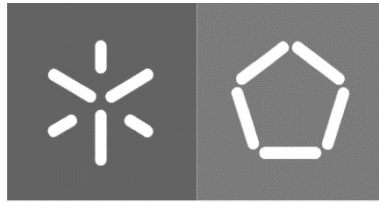




Ana Patrícia Pires Martins

**Desenvolvimento de uma Aplicação
de Gestão de Dados para Empresas
de Retalho**



Universidade do Minho
Escola de Engenharia

Ana Patrícia Pires Martins

**Desenvolvimento de uma aplicação de
gestão de dados para empresas de retalho**

Dissertação de Mestrado
Mestrado em Engenharia de Sistemas

Trabalho efetuado sob a orientação de
**Professora Doutora Ana Maria Alves Coutinho
Rocha**
**Professor Doutor Francisco José Monteiro
Duarte**

outubro de 2023

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Licença concedida aos utilizadores deste trabalho:



CC BY-NC-SA

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

AGRADECIMENTOS

A minha jornada nos últimos dois anos, como estudante de ensino superior, foi marcada por desafios e conquistas. o Mestrado em Engenharia de Sistemas, ampliou significativamente o meu campo de estudo e aprofundou o meu entendimento em engenharia.

Gostaria de expressar a minha gratidão à minha família, em especial aos meus pais, pelo constante apoio e motivação ao longo da minha jornada académica.

A *Accenture*, por me ter recebido de braços abertos no desenvolvimento deste projeto cativante e pela primeira oportunidade no mundo de trabalho. Além disso, pelos ensinamentos das boas práticas de trabalho, dando destaque à Alexandra, Mizael e Guilherme. Mas também aos meus colegas de equipa.

Aos professores Ana Maria Alves Coutinho Rocha e Francisco José Monteiro Duarte, que me proporcionaram orientação valiosa neste desafio.

A Universidade do Minho desempenhou um papel fundamental na minha trajetória.

Finalmente, gostaria de agradecer a todos os meus amigos que compartilharam comigo as alegrias e desafios desta jornada.

O meu percurso académico na Universidade do Minho moldou-me e proporcionou-me oportunidades de crescimento. Estou ansiosa pelo que o futuro reserva e pelo impacto que posso criar na área da Engenharia de Sistemas.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Universidade do Minho, Braga, outubro 2023

*“Many of the things that seem impossible now,
will become realities tomorrow.”*

-Walt Disney

RESUMO

Desenvolvimento de uma aplicação de gestão de dados para empresas de retalho

Este trabalho aborda a partilha de dados por parte dos utilizadores nas redes sociais, bem como a utilização desses dados para publicidade personalizada e o crescimento das empresas. Muitas vezes, os utilizadores não tem plena consciência do tipo das informações que estão a partilhar.

Como resposta a este desafio, propõe-se uma solução inovadora. O principal objetivo é incentivar os utilizadores, recompensando-os pela utilização comercial dos seus dados pessoais. Pretende-se, assim, criar uma plataforma que seja atrativa para todas as partes envolvidas no ecossistema de partilha de dados.

O objetivo central deste documento é documentar de forma abrangente todo o ciclo de vida de um projeto de desenvolvimento de uma aplicação. Esta aplicação foi concebida para permitir que os utilizadores monitorizem a partilha dos dados pessoais. Esta abordagem inovadora confere aos utilizadores a autonomia para escolher seletivamente quais as partes dos dados pessoais que desejam partilhar, abrangendo informações desde redes sociais até dados de localização.

Para alcançar este objetivo ambicioso, o processo de desenvolvimento de *software* faz uso de uma ampla gama de tecnologias de ponta. Estas tecnologias são cuidadosamente selecionadas para garantir que a aplicação possa ser implantada e utilizada de forma fluida em dispositivos *iOS* e *Android*, simplificando o processo de desenvolvimento *mobile*. Além disso, várias outras tecnologias são integradas para facilitar a implementação global do projeto.

É importante sublinhar que este projeto não apenas inova com um conceito pioneiro, mas também possui um significativo potencial de comercialização. No entanto, é fundamental salientar que a aplicação encontra-se atualmente numa fase inicial de desenvolvimento, necessitando de refinamento adicional e de um foco contínuo para alcançar uma prontidão operacional completa antes de entrar no mercado.

Palavras-chave: Aplicação, *Big Data*, Engenharia de *software*, Extração de dados, Gestão de campanhas, Metodologia de desenvolvimento ágil

ABSTRACT

Development of a data management application for retail companies

This work discusses data sharing by users on social networks, as well as the use of this data for personalized advertising and for companies growth. Users are often not fully aware of the type of information they are sharing.

In response to this challenge, an innovative solution is proposed. The main objective is to encourage users by rewarding them for the commercial use of their personal data. The aim is to create a platform that is attractive to all parties involved in the data-sharing ecosystem.

The main goal of this document is to comprehensively document the entire life cycle of an application development project. This application is designed to allow users to monitor the sharing of their personal data. This innovative approach gives users the autonomy to selectively choose which parts of their personal data they wish to share, encompassing information from social networks to location data.

To achieve this ambitious goal, the software development process makes use of a wide range of cutting-edge technologies. These technologies are carefully selected to ensure that the application can be implemented and used fluidly on iOS and Android devices, simplifying the mobile development process. In addition, various other technologies are integrated to facilitate the overall implementation of the project.

It is important to underline that this project not only innovates with a pioneering concept, but also holds significant commercialization potential. However, it is crucial to emphasize that the application is currently at an early stage of development, requiring further refinement and continuous focus to achieve full operational readiness before going being launched onto the market.

Keywords: Agile development methodology, Application (App), Big Data, Campaign management, Data extraction, Software engineering

TABELA DE CONTEÚDO

1	Introdução	1
1.1	Enquadramento	1
1.2	Objetivos	3
1.3	Metodologia de trabalho	5
1.4	Estrutura de relatório	12
2	Processos e requisitos de software	14
2.1	Introdução	14
2.2	Processos de desenvolvimento de <i>software</i>	14
2.2.1	Gestão de projetos	14
2.2.2	Processos tradicionais	16
2.2.3	Processos ágeis	17
2.3	Requisitos de <i>software</i>	20
2.4	Conclusão	21
3	Tecnologias e frameworks utilizadas na gestão de dados para empresas de retalho	23
3.1	Introdução	23
3.2	Tecnologias para <i>Frontend</i>	23
3.3	Tecnologias para Backend	27
3.4	Outras tecnologias utilizadas	30
3.5	Conclusão	32
4	Análise de requisitos e Arquitetura de software	34
4.1	Introdução	34
4.2	Requisitos funcionais	34
4.3	Requisitos não funcionais	40
4.4	Arquitetura de <i>software</i> e Plataforma tecnológica	41
4.5	Modelo de dados	45
4.6	Conclusão	46
5	Desenvolvimento da solução	48
5.1	Introdução	48

5.2	Componentes de Frontend	48
5.2.1	Ecrãs para o registo de conta	48
5.2.2	Ecrãs de <i>Log In</i>	49
5.2.3	Ecrãs de navegação	50
5.3	Configurações iniciais	52
5.3.1	Arquitetura do <i>Frontend</i>	53
5.3.2	Desenvolvimento da API	60
5.3.3	Firebase	61
5.3.4	Pub/Sub	63
5.3.5	Big Query	64
5.4	Criação de uma campanha	67
5.5	Conclusão	70
6	Utilização da metodologia Scrum	72
6.1	Introdução	72
6.2	Implementação do <i>Scrum</i>	72
6.3	Divisão dos <i>Sprints</i>	76
6.4	Conclusão	84
7	Considerações finais	86
7.1	Conclusão	86
7.2	Análise crítica	87
7.3	Trabalho futuro	88
	Referências	90

ÍNDICE DE FIGURAS

Figura 1: Accenture logo	3
Figura 2: Diagrama <i>Bull's-Eye</i> do “Take My Data”	6
Figura 3: Detalhe das funcionalidades com maior prioridade	7
Figura 4: Journey step 1	7
Figura 5: Journey step 2	8
Figura 6: Journey step 3	9
Figura 7: Journey step 4	9
Figura 8: Journey step 5	10
Figura 9: Journey step 6	11
Figura 10: Journey step 7	11
Figura 11: Journey step 8	12
Figura 12: Journey step 9	12
Figura 13: Características chave de um projeto (adaptado de PM ² (European Commission, 2016))	15
Figura 14: Modelo em cascata (adaptado de Sommerville (2016))	16
Figura 15: <i>Scrum</i> Life cycle. Fonte: Scrum.org (2021)	18
Figura 16: Estrutura do React Native. Fonte: Frachet (2020)	24
Figura 17: Classificação das linguagens de programação mais populares. Fonte: Stack Overflow (2022)	25
Figura 18: Estrutura dos dados nas diferentes bases de dados	30
Figura 19: Produtos da Google Cloud Platform	31
Figura 20: Diagrama de caso de uso	35
Figura 21: Diagrama de “Gerir perfil” mais detalhado	35
Figura 22: Diagrama com o caso de uso “Gerir Campanha” mais detalhado	36
Figura 23: Diagrama de arquitetura de alto nível	42
Figura 24: Plataforma Tecnológica	43
Figura 25: Esquema da Base de dados relacional	45
Figura 26: Registo de conta do utilizador	49
Figura 27: Entrada	50
Figura 28: Ecrãs de navegação	51
Figura 29: Git Flow, Fonte: Buddy: The DevOps Automation Platform (2020)	52
Figura 30: Arquitetura de pastas <i>Frontend</i>	53

Figura 31: Data Sharing	55
Figura 32: Ecrã da aplicação ainda em fase de desenvolvimento e o pedido de permissão ao <i>Expo</i>	56
Figura 33: Menu Principal	57
Figura 34: Barra de baixo da aplicação	57
Figura 35: Ecrã com os detalhes de uma campanha	58
Figura 36: Arquitetura de pastas <i>API</i>	59
Figura 37: Capturar dados através do endpoint <i>Debug Token</i>	60
Figura 38: Dados recebidos através do endpoint <i>Debug Token</i>	61
Figura 39: Lógica desenvolvida para executar um refresh Token à API do Facebook	61
Figura 40: Modelação dos dados para a Cloud Firestore	62
Figura 41: Firestore	62
Figura 42: Serviço "Publish"	63
Figura 43: Serviço "addColumnToDocument"	64
Figura 44: Método Post	64
Figura 45: Exemplo de "Personas"	65
Figura 46: Tabela segmentação	66
Figura 47: Tabela UserSegmentation	66
Figura 48: <i>Layout</i> da pagina web para a criação de uma campanha	67
Figura 49: Exemplo de seleção para publico-alvo	68
Figura 50: Visualização da campanha na aplicação	69
Figura 51: Visualização da campanha na barra de notificações	69
Figura 52: Visualização da campanha em detalhe	70

ÍNDICE DE TABELAS

Tabela 1: Detalhes da Implementação do <i>Scrum</i> no projeto	72
Tabela 2: Detalhes dos eventos do <i>Scrum</i>	74
Tabela 3: Propósitos dos Eventos <i>Scrum</i>	74
Tabela 4: Questões principais nos Eventos	75
Tabela 5: Atividades e <i>Story Points</i> do <i>Sprint</i> 1	76
Tabela 6: Atividades e <i>Story Points</i> do <i>Sprint</i> 2	78
Tabela 7: Atividades e <i>Story Points</i> do <i>Sprint</i> 3	79
Tabela 8: Atividades e <i>Story Points</i> do <i>Sprint</i> 4	81
Tabela 9: Atividades e <i>Story Points</i> do <i>Sprint</i> 5 e <i>Sprint</i> 6	82
Tabela 10: Atividades e <i>Story Points</i> do <i>Sprint</i> 7	83
Tabela 11: Atividades e <i>Story Points</i> do <i>Sprint</i> 8	84
Tabela 12: Resumo do Estado das Atividades por <i>Sprint</i>	89

SIGLAS E ACRÓNIMOS

ACID Atomicity, Consistency, Isolation, Durability.

API Application Programming Interface.

BaaS Backend-as-a-Service.

BLL Business Logic Layer.

CRUD Create, Read, Update e Delete.

DAL Data Access Layer.

ETL Extract, Transform, Load.

FCM Firebase Cloud Messaging.

FP Functional Programming.

FRP Functional Reactive Programming.

GCP Google Cloud Platform.

HTTP Hypertext Transfer Protocol.

IoT Internet of Things.

Json JavaScript Object Notation.

KA Area of Knowledge.

MVP Minimum Viable Product.

OOP Object-Oriented Programming.

PL Presentation Layer.

PMO Project Management Office.

PM² Project Management Methodology.

PO Product Owner.

PR Pull Request.

SM Scrum Master.

SQL Structured Query Language.

SWEBOK Guide Guide to the Software Engineering Body of Knowledge.

UX User Experience.

GLOSSÁRIO

Backend Parte de um sistema ou aplicação que não está ligado diretamente ao utilizador.

Backlog Lista de tarefas num projeto de desenvolvimento a serem executadas.

Backoffice Software que uma empresa usa para administrar operações e interfaces que não são vistas pelo consumidor.

Bull's-Eye Ferramenta que permite à equipa clarificar as prioridades antes de tomar decisões.

CSS É uma linguagem de estilo usada para descrever a apresentação de um documento escrito em HTML ou em XML. O CSS foi projetado para permitir a separação de conteúdo e apresentação incluindo layout, cores e fontes.

Dashboard Interface gráfica do utilizador, que geralmente fornece visualizações rápidas dos principais indicadores de desempenho relevantes para um objetivo ou processo de negócios específico.

Framework Biblioteca que aborda funcionalidades e estruturas, para o desenvolvimento de aplicações, a fim de fornecer soluções para um mesmo domínio de problema, permitindo assim reutilização do código.

Git Sistema que permite gerir e controlar as mudanças no código-fonte de um projeto. Permite a colaboração mantendo um registo de histórico das alterações, ramificações do código e facilita a fusão de branch.

JavaScript Linguagem de programação.

Journey Conjunto de etapas que o sistema percorre para atingir determinado objetivo.

Marketplace Plataforma de vendas online.

Meta for Developer Plataforma que fornece aos desenvolvedores ferramentas e recursos para criar aplicações que se integram com o *Facebook*.

Serverless Modelo de execução de programação na nuvem (cloud), em que o fornecedor da cloud, como Google Cloud, fornece recursos de servidores dependendo da procura, sendo que cuidam dos servidores pelos clientes.

Token Valor monetário.

1 INTRODUÇÃO

Neste capítulo, são introduzidos conceitos que serão explorados ao longo da dissertação. São descritos os objetivos, a estrutura do relatório, bem como a metodologia de trabalho adotada para alcançar os objetivos pretendidos. Por fim, é apresentada uma descrição da estrutura do documento.

1.1 Enquadramento

A evolução tecnológica destacou a crescente importância do *Big Data* na indústria e, tornou-se uma área essencial na era de informação que vivemos. No entanto, o acesso e quantidade a este volume de dados suscita preocupações no que respeita a quem detém os dados e quem possui direitos sobre eles (Cole et al., 2015). Os principais fornecedores de comércio eletrónico, como a *Amazon* e o *eBay*, conseguiram uma transformação significativa do mercado através das suas plataformas inovadoras e altamente escaláveis, e dos seus sistemas de recomendação de produtos (Chen et al., 2012). Atualmente, as empresas não se limitam apenas a recolher dados, têm também um forte interesse em compreender o seu significado e importância ao utilizá-los na tomada de decisões. Todas as economias, produções, organizações, funções empresariais e indivíduos dependem agora dos dados de alguma forma. A utilização da *Internet*, dos *smartphones*, das redes sociais, o desenvolvimento da computação omnipresente e vários outros avanços tecnológicos estão a contribuir para o aumento do volume de dados a nível mundial (Byali, 2022). A análise de grandes volumes de dados reveste-se de grande importância numa era de sobrecarga de dados e pode proporcionar perspetivas e benefícios inesperados na tomada de decisões globais em vários sectores (Nitnaware et al., 2023).

As grandes empresas como a *Google* e o *Facebook*, continuam a liderar o desenvolvimento de plataformas de análise da *Web* e de redes sociais. A emergência de conteúdos da *Web 2.0* gerados pelos clientes em vários fóruns, grupos de discussão, plataformas de redes sociais e sistemas de *crowdsourcing* oferece outra oportunidade para os investigadores e profissionais “ouvirem” a voz do mercado de um vasto número de constituintes das empresas, incluindo clientes, funcionários, investidores e os meios de comunicação social (Sangeetha et al., 2018). Ao contrário dos registos de transações tradicionais recolhidos a partir de vários sistemas da década de 1980, os dados que os sistemas de comércio eletrónico recolhem da *Web* são menos estruturados e contém informações ricas sobre a opinião e o comportamento dos clientes.

A análise de grandes volumes de dados é o ato de analisar grandes quantidades de dados de vários tipos,

para encontrar padrões, relações desconhecidas e outras informações valiosas (Ali et al., 2019). Esse conhecimento, pode dar a uma organização uma vantagem competitiva sobre os seus rivais e ter um impacto positivo nos resultados da empresa, através de um marketing melhorado e de um aumento das vendas (Reddy et al., 2022).

Tendo em conta a evolução verificada na tecnologia é importante realçar o que implica o consentimento da partilha destes dados. Os três "V" dos megadados - Velocidade, Volume e Variedade - ampliam as vulnerabilidades de segurança e privacidade. Estas variáveis incluem infra-estruturas de *cloud*, variedade de fontes e formatos de dados, aquisição de dados que ocorrem em fluxos e uma quantidade crescente de migrações entre os serviços *cloud* (Nitnaware et al., 2023).

De acordo com o *Eurostat*, em 2019, 44% dos cidadãos da União Europeia com idades entre 16 e 74 anos limitaram as suas atividades *online* devido a preocupações de segurança. Isso incluiu a recusa em fornecer informações pessoais para serviços de redes sociais ou profissionais (em 25% dos entrevistados) e o evitar efetuar compras *online* (16%) e uso das aplicações dos bancos (13%). No entanto, a maioria das pessoas ainda partilha os seus dados pessoais ao utilizar serviços *online* ou navegar na *web* (OECD, 2020). Estes dados são utilizados para publicidade *online* personalizada, permitindo experiências de comércio eletrónico envolventes e construindo empresas multibilionárias que dominam o mundo de hoje, sem que os utilizadores tenham qualquer recompensa.

O tema desta dissertação tem como premissa oferecer ao consumidor a opção de partilhar esses dados voluntariamente, em troca de uma parte do lucro que essas empresas estão a obter. Neste contexto, pretende-se desenvolver uma aplicação *mobile*, de forma que os clientes sejam recompensados pela partilha de dados, utilizando uma metodologia ágil de desenvolvimento de *software*.

Este projeto de dissertação teve origem numa proposta dos colaboradores da empresa *Accenture*, Figura 1, no departamento de inovação, designado por "*NanoLab*", um espaço de incubação e desenvolvimento de ideias. O "*NanoLab*" é um laboratório de inovação digital que utiliza tecnologias de ponta, como a inteligência artificial, a realidade aumentada e a *Internet of Things (IoT)*, para desenvolver protótipos de novas soluções e serviços num ambiente seguro e controlado (Accenture, 2023).



Figura 1: Accenture logo

A *Accenture* é uma empresa líder em consultadoria de negócios, tecnologia e transformação digital. cujo principal objetivo é aprimorar os processos empresariais, aumentando a sua eficiência e auxiliando as organizações a enfrentar os desafios de um mercado em constante evolução. A empresa foi fundada em 1989, tem sede em *Dublin* (Irlanda) e emprega mais de 730 mil funcionários distribuídos em mais de 120 países, incluindo Portugal. Tem uma forte cultura de inovação e investe em investigação e desenvolvimento para criar soluções inovadoras e tecnologicamente avançadas para os seus clientes (Sweet e Sharma, 2023).

1.2 Objetivos

O principal objetivo deste projeto incide sobre a conclusão de uma primeira versão de um *Minimum Viable Product (MVP)* - definido pela empresa, que consiste na criação de uma aplicação *mobile*. É importante realçar que este projeto apenas possuía uma ideia concebida, ainda sem estrutura definida. Portanto, parte dos objetivos incluem decisões fundamentais sobre como a aplicação vai ser estruturada, tendo em conta os recursos disponibilizados para o projeto, por parte da empresa.

Para que o *MVP* seja considerado concluído no final da dissertação, é necessário atender às seguintes tarefas:

- **Criar a arquitetura da aplicação** - esta etapa envolve o planeamento da estrutura fundamental da aplicação, definindo como é que os componentes se relacionam e interagem. Este é um passo vital para um desenvolvimento bem-sucedido da aplicação, pois estabelece bases sólidas sobre as quais todo o projeto vai ser construído;
- **Desenhar mockups dos ecrãs** - criar uma visão geral de como a *interface* vai ser projetada;
- **Desenvolver alguns ecrãs** - criar *Frontend* da aplicação;

- **Configurar o backend da aplicação** - envolve a configuração da parte lógica da aplicação;
- **Extração de dados** - efetuar a interligação com as redes sociais, focando inicialmente em apenas uma;
- **Criação de campanhas** - desenhar o processo que envolve a criação de campanhas dos retalhistas.

A realização bem-sucedida dessas tarefas é essencial para a criação de um *MVP* funcional. Desenhar a arquitetura da aplicação fornece a base sólida necessária para o desenvolvimento subsequente, enquanto a criação de *mockups* e o desenvolvimento do *frontend* permitem a visualização e interação com o produto. A configuração do *backend*, a extração de dados e a criação de campanhas garantem que a aplicação seja capaz de cumprir as funções essenciais pretendidas.

A concretização destes objetivos tem como propósito:

- Manter ou aumentar a eficiência do planeamento da equipa, priorizando tarefas;
- Aumentar a motivação.

Por decisão da *Accenture*, inicialmente foi proposta a um conjunto de três alunos, no contexto de projetos de estágio, a realização de um projeto que consistia na criação de uma aplicação *mobile*, designada por “*Take My Data*”. O “*Take My Data*” é uma aplicação *mobile* projetada para gerir a recolha de dados, para empresas de retalho. A elaboração e desenvolvimento deste projeto pressupõe a análise, *design*, testagem e implementação da aplicação. Assim, após a criação da aplicação *mobile* “*Take My Data*”, os dados recolhidos vão permitir que as empresas recompensem os seus clientes, através de um sistema de *Token*, enquanto obtêm informações valiosas para estratégias de marketing. Faz-se notar, que, posteriormente, foi dada à autora desta dissertação, a oportunidade de continuar o desenvolvimento do projeto de forma individual.

Devido à complexidade deste projeto, é crucial definir os papéis da equipa envolvida e estabelecer métricas de trabalho específicas. Para garantir a gestão eficaz do projeto, bem como, para lidar com mudanças inesperadas nos requisitos e na equipa, foi escolhida a metodologia ágil *Scrum* para orientar a execução do projeto. Espera-se que a adoção desta metodologia resolva problemas relacionados à falta de planeamento, comunicação, sincronização e especificações do produto.

No entanto, embora o *Scrum* seja uma metodologia ágil eficaz, pode haver desafios associados à sua implementação. A comunicação é essencial no projeto *Scrum*, e a falta dela pode ser um grande obstáculo. A equipa precisa de trabalhar de uma forma colaborativa e manter um diálogo constante para garantir que todos estejam alinhados com os objetivos do projeto.

Além disso, o *Scrum* depende fortemente da capacidade da equipa de se auto-organizar e gerir seu próprio trabalho. Isso pode ser um desafio, especialmente se a equipa não possuir experiência com o método, como se verifica. É importante garantir que todos os membros da equipa entendam os seus papéis e responsabilidades e sejam capazes de trabalhar de forma autónoma para garantir que o projeto seja entregue no prazo estipulado. Outro possível desafio que pode surgir durante o projeto é a falta de recursos, especialmente se a equipa tiver outras responsabilidades além desse projeto. Por isso, é fundamental para o sucesso do projeto, garantir que todos os recursos necessários estejam disponíveis e que a equipa tenha tempo suficiente para trabalhar no projeto. Por fim, é importante lembrar que, como este projeto está a ser desenvolvido internamente na *Accenture*, pode haver restrições ou diretrizes específicas que precisam de ser seguidas.

Em suma, a implementação do *Scrum*, como metodologia ágil, pode ajudar a equipa a gerir melhor o projeto “*Take My Data*”, mas há desafios que precisam ser superados. A comunicação constante, a auto-organização da equipa, a disponibilidade de recursos e o cumprimento das políticas internas da empresa são alguns dos desafios que precisam ser abordados para garantir o sucesso do projeto.

1.3 Metodologia de trabalho

Este projeto, como abordagem sistemática para o desenvolvimento, usa métodos de definição e priorização de ideias. Uma das ferramentas cruciais incorporadas neste processo é o “Diagrama *Bull's-Eye*”, que foi elaborado pelo *Product Owner (PO)*. O objetivo principal é identificar e destacar as funcionalidades que a aplicação deve abordar, com base nas suas prioridades. O “Diagrama *Bull's-Eye*”, apresentado na Figura 2, é uma representação gráfica das funcionalidades planeadas para a aplicação. As funcionalidades são dispostas em anéis concêntricos, estando as mais prioritárias no centro e as menos prioritárias nos anéis exteriores.

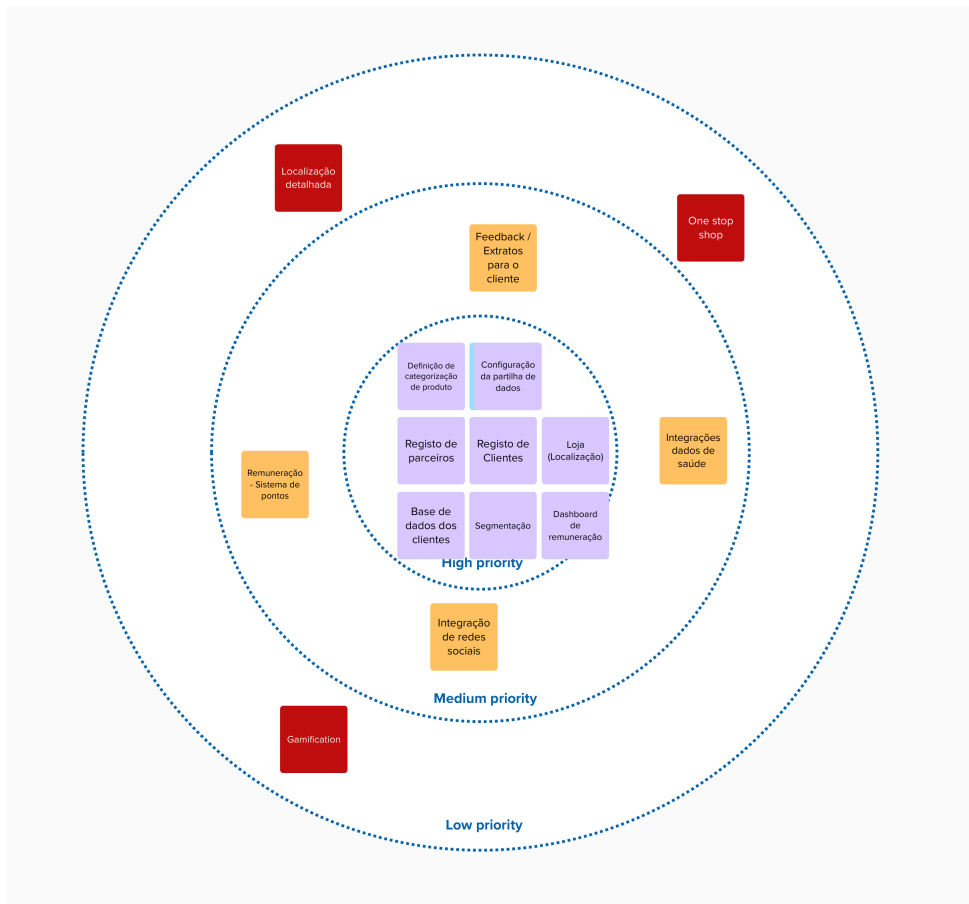


Figura 2: Diagrama *Bull's-Eye* do "Take My Data"

Este diagrama desempenha um papel fundamental, como guia, para a equipa de desenvolvimento, delineando as principais atividades e o foco que a solução dever abordar. Combinado com a realização de reuniões, o diagrama contribuiu para uma visão mais clara da solução desejada pelos *POs*.

Foram definidas como prioritárias as funcionalidades, apresentadas na Figura 3, que serviram como ponto de partida.



Figura 3: Detalhe das funcionalidades com maior prioridade

Após este exercício, e já com análise das *journeys*, mais uma vez, realizada pelos *PO*, começou-se a dividir o fluxo de trabalho por *Journey steps*. Essa abordagem permitiu dividir a jornada em pequenas partes, facilitando assim o desenvolvimento do projeto e a criação do *Backlog*.

Como primeiro passo, foi definido o registo do utilizador e a aceitação dos termos e condições da aplicação, como ilustrado na Figura 4. Este é um dos passos mais críticos no desenvolvimento da aplicação, pois o utilizador só pode utilizar a aplicação após a aceitação dos termos e condições relacionados com a partilha de dados pessoais. Além disso, o registo do utilizador é igualmente crucial, uma vez que permite que os seus dados sejam armazenados para o uso da aplicação.

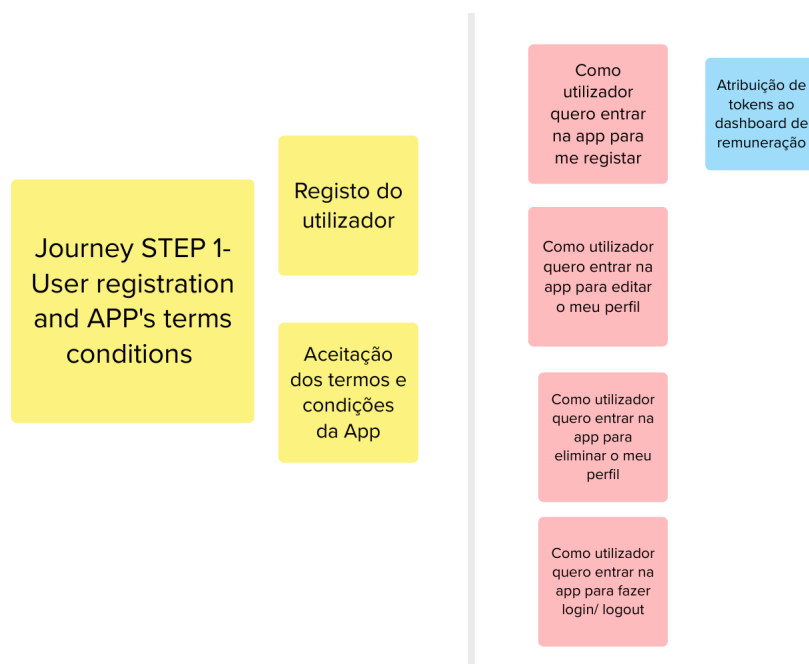


Figura 4: Journey step 1

Conforme demonstrado na Figura 5, o segundo passo na jornada implica a obtenção de permissão para compartilhar os dados recolhidos através das redes sociais e da localização. Sem esse passo, a aplicação perde a sua funcionalidade principal, tornando, portanto, crucial que os utilizadores possam autorizar a partilha das suas informações pessoais, seja concedendo acesso aos dados das redes sociais ou mesmo à sua localização.

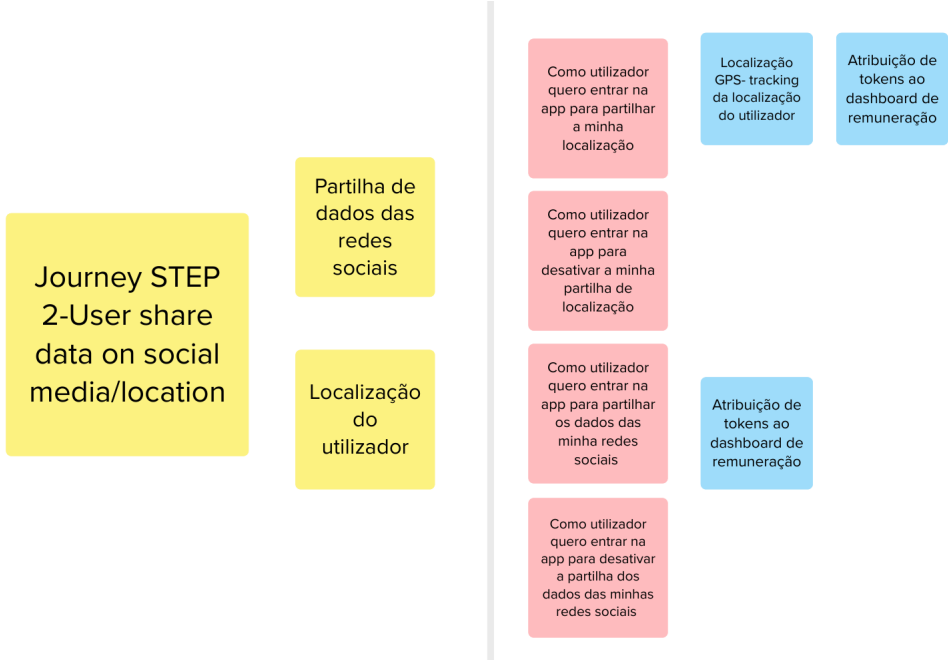


Figura 5: Journey step 2

Posteriormente, após a partilha dos dados pessoais, é expectável a extração desses dados das redes sociais, possibilitando a segmentação dos perfis dos utilizadores. Isso leva-nos ao terceiro passo na jornada, que engloba não apenas a segmentação, mas também a combinação de campanhas. Estas campanhas são criadas pelas empresas através de um *Backoffice* e são direcionadas aos perfis dos utilizadores que partilham informações correspondentes à campanha, conforme ilustrado na Figura 6.

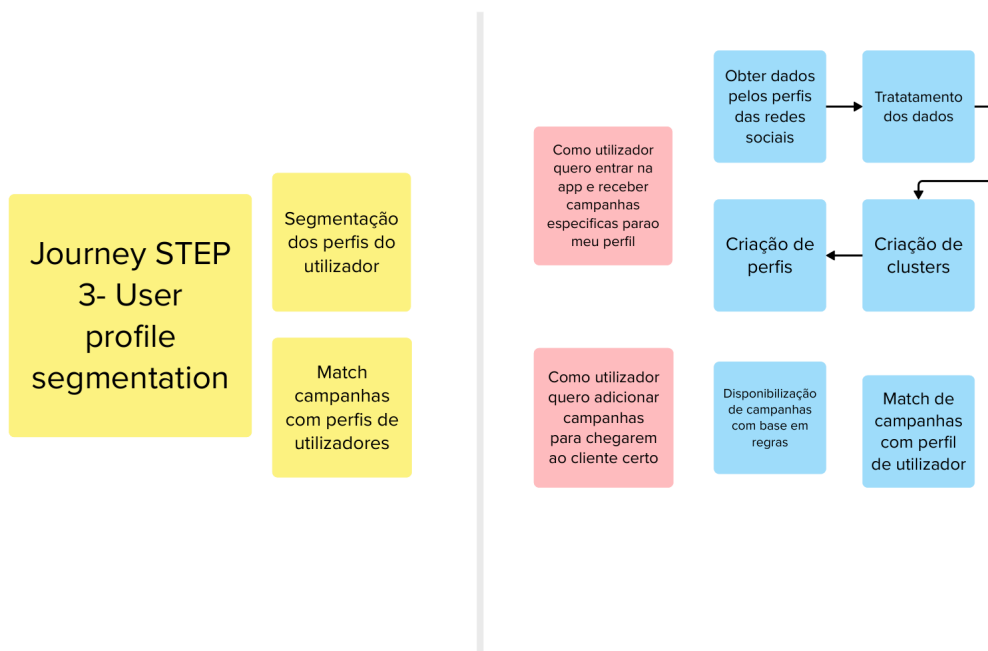


Figura 6: Journey step 3

O quarto passo na jornada consiste na implementação do sistema de notificações para os utilizadores, sendo esta a principal forma de interação entre o utilizador e a aplicação. Este passo envolve o desenvolvimento de um sistema de notificações que funciona tanto através de email como diretamente dentro da aplicação, como representado na Figura 7. Estas notificações têm a finalidade de informar os utilizadores sobre novas campanhas já segmentadas. Além disso, podem ser utilizadas para comunicar promoções ou campanhas em lojas nas proximidades, caso o utilizador opte por partilhar a sua localização.

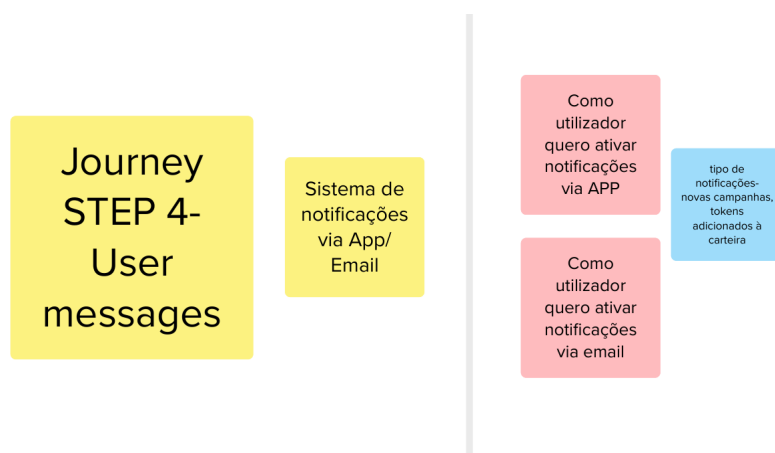


Figura 7: Journey step 4

No quinto passo da jornada, é elaborada algo fundamental e diferenciador para a aplicação: o sistema de recompensas. Este sistema vai oferecer, aos utilizadores que concordem em partilhar os seus dados, recompensas pela sua colaboração. Além disso, como podemos ver na Figura 8, os utilizadores também podem ser recompensados ao receberem notificações de campanhas do seu interesse, bem como ao adquirirem produtos ou participarem em campanhas que ofereçam recompensas. As empresas vão ter a liberdade de escolher e aplicar as recompensas que considerem adequadas aos seus produtos ou campanhas. Quanto mais atrativa for a oferta, maior será a probabilidade de conversão. Desta forma, pretende-se criar um ambiente justo e atrativo, tanto para os utilizadores, como para as empresas, no que diz respeito à partilha de dados pessoais.

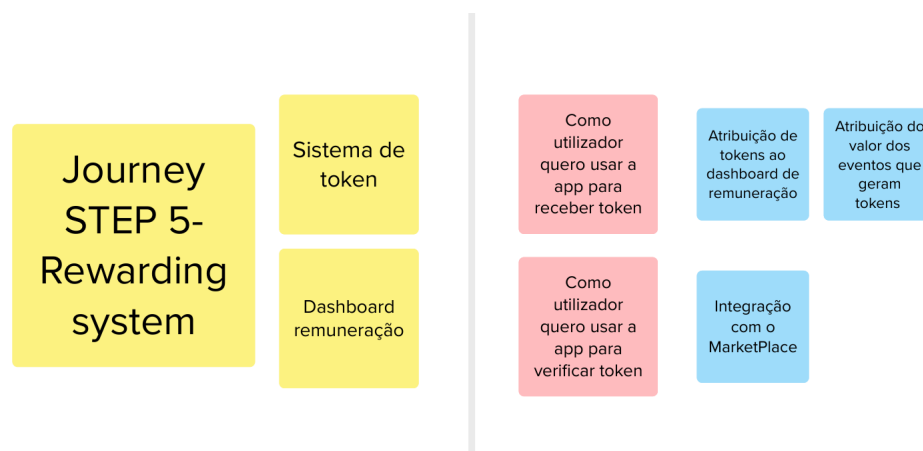


Figura 8: Journey step 5

Para que as empresas possam participar na plataforma, o sexto passo da jornada é crucial. Neste passo, desenvolve-se um *Backoffice web*, que permite que as empresas criem os seus perfis e integrem a aplicação. Isso possibilita o registo, a edição e a eliminação de perfis empresariais, como ilustrado na Figura 9.

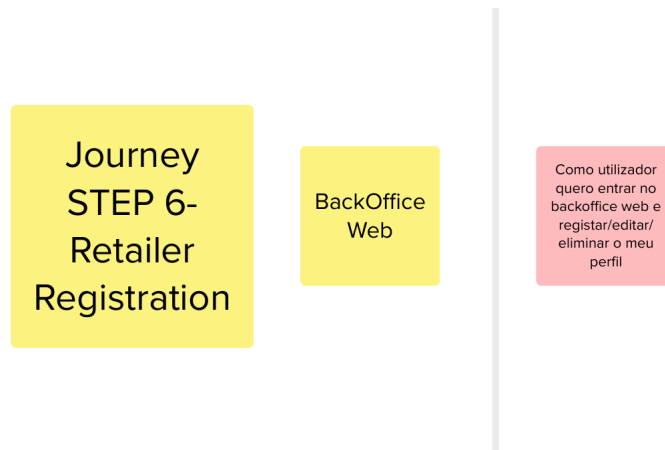


Figura 9: Journey step 6

Como sétimo passo da jornada, as empresas vão ter a capacidade de criar as suas próprias campanhas. Ou seja, têm a possibilidade de definir o público-alvo, com base na segmentação dos dados pessoais dos utilizadores realizada anteriormente. Além disso, este passo vai permitir que as empresas façam a gestão das compras dos utilizadores da aplicação, proporcionando uma análise mais detalhada do desempenho das campanhas, através de um *Dashboard*, como é evidenciado na Figura 10.

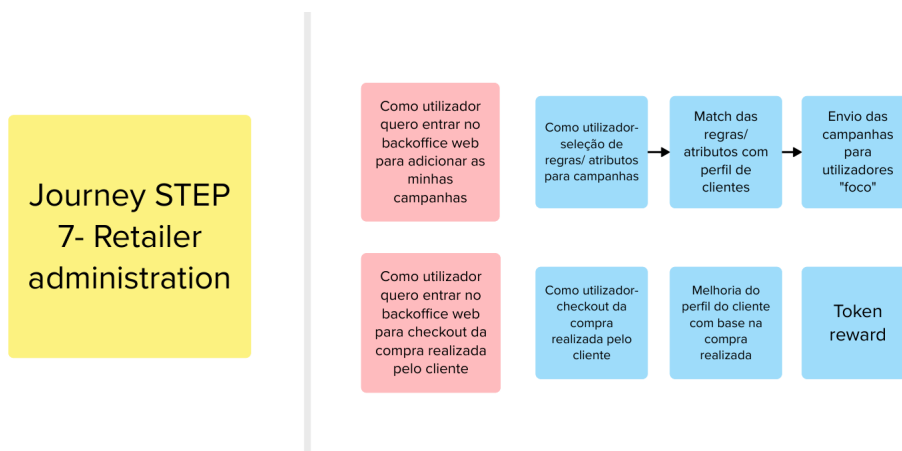


Figura 10: Journey step 7

A Figura 11 ilustra o oitavo passo da jornada, onde vai ser desenvolvido um sistema de *QR code* que permitirá aos utilizadores utilizar as campanhas em lojas físicas.

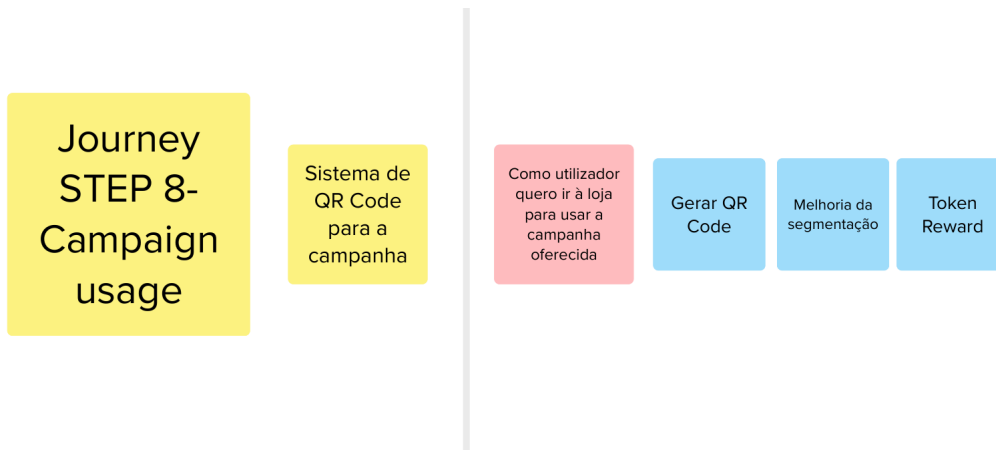


Figura 11: Journey step 8

Por fim, na Figura 12, encontra-se o nono passo da jornada, que envolve a criação de um sistema para avaliar a experiência do utilizador, nas lojas onde resgataram as campanhas. Este processo também vai oferecer recompensas aos utilizadores, incentivando-os a avaliar as suas experiências com os serviços da aplicação.

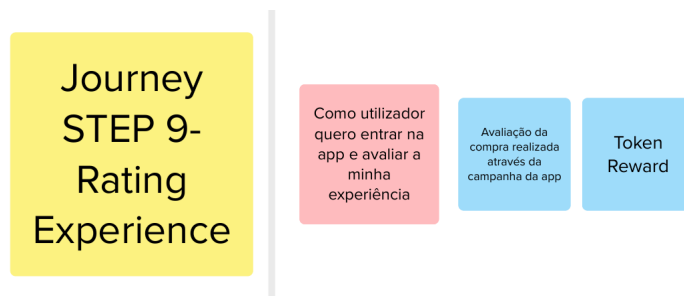


Figura 12: Journey step 9

1.4 Estrutura de relatório

A presente dissertação é constituída por sete capítulos. Neste primeiro capítulo é efetuado o enquadramento sobre o tema do projeto de dissertação, fornecendo uma compreensão inicial das questões e conceitos-chave abordados. A introdução aborda a importância do projeto, destacando o seu contexto, objetivos e relevância no cenário atual.

O segundo capítulo aprofunda os tópicos mencionados no enquadramento, criando uma revisão bibliográfica detalhada. Esta secção serve como base teórica para a pesquisa, explorando conceitos essenciais relacionados a processos de desenvolvimento de *software* e requisitos de *software*.

O terceiro capítulo, apresenta as tecnologias e *frameworks* utilizadas para a gestão de dados em empresas de retalho. Este capítulo oferece uma compreensão abrangente das ferramentas e abordagens tecnológicas relevantes para o projeto.

O quarto capítulo, descreve a conceção e a arquitetura do projeto. Abrange o reconhecimento do problema e os intervenientes envolvidos. Inclui a arquitetura de alto nível, com as suas diferentes camadas (*Frontend, Backend e API*), a plataforma tecnológica escolhida e a modelo de dados adotada.

O quinto capítulo, apresenta a análise e o design detalhados da aplicação *mobile*, incluindo a *interface* do utilizador, a estrutura de funcionamento e a lógica geral de implementação.

No Capítulo 6, é detalhada a implementação da metodologia *Scrum* no projeto. É explicado como essa abordagem ágil foi adotada e como os principais eventos do *Scrum* foram aplicados, para garantir o progresso eficiente do projeto.

No último capítulo, apresentam-se as conclusões finais da dissertação, resumindo as principais descobertas e destacando seu impacto e significado. Este capítulo encerra a dissertação, proporcionando uma visão abrangente do trabalho realizado.

2 PROCESSOS E REQUISITOS DE SOFTWARE

2.1 Introdução

Para compreender conceitos essenciais e relevantes para a realização desta dissertação, foi necessário fazer uma revisão e levantamento de conceitos relativamente a processos de desenvolvimento de *software* e também sobre requisitos de *software*. Este capítulo serve de apoio à elaboração deste projeto.

É necessário ressaltar a importância de uma revisão bibliográfica para fundamentar o projeto realizado. O objetivo principal é fornecer uma base teórica, possibilitando a identificação de metodologias presentes no mercado.

A revisão bibliográfica permite, também, a identificação de metodologias e abordagens utilizadas em estudos anteriores. Embora a metodologia já tenha sido escolhida pela empresa em questão, foi necessário revisitar essas abordagens para fornecer maior contextualização ao trabalho.

2.2 Processos de desenvolvimento de software

O desenvolvimento de *software* tornou-se uma atividade de suma importância para as empresas atualmente, devido ao seu impacto no desenvolvimento e qualidade do produto. O IEEE (1990) definiu a engenharia de *software* como a aplicação de uma abordagem sistemática, disciplinada e quantificável ao desenvolvimento, operação e manutenção de *software*. O desenvolvimento do *software* deve ser feito por fases, com a participação contínua dos utilizadores, e com a programação da conceção do custo em cada fase. A metodologia de desenvolvimento de *software* também inclui valores fundamentais que são defendidos pela equipa do projeto e pelas ferramentas utilizadas no planeamento, desenvolvimento e, no processo de implementação (Jhanjhi et al., 2019). Posteriormente são apresentadas diferentes classificações nas metodologias de desenvolvimento de *software*.

2.2.1 Gestão de projetos

Para a definição do conceito de gestão de projetos, é necessário primeiro definir o que é um projeto. De acordo com European Commission (2016), este pode ser descrito como uma estrutura organizacional temporária que visa criar um produto ou serviço único, tendo em conta determinados limites, como tempo, custo e qualidade. A sua característica principal é a temporalidade, o que significa que ele deve possuir um fim e início claramente definidos. Além disso, o seu *output* deve ser singular, ou seja, o produto ou serviço desenvolvido não deve existir previamente na sua forma atual, embora possa ter semelhanças

com outros produtos ou serviços. Na Figura 13, apresenta-se um resumo das características chave de um projeto.

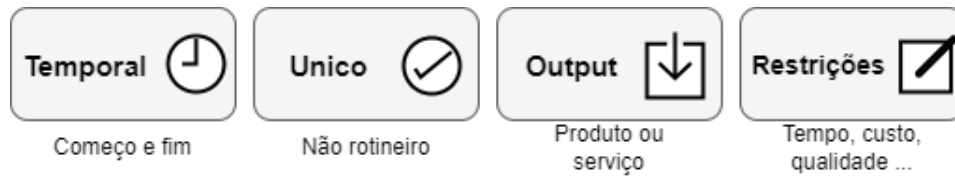


Figura 13: Características chave de um projeto (adaptado de PM² (European Commission, 2016))

A gestão de projetos pode ser descrita, como as atividades de planeamento, organização, segurança, controlo e gestão dos recursos, para atingir as metas e os objetivos específicos do projeto, de uma forma eficaz e eficiente.

A abordagem de gestão de projetos utilizada deve ser sempre adaptada às necessidades do projeto. Seymour e Hussein (2014) diz-nos que a gestão de projetos na sua essência, preocupa-se com a criação de um ambiente em que as pessoas possam trabalhar em conjunto, para atingir um objetivo mútuo, a fim de realizar projetos bem sucedidos dentro do prazo e orçamento. Ou seja, segundo Project Management Institute (2021) a gestão do projeto consiste em orientar o trabalho do projeto para obter os resultados pretendidos e para isso as equipas do projeto podem alcançar os resultados, utilizando uma vasta gama de abordagens (por exemplo, preditiva, híbrida e adaptativa). A entrega de resultados de negócios de uma organização é realizada através do sucesso de projetos, portanto, a gestão de projetos é a estratégia e o processo através dos quais as organizações atingem os seus objetivos e o sucesso (Salameh, 2014).

Os resultados de um projeto podem variar de muitas formas, podendo ser um produto, como uma nova aplicação de *software* ou um serviço. A execução de um projeto deve ocorrer sob certas restrições, que podem ser impostas externamente, como prazos, orçamentos e padrões de qualidade, ou autoimpostas, baseadas nas necessidades e recursos da organização. Essas restrições podem incluir considerações ambientais, capacidades organizacionais e disponibilidade de recursos (European Commission, 2016).

A metodologia *Project Management Methodology (PM²)* desenvolvida pela Comissão Europeia é uma abordagem que pode ser usada para executar os processos de desenvolvimento de *software* numa organização. Esta metodologia concentra-se na gestão eficaz de projetos e na entrega de soluções e benefícios. Segundo a European Commission (2016) a eficácia da gestão de projetos pode ser melhorada através de comunicação e disseminação de informações, clarificando as expectativas o mais cedo possível no ciclo de vida do projeto e definindo o ciclo de vida do projeto (desde o início até ao

encerramento). Para isso é necessário fornecer orientações para o planeamento do projeto, introduzir atividades de monitorização e controlo necessárias para gerir um projeto, propor atividades de gestão e resultados (planos, reuniões, decisões) e fornecer uma ligação às práticas ágeis (por exemplo, *Agile PM*).

2.2.2 Processos tradicionais

Um dos modelos mais comuns associado à metodologia de gestão de projetos tradicional é o modelo em cascata. Segundo Sommerville (2016), ao lidar com a engenharia de sistemas complexos, torna-se sensato integrar as melhores características de diversos processos gerais. Isso se justifica pela dificuldade de ter acesso a todas as informações necessárias sobre os requisitos fundamentais do projeto desde o início. O modelo em cascata é frequentemente empregado para garantir uma compreensão abrangente dos requisitos, apresentando o desenvolvimento do projeto como uma série de fases, conforme ilustrado na Figura 14.

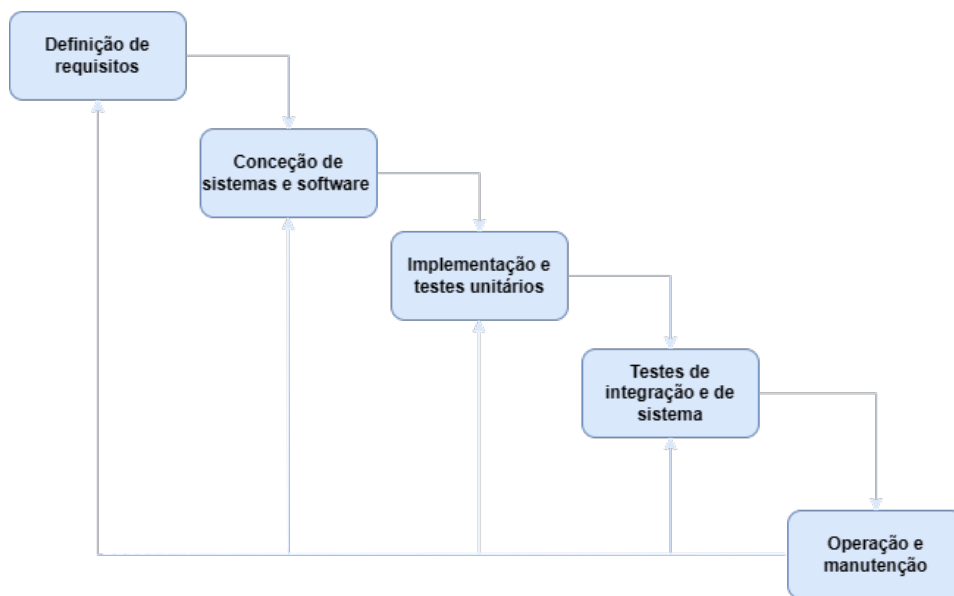


Figura 14: Modelo em cascata (adaptado de Sommerville (2016))

Devido à passagem em cascata de uma fase para outra, este modelo é conhecido como modelo em cascata ou ciclo de vida do *software*. O modelo em cascata é um exemplo de um processo orientado para o planeamento (Sommerville, 2016). Antes de se iniciar o processo de desenvolvimento, devem ser planeadas todas as atividades do processo. Cada fase apresentada na Figura 14 deve definir exatamente o que se pretende alcançar e por vezes é necessário incluir a descrição da sua conceção. Além disso, é fundamental a criação de documentos que descrevam a realização dos resultados obtidos, normalmente

estes são aprovados por um *Project Management Office (PMO)*. Normalmente, cada fase começa aquando da finalização da fase anterior, de forma que os resultados sirvam como métrica para a fase seguinte.

Na primeira fase, é feita a análise e definição dos requisitos. Nesta fase, é necessário estabelecer restrições, objetivos e serviços dos sistemas, através de uma consulta ao utilizador. Na conceção do sistema e *software* é onde se realiza a análise de requisitos e estabelece-se a arquitetura global do sistema.

Depois, segue-se a implementação dos testes unitários. Durante esta fase, a conceção do *software* é concretizada como um conjunto de programas ou unidades de programa. Os testes unitários envolvem a verificação de que cada unidade cumpre a sua especificação (Sommerville, 2016). Na integração e testagem do sistema são realizados testes, para garantir que os requisitos do *software* foram cumpridos. Por fim, existe a operação e manutenção do sistema. Sommerville (2016) caracteriza esta fase como a mais longa do modelo, visto que é realizada a instalação e colocado em prática todas as fases realizadas anteriormente. A manutenção envolve a correção de erros, observados com a instalação, e que não foi possível serem identificados antes. Além disso, é feita uma melhoria e reforço do sistema.

O processo de *software*, na prática, nunca é um modelo linear simples, mas envolve uma crescente comunicação de uma fase para outra. À medida que surgem novas informações numa fase do processo, os documentos produzidos nas fases anteriores devem ser modificados para refletir as alterações necessárias ao sistema. Por exemplo, se se descobrir que um requisito é demasiado caro para ser implementado, o documento de requisitos deve ser alterado para remover esse requisito. Como resultado, tanto os clientes como os programadores podem congelar prematuramente a especificação do *software*, para que não sejam feitas mais alterações. Infelizmente, isto significa que os problemas são deixados para uma resolução posterior, ignorados ou programados (Sommerville, 2016). Segundo o mesmo autor, o modelo em cascata não é um modelo adequado para projetos onde os requisitos estão constantemente a ser alterados.

2.2.3 Processos ágeis

O desenvolvimento ágil de *software* representa uma nova abordagem para o planeamento e gestão de projetos de *software*. Dá menos ênfase aos planos iniciais e ao controlo rigoroso e baseia-se mais na colaboração informal, na coordenação e na aprendizagem (Dybå et al., 2014). As metodologias ágeis seguem um estilo de desenvolvimento iterativo e incremental que se ajusta dinamicamente à evolução dos requisitos e permite uma melhor gestão dos riscos Hoda et al. (2008).

De acordo com o Alliance (2001), existem quatro princípios fundamentais para o desenvolvimento do *software* através das metodologias ágeis. Estes princípios valorizam indivíduos e interações em vez de processos e ferramentas; *software* funcional em vez de documentação exaustiva; a colaboração com o cliente em vez da negociação de contratos; e responder à mudança em vez de seguir um plano.

As metodologias ágeis são orientadas para a satisfação do cliente e permitem modificações por meio de um estilo iterativo de desenvolvimento, no qual apenas as funcionalidades necessárias são implementadas em cada iteração. Isso facilita a adaptação das funcionalidades conforme necessário e uma gestão mais eficaz dos riscos associados Hoda et al. (2008).

O *Scrum* é uma *Framework* iterativa e incremental, baseada em transparência, inspeção e adaptação, e é projetada para ser flexível o suficiente, para lidar com mudanças durante o processo de desenvolvimento. Faz parte das metodologias ágeis para o apoio da gestão de projetos de *software*. É amplamente implementado por empresas de diversas áreas, devido à inovação que proporciona no mercado, permitindo combinar rapidez no desenvolvimento dos processos com a garantia de qualidade necessária.

De acordo com Morandini et al. (2021), o *Scrum* é uma das metodologias mais utilizadas atualmente em projetos de *software*. Criado em meados de 1990 por *Ken Schwaber* e *Jeff Sutherland*, este foi desenvolvido a partir da combinação de métodos *Lean*, resultando numa *framework* mais simples e leve. O ciclo de vida da metodologia *Scrum* pode ser observada na Figura 15.

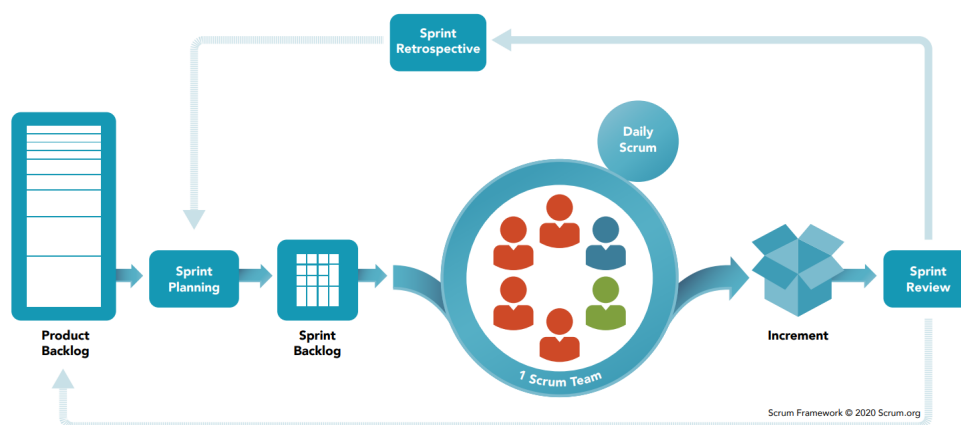


Figura 15: *Scrum* Life cycle. Fonte: Scrum.org (2021)

De acordo com o Scrum.org (2023), de forma global, o *Scrum* consiste em eventos, artefactos, regras e

numa equipa com diferentes papéis que trabalham juntos para fornecer uma estrutura eficaz. A equipa *Scrum* participa em quatro eventos e produz três artefactos. O Guia do *Scrum*, escrito pelos cocriadores e mantido por eles, descreve de forma clara e sucinta o *framework* do *Scrum*, definindo as suas responsabilidades, eventos, artefactos e as orientações que os unem.

A equipa *Scrum* é composta pelo *Product Owner (PO)*, *Scrum Master (SM)* e a equipa de desenvolvimento, que trabalham em conjunto durante as cerimónias de acordo com as necessidades do cliente. Os eventos do *Scrum* incluem a Reunião de Planeamento do *Sprint*, a Reunião Diária, a Revisão do *Sprint* e a Retrospectiva do *Sprint*. Estes eventos fornecem uma estrutura para o trabalho diário da equipa, permitindo que eles se comuniquem de forma eficaz, identifiquem obstáculos e planeiem o trabalho futuro. Os artefactos do *Scrum* incluem o *Product Backlog*, que é uma lista ordenada de itens a serem entregues, o *Sprint Backlog*, que é uma lista de itens selecionados para o *Sprint* atual, e o Incremento, que é o resultado do trabalho da equipa durante o *Sprint* atual. Em resumo, o *Scrum* é uma metodologia ágil popular e eficaz, para o desenvolvimento de produtos de *software* complexos, que oferece uma estrutura flexível para lidar com mudanças durante o processo de desenvolvimento.

De forma a compreender a *framework* é importante especificar as diferentes etapas do *Scrum*, começando pela definição do *Sprint* e a sua importância. O *Sprint* é uma estrutura fundamental na dinâmica do *Scrum*, onde a equipa adapta o produto em direção ao *Product Goal* e ao *Sprint Goal*, que é estabelecido no início do ciclo de trabalho. Neste caso, o *Product Goal* descreve um estado futuro do produto em desenvolvimento, que faz parte do *Product Backlog*, que descreve o objetivo a longo prazo. O *sprint goal* reflete métricas a curto prazo e são as tarefas que vão ser realizadas no *Sprint*. Embora seja um compromisso dos *Developers*, ele proporciona flexibilidade em termos do trabalho exato necessário para alcançá-lo. O *Sprint Goal* também cria coerência e foco, encorajando o *Scrum Team* a trabalhar em conjunto em vez de iniciativas separadas.

Através da utilização do *Scrum*, é possível aprimorar a previsibilidade e o controle de riscos, acompanhando as etapas do projeto e promovendo a inspeção e adaptação contínua. Esta *framework* tem como estrutura de desenvolvimento, ciclos de trabalho e os chamados “*Sprints*”. O *Sprint* organiza todos os eventos realizados. Cada evento no *Scrum* é uma oportunidade de inspecionar e adaptar os diferentes artefactos ao projeto.

No início de cada *Sprint*, é imperativo que se proceda à seleção dos itens prioritários, que espelham os requisitos fornecidos pelo cliente. A equipa deve comprometer-se com a sua realização até ao final do

Sprint, tornando o planeamento uma etapa crucial para garantir que tal se concretize. Para que o *Sprint* seja bem-sucedido, é igualmente necessário que a equipa tenha em consideração o conceito de "done", que deve ser implementado e ajustado às necessidades específicas do projeto. Isto implica, em casos específicos, a integração, a realização de testes e obtenção de aprovações.

Como foi mencionado anteriormente, o *Scrum* baseia-se no conceito de "inspecionar e adaptar", que enfatiza a tomada de um passo curto no desenvolvimento, inspecionando tanto o produto resultante quanto a eficácia das práticas atuais e, em seguida, adapta os objetivos do produto e as práticas de processo. Este processo é repetido continuamente em todos os *Sprints* realizados ao longo do projeto.

2.3 Requisitos de software

Pode-se definir que a *Area of Knowledge (KA)* de requisitos de *software* diz respeito à obtenção, análise, especificação e validação de requisitos de *software*, bem como a gestão dos requisitos durante todo o ciclo de vida do produto de *software*. Os requisitos de *software* exprimem as necessidades e restrições impostas a um produto de *software*, que contribuem para a solução de algum problema do mundo real (IEEE Computer Society, 2014).

No contexto da engenharia de *software*, os requisitos de *software* desempenham um papel crucial. De acordo com o *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide)*, um requisito de *software* é uma propriedade essencial que um sistema de *software* deve possuir para resolver um problema do mundo real. Esses problemas podem variar em complexidade, desde a automação de tarefas simples até o suporte a processos de negócios complexos, correção de deficiências em sistemas existentes ou o controle de dispositivos especializados (IEEE Computer Society, 2014). A complexidade dos requisitos de *software* reflete a natureza dos problemas que o *software* visa resolver. Em muitos casos, os requisitos são uma combinação complexa de necessidades e expectativas de diversas partes envolvidas, que podem estar em diferentes níveis na empresa. Eles podem estar envolvidos direta ou indiretamente com a funcionalidade específica que o *software* vai proporcionar no seu ambiente de operação. Em resumo, os requisitos de *software* podem ser considerados como as características essenciais que definem o que o *software* deve realizar, e representam uma peça fundamental na engenharia de *software*. Compreender, documentar e gerir efetivamente esses requisitos é essencial para o sucesso de qualquer projeto de desenvolvimento de *software*.

Outra característica essencial de todos os requisitos de *software* é a capacidade de serem verificáveis, quer

estejamos a falar de requisitos funcionais, que descrevem o que o *software* deve fazer, ou de requisitos não funcionais, que estipulam restrições sobre como o *software* deve operar. Garantir que os requisitos possam ser verificados eficazmente é uma preocupação crítica para a equipa de desenvolvimento de *software*, a equipa de teste e a equipa de garantia de qualidade. Estes devem trabalhar em conjunto, para assegurar que a verificação seja possível dentro das limitações de recursos disponíveis.

Os requisitos de *software* também possuem outros atributos além de suas propriedades comportamentais. Ou seja, incluem atributos como uma classificação de prioridade, que ajuda a determinar a importância relativa de cada requisito quando há recursos limitados, e um valor de estado, que permite monitorizar o progresso do projeto. A identificação única dos requisitos de *software* é uma prática comum, pois facilita a gestão da configuração de *software* ao longo de todo o ciclo de vida do desenvolvimento e manutenção do *software*.

Relativamente aos requisitos funcionais e não funcionais o *SWEBOK Guide* apresenta as definições seguintes: Os requisitos funcionais devem descrever as funções que o *software* executa, como a formatação de um texto ou a modulação de um sinal. Também podem ser referidos como capacidades ou características do *software*. Um requisito funcional pode ser identificado como tal, quando um conjunto finito de passos de teste pode ser desenvolvido para validar o seu comportamento (IEEE Computer Society, 2014). Por outro lado, Nicolás e Toval (2009) já os define como unidades funcionais da aplicação, os limites de cada funcionalidade e as limitações do impacto nas operações individuais e do sistema.

Em relação aos requisitos não funcionais, estes devem atuar como restrições ou metas de qualidade que moldam a solução final. Eles abrangem diversas áreas como desempenho, manutenção, segurança, entre outros. De acordo com Chazette e Schneider (2020), os requisitos não funcionais são essenciais para orientar a implementação de todos os possíveis serviços em um ou vários sistemas. Eles fornecem diretrizes explícitas que permitem quantificar os elementos que destacam o serviço, garantindo a continuidade das funcionalidades desenvolvidas diante dos desafios encontrados.

2.4 Conclusão

Concluindo, neste capítulo foram explorados dois tópicos principais para o sucesso desta dissertação: os processos de desenvolvimento de *software* e os requisitos de *software*. Relativamente ao processo de desenvolvimento de *software* foram apresentados algumas definições no âmbito da engenharia de

software e como esta se aplica. Foram também exploradas as diferentes metodologias, desde as tradicionais, com foco no modelo em cascata, até às ágeis, com destaque para a *framework Scrum*.

Em seguida, foi realizada uma pesquisa dos Requisitos de *Software*, que são essenciais para definir o que um *software* deve fazer e como ele se deve comportar. Foi discutida a importância dos requisitos serem verificáveis e como isso afeta o processo de desenvolvimento e teste de *software*. Também foram abordados outros atributos dos requisitos, como prioridade e estado, que desempenham um papel fundamental na gestão de projetos de *software*.

Finalmente, foram exploradas as diferenças entre requisitos funcionais e não funcionais, destacando como eles contribuem para a definição da solução de *software*. Foi também compreendida a importância deste tipo de requisitos na implementação de serviços eficazes e na garantia da continuidade das funcionalidades desenvolvidas.

No geral, este capítulo fornece um alicerce sólido de conhecimento sobre os processos de desenvolvimento de *software* e requisitos de *software*. Estes conceitos serão fundamentais para a dissertação, pois permitirão abordar o desenvolvimento de uma solução de *software* com compreensão e rigor.

3 TECNOLOGIAS E FRAMEWORKS UTILIZADAS NA GESTÃO DE DADOS PARA EMPRESAS DE RETALHO

3.1 Introdução

Atualmente, há uma variedade de *softwares* disponíveis que permitem o desenvolvimento de aplicações *mobiles*. Portanto, é de suma importância selecionar as tecnologias mais adequadas.

No contexto do avanço deste projeto, a empresa não estabeleceu nenhuma obrigação específica em relação às *frameworks* e tecnologias a serem utilizadas. No entanto, foi conduzida uma pesquisa sólida que resultou em várias sugestões, visando facilitar o desenvolvimento. As sugestões levaram em consideração não apenas a falta de experiência da equipa e a dimensão, mas também os requisitos do projeto.

Seguidamente, são abordadas em detalhe todas as ferramentas de *software* utilizadas, juntamente com a justificação da sua escolha e a necessidade que cada uma delas atende no desenvolvimento do projeto. Esta análise aprofundada vai proporcionar uma compreensão mais abrangente das tecnologias selecionadas e da sua relevância para o sucesso do projeto.

3.2 Tecnologias para Frontend

Em 2015, foi lançada pelo *Facebook* uma plataforma em *JavaScript* construída com base na *framework React*, denominada de **React Native**. De acordo com Occhino (2020), esta plataforma foi originalmente criada para o desenvolvimento de páginas *web*, e permite o desenvolvimento de aplicações para os sistemas operativos *iOS* e *Android*. Atualmente são testadas com o auxílio de simuladores de *software* de terceiros, como o *Expo*, tanto no ambiente de desenvolvimento quanto nas plataformas *iOS* e *Android* (Gülcüoğlu et al., 2021). A estrutura do *React Native* está representado na Figura 16, confirma o mencionado anteriormente, ou seja *React Native* é uma *framework* com fundamentada em *javascript*, que permite o desenvolvimento de aplicações para *iOS* e *Android*, mas também permite, com o auxílio de simuladores, criar um ambiente de desenvolvimento do qual não é necessário especificar qual o sistema operativo que se pretende. Esta *framework*, de código aberto de rápido desenvolvimento, desde o seu lançamento ganhou bastante destaque, sendo incluída na lista de favoritos por mais de 30.000 programadores no GitHub (Team, 2020).

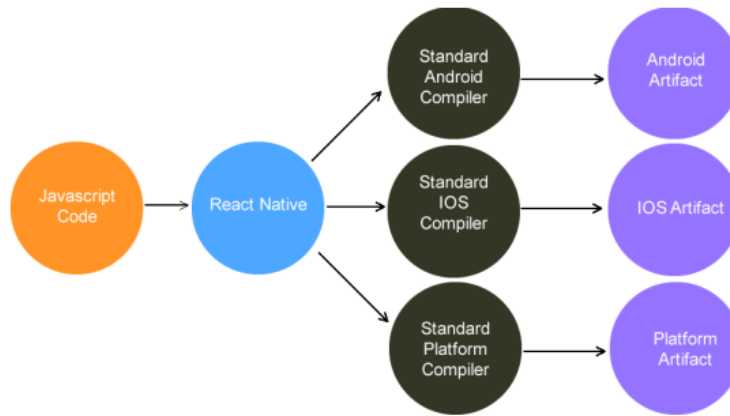


Figura 16: Estrutura do React Native. Fonte: Frachet (2020)

Além desta *framework*, identifica-se uma outra que se encontra em grande crescimento, *Flutter*, criada pela *Google*, em 2015, que também seria uma excelente ferramenta para o desenvolvimento desta aplicação. É outra *framework* multiplataforma que visa o desenvolvimento para aplicações móveis, que funciona tanto em *Android* como em *iOS*. Além disso, o *Flutter* foi escolhido como a plataforma para aplicações no *Fuchsia*, que é o sistema operacional da próxima geração da *Google* (Wu, 2018).

No caso do *Flutter*, todas as aplicações são escritas em *Dart*, uma linguagem de programação também desenvolvida pela *Google*. Atualmente a *Google* utiliza esta linguagem internamente e já foi testada em aplicações *web* como o *AdWords*. O *Dart* foi desenvolvido como uma alternativa ao *JavaScript*, e por isso, incorpora algumas características importantes próximas do padrão do *JavaScript*. No entanto para atrair mais programadores que não estão familiarizados com o *JavaScript*, o *Dart* adota uma sintaxe semelhante à do *Java* (Wu, 2018).

Na Figura 17, visualiza-se os resultados de uma sondagem realizada pelo *StackOverflow*, que teve em consideração a opinião de programadores profissionais. Segundo o site Stack Overflow (2022), o *Flutter* e o *React Native* são as duas ferramentas, dentro das ferramentas multiplataforma, mais populares, sendo que neste ano foi o *React Native* que deteve mais destaque na comunidade.

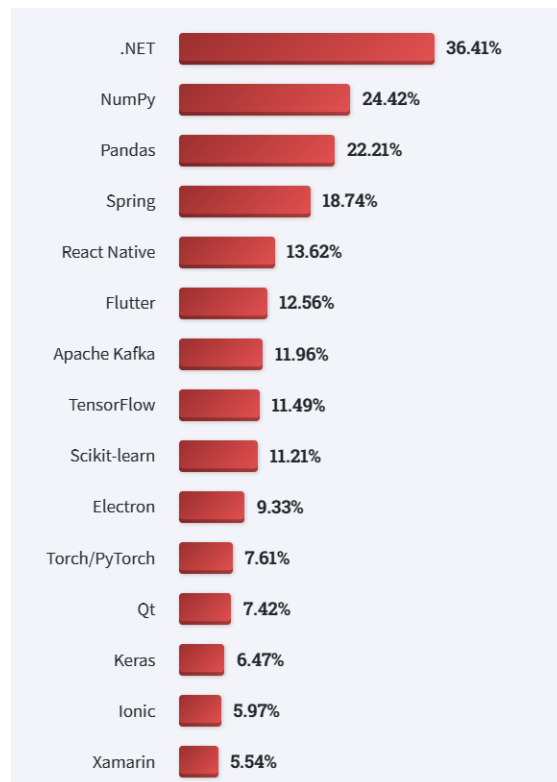


Figura 17: Classificação das linguagens de programação mais populares. Fonte: Stack Overflow (2022)

Assim, como o *React Native* é uma *framework* baseada em *React*, tem o suporte de uma grande comunidade de especialistas no desenvolvimento de *software*, da mesma forma que oferece um constante desenvolvimento e atualização por parte dos especialistas, torna-se assim adequada para o propósito da aplicação (Wu, 2018). A escolha permitiu criar uma solução que é compatível com os dois sistemas operacionais, atualmente líderes do mercado. Da mesma forma, o desenvolvimento nativo da *framework* trará maior fiabilidade e performance, assim como uma melhor interação do utilizador (Danielsson, 2016).

Outra ferramenta valiosa no desenvolvimento *frontend* é o **Expo**, já mencionada anteriormente, que é uma *framework* de código aberto que simplifica o processo de criação de aplicações em *React Native* (Expo, 2022). O uso de *Expo* deve-se ao facto de não ser necessário programar em *iOS* ou *Android* nativo, visto que faz o processamento todo do código nativo em segundo plano, tornando assim o desenvolvimento mais simplificado. Além disso, tem a funcionalidade da aplicação *Expo Go*, para *Android* e *iOS*, que dá a possibilidade de executar o projeto no dispositivo móvel físico, conseguindo assim ter uma perceção mais real da interação que o utilizador possa ter com a aplicação (Expo, 2022).

Tendo em conta estas tecnologias mencionadas foi essencial escolher entre **JavaScript** e **TypeScript**.

O *JavaScript* é uma linguagem de *script* amplamente utilizada do lado do cliente, semelhante ao *Python*. As linguagens de *script*, focadas no cliente, são frequentemente utilizadas para tornar os sites mais interativos, dinâmicos e responsivos (Prescott, 2016). Além disso, o *JavaScript* desempenha um papel fundamental ao possibilitar uma melhor interação dos utilizadores com aplicações, uma vez que é usado para controlar o navegador, realizar comunicações assíncronas com servidores, alterar dinamicamente o conteúdo das páginas da *web* e desenvolver jogos e aplicações *mobile* (Prescott, 2016).

Por outro lado, o *TypeScript* é uma extensão do *JavaScript* que visa facilitar o desenvolvimento de aplicações *JavaScript* de grande escala. Embora todo o programa *JavaScript* seja válido em *TypeScript*, este último oferece um sistema de módulos, classes, *interfaces* e um sistema fortemente tipado. O *TypeScript* foi projetado para proporcionar um suporte leve aos programadores, tornando os sistemas de módulos e *types* flexíveis e fáceis de usar. O *TypeScript* é também compatível com muitas práticas de programação comuns em *JavaScript*. O propósito do *TypeScript* é permitir uma transição suave para programadores *JavaScript*. É importante destacar que o sistema de *types* do *TypeScript* não é estático por *design*, o que oferece flexibilidade adicional aos que desenvolvem *software* (Bierman et al., 2014).

Em suma as principais diferenças entre estas duas linguagens resumem-se em:

- *TypeScript* tem um recurso conhecido como tipagem estática, mas o *JavaScript* não tem esse recurso.
- Qualquer código escrito em *JavaScript* pode ser convertido para *TypeScript*.
- *TypeScript* suporta bibliotecas do *JavaScript*.
- *TypeScript* requer mais desempenho na compilação do código.

As características distintas do *TypeScript* como a tipagem estática é importante para garantir a integridade dos dados e reduzir erros durante o desenvolvimento. Ou seja, no *JavaScript* é possível atribuir qualquer tipo de valor a uma variável. Enquanto que, no *TypeScript*, deve-se declarar os tipos de variáveis, parâmetros de função e valores de retorno. Este sistema de *types*, do *TypeScript* ajuda a evitar erros a longo prazo. Isto provoca, uma sobrecarga no tempo de compilação (quando está a fazer a verificação do código) devido à tipagem, mas não afeta o desempenho do código em tempo de execução (quando se executa o programa), o *JavaScript*, por sua vez não necessita de lidar com as verificações de *types*.

Inicialmente, o *JavaScript* era a escolha natural para *frameworks* como o *React Native*. No entanto, para aplicações de grande escala, o *JavaScript* pode apresentar problemas de escalabilidade e estruturação do código a longo prazo. Além disso, o *JavaScript* é uma linguagem não tipada, baseada em protótipos, multi-paradigma e dinâmica, suportando estilos de orientação a objetos (Flanagan, 2004). Por isso, a solução será *TypeScript*, uma linguagem de código aberto que é um superconjunto de tipagem estática do *JavaScript*, desenvolvida e suportada pela *Microsoft*, que tem como objetivo resolver alguns dos erros e problemas ao adicionar algumas melhorias ao *JavaScript*. Com o uso de *TypeScript* no projeto, o nível de exigência e dificuldade vão aumentar, dadas as especificidades da linguagem. No entanto, a médio-longo prazo será benéfico visto que tem vantagens comparativamente a *JavaScript*, tais como, integração com outras tecnologias, forte tipagem, o código é mais legível e incorpora *interfaces* e classes, visto que pode ser usada como uma linguagem de programação orientada a objetos (Vinet e Zhedanov, 2011).

3.3 Tecnologias para Backend

Uma *Application Programming Interface (API)*, é uma *interface* que possibilita a comunicação de um *software* com outro. Atua como uma camada, e processa a transferência de dados entre sistemas, permitindo assim a partilha de dados e as funcionalidades com aplicações externas (IBM, 2023).

No contexto das *API*, encontra-se o conceito de *REST API*. Uma *REST API* (Transferência Representativa de Estado) é um estilo de arquitetura de *software* padronizado que possibilita a comunicação entre cliente e servidor, permitindo pedidos em ambas as direções (AWS, 2023).

Um serviço *REST API* contém em dois componentes principais:

- Pedido: enviado pelo cliente para o servidor.
- Resposta: enviado de volta ao cliente.

Uma *REST API* utiliza requisições *Hypertext Transfer Protocol (HTTP)* para realizar operações fundamentais na manipulação de dados. Estas operações são referidas como *Create, Read, Update e Delete (CRUD)*, definem as principais interações possíveis com o servidor:

- *POST*: usado para criar dados no servidor.
- *GET*: usado para ler dados do servidor (usando a *URL*).
- *DELETE*: usado para excluir informações.

- *PUT*: usado para atualizar registros.

As respostas a estas solicitações, feitas por meio do protocolo *HTTP*, são geralmente formatadas em *JavaScript Object Notation (Json)*, que é interpretado por várias linguagens de programação modernas.

No contexto das *API* e da comunicação entre sistemas, é fundamental ter uma plataforma robusta que permita executar o código do lado do servidor de maneira eficaz. Como é o caso do *Node.js*. O *Node.js* não é uma linguagem de programação, mas sim um ambiente de execução *JavaScript*. Enquanto o *JavaScript* nasceu originalmente para melhorar a interatividade em navegadores, o *Node.js* permite que o *JavaScript* seja executado de forma independente nos servidores. Esta capacidade torna o *Node.js* uma escolha versátil para a criação de servidores *web*, permitindo o desenvolvimento de uma ampla variedade de aplicações *web*, desde servidores para sites estáticos e dinâmicos até sistemas robustos baseados em microserviços. Assim, o *Node.js* oferece uma arquitetura flexível, adequada para o desenvolvimento de microserviços. Além disso, muitos provedores de serviços de *cloud* já oferecem suporte ao *Node.js*, permitindo a criação de soluções altamente escaláveis.

No cenário do desenvolvimento *backend* em *Node.js*, o **NestJS** surge como uma opção eficiente e altamente organizada. É uma *framework* projetada para a construção de aplicações *Node.js*. O que a torna notável é sua abordagem progressiva ao *JavaScript* e a compatibilidade com *TypeScript*, permitindo que os programadores combinem conceitos de *Object-Oriented Programming (OOP)*, *Functional Programming (FP)* e *Functional Reactive Programming (FRP)*. Além disso, O *NestJS* fornece um alto nível de abstração, evitando grande parte da complexidade das *frameworks* tradicionais de servidores *HTTP* e, também disponibiliza *APIs* para oferecer aos programadores um controle refinado sobre o desenvolvimento das aplicações (NestJS, 2023).

Resumidamente, o *Node.js* desempenha um papel fundamental na área do desenvolvimento de aplicações *server-side*, atuando como a infraestrutura que permite que muitas aplicações *online* funcionem de maneira eficaz nos servidores. Assim, o *NestJS*, por sua vez, é uma escolha eficiente e robusta para criar aplicações *Node.js* altamente escaláveis e bem estruturadas. Juntos, proporcionam uma base sólida para atender a procuras complexas.

Outra ferramenta crucial é o **Firebase**, um *Backend-as-a-Service (BaaS)* fornecido pela *Google*. Isto é, um modelo de serviço *cloud* no qual existe a automatização do desenvolvimento do *Backend*. Este *BaaS* fornece uma variedade de ferramentas e serviços que facilitam a construção e crescimento de aplicações de qualidade. Uma das ferramentas proporcionada pelo *Firebase* é *Firebase Cloud Messaging (FCM)*,

de acordo com Firebase Docs (2023) é uma solução para enviar notificações entre plataformas. Isto é, significa a permissão de direcionar mensagens utilizando segmentos predefinidos ou personalizados, com informações demográficas ou de comportamento. Outra ferramenta proporcionada pelo *Firebase* é a *Firebase Authentication* (Google Developer, 2020), que possibilita a autenticação com o *Facebook*, *Google* ou *GitHub*, mas também permite a autenticação sem o uso de terceiros, através da criação de uma conta diretamente na aplicação com email e *password*. Estes dados ficam armazenados na *Cloud Firestore* do *Firebase*. A *Cloud Firestore* é uma base de dados flexível e escalável, *NoSQL*, que mantém os dados sincronizados entre as aplicações cliente, em tempo real (Google Inc., 2021).

Estas bases de dados *NoSQL*, são diferentes quando comparadas com as bases *Structured Query Language (SQL)*.

As bases de dados *SQL* são compostas por um conjunto de tabelas ligadas entre si de forma relacional, com base em chaves comuns associadas a cada uma, e têm um esquema predefinido e rígido (Helland, 2016). Neste caso, todos os dados inseridos na base de dados devem seguir o esquema definido. Isso implica a integridade referencial entre tabelas, o que torna as bases de dados *SQL* mais eficazes na preservação da consistência dos dados. No entanto, essa rigidez reduz a sua flexibilidade, uma vez que se torna mais difícil alterar o esquema de dados ao longo do tempo (Watt, 2014). Por outro lado, o *NoSQL*, que significa “*Not Only SQL*”, é uma categoria de base de dados que oferece uma alternativa às estruturas de dados tabulares utilizadas em base de dados relacionais. Estes sistemas proporcionam um ambiente flexível para armazenamento e recuperação de dados, sendo especialmente adequados para dados não estruturados Becker e Sewell (2004). Assim, as bases de dados *NoSQL* modelam os dados de uma forma que exclui as relações tabulares fornecidas pelas bases de dados relacionais, o que permite à equipa de desenvolvimento utilizar esquemas de dados mais flexíveis (IBM, 2019). Isto significa que as bases de dados *NoSQL* são mais adequadas para dados semiestruturados e não estruturados. No entanto, não proporcionam o mesmo nível de integridade de dados que as bases de dados *SQL* impõem (Srinivasa e Hiriyannaiah, 2018). As bases de dados *NoSQL* armazenam os dados em coleções ou documentos, seguindo uma estrutura não normalizada. Ao contrário das bases de dados *SQL*, as bases de dados *NoSQL* não seguem necessariamente as propriedades *Atomicity, Consistency, Isolation, Durability (ACID)*, o que as torna mais flexíveis do que as bases de dados *SQL*. No entanto, elas não suportam operações *JOIN* e consultas complexas da mesma forma que as bases de dados *SQL*. Uma das vantagens das bases de dados *NoSQL* é a capacidade de dimensionamento horizontal para lidar com grandes volumes de dados.

Na Figura 18, apresenta-se a estrutura dos dados nas diferentes bases de dados. As bases de dados *SQL* são mais adequadas para dados estruturados com esquema predefinido. Os dados nesses sistemas são armazenados em tabelas com colunas e linhas, e eles seguem rigorosamente as propriedades *ACID* para a gestão de transações. Além disso, as bases de dados *SQL* suportam operações *JOIN* e consultas complexas, utilizando uma estrutura de dados normalizada. Já as bases de dados *NoSQL*, utilizam coleções e documentos, onde os documentos podem ter diferentes campos e estruturas. As bases de dados *NoSQL* são altamente escaláveis e são frequentemente usadas em cenários onde a velocidade e a escalabilidade são mais importantes do que a consistência absoluta dos dados.

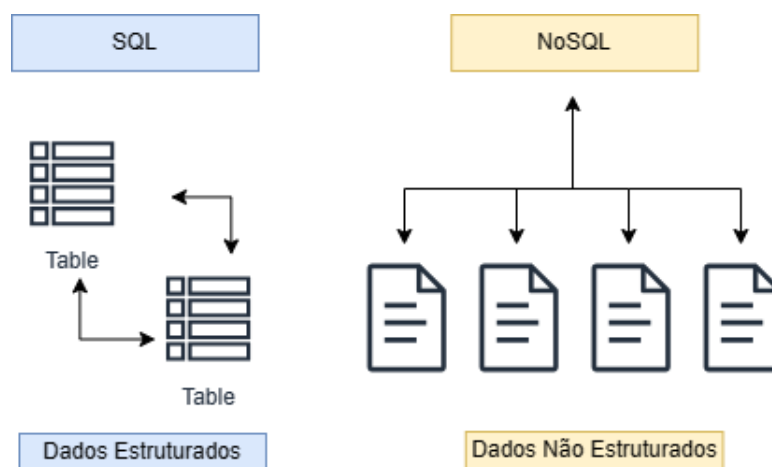


Figura 18: Estrutura dos dados nas diferentes bases de dados

3.4 Outras tecnologias utilizadas

Atualmente os sistemas *cloud* são cada vez mais necessários de forma a otimizar o processamento e o espaço de armazenamento, para isso existem vários fornecedores de serviços de *cloud*, incluindo a *Google*, a *Amazon*, a *Microsoft*. Embora estes fornecedores ofereçam produtos semelhantes, a implementação e os detalhes tendem a variar bastante (Geewax, 2018).

A *Google Cloud Platform (GCP)* permite o acesso a algumas das infra-estruturas internas da *Google*. *GCP* é uma coleção de produtos que inclui serviços comuns a todos provedores de *cloud* como máquinas virtuais através do *Google Compute Engine* ou armazenamento de objetos para guardar ficheiros através do *Google Cloud Storage*. Também inclui *API* para algumas das tecnologias mais avançadas criadas pelo *Google*, como *Bigtable*, *BigQuery*, *Cloud Pub/Sub* e *Cloud Datastore* (Geewax, 2018).

Na Figura 19 são apresentados vários produtos da *GCP* organizados por categorias. No contexto deste



Figura 19: Produtos da Google Cloud Platform

projeto, é de destacar a categoria de *Big Data*, que inclui produtos como o *Pub/Sub* e o *BigQuery*.

O **Pub/Sub** pode ser descrito como um serviço de mensagens assíncrono e escalável que separa os serviços que produzem mensagens dos serviços que processam essas mensagens (Google Cloud, 2021).

- Publica (*Pub*): envia a informação, pode corresponder aos serviços que desejam enviar dados.
- Subscrição (*Sub*): recebe a informação, corresponde aos serviços que recebem os dados.

Uma característica notável do *Pub/Sub* é a capacidade de distribuir eventos em tempo real. Eventos, tanto em estado bruto como processados. O *Pub/Sub* é compatível com padrões de *design* de aplicações orientadas a eventos. Além disso, oferece integração com muitos sistemas do *Google* que exportam eventos para essa plataforma (Google Cloud, 2021).

Para análise de dados, existe o **BigQuery** que consiste numa base de dados corporativa, que ajuda a gerir e analisar dados com recursos integrados. A arquitetura *Serverless* do *BigQuery* permite usar consultas *SQL* com facilidade, sem ser necessário gerir a estrutura. O mecanismo de análise distribuída e escalável do *BigQuery*, permite consultar *terabytes* em segundos e *pentabytes* em minutos, sendo, por isso, ideal para o tratamento de *Big Data*, grandes quantidades de dados, que é o caso dos dados dos utilizadores

que permitirem partilhar os seus dados pessoais, que serão tratados e eventualmente servirão para criar as campanhas das empresas que os utilizadores poderão usufruir (Google, 2023).

Para a gestão e organização de projetos, o **Jira Software** é uma ferramenta que suporta várias metodologias ágeis, incluindo *scrum*, *kanban* e metodologias mistas. Esta ferramenta consegue conter todo o trabalho que a equipa precisa de entregar durante um *sprint* (Atlassian, 2023b). O *Jira*, permite uma organização mais coesa e facilitadora, mas para além disso, possui outras ferramentas como o *Confluence*, onde é possível gerir o projeto e criar um espaço para toda a documentação de suporte (Atlassian, 2023a).

O **Postman**, é uma ferramenta projetada para simplificar o processo de teste e documentação de *APIs*. Este permite a criação e execução de testes automatizados, como solicitações *HTTP*, definir parâmetros e validar respostas. Uma das suas vantagens mais notáveis é a capacidade de organizar solicitações em coleções, tornando a gestão de várias requisições mais fácil, e também proporciona uma visualização clara dos dados recebidos.

Por fim, foi o **Figma** é uma ferramenta que oferece recursos avançados para criação de *wireframes*, *textitmockups* e protótipos de aplicações e sites. Esta ferramenta permite identificar possíveis problemas de usabilidade e contribui para o desenvolvimento de uma *interface* intuitiva.

3.5 Conclusão

A seleção das tecnologias e *frameworks* para o desenvolvimento deste projeto de gestão de dados para empresas de retalho foi um processo cuidadoso e estratégico. Embora a empresa não tenha imposto requisitos específicos em relação a todas as *frameworks* a serem utilizadas, a escolha foi fundamentada numa pesquisa sólida e nas necessidades específicas do projeto, tendo sempre em conta a falta de experiência da equipa e a validação da empresa.

No que diz respeito ao *Frontend*, a escolha incidiu pelo uso do *React Native*. O uso do *Expo* simplifica ainda mais o processo de desenvolvimento, eliminando a necessidade de programar em *iOS* ou *Android* nativo. A possibilidade de executar o projeto em dispositivos móveis reais utilizando o *Expo Go* proporciona uma visualização mais precisa da interação do utilizador. A decisão entre *JavaScript* e *TypeScript* foi baseada na escalabilidade e na estruturação sustentável do código, com o *TypeScript* a oferecer vantagens significativas a longo prazo, apesar de aumentar a dificuldade do projeto. Esta escolha foi estendida para todas as componentes necessárias do projeto, uma vez que melhora, de forma significativa, a qualidade

do código final.

Para o *Backend*, optamos pelo *NestJS*, uma *framework* para criar aplicações eficientes e escaláveis em *Node.js*. Essa escolha proporciona uma combinação de Programação Orientada a Objetos, Programação Funcional e Programação Reativa Funcional, juntamente com uma abstração acima das *frameworks* convencionais de servidor *HTTP Node.js*, esta *framework* foi sugerida por um dos membros da equipa tendo em conta a sua experiência.

O *Firebase*, fornecido pela *Google*, já pré-definido pela empresa, desempenha um papel fundamental, fornecendo uma variedade de ferramentas e serviços para simplificar o desenvolvimento do *Backend*, incluindo autenticação de utilizadores, armazenamento de dados flexível na *Cloud Firestore* e o uso do *FCM*.

Para análise de dados, o *BigQuery* é a escolha, graças à sua capacidade de gerir e analisar grandes volumes de dados de forma eficiente. Esta capacidade é fundamental para processar os dados dos utilizadores e criar campanhas personalizadas.

Em termos de gestão de projetos, o *Jira Software* oferece suporte a várias metodologias ágeis, proporcionando uma organização coesa para a equipa de desenvolvimento. O *Confluence* complementa essa capacidade, fornecendo um espaço para a documentação de suporte do projeto.

Por fim, o *Figma* permite a criação visual sólida da interface do utilizador coesa e integrada. Facilita na colaboração e desenvolvimento de *mockups* e protótipos de *interfaces*.

Em resumo, a seleção criteriosa destas tecnologias e *frameworks* está alinhada com os objetivos e requisitos do projeto de gestão de dados para empresas de retalho, garantindo não apenas o desenvolvimento eficiente da aplicação, mas também a capacidade de manutenção, escalabilidade e análise de dados de forma eficaz.

4 ANÁLISE DE REQUISITOS E ARQUITETURA DE SOFTWARE

4.1 Introdução

Este capítulo surge com principal foco na análise do problema presente nesta dissertação. O processo de análise e levantamento de requisitos funcionais e não funcionais, advém da necessidade de perceber o problema levantado e retirar ideias preliminares à formulação da solução pretendida. É focado em três grandes domínios: o reconhecimento do problema, o reconhecimento dos motivos pelos quais solucionar e, por último, o reconhecimento dos intervenientes envolvidos (Nicolás e Toval, 2009). Esta fase de planeamento, leva a uma análise mais detalhada e objetiva do problema, que irá facilitar a compreensão e posterior implementação do mesmo.

Além disso, o capítulo inclui a solução desenvolvida, em termos da arquitetura de *software* e da plataforma tecnológica, que resume algumas das tecnologias apresentadas na Secção 3.1 e, como estas se completam para o projeto. Por fim, é apresentado o modelo de dados adotado.

4.2 Requisitos funcionais

Os requisitos funcionais devem ser estritamente alinhados com a realidade do negócio. É essencial realizar verificações contínuas para garantir que as funcionalidades estão a ser desenvolvidas conforme planeado e o *design* da aplicação. Neste cenário, identificam-se quatro entidades principais: o utilizador e a empresa, que representa os retalhistas dentro da nossa aplicação, o gestor de campanhas e o gestor da aplicação.

O Diagrama de Caso de Uso na Figura 20 representa uma visão geral das funcionalidades da aplicação “*Take My Data*”. Este diagrama serve como uma representação de alto nível dos principais atores e casos de uso envolvidos no sistema. As quatro entidades principais representam diferentes funções:

- Cliente: representa os clientes da aplicação que partilham os seus dados pessoais e interagem com as campanhas.
- Empresa (Retalhista): representa as empresas que utilizam a aplicação para criar e gerir campanhas direcionadas aos utilizadores.
- Gestor de Campanhas: refere-se à entidade responsável pela gestão das campanhas e a sua interação com os utilizadores.

- Gestor da Aplicação: representa uma entidade que supervisiona o funcionamento geral da aplicação e assegura o cumprimento das políticas e regulamentos.

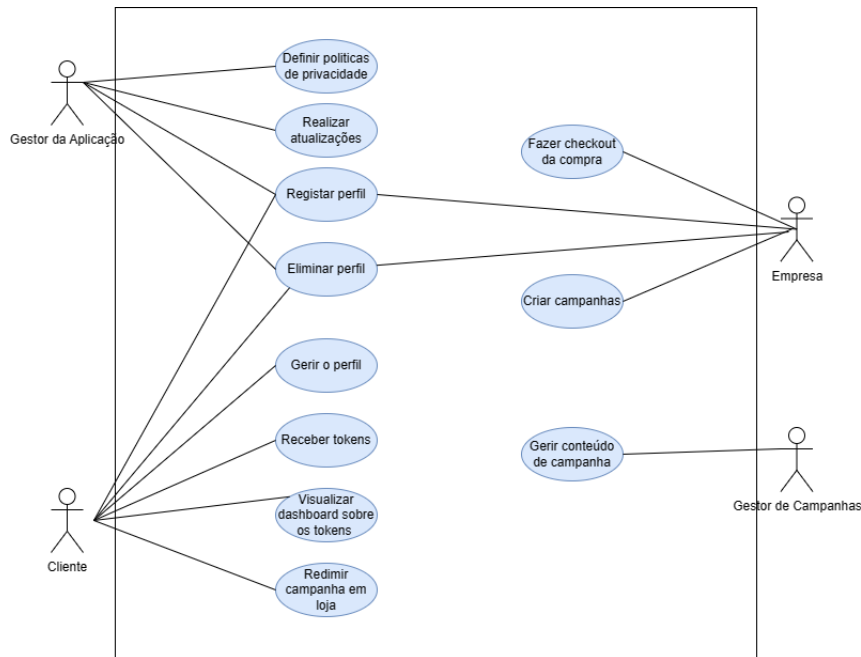


Figura 20: Diagrama de caso de uso

O Diagrama de Caso de Uso da Figura 21 é uma extensão do diagrama apresentado anteriormente. Que destaca as funcionalidades específicas relacionadas à gestão de perfis. A partir deste caso de uso principal, surgem vários casos de uso secundários, como “Eliminar Perfil” e “Efetuar o Login”. Relacionado com estes casos de uso secundários temos a entidade “Cliente”.

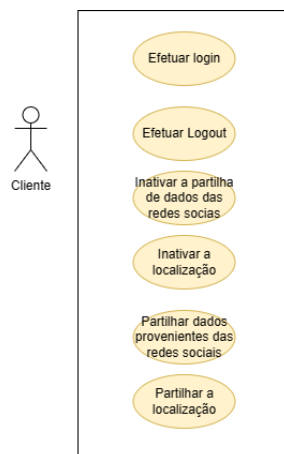


Figura 21: Diagrama de “Gerir perfil” mais detalhado

O mesmo acontece neste último diagrama, na Figura 22, da qual representa uma extensão da funcionalidade “gerir conteúdo da campanha”. Para este caso de uso, apenas temos uma entidade

envolvente- o gestor de campanhas. Este diagrama é fundamental para compreender como é que os gestores de campanhas interagem com a aplicação, permitindo-lhes realizar ações específicas relacionadas à criação, edição e gestão das campanhas.

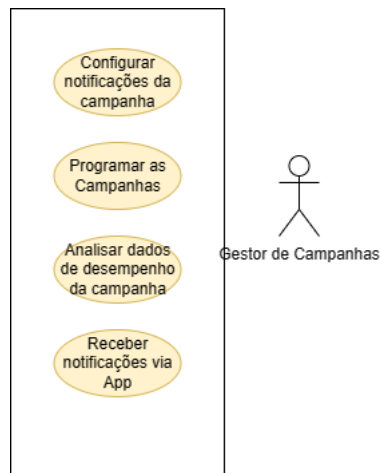


Figura 22: Diagrama com o caso de uso “Gerir Campanha” mais detalhado

De seguida, são apresentados, em detalhe, os diferentes requisitos funcionais.

RF001: Como utilizador (cliente) , quero criar perfil

O sistema deve ser capaz de permitir ao cliente (ou empresa) de registar o seu perfil, de forma a permitir utilizar a aplicação. Para registar o perfil, o cliente deve definir o nome, data de nascimento, nome de utilizador, email e palavra-passe.

RF002: Como utilizador (cliente), quero efetuar o login

O sistema deve ser capaz de permitir ao cliente (ou empresa) iniciar sessão na aplicação para poder usar a aplicação, na sua conta. O cliente deve inserir as suas credenciais, email e a palavra-passe, que se estiverem corretos, conseguirá ter acesso à sua conta.

RF003: Como utilizador (cliente), quero editar o meu perfil

O sistema deve ser capaz de permitir ao cliente (ou empresa) editar os seus dados na aplicação, quer seja o email ou palavra-passe.

RF004: Como utilizador (cliente), quero eliminar o meu perfil

O sistema deve ser capaz de permitir ao cliente (ou empresa) eliminar definitivamente o seu perfil, se

entender que não pretende que a aplicação não continue com os seus dados.

RF005: Como utilizador (cliente), quero partilhar a minha localização

O sistema deve ser capaz de permitir o cliente partilhar a localização com a aplicação. Desta forma, a aplicação vai ter acesso à sua localização, que poder ser usada para, por exemplo, o avisar de alguma campanha de seu interesse, se estiver localizado perto de uma loja parceira da aplicação. Quando o cliente inicia a aplicação pela primeira vez, depois de criar a sua conta, vai lhe ser pedido para partilhar a sua localização, ou, se o cliente recusar inicialmente, ele pode partilhar a localização a partir das definições.

RF006: Como utilizador (cliente), quero desativar a partilha da minha localização

O sistema deve ser capaz de permitir o cliente desativar a partilha da localização com a aplicação. Deste modo, a aplicação não vai ter mais acesso à localização.

RF007: Como utilizador (cliente), quero partilhar os dados das minhas redes sociais

O sistema deve ser capaz de permitir o cliente escolher que rede social deseja permitir que os dados sejam extraídos. Desta forma, irá permitir que os dados sejam processados posteriormente. Tal como a localização, ao iniciar a aplicação pela primeira vez, o cliente pode escolher se deseja partilhar os dados de alguma rede social, ou pode alterar a sua escolha nas definições da aplicação.

RF008: Como utilizador (cliente), quero desativar a partilha dos dados das minhas redes sociais

O sistema deve ser capaz de permitir o cliente desativar a partilha de dados de uma ou mais redes sociais. Com isto, a aplicação deixa de ter acesso aos dados pessoais do cliente nessa rede social.

RF009: Como utilizador (cliente), quero receber campanhas específicas para o meu perfil

O sistema deve ser capaz de enviar campanhas focadas nos gostos do cliente, a partir do processamento e tratamento dos dados extraídos das redes sociais do cliente. Desta forma, o cliente recebe campanhas do seu interesse.

RF010: Como utilizador (cliente), quero receber notificações pela aplicação

O sistema deve ser capaz de enviar notificações para o cliente pela aplicação, se o cliente as ativar.

Estas notificações podem englobar desde campanhas que tenham sido inseridas no sistema que sejam do seu interesse, como também alguma campanha que esteja no seu “carrinho” tenha sido alterada, por exemplo.

RF011: Como utilizador (cliente), quero desativar o envio de notificações pela aplicação

O sistema deve ser capaz de desativar o envio de notificações para o cliente pela aplicação, se este assim o desejar.

RF012: Como utilizador (cliente), quero receber tokens

O sistema deve ser capaz de recompensar o cliente pelas suas ações na aplicação. Estão englobadas ações, tais como permissão para partilhar dados numa rede social, como ao receber uma campanha de uma empresa parceira, visto estar no perfil que a empresa tinha escolhido para que fossem enviadas tal campanha, por exemplo.

RF013: Como utilizador (cliente), quero verificar os meus tokens

O sistema deve ser capaz de apresentar os *tokens* que o cliente possui, assim como as transações de *tokens*, através de uma *dashboard*. Desta forma, o cliente vai poder não só ver as campanhas em que ganhou mais recompensas, como também os gastos em descontos ou outras vantagens na loja da aplicação.

RF014: Como utilizador (empresa), quero adicionar campanhas - selecionar a segmentação de perfil para campanha, carregar uma campanha (imagem) e enviar para base de potenciais clientes.

O sistema deve ser capaz de permitir a empresa criar campanhas, selecionando as regras e atributos necessários. Isto é, a empresa vai poder decidir campos como o público-alvo, dentro dos perfis (grupos) de utilizadores disponíveis, e também qual a recompensa pelos utilizadores que receberem essa campanha.

RF015: Como utilizador (empresa), quero fazer o checkout da compra realizada pelo cliente

O sistema deve ser capaz de permitir à empresa realizar *checkouts* pelas compras realizadas pelos clientes. Desta forma, a empresa conseguirá permitir atribuir os *tokens* aos clientes que usufruírem das suas campanhas e produtos, sendo os clientes também recompensados pelos dados fornecidos

anteriormente.

RF016: Como utilizador, quero usar uma campanha oferecida, numa loja física

O sistema deve ser capaz de permitir ao cliente usufruir de campanhas em lojas físicas. Isto vai permitir às empresas atraírem clientes para as lojas, como forma de levar o cliente a comprar mais do que até estava a pensar comprar. Além disso, o cliente também vai ser recompensado pela compra.

RF017: Como utilizador (gestor de campanhas), quero analisar dados de desempenho de campanhas

O sistema deve ser capaz de armazenar e organizar dados de desempenho de campanhas. Deve fornecer uma *interface* de *dashboard* que permita aos Gestores de Campanhas consultar esses dados de maneira eficiente. Além disso, o sistema deve ser capaz de calcular métricas de desempenho, como taxas de conversão, para cada campanha.

RF018: Como utilizador (gestor de campanhas), quero programar campanhas

O sistema deve permitir aos Gestores de Campanhas agendar campanhas, definindo datas e horários específicos para a ativação automática.

RF019: Como utilizador (gestor de campanhas), quero configurar as notificações da campanha

O sistema deve ter a capacidade de configurar notificações para utilizadores quando novas campanhas são lançadas. Isso envolve a criação de modelos de notificação e a capacidade de segmentar utilizadores com base em critérios específicos.

RF020: Como utilizador (gestor de campanhas), quero gerir o conteúdo de campanha

O sistema deve permitir que os Gestores de Campanhas atualizem o conteúdo das campanhas de maneira intuitiva. Deve incluir ferramentas de edição de texto, imagens e outros elementos de campanha.

RF021: Como utilizador (gestor da aplicação), quero definir políticas de privacidade

O sistema deve fornecer uma *interface* para que os Gestores da Aplicação definam e atualizem as políticas de privacidade da aplicação. Deve permitir a criação de políticas personalizadas, bem como a documentação das políticas de acordo com os regulamentos de privacidade.

RF022: Como utilizador (gestor da aplicação), quero realizar atualizações

O sistema deve oferecer uma funcionalidade de atualização que permita aos Gestores da Aplicação implementar novas versões da aplicação de forma eficaz. Isso envolve a possibilidade de fazer o *upload* e distribuição de atualizações para os utilizadores finais.

RF023: Como utilizador (gestor da aplicação), quero registar perfis

O sistema deve ter um mecanismo de registo de perfis que permita aos Gestores da Aplicação criar novos perfis de clientes. Deve incluir campos para informações do perfil, como nome, endereço de email, fotografia de perfil.

RF024: Como utilizador (gestor da aplicação), quero eliminar perfis

O sistema deve permitir a eliminação de perfis dos cliente. Isso envolve a remoção segura de dados do cliente e a confirmação da ação pelo Gestor da Aplicação.

4.3 Requisitos não funcionais

Para os requisitos não funcionais, foram considerados os seguintes:

RNF001: Usabilidade

A solução deve ser intuitiva e de fácil utilização, de modo que tanto os utilizadores finais que utilizam a aplicação, como as empresas que operam no *backoffice* possam interagir com ela sem necessidade de formação ou apoio adicional. A *interface* deve ser clara, com tamanho de fonte legível e uma paleta de cores acessível. Deve ser implementada com foco na eficiência e usabilidade.

RNF002: Confiabilidade

A solução deve manter a confidencialidade dos dados dos utilizadores, quer de autenticação, como os seus dados pessoais, de acordo com as medidas de segurança da proteção de dados. A sua utilização deve ser exclusivamente para as funcionalidades requeridas.

RNF003: Desempenho

A solução deve ter em conta o desempenho do *software*, tal como o tempo de resposta ou consumo de memória.

RNF004: Suportabilidade

A solução deve ser testada na sua totalidade, bem como deve ser mantida a escalabilidade e a sua manutenção, de modo a ter uma solução estável e segura.

RNF005: Requisitos de Design

A solução deve manter a uniformidade nas bibliotecas de classes (ex: *styled-components*), ícones, bem como o processo de *software*.

RNF006: Restrições de Implementação

A solução deve ser implementada de forma controlada pelas suas diferentes versões, com o uso de ferramentas como o *Git*. Além disso, também segue padrões de desenvolvimento na implementação, de forma a garantir a homogeneidade.

4.4 Arquitetura de software e Plataforma tecnológica

A arquitetura apresentada é constituída por diferentes componentes, cada um com uma função específica, que comunicam entre si através de protocolos de rede para fornecer funcionalidades ao sistema como um todo.

De forma a explicar e compreender qual é o fluxo de dados entre as camadas da nossa aplicação, foi desenvolvido o diagrama de infraestrutura apresentado na Figura 23.

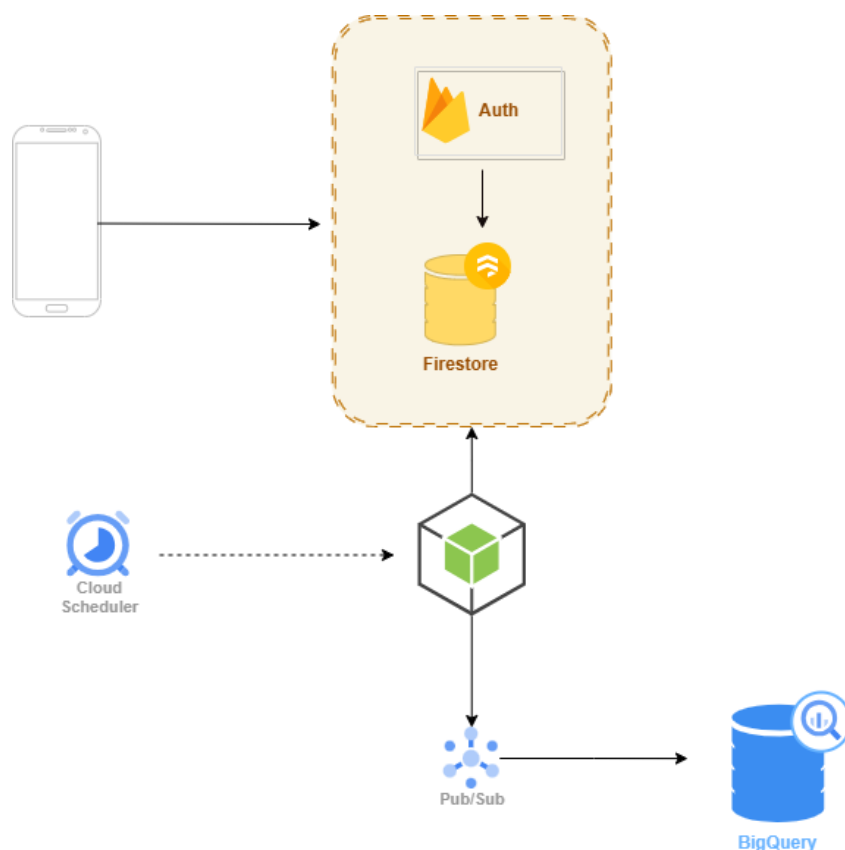


Figura 23: Diagrama de arquitetura de alto nível

A Camada de *interface* do utilizador inclui a *interface* do cliente, que os utilizadores finais utilizam para interagir com o sistema. Esta conecta-se ao *Firebase* para autenticação e armazenamento de dados.

A segunda camada atua fornecendo uma *API* para gerir as solicitações do dispositivo. A *API*, que foi construída utilizando o *Node.js* é projetada para recolher dados e publicá-los na fila de espera. A interação da *API* com o *Cloud Schedule* permite a configuração do agendamento para recolher dados automaticamente de acordo com uma programação definida.

Por fim, é utilizado o *Big Query*, para armazenar grandes volumes de dados e executar consultas analíticas complexas. Para isto ser possível é utilizado o *Pub/Sub* para pré-processamento dos dados recebidos antes de serem armazenados no *Big Query*.

De forma a serem apresentadas as diferentes tecnologias foi desenvolvido um diagrama, dividido por camadas. Neste diagrama estão referidas as *frameworks* escolhidas no desenvolvimento.

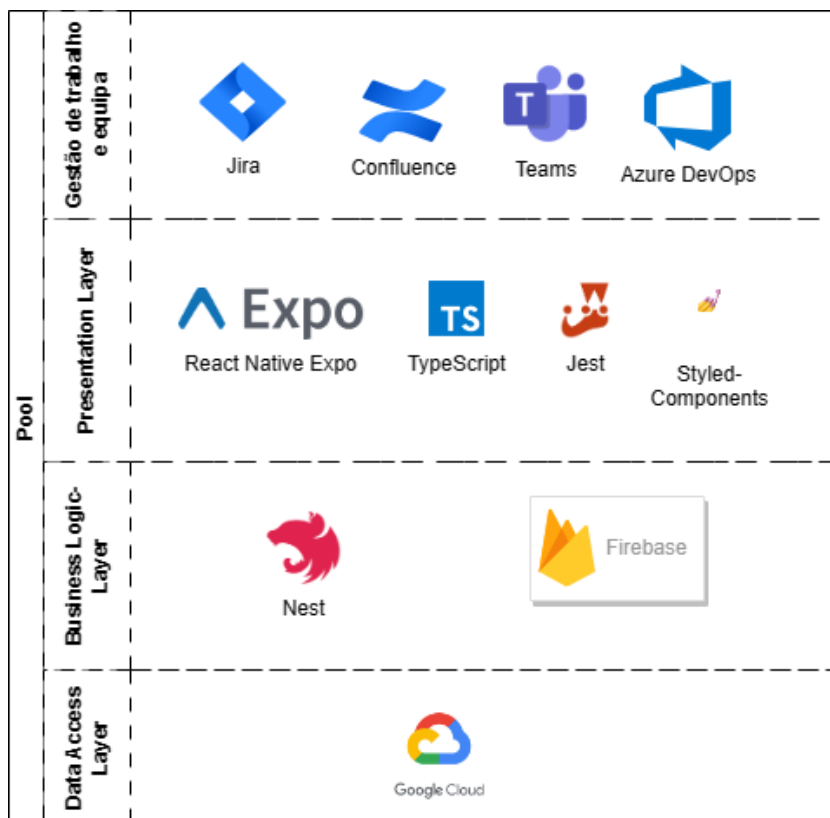


Figura 24: Plataforma Tecnológica

O padrão *Layers*, é um dos padrões mais comuns utilizados na arquitetura de *software* (Harrison e Avgeriou, 2010). Como definição, é citado Larman (2005), que considera como padrão *Layers*: “Organizar a estrutura lógica de grande escala de um sistema em camadas discretas de responsabilidades distintas e relacionadas, com uma separação limpa e coesa de preocupações, de modo que as camadas “inferiores” sejam serviços de baixo nível e serviços gerais, e as camadas superiores são mais específicas da aplicação”.

Como se pode verificar na Figura 24, as *layers* identificadas foram: *Presentation Layer (PL)*, *Business Logic Layer (BLL)* e *Data Access Layer (DAL)*.

Primeiramente temos a *PL*, onde é apresentada a *interface* com o utilizador, ou seja, a *interface* de quem acede à nossa aplicação. Para esta primeira versão do *MVP* apenas é desenvolvida a *interface* com os utilizadores finais, ou seja, aqueles que utilizam a aplicação para aceder a campanhas. Posteriormente, é necessário desenvolver a *interface* destinada aos retalhistas, onde o foco recai no desenho da ideia, sem a necessidade da implementação final.

Como intermediário temos a camada de negócio (*BLL*). Esta camada é responsável pela lógica e pelas

regras de negócio (GeeksforGeeks, 2020). Para esta camada foram utilizados vários recursos disponíveis do *firebase* e *NEST.js* para fornecer funcionalidades adicionais e facilitar a implementação das regras de negócio da aplicação. Para isto, são necessários 3 componentes para garantir a gestão ao acesso aos dados, autenticação de utilizadores e processamento de eventos. Cada um requer o desenvolvimento de três projetos novos. Sendo estes:

- **Publicação da queue (Tópico Pub/Sub):** Um dos componentes é um método *POST* que permite publicar mensagens na fila de espera (tópico Pub/Sub). Esse componente desempenha um papel crucial na integração com o *Firebase*, pois este recebe dados dessa plataforma e encaminha-os para a fila de espera.
- **Processamento de Dados Contínuo:** O segundo componente permite realizar o processamento contínuo de dados, recebendo informações da fila de espera (*Pub/Sub*) e executando as etapas necessárias para garantir a integridade e consistência dos dados ao longo do tempo. Isso envolve o processo de *Extract, Transform, Load (ETL)* de forma contínua e, é fundamental para manter a qualidade dos dados ao longo do tempo.
- **Envio de Notificações:** O último componente tem como objetivo lidar com a lógica das notificações com base em critérios específicos.

Por último, a camada de acesso aos dados (*DAL*) contém todos os aspetos relacionados com os dados: como aceder, os seus comportamentos, como validar e as relações entre si.

E para isto foi utilizado um recurso do *GCP*, o *Big Query*, que é utilizado para realizar análises e consultas avançadas nos conjuntos de dados armazenados. Com o *Big Query*, é possível obter *insights* valiosos e realizar transformações nos dados de forma ágil, facilitando a obtenção de informações relevantes para a tomada de decisões no contexto do sistema.

4.5 Modelo de dados

Foi necessário compreender como os dados podem ser estruturados, de forma a armazenar informações relacionadas com os utilizadores do *Facebook*, incluindo detalhes pessoais, educação, desporto, publicações, comentários e interesses. Assim, o esquema do modelo da base de dados apresenta-se na Figura 25.

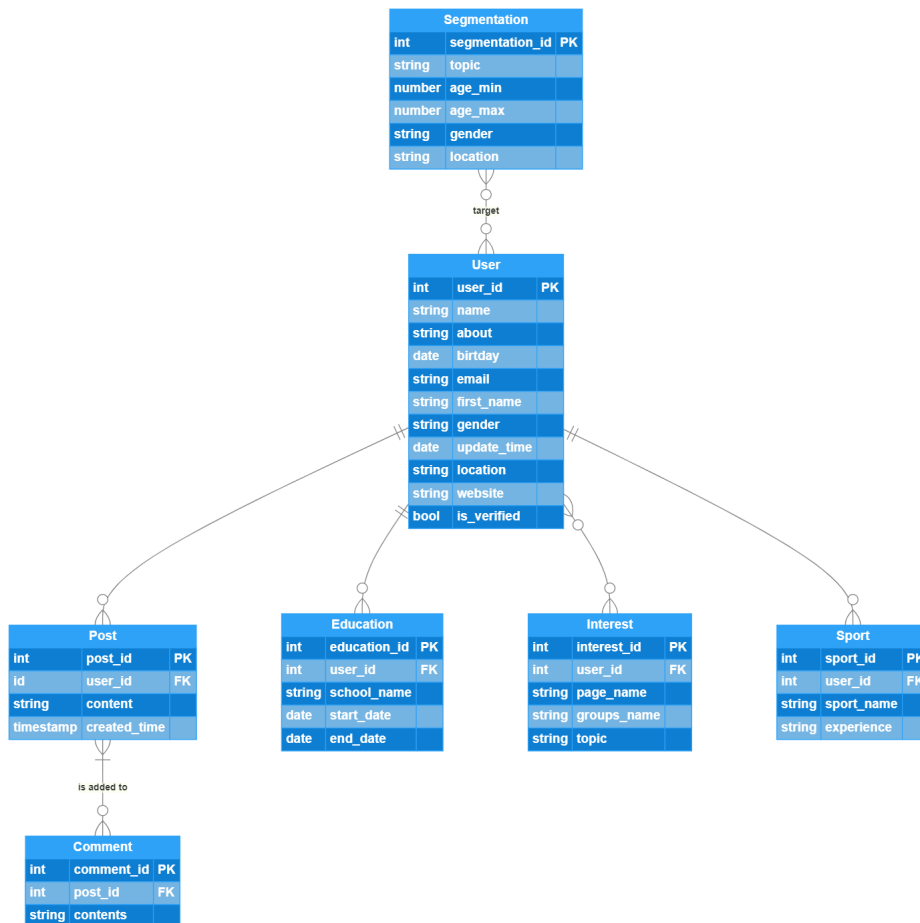


Figura 25: Esquema da Base de dados relacional

Com base no modelo de dados, conseguiu-se compreender alguns aspetos sobre o sistema:

- Cada utilizador é representado pela tabela *User*, a entidade principal que se relaciona com outras entidades;
- No caso da tabela *Post*, cada publicação está associada a um utilizador (*User*) específico.
- A tabela *Interest* contém informações sobre as páginas e grupos que um utilizador segue, que são

organizadas através do campo “*topic*”. Esta tabela tem um relacionamento de muitos para muitos com a tabela *User*. Ou seja, um utilizador pode ter vários interesses, e um interesse pode estar associado a vários utilizadores.

- A tabela *Education* contém informações sobre a escola onde o utilizador estudou, bem como as datas de início e término do período de estudo.
- A tabela *Sport* contém informações sobre o desporto praticado pelo utilizador e a sua experiência.
- Cada comentário está associado a uma publicação específica. Portanto, cada comentário pertence a uma publicação.
- Por fim, a tabela *Segmentation* é utilizada para armazenar características relevantes para uma campanha específica. Embora não esteja diretamente relacionada com as outras tabelas mencionadas, desempenha um papel fundamental na segmentação de utilizadores para campanhas específicas. Esta tabela também possui um relacionamento de muitos para muitos com a tabela *User*.

4.6 Conclusão

Neste capítulo foram explorados os principais aspetos relacionados com a compreensão do problema. Foi realizada a análise e levantamentos de requisitos para obter uma compreensão clara das necessidades e metas do projeto. Este passo foi crucial para a construção do *backlog* inicial. Ou seja, ao identificar os requisitos funcionais, pudemos mapear com precisão as funcionalidades essenciais da aplicação, como a criação de perfis de utilizadores e perceber outros aspetos que necessitavam de um estudo mais aprofundado.

A definição dos requisitos não funcionais, por outro lado, contribuiu para estabelecer padrões rigorosos para a usabilidade, confiabilidade, desempenho, suportabilidade, *design* e implementação da aplicação. Estes requisitos garantem, não apenas que o sistema seja seguro e eficiente, mas também que ofereça uma experiência de utilização satisfatória.

A arquitetura do sistema também se revelou uma peça fundamental neste projeto. Ao delinear a estrutura global do sistema, permitiu ter uma visão clara das diferentes camadas do sistema e da interação entre elas. A colaboração de todos os membros da equipa, incluindo *PO* e o *Tech Lead*, foi fundamental para garantir que todos estivessem alinhados e fosse transmitida com clareza o conceito da aplicação.

As tecnologias selecionadas, como o *Firebase*, *Node.js*, *BigQuery* e *Pub/Sub*, foram escolhidas com base na sua capacidade de atender aos requisitos do projeto e proporcionar uma infraestrutura sólida, mas também, tendo em conta as tecnologias recentes no mercado, o que criou ainda mais um desafio.

5 DESENVOLVIMENTO DA SOLUÇÃO

5.1 Introdução

Neste capítulo, é apresentada a análise e a conceção da solução para a aplicação *mobile*, abordando os principais aspetos da *interface* do utilizador e a estrutura de funcionamento da aplicação. Além disso é apresentado a lógica geral da implementação da solução.

Desta forma, a equipa de desenvolvimento começou por desenvolver uma *maquete* de como seria o *Frontend* da aplicação, na qual foram elaborados todos os ecrãs que os *PO* pretendiam.

5.2 Componentes de Frontend

Para o desenvolvimento da aplicação *mobile*, a experiência do utilizador foi um dos principais aspetos considerados. A criação de uma *interface* intuitiva e eficiente esteve sempre presente na conceção das *mockups* e, conseqüentemente, no desenvolvimento da aplicação. Como mencionado na Secção 3.4, foi utilizado o *Figma* para este propósito. É relevante destacar que este processo passou por duas fases distintas: a fase de conceção e a fase de desenvolvimento apresentadas a seguir. Para a fase de desenvolvimento, foi adotada a tecnologia *React Native* no ambiente *EXPO*, como já foi referido na Secção 3.2.

5.2.1 Ecrãs para o registo de conta

Em primeiro lugar, foram criados os ecrãs de registo de conta, dos termos e condições, confirmação de email e autorização para partilha de localização e dados das redes sociais, aquando o utilizador abre a aplicação pela primeira vez, conforme representado na Figura 26.

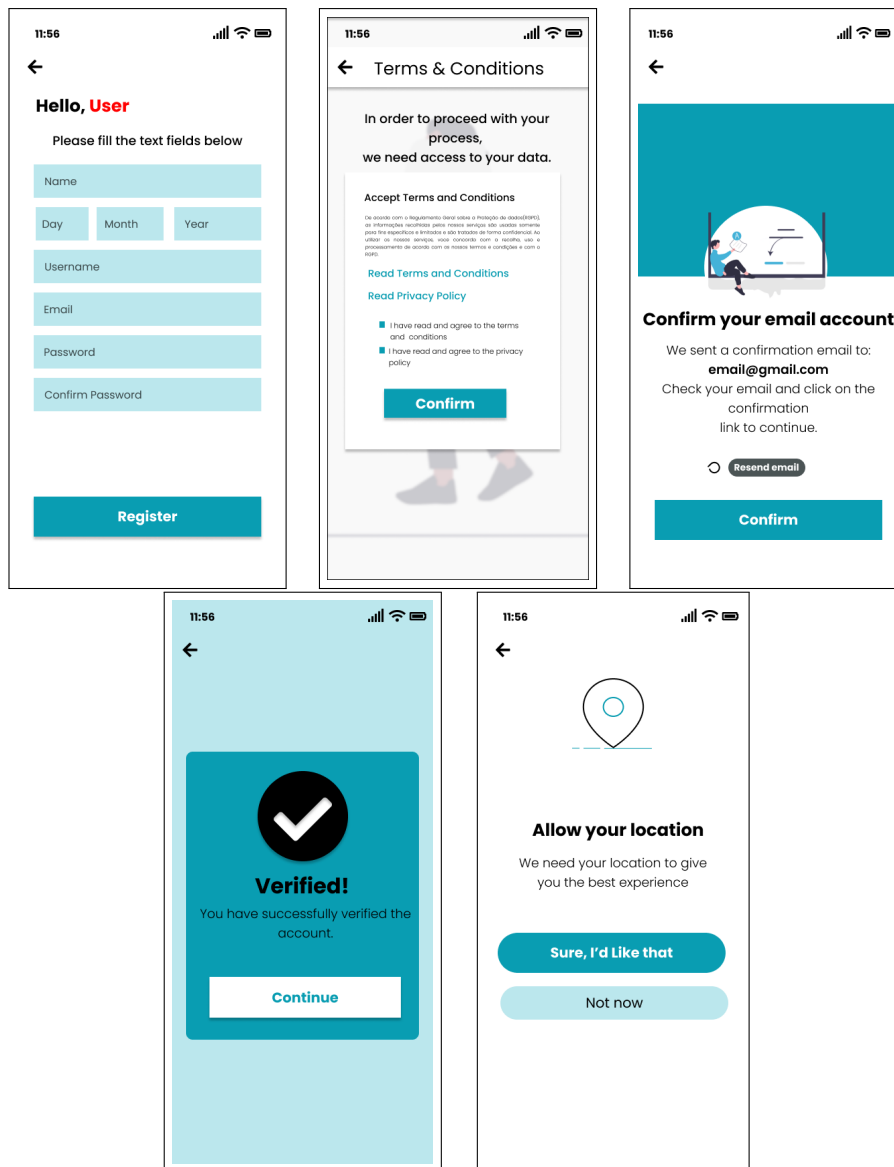


Figura 26: Registo de conta do utilizador

5.2.2 Ecrãs de Log In

Para os utilizadores que já possuem conta e apenas pretendem efetuar o *login*, foram criados os ecrãs da Figura 27. No processo de *Log In*, existe a opção de aceder à aplicação utilizando o *email* ou conta do *Google* (com a possibilidade de expandir para outras plataformas, como o *Facebook*). Além disso foram também elaborados os ecrãs que permitem que o utilizador recupere a palavra-passe, caso se tenham esquecido dela.

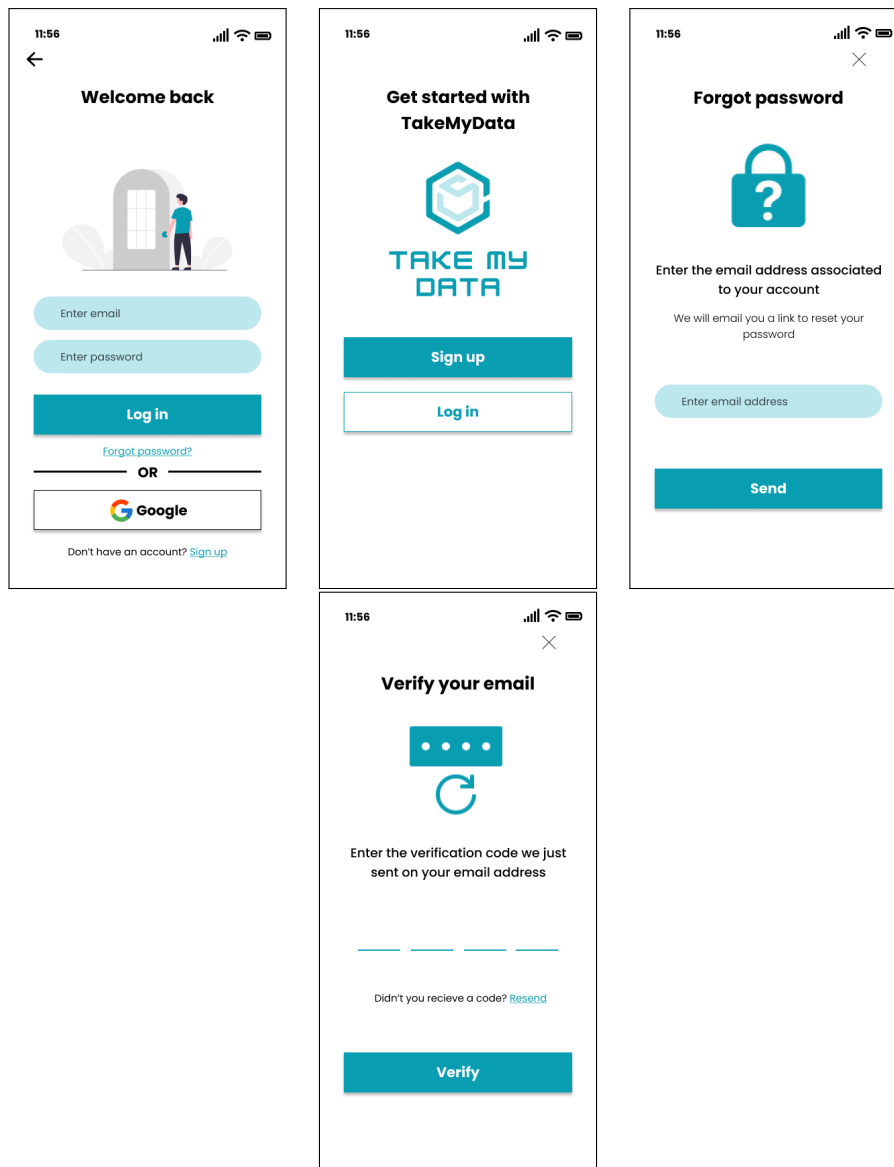


Figura 27: Entrada

5.2.3 Ecrãs de navegação

Foram desenhados ecrãs que permitem que os utilizadores naveguem por todas as funcionalidades oferecidas pela aplicação: desde o seu perfil, definições para alterar dados, notificações onde recebem as campanhas, *dashboard* onde podem acompanhar o seu saldo de *tokens*, movimentos, entre outras estatísticas importantes. Além disso, existe o *Marketplace* onde se pode redimir produtos oferecidos pela lojas. Também há a possibilidade de aceder ao carrinho de compras, para finalizar compras permitidas pela aplicação, ou até, caso se dirijam a uma loja física parceira, fazer a leitura do *QR code* associado ao produto disponível no *marketplace* ou campanha.

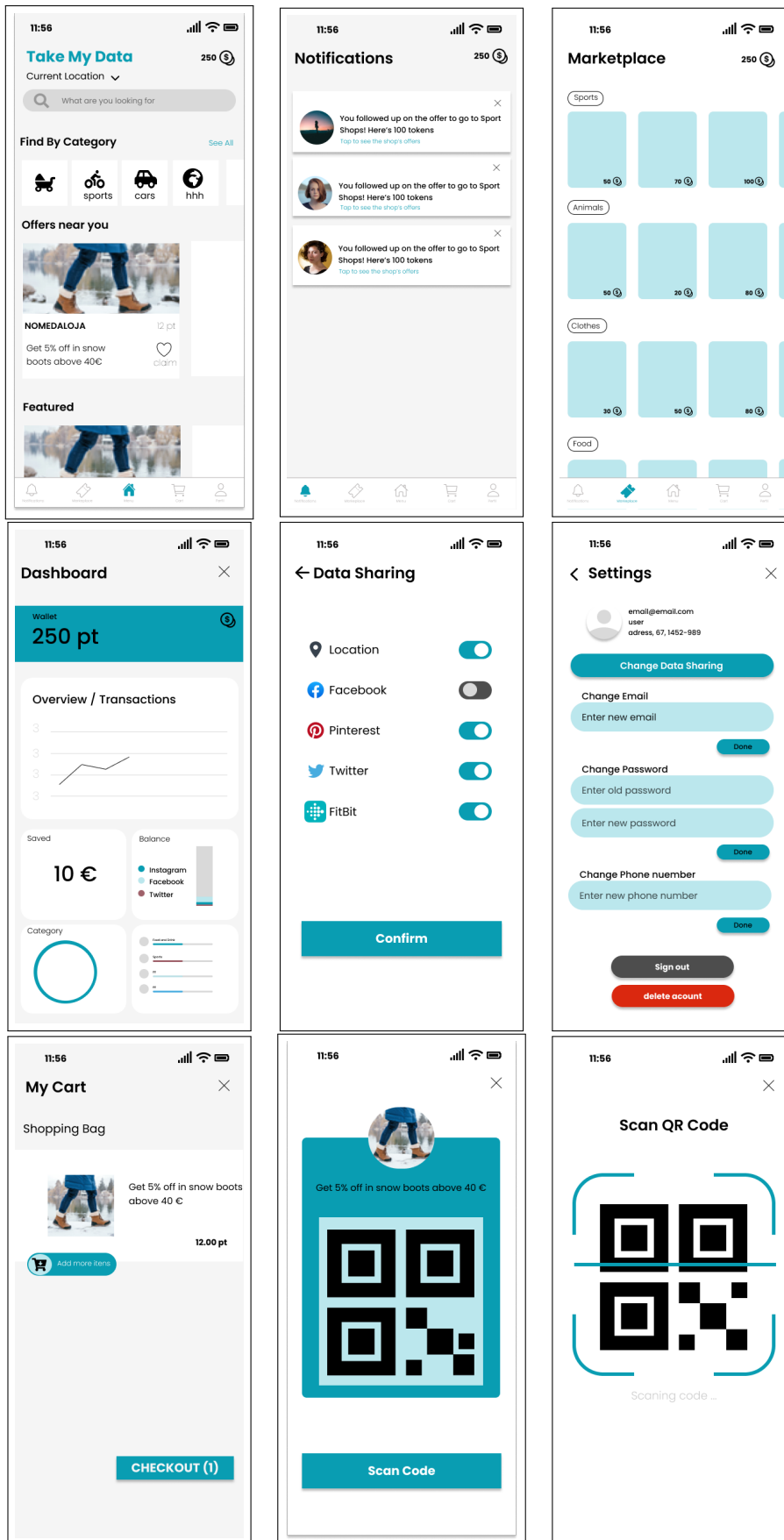


Figura 28: Ecrãs de navegação

5.3 Configurações iniciais

Após o desenvolvimento das *mockups*, e da sua devida aprovação, foi necessário definir padrões essenciais para o projeto.

A criação de uma base uniforme foi um passo crucial no início deste projeto. Foi estabelecido um conjunto de padrões e diretrizes de codificação que permitiu que a equipa trabalhasse de forma consistente e eficaz. Além disso, os processos de qualidade da aplicação foram cuidadosamente definidos para garantir que os requisitos e expectativas fossem atendidos.

Ao estabelecer o fluxo de trabalho, a empresa conseguiu otimizar a colaboração entre os membros da equipa. Isso envolveu a definição clara do modelo a seguir em cada tarefa relacionada com o projeto, com a definição do *git flow*, representado na Figura 29. Ou seja, foi definido que existe uma *branch* principal, denominada de *main*, representada na Figura 29 por “*master*”, onde apenas se vão verificar versões estáveis da aplicação. Abaixo desta, existe a *branch* denominada *develop*, que vai sendo atualizada ao longo dos *sprints*, tendo sempre em atenção que esta também não deve conter *bugs*. Por último, existem múltiplas *branches*, que advêm da *branch develop*, sendo criada uma *branch* por cada funcionalidade que foi sendo desenvolvida. No final desta funcionalidade estar devidamente estável e testada, é aberta um *Pull Request (PR)*, de modo a que depois de uma verificação seja feita uma fusão com a *branch develop*.



Figura 29: Git Flow, Fonte: Buddy: The DevOps Automation Platform (2020)

Com estes aspetos definidos foi permitido o desenvolvimento de uma aplicação de alta qualidade, com um código bem estruturado e processos de qualidade eficientes. Um aspeto importante deste processo foi a adoção de tarefas pequenas e de fácil revisão, o que possibilitou a identificação rápida de erros e a

implementação de melhorias contínuas.

Em relação à *code base*, foi utilizado como base o *TypeScript*, e foram implementados outros elementos de padronização e facilitação do desenvolvimento colaborativo. Como por exemplo, a inclusão do ficheiro *Prettier*, garantindo que toda a equipa de desenvolvimento seguisse as mesmas regras de formatação.

5.3.1 Arquitetura do Frontend

O *Frontend* adotou a arquitetura recomendada pelo *Expo* e *React Native*, a qual é também amplamente utilizada pela comunidade. Na Figura 30 está demonstrado como foi realizado.

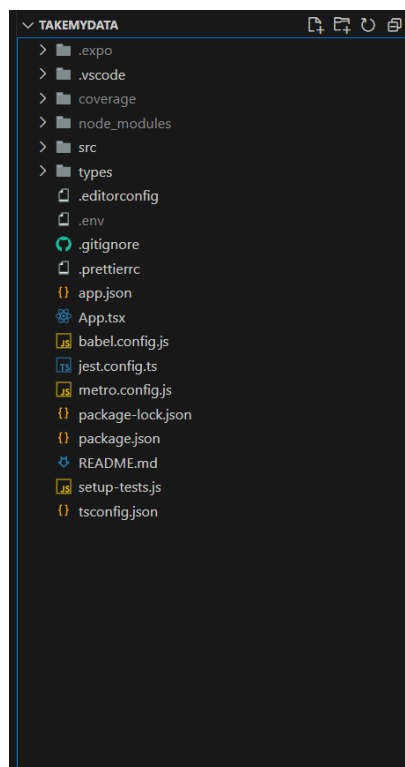


Figura 30: Arquitetura de pastas *Frontend*

Nesta arquitetura constam, entre outras, as seguintes pastas:

- **assets** - armazena todas as imagens e fontes;
- **components** - contém os componentes criados;
- **config** - armazena os diferentes ficheiros de configuração de diferentes ambientes do projeto;
- **constants** - contém todas as constantes utilizadas no projeto;

- **hooks** - contém todos os *hooks* do projeto, que são funções que podem ser usadas para isolar a parte reutilizável de um componente funcional;
- **interface** - contém as *interfaces* de todos os objetivos utilizados pelo projeto;
- **pages** - contém os ecrãs/páginas utilizados no projeto;
- **styles** - pasta para a definição e implementação do *Cascading Style Sheets (CSS)*. Foi utilizado o *styled-components*, visto que este é integrado com o *React* e *React Native*, além disso, permite a criação de estilos personalizados como fontes, cores e margens.

Desenvolvimento de ecrãs

Após este primeiro passo - a definição da *code base* -, o mais importante recaiu na funcionalidade principal da aplicação, ou seja, o desenvolvimento do ecrã da Partilha de Dados, representado na Figura 31, cujo propósito é fornecer ao utilizador uma seleção de redes sociais disponíveis, nas quais ele pode escolher deliberadamente os dados que o utilizador pretende partilhar com a aplicação. Este ecrã foi concebido para promover a transparência e controlo, por parte do utilizador, em relação à partilha de informações pessoais. De uma forma genérica, o ecrã apresenta uma lista das principais redes sociais, como *Facebook*, *Twitter*, *Pinterest* e *FitBit*. Esta escolha foi feita com base no pressuposto de conseguir informações relevantes a nível de preferências do utilizador, mas também de forma a perceber como ele se comporta no dia a dia. Para cada rede social listada, o utilizador tem acesso a detalhes sobre os tipos de dados que podem ser partilhados, como informações de perfil, publicações, contactos, interesses e qualquer outra informação adicional relevante. Para um estado inicial da aplicação, apenas foi utilizado o *Facebook*.

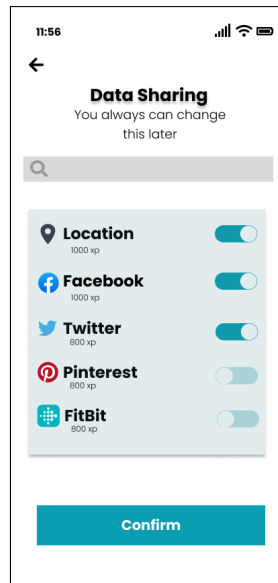


Figura 31: Data Sharing

Através deste ecrã foi realizada a implementação da extração dos dados, com o uso da *API Facebook*, onde é possível aceder a um conjunto dos dados, através das permissões que o utilizador fornece. Numa primeira instância apenas foram recolhidos dados de *login*, sendo estes: a foto de perfil do utilizador, o *access token*, para validar a autenticação, e nome do utilizador. Por meio desta funcionalidade, podem obter-se outras informações, como preferências do utilizador, grupos aos quais ele pertence e outros detalhes mais específicos. No entanto, para aceder a estes dados adicionais, são necessárias permissões avançadas no *Meta for Developer* que envolvem registar a aplicação como um negócio. Embora isto tenha sido inicialmente um obstáculo para prosseguir com a extração de dados, logo foi superado ao utilizar dados fictícios para o desenvolvimento.

Na Figura 32, é apresentado o pedido de permissão efetuado ao utilizador através do *Expo* e depois da sua confirmação, o *toggle* é acionado, demonstrando assim que o utilizador confirmou a partilha de dados através do *facebook*.

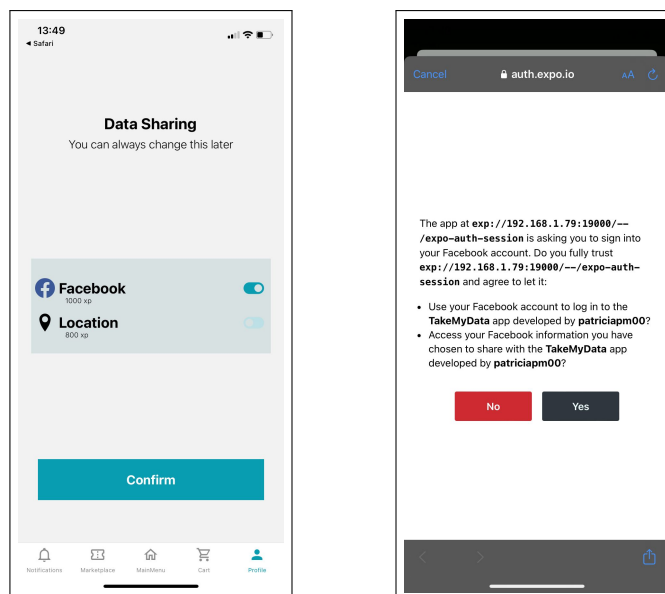


Figura 32: Ecrã da aplicação ainda em fase de desenvolvimento e o pedido de permissão ao *Expo*

O ecrã do Menu principal também foi desenvolvido, embora tenha sido, neste caso, implementada apenas a proposta do seu *layout*, com o principal objetivo de apresentar ao utilizador as campanhas agrupadas por categorias e localização, como se pode visualizar na Figura 33. Além disso, no menu é apresentado o saldo atual do utilizador e as opções de navegação entre os diferentes ecrãs. Na secção “*Find By Category*” as campanhas devem estar agrupadas pelos diferentes “domínios” identificados na recolha de dados. A secção “*Offers near you*” deve ser alterada consoante a localização do utilizador. Ao contrário da outra secção, aqui deve ser apresentada a campanha com informações detalhadas, como o nome da loja, detalhes da campanha e preço. Neste caso, como a campanha não é direcionada especificamente ao utilizador, ele deve “pagar”, utilizando os *tokens* oferecidos pela aplicação, caso queira beneficiar da oferta.

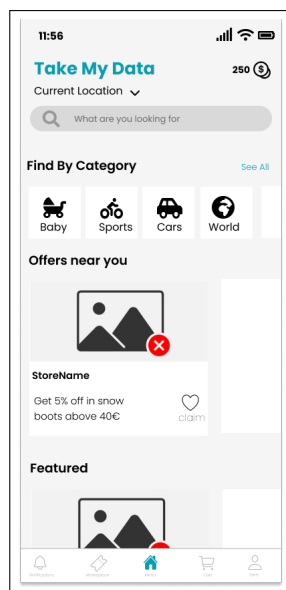


Figura 33: Menu Principal

Para o desenvolvimento do ecrã foi criado a *bottom bar*, para a navegação entre ecrãs. Para isso, foi utilizada a biblioteca *React Navigation*, que serve para facilitar o desenvolvimento de toda a navegação e criação de rotas em aplicações *Expo* e *React Native*.



Figura 34: Barra de baixo da aplicação

Tendo em conta a funcionalidade de navegação de ecrãs já implementada, foi desenvolvido o ecrã das notificações (ver Figura 35). Este ecrã de notificações vai permitir ao retalhista informar os utilizadores sobre as campanhas já devidamente processadas e específicas para cada utilizador. Para tal foi utilizada a funcionalidade de mensagens do *firebase* – *FCM*, para fornecer a capacidade de envio de notificações. Esta integração com o *Firebase* permite que as notificações sejam entregues de forma eficiente e, em tempo real, garantindo que os utilizadores recebem informações relevantes e atualizadas sobre as campanhas e promoções disponíveis.

Foi também proposto um outro ecrã, que não estava nas *mockups* iniciais, representado na Figura 35. O objetivo é possibilitar a visualização dos detalhes da campanha, quando o utilizador clica na notificação associada à campanha. Ou seja, esta janela proporciona mais informação ao utilizador sobre a notificação que recebeu.

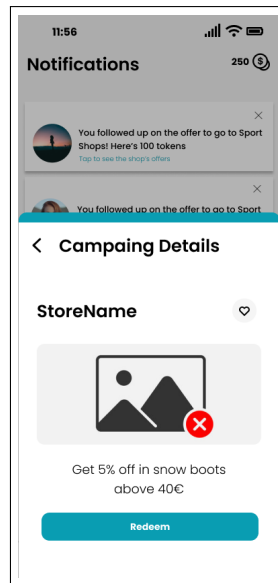


Figura 35: Ecrã com os detalhes de uma campanha

Arquitetura da API

Para sustentar toda a aplicação e *backend*, foi criado uma *API* e outros serviços adicionais, sempre utilizando a *framework Nest.js*, já apresentada na Secção 4.4. Em relação à *API*, esta seguiu a arquitetura sugerida pela própria documentação do *Nest*.

O uso do *Nest* facilitou a organização, devido à funcionalidade de gerar pastas automáticas. Através do uso do gerador, foi possível criar uma pasta para a entidade “*user-info*”, juntamente com os módulos e controladores necessários para a sua implementação. O *Nest*, tem a vantagem de ser compatível com a abordagem de desenvolvimento baseada em microsserviços. o termo “microsserviços” refere-se a uma abordagem de arquitetura de software na qual uma aplicação é dividida em pequenos serviços independentes que se comunicam entre si. Na Figura 36 pode ver-se a arquitetura geral.

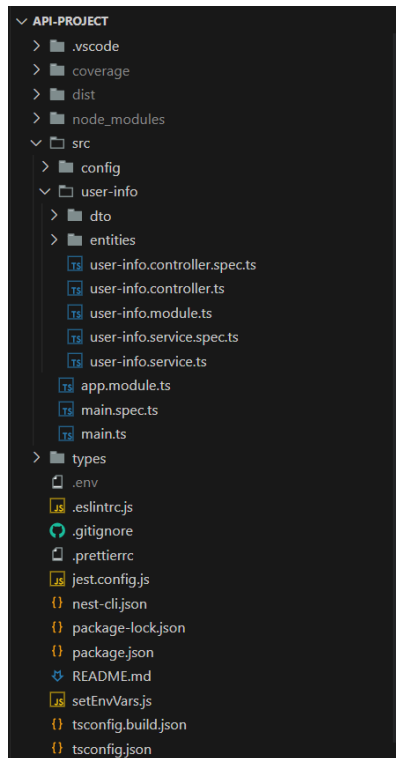


Figura 36: Arquitetura de pastas API

Com a utilização da *framework Nest*, foi possível utilizar uma camada de transporte diferente do *HTTP*. Isto significa que, em vez de usar as comunicações tradicionais baseadas em *HTTP*, como o acesso de um site através do navegador, esta funcionalidade permite que o sistema utilize uma abordagem de comunicação diferente, que envolve a utilização de protocolos de comunicação mais personalizados, adequados às necessidades específicas da aplicação.

Estas camadas, denominam-se de transportadores, que são responsáveis pela transmissão de mensagens entre diferentes instâncias de microsserviços, baseados em eventos. O *Nest* abstrai os detalhes de implementação de cada transportador através de uma *interface* para mensagens baseadas em solicitação-resposta e em eventos. Isto facilita a alternância de uma camada de transporte para outra – por exemplo, para aproveitar os recursos específicos de confiabilidade ou desempenho de uma camada de transporte específica – sem afetar o código da aplicação NestJS (2023).

Neste caso, em específico, foi utilizado o pacote “*nestjs-google-pubsub-microservice*” para configuração do transportador (*Google Pub/Sub*) e estabelecer a comunicação entre as instâncias de microsserviço através de eventos, ou seja, disputadas por algo.

5.3.2 Desenvolvimento da API

Um dos desafios deste projeto, foi estabelecer um fluxo contínuo de dados, de uma forma segura, capaz de escalar à medida que a aplicação evolui. E por isso, foi escolhido o uso das tecnologias já referidas anteriormente na Secção 3.1.

Assim, foi verificado que para realizar uma extração dos dados do *Facebook* com sucesso era necessário garantir um *token* de acesso do utilizador sempre válido. A *API*, mencionada anteriormente, teve como objetivo principal efetuar uma atualização do *token* de acesso através da *API* do *Facebook*. Para alcançar esta finalidade, foi necessário, primeiramente, implementar um sistema de publicação de fila no tópico do *Pub/Sub*, a ser detalhado na Secção 5.3.4. Nesta componente da aplicação é utilizada a funcionalidade de *subscription* do *Pub/Sub* para aceder aos dados publicados na fila do tópico. A *subscription* permite que a *API* receba as informações do *access token* sempre que são publicadas na fila. Desta forma, a aplicação pode obter os *tokens* de acesso dos utilizadores de forma assíncrona e segura.

Após receber os *tokens* de acesso através da *subscription*, a *API* realiza a verificação da validade de cada *token*. Esta verificação é essencial para garantir que o *token* ainda é válido e não foi expirado. Caso o *token* seja considerado inválido, a aplicação pode tomar as medidas necessárias, como solicitar um *refresh token* ao *Facebook*.

Assim, através de um *endpoint* disponibilizado pelo *facebook*, denominado *debug token*, é permitido fazer uma validação do *token*. Este *endpoint* fornece metadados sobre um *token* de acesso específico, incluindo informações sobre o utilizador associado, a validade do *token*, a data de expiração e as permissões concedidas à aplicação. Isto é útil para depurar problemas com grandes conjuntos de *tokens* de acesso. As Figuras 37 e 38 ilustram a chamada a esse *endpoint* através do *Postman* e os dados recebidos.

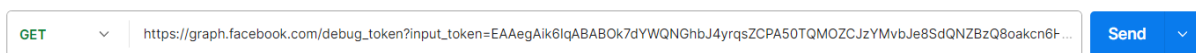


Figura 37: Capturar dados através do endpoint *Debug Token*

```

1  |
2  |   "data": {
3  |     "app_id": "2146255979029152",
4  |     "type": "USER",
5  |     "application": "Take My Data",
6  |     "data_access_expires_at": 1696113392,
7  |     "expires_at": 1694773064,
8  |     "is_valid": true,
9  |     "issued_at": 1688336114,
10 |     "scopes": [
11 |       "user_birthday",
12 |       "user_hometown",
13 |       "user_friends",
14 |       "user_gender",
15 |       "user_age_range",
16 |       "gaming_user_locale",
17 |       "public_profile"
18 |     ],
19 |     "user_id": "6468090409944644"
20 |   }
21 |

```

Figura 38: Dados recebidos através do endpoint *Debug Token*

Após esta verificação, se necessário, é iniciado o processo de *refresh token*. O *refresh token* é responsável por solicitar um novo *token* de acesso de longa duração ao *Facebook*, que é alcançado solicitando a troca do *token* de curta duração por um *token* de longa duração, como se pode ver na Figura 39.

```

async refreshToken(accessToken: string) {
  try {
    const response = await API.get<FacebookResponseDto>(`/v14.0/oauth/access_token?grant_type=fb_exchange_token&client_id=${appId}&client_secret=${appSecret}&fb_exchange_token=${accessToken}&redirect_uri=${redirectUri}`);
    return response;
  } catch (error) {
    console.error('Erro ao obter o token de atualização:', error);
  }
}

```

Figura 39: Lógica desenvolvida para executar um refresh Token à API do Facebook

Quando o *token* de acesso do utilizador está próximo de expirar ou já expirou, é vital obter um novo *token* válido sem a necessidade de pedir novamente as credenciais do utilizador. Para fazer isso, a *API* realiza uma chamada ao *endpoint* do *Facebook*, utilizando as credenciais adequadas. Se a autenticação for bem-sucedida, um *token* de acesso de longa duração é fornecido. Esse *token* é então atualizado nos registos do utilizador no sistema, permitindo que ele continue a utilizar a aplicação sem a necessidade de efetuar um novo *login*.

5.3.3 Firebase

Foi escolhido o *Firebase* para armazenar informações fundamentais do *login* dos utilizadores, incluindo o *token* de acesso.

A tabela “*user*” é especialmente projetada para guardar informações básicas sobre os utilizadores, garantindo a singularidade dos campos “*id*” e “*access token*” para evitar duplicação de utilizadores. Na

Figura 40 podemos ver a modelação realizada para os dados extraídos.

User	
* *	_id { Objectid }
* *	access_token { String }
*	device_token { String }
*	name { String }
*	picture { String }

Figura 40: Modelação dos dados para a Cloud Firestore

O uso da *Cloud Firestore* como base de dados, foi pensado tendo em conta as suas vantagens para a aplicação. Primeiramente, o *Firestore* é altamente escalável, permitindo que a aplicação cresça sem problemas à medida que o número de utilizadores e dados aumenta. Isto é essencial para garantir que a aplicação possa acomodar um grande número de utilizadores, sem comprometer o desempenho.

A Figura 41 faz referência a um utilizador que concedeu acesso aos seus dados de *login*. É importante salientar que, como a aplicação ainda se encontra em fase de desenvolvimento, foi utilizado o meu próprio perfil, visto que apenas é possível efetuar *login* para utilizadores com contas associadas ao *Meta for Developers*, inscritos na aplicação.

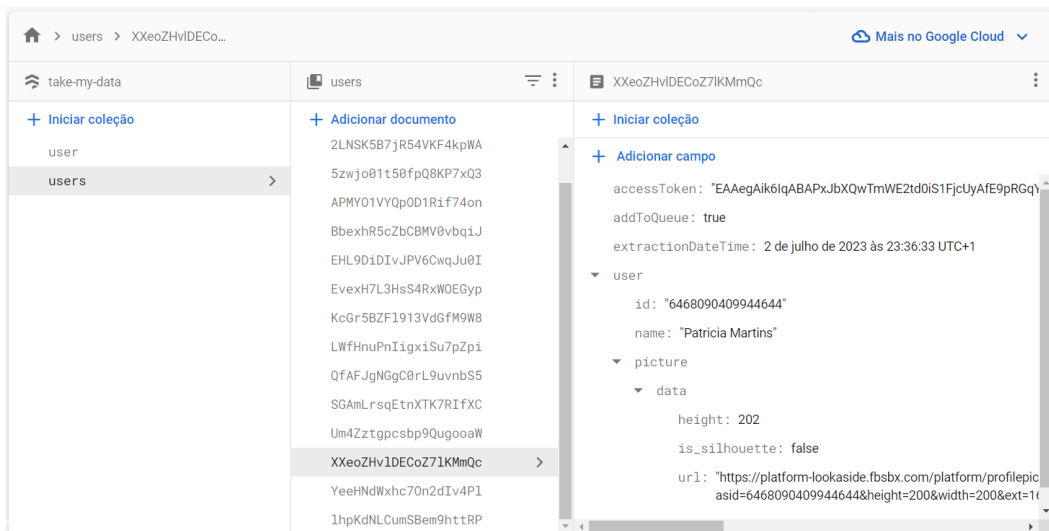


Figura 41: Firestore

5.3.4 Pub/Sub

De forma a dar continuidade ao fluxo de dados, após estes serem extraídos e armazenados no *Firestore*, como mencionado na Secção 5.3.3, é necessário criar uma fila de espera, possibilitando que estes sejam publicados através de um tópico no *Pub/Sub*. Este recurso foi cuidadosamente pensado e projetado para garantir a eficiência e escalabilidade do sistema. A adoção deste método foi possível através da criação de um método *Post*, com a funcionalidade de extrair utilizadores de uma fonte de dados (no caso, a coleção “*users*” do *Firestore*) e, em seguida, colocar esses utilizadores modificados numa fila de espera para processamento adicional.

Na Figura 42, está representada a função “*publish*” que é responsável por enviar mensagens no formato *Json* para um tópico do *Google Cloud Pub/Sub*, e é projetada para lidar com vários utilizadores contidos no “*messageData.users*”.

```
async publish(messageData: any) {
  const topicNameOrId = "projects/bpjoxmv-takemydata/topics/ETL_CHANNEL"

  const pubsub = new PubSub();

  const topic = pubsub.topic(topicNameOrId);

  for (const user of messageData.users) {
    await topic.publishJSON(user);
  }

  return { value: "Mensagens enviadas com sucesso!" };
}
```

Figura 42: Serviço “Publish”

Foi também realizado o serviço para recolher os dados. Neste caso, o método “*addColumnToDocument*”, representado na Figura 43, é responsável por recolher os documentos da coleção do *firestore*, limitar o número de documentos processados, enviar os dados para o *Pub/Sub* e atualizar os documentos, com a informação que estes foram inseridos ao tópico.

```

async *addColumnToDocument(users: QueryDocumentSnapshot<DocumentData>[]) {
  const queryLimit = Number(process.env.QUERY_LIMIT);

  const limitOfUsers = query(
    collection(db, "users"),
    orderBy("extractionDateTime", "desc"),
    limit(queryLimit)
  );
  const querySnapshot = await getDocs(limitOfUsers);

  const limitedUsers = querySnapshot.docs.slice(0, queryLimit);
  const userData = limitedUsers.map((user) => user.data());
  const messageData = { users: userData };

  await this.publish(messageData);

  for await (const user of limitedUsers) {
    const userId = user.id;
    const userRef = doc(db, "users", userId);

    await updateDoc(userRef, { addToQueue: true });

    yield `Usuário ${user.id} publicado na fila com sucesso!`;
  }
  return;
}

```

Figura 43: Serviço “addColumnToDocument”

Por fim, é realizada a gestão desta informação no *controller*, que executa o *Post* como se pode ver na Figura 44. Atualmente este *Post* é acionado por o *Postman*, através de uma solicitação ao *endpoint* “*publish*” , mas mais tarde vai ser acionado através do *Google Schedule*.

```

@Controller("user")
export class UserController {
  constructor(private readonly userService: UserService) {}

  @Post("publish")
  async publishUsersToQueue() {
    const usersSnapshot = await getDocs(query(collection(db, "users")));
    const users = usersSnapshot.docs;

    const response = this.userService.addColumnToDocument(users);

    return response.next();
  }
}

```

Figura 44: Método Post

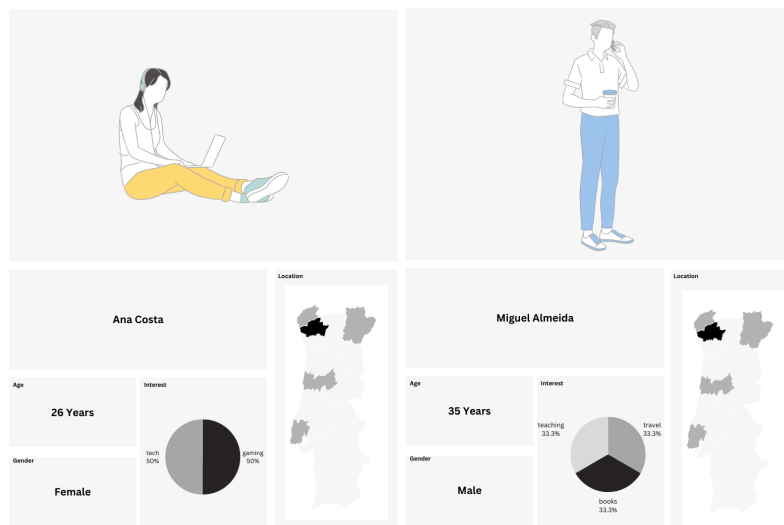
5.3.5 Big Query

Para dar continuidade ao processo da aplicação, optou-se por utilizar o *Big Query* como um sistema de armazenamento de dados. Como mencionado na Secção 5.3.1, quando, no ecrã *Data Sharing*, ocorre um impedimento na extração dos dados do *facebook*, e, para possibilitar uma análise mais realista e

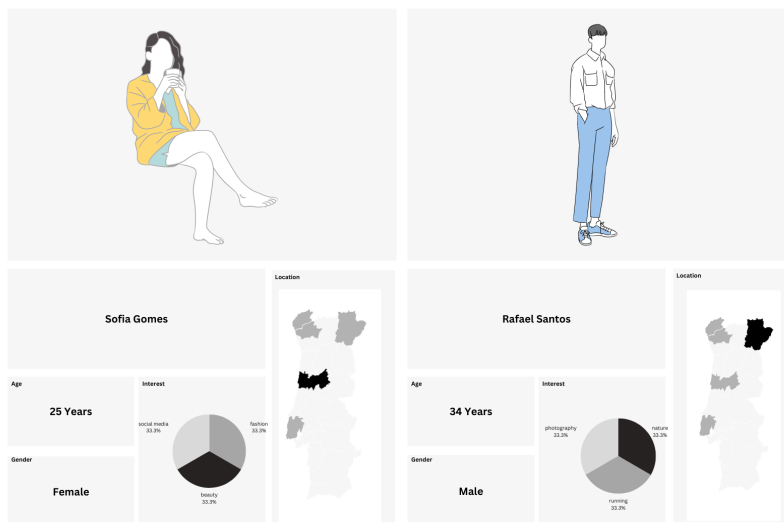
criar uma tabela de segmentação com os perfis dos clientes registados, foram gerados dados fictícios para povoar a estrutura de dados previamente estabelecida em 4.5.

A fim de garantir que os dados fossem coerentes, foram criados perfis de utilizadores, proporcionando uma representação mais fiel à realidade. Essa abordagem permitiu uma análise mais precisa e a realização de testes em cenários mais próximos de situações reais.

Neste sentido, foram feitas “*personas*” de forma a sustentar alguns conteúdos publicados nas redes sociais. Na Figura 45, apresentam-se alguns exemplos.



(a)



(b)

Figura 45: Exemplo de “Personas”

Para segmentar os clientes foi adicionado, como já referido, dados fictícios através de um documento *Json* diretamente no *Big Query*. No futuro pretende-se utilizar uma ferramenta de *machine learning* que faça o reconhecimento automático dos utilizadores, consoante as suas preferências, comentários, *posts*, localização e outros critérios, e com isto ser criado uma tabela de segmentação para campanhas específicas. Atualmente, são utilizadas as *personas* elaboradas, e a partir destas, são agrupadas em categorias alguns campos de interesse. Com isso em mente, conseguiu-se segmentar os clientes, considerando faixas de idade, género, interesses e localização. Esta abordagem proporcionou um método eficaz de direcionar as campanhas.

Resumindo, tendo em conta a tabela da segmentação povoada, é possível direcionar campanhas aos utilizadores inscritos na aplicação. As Figuras 46 e 47 apresentam a visualização das tabelas resultantes, extraídas diretamente do *BigQuery*.

Row	segmentation_id	age_min	age_max	gender	location	topic
1	1	20	35	Male	Viana do Castelo, Portugal	sports
						fitness
2	2	25	40	Male	Bragança, Portugal	travel
						tech
3	3	18	50	Female	Lisboa, Portugal	fitness
						health

Figura 46: Tabela segmentação

Row	user_id	segmentation_id	topic
1	9	1	sports
			betting
2	7	2	nature
			fitness
3	6	3	tech
			music
4	10	3	fitness
			health

Figura 47: Tabela UserSegmentation

A Figura 46 ilustra a tabela de segmentação, enquanto a Figura 47 mostra a tabela “*UserSegmentation*”, resultante de um *join* entre a tabela “*Segmentation*” com a tabela “*User*”. Com base na tabela apresentada, o processo subsequente para o retalhista é simplificado e direto.

5.4 Criação de uma campanha

Nesta secção, é explorado o processo completo de criação de uma campanha e como isso foi possível através das componentes apresentadas.

Como mencionado anteriormente, o *Big Query*, possui uma tabela de segmentação. Esta é povoada consoante campos que permitem criar uma campanha, tendo em conta características como o género, interesse e localização. Depois das campanhas estarem configuradas na tabela de segmentação, os retalhistas só precisam escolher o seu público-alvo. O sistema utiliza os critérios da tabela de segmentação para seleccionar utilizadores que se ajustam às características desejadas da campanha. Os retalhistas podem seleccionar o seu público-alvo através de uma página *web*, onde têm a possibilidade de definir o título e associar uma imagem à campanha. Na Figura 48, pode ver-se o *layout* criado para os retalhistas.

The image shows a web interface for creating a campaign. At the top center is the logo 'TAKE MY DATA'. The interface is split into two columns. The left column, titled 'Create new campaign', contains a 'Target Audience' dropdown menu, a 'Campaign Title' text input field, a 'Push Notification Text' text area, and a 'Push Notification Image' image upload area with a 'Drop or browse images here' prompt. A 'Launch Campaign' button is at the bottom. The right column, titled 'Campaign Preview', has three tabs: 'Notification', 'Expanded', and 'Detailed'. The 'Expanded' tab is active. Below the tabs, it says 'Complete all fields to show Preview'.

Figura 48: *Layout* da página web para a criação de uma campanha

Na Figura 49, representam-se alguns exemplos de selecção para o campo público-alvo.


Create new campaign

Target Audience

F ▼

- Fado
- Family
- Fencing
- Fencing
- Fitness
- Food
- Football

Push Notification Image



Drop or browse images here

Launch Campaign

Campaign Preview

Notification

Expanded

Detailed

Complete all fields to
show Preview

Figura 49: Exemplo de seleção para público-alvo

Esta metodologia é fundamental para salvaguardar a segurança dos dados dos utilizadores, uma vez que restringe o acesso dos retalhistas.

Uma vez concluído este processo, os utilizadores que correspondem ao público-alvo recebem notificações com as propostas de campanha já devidamente processadas e filtradas, de acordo com as suas preferências. Isto garante que as campanhas sejam altamente direcionadas e relevantes para cada utilizador. Este nível de personalização é alcançado por meio de um serviço dedicado à entrega de notificações *push* em dispositivos *mobiles*, com base em critérios de segmentação previamente obtidos a partir da base de dados do *Big Query*.

Quando todos os campos, relativos à campanha – título, público-alvo, texto e imagem; estão preenchidos pelos retalhistas, existem três formas de visualização através do sistema de notificações implementado. O primeiro, representa uma notificação na aplicação, representado na Figura 50, com um exemplo de como é apresentado nos diferentes sistemas operativos - *iOS* e *Android*.

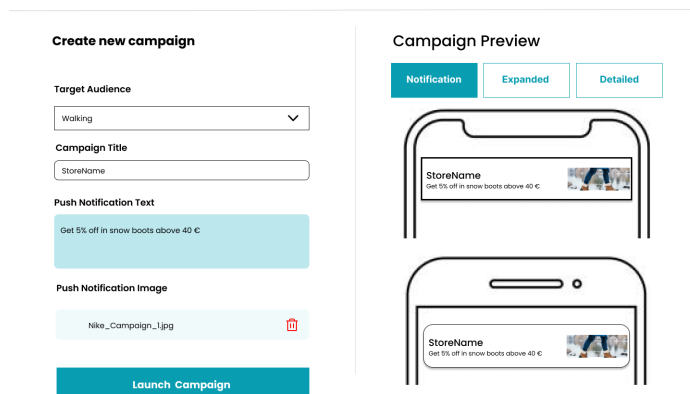


Figura 50: Visualização da campanha na aplicação

A segunda forma de visualização, na Figura 51, diz respeito à barra de notificações do telemóvel, ou seja, quando o utilizador não está a utilizar diretamente a aplicação e tem as permissões de notificação ativadas.

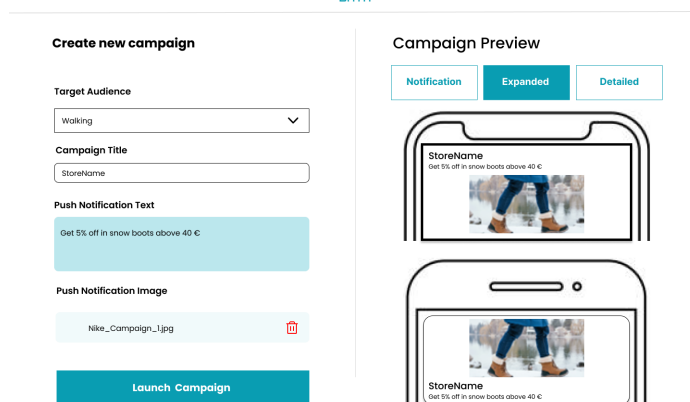


Figura 51: Visualização da campanha na barra de notificações

Por fim, o terceiro inclui a visualização dos detalhes quando o utilizador clica na notificação dentro da aplicação.

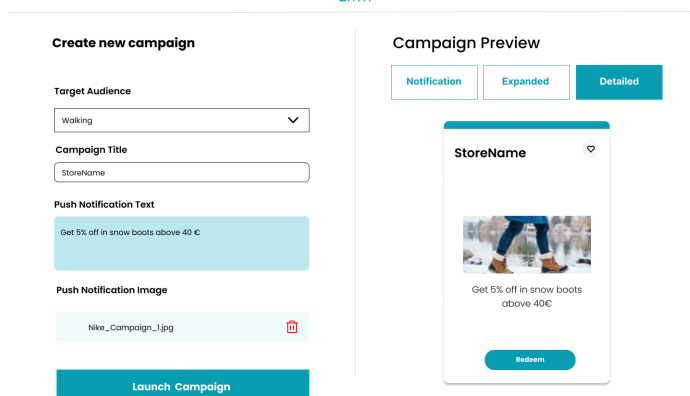


Figura 52: Visualização da campanha em detalhe

5.5 Conclusão

Este capítulo apresenta uma visão detalhada do processo de desenvolvimento da aplicação e do sistema de gestão de campanhas. Nestes processos foram abordados diversos aspetos, desde a criação da *interface* do utilizador até à integração de serviços externos.

O desenvolvimento seguiu uma abordagem sempre focada numa *interface* intuitiva e eficiente, utilizando tecnologias para garantir compatibilidade de multiplataforma. Foram elaboradas diversas telas, desde o registo de conta até a navegação principal da aplicação. É importante destacar que este foi um esforço de equipa, e cada membro desempenhou um papel fundamental. A autora desta dissertação, participou no desenvolvimento do *layout* da tela de *Data Sharing*, mesmo que a lógica da extração dos dados do *login* do *Facebook* tenha sido implementada por outros colegas, como era um passo importante para a continuidade do fluxo não poderia deixar de o mencionar.

A implementação seguiu as práticas recomendadas pelo *Expo* e *React Native* na arquitetura do projeto, assegurando a sua organização e escalabilidade. A integração de serviços externos, como o *Facebook* para extração de dados de *login*, foi planeada para assegurar o fluxo de dados contínuo. A utilização do *Firebase*, para armazenar informações de *login* dos utilizadores, e do *Google Cloud Pub/Sub*, para criar uma fila de mensagens, garantiu a segurança e escalabilidade da aplicação.

O *Big Query* foi utilizado como sistema de armazenamento de dados, permitindo a segmentação eficaz de clientes com base em critérios específicos, possibilitando a criação de campanhas altamente direcionadas e relevantes para cada utilizador.

No processo de criação de campanhas, os retalhistas têm a capacidade de escolher o seu público-alvo, com base em critérios específicos, garantindo que as campanhas sejam entregues apenas aos utilizadores que correspondem às características desejadas.

Em resumo, este capítulo apresenta uma visão abrangente do trabalho realizado com o objetivo final da aplicação proposta.

6 UTILIZAÇÃO DA METODOLOGIA SCRUM

6.1 Introdução

Neste capítulo, é apresentado, em detalhe, a implementação da metodologia *Scrum* no projeto. Além disso, são apresentados os principais eventos do *Scrum*, como *Sprint Planning*, *Daily Stand-up*, *Sprint Review* e *Sprint Retrospective*, aplicados no projeto.

A adoção do *Scrum* como metodologia de desenvolvimento, traz uma série de benefícios, conferindo uma estrutura bem definida e eficaz. Faz-se notar, que esta abordagem foi direcionada pela empresa, acompanhada de uma explicação detalhada sobre o facto de ser a escolha ideal para o projeto, em comparação com o método tradicional em cascata. Além disso, a equipa recebeu uma formação abrangente sobre as práticas do *Scrum*.

6.2 Implementação do Scrum

A implementação do *Scrum* é realizada seguindo os princípios essenciais da metodologia. Na Tabela 1 são detalhados os principais aspetos da implementação.

Tabela 1: Detalhes da Implementação do *Scrum* no projeto

Aspeto	Detalhes
Metodologia	<i>Scrum</i>
Papéis	- <i>Product Owner (PO)</i> - <i>Scrum Master</i> - Equipa de Desenvolvimento
Duração do <i>Sprint</i>	2 semanas
Reuniões	- Reunião de Planeamento de <i>Sprint</i> - Reunião Diária (<i>Daily Scrum</i>) - Revisão de <i>Sprint</i> (<i>Sprint Review</i>) - Retrospectiva de <i>Sprint</i> (<i>Retrospective</i>)
<i>Backlog</i>	- Definido e priorizado pelo PO - Incluindo <i>UserStories</i> e tarefas

Além das vantagens gerais do *Scrum*, como abordagem iterativa e adaptativa, salientam-se a comunicação melhorada e a entrega incremental. A metodologia também introduz papéis fundamentais que desempenham um papel vital no sucesso do projeto. Um destaque notável reside no papel do *PO*, que tem uma atuação excepcional devido à natureza interna do projeto, que não está diretamente ligado a um cliente externo.

Dentro deste contexto, o *PO* desempenha um papel de dualidade, atuando não apenas como o elo entre as necessidades internas, mas também assumindo o papel crucial de *PO*. Esta abordagem tem por objetivo garantir que tanto as necessidades internas sejam satisfeitas, quanto as funcionalidades do produto sejam enriquecidas, agregando valor ao *backlog* e direcionando o processo de desenvolvimento, para cumprir tanto com os objetivos internos quanto os organizacionais. Todos os restantes papéis fundamentais no *scrum* são desempenhados seguindo o padrão da metodologia. Ou seja, existe o *Scrum Master (SM)* que está encarregue de garantir que todas as práticas sejam seguidas, removendo obstáculos e provendo um ambiente produtivo. A presença do *SM* desempenha um papel vital na resolução eficiente de impedimentos que surgem ao longo do projeto.

Nas Tabelas 2 – 3 é apresentado um resumo da metodologia adotada e os seus propósitos.

De uma forma geral, o *Sprint Planning* ocorre no início de cada *sprint*. Nessa reunião, após um refinamento das tarefas por parte do *PO*, *SM* e do *Tech Lead*, são discutidos os conteúdos das tarefas planeadas para o *sprint* em questão. Desta forma, são atribuídos os *Story Points* a cada tarefa, empregando a sequência de *Fibonacci* como técnica de estimativa.

Além disso, todos os dias é realizada uma reunião de quinze minutos - as *daily*s, na qual são detalhados o progresso das tarefas, são reportados obstáculos e, delineados os passos seguintes. As *daily*s têm como objetivo promover o avanço da equipa de desenvolvimento, mas também garantir uma comunicação contínua com os restantes membros da equipa, com a presença sempre assegurada do *PO*, *SM* e do *Tech Lead*. Depois de cada *Sprint* é realizada uma *Sprint Review*, neste caso em particular os *stakeholders*, eram substituídos pelos *POs*, que validavam o nosso progresso obtido no *sprint*, e verificavam os impedimentos da mesma. Por fim, era realizado uma retrospectiva, onde apontávamos os pontos positivos e aspetos a melhorar e, através dos negativos eram criadas métricas ou ações de melhoria.

Tabela 2: Detalhes dos eventos do *Scrum*

Evento	Participantes	Quando	Duração (sprint de 2 semanas)
<i>Sprint Planning</i>	- <i>Product Owner</i> - <i>Scrum Master</i> - <i>Development Team</i>	Início da <i>Sprint</i>	4 horas
<i>Daily Stand-up</i>	- <i>Development Team</i> - <i>Scrum Master</i>	Horário fixo definido pela equipa	15 minutos
<i>Sprint Review</i>	- <i>Product Owner</i> - <i>Scrum Master</i> - <i>Development Team</i> - <i>Stakeholders</i>	Fim do <i>sprint</i>	2 horas para 1 <i>sprint</i>
<i>Sprint Retrospective</i>	- <i>Scrum Master</i> - <i>Development Team</i>	Depois da <i>Sprint Review</i>	1.5 horas

Tabela 3: Propósitos dos Eventos *Scrum*

Evento	Propósito
<i>Sprint Planning</i>	Planear o trabalho para a próxima <i>sprint</i> .
<i>Daily Stand-up</i>	Partilhar o progresso, discutir obstáculos e planear o dia.
<i>Sprint Review</i>	Demonstrar o trabalho concluído aos <i>stakeholders</i> e obter <i>feedback</i> .
<i>Sprint Retrospective</i>	Refletir sobre a <i>sprint</i> anterior e identificar melhorias.

Na tabela 4, é apresentado as questões essenciais para cada evento no contexto do *Scrum*, visando direcionar as discussões e atividades durante o ciclo de desenvolvimento- *sprint*. Cada evento possui um conjunto específico de perguntas com o propósito de orientar os participantes e garantir que o foco está alinhado com os objetivos daquele evento em particular. Estas perguntas servem de guia para que os eventos corram segundo a metodologia.

Tabela 4: Questões principais nos Eventos

Event	Questions
<i>Sprint Planning</i>	O que pode ser alcançado nesta <i>sprint</i> ? Como será realizado o trabalho?
<i>Daily Stand-up</i>	O que fiz ontem? O que farei hoje? Existem obstáculos no meu caminho?
<i>Sprint Review</i>	O que foi concluído nesta <i>sprint</i> ? O que correu bem? O que poderia ser melhorado?
<i>Sprint Retrospective</i>	O que correu bem na <i>sprint</i> anterior? O que poderia ter sido melhor? Que ações podemos tomar para melhorar?

O projeto global foi dividido em 8 *Sprints*, cada uma com a duração de duas semanas, tirando algumas exceções. Para ser possível este planeamento, como mencionado anteriormente, utilizou-se o *Jira Software* como uma ferramenta de apoio essencial. O quadro (*board*) de trabalho foi segmentado em três etapas principais, antes de considerar uma tarefa como concluída: "Em progresso" (*in progress*), "Testes" (*testing*) e "Aguardar" (*waiting for acceptance*). Esta abordagem facilitou, tanto a organização do trabalho, quanto a coordenação das tarefas no projeto. A divisão em etapas claras tornou o acompanhamento do progresso mais eficiente para toda a equipa.

Na fase "Em progresso", os membros da equipa estão dedicados à implementação ativa das funcionalidades planeadas para cada *sprint*. Esta etapa representa o ponto de partida para cada tarefa, onde a equipa de desenvolvimento trabalha na codificação e no desenvolvimento.

Na etapa de "Testes", coloca-se um forte foco na qualidade e na integridade das funcionalidades que são desenvolvidas. Foi decidido que apenas iriam ser realizados testes unitários para identificar e resolver possíveis problemas, devido à falta de experiência da equipa. Além disso, estabeleceu-se um critério de cobertura de código de 70%, garantindo que a maior parte do código fosse submetido a testes, embora houvesse o comprometimento de tentar realizar uma cobertura de 100%.

Quando uma tarefa alcançava a fase final, era submetida a uma verificação minuciosa pelo *Tech Lead*.

Esta verificação incluía um *desk check*, realizado com a participação de toda a equipa de desenvolvimento, o *PO* e o *Tech Lead*. No caso de tarefas que dependiam principalmente da codificação, também era realizada uma Revisão de *PR* no repositório da *Azure DevOps* para garantir que todos os critérios fossem atendidos.

6.3 Divisão dos Sprints

Como mencionado anteriormente, foram realizados 8 *Sprints* no total. No primeiro *sprint*, conforme apresentado na Tabela 5, o foco consiste em tarefas essenciais para o desenvolvimento inicial da aplicação. A primeira tarefa era criar o *design* da *User Experience (UX)* e analisar os dados que poderiam ser extraídos das redes sociais. Esta etapa era crucial para definir a direção do projeto. Foi, também necessário criar documentação para o fluxo de trabalho no repositório, definir estratégias de teste e especificar a cobertura necessária para que as tarefas fossem consideradas concluídas.

Tabela 5: Atividades e *Story Points* do *Sprint 1*

Sprint 1 [31 Mar - 19 Abr]	
Descrição	Story Points
Criação do desenho <i>UX</i> - Navegação entre ecrãs	5
Criação das <i>Mockups</i> em <i>Figma</i> Aprovação das <i>mockups/flows</i>	
Análise dos dados fornecidos pelas <i>API</i> das redes sociais	5
Identificar os dados possíveis de extrair na <i>API</i> do <i>Facebook</i> Identificar os dados possíveis de extrair na <i>API</i> do <i>Twitter</i> Identificar os dados possíveis de extrair na <i>API</i> do <i>FitBit</i> Organização dos dados identificados em cada <i>API</i>	
Fluxo de trabalho para o repositório	5
Resolução do <i>issue</i> de autenticação com o <i>InnerSource</i> Criar documentação do fluxo de trabalho no repositório Criar autenticação no <i>Azure DevOps</i>	

No segundo *Sprint*, conforme apresentado na Tabela 6, deu-se a continuação de algumas das tarefas pendentes do *Sprint* anterior e foram iniciadas outras. Estas tarefas, como estavam dependentes de

aprovação e, caso necessário, apenas iriam necessitar de algumas alterações, foram adicionadas novas. Assim, continuou-se com o desenvolvimento do fluxo de trabalho do repositório, resolvendo problemas de autenticação e criando documentação adicional. Ao mesmo tempo, era realizado a análise dos dados fornecidos pelas *API* das redes sociais, identificando o que poderia ser extraído das *API* do *Facebook*, *Twitter* e *FitBit*. Como novas tarefas, foi iniciada a criação da (*code base*), foram configurados os testes unitários, definidos os padrões de desenvolvimento e o ficheiro *README.md* do projeto. Foi trabalhada também a arquitetura e o esquema geral do projeto, bem como o *layout* do ecrã relacionado com a partilha de dados. Adicionalmente, foi criada a configuração inicial do *Firebase* e deu-se o desenvolvimento do código para extrair dados e publicar na fila de espera. Esta fase também envolveu a autenticação com o *Facebook* e o uso do *EXPO* para autenticação, visto que foi reportado um impedimento, e foi necessário fazer uma pesquisa adicional e procurar uma solução, daí a sua pontuação mais elevada. Tendo em conta a procura realizada sobre a análise dos dados extraídos pela redes sociais, foi decidido o foco na *API* do *Facebook*.

Tabela 6: Atividades e *Story Points* do *Sprint 2*

Sprint 2 [20 Abr - 11 Mai]	
Descrição	Story Points
Fluxo de trabalho para o repositório	5
Resolução do <i>issue</i> de autenticação com o <i>InnerSource</i> Criar documentação do fluxo de trabalho no repositório Criar autenticação no <i>Azure DevOps</i>	
Análise dos dados fornecidos pelas APIs das redes sociais	5
Identificar os dados possíveis de extrair na API do <i>Facebook</i> Identificar os dados possíveis de extrair na API do <i>Twitter</i> Identificar os dados possíveis de extrair na API do <i>FitBit</i> Organização dos dados identificados em cada API	
Criar <i>code base</i>	5
Configuração dos testes unitários Configuração dos padrões de desenvolvimento Criação do README.md do projeto	
Arquitetura/Esquema do projeto	2
Desenvolvimento do <i>Layout</i> do ecrã- <i>Data sharing</i>	3
Criação da configuração do <i>Firebase</i>	3
Criar <i>code base</i> para a recolha de dados e publicação na fila de atualização	5
Definição de arquitetura Configuração dos testes unitários Criar documentação de tecnologias utilizadas para desenvolvimento	
Autenticação com <i>Facebook</i> e <i>EXPO</i>	8

No *Sprint 3* foram adicionados novos membros à equipa, de forma a reforçar o apoio técnico e, os mesmos mantiveram-se até ao *sprint* quatro. Durante o terceiro *sprint*, ver Tabela 7, ocorreram várias atividades cruciais no desenvolvimento do projeto. Salientam-se o desenvolvimento do *layout* do ecrã de

Data Sharing, a arquitetura e esquema geral do projeto, bem como a criação da base de código (*code base*), que envolveu a configuração dos testes unitários, a definição dos padrões de desenvolvimento que iriam ser seguidos e a criação do ficheiro *README.md* do projeto para documentação. Além disso, foi implementada a capacidade de armazenar os dados do utilizador no *Firestore*, foi feita a modelação dos dados e deu-se continuidade ao desenvolvimento do *layout* da *interface* do utilizador, desta vez a tela de notificações. Por fim, foi finalizado o *layout* do menu principal da aplicação. Foram propostas configurações mais avançadas, como a configuração do *Pub/Sub* na *cloud*, a implementação de um mecanismo de *Refresh Token* para manter a autenticação e a configuração da *subscription*, a qual não foi iniciada neste *Sprint*.

Tabela 7: Atividades e *Story Points* do *Sprint 3*

Sprint 3 [12 Mai - 24 Mai]	
Descrição	Story Points
Desenvolvimento do <i>Layout</i> do ecrã - <i>Data sharing</i>	3
Arquitetura/Esquema do projeto	2
Criar <i>code base</i>	5
Configuração dos testes unitários Configuração dos padrões de desenvolvimento Criação do <i>README.md</i> do projeto	
Como utilizador quero autenticar a conta das redes sociais, após aprovar a partilha dos dados	5
Guardar dados do utilizador no <i>Firestore</i>	3
Extração dos dados do utilizador do <i>Facebook</i>	8
Configurar <i>Pub/Sub</i> na <i>cloud</i> Fazer <i>Refresh Token</i> Configurar <i>subscription</i> Criar <i>mock</i> da base de dados	
Modelação dos dados na Base de Dados	3
Modelação dos dados no <i>Firestore</i> Modelação dos dados do <i>Facebook</i> para o <i>BigQuery</i>	
Desenvolvimento do <i>Layout</i> do ecrã - <i>Notifications</i>	3
Desenvolvimento do <i>Layout</i> do ecrã - <i>Main Menu</i>	3

Na Tabela 8, está representado o *Sprint 4*, cujo foco principal foi a implementação do *Pub/Sub* no *GCP*, e para isso foi chamado um especialista nessa área, com o objetivo de nos dar apoio técnico.

Inicialmente era necessário criar dois tópicos no *GCP*. Um tópico é um canal de comunicação no qual quem envia os dados publicam mensagens e os consumidores de dados inscrevem-se para receber essas mensagens. Para criar estes tópicos, foram criadas duas contas de serviço (*service accounts*) para duas *Cloud Function* diferentes. Foi cedida permissão de escrita à primeira conta de serviço para o primeiro tópico. Com isso, a *service account* podia interagir com o *Firestore* e publicar mensagens no tópico. Seguidamente, era necessário criar a *Cloud Function* que é acionada pelo *Cloud Scheduler*. Isto é, quando a função é acionada, ela vai executar o código que foi fornecido para inserir os dados na fila especificada. Depois era necessário criar outra *service account* e conceder o papel de administrador ao *Cloud Scheduler* para esta conta. Desta forma, permitia que a *service account* fosse usada como invocador no *Cloud Scheduler* para acionar a *Cloud Function* de acordo com o cronograma definido.

Esta etapa foi particularmente desafiadora, embora tenha havido atrasos, devido a problemas de permissões e configurações na *GCP*. Como resultado, a implementação do *Pub/Sub* no *GCP* não foi concluída neste *Sprint* e essa parte do trabalho foi direcionada a um especialista para resolução, apenas foi pedido a implementação da fila espera.

Tabela 8: Atividades e *Story Points* do *Sprint 4*

Sprint 4 [30 Maio - 13 Junho]	
Descrição	Story Points
Guardar dados do utilizador no <i>Firestore</i>	3
Desenvolvimento do <i>Layout</i> do ecrã - <i>Main Menu</i>	3
Desenvolvimento do <i>Layout</i> do ecrã - <i>Notifications</i>	3
Modelação dos dados na Base de Dados	3
Modelação dos dados no <i>Firestore</i> Modelação dos dados do <i>Facebook</i> para o <i>BigQuery</i>	
Extração dos dados do utilizador do <i>Facebook</i>	8
Configurar <i>Pub/Sub</i> na <i>cloud</i> Fazer <i>Refresh Token</i> Configurar <i>subscription</i> Criar <i>mock</i> da base de dados	
Como utilizador quero autenticar a conta das redes sociais, após aprovar a partilha dos dados	5
Recolher dados do <i>Firestore</i> e publicar na fila	5
Configurar <i>SDK Firebase</i> para aceder <i>Firestore</i> Criar método <i>post</i>	

No *Sprint 5*, Tabela 9, a equipa foi impactada por uma série de desafios que tiveram um efeito cascata no projeto, como um todo. Estes desafios incluíram atrasos acumulados dos *sprints* anteriores e mudanças na equipa que coincidiram com a finalização do estágio dos restantes membros. Além disso, foram enfrentados obstáculos na extração de dados do utilizador do *Facebook*, o que resultou na repetição das mesmas atividades nos *Sprints* cinco e seis.

Apesar desses desafios, a qualidade do projeto foi sempre o principal foco. Este período foi caracterizado por uma abordagem resiliente, onde a autora trabalhou arduamente para garantir que o projeto continuasse a progredir, mesmo diante de adversidades.

Tabela 9: Atividades e *Story Points* do *Sprint 5* e *Sprint 6*

Sprint 5 e 6 [16 Junho - 24 Julho]	
Descrição	Story Points
Guardar dados do utilizador no <i>Firestore</i>	3
Modelação dos dados na Base de Dados	3
Modelação dos dados no <i>Firestore</i> Modelação dos dados do <i>Facebook</i> para o <i>BigQuery</i>	
Extração dos dados do utilizador do <i>Facebook</i>	8
Configurar <i>Pub/Sub</i> na <i>cloud</i> Fazer <i>Refresh Token</i> Configurar <i>subscription</i> Criar <i>mock</i> da base de dados	
Recolher dados do <i>Firestore</i> e publicar na fila	5
Configurar <i>SDK Firebase</i> para aceder <i>Firestore</i> Criar método <i>post</i>	

O *Sprint 7*, começou com novas tarefas, já focadas no fluxo da criação de campanhas. Nesta fase, foram inseridos novos membros, de forma que o projeto desse continuidade aquando a saída da autora da dissertação. Durante esta transição, a responsabilidade foi facilitar a transferência de conhecimento e orientar os novos membros nos primeiros passos do projeto. Desta forma, os novos membros, e a autora iniciaram tarefas, como está representado na Tabela 10. Foi então possível começar tarefas, como a implementação de notificações *push* para dispositivos *Android*, possibilitando que os utilizadores recebessem notificações nos seus dispositivos *Android* de maneira eficiente e eficaz. Além disso, foi desenvolvida a funcionalidade de armazenamento local de notificações no *local storage* do navegador. Isto permitiu que os utilizadores visualizassem notificações na aplicação mesmo após tê-las recebido.

Outro destaque deste *sprint* foi o desenvolvimento do *layout* da página *web* dedicada aos retalhistas, que lhes permitiu configurar e lançar campanhas. Isto envolveu a criação de *wireframes* e a implementação do *layout* utilizando a plataforma *Figma*. A atividade relacionada com o processo de disparo de campanha por parte dos retalhistas envolveu a criação do *backend* necessário para permitir que eles enviassem campanhas por meio de uma *API*. É importante notar que esta tarefa específica não contou com a influência da autora, e, portanto, não foi mencionada nesta dissertação, assim como a

implementação do *layout* da visualização das campanhas na aplicação, onde a contribuição se limitou à projeção do *layout*.

É de notar que a atividade relativa à extração de dados tornou-se uma atividade bloqueada no quadro de tarefas, devido aos obstáculos já mencionados.

Tabela 10: Atividades e *Story Points* do *Sprint 7*

Sprint 7 [4 Agosto - 18 Agosto]	
Descrição	Story Points
Guardar notificações no <i>localStorage</i>	5
Pagina Web para disparo de campanha <i>Retailers</i>	3
Desenvolver <i>wireframe</i> da pagina Implementar <i>layout</i> no <i>figma</i>	
Processo disparo de campanha do <i>retailer</i>	5
Criar <i>backend</i> do <i>retailer</i> Criar método <i>post</i> para receber campanhas Enviar <i>push notification</i>	
Como utilizador visualizar campanhas relevantes para melhor experiência de consumo	3
<i>Layout</i> da página de visualização de campanhas Implementar <i>layout</i> na <i>App</i>	
Extração dos dados	
Implementar <i>Push Notification</i> para <i>Android</i>	5

No *sprint 8*, Tabela 11, a autora da dissertação concentrou-se em incluir atividades que ficaram inacabadas, garantindo que todas as funcionalidades fossem implementadas e refinadas adequadamente. Esta atividade estava relacionada com a funcionalidade do armazenamento local de notificações no *local storage*, que já estava num estágio avançado e só precisava de alguns ajustes finais para ser concluída e aprovada. Além disso, dedicou-se a tarefas como armazenar o *token* do dispositivo no *Firebase*, que é fundamental, pois o *token* do dispositivo é uma parte essencial para a entrega de notificações *push*. Outra tarefa importante foi inserir o *Device Token* no tópico do *Facebook* para posteriormente ser enviado para o *Big Query*. Esta atividade envolveu as componentes

mencionadas na Secção 4.4). Como estas atividades envolviam áreas onde os novos membros da equipa não tinham experiência, a autora assumiu a responsabilidade de garantir que essas tarefas fossem concluídas com sucesso.

O *Terraform* era uma tarefa que ia envolver a formação de um especialista, e por isso, estavam dependentes da sua disponibilidade. Este seria um dos próximos passos fundamentais para gerir as diferentes componentes e para unificar os serviços da *GCP*.

Tabela 11: Atividades e *Story Points* do *Sprint 8*

Sprint 8 [21 Agosto - 31 Agosto]	
Descrição	Story Points
Armazenar <i>token</i> do <i>device</i> no <i>Firebase</i>	2
Inserir <i>Device Token</i> no tópico do <i>Facebook</i>	2
Disparar campanha a partir do <i>Retailer Web (Frontend)</i>	5
<i>Terraform</i>	
Como utilizador quero usar uma campanha introduzida na <i>app</i>	
Guardar notificações no <i>localStorage</i>	5
Processo disparo de campanha do <i>retailer</i>	5
Criar <i>backend</i> do <i>retailer</i> Criar método <i>post</i> para receber campanhas Enviar <i>push notification</i>	

6.4 Conclusão

Neste capítulo, foi detalhada a implementação da metodologia *Scrum* no projeto, enfatizando os desafios enfrentados, especialmente num ambiente caracterizado por mudanças frequentes na equipa de desenvolvimento.

A escolha do *Scrum* como metodologia de desenvolvimento de *software* trouxe uma estrutura bem definida e eficaz para o projeto, apoiada pela formação abrangente fornecida à equipa, sobre as práticas do *Scrum*. Cada *Sprint* foi acompanhado por desafios, como a implementação do *Pub/Sub* no *GCP*, que exigiu assistência especializada. Enfrentaram-se atrasos acumulados e obstáculos na extração de dados do *Facebook*, que resultaram na repetição de atividades em *Sprints* subsequentes.

Apesar desses desafios, a qualidade do projeto sempre foi a prioridade imposta pelo *Tech Lead*, e foi mantida uma abordagem resiliente para garantir que o projeto continuasse a progredir.

7 CONSIDERAÇÕES FINAIS

Neste capítulo, É apresentada uma análise crítica do trabalho realizado e algumas sugestões de trabalho futuro.

7.1 Conclusão

A partilha de dados na, era atual, transformou-se numa ação comum na vida de muito utilizadores. Hoje em dia, as informações pessoais são frequentemente solicitadas em troca de serviços e experiências. No entanto, esta partilha suscitou crescentes preocupações da privacidade e segurança dos dados. À medida que as empresas procuram aceder a informações valiosas, os utilizadores exigem, cada vez mais, transparência e recompensas pela partilha. Esta era a premissa para a aplicação desenvolvida - “*Take My Data*”.

A dissertação realizada na empresa *Accenture* Portugal, concentrou-se no desenvolvimento de uma aplicação *mobile*, que oferece aos utilizadores a possibilidade de gerir a partilha dos dados, de forma voluntária e, em troca, serem recompensados. Ao longo deste trabalho, foi explorada a construção da ideia, o desenvolvimento e a implementação física, utilizando metodologias ágeis baseado no *framework Scrum*. O objetivo era a criação de um *MVP* funcional. Para alcançar este objetivo foram estabelecidos passos cruciais que incluíram a criação da arquitetura da aplicação, o desenvolvimento de *interfaces* de utilizador, a configuração do *backend*, a extração de dados das redes sociais e a criação de campanhas.

É importante observar que os tópicos abaixo discriminados, que indicam o estado de conclusão do projeto, foram todos concluídos com sucesso:

- **Criar a arquitetura da aplicação:** esta etapa envolveu o planeamento da estrutura fundamental da aplicação, definindo como os componentes se relacionam e interagem. Essa fase foi concluída com sucesso, fornecendo uma base sólida para o desenvolvimento subsequente.
- **Desenhar mockups dos ecrãs:** os *mockups* dos ecrãs foram criados conforme planeado, proporcionando uma visão clara da futura *interface* da aplicação.
- **Desenvolver alguns ecrãs (Frontend da aplicação):** o desenvolvimento do *frontend* da aplicação progrediu conforme o planeado, com a conclusão de vários ecrãs-chave.
- **Configurar o backend da aplicação:** a parte lógica da aplicação foi configurada e encontra-se

operacional, cumprindo os requisitos estabelecidos.

- **Extração de dados:** este tópico não foi possível realizar, devido ao impedimento de usar na totalidade a API do *facebook*, mas foi demonstrado que era possível com as autorizações necessárias.
- **Criação de campanhas:** o processo de criação de campanhas dos retalhistas foi desenhado com sucesso.

Portanto, com base na conclusão bem-sucedida destes tópicos e na realização do *MVP*, pode-se afirmar que o projeto foi concluído com êxito.

Note-se que, um dos principais objetivos desta dissertação foi responder a uma partilha de dados mais justa, de forma a devolver aos utilizadores a possibilidade de escolherem se querem ver os seus dados das redes sociais partilhados, e se fosse esse o caso, rentabilizar essa venda, sempre tendo em conta as empresas, tratando-se assim de uma solução *win-win*.

Por fim, pode-se concluir que este projeto representou um passo importante em direção a uma abordagem mais justa, para a partilha de dados entre utilizadores e empresas, proporcionando benefícios mútuos.

7.2 Análise crítica

Durante a execução deste projeto, a metodologia *Scrum* desempenhou um papel fundamental na realização do *MVP*. A realização das reuniões de *Sprint Planning*, nas quais foram definidos cada *user story* e seu valor, permitiu entregar funcionalidades de alta qualidade em prazos bastante curtos. No entanto, é importante notar que, em algumas ocasiões, a falta de experiência impactou negativamente algumas dessas entregas, em particular no domínio da programação. Foi necessário obter um conhecimento sólido em diversas áreas, das quais a autora nunca tinha trabalhado anteriormente. As dificuldades surgiram nomeadamente em:

- *TypeScript*: foi essencial adquirir conhecimentos sólidos em *TypeScript*, uma vez que, desempenhou um papel crucial no desenvolvimento. A sua abordagem à tipagem estática apresentou um desafio adicional.
- *JEST*: a implementação de testes foi uma parte fundamental do processo, que exigiu a aprendizagem e aplicação do *framework* de testes *JEST*.

Cada um destes desafios contribuiu para o enriquecimento de conhecimentos na área do desenvolvimento de *software* e, em particular, na implementação de sistemas seguros e escaláveis.

Contudo, apesar destes desafios, a adoção do *Scrum* revelou-se fundamental devido às suas vantagens. Uma das principais vantagens identificadas foi a flexibilidade oferecida pelo *Scrum*, em relação aos requisitos, que não são estáticos. Isto significa que tinha a capacidade de realizar ajustes e modificações sempre que necessário ao longo do projeto. Essa adaptabilidade provou ser inestimável, permitindo responder às mudanças nas necessidades e aos *feedbacks* dos *PO* de maneira ágil.

Portanto, pode-se concluir que o *Scrum* se alinhou perfeitamente com as necessidades do projeto, conforme delineado nos objetivos definidos no primeiro capítulo. A capacidade de entrega rápida e a flexibilidade para se ajustar às mudanças foram elementos-chave que contribuíram para o sucesso do projeto.

Apesar dos benefícios inegáveis proporcionados por esta metodologia ágil, é importante reconhecer que este projeto enfrentou desafios significativos ao longo da sua execução. Um dos desafios mais notórios foi a disponibilidade limitada por parte dos líderes da equipa, que estavam simultaneamente envolvidos em outros projetos. Esta sobrecarga de trabalho muitas vezes dificultou a obtenção do suporte necessário, o que acabou por afetar a entrega de algumas atividades, visto que devido à falta de experiência da equipa em certas áreas técnicas, era frequentemente necessária assistência adicional no desenvolvimento técnico.

7.3 Trabalho futuro

A proposta de trabalho futuro a desenvolver no âmbito deste projeto, baseia-se na conclusão das tarefas pendentes. A Tabela 12 resume o estado de cada *sprint* em relação às atividades planeadas por *sprint*, com a maioria delas já concluídas. A conclusão bem-sucedida dessas *sprints* demonstra que o projeto está avançando de acordo com o planeado.

Tabela 12: Resumo do Estado das Atividades por Sprint

Sprint	Descrição das Atividades	Estado
1	Desenvolvimento de mockups e análise de APIs	Concluído
2	Configuração de ambiente e início do código	Concluído
3	Integração com Firebase e conclusão de layouts	Concluído
4	Testes	Concluído
5	Desenvolvimento do layout e início da autenticação	Em andamento
6	Implementação do Pub/Sub no GCP	Em andamento
7	Terraform	Não realizado

Além da conclusão das tarefas pendentes, a empresa está a avaliar a possibilidade de transformar esta aplicação num produto interno. Os próximos passos envolverão algum investimento adicional, principalmente no que diz respeito ao desenvolvimento de recursos adicionais e à escalabilidade da aplicação para acomodar um maior número de utilizadores e redes sociais. Esta avaliação baseia-se na perspetiva de que a aplicação tem o potencial de se tornar uma ferramenta valiosa para a empresa. Se isto se concretizar, a extração dos dados por parte da API do *Facebook* irá ser implementada e, irão ser concluídas as restantes componentes, como o *Dashboard*. Isso permitirá ao cliente compreender que tipo de informação está a ser extraída e os ganhos que obtém com ela (sistema de *tokens*), além disto irá ser realizada a implementação do *MarketPlace*.

Além das atividades do projeto, esta dissertação abre caminho para futuras investigações e expansões. Com o desenvolvimento do *MVP* sucedido com êxito, os desafios éticos e legais associados à partilha de dados também serão objeto de estudo, com o objetivo de proporcionar uma base sólida para a compreensão e implementação de práticas justas. Além disso, a pesquisa na área da partilha de dados e na escolha do utilizador continuará a ser um foco principal.

Referências

- Accenture (2023). Accenture service labs: Advanced technology. <https://www.accenture.com/us-en/service-labs-advanced-technology>. Acedido em 30 de maio de 2023.
- Ali, W., Shafique, M. U., Majeed, M. A., e Raza, A. (2019). Comparison between sql and nosql databases and their relationship with big data analytics. *Asian Journal of Research in Computer Science*, 4(2):1–10.
- Alliance, T. A. (2001). Manifesto for agile software development. <http://agilemanifesto.org/>. Acedido a 08 de setembro de 2023.
- Atlassian (2023a). Jira, confluence, and scrum | atlassian. <https://www.atlassian.com/agile/scrum/jira-confluence-scrum>. Acedido em 25 de setembro de 2023.
- Atlassian (2023b). Using jira and confluence for sprint planning and refinement. <https://www.atlassian.com/agile/tutorials/jira-confluence-sprint-refinement>. Acedido a 11 de setembro de 2023.
- AWS (2023). O que é uma api? – explicação sobre interfaces de programação de aplicações – aws. <https://aws.amazon.com/pt/what-is/api/>. Acedido a 12 de agosto de 2023.
- Becker, M. Y. e Sewell, P. (2004). Cassandra: Flexible trust management, applied to electronic health records. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop, 2004*, pages 139–154. IEEE.
- Bierman, G., Abadi, M., e Torgersen, M. (2014). Understanding typescript. In Jones, R., editor, *ECOOP 2014 – Object-Oriented Programming*, pages 257–281, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Buddy: The DevOps Automation Platform (2020). 5 types of git workflows that will help you deliver better code. Acedido a 20 de agosto de 2023.
- Byali, R. (2022). Analysis of big data in social network marketing and social media.
- Chazette, L. e Schneider, K. (2020). Explainability as a non-functional requirement: Challenges and recommendations. *Requirements Engineering*, 25(4):493–514.
- Chen, H., Chiang, R. H. L., e Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. *MIS Quarterly*, 36(4):1165–1188.

Cole, D., Nelson, J., e McDaniel, B. (2015). Benefits and risks of big data. Acedido a 12 de maio de 2023.

Danielsson, W. (2016). A comparison between native android and react native.

Debug Token - Graph API (2023). Debug token - graph api. https://developers.facebook.com/docs/graph-api/reference/v17.0/debug_token. Acedido a 17 de julho de 2023.

Dybå, T., Dingsøy, T., e Moe, N. B. (2014). Agile project management.

European Commission (2016). The pm² project management methodology guide – open edition. Current Edition: The PM² Guide – Open Edition, v.0.9, November 2016.

Expo (2022). <https://docs.expo.dev/versions/latest/>. Acedido a 21 de julho de 2023.

Expo (2023). <https://expo.dev/client>. Acedido a 21 de julho de 2023.

Firebase Docs (2023). Firebase cloud messaging. <https://firebase.google.com/docs/cloud-messaging?hl=pt-br>. Acedido a 02 de setembro de 2023.

Flanagan, D. (2004). *JavaScript: O Guia Definitivo*. Bookman Ed.

Frachet, M. (2020). Understanding the react native bridge concept. <https://hackernoon.com/understanding-react-native-bridge-concept-e9526066ddb8>. Acedido a 12 de agosto de 2023.

GeeksforGeeks (2020). Business logic layer. <https://www.geeksforgeeks.org/business-logic-layer/>. Acedido a 16 de junho de 2023.

Geewax, J. J. J. (2018). *Google Cloud platform in action*. Simon and Schuster.

Google (2023). Bigquery enterprise data warehouse. <https://cloud.google.com/bigquery>. Acedido a 6 de Junho de 2023.

Google Cloud (2021). O que é o pub/sub? <https://cloud.google.com/pubsub/docs/overview?hl=pt-br>. Acedido a 13 de agosto de 2023.

Google Developer (2020). Firebase authentication. <https://firebase.google.com/docs/firestore>. Acedido a 12 de agosto de 2023.

- Google Inc. (2021). Firebase documentation. <https://firebase.google.com/docs/flutter/setup?platform=android>. Acedido a 6 de Junho de 2023.
- Gülcüoğlu, E., Ustun, A. B., e Seyhan, N. (2021). Comparison of flutter and react native platforms. *Journal of Internet Applications and Management*, 12(2):129 – 143.
- Harrison, N. B. e Avgeriou, P. (2010). How do architecture patterns and tactics interact? a model and annotation. *Journal of Systems and Software*, 83(10):1735–1758.
- Helland, P. (2016). O sucesso singular de sql. *Communications of the ACM*, 59(8):38–41.
- Hoda, R., Noble, J., e Marshall, S. (2008). Agile project management. In Holland, J., Nicholas, A., e Brignoli, D., editors, *New Zealand Computer Science Research Student Conference, NZCSRSC 2008 - Proceedings*, pages 218–221, Christchurch, New Zealand.
- IBM (2019). Nosql databases. <https://www.ibm.com/cloud/learn/nosql-databases>. Acedido a 12 de agosto de 2023.
- IBM (2023). What is an application programming interface (api)? <https://www.ibm.com/topics/api>. Acedido a 12 de agosto de 2023.
- IEEE (1990). Ieee standard glossary of software engineering terminology.
- IEEE Computer Society (2014). Swebok: Guide to the software engineering body of knowledge. Versão 3.0.
- Institute, P. M. (2017). *Agile Practice Guide*. Project Management Institute, Newtown Square, Pennsylvania.
- Jhanjhi, N., Naqvi, M., et al. (2019). Analysis of software development methodologies. *International Journal of Computing and Digital Systems*, 8(5).
- Larman, C. (2005). *Applying UML And Patterns*. Prentice Hall PTR.
- Morandini, M., Coleti, T. A., Oliveira Jr, E., e Corrêa, P. L. P. (2021). Considerations about the efficiency and sufficiency of the utilization of the scrum methodology: A survey for analyzing results for development teams. *Computer Science Review*, 39:100314.
- Nance, C. (2014). *TypeScript Essentials: Develop Large Scale Responsive Web Applications with TypeScript*. Packt Publishing Ltd.

- NestJS (2023). Microservices | nestjs - a progressive node.js framework. <https://docs.nestjs.com/microservices/basics>. Acedido a 19 de julho de 2023.
- Nicolás, J. e Toval, A. (2009). On the generation of requirements specifications from software engineering models: A systematic literature review. *Information and Software Technology*, 51(9):1291–1307.
- Nitnaware, S., Nitnaware, R., e Masleker, A. (2023). A comparative study on big data and big data analytics (bda) and challenges.
- Occhino, T. (2020). React native: Bringing modern web techniques to mobile. <https://engineering.fb.com/android/react-native-bringing-modern-web-techniques-to-mobile/>. Acedido a 30 de julho de 2023.
- OECD (2020). Privacy and data protection. https://www.oecd-ilibrary.org/sites/bb167041-en/1/3/6/index.html?itemId=/content/publication/bb167041-en&_csp_=509e10cb8ea8559b6f9cc53015e8814d&itemIG0=oecd&itemContentType=book#chapter-6. Acedido a 3 de maio de 2023.
- Prescott, P. (2016). *Programação em JavaScript*. Babelcube Inc.
- Project Management Institute (2021). A guide to the project management body of knowledge (pmbok® guide) – seventh edition. Acedido a 19 de julho de 2023.
- Reddy, H. B. S., Reddy, R. R. S., Jonnalagadda, R., Singh, P., e Gogineni, A. (2022). Analysis of the unexplored security issues common to all types of nosql databases. *Asian Journal of Research in Computer Science*, 14(1):1–12.
- Salameh, H. (2014). What, when, why and how: A comparison between agile project management and traditional project management methods. <https://d1wqtxts1xzle7.cloudfront.net/46679893/What-When-Why-and-How-A-Comparison-between-Agile-Project-Management-and-Traditional-Project-Management-Methods-libre.pdf>. Acedido a 19 de julho de 2023.
- Sangeetha, K., Poongothai, T., Anguraj, S., e Kalyani, S. N. (2018). An overview of applications of big data analytics.
- Scrum.org (2023). What is scrum? <https://www.scrum.org/resources/what-scrum-module>. Acedido a 12 de julho de 2023.

- Seymour, D. T. e Hussein, S. (2014). The history of project management. *International Journal of Management & Information Systems (IJMIS)*, 18:233.
- Sommerville, I. (2016). *Software Engineering*. Pearson, 10th edition.
- Srinivasa, K. e Hiriannaiah, S. (2018). A deep dive into nosql databases: The use cases and applications.
- Stack Overflow (2022). Stack overflow developer survey 2022. <https://survey.stackoverflow.co/2022#professional-developers>. Acedido a 12 de julho de 2023.
- Sweet, J. e Sharma, M. (2023). Accenture fact sheet q2 fy23 clients & industry expertise. *Obtido de www.accenture.com*.
- Team, W. A. (2020). React native on the universal windows platform. <https://blogs.windows.com/windowsdeveloper/2016/04/13/react-native-on-the-universal-windows-platform/>. Acedido a 1 de agosto de 2023.
- Vinet, L. e Zhedanov, A. (2011). A 'missing' family of classical orthogonal polynomials. *Journal of Physics A: Mathematical and Theoretical*, 44:085201.
- Watt, A. (2014). *Database Design*. BCcampus, BC Open Textbook Project.
- Wu, W. (2018). React native vs flutter, cross-platform mobile application frameworks.