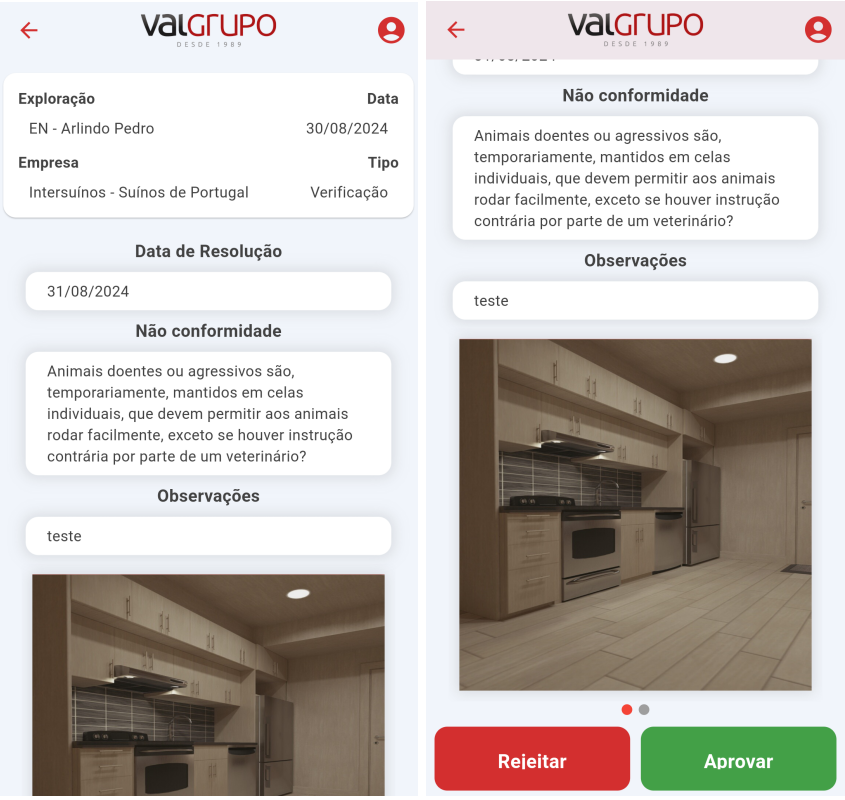


Figura 53: Página de Verificação

Quando todas as não conformidades de uma auditoria são aprovadas, o estado da auditoria é alterado para "Concluída". No entanto, essa alteração de estado pode ocorrer antes da aprovação de todas as não conformidades, caso as não conformidades identificadas não comprometam o funcionamento ou a conformidade de acordo com a legislação em vigor.

5.4.8 *Controlo de Versões*

No intuito de melhorar a assistência ao utilizador e facilitar a identificação de falhas na aplicação, na página principal da Aplicação Móvel foi introduzido um texto no canto inferior direito com o numero da versão da aplicação instalada, Figura 54.

Figura 54: Visualização da Versão da Aplicação Móvel

Ao efetuar o *login* na aplicação é enviada a informação referente à versão instalada da Aplicação Móvel para a API do Portal de Gestão. No Portal, caso ainda não existam dados referentes à versão da aplicação desse utilizador é guardada a informação, e caso já exista é apenas atualizada. Desta maneira é possível identificar facilmente quais os utilizadores que possuem a aplicação instalada, permitindo também identificar se estão a utilizar a versão mais recente.

5.4.9 *Sistema de atualização de Versão*

Dado que a aplicação necessita de atualizações periódicas e, por ser uma aplicação interna, não pode utilizar sistemas como o *Google Play Store* [40] para validar automaticamente a disponibilidade de novas versões, foi necessário desenvolver uma solução prática e eficiente para garantir que os utilizadores possam manter a aplicação atualizada.

Com isso em mente, implementou-se um sistema de verificação de versão na aplicação. Sempre que a aplicação é iniciada, é realizado um pedido à API do Portal de Gestão para verificar qual é a versão mais recente disponível, e é efetuada uma comparação com a versão atualmente instalada no dispositivo móvel. Se forem detetadas diferenças entre as versões, o utilizador é notificado imediatamente sobre a necessidade de atualização da mesma.. Esta atualização é obrigatória para garantir que todos os utilizadores estejam sempre a utilizar a versão mais atual, que inclui as últimas funcionalidades e correções de segurança.

Na Figura 55 é possível observar a função que verifica a existência de uma atualização. A função `checkConnectivity` permite ter acesso às ligações disponíveis no dispositivo móvel, sejam elas dados móveis, bluetooth, wifi, etc. Após obter essa informação é validado se é dados móveis (*mobile*) ou se é wifi, caso exista uma das duas é obtido a versão atual da Aplicação Móvel através do `PackageInfo.fromPlatform` e feito o pedido ao Portal para saber a versão mais recente, utilizando a função `Auth.getAppVersion`. De seguida, comparando ambos os valores, se forem diferentes é despoletado um *pop-up* através da função `vgDialog`, que informa o utilizador que existe uma nova versão. Caso a aplicação já se encontre na versão mais recente é chamada a função `deleteUpdateFileFromPrivateStorage` que apaga, caso exista, o *Android Application Package* (APK) que possa ter sido descarregado anteriormente.

```
void checkVersion() async {
    var connectivityResult = await (Connectivity().checkConnectivity());

    if (connectivityResult == ConnectivityResult.wifi ||
        connectivityResult == ConnectivityResult.mobile) {
      PackageInfo packageInfo = await PackageInfo.fromPlatform();
      if ((await Auth.getAppVersion()) != packageInfo.version) {
        vgDialog();
      } else {
        deleteUpdateFileFromPrivateStorage();
      }
    }
  }
```

Figura 55: Código de Verificação de Versão da Aplicação Móvel

Ao optar por atualizar, o novo APK é descarregado e instalado automaticamente pela própria aplicação, sem que o utilizador precise sair da interface da aplicação. Este processo garante uma experiência de utilizador fluida e sem interrupções, minimizando o tempo de inatividade e a necessidade de suporte técnico.

Para possibilitar este processo de atualização automática, foi criado um *endpoint* público na API do Portal de Gestão que armazena o APK mais recente. Este *endpoint* permite que a aplicação aceda ao ficheiro mais atualizado de forma segura e confiável, garantindo que todos os utilizadores tenham acesso às melhorias e correções implementadas na última versão.

5.5 TESTES E RESULTADOS

Assim que o desenvolvimento da Aplicação Móvel e do Portal de Gestão foi concluído, iniciou-se o processo de testes. Este processo não foi realizado pela equipa de desenvolvimento diretamente, pelo que não será abordado em profundidade neste documento. No entanto, os testes consistiram em colocar o *branch* principal das auditorias no servidor de testes, no *branch* de *Staging* e gerar um APK de teste que apontava para a API colocada nesse servidor. Este APK foi, então, instalado no dispositivo móvel de um dos gestores de auditores, que realizou uma série de testes práticos de usabilidade e funcionalidade, simulando o uso real da aplicação.

Com esses testes houve a necessidade de realizar algumas correções e pequenas atualizações, no entanto o conceito geral encontrava-se implementado.

Uma das áreas mais importantes testadas foi a sincronização entre a Aplicação Móvel e o Portal de Gestão, um ponto crítico para garantir a consistência dos dados de auditorias. Estes testes confirmaram que o processo de sincronização funcionava corretamente, com a aplicação móvel a conseguir armazenar dados localmente, e sincronizá-los com o Portal, assim que a conectividade fosse restaurada.

Este processo de testes e correções resultou numa Aplicação Móvel robusta e eficiente, que cumpria com os requisitos pretendidos. A aplicação agora permite aos utilizadores realizar auditorias de forma eficiente, tanto *online* quanto *offline*, e garantir que os dados são armazenados com segurança. Além disso, a arquitetura modular da aplicação facilita futuras atualizações e adições de novas funcionalidades.

6

CONCLUSÃO

Este estágio proporcionou uma grande aprendizagem e evolução profissional, permitindo-me adquirir novas competências e enfrentar novos desafios. O objetivo principal de desenvolver uma nova Aplicação Móvel e melhorar o Portal de Gestão do ValGrupo foi atingido, proporcionando soluções tecnológicas que melhoraram a eficiência dos processos internos do grupo.

A mudança na metodologia de desenvolvimento, foi um ponto importante, pois permitiu uma melhor organização do trabalho e uma resposta mais rápida às mudanças e necessidades do ValGrupo. Esta experiência permitiu-me compreender melhor a importância de uma metodologia bem estruturada e colaborativa.

No que diz respeito ao conhecimento adquirido, o estágio proporcionou-me uma aprendizagem valiosa, tanto no domínio técnico, como no desenvolvimento pessoal. A integração das tecnologias utilizadas, como *Flutter*, *Laravel* e *Vue*, foi um processo que exigiu dedicação e a superação de diversos obstáculos, mas que resultou num produto final robusto e perfeitamente alinhado com as necessidades da empresa.

Com este conhecimento, foi possível alcançar uma transformação significativa na forma como as auditorias são geridas e executadas dentro do ValGrupo. A nova Aplicação Móvel, em conjunto com o Portal de Gestão, permitiu automatizar grande parte dos processos, desde o agendamento até à execução das auditorias, garantindo uma maior eficiência e precisão na recolha de dados. A funcionalidade de sincronização *offline* assegura que os colaboradores no terreno possam realizar auditorias em locais remotos, sem dependência de acesso a *Internet*, enquanto a integração com o Portal facilita o acesso e visão dos dados das auditorias, proporcionando uma visão mais clara das operações. Um dos grandes benefícios desta transformação foi a redução no consumo de papel, já que alguns dos processos anteriormente realizados manualmente foram digitalizados. Além disso, a capacidade de gerar relatórios detalhados e monitorizar as não conformidades permitiu melhorar o controlo de qualidade e a resposta a problemas operacionais.

Ao longo deste estágio, tornou-se claro a importância do desenvolvimento de software interno para o funcionamento eficaz das empresas. A criação de soluções personalizadas, como o Portal de Gestão e a Aplicação Móvel da ValGrupo, não

só permite otimizar processos, como também garantem uma melhor adaptação às necessidades específicas do grupo. A gestão de dados, a automatização de tarefas e a integração entre diferentes departamentos desempenham um papel vital na eficiência e na competitividade da empresa.

Em termos de trabalho futuro, tanto o Portal como a Aplicação Móvel foram desenvolvidos tendo em mente a capacidade de suportar novas funcionalidades e melhorias. A Aplicação Móvel foi desenvolvida com uma arquitetura modular o que oferece grande flexibilidade, permitindo a integração de novas funcionalidades de forma rápida e eficiente, sem comprometer a estabilidade ou o desempenho das restantes funcionalidades. Esta modularidade é importante para garantir que a aplicação possa evoluir à medida que as necessidades do ValGrupo mudam, seja para integrar novos processos ou responder a exigências regulamentares.

Do mesmo modo, o Portal de Gestão foi concebido para ser constantemente atualizado, garantindo a sua capacidade de adaptação às evoluções tecnológicas e às exigências operacionais. A estrutura do Portal permite a adição de novos módulos ou funcionalidades que possam melhorar a gestão interna, a análise de dados ou a interação entre diferentes departamentos. Além disso, o foco na segurança e na escalabilidade garante que o sistema continuará a ser um pilar central na gestão dos processos da ValGrupo, suportando o crescimento da empresa sem perda de eficiência ou fiabilidade.

Apesar de não ser requisito prioritário penso que no futuro seria interessante a aplicação evoluir de modo a integrar ferramentas de análise através de gráficos diretamente no Portal, que permitiriam à ValGrupo uma visão mais clara do estado das auditorias.

BIBLIOGRAFIA

[1] iconO2, *Início ValGrupo - transformação e criação agropecuária*, [Online; accessed 22. Apr. 2024], abr. de 2024. URL: https://www.valgrupo.pt.

[2] *Xamarin | Open-source mobile app platform for .NET*, [Online; accessed 22. Apr. 2024], abr. de 2024. URL: https://dotnet.microsoft.com/en-us/apps/xamarin.

[3] [Online; accessed 22. Apr. 2024], mar. de 2023. URL: https://www.portugalexporta.pt/sites/default/files/2023-03/ficha-entrada-mercado_china_carne-porco-congelada.pdf.

[4] *Eticadata – Software de gestão*, [Online; accessed 22. Apr. 2024], abr. de 2024. URL: https://eticadata.com.

[5] *SAP Portugal Products and Services Inquiries*, [Online; accessed 22. Apr. 2024], abr. de 2024. URL: https://www.sap.com/portugal/index.html?url_id=auto_hp_redirect_portugal.

[6] *Laravel - The PHP Framework For Web Artisans*, [Online; accessed 22. Apr. 2024], abr. de 2024. URL: https://laravel.com.

[7] *Vue.js*, [Online; accessed 22. Apr. 2024], abr. de 2024. URL: https://vuejs.org.

[8] Atlassian, *Jira | Issue & Project Tracking Software | Atlassian*, [Online; accessed 17. Apr. 2024], abr. de 2024. URL: https://www.atlassian.com/software/jira.

[9] Atlassian, *Bitbucket*, [Online; accessed 1. Apr. 2024], abr. de 2024. URL: https://bitbucket.org/product.

[10] *Home | Scrum Guides*, [Online; accessed 31. Mar. 2024], nov. de 2023. URL: https://scrumguides.org/index.html.

[11] *What is Scrum?*, [Online; accessed 1. Apr. 2024], mar. de 2024. URL: https://www.scrum.org/resources/what-scrum-module.

[12] L. D. Beju e S. Legutko, «Kanban Systems in the Context of the Enterprise Systems», *MATEC Web of Conferences*, vol. 343, n.º 2-3, p. 03 001, jan. de 2021, ISSN: 2261-236X. DOI: 10.1051/matecconf/202134303001.

[13] Atlassian, *Collaboration software for software, IT and business teams*, [Online; accessed 17. Apr. 2024], abr. de 2024. URL: https://www.atlassian.com.

[14] *GitHub: Let's build from here*, [Online; accessed 17. Apr. 2024], abr. de 2024. URL: https://github.com.

[15] *The most-comprehensive AI-powered DevSecOps platform*, [Online; accessed 17. Apr. 2024], abr. de 2024. URL: https://about.gitlab.com.

[16] *Jenkins*, [Online; accessed 17. Apr. 2024], abr. de 2024. URL: https://www.jenkins.io.

[17] Slack, *Slack is your productivity platform*, [Online; accessed 17. Apr. 2024], abr. de 2024. URL: https://slack.com.

[18] *Laravel - Blade*, [Online; accessed 4. Jul. 2024], jul. de 2024. URL: https://laravel.com/docs/11.x/blade.

[19] *Vue.js - Ways of Using Vue*, [Online; accessed 4. Jul. 2024], jul. de 2024. URL: https://vuejs.org/guide/extras/ways-of-using-vue.

[20] Team, «Single-Page Application vs Multi-Page Application: Pros, Cons, and Which is Better?», *Lvivity*, mar. de 2023. URL: https://lvivity.com/single-page-app-vs-multi-page-app.

[21] Ibm, *REST APIs*, [Online; accessed 23. Sep. 2024], ago. de 2024. URL: https://www.ibm.com/topics/rest-apis.

[22] *Xamarin official support policy | .NET*, [Online; accessed 6. Apr. 2024], abr. de 2024. URL: https://dotnet.microsoft.com/en-us/platform/support/policy/xamarin.

[23] *React Native · Learn once, write anywhere*, [Online; accessed 4. Jul. 2024], jul. de 2024. URL: https://reactnative.dev.

[24] *Ionic Framework - The Cross-Platform App Development Leader*, [Online; accessed 4. Jul. 2024], jul. de 2024. URL: https://ionicframework.com.

[25] *Flutter - Build apps for any screen*, [Online; accessed 4. Jul. 2024], jun. de 2024. URL: https://flutter.dev.

[26] *Dart programming language*, [Online; accessed 4. Jul. 2024], jul. de 2024. URL: https://dart.dev.

[27] *Flutter - Hot reload*, [Online; accessed 4. Jul. 2024], jul. de 2024. URL: https://docs.flutter.dev/tools/hot-reload.

[28] *Rational Application Developer for WebSphere Software 9.7.0*, [Online; accessed 11. Sep. 2024], mar. de 2021. URL: https://www.ibm.com/docs/en/radfws/9.7?topic=cycle-model-view-controller-architecture.

[29] *Laravel - CSRF Protection*, [Online; accessed 12. Sep. 2024], set. de 2024. URL: https://laravel.com/docs/11.x/csrf.

[30] *Laravel - Eloquent*, [Online; accessed 11. Sep. 2024], set. de 2024. URL: https://laravel.com/docs/master/eloquent.

[31] *Single Page App: SPA Frameworks & Their Role in Development*, [Online; accessed 12. Sep. 2024], set. de 2024. URL: https://www.outsystems.com/tech-hub/app-dev/what-is-single-page-application/#definition.

[32] vuetifyjs, *vuetify - Calendar*, [Online; accessed 17. Sep. 2024], set. de 2024. URL: https://github.com/vuetifyjs/vuetify/tree/master/packages/docs/src/examples/v-calendar.

[33] nathanreyes, *v-calendar*, [Online; accessed 17. Sep. 2024], set. de 2024. URL: https://github.com/nathanreyes/v-calendar.

[34] richardtallent, *vue-simple-calendar*, [Online; accessed 17. Sep. 2024], set. de 2024. URL: https://github.com/richardtallent/vue-simple-calendar.

[35] nhn, *tui.calendar*, [Online; accessed 12. Sep. 2024], set. de 2024. URL: https://github.com/nhn/tui.calendar.

[36] J. net, *Welfare Quality*, [Online; accessed 11. Sep. 2024], set. de 2024. URL: https://www1.clermont.inrae.fr/wq/?simu=on.

[37] Sadika, «The MVC Architecture - Sadika - Medium», *Medium*, set. de 2023, ISSN: 9747-0712. URL: https://medium.com/@sadikarahmantanisha/the-mvc-architecture-97d47e071eb2.

[38] *Home | Isar Database*, [Online; accessed 12. Sep. 2024], mai. de 2024. URL: https://isar.dev.

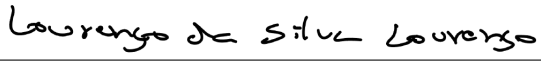
[39] *Laravel - Sanctum*, [Online; accessed 12. Sep. 2024], set. de 2024. URL: https://laravel.com/docs/11.x/sanctum.

[40] *Android Apps on Google Play*, [Online; accessed 12. Sep. 2024], set. de 2024. URL: https://play.google.com/store/games?hl=en&pli=1.

DECLARAÇÃO

Declaro, sob compromisso de honra, que o trabalho apresentado nesta dissertação, com o título *"Desenvolvimento de Aplicação Móvel para Integração e Melhoria de Processos em Plataforma Web"*, é original e foi realizado por Lourenço da Silva Lourenço (2220649) sob orientação de Professora Doutora Marisa da Silva Maximiano (marisa.maximiano@ipleiria.pt).

Leiria, setembro de 2024



Lourenço da Silva Lourenço