

Relatório da 2.ª Entrega

Projeto de Desenvolvimento Móvel – **Lane**

Universidade Europeia / IADE – Engenharia Informática

Unidade Curricular: Desenvolvimento Móvel (3.º semestre – 2025/2026)

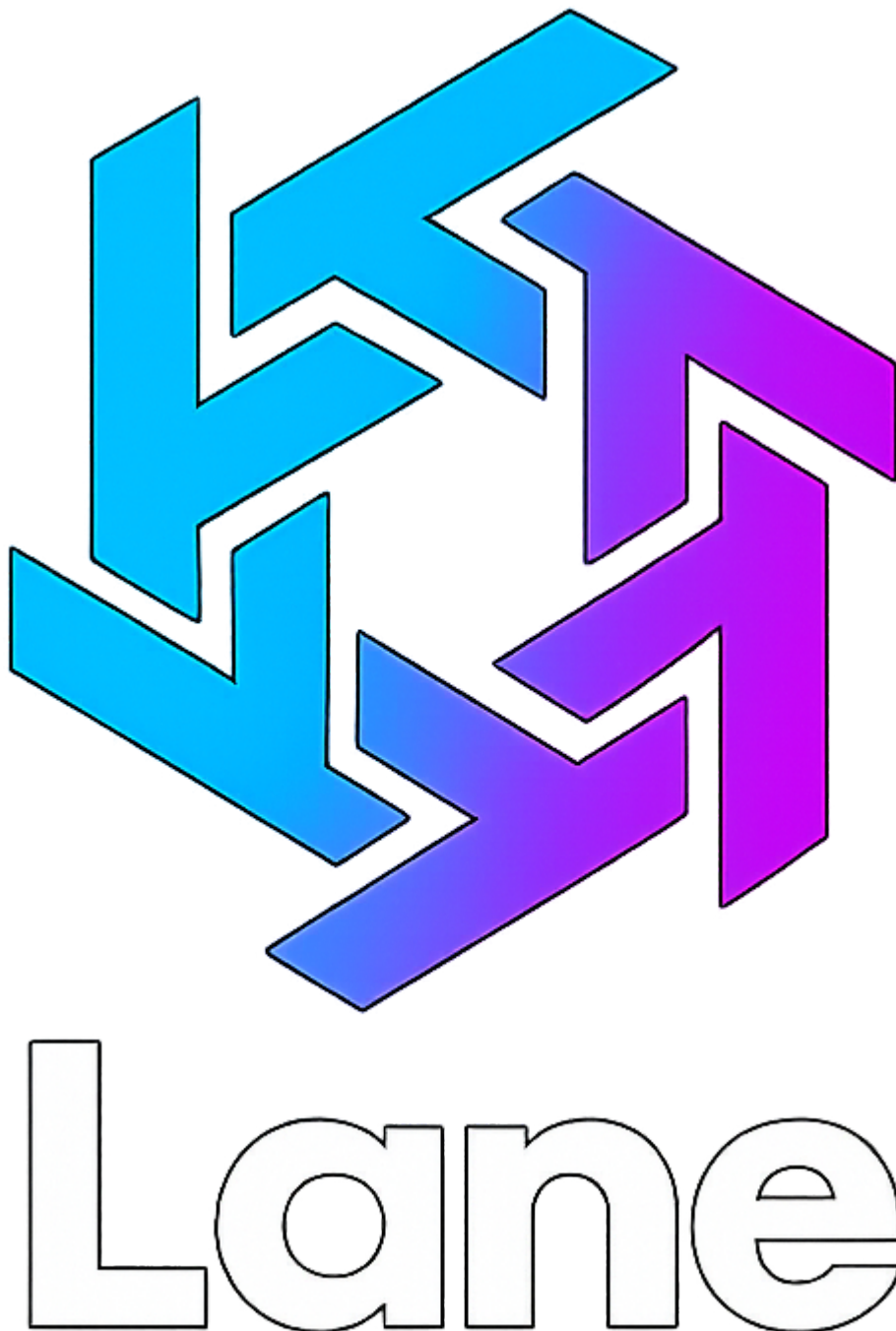
Grupo 08

Elementos: Francisco Abecasis (20240120), Pedro António (20241273), Filomeno Sabino (20241963), Savio Casimira (20240896), Gianni Lopes (20240593)

Figma: [Protótipo Figma](#)

ClickUp: [Espaço ClickUp](#)

GitHub: [Repositório do Projeto](#)



Palavras-chave

aplicações móveis, geolocalização, eventos, convites digitais, perfis verificados, mapas interativos

1. Introdução

A aplicação **Lane** visa facilitar a descoberta e partilha de eventos próximos, permitindo que os utilizadores possam **criar, explorar e participar** em eventos de forma simples e intuitiva.

Na **primeira entrega**, foram definidos o conceito, público-alvo, arquitetura e mockups base.

Nesta **segunda entrega**, o foco foi o **desenvolvimento do protótipo funcional mínimo**, incluindo **criação de eventos, login, e integração inicial com APIs externas**.

2. Enquadramento e Problema

A informação sobre eventos encontra-se dispersa entre múltiplas plataformas, dificultando o acesso a eventos locais e relevantes.

A **Lane** centraliza este processo, promovendo a **proximidade social** e a **autenticidade dos eventos**, com um sistema de **perfis verificados**, **filtros por localização**, e **convites personalizados**.

3. Público-alvo

Segmento	Características	Necessidades
Jovens universitários	Frequentam festas e eventos culturais	Encontrar eventos próximos e seguros
Promotores e artistas	Criam eventos e desejam promover	Ferramentas simples de gestão e partilha
Cidadãos ativos	Procuram lazer e convívio	Ver eventos verificados e próximos

4. Casos de Utilização do Objeto Core e Personas

Caso 1 – Criar um evento privado

1. O utilizador faz login.
2. Seleciona “Criar evento”.
3. Introduce nome, descrição, categoria, data e localização.
4. Escolhe o tipo **Privado** e adiciona amigos convidados.
5. Confirma e o evento é registado na base de dados.

Resultado esperado: Apenas os convidados podem visualizar e participar.

Caso 2 – Explorar eventos próximos


1. O utilizador concede acesso à localização.
2. A app apresenta um mapa com eventos próximos.
3. O utilizador filtra por tipo e distância.
4. Ao selecionar um evento, pode ver detalhes e navegar até lá.

Resultado esperado: O utilizador encontra eventos relevantes de forma rápida.

Personas

User Persona

Perfil



Nome: Rafael Duarte

Idade: 40

Profissão: Informático

Localização: Portugal

Background

Biografia:

Rafael Duarte tem 40 anos e nasceu em Coimbra. Formou-se em Gestão e começou a carreira numa empresa de tecnologia como analista de marketing. Ao longo dos anos, especializou-se em estratégias digitais e branding. É conhecido por ser comunicativo, organizado e por querer sempre trazer novas ideias para dentro da equipa. Fora do trabalho, gosta de futebol, fotografia urbana e viagens curtas de fim de semana.

Audience Insights

Objetivos:


- Encontrar uma app que facilite a descoberta de eventos relevantes para poder voltar a viver e socializar com quando era mais novo

Frustrações

- difuculdade na localização de eventos por isso deixou de sair tanto de casa

User Persona

Perfil



Nome: Miguel Moreira

Idade: 34

Profissão: Informático

Localização: Portugal

Background

Biografia:

Miguel Moreira tem 34 anos e vive no Porto. Estudou Engenharia Informática e trabalhou durante muitos anos em grandes empresas de TI, em manutenção e infraestrutura. Nos últimos dois anos, mudou-se para uma pequena empresa de videojogos indie, onde atua como programador sénior. Fora do trabalho, é apaixonado por música eletrónica e organiza pequenos eventos com amigos.

Audience Insights

Objetivos:

- Usar uma app que o ajude a encontrar eventos alinhados com os seus gostos, como showcases de música eletrónica e encontros criativos.

Frustrações:

- Perde muito tempo a procurar eventos interessantes e atualizados.

5. Arquitetura e Tecnologias

5.1 Arquitetura

- **Camada Mobile:** Android (Kotlin + Jetpack Compose)
- **Camada Backend:** Spring Boot (REST API)
- **Base de Dados:** PostgreSQL
- **Integrações Externas:** Google Maps API

Descrição:

A arquitetura segue o modelo **Cliente-Servidor**, com comunicação via **API RESTful**. O backend gere autenticação, eventos e utilizadores, enquanto o frontend apresenta os dados em tempo real através de endpoints. Uma decisão de arquitetura chave foi a gestão das relações sociais. A regra de negócio define 'amizade' como um seguimento mútuo. Para evitar redundância de dados e problemas de integridade, a tabela friends (Tabela que iria gerir o estado de amizade na primeira versão da BD) foi eliminada. A tabela followers é agora a única fonte da verdade, e o estado de 'amizade' é um dado calculado dinamicamente pelo backend (API) sempre que a app o solicita.

5.2 Tecnologias e Ferramentas

Categoria	Tecnologia	Função
Mobile	Kotlin, Jetpack Compose	Interface e lógica de apresentação
Backend	Java/Spring Boot	Gestão de rotas e API REST
Base de Dados	PostgreSQL	Armazenamento de utilizadores e eventos
APIs	Google Maps SDK	Localização e rotas
Gestão	GitHub, ClickUp, Discord	Controlo de versões e comunicação
Design	Figma	Mockups e UI/UX

6. Requisitos Funcionais e Não Funcionais

Requisitos Funcionais

- Login e registo de utilizadores
- Criação e edição de eventos
- Pesquisa e filtragem de eventos por tipo e distância
- Sistema de convites privados
- Perfis verificados com selo

Requisitos Não Funcionais

- Interface intuitiva e responsiva
- Segurança de Autenticação via Spring Security
- Conformidade com o RGPD
- Chat privado

6.2 Requisitos Não Funcionais (Análise de Recursos)

A natureza da "Lane" como uma app baseada em mapas e dados em tempo real implica um consumo de recursos significativo:

- **No Cliente (App):** O consumo de **Bateria** e **GPS** é o ponto mais crítico, devido à necessidade de renderizar o Google Maps SDK e obter a localização do utilizador para calcular distâncias. O consumo de **Dados Móveis** também é relevante, para descarregar os "azulejos" do mapa e as listas de eventos da API.
 - **No Servidor (Custos):** A app irá incorrer em custos de *hosting* (para o *backend* Spring Boot e a BD PostgreSQL) e custos de **API da Google Maps Platform**. Especificamente, as chamadas de **Geocoding** (para converter endereços de eventos em coordenadas) e **Directions** (para calcular rotas) são feitas pelo *backend* por razões de segurança, e serão debitadas (embora cobertas, numa fase inicial, pelo crédito mensal de 258€ da Google).
-

7. Interfaces

Incluem-se abaixo os principais ecrãs da versão alfa:



Lane

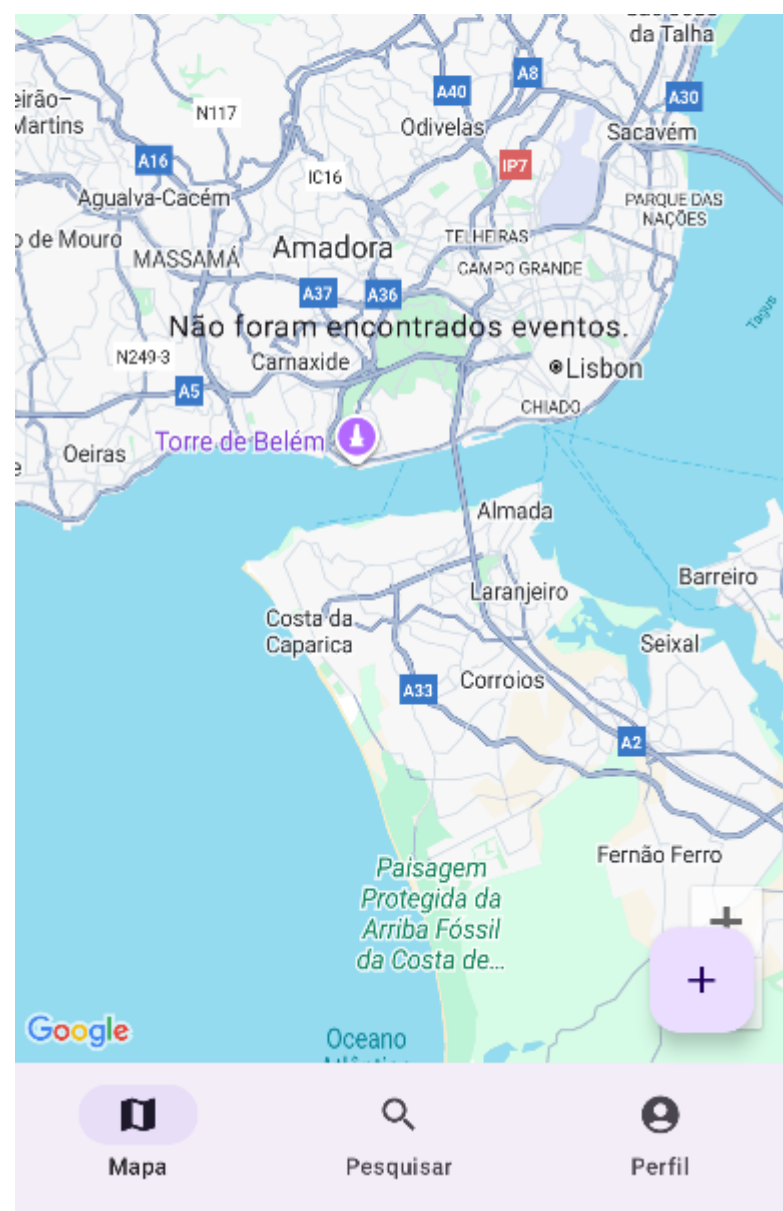
Nome de utilizador ou email

Palavra-passe

Iniciar Sessão

Não tens uma conta? Regista-te





← Criar Evento

Título do Evento

Descrição

Tipo de Evento

Data

YYYY-MM-DD

Hora

--:--

Localização

Preço (€)

0

Max. Participantes

100

Privacidade

public

Criar Evento

[Ver protótipo no Figma](#)

8. Planeamento e Execução

O projeto foi organizado no ClickUp e Discord.
Nesta fase, foram concluídas as seguintes tarefas:

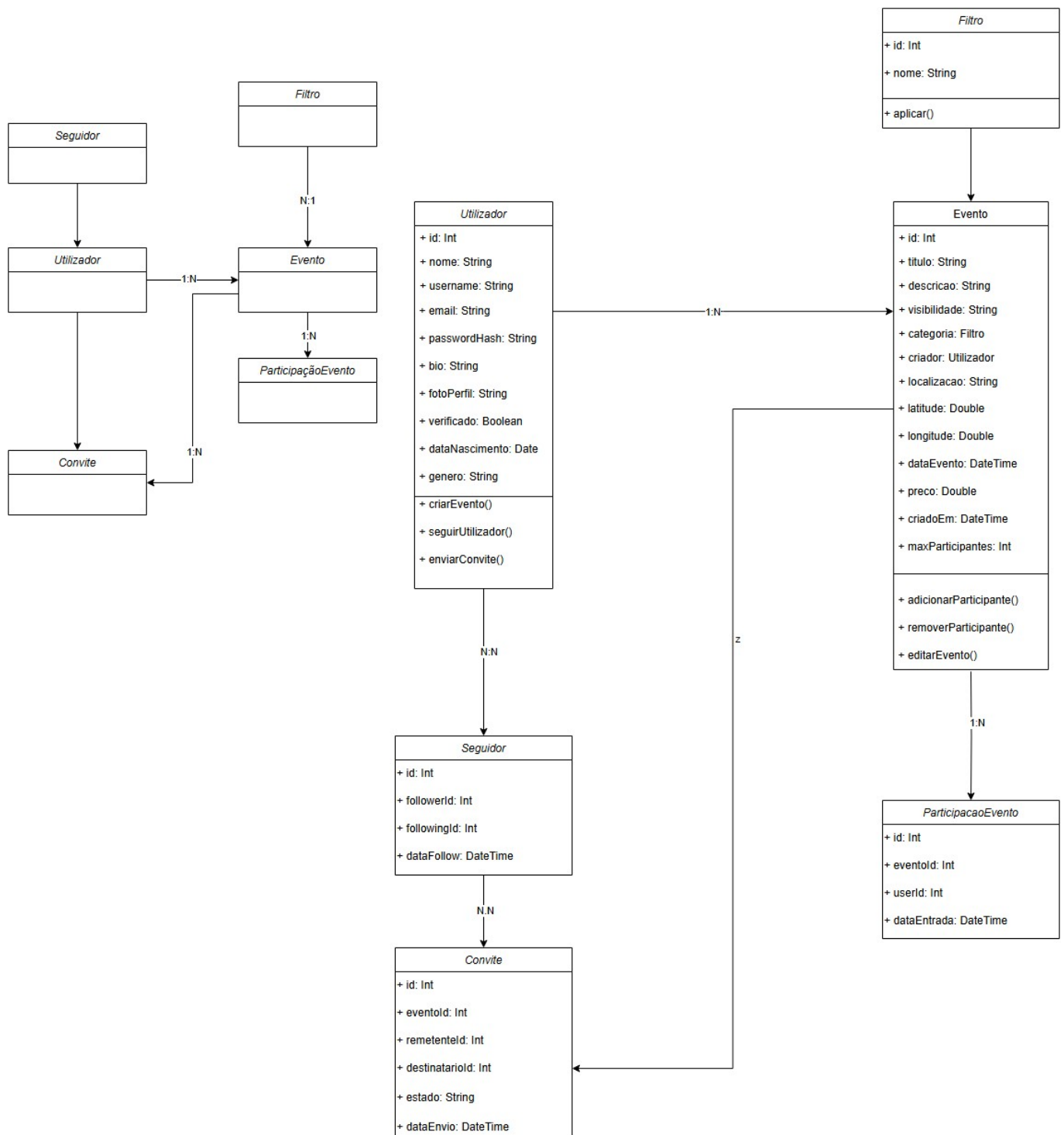
- Integração entre frontend e backend
- Ligação da base de dados PostgreSQL

- Criação de eventos
- Implementação do login e registo de utilizadores
- Planeamento da verificação de perfis

Planeado para a 3ª entrega:

- Implementação de tokens JWT
- Finalização de todos os ecrãs da app
- Integração completa do mapa

9. Diagrama de Classes



Descrição: O diagrama de classes reflete as principais entidades (Utilizador, Evento, Convite, Seguidor) e as suas relações. Uma decisão de arquitetura chave foi a gestão das relações sociais: a regra de negócio define 'amizade' como um seguimento mútuo. Para evitar redundância de dados, a tabela `friends` foi eliminada. A tabela `followers` é agora a única fonte da verdade, e o estado de 'amizade' é um dado calculado dinamicamente pelo *backend* (API).

10. Dicionário de Dados (Guia de Dados)

Esta secção detalha a estrutura física (Modelo Físico) das tabelas implementadas na base de dados PostgreSQL, cumprindo os requisitos de um "Dicionário de Dados".

Tabela	Campos Principais	Descrição
<code>user_details</code>	<code>account_id</code> (PK), <code>account_username</code> , <code>account_email</code> , <code>account_password_hash</code> , <code>account_dob</code> , <code>account_gender</code>	Armazena os dados do utilizador, incluindo UNIQUE constraints para username/email e CHECK para género.
<code>events</code>	<code>event_id</code> (PK), <code>event_title</code> , <code>event_creator_id</code> (FK), <code>event_category_id</code> (FK), <code>max_participants</code> , <code>event_latitude</code>	Armazena todos os eventos. Inclui o limite de participantes e as coordenadas de GPS.
<code>filters</code>	<code>filters_id</code> (PK), <code>filters_name</code>	Tabela de "lookup" para as categorias dos eventos (ex: "Música", "Cultural").
<code>followers</code>	<code>follow_id</code> (PK), <code>follower_id</code> (FK), <code>following_id</code> (FK)	Tabela central da lógica social. Armazena a relação "seguir". Usada para <i>calcular</i> amizades (seguimento mútuo).
<code>event_participants</code>	<code>participant_id</code> (PK), <code>event_id</code> (FK), <code>user_id</code> (FK)	Tabela de junção que regista quem "aderiu" a um evento (o botão "Participar").
<code>invitations</code>	<code>invitations_id</code> (PK), <code>event_id</code> (FK), <code>sender_id</code> , <code>receiver_id</code> , <code>status</code>	Gere os convites para eventos privados, com um <code>status</code> (<code>pending</code> , <code>accepted</code> , <code>rejected</code>).

11. Documentação REST

A tabela seguinte resume os principais *endpoints* da API REST implementados no *backend* (Spring Boot) para suportar as funcionalidades do MVP (Protótipo Funcional Mínimo).

Verbo	Endpoint	DTO (Request Body)	Resposta	Função
POST	<code>/api/users/users</code>	<code>RegisterRequestDTO</code>	<code>User</code>	Registar um novo utilizador.

Verbo	Endpoint	DTO (Request Body)	Resposta	Função
POST	/api/users/login	LoginRequestDTO	LoginResponseDTO	Autenticar um utilizador e devolver um token.
GET	/api/events/events	-	List<Event>	Obter a lista de todos os eventos.
POST	/api/events/create/events	CreateEventDTO	Event	Criar um novo evento.
GET	/api/filters/get/filters	-	List<Filters>	Obter as categorias para o <i>dropdown</i> .

12. Conclusão

O desenvolvimento da Lane encontra-se numa fase sólida, com o MVP funcional e integração entre os principais módulos. A integração total das três camadas (Frontend Android/Compose, Backend Spring Boot, e Base de Dados PostgreSQL) foi um marco complexo, mas está agora funcional.

Os fluxos de **Registo de Utilizador** (com *hashing* de password via **BCrypt**) e **Login de Utilizador** (com gestão de sessão) estão a funcionar e ligados ao *backend*. A app consegue agora criar e (em breve) visualizar eventos (**POST** /api/events/create/events), e ir buscar dados dinâmicos da API, como as categorias de filtros (**GET** /api/filters/get/filters).

Os próximos passos (3ª entrega) focar-se-ão em implementar a segurança total com *tokens JWT*, finalizar a lógica de *geocoding* no *backend* e desenvolver as restantes funcionalidades sociais, como o ecrã de perfil e a interação com o mapa.

13. Bibliografia e Referências

- **Google.** [Documentação Oficial do Jetpack Compose.](#)
- **Google.** [Google Maps SDK for Android Documentation.](#)
- **Spring.** [Spring Boot Documentation.](#)
- **Spring.** [Spring Security - Password Storage \(BCrypt\).](#)
- **PostgreSQL.** [Documentação Oficial do PostgreSQL.](#)
- **Square.** [Retrofit: A type-safe HTTP client for Android and Java.](#)