

Departamento de Engenharia Informática e de Computadores

PlanIt

49510 : Tiago Neutel (a49510@alunos.isel.pt)

49521 : Daniel Pojega (a49521@alunos.isel.pt)

Relatório para a Unidade Curricular de Projeto e Seminário
da Licenciatura em Engenharia Informática e de Computadores

Supervisores : Professor Luís Falcão
Professor Paulo Pereira

Resumo

Divulgar eventos tende a ser uma tarefa complicada, seja combinar um simples encontro com amigos ou tentar promover um concerto que vai acontecer localmente. Isto deve-se ao facto de que, atualmente, as ferramentas necessárias para gerir estas necessidades estão fragmentadas, tornando o processo mais difícil e ineficiente.

PlanIt, uma plataforma Web e Mobile, visa facilitar este processo, proporcionando um local único para a divulgação e descoberta de eventos. Seja alguém que queira planejar uma reunião casual entre amigos ou um promotor que deseja promover um evento de grande escala, PlanIt simplifica este processo, tornando-o acessível para todos.

Os utilizadores podem criar eventos, categorizá-los e torná-los visíveis para um público mais amplo. Por eventos entende-se qualquer tipo de encontro, reunião ou acontecimento que vai acontecer numa dada data, sejam eles físicos ou online, públicos ou privados.

A plataforma oferece uma funcionalidade de pesquisa abrangente, permitindo aos utilizadores encontrar eventos com base em vários critérios, garantindo que possam descobrir eventos que correspondam aos seus interesses. Cada evento tem um canal de comunicação dedicado, que permite a interação entre participantes e organizadores, melhorando a experiência geral do evento.

Abstract

Publicizing events can be a complicated task, whether it's planning a simple gathering with friends or trying to promote a local concert. Equally challenging is the task of finding events that people are interested in attending. Both of these are due to the fact that the tools currently needed to manage these needs are fragmented, making the process cumbersome and inefficient.

PlanIt, a web and mobile platform, aims to solve these challenges by providing a centralized place for event divulgation and discovery. Whether it is someone who wants to plan a casual get-together with friends or a promoter who wants to promote a large-scale event, PlanIt simplifies this process, making it accessible and straightforward for everyone.

Users can create events, categorize them, and make them visible to a wider audience. Events are defined as any type of gathering, meeting or occasion that will take place on a given date, whether physical or online, public or private.

The platform offers a comprehensive search functionality, allowing users to find events based on multiple criteria, ensuring that they can discover events that match their interests. Each event has a dedicated communication channel, which allows interaction between participants and organizers, enhancing the overall event experience.

Índice

Lista de Figuras	ix
Lista de Tabelas	x
1 Introdução	1
2 Funcionalidades e conceitos da Plataforma	3
2.1 Utilizadores	3
2.2 Eventos	4
2.2.1 Definição de Eventos	4
2.2.2 Detalhes dos Eventos	5
2.2.3 Lista de Participantes e Organizadores	6
2.2.4 Canais de comunicação	6
2.2.5 Votações	7
2.3 NearMe	7
3 Ferramentas	9
3.1 Desenvolvimento Backend (API)	9
3.1.1 Kotlin	9
3.1.2 Spring MVC	10
3.1.3 Gradle	10

3.1.4	JDBI	10
3.1.5	SQL	10
3.2	Desenvolvimento Frontend	11
3.2.1	Web	11
3.2.2	Mobile	11
3.2.3	Google Maps API	12
3.2.4	Geocoding API	12
3.2.5	Google Calendars API	12
3.3	Armazenamento de Dados	13
3.3.1	PostgreSQL	13
3.3.2	Cloud Firestore	13
3.4	Ferramentas DevOps	13
3.4.1	Git e GitHub	13
3.4.2	Postman	14
3.4.3	PgAdmin	14
4	Arquitetura	15
4.1	Visão Global da Arquitetura	15
4.2	Base de Dados	17
4.2.1	Estrutura e Funcionamento	17
4.2.2	Importância da Base de Dados	17
4.2.3	Interação com os restantes Módulos	18
4.2.4	Implementação	18
4.2.5	Canais de Comunicação (Chats)	22
4.3	Backend	23
4.3.1	Divisão do Backend	23
4.3.2	Autenticação	24
4.3.3	Spring Framework: Base do Backend	25
4.3.4	Camada de Apresentação	26

4.3.4.1	Definição e Função	26
4.3.4.2	Responsabilidades	26
4.3.4.3	Implementação da camada de apresentação	27
4.3.5	Camada de Serviços	31
4.3.6	Camada de Repositório	32
4.4	Frontend	32
4.4.1	Web	33
4.4.2	Android	34
4.4.3	Navegabilidade	34
4.4.4	Google Maps API	37
4.4.5	Google Calendar API	38
4.4.6	Chat	38
5	Enquadramento	39
5.1	Conceptualização	39
5.2	Casos de Uso	40
5.2.1	Divulgação e Pesquisa de Eventos	40
5.2.2	Calendário Pessoal	40
5.3	Produtos Semelhantes	40
5.3.1	Eventbrite	41
5.3.2	Meetup	41
5.3.3	Google Calendar	41
5.3.4	Asana	41
5.4	Requerimentos	42
5.4.1	Acesso à Internet	42
5.4.2	Acesso via Web	42
5.4.3	Acesso via Mobile	42
6	Conclusão	43
6.1	Conclusões	43
6.2	Planos Futuros	44

7 Referências	46
----------------------	-----------

A Anexos	i
-----------------	----------

Lista de Figuras

4.1	Esquema das camadas da arquitetura da Plataforma PlanIt	16
4.2	Esquema das camadas da arquitetura da Plataforma PlanIt	22
4.3	Esquema representativo do caminho realizado pelos pedidos dos clientes no Backend	24
4.4	Esquema da Camada de Apresentação	30
4.5	Esquema da Camada de Serviços e a interação com as restantes camadas	31
4.6	Esquema do frontend	33
4.7	Esquema das camadas da aplicação Web	34
4.8	Esquema das camadas aplicação Mobile	34
4.9	Diagrama de Navegabilidade - Web	36
4.10	Diagrama de Navegabilidade - Mobile	37
A.1	Modelo EA da base de dados PlanIt	v

Lista de Tabelas

A.1	Símbolos do Modelo EA	i
A.2	Métodos do UserController	ii
A.3	Métodos do EventController	iii
A.4	Métodos do PollController	iv



Introdução

O mercado de eventos é altamente fragmentado, com uma grande variedade de plataformas e ferramentas que atendem a diferentes aspectos da promoção de eventos. No entanto, não existe uma plataforma única que centralize todas as funcionalidades necessárias para atender às diversas necessidades dos utilizadores.

A fragmentação no mercado de gestão de eventos manifesta-se através da ausência de mecanismos centralizados de busca e descoberta, resultando naqueles que procuram ou desejam divulgar eventos necessitem de navegar por várias plataformas para encontrar eventos que correspondam aos seus interesses. Esta abordagem dispersa não só limita a visibilidade dos eventos, mas também dificulta a busca por parte dos interessados, que resulta numa participação no evento inferior à possível.

Estudos e análises do setor de eventos destacam essa fragmentação. De acordo com um relatório da Allied Market Research, o mercado global de gestão de eventos está projetado para crescer significativamente nos próximos anos, mas enfrenta desafios devido à falta de integração e centralização das ferramentas disponíveis. Além disso, uma pesquisa da Event Manager Blog revela que a maioria dos organizadores de eventos utiliza mais de cinco ferramentas diferentes para gerir um único evento, indicando uma clara necessidade de uma solução unificada.

Uma plataforma que pudesse centralizar todas essas funcionalidades não só simplificaria o processo de organização e participação em eventos, mas também melhoraria a experiência do utilizador, permitindo uma gestão mais eficiente e uma comunicação mais fluida.

PlanIt tem como objetivo preencher este buraco no mercado, oferecendo uma plataforma que integra funcionalidades essenciais para a divulgação e descoberta de eventos. A plataforma é desenhada para tentar atender às diversas necessidades dos organizadores e participantes de eventos, fornecendo uma interface unificada que simplifica todo o processo de divulgação de eventos. Ao consolidar ferramentas para criação, divulgação e descoberta de eventos, PlanIt busca reduzir a fragmentação e facilitar a chegada de eventos aos interessados.

Os principais recursos que PlanIt disponibiliza incluem a capacidade de criar e categorizar eventos, tornando-os facilmente acessíveis aos utilizadores por meio de opções avançadas de pesquisa e filtragem. A plataforma suporta uma ampla variedade de tipos de eventos, desde pequenas reuniões privadas até grandes eventos públicos, permitindo versatilidade e flexibilidade. Além disso, PlanIt oferece canais de comunicação dedicados para cada evento, facilitando a interação entre participantes e organizadores. Este recurso de comunicação integrado garante que todas as informações e atualizações relevantes sejam facilmente reconhecidas, aprimorando a experiência geral do evento.

PlanIt tem como público-alvo indivíduos que planeiam eventos pessoais, pequenos grupos (como grupos comunitários que organizam reuniões regulares ou eventos especiais) e organizadores profissionais de eventos que administram eventos públicos de grande dimensão. Ao atender às necessidades específicas destes segmentos de indivíduos variados, PlanIt tem como objetivo tornar-se a plataforma preferencial para a divulgação de eventos.

Assim, a estratégia de PlanIt para a gestão de eventos passa por tentar responder às lacunas críticas do mercado atual, oferecendo uma solução coesa que melhore tanto a organização quanto a descoberta de eventos. Os recursos e o design centrado no utilizador da plataforma têm o objetivo de agilizar o processo de gestão de eventos, tornando-o mais acessível para todos os envolvidos. Desta forma, PlanIt tem o potencial de impactar a forma como os eventos são divulgados, promovendo um maior envolvimento e participação da comunidade.



Funcionalidades e conceitos da Plataforma

2.1 Utilizadores

Um utilizador é um indivíduo que utiliza a plataforma PlanIt. Desde o momento em que acede à plataforma, o indivíduo passa a ser considerado um utilizador. Antes de efetuar qualquer forma de registo, o utilizador é considerado um "guest", ou seja, um utilizador não autenticado e tem a opção de se autenticar na plataforma através do login ou criar uma nova conta.

Quando um guest acede à plataforma PlanIt, ele pode optar por uma de duas ações: login ou registo. Se o utilizador já possui uma conta, ele pode aceder à sua conta inserindo as suas credenciais de acesso. Após a inserção correta do nome de utilizador (ou e-mail) e palavra-passe, ele é redirecionado para a página inicial da plataforma, dentro da sua conta. Caso o utilizador não possua uma conta, ele pode criar uma nova conta na plataforma. Durante o processo de registo, são solicitadas várias informações. O nome de utilizador (username) é único na plataforma, com pelo menos 5 dígitos, identificando o utilizador de forma exclusiva. Além disso, é pedido um nome de apresentação (também conhecido como "Display Name"), que é mostrado aos outros utilizadores. É também pedido um e-mail e uma palavra-passe para a conta. O e-mail, tal como o username, é único para cada utilizador.

Após concluir o registo na plataforma, o utilizador é convidado a personalizar o seu

perfil com informações adicionais. Ele pode inserir os seus interesses e além disso, pode fornecer uma breve descrição pessoal, permitindo que outros utilizadores da plataforma conheçam um pouco mais sobre ele. Estes detalhes foram implementados para fornecer mais informações sobre os utilizadores, além de potencialmente servirem como base para futuras funcionalidades da plataforma.

Os utilizadores têm a capacidade de editar dados das suas contas a qualquer momento. O nome de apresentação pode ser atualizado conforme necessário. Os interesses podem ser alterados para refletir mudanças nas preferências do utilizador. A descrição pessoal pode ser atualizada para manter o perfil do utilizador atual e relevante.

Na página inicial da plataforma, os utilizadores podem visualizar os eventos nos quais estão inscritos, bem como criar novos eventos. Ao juntar-se a um evento, o utilizador recebe o título de "participante" dentro do evento. Os participantes têm acesso à visualização dos detalhes do evento, à capacidade de visualizar e enviar mensagens no canal de comunicação do evento, à visualização da lista de participantes e organizadores, e à possibilidade de votar em votações relacionadas ao evento.

Quando um utilizador cria um evento, ele assume o título de "organizador" do evento. Os organizadores possuem todas as capacidades dos participantes, além de funções adicionais específicas. Podem alterar os detalhes dos eventos conforme necessário, expulsar participantes do evento, promover participantes a organizadores, rebaixar organizadores a participantes, criar novas votações para o evento e eliminar votações existentes. Estas capacidades adicionais garantem que os organizadores têm o controle necessário para gerir e coordenar efetivamente as informações dos eventos. Em suma, os organizadores são participantes dos eventos com permissões especiais relativamente aos restantes participantes.

2.2 Eventos

2.2.1 Definição de Eventos

Esta plataforma adota um conceito abstrato daquilo a que são chamados "eventos". Os utilizadores podem criar encontros privados simples, como uma reunião casual com amigos, garantindo privacidade e exclusividade. Eventos comunitários locais, como as reuniões de um clube de leitura ou de um grupo de ciclismo, podem ser facilmente organizados e partilhados dentro da comunidade através da plataforma. Os eventos também podem ser de maior escala, como concertos de música, peças de teatro ou festivais públicos.

Assim, no contexto de PlanIt, os eventos podem ser definidos como qualquer tipo de encontro, reunião ou acontecimento que vai acontecer numa data específica, sejam eles físicos ou online, públicos ou privados. Desta forma, o uso da plataforma ganha uma grande versatilidade.

2.2.2 Detalhes dos Eventos

Os eventos possuem uma variedade de informações que são cruciais para definir as suas características e facilitar a sua descoberta. Ao criar um evento, alguns parâmetros são obrigatórios enquanto que outros são opcionais, que significa que nem todos os detalhes possíveis de inserir num evento têm que ser colocados. Esta medida foi tomada pela natureza abstrata dos eventos, já que nem todos os eventos necessitam dos mesmos detalhes, e também para possibilitar aos organizadores de definir mais tarde esses detalhes no caso de ainda não terem a certeza dos mesmos. Estes são os detalhes dos eventos que se encontram visíveis pelos participantes quando se juntam aos eventos:

- **Título** (Obrigatório): Nome do evento, utilizado para identificar o evento e distinguir dos restantes.
- **Descrição** (Opcional): Uma descrição detalhada do evento, onde são fornecidas informações adicionais que podem incluir o propósito do evento, agenda, atividades previstas, e outros detalhes relevantes.
- **Categoria** (Obrigatório): A categoria à qual o evento pertence. Exemplos de categorias podem incluir tecnologia, arte, desporto, entre outros.
- **Tipo de Localização** (Opcional): Indica se o evento será realizado fisicamente ou online. Este parâmetro ajuda a definir a natureza do evento.
- **Localização** (Opcional): Localização específica do evento. Para eventos físicos, isso pode incluir uma morada completa com coordenadas geográficas, que pode ser visualizada através de um mapa. Para eventos online, este campo pode conter um link para a plataforma onde o evento será realizado.
- **Visibilidade** (Obrigatório): Define se o evento é público ou privado. Eventos públicos são acessíveis a todos os utilizadores da plataforma, enquanto que eventos privados requerem um convite ou senha para acesso.
- **Data de Começo** (Obrigatório): A data e hora de início do evento.

- **Data de Fim** (Opcional): A data e hora de término do evento.
- **Preço** (Opcional): O valor e a moeda do preço do evento, caso haja algum custo associado à participação.

Existem outros parâmetros associados aos eventos que não são visíveis pelos utilizadores, como a palavra-passe e o código de acesso, que são utilizados para os utilizadores poderem aceder a eventos e por isso mesmo não são visíveis.

2.2.3 Lista de Participantes e Organizadores

Dentro de cada evento, existe uma lista de participantes e organizadores que é específica para esse evento. Esta lista fornece uma forma de visualizar todos os membros envolvidos no evento, incluindo tanto os participantes quanto os organizadores. A lista é acessível a todos os participantes do evento, permitindo que os membros possam ver quem está envolvido e entender a dinâmica do grupo.

Os organizadores do evento têm privilégios administrativos adicionais e são responsáveis pela gestão do evento. Eles podem adicionar ou remover participantes, promover participantes a organizadores, ou rebaixar organizadores a participantes. Além disso, os organizadores são identificados na lista com uma designação especial.

Através desta lista, os participantes têm a capacidade de aceder às páginas de perfil dos outros utilizadores.

A lista de participantes e organizadores também desempenha um papel importante na transparência e na segurança do evento, pois permite saber quem participa e quem está no controlo do evento, o que ajuda a criar um ambiente de confiança, onde os membros se sentem seguros e informados.

2.2.4 Canais de comunicação

Dentro de cada evento existe um canal de comunicação específico dedicado exclusivamente a esse evento. Toda a comunicação realizada dentro deste canal é restrita aos participantes e organizadores do evento, garantindo a privacidade e a relevância das informações trocadas. O objetivo principal deste canal é proporcionar um meio de comunicação eficaz entre todos os envolvidos no evento, facilitando a coordenação e a colaboração. Os participantes podem usar este canal para discutir detalhes logísticos, fazer perguntas, compartilhar atualizações e informações relevantes, e resolver quaisquer questões que possam surgir. Além disso, o canal de comunicação permite uma

interação direta e contínua, o que é essencial para o sucesso do evento, pois assegura que todos estejam bem informados e alinhados com os objetivos e cronogramas do evento. Este meio de comunicação integrado é fundamental para garantir a eficiência na organização e na participação dos eventos, promovendo uma experiência mais envolvente e colaborativa para todos os participantes.

2.2.5 Votações

Dentro de cada evento existe a funcionalidade de votações, que é específica para aquele evento. Esta ferramenta permite que os organizadores criem votações sobre diversos tópicos relacionados ao evento, tais como a escolha de datas, locais, atividades, ou quaisquer outras decisões que necessitem o consenso dos participantes. As votações são acessíveis a todos os participantes do evento, garantindo que cada membro tenha a oportunidade de expressar sua opinião e contribuir para o processo decisório.

Os organizadores têm a capacidade de criar novas votações, especificar as opções disponíveis, definir prazos para a votação e acompanhar os resultados em tempo real. Esta funcionalidade foi implementada para permitir que decisões importantes sobre os eventos sejam tomadas de forma transparente e participativa. Após o término do período de votação, os resultados são disponibilizados a todos os participantes, assegurando a transparência e a legitimidade do processo. É também possível visualizar a quantidade de votos em cada votação enquanto estas estão em progresso.

Através das votações, é possível obter um feedback valioso e assegurar que as preferências e opiniões dos participantes sejam consideradas, contribuindo para uma experiência de evento mais satisfatória e alinhada com as expectativas de todos os envolvidos.

2.3 NearMe

A funcionalidade NearMe permite ao utilizador encontrar eventos próximos à sua localização. Esta funcionalidade utiliza as coordenadas do utilizador, um raio definido por ele e o número de eventos que deseja procurar. Com base nestes parâmetros, é calculada a distância entre as coordenadas do utilizador e as coordenadas de cada evento armazenado na base de dados. Se a distância for inferior ao raio definido, o evento é incluído nos resultados apresentados ao utilizador. Visto que é necessário obter as coordenadas do utilizador, requer-se que este permita que se aceda à sua localização.

Após obter cada evento, é ainda possível navegar para a página correspondente ao evento selecionado, de modo a poder aceder aos seus detalhes.

3

Ferramentas

Para a construção de uma plataforma como PlanIt é necessário o uso de diversas ferramentas, cada uma escolhida pelas suas características e vantagens específicas. Estas ferramentas garantem a robustez, escalabilidade, segurança e facilidade do uso da plataforma para os utilizadores finais. Este capítulo é dedicado às principais ferramentas e tecnologias empregadas no desenvolvimento de PlanIt.

3.1 Desenvolvimento Backend (API)

O backend (API) é a parte da aplicação que lida com a lógica de negócio, o processamento de dados e a interação com a base de dados. É responsável por garantir que a aplicação funciona corretamente e que as informações são geridas de forma eficiente e segura.

3.1.1 Kotlin

Kotlin é uma linguagem de programação moderna e concisa. Por ser uma linguagem moderna, Kotlin possibilita a utilização de ferramentas modernas, concebidas para poderem ser utilizadas com esta linguagem, como é o caso da framework Spring MVC. Foi escolhida devido a esta possibilidade que oferece, assim como à sua capacidade de reduzir código boilerplate, o que facilita a manutenção e diminui a probabilidade de erros. Para além disso, dado que foi a principal linguagem lecionada ao longo do

curso, é a linguagem sobre a qual os desenvolvedores de PlanIt têm melhor domínio, permitindo aplicar com maior eficácia todos os conceitos aprendidos.

3.1.2 Spring MVC

Spring MVC é uma framework amplamente utilizada para a criação de aplicações empresariais. O Spring Boot, uma parte do ecossistema Spring, simplifica a configuração e a criação de aplicações standalone, acelerando o processo de desenvolvimento. A framework Spring MVC simplifica a definição de rotas através de controllers, permitindo que se especifique tanto o URI, como o respetivo método HTTP do pedido a ser feito, através de anotações. Spring também fornece filters que interceptam os pedidos antes e após chegarem aos Controllers, o que possibilita uma maior manipulação sobre os dados do pedido. Para além disso, é uma framework que permite uma melhor separação das responsabilidades por entre os diferentes componentes. Tendo em conta a robustez e segurança desta framework, estas funcionalidades são pontos muito fortes, que apoiaram a tomada de decisão na escolha desta ferramenta.

3.1.3 Gradle

Gradle é uma ferramenta de automação de construção que é utilizada para compilar, construir e gerir dependências do projeto. No backend, Gradle facilita a configuração e a gestão das dependências a usar no projeto.

3.1.4 JDBI

JDBI é uma biblioteca que simplifica a interação com a base de dados em aplicações Java/Kotlin. Esta oferece uma API de alto nível que permite a execução de consultas SQL de forma eficiente, integrando-se perfeitamente com a linguagem Kotlin. A utilização de JDBI facilita a escrita e manutenção de código de acesso a dados, promovendo a clareza nas transações com a base de dados.

3.1.5 SQL

SQL (Structured Query Language) é a linguagem padrão para gestão e manipulação de bases de dados relacionais. Em PlanIt, SQL é utilizada para realizar consultas, inserções, atualizações e remoções de dados no PostgreSQL. O uso de SQL garante que

as operações sobre a base de dados sejam realizadas de forma eficiente e que a integridade dos dados seja mantida.

3.2 Desenvolvimento Frontend

O frontend é a parte da aplicação que interage diretamente com o utilizador. É responsável pela interface de utilizador (UI) e pela experiência do utilizador (UX), tentando garantir que a aplicação seja intuitiva e fácil de usar. O desenvolvimento frontend divide-se em duas partes: Web e Mobile.

3.2.1 Web

O frontend Web é a parte da aplicação que interage diretamente com o utilizador através de um navegador (browser).

Para a construção dos diversos componentes da interface do utilizador, foi utilizado React. A sua abordagem baseada em componentes permite a criação de UIs complexas de forma modular e reutilizável. Para além disso, HTML foi utilizado para definir a estrutura e o conteúdo das páginas, enquanto que CSS foi utilizado para a apresentação visual e layout da interface do utilizador.

Para complementar, TypeScript é uma extensão do JavaScript que adiciona tipagem estática ao código. Isso ajuda a detetar erros durante o desenvolvimento e melhora a manutenção do código a longo prazo. A utilização de TypeScript em conjunto com React proporcionou uma maior robustez e confiabilidade no desenvolvimento do frontend.

Por fim, é utilizado WebPack. Esta é uma ferramenta que, através do ts-loader, compila ficheiros TypeScript para JavaScript que posteriormente sofrem empacotamento num só ficheiro. No contexto do projeto, WebPack é responsável por fazer este empacotamento para além de fornecer um servidor de desenvolvimento, usado para auxiliar todo o processo de desenvolvimento da aplicação.

3.2.2 Mobile

O frontend Mobile é a parte da aplicação com a qual o utilizador interage através de dispositivos Android.

Android Studio é um IDE (Integrated Development Environment) que fornece a possibilidade de desenvolver aplicações Android de forma facilitada, e é a principal ferramenta utilizada para desenvolvimento de software Android. Neste é possível selecionar uma variedade de linguagens de programação, no entanto, foi optado por utilizar Kotlin por motivos de consistência e pelas mesmas razões enumeradas na secção de backend. Dentro do Android Studio é ainda possível utilizar a ferramenta Jetpack Compose que possibilita criar os componentes visuais da aplicação com os quais os utilizadores interagem.

Gradle é também utilizado no desenvolvimento Android para compilar o código, gerir dependências e construir a aplicação. Permite configurar diferentes variantes de build e facilita a integração contínua e a entrega contínua (CI/CD).

3.2.3 Google Maps API

A API Google Maps fornece recursos avançados para a visualização e manipulação de mapas, incluindo a capacidade de marcar localizações e calcular rotas. A sua integração no frontend da plataforma melhora significativamente a experiência do utilizador, permitindo uma fácil localização dos eventos e definição do local de acontecimento dos mesmos.

3.2.4 Geocoding API

A Geocoding API foi utilizada para converter endereços em coordenadas geográficas (geocodificação) e coordenadas geográficas em endereços (geocodificação inversa). Esta API permitiu que as aplicações web e Android apresentassem endereços legíveis aos utilizadores e convertessem entradas de localização em dados geográficos utilizáveis.

3.2.5 Google Calendars API

A API Google Calendars possibilita aos utilizadores visualizarem as marcações que realizam neste calendário. Ao integrar esta API no frontend da plataforma PlanIt, os seus utilizadores podem ter acesso aos seus calendários pessoais da Google para marcarem também nestes os eventos nos quais estão inscritos em PlanIt.

3.3 Armazenamento de Dados

O armazenamento de dados é crucial para esta aplicação, pois garante a persistência e a integridade das informações. Para esta finalidade, foi escolhida a base de dados PostgreSQL e Cloud Firestore.

3.3.1 PostgreSQL

PostgreSQL é um sistema de gerenciamento de base de dados relacional open-source, conhecido pela sua robustez, desempenho e conformidade com os padrões SQL. Oferece funcionalidades avançadas, como suporte a tipos de dados personalizados e transações ACID, garantindo a integridade e a consistência dos dados da plataforma.

3.3.2 Cloud Firestore

O Cloud Firestore, pertencente à plataforma Firebase da Google, é uma base de dados escalável para desenvolvimento focado em dispositivos móveis e aplicações Web. É uma ferramenta que permite a sincronização em tempo de real dos dados entre o cliente e o servidor, sendo esta uma das principais razões para a escolha desta ferramenta para o projeto. Nesta base de dados, os dados são armazenados em documentos que, por sua vez, são armazenados em coleções. Os documentos suportam vários tipos de dados diferentes e contêm mapeamentos de campos para cada valor.

3.4 Ferramentas DevOps

Dá-se o nome de DevOps às ferramentas que auxiliam e agilizam o processo de desenvolvimento da aplicação, seja na integração contínua, monitorização de dados, ou implementação de testes. Para PlanIt, as ferramentas DevOps desempenharam um papel crucial na melhoria da colaboração e garantia da qualidade do código.

3.4.1 Git e GitHub

Git é um sistema de controlo de versões distribuído que permite às equipas gerir alterações no código-fonte, colaborar e rastrear mudanças ao longo do tempo. GitHub, por outro lado, é um serviço de hospedagem de repositórios Git que fornece funcionalidades adicionais de colaboração e gestão de projetos. No caso de PlanIt, deu-se uso a

dois repositórios para armazenar o código tanto da aplicação Web como da Android e para acompanhar o progresso de ambas.

3.4.2 Postman

Postman é uma ferramenta de desenvolvimento amplamente utilizada que permite testar e documentar pedidos HTTP. No contexto da plataforma, permitiu testar minuciosamente os endpoints da API (Backend) de modo a garantir a sua fiabilidade. Esta ferramenta foi escolhida devido à experiência positiva obtida anteriormente e dada à familiaridade com a mesma.

3.4.3 PgAdmin

PGAdmin é uma ferramenta que auxilia na gestão de bases de dados PostgreSQL. Para além disso, possibilita o teste de queries SQL que diminuem a presença de erros. No cenário da plataforma, esta ferramenta apoiou o desenvolvimento do backend, dado que permitiu consultar e gerir dados através de uma interface gráfica.

4

Arquitetura

Este capítulo destaca os principais componentes da plataforma e a forma como interação entre si, assim como as razões que levaram às decisões das formas de implementação selecionadas na arquitetura.

4.1 Visão Global da Arquitetura

A arquitetura da plataforma PlanIt foi desenhada para proporcionar um sistema robusto e escalável, capaz de lidar com as exigências de uma aplicação moderna de divulgação e descoberta de eventos. Para tal é necessário, em primeiro lugar, formular o desenho da arquitetura. É necessário uma interface com a qual o utilizador possa interagir e também é necessário um meio que seja capaz de dar acesso aos dados a que o utilizador deseja aceder, mas que também valide os pedidos que o utilizador esteja a realizar antes de aceder aos dados, para manter a integridade dos mesmos. Assim, é possível chegar à conclusão que para desenhar a plataforma é necessário dividir esta em 3 partes; aquilo a que se chama de Frontend (interface), Backend (meio de acesso aos dados) e a base de dados.

A formulação da arquitetura requer uma abordagem estruturada que assegure tanto a interação eficiente do utilizador quanto a integridade e segurança dos dados.

A plataforma é constituída por três módulos principais:

1. **Frontend:** Este módulo da arquitetura é responsável pela interface com a qual o utilizador interage. É a camada de apresentação da aplicação, onde se realiza a visualização e a interação com os dados. É suposto que o Frontend seja intuitivo e responsivo, de forma a garantir uma experiência agradável e eficiente ao utilizador.
2. **Backend:** O Backend é o intermediário entre o Frontend e a base de dados. É onde são processados os pedidos dos utilizadores, aplicada a lógica de negócio, feita a validação dos pedidos e realizada a interação com a base de dados para armazenar ou recuperar informações. Esta camada é crucial para manter a integridade e a segurança dos dados, assegurando que somente operações válidas sejam executadas.
3. **Base de Dados:** A base de dados é o repositório onde todas as informações da aplicação são armazenadas. É vital que esta seja robusta e capaz de lidar com grandes volumes de dados, de maneira a garantir a integridade, consistência e disponibilidade das informações.

Cada um destes módulos da arquitetura da plataforma desempenha um papel fundamental na operação da plataforma PlanIt. Esta arquitetura modular permite uma separação clara de responsabilidades, facilitando a manutenção e escalabilidade do sistema. Esta divisão clara de responsabilidades também assegura que a plataforma possa evoluir e adaptar-se às necessidades dos utilizadores.

A Figura 4.1 representa um esquema das camadas da arquitetura da plataforma PlanIt, dividido em Frontend, Backend e Base de dados, assim como é realizada a interoperação destas e o contexto no qual o utilizador enquadra-se face a esta arquitetura.

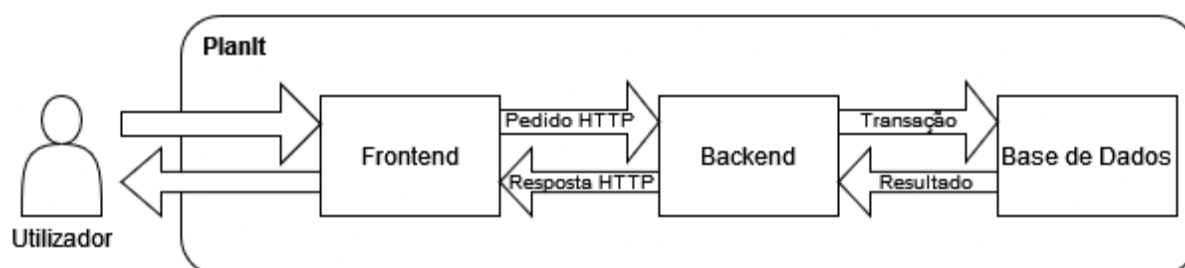


Figura 4.1: Esquema das camadas da arquitetura da Plataforma PlanIt

4.2 Base de Dados

A base de dados é um módulo fundamental em qualquer aplicação moderna. Esta é o repositório central onde todas as informações necessárias são armazenadas e geridas. No contexto de PlanIt, a base de dados desempenha um papel crucial na persistência e gestão de dados relacionados a eventos, utilizadores e outras entidades relevantes.

4.2.1 Estrutura e Funcionamento

Uma base de dados relacional, como PostgreSQL, organiza os dados em tabelas que relacionam-se entre si através de chaves primárias e estrangeiras. Cada tabela armazena dados em linhas e colunas, permitindo uma estrutura organizada e eficiente para consulta e manipulação dos dados. Através da linguagem SQL é possível para executar operações de criação, leitura, atualização e eliminação (CRUD) de dados sobre a base de dados.

4.2.2 Importância da Base de Dados

A base de dados é essencial para garantir a integridade, consistência e disponibilidade dos dados. É importante que uma boa base de dados garanta a persistência, integridade e segurança dos dados. Também é importante que a base de dados permita a escalabilidade da própria e possua um elevado desempenho. Estes aspetos são fundamentais para uma base de dados pelos seguintes motivos:

- **Persistência de Dados:** É necessário armazenar as informações de forma permanente, permitindo que os dados sejam recuperados e utilizados a qualquer momento, independentemente de falhas no sistema ou reinicializações, para que os utilizadores não percam os seus dados.
- **Integridade dos Dados:** Os dados dos utilizadores têm que ser mantidos de forma consistente, isto é, os utilizadores não podem inserir os dados de forma incorreta. Para tal, existem restrições e regras de integridade que garantem que os utilizadores inserem os dados de forma correta.
- **Segurança dos Dados:** Mecanismos de segurança, como a autenticação e a autorização, são vitais para proteger os dados contra acessos não autorizados e vazamentos. Estes mecanismos garantem que apenas utilizadores autorizados possam acessar e modificar os dados.

- **Escalabilidade:** À medida que a plataforma evolui, é crucial que a base de dados possa ser adaptada para integrar novas capacidades e lidar com um volume crescente de dados. A escalabilidade permite que a base de dados se expanda e se ajuste conforme necessário.
- **Desempenho:** Um elevado desempenho na base de dados é necessário para oferecer aos utilizadores tempos de resposta rápidos. Isso melhora a experiência do utilizador e garante a eficiência das operações da aplicação.

4.2.3 Interação com os restantes Módulos

A base de dados PostgreSQL interage diretamente com o backend, que atua como um intermediário entre ela e o frontend. O backend realiza operações CRUD na base de dados em resposta aos pedidos dos utilizadores, aplicando a lógica de negócio e garantindo que todas as interações sejam válidas e seguras.

4.2.4 Implementação

Como foi referido no capítulo 3, o tipo de base de dados que foi escolhida para a plataforma PlanIt foi PostgreSQL. PostgreSQL organiza os dados de forma hierárquica com o objetivo de facilitar o armazenamento, acesso e gestão das informações. A estrutura básica é composta por bases de dados, esquemas e tabelas.

A base de dados é a unidade principal de armazenamento. Cada base de dados em PostgreSQL contém os seus próprios conjuntos de tabelas, esquemas, funções, e outros objetos.

Dentro de uma base de dados, os esquemas ("schemas") são utilizados para organizar os objetos de forma lógica. Um esquema pode ser visto como um contentor que agrupa tabelas, vistas, índices, funções e outros objetos relacionados. Isso permite uma melhor organização e gestão de permissões dentro da base de dados.

As tabelas são onde os dados são efetivamente armazenados. Cada tabela é composta por linhas e colunas, onde cada coluna representa um atributo de uma entidade e cada linha representa um registro. As tabelas são definidas dentro dos esquemas e podem ter restrições como chaves primárias, estrangeiras, índices e regras de integridade para garantir a consistência dos dados.

Para facilitar a visualização e a implementação da base de dados, quem faz a base de

dados geralmente utiliza Modelos "Entidade-Relacionamento"(ER), também conhecidos como Modelos "Entidade-Associação"(EA). O Modelo ER é uma ferramenta fundamental no processo de design de banco de dados, pois permite aos desenvolvedores uma representação visual clara e estruturada de como a base de dados será implementada.

A tabela A.1 demonstra e explica os principais símbolos do Modelo EA.

As entidades, atributos e associações foram criados com o objetivo de dar resposta às necessidades da plataforma e possibilitar o armazenamento dos dados que seriam importantes para as funcionalidades de PlanIt. Todas as entidades do modelo EA possuem identificadores únicos, chamados de "Chaves Primárias", que são utilizados para possibilitar a distinção entre entidades iguais. Tipicamente, este identificador único tem o nome de "id" ou tem "id" no nome. As chaves primárias são atributos das entidades e em certos casos estas chaves são formadas por mais do que um atributo.

A Figura A.1 representa o Modelo EA da plataforma PlanIt.

Entidades

- **User (Utilizador)**

- **Atributos:** *id, name, email, username, description, interests, hashed_password*
- **Descrição:** Armazena as informações dos utilizadores, permitindo as suas autenticações. A sua chave primária é constituída pelo atributo "id". Os restantes atributos correspondem às informações essenciais aos utilizadores, que encontram-se explicadas na secção 2.1. "username" corresponde ao nome de utilizador, "name" ao nome de apresentação, "description" à descrição e "interests" aos interesses. A palavra-passe do utilizador é guardada na base de dados de forma "hashed" por motivos de segurança. Quando é aplicada uma função "hash" sobre um conjunto de caracteres, estes ficam "hashed", o que significa que ficam difíceis de decifrar e por isso fornecem maior segurança.

- **Event (Evento)**

- **Atributos:** *id, title, description, date, end_date, location, locationType, latitude, longitude, visibility, password, code, category, priceCurrency, priceAmount*
- **Descrição:** Entidade central da plataforma, representando os eventos em que os utilizadores podem participar. Armazena os detalhes necessários para descrever e localizar o evento, explicados na secção 2.2. "title" corresponde

ao título, "description" à descrição, "date" à data de começo, "end_date" à data de fim, "location" à morada da localização, "locationType" ao tipo de localização, "latitude" à latitude da morada do evento, "longitude" à longitude da morada do evento, "visibility" à visibilidade, "password" à palavra-passe de eventos privados, "code" ao código de acesso do evento, "category" à categoria, "priceCurrency" à moeda do preço e "priceAmount" ao valor do preço. A chave primária é constituída pelo atributo "id".

- **Role (Papel/Função no Evento)**

- **Atributos:** *id, name*
- **Descrição:** Necessária para definir os papéis que um utilizador pode ter em relação a um evento (participante ou organizador). É uma entidade fraca de "Event" e por isso recebe a chave primária do evento ao qual está associado como atributo. A chave primária de "Role" é o seu atributo "id".

- **Feedback (Opinião)**

- **Atributos:** *id, date, text*
- **Descrição:** Utilizada para coleta as opiniões dos participantes sobre os eventos. A chave primária é o seu atributo "id". Tem uma data (atributo "date") e o texto com a opinião do utilizador (atributo "text")

- **Poll (Votação)**

- **Atributos:** *id, title, created_at, duration*
- **Descrição:** Armazena as informações essenciais das votações, explicadas na secção 2.2. É uma entidade fraca de "Event" e por isso recebe a chave primária do evento à qual está associada como atributo. Tem um título (atributo "title"), data de criação (atributo "created_at") e duração (atributo "duration")

- **Option (Opção)**

- **Atributos:** *id, text*
- **Descrição:** Representa as diferentes opções disponíveis numa votação, essencial para a funcionalidade de votação. É uma entidade fraca de "Poll", pelo que recebe a chave primária da votação à qual está associada como atributo. O atributo "id" é a sua chave primária. Tem o nome da opção (atributo "text").

- **Refresh-Token**

- **Atributos:** *token_validation, expiration_date*
- **Descrição:** Utilizado para a gestão de sessões de utilizador, permitindo manter a sessão contínua e segura. É uma entidade fraca de User, pelo que recebe a sua chave primária como atributo. A chave primária é o atributo "token_validation".

Associações

- **User participates in Event**
 - **Cardinalidade:** N:M
 - **Justificação:** Cada utilizador pode participar em vários eventos, o que resulta numa cardinalidade N:M.
- **User has Role**
 - **Cardinalidade:** N:1
 - **Justificação:** Os utilizadores podem desempenhar apenas 1 papel (role) de cada vez.
- **User has Refresh_Token**
 - **Cardinalidade:** 1:N
 - **Justificação:** Cada utilizador pode ter vários Refresh_Tokens associados.
- **Event has Poll**
 - **Cardinalidade:** 1:N
 - **Justificação:** Cada evento pode ter diversas votações associadas.
- **Poll has Option**
 - **Cardinalidade:** 1:N
 - **Justificação:** Cada votação pode ter várias opções para os participantes votarem.
- **User votes on Poll Option**
 - **Cardinalidade:** 1:N:1
 - **Justificação:** Cada utilizador pode escolher uma opção em cada votação que decidir votar.

4.2.5 Canais de Comunicação (Chats)

Para implementar a funcionalidade de canais de comunicação (chats) entre os utilizadores, foi utilizado a ferramenta Cloud Firestore. A escolha do Firebase para a funcionalidade de chat deve-se às suas capacidades de fornecer atualizações em tempo real, gestão de conexões, escalabilidade e à facilidade que fornece em não necessitar de integração com Backend. No caso específico dos canais de comunicação o comportamento da aplicação funciona como demonstrado na Figura 4.2.

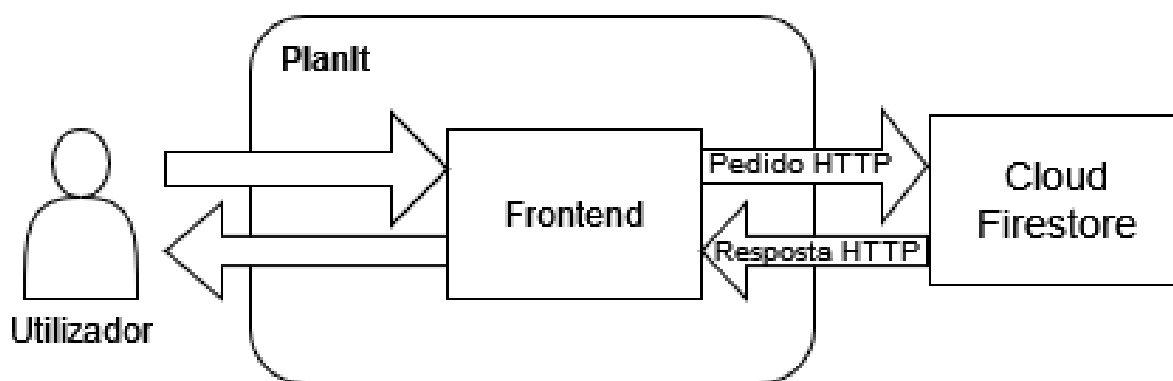


Figura 4.2: Esquema das camadas da arquitetura da Plataforma PlanIt

Na Cloud Firestore, os dados são armazenados em coleções que contêm documentos. Cada documento pode conter subcoleções, que por sua vez também contêm documentos. Estes documentos finais armazenam as informações relevantes. A estrutura de organização da informação na Cloud Firestore pode parecer confusa, mas torna-se clara com uma explicação detalhada.

No contexto de PlanIt é possível chamar a coleção de "eventos". Cada evento (documento) é identificado por um identificador único (id) e contém uma subcoleção de mensagens associadas a ele. Esta subcoleção de mensagens (documentos) armazena detalhes importantes como a data de criação, o nome do remetente, o conteúdo da mensagem e o identificador do utilizador que enviou a mensagem. Essa estrutura permite organizar e acessar os dados de forma eficiente e clara, mantendo todas as informações relacionadas a um evento e as suas mensagens bem estruturadas.

Os eventos e utilizadores associados à base de dados têm os mesmos identificadores da base de dados PostgreSQL da plataforma. Esta gestão é realizada no Frontend que permite definir os identificadores corretos dos eventos e utilizadores aos quais as mensagens estão associadas.

4.3 Backend

O backend é responsável por gerir a lógica de negócio, o processamento de dados e a interação com a base de dados. Implementado utilizando a linguagem de programação Kotlin e a framework Spring MVC, o backend oferece uma série de serviços que são acessados através de APIs RESTful. É neste que são definidas as rotas HTTP através das quais o utilizador pode realizar as ações que pretende na plataforma.

4.3.1 Divisão do Backend

Existem diferentes tipos de necessidades que têm que ser atendidas ao módulo do Backend. Estas necessidades criam uma divisão deste módulo em camadas. Para PlanIt, o Backend foi dividido de acordo com a convenção que tipicamente é utilizada para a divisão de camadas: camada de apresentação, camada de serviços e camada de repositório.

Um cliente é algo capaz de realizar pedidos HTTP para o backend. Pode ser uma aplicação ou um dispositivo que funcione de forma automática ou operada por um humano. No contexto de PlanIt, o Frontend é o cliente, no entanto, durante o processo de testagem foi utilizada a ferramenta "Postman" para realizar os pedidos e realizar a testagem das rotas, pedidos e respostas do Backend.

Para que os clientes possam obter os dados que procuram é necessário realizarem pedidos HTTP para a rota do backend que lhes entrega os dados. Os pedidos são recebidos pelo backend na camada de apresentação. Esta camada é a que está responsável por receber os pedidos, fazer verificações sobre os tipos de dados que recebe nos pedidos e encaminhar de volta para o cliente uma resposta. Ao receber um pedido de um cliente e realizar as correspondentes verificações, a camada de apresentação passa os dados organizados à camada de serviços que é responsável por fazer verificações do tipo de lógica de negócio e também por reencaminhar para a camada de apresentação os dados que receber da camada de repositório de forma organizada. Após realizar as verificações, a camada de serviços vai passar à camada de repositório os dados que necessita para que esta possa realizar as operações CRUD sobre a base de dados que realizam as manipulações correspondentes ao que o pedido HTTP pede. Após realizar as operações, esta camada passa os resultados de volta à camada de serviços.

Resumidamente, o pedido chega à camada de apresentação, onde são extraídos e feitas validações sobre os dados, que são passados à camada de serviços, onde são realizadas verificações de lógica de negócio sobre os dados. Após estas verificações os

dados são transmitidos à camada de repositório onde são realizadas operações sobre o repositório. O resultado destas operações é retornado à camada de serviços, onde são preparados e transmitidos à camada de apresentação. Após receber os resultados já organizados, a camada de apresentação formula uma resposta apropriada com os mesmos e envia-a para o cliente. A Figura 4.3 demonstra este caminho através de um esquema.

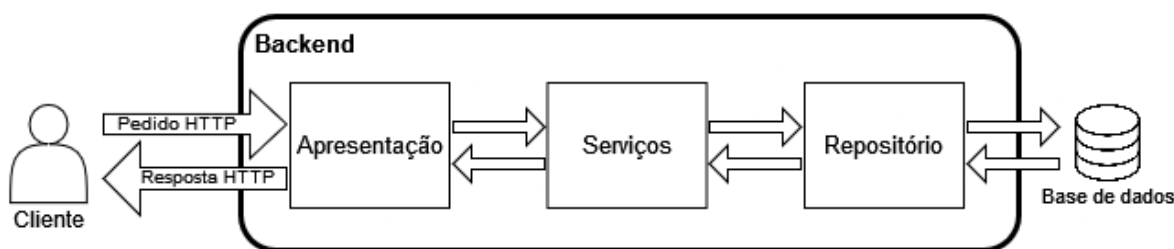


Figura 4.3: Esquema representativo do caminho realizado pelos pedidos dos clientes no Backend

Caso uma das camadas detete algum erro ou problema relativo à função que essa camada possui, então os dados não são enviados para a camada seguinte. Ao invés, o caminho é interrompido e é formulada uma resposta para a camada anterior que indica o problema que foi detetado e por sua vez é enviada ao cliente uma resposta com a informação relativa ao erro.

4.3.2 Autenticação

A autenticação na plataforma PlanIt é fundamental para garantir a segurança e a privacidade dos utilizadores. Para tal, foi implementado um sistema de autenticação que utiliza Access Tokens e Refresh Tokens, geridos por uma classe personalizada desenvolvida internamente.

O processo de autenticação é realizado através de dois tipos de tokens: Access Tokens e Refresh Tokens. Ambos são essenciais para manter a sessão do utilizador segura e eficiente. Os Access Tokens são tokens de curta duração utilizados para autenticar solicitações às APIs protegidas. São gerados por um método interno e incluem informações essenciais sobre o utilizador, como o nome de utilizador (username). Estes tokens têm um tempo de vida limitado, definido por uma configuração interna. Os Refresh Tokens são tokens de longa duração utilizados para obter novos Access Tokens sem a necessidade de o utilizador efetuar login novamente. Internamente, existe um método responsável por gerar um Refresh Token, juntamente com a data de expiração.

A classe UsersDomain, presente na camada de serviços, é responsável pela criação e validação dos tokens. É a classe que contém os métodos internos associados ao Access

e Refresh Token. Para garantir a segurança dos tokens, forma utilizadas chaves secretas e o algoritmo HMAC-SHA512. Além dos tokens, a UsersDomain também lida com a segurança das senhas dos utilizadores, aplicando a função de hash sobre as senhas dos utilizadores para que estas sejam armazenadas de forma segura na base de dados.

No Login e Register caso os dados que o utilizador forneça estejam válidos, o sistema gera um Access Token e um Refresh Token e devolve-os aos utilizador através de uma resposta HTTP.

Este sistema de autenticação, baseado na utilização de tokens e implementado pela classe UsersDomain, oferece uma camada adicional de segurança e eficiência na gestão de sessões dos utilizadores, assegurando que apenas utilizadores autenticados têm acesso às funcionalidades protegidas da plataforma PlanIt.

4.3.3 Spring Framework: Base do Backend

Através da ferramenta Spring Boot da Spring Framework é possível ter acesso a uma base para o backend. O Spring Boot é um projeto da Spring Framework que simplifica a configuração e o desenvolvimento de aplicações standalone, proporcionando uma maneira rápida de iniciar novos projetos. Através de Spring Boot, é possível criar uma aplicação que é executável de forma independente, ou seja, uma aplicação standalone. Esta aplicação possui dependências pré-configuradas e formata automaticamente os componentes necessários com base nas dependências incluídas. Isto facilita a configuração e inicialização do projeto, eliminando a necessidade de configuração manual extensa.

Uma das características principais desta framework é a capacidade de definir `Controllers`, que são responsáveis por implementar as rotas HTTP do backend. Os `Controllers` desempenham um papel crucial na arquitetura do Spring Boot. Estes gerem os pedidos HTTP que chegam ao servidor e formulam as respostas apropriadas. Cada `Controller` é uma classe Kotlin que é anotada com `@Controller` ou `@RestController`, e os seus métodos são mapeados para rotas específicas usando anotações como `@RequestMapping`, `@GetMapping`, `@PostMapping`, `@PutMapping` e `@DeleteMapping`. Estes métodos são utilizados para definir o tipo de pedido HTTP da rota ao qual está associado (GET, POST, PUT, DELETE) e também para definir o URI da rota. Os métodos dos `Controllers` são os pontos onde são recebidos e tratados os pedidos e as respostas relativos à rota que foi definida na anotação.

Antes de chegarem aos `Controllers`, os pedidos HTTP e as respostas atravessam uma cadeia de filtros configurados na aplicação. Estes filtros permitem a execução de

lógica adicional antes e depois do processamento dos pedidos pelos `Controllers`. Eles podem ser utilizados para autenticação, autorização, registro de logs, manipulação de exceções e outras tarefas transversais.

Em suma, Spring Framework fornece a aplicação que dá a base do Backend a PlanIt e possibilita a criação da camada de apresentação através de `Controllers` e `Filters` que permitem definir as rotas da aplicação, receber os pedidos e manipular os mesmos.

4.3.4 Camada de Apresentação

A camada de apresentação do backend, também conhecida como camada de controle ou camada de interface, é uma parte crucial da arquitetura de uma aplicação web. Esta camada é responsável pela gestão da interação entre os clientes (utilizadores finais ou outras aplicações) e os serviços de backend. A função principal da camada de apresentação é receber, processar e responder aos pedidos HTTP feitos pelos clientes.

4.3.4.1 Definição e Função

A camada de apresentação atua como a interface entre o frontend e as outras camadas do backend. Esta camada expõe endpoints HTTP (pontos acessíveis através das rotas HTTP) que permitem aos clientes interagir com a aplicação através de operações CRUD (Create, Read, Update, Delete) e outras operações necessárias para o funcionamento da aplicação.

4.3.4.2 Responsabilidades

As principais responsabilidades da camada de apresentação incluem:

- **Receção de Pedidos HTTP:** A camada de apresentação recebe pedidos HTTP dos clientes. Estes pedidos podem ter o propósito de obter dados, enviar dados, atualizar informações existentes ou eliminar dados.
- **Validação de Dados:** Antes de processar um pedido, a camada de apresentação valida os dados recebidos para garantir que estão no formato correto e atendem aos requisitos esperados. Isto ajuda a prevenir erros e garantir a integridade dos dados.

- **Autenticação e Autorização:** Esta camada também é responsável por garantir que apenas utilizadores autenticados e autorizados possam aceder a certos recursos e realizar determinadas operações. Através de Spring, isto pode ser implementado utilizando `filters` que verificam tokens de autenticação ou credenciais de acesso antes de permitir o acesso aos endpoints.
- **Encaminhamento de Pedidos:** Após a validação e autenticação, a camada de apresentação encaminha as solicitações para a camada seguinte.
- **Formatação de Respostas:** A camada de apresentação também formata as respostas que serão enviadas de volta aos clientes. As respostas geralmente são formatadas em JSON ou XML, proporcionando uma maneira padronizada de transmitir dados.
- **Tratamento de Exceções:** Qualquer erro que ocorra durante o processamento da solicitação é tratado nesta camada. O tratamento de exceções inclui a criação de mensagens de erro apropriadas e o envio de respostas HTTP adequadas, com os devidos erros de código.

4.3.4.3 Implementação da camada de apresentação

A camada de apresentação do Backend da plataforma PlanIt foi implementada fazendo uso da framework Spring MVC. Esta fornece a aplicação standalone responsável por correr o Backend. Nesta aplicação são criados `Controllers` ao definir classes `kotlin` e anotando-as com a anotação `@RestController`, que foi o tipo de controller selecionado para a plataforma. O `RestController` providencia um serviço REST que fornece benefícios em termos de operação da aplicação como caching, redirecionamento e encaminhamento de pedidos e respostas e ainda vantagens em termos de segurança.

Para criar organização na camada de apresentação, esta foi dividida em três `Controllers`: `Event`, `Poll` e `User`. Estes `Controllers`, implementados como `RestControllers`, são separados desta forma para facilitar a estrutura e a manutenção da camada de apresentação.

O `EventController` é responsável pelas operações relacionadas aos eventos. Isso inclui a criação, atualização, eliminação e recuperação de eventos, além de qualquer outra funcionalidade diretamente associada a eventos.

O `PollController` gere as operações relacionadas às votações. Este `Controller` lida com a criação, eliminação e obtenção de votações, garantindo que todas as interações com as votações sejam processadas corretamente.

O `UserController` trata das operações relacionadas aos utilizadores. Este `Controller` é responsável pela gestão de contas de utilizadores, incluindo registo, login e atualização de informações do utilizador.

Esta divisão clara de responsabilidades entre os `Controllers` facilita a manutenção e a escalabilidade do código, permitindo que as funcionalidades específicas sejam geridas de forma isolada e eficiente.

As tabelas A.2, A.3 e A.4 organizam as informações relativas aos `Controllers` com o objetivo de expor as suas funcionalidades. Estas tabelas oferecem uma visão geral dos métodos disponíveis nestes `Controllers`, as suas respectivas rotas e funcionalidades, destacando como cada endpoint contribui para a plataforma.

As rotas (paths) presentes nas tabelas têm um prefixo definido no Backend, permitindo o acesso às funcionalidades do sistema. Esse prefixo é um domínio fornecido pelo método que publica o Backend, sendo necessário para aceder aos métodos dos controllers. As rotas podem conter partes variáveis, identificadas por "{}". Estas partes são usadas para fornecer informações específicas, como identificar recursos individuais.

Métodos como `searchEvents`, que retornam listas, têm parâmetros de rota (queries) como `limit` e `offset`. Estes parâmetros indicam o tamanho máximo da lista a ser retornada e o ponto inicial da lista, respetivamente. No caso específico do método `searchEvents`, é possível definir o parâmetro `searchInput`, utilizado para especificar o evento que o cliente procura.

Todas as informações que não estão presentes na rota que são recolhidas pelos métodos podem ser encontradas no `body` dos pedidos, com exceção dos tokens que podem ser encontrados nos cookies.

Um aspeto importante da implementação é a necessidade de autenticação para aceder a quase todos os métodos, exceto aqueles que estão relacionados com a autenticação (`register`, `login`, `logout`, `refresh-token`). Isto garante que apenas clientes autenticados tenham acesso à plataforma e que os clientes não autenticados consigam ter acesso às funcionalidades de autenticação. A verificação da autenticação é realizada por meio de `filters`.

A plataforma utiliza um filtro, ao qual foi atribuído o nome de `SecurityFilter`, para gerir a autenticação. O `SecurityFilter` é uma classe anotada como um componente Spring (`@Component`). A sua principal função é verificar a autenticidade dos tokens de acesso nos pedidos e garantir que apenas utilizadores autenticados podem aceder a determinados endpoints.

O `SecurityFilter` intercepta os pedidos que são realizados para os endpoints, extrai o token de acesso dos cookies nas rotas que necessitam de autenticação e verifica

se o token é válido. Caso o token seja válido, é colocado no pedido o identificador do cliente que realizou o pedido (atributo `userId`) e este é então redirecionado para o endpoint. No caso das rotas que não necessitam de autenticação é interceptada a resposta gerada pelo endpoint de onde é selecionado o atributo do identificador do utilizador, que é colocado numa cache associado ao seu Access Token. A resposta é depois redirecionada para o cliente. A cache é utilizada para evitar a necessidade de realizar um acesso à base de dados sempre que é necessário validar a autenticação de um cliente, já que é um processo demoroso. Portanto, quando um pedido é interceptado pelo `SecurityFilter`, este faz uso da sua cache para validar a autenticação, ao invés de fazer um acesso à base de dados.

Após serem interceptados pelo `SecurityFilter`, os pedidos são redirecionados para os respectivos métodos do `Controller` (endpoint). Dentro da camada de apresentação, existe uma subcamada dedicada às validações dos dados recebidos por cada `Controller`. Cada `Controller` possui as suas próprias validações, denominadas de "validações de domínio". Essas validações têm o propósito de separar e delegar claramente as diferentes responsabilidades dentro do backend, além de identificar erros nos dados antecipadamente, evitando que alcancem camadas subsequentes. Isto permite retornar ao cliente uma resposta apropriada com o erro identificado.

Por exemplo, no caso de um pedido de criação de evento, os detalhes do evento são enviados no `body` da mensagem. A subcamada de validações verifica a validade desses detalhes, como assegurar que a categoria especificada pelo cliente é válida na plataforma. Se a categoria não for válida, é gerada uma exceção indicando ao método `createEvent` que a categoria selecionada não é permitida. O método `createEvent`, então, produz uma resposta de erro informando o cliente sobre a invalidade da categoria selecionada. Assim, os erros são detectados e uma resposta é gerada antecipadamente para o cliente. Todos os métodos de todos os `Controllers` seguem este procedimento em relação às validações de domínio.

Após as validações, o método trata de encaminhar os dados validados e organizados para a camada de serviços. A camada de serviços por sua vez encaminhará de volta para o método os dados da resposta. O método fica responsável de preparar e submeter ao `Controller` a resposta que será transmitida ao cliente pelo próprio, sendo antes interceptada pelo `SecurityFilter`. O caminho percorrido por um pedido feito ao Backend pode ser observado no esquema da Figura 4.5

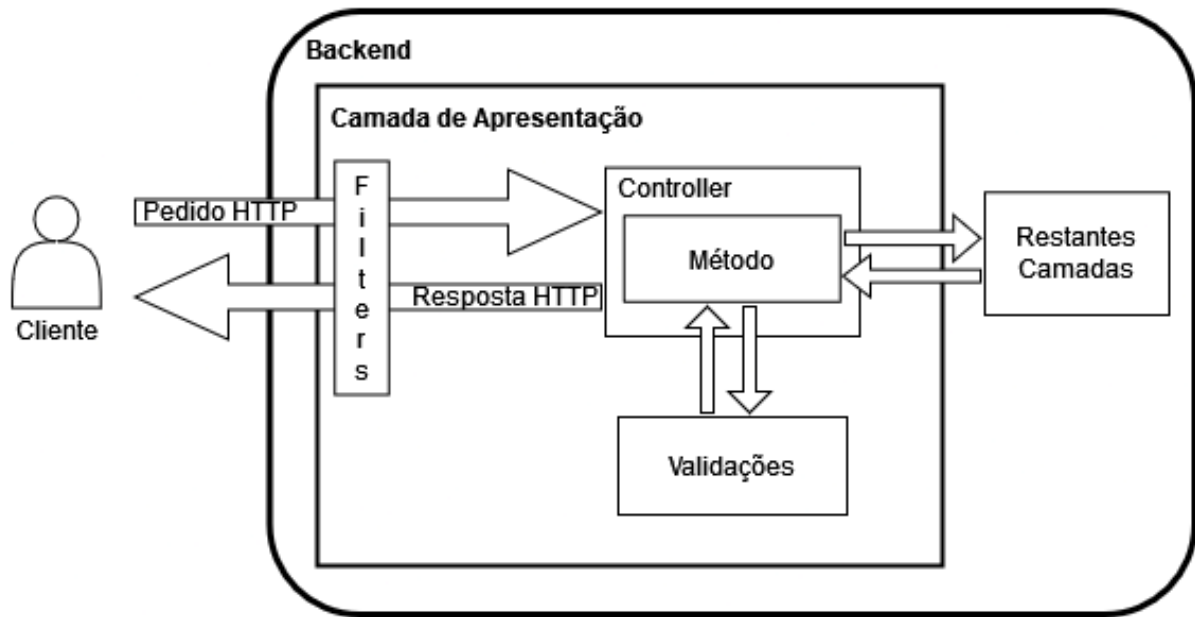


Figura 4.4: Esquema da Camada de Apresentação

Exemplo de um pedido HTTP de criação de evento para o método `createEvent`:

```

POST /event HTTP/1.1
Host: example.com
Content-Type: application/json
Authorization: Bearer <access_token>

```

```

{
  "title": "Event Example",
  "description": "This is an event example.",
  "category": "Arts and Culture",
  "locationType": "Online",
  "location": "www.zoom.com",
  "visibility": "Public",
  "date": "2024-12-31 00:01",
  "endDate": "2024-12-31 23:59",
  "price": "00.00E",
  "password": ""
}

```


4.3.5 Camada de Serviços

A camada de serviços é um componente fundamental na arquitetura de software, responsável por encapsular a lógica de negócios e orquestrar a comunicação entre diferentes partes do sistema. Esta camada atua como intermediária entre a camada de apresentação (frontend) e a camada de repositório.

A camada de serviços, também conhecida como camada de lógica de negócios, tem como principal objetivo centralizar e gerir as regras e operações do negócio. Esta camada recebe os dados dos pedidos vindos da camada de apresentação, processa-os conforme as regras de negócios e entrega-os à camada de dados caso estes estejam de acordo com a lógica. Caso os dados não sigam a lógica de negócios da plataforma, então é gerada uma exceção que é enviada para a camada de apresentação, para que esta saiba qual o problema com os dados.

A lógica de negócios é responsável por garantir que as operações realizadas no sistema sigam as regras estabelecidas pela plataforma. Estas regras podem incluir políticas, procedimentos, cálculos e outras operações que refletem os objetivos e práticas da plataforma.

A camada de serviços é representada através de uma classe Kotlin e está diretamente associada aos Controllers na medida que para cada Controller existe uma classe Kotlin da camada de serviços, com os métodos aos quais os métodos dos Controllers recorrem. Existem ainda métodos privados dentro da classe, mas que apenas podem ser utilizados por outros métodos da própria classe. A Figura 4.5 representa a camada de serviços e a sua interação com as restantes camadas.

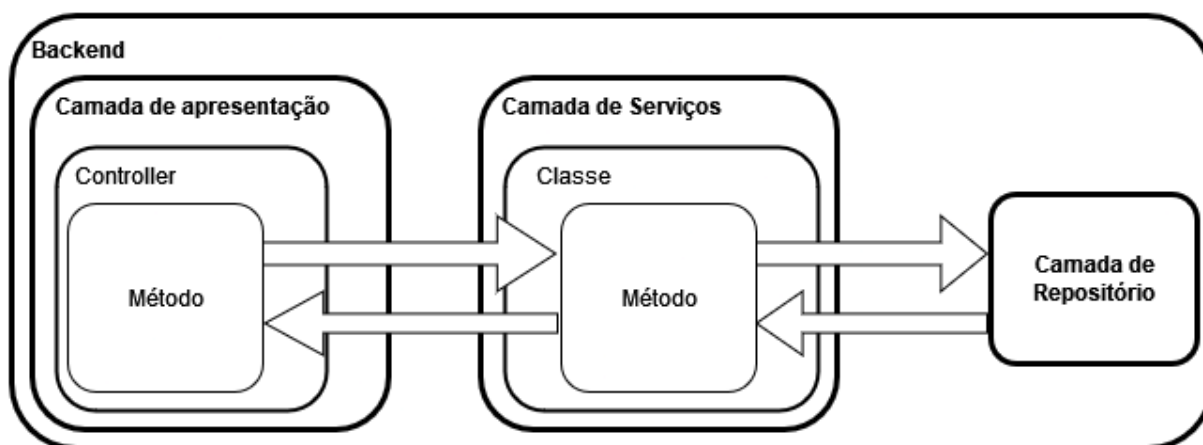


Figura 4.5: Esquema da Camada de Serviços e a interação com as restantes camadas

É comum a camada de serviços interagir diversas vezes com a camada de repositório já

que é necessário obter certos dados da base de dados para se poder fazer confirmações da lógica de negócios.

Através do exemplo do método de registo da camada de serviços, que está associado ao método Register da camada de apresentação, é possível compreender melhor o funcionamento da camada de serviços. Este método recebe os dados que o utilizador forneceu (nome de apresentação, username, email e palavra-passe) e realiza verificações da lógica de negócios sobre estes dados. Isto é, de acordo com a lógica da plataforma, o username e o email são únicos, logo é necessário realizar acessos à base de dados antes de se criar o utilizador para verificar-se se não existem outros utilizadores com o username ou email que o utilizador forneceu. Isto é um exemplo da lógica de negócios e de como é necessário muitas vezes realizar vários acessos à base de dados na camada de serviços. Estes acessos são realizados através da chamada de métodos da camada de repositório.

4.3.6 Camada de Repositório

A camada de repositório é um componente crucial na arquitetura de software, responsável pela comunicação entre a aplicação e a base de dados. Esta camada implementa operações de criação, leitura, atualização e eliminação (CRUD) de dados de maneira segura para responder ao que é pretendido pelo pedido do utilizador.

A camada de repositório é um conjunto de classes Kotlin que utilizam a ferramenta JDBI para aceder à base de dados. A organização desta camada está dividida como as outras camadas do Backend: Event, User e Poll. Esta divisão permite estruturar de forma mais clara a camada e integrar-se com a organização da camada de serviços.

A camada de repositório faz uso do *JDBI Transaction Manager*, que facilita o gerenciamento de transações. Utilizando o *Transaction Manager*, é possível garantir que todas as operações dentro de uma transação são executadas de forma atômica, ou seja, todas são bem-sucedidas ou todas falham, mantendo a integridade dos dados.

4.4 Frontend

Como foi previamente mencionado, o Frontend é a parte da aplicação com a qual o utilizador interage seja através de um navegador ou de um dispositivo móvel. Como se constata na figura 4.6, o cliente tem a possibilidade de escolher com qual das aplicações interage, sendo que ambas se conectam ao backend. Esta camada possui a responsabilidade de apresentar os dados pretendidos ao utilizador, assegurando que as

interações do mesmo sejam transmitidas corretamente para o backend. Nesta secção, serão abordadas a estrutura do código, a navegabilidade e outros aspetos relacionados com a interface tanto na web como em dispositivos Android.

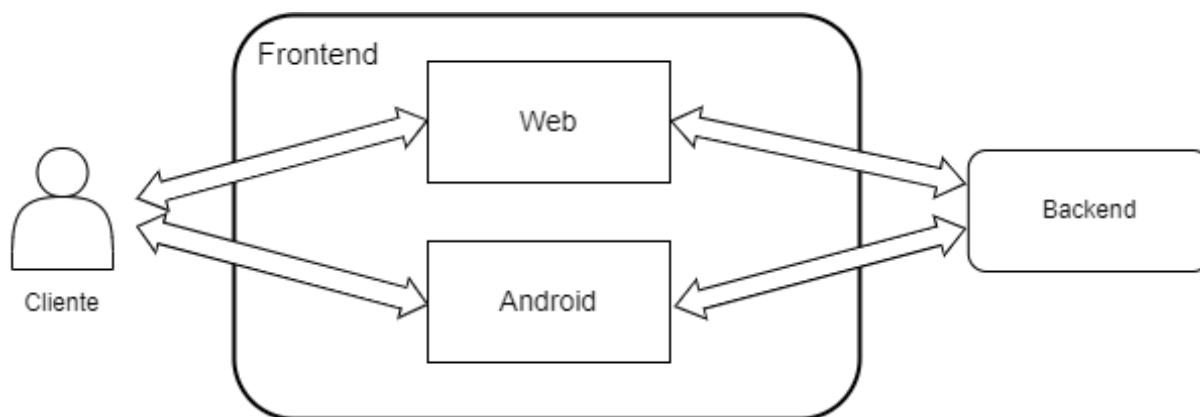


Figura 4.6: Esquema do frontend

4.4.1 Web

Na arquitetura do frontend para a Web, a estrutura é dividida em duas principais áreas: Components e Services, como é possível observar na Figura 4.7.

Os Components compõem a parte visual da aplicação, sendo assim responsáveis pela construção da interface do utilizador. Cada componente representa uma parte específica da interface, desde botões até elementos mais complexos, como listas ou até um calendário. Assim sendo, cada componente possui a sua estrutura em HTML e CSS e a respetiva lógica em TypeScript. Esta lógica engloba a manipulação dos dados provenientes dos Services e a atualização dinâmica do estado do componente consoante as respetivas necessidades.

Por outro lado, os Services são responsáveis por a comunicação entre o frontend e a API, agindo como intermediários entre os componentes e os recursos fornecidos pela API.

Ao receber os pedidos de cada componente derivados da interação do utilizador com o mesmo, os Services realizam chamadas à API para recuperar os dados necessários e retorná-los aos componentes correspondentes. Para além disso, estes serviços também lidam com a gestão de estado e a manipulação de erros.

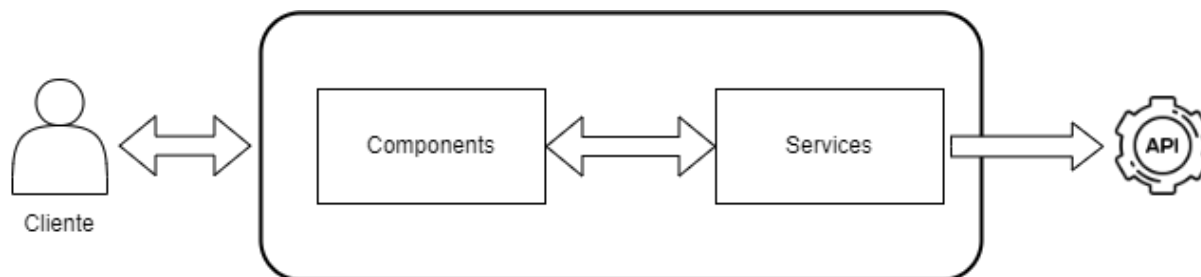


Figura 4.7: Esquema das camadas da aplicação Web

4.4.2 Android

No frontend em Android, a arquitetura adotada foi do tipo MVVM (Model - View - ViewModel) (figura 4.8) com o intuito de separar as responsabilidades entre os diferentes elementos da aplicação e evitar dependências demasiado “fortes”. Em primeiro lugar, a camada View é a responsável pela UI da aplicação, sendo composta pelas Screens e pelas Activities. Esta apenas exibe os dados necessários e, ao captar as interações do utilizador, reencaminha-as para o Viewmodel. Assim sendo, não armazena dados e não tem conhecimento da camada Model. O Viewmodel atua como ponte entre a View e o Model, para além de deter o estado do ecrã do utilizador e ser o encarregado de obter os dados a produzir para as Screens. Por fim, a camada Model encontra-se dividida em duas partes: Domain e Services. O Domain consiste nas entidades e nos modelos de dados que a aplicação utiliza e os Services, à semelhança da Web, realizam a interação com a API, enviando e recebendo dados através das diversas operações CRUD.

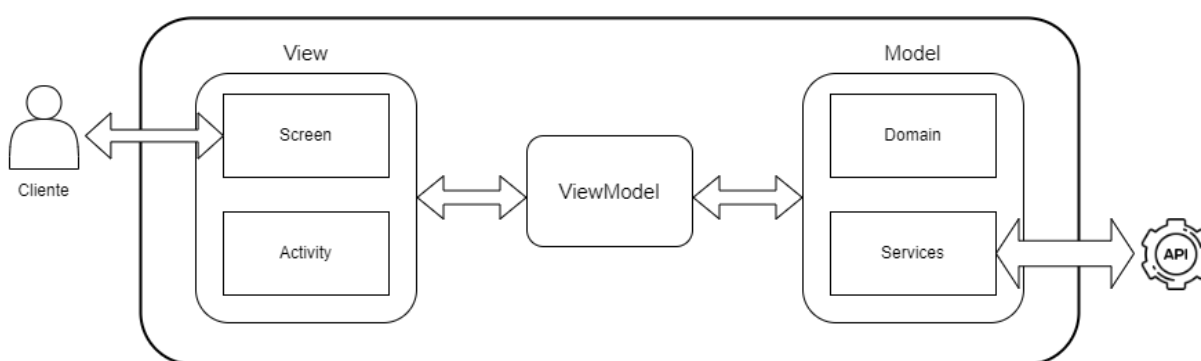


Figura 4.8: Esquema das camadas aplicação Mobile

4.4.3 Navegabilidade

Nesta subsecção, será abordada a forma como se navega e os destinos de navegação a partir de cada componente ou página, tanto na web como em dispositivos móveis.

Serão detalhados os fluxos de navegação, as rotas disponíveis e as interações que permitem ao utilizador mover-se entre diferentes partes da aplicação. Abaixo encontra-se a descrição dos componentes que compõem ambas as aplicações:

1. **Guest Home:** Ponto de entrada para utilizadores não autenticados.
2. **Login:** Página para autenticação de utilizadores já autenticados na plataforma.
3. **Register:** Página para registo de novos utilizadores na plataforma.
4. **Events:** Página que lista os eventos pesquisados e as categorias disponíveis.
5. **Create Event:** Popup que possibilita a criação de um novo evento.
6. **Event:** Página detalhada de um evento em específico.
7. **My Events:** Página onde se encontram os eventos criados pelo utilizador bem como aqueles em que participa.
8. **My Details:** Página com os detalhes e informações pessoais do utilizador.
9. **Edit Details:** Página do Mobile onde o utilizador pode editar as suas informações..
10. **User Profile:** Página com os detalhes de outro utilizador que participe no mesmo evento.
11. **View Polls:** Popup para visualizar as votações disponíveis relativas ao evento em questão.
12. **View Poll:** Popup para visualizar uma votação, bem como os seus detalhes (votos e duração).
13. **Create Poll:** Popup que permite criar uma votação, definindo a sua duração e as opções disponíveis.
14. **Feedback:** Popup que permite o envio de feedback acerca da plataforma.
15. **NearMe:** Página que permite ao utilizador encontrar eventos próximos à sua localização.
16. **Calendar:** Página que fornece ao utilizador um calendário com os eventos agendados.

17. **Chat:** Possibilita a comunicação entre os participantes. Apesar de não ter uma página própria, é um componente importante em ambas as aplicações.

O diagrama de navegabilidade correspondente à aplicação Web encontra-se representado na figura 4.9, detalhando as principais páginas e as suas interconexões.

O fluxo de navegação começa na página Guest Home(1), com opções de login(2) e register(3). Após a autenticação, o utilizador é direcionado para a página Events (4), onde encontram-se exibidos todos os eventos criados até o momento. Para além disso, existe um menu ou barra de categorias, que permite ao utilizador filtrar os eventos de acordo com a categoria pretendida. Este ainda pode criar novos eventos (5) ou seleccionar um evento para ver os seus detalhes (6). Na página Event, o utilizador pode aceder a perfis de outros utilizadores (10), ver e criar votações (11 e 13) e ainda tem acesso ao chat respetivo do evento (17). Para além disso, as opções My Events (7), My Details (8), Feedback (14), NearMe (15), e Calendar (16) são acessíveis a partir de qualquer página através de uma barra de navegação.

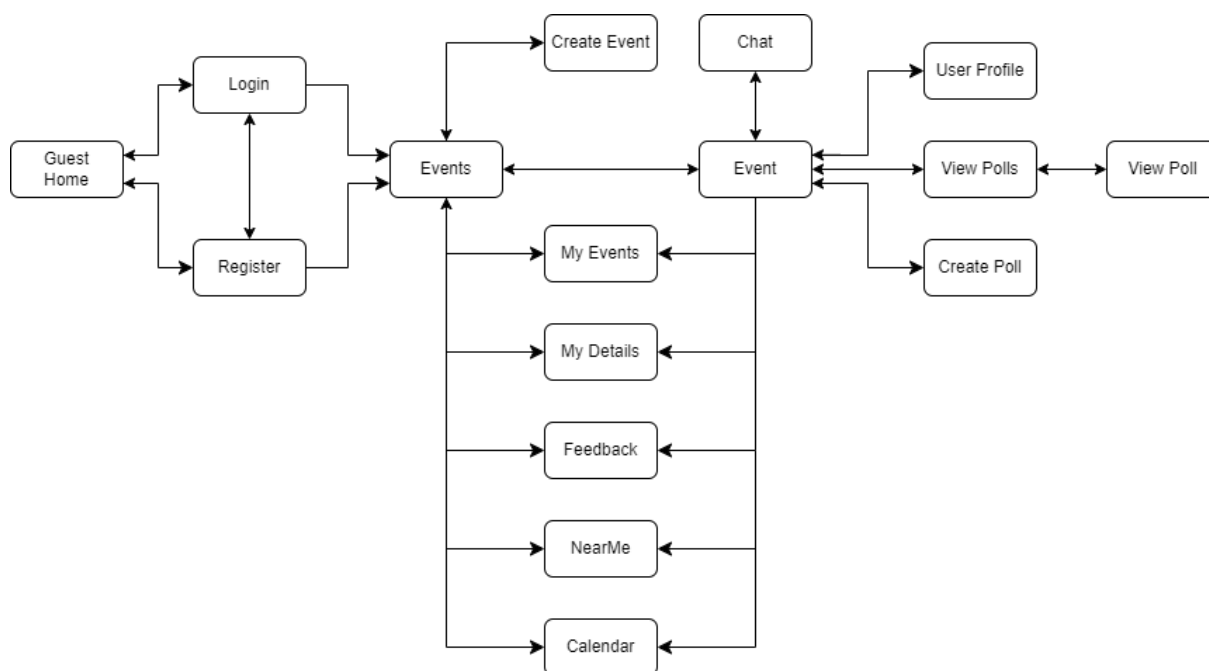


Figura 4.9: Diagrama de Navegabilidade - Web

Da mesma maneira, segue-se o diagrama de navegabilidade da aplicação Mobile, representado na figura 4.10,

Tal como na Web, a autenticação inicial inicia-se com a página Guest seguida de autenticação por login ou register. Posto isto, o utilizador é reencaminhado para a página Home onde tem a possibilidade de navegar para os Events (4) ou para os My Details

(8). Esta navegação é feita nos dois sentidos visto que é realizada através de uma barra de navegação. Ainda na Home, o utilizador pode criar um evento (5), navegar para a funcionalidade NearMe (15), ver os eventos em que participa (7) e, por fim, ainda pode aceder ao seu calendário (16). No componente My Details, o utilizador tem a opção de ser redirecionado para uma página onde pode editar os seus detalhes (9) ao contrário da Web, onde o utilizador edita informação diretamente na sua página. O fluxo de navegação a partir da página Event (6) é idêntico ao da Web.

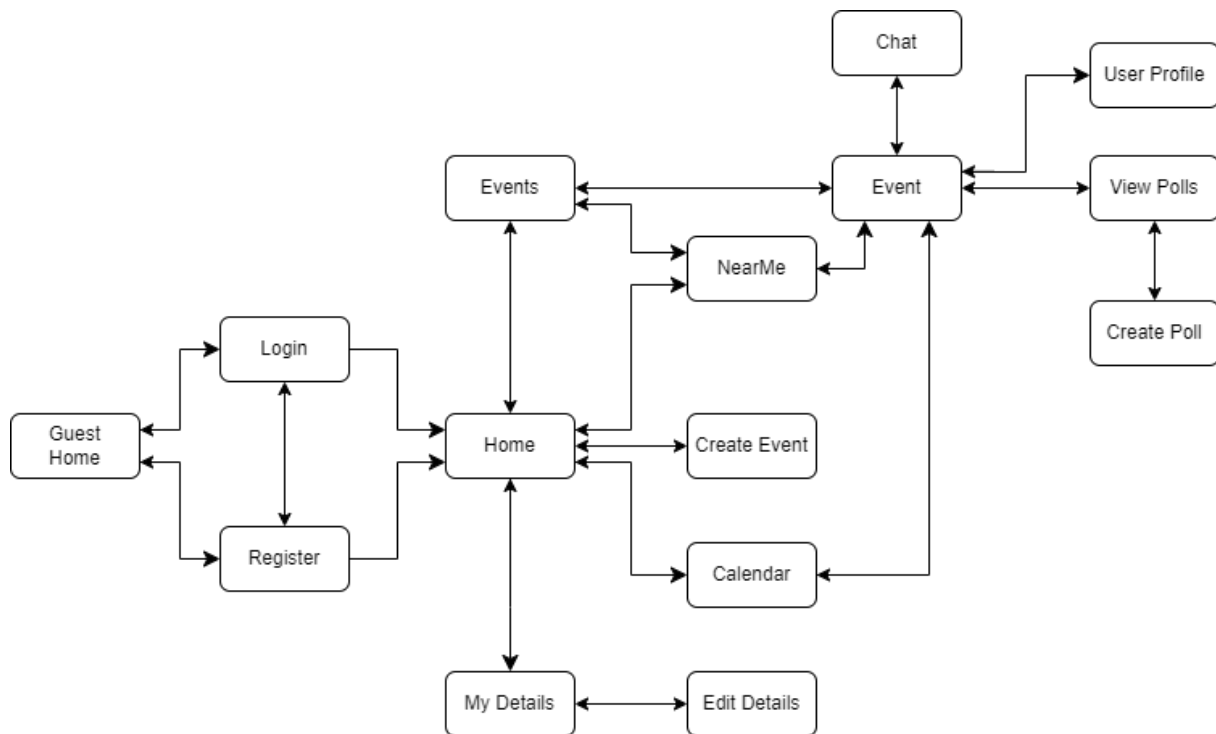


Figura 4.10: Diagrama de Navegabilidade - Mobile

É ainda importante referir que caso o utilizador deixe de estar autenticado, este será automaticamente redirecionado para o componente Guest Home, em ambas as aplicações.

4.4.4 Google Maps API

A Google Maps API foi integrada nas aplicações Web e Android de modo a permitir funcionalidades de localização essenciais ao projeto. Esta API foi utilizada principalmente para duas funcionalidades: selecionar a localização de um evento e a funcionalidade NearMe.

A API do Google Maps foi utilizada para permitir que os utilizadores selecionem a localização exata de um evento no mapa. Este recurso facilita a escolha da localização

do evento, permitindo que o utilizador visualize no mapa o local onde pretende criar o evento.

A funcionalidade NearMe, que permite aos utilizadores encontrar eventos próximos à sua localização atual, também se baseia na Google Maps API. Através desta, foi possível calcular distâncias e exibir no mapa os eventos que estão dentro de um raio definido pelo utilizador. Muito sucintamente, através de geolocalização são retornadas as coordenadas da localização do utilizador que, com base num raio definido pelo mesmo, são utilizadas para calcular a distância entre o utilizador e cada evento na proximidade.

Para complementar a utilização da Google Maps API, foi integrada a Geocoding API. Esta API permite converter coordenadas geográficas em endereços (geocodificação inversa) e endereços em coordenadas geográficas (geocodificação). Este recurso foi essencial para exibir os endereços escolhidos pelos utilizadores após selecionarem a localização de um evento no mapa e para converter endereços fornecidos pelos mesmos em coordenadas geográficas, com o objetivo de serem processadas e exibidas no mapa.

4.4.5 Google Calendar API

Esta API foi utilizada nas aplicações para dar a possibilidade de integrar eventos no Google Calendar dos utilizadores. Esta integração permitiu adicionar os eventos da aplicação ao Google Calendar, caso o utilizador desejasse ter acesso às informações destes eventos na aplicação da Google. Para tal, é necessário que o utilizador se autentique com a sua conta Google e que aceite as permissões necessárias.

4.4.6 Chat

O chat foi implementado dando uso à Firestore. Esta funcionalidade permite que os utilizadores se comuniquem em tempo real dentro dos eventos, tentando ao máximo proporcionar uma experiência fluída. O sistema de chat está integrado na página de detalhes do evento, permitindo aos participantes do evento trocar mensagens sem sair da interface principal. A receção de mensagens é feita com base na ordem de criação de cada documento no Firestore. Sendo assim, sempre que há uma alteração na base de dados do Firestore, são retornadas 100 mensagens de cada vez, estando estas ordenadas por ordem de data de envio. O envio de mensagens faz-se através de uma função que adiciona novas mensagens (documentos) à coleção correspondente no Firestore. Estas mensagens são então armazenadas no Firestore e imediatamente disponibilizadas para todos os utilizadores que estão a visualizar o chat.

5

Enquadramento

5.1 Conceptualização

A inspiração para PlanIt surgiu de uma frustração compartilhada pelos fundadores - a dificuldade de organizar e participar em eventos. Coordenar agendas, encontrar atividades que atendessem aos interesses de todos, ou simplesmente superar a inércia de fazer planos eram tarefas frequentemente desanimadoras. Ao explorarem ferramentas e plataformas existentes no mercado da promoção e descoberta de eventos, perceberam que essas soluções são muito específicas, voltadas para tipos específicos de eventos ou atividades, ou careciam a integração e abrangência desejadas.

Reconhecendo esta lacuna no mercado, idealizou-se PlanIt - uma plataforma única que atenderia às diversas necessidades dos organizadores e participantes de eventos. PlanIt não apenas forneceria um local centralizado para descobrir e divulgar eventos, mas também facilitaria a comunicação e colaboração entre os participantes. Através destas propostas PlanIt distinguir-se-ia das soluções existentes, permitindo aos utilizadores conectarem-se, envolverem-se e aproveitarem ao máximo as funcionalidades que procuram.

5.2 Casos de Uso

Os eventos são o centro da plataforma, e através da definição destes, é possível perceber que estes transmitem uma grande versatilidade à forma como a plataforma é utilizada.

5.2.1 Divulgação e Pesquisa de Eventos

Os principais casos de uso da aplicação são a divulgação e pesquisa de eventos. Através da criação de eventos é possível disponibilizá-los aos restantes utilizadores da aplicação, que através de critérios de pesquisa, como a pesquisa direta, a filtragem por categorias ou eventos próximos, conseguem encontrar eventos que desejam atender. Desta forma, os utilizadores são capazes de divulgar os seus eventos e também são capazes de procurar por eventos, caso assim o queiram.

A divulgação dos eventos pode ser feita de forma pública ou privada, permitindo que sejam acessíveis a todos os utilizadores ou restritos apenas àqueles com acesso autorizado, respetivamente. Ao privatizar os eventos, os organizadores restringem o acesso à informação relativa aos eventos apenas aos utilizadores que se juntarem através de um código de convite ou palavra-passe.

5.2.2 Calendário Pessoal

Um dos casos de uso que é fornecido à plataforma é a utilização da plataforma como calendário pessoal. Pela definição de evento, o utilizador é capaz de criar eventos privados, aos quais apenas este tem acesso e que pode ver, através de uma vista de calendário da plataforma, os eventos nos quais está inscrito.

5.3 Produtos Semelhantes

Como já foi referido no documento, existem diversas ferramentas disponíveis no mercado que realizam funcionalidades que se encontram integradas em PlanIt. No entanto, estas ferramentas estão fragmentadas em diferentes locais, pelo que PlanIt centraliza-as todas numa só plataforma. Esta secção foca-se em produtos semelhantes a PlanIt, que possuem essas mesmas ferramentas.

5.3.1 Eventbrite

Eventbrite é uma plataforma bem estabelecida que permite aos utilizadores descobrirem e criarem eventos. A sua principal força reside na capacidade de facilitar a organização de eventos públicos, como conferências, workshops, concertos e outros eventos de grande escala. Eventbrite oferece ferramentas robustas para a venda de bilhetes, gestão de inscrições e promoção de eventos através de várias integrações de marketing. No entanto, a plataforma é predominantemente voltada para eventos comerciais e públicos, o que a torna menos adequada para eventos privados ou comunitários de menor escala.

5.3.2 Meetup

Meetup é uma plataforma focada na organização de encontros comunitários e grupos com interesses comuns. Através de Meetup, os utilizadores podem criar grupos baseados em interesses específicos e organizar encontros regulares. A plataforma é excelente para construir comunidades e promover interações sociais entre pessoas com interesses semelhantes. Esta plataforma é bastante semelhante a PlanIt, no entanto, embora eficaz para eventos comunitários e informais, Meetup carece de funcionalidades robustas para a gestão de eventos privados.

5.3.3 Google Calendar

Google Calendar oferece funcionalidades para criar e gerir eventos pessoais e profissionais. A sua integração com outros serviços Google e a sincronização em vários dispositivos tornam-no numa ferramenta poderosa para a gestão de agendas. Google Calendar permite a criação de eventos, envio de convites, definição de lembretes e partilha de calendários com outras pessoas. No entanto, a plataforma não é especificamente uma ferramenta de descoberta de eventos e carece de funcionalidades sociais e de promoção robustas, o que limita a sua utilidade para eventos públicos e comunitários.

5.3.4 Asana

Asana é uma plataforma de gestão de projetos que oferece ferramentas abrangentes para a organização e coordenação de tarefas e eventos. Utilizada amplamente em ambientes empresariais, Asana permite a criação de projetos, atribuição de tarefas, definição de prazos e acompanhamento do progresso. A sua flexibilidade permite que seja

utilizada para a organização de eventos, especialmente aqueles que requerem coordenação detalhada e gestão de múltiplos participantes e atividades. No entanto, Asana não é uma plataforma dedicada à descoberta de eventos, e a sua complexidade pode ser excessiva para utilizadores que procuram apenas uma ferramenta simples para promover ou descobrir eventos.

5.4 Requerimentos

Para garantir que os utilizadores possam consultar a plataforma PlanIt de forma contínua e eficiente, vários requisitos devem ser atendidos. Estes requerimentos são essenciais para assegurar que os utilizadores possam usufruir plenamente de todas as funcionalidades oferecidas pela plataforma PlanIt, quer seja através da web ou de dispositivos móveis.

5.4.1 Acesso à Internet

Dado que PlanIt é uma plataforma virtual, a sua disponibilização e consulta dependem de uma conexão estável à internet. Portanto, é imprescindível que os utilizadores tenham acesso à internet para utilizar todas as funcionalidades da plataforma.

5.4.2 Acesso via Web

Para utilizar a versão Web da plataforma, é necessário que os utilizadores possuam um navegador (browser) compatível. Embora seja possível aceder à plataforma sem o uso de um navegador, esta prática seria árdua e dificultosa, pois o conteúdo da plataforma é otimizado para ser interpretado por navegadores, pelo que torna-se difícil de ser interpretado por humanos e por consequência torna complexa a interação e navegação pela plataforma.

5.4.3 Acesso via Mobile

Para ter acesso à versão móvel da plataforma, os utilizadores necessitam possuir um dispositivo Android compatível. A instalação da aplicação móvel requer a capacidade de descarregar e instalar o pacote de instalação (APK) da plataforma no dispositivo.



Conclusão

6.1 Conclusões

Neste documento, introduzimos a plataforma PlanIt e abordamos os desafios associados à divulgação de eventos. Identificamos a necessidade de uma solução web que integre ferramentas existentes e proporcione a capacidade de divulgar de eventos.

A plataforma PlanIt foi desenvolvida para atender a esses requisitos, oferecendo uma interface interativa e funcionalidades avançadas para a divulgação e procura de eventos. Com uma arquitetura modular da plataforma, que integra as funcionalidades de gerenciamento de eventos e comunicação num ambiente coeso e escalável, é possível garantir uma experiência de utilizador contínua e possibilita a manutenção e expansão futura da plataforma.

Em resumo, o desenvolvimento da plataforma PlanIt, com a sua arquitetura modular e funcionalidades, representa um avanço significativo no campo da divulgação de eventos. Ao fornecer uma plataforma web amigável e aplicação Android, esta nova ferramenta capacitará organizadores e participantes a poderem divulgar e procurar eventos de forma mais eficaz, promovendo a comunicação e a colaboração. Com isto, PlanIt poderá contribuir para a realização e divulgação de eventos mais bem-sucedidos e produtivos, assim como providenciar aos utilizadores a possibilidade de conseguirem encontrar os eventos que desejam encontrar.

6.2 Planos Futuros

À medida que a plataforma PlanIt continua a evoluir, temos várias iniciativas planejadas para aprimorar ainda mais a funcionalidade e a experiência do usuário. Estas iniciativas visam não apenas melhorar a segurança e a usabilidade, mas também expandir as capacidades sociais da plataforma. Abaixo destacamos os principais planos futuros para a plataforma PlanIt:

Gestão de Tokens na Cache: Atualmente, a plataforma PlanIt utiliza tokens de acesso e atualização para autenticação e autorização dos usuários. Um dos planos é no futuro implementar um mecanismo para remover tokens da cache após um determinado período de inatividade. Esta medida visa melhorar a segurança ao garantir que tokens antigos ou não utilizados sejam automaticamente invalidados, reduzindo o risco de acesso não autorizado.

Navegação Não Autenticada: Para tornar a plataforma mais acessível e atrativa para novos utilizadores, pretende-se possibilitar a navegação por determinadas secções da plataforma sem a necessidade de autenticação. Isto permitirá que utilizadores explorem funcionalidades básicas e obtenham uma visão geral da plataforma sem a necessidade de estarem autenticados. A navegação não autenticada incluirá acesso a informações públicas sobre eventos, pesquisa de eventos e descrições gerais das funcionalidades da PlanIt.

Funcionalidades Sociais: Outro plano futuro para PlanIt é explorar desenvolver a plataforma num sentido mais social, permitindo uma interação mais direta entre os utilizadores. As funcionalidades sociais que poderiam melhorar a plataforma incluem:

- **Seguir Outros Utilizadores:** Os utilizadores poderão seguir outros utilizadores, criando uma rede social dentro da plataforma. Isso permitirá que os utilizadores mantenham-se atualizados sobre as atividades dos seus amigos e contactos.
- **Eventos Públicos de Amigos:** Possibilitar visualizar os eventos públicos onde os amigos estão inscritos. Esta funcionalidade incentiva a participação em eventos populares dentro da rede de contactos do utilizador, promovendo uma maior interação social.
- **Canais de Comunicação Privados:** A implementação de canais de comunicação privados, permite que os utilizadores conversem diretamente uns com os outros. Esta funcionalidade proporcionará uma comunicação mais eficaz e imediata, facilitando a organização de eventos e a troca de informações relevantes.

- **Canais de Comunicação para Organizadores:** Para que os organizadores de eventos consigam comunicar entre si conteúdo que diz respeito apenas a organizadores seria interessante dentro dos eventos haver para além do canal de comunicação do evento, existir ainda outro canal apenas para organizadores.
- **Procura de Utilizadores com Interesses Similares:** Permitir que os utilizadores procurem outros com interesses parecidos pode não só aumentar a interação social entre os utilizadores, mas também ajudar a formar comunidades e a promover eventos de maneira mais eficiente.

Ao integrar estas funcionalidades sociais, PlanIt tem o potencial de se tornar numa plataforma mais interativa e colaborativa, atendendo às necessidades de comunicação dos utilizadores de maneira mais abrangente. Estas melhorias não só podem aumentar o envolvimento dos utilizadores na plataforma, mas também fortalecer a comunidade dentro da plataforma, tornando PlanIt num "hub" central para a divulgação de eventos e interações sociais.



Referências

Referências bibliográficas:

Allied Market Research. "Event Management Software Market by Component (Software (Event Registration and Ticketing Software, Event Planning Software, Event Marketing Software, Venue Management Software, Analytics and Reporting Software, and Others) and Services), Deployment Mode (On-Premise and Cloud), Organization Size (Small & Medium Enterprises and Large Enterprises), and End User (Event Organizers and Planners, Corporate, Government, Education, and Others): Global Opportunity Analysis and Industry Forecast, 2021-2030."

Event Manager Blog. "The State of Event Management Technology Report 2023."

Referências web:

<https://www.geeksforgeeks.org/introduction-of-er-model/>

<https://www.meetup.com/>

<https://www.eventbrite.pt/>

https://en.wikipedia.org/wiki/Google_Calendar

<https://asana.com>

<https://kotlinlang.org/>

<https://spring.io/>

<https://gradle.org/>

<https://react.dev/>

<https://firebase.google.com/docs/firestore?hl=pt-br>

https://en.wikipedia.org/wiki/Uniform_Resource_Identifier

<https://www.typescriptlang.org/>

<https://webpack.js.org/>

<https://developers.google.com/maps/documentation?hl=pt-br>

<https://learn.microsoft.com/pt-pt/dotnet/architecture/maui/mvvm>

<https://developers.google.com/maps/documentation/geocoding/overview?hl=pt-br>

<https://developers.google.com/maps/documentation/javascript?hl=pt-br>

<https://developers.google.com/calendar/api/guides/overview?hl=pt-br>



Anexos







Figura	Símbolo	Explicação
	Entidade	Representam objetos ou conceitos do mundo real que possuem existência independente dentro do domínio da aplicação
	Atributo	Propriedades das entidades
	Relação/ Associação	Relação/ Associação entre entidades
	Ligação de símbolos	Identificam a relação/ associação entre entidades e identificam o seu tipo.
	Atributos multi-variados	Atributos com mais do que um valor para uma dada entidade
	Entidade fraca	Entidades que dependem de outras entidades

Tabela A.1: Símbolos do Modelo EA

Método	Rota (Path)	Tipo de Pedido	Descrição
register	/register	POST	Regista um novo utilizador com base nos dados fornecidos no pedido.
login	/login	POST	Realiza o login de um utilizador com os dados fornecidos.
logout	/logout	POST	Realiza o logout do utilizador autenticado.
getUser	/user/{pathId}	GET	Fornece os detalhes do utilizador identificado pelo ID.
getUserEvents	/user/events	GET	Retorna a lista de todos os eventos em que o utilizador participa.
editUser	/user	PUT	Edita os detalhes do utilizador com base nos dados fornecidos pelo mesmo.
assignRole	/user/{userId}/event/{eventId}/role	POST	Atribui um papel a um utilizador num evento específico.
removeRole	/user/{userId}/event/{eventId}/role/{roleId}	DELETE	Remove o papel de um utilizador num evento específico.
getUserRole	/user/{userId}/event/{eventId}/role	GET	Retorna o papel de um utilizador num evento específico.
sendFeedback	/feedback	POST	Envia o feedback fornecido pelo utilizador.
getFeedback	/feedback	GET	Retorna uma lista com os feedbacks recebidos.
refreshToken	/refresh-token	POST	Atualiza o token de acesso através do token de atualização.

Tabela A.2: Métodos do UserController

Método	Rota (Path)	Tipo de Pedido	Descrição
createEvent	/event	POST	Cria um novo evento com os dados fornecidos pelo utilizador.
getEvent	/event/{id}	GET	Fornece os detalhes do evento identificado pelo ID
getUsersInEvent	/event/{id}/users	GET	Retorna a lista de participantes do evento.
searchEvents	/events	GET	Pesquisa eventos com base nos parâmetros fornecidos, como termo de busca, limite e offset.
findNearby Events	/events/{radius}/{latitude}/{longitude}	GET	Encontra eventos próximos com base na localização e raio fornecidos, incluindo parâmetros de limite e offset.
joinEvent	/event/{eventId}/join	POST	Permite que um utilizador se junte a um evento específico, fornecendo a senha do evento, se necessário.
joinEventWith Code	/event/{code}	POST	Permite que um utilizador se junte a um evento específico usando um código de convite.
leaveEvent	/event/{eventId}/leave	POST	Permite que um utilizador saia de um evento específico.
deleteEvent	/event/{id}	DELETE	Elimina um evento específico criado pelo utilizador autenticado.
editEvent	/event/{id}	PUT	Edita os detalhes de um evento específico com base nos dados fornecidos pelo utilizador autenticado.
getCategories	/event/categories	GET	Retorna a lista de categorias de eventos da plataforma PlanIt.
kickUser	/event/{eventId}/kick/{userId}	DELETE	Remove um utilizador específico de um evento.

Tabela A.3: Métodos do EventController

Método	Rota (Path)	Tipo de Pedido	Descrição
createPoll	/event/{eventId}/poll	POST	Cria uma votação com os dados fornecidos pelo utilizador para um evento específico.
getPoll	/event/{eventId}/poll/{pollId}	GET	Fornece os detalhes de uma votação específica identificada pelo ID para um evento específico.
deletePoll	/event/{eventId}/poll/{pollId}	DELETE	Elimina uma votação específica de um evento específico criada pelo utilizador autenticado.
votePoll	/event/{eventId}/poll/{pollId}/vote/{optionId}	PUT	Permite que um utilizador vote numa opção específica de uma votação de um evento específico.
getPolls	/event/{eventId}/polls	GET	Retorna a lista de votações de um evento específico.

Tabela A.4: Métodos do PollController



Figura A.1: Modelo EA da base de dados PlanIt