Pontifícia Universidade Católica do Rio Grande do Sul
Escola Politécnica
Bacharelado em Sistemas de Informação

SAFE TRAVELS

Plataforma Colaborativa de Dados de Localização em Viagens

VICENTE LEAL HOFMEISTER

Orientador: Prof. Me. Filipe Novo Mor

Porto Alegre
2025

Lista de Figuras

1 Comparativo entre aplicações e solução proposta 9
2 Diagrama de arquitetura de serviços da API 16
3 Diagrama de arquitetura da Lambda de processamento da API 18
4 Diagrama de arquitetura da Lambda de processamento da fila 19
5 Diagrama de modelagem do banco de dados SQL 20
6 Splash screen 26
7 Página de login 26
8 Página de cadastro 27
9 Página de viagens 27
10 Página de mapa 28
11 Página de grupo 28

Lista de Siglas

API - Application Programming Interface

AWS - Amazon Web Services

IA - Inteligencia artificial

IAM - Identity and Access Management

KDS - Kinesis Data Streams

MFA - Autenticação Multifator

MVP - Produto mínimo viável

noSQL - Linguagem de consulta não estruturada

RDS - Relational Database Service

SES - Simple Email Service

SQL - Linguagem de consulta estruturada

SQS - Simple Queue Service

S3 - Simple Storage Service

UUID - Identificador único universal

vCPU - Unidade central de processamento virtual

Sumário

Lista de Figuras **2**

Lista de Siglas **3**

1 Introdução **5**

2 Fundamentação Teórica **7**
2.1 Aplicações de viagens e compartilhamento de localização 7
2.2 Privacidade no compartilhamento de posição 8
2.3 Cobertura de rede móvel de telefonia 8
2.4 Comparativo com a solução proposta 9

3 Objetivos **10**

4 Solução **11**
4.1 Tecnologias 11
4.1.1 AWS 11
4.1.2 TypeScript 13
4.1.3 React Native 13
4.1.4 Figma 14
4.1.5 GitHub 14
4.2 Fluxo de desenvolvimento 14
4.3 Servidor *Cloud* 15
4.3.1 Arquitetura 15
4.3.2 Banco de dados 19
4.3.3 Mapeamento de Rotas 21
4.4 Aplicação *Mobile* 21
4.4.1 Arquitetura 22
4.4.2 Prototipação de Telas 22

5 Planejamento de Continuidade **24**

6 Considerações Finais **25**

Apêndice A Prototipação de telas **26**

Apêndice B Mapeamento de rotas da API **29**

Apêndice C Modelagem do banco de dados relacional **53**

Referências **55**

1 Introdução

O avanço tecnológico observado nos últimos anos mudou diversos aspectos da sociedade. Tecnologias de geolocalização, como o GPS, presentes em aparelhos celulares, permitem que o usuário saiba exatamente onde está, enquanto a internet possibilita que ele se comunique com amigos e familiares de qualquer lugar do globo. O próprio compartilhamento de localização tornou-se uma prática comum e mostra-se um grande auxílio na convivência e organização diária com pessoas próximas, seja para coordenar previsões, encontros ou acompanhar jornadas a distância. Em viagens, a habilidade de compartilhar a localização ajuda a assegurar amigos e familiares sobre a segurança do viajante, e registrar esses dados para poder exibir e revisitar o caminho percorrido torna-se uma recordação uma vez que a viagem termina.

Porém, essa habilidade é afetada por limitações de *hardware*, como o uso da bateria e, principalmente, a conectividade. Os aparelhos mais comuns utilizados para transmitir a localização são os próprios *smartphones* usados no cotidiano, que, geralmente, dependem da conexão com a internet para enviar dados. Levando em consideração o uso de compartilhar a localização em tempo real, é necessário ter acesso à rede móvel, o que pode se tornar um empecilho, dependendo do destino da viagem, ou trazer custos adicionais. Além disso, a própria rede móvel pode estar instável, indisponível ou acessível somente para clientes de operadoras de telefonia celular específicas, o que é especialmente comum em destinos mais remotos ou afastados das cidades.

Apesar disso, ao viajar em grupo, é possível combinar os serviços de diferentes operadoras de telefonia móvel. Dessa forma, cada integrante com acesso a uma operadora distinta pode compartilhar a conexão de dados com os demais, quando necessário, ampliando temporariamente a cobertura do grupo. No entanto, essa solução apresenta contrapartidas: o compartilhamento de internet via *hotspot* consome mais bateria e a franquia de dados do aparelho que estiver servindo de ponto de acesso. Além disso, se a opção for não compartilhar a internet, apenas os contatos selecionados pelo portador da conexão receberão as atualizações de localização, o que dispersa as informações e exige que múltiplas pessoas gerenciem diferentes listas de destinatários, dificultando a notificação simultânea de todos os amigos e familiares.

Uma alternativa à rede móvel, que busca comunicação com maior disponibilidade, é a comunicação via satélite. Esta modalidade oferece maior disponibilidade ao redor do globo, mesmo em áreas isoladas. Existem duas modalidades principais de aparelhos com comunicação via satélite: Antenas de internet e Comunicadores. As antenas de internet via satélite tendem a ser mais robustas, necessitando que o usuário fique próximo ao local ou veículo onde a antena se encontra, mas permitem o uso de soluções existentes para *smartphones* e computadores que precisam de acesso à internet. Comunicadores

via satélite, por outro lado, são menores e mais portáteis, e podem oferecer comunicação unidirecional ou bidirecional. Geralmente, possuem a funcionalidade de compartilhar a localização, enviando dados em determinados intervalos de tempo. O problema desta comunicação é o custo e a acessibilidade, uma vez que necessita de um investimento inicial para adquirir o aparelho, além do custo mensal para manter o acesso aos serviços associados.

Com base nos problemas apresentados, o presente trabalho de conclusão busca abordar o problema de compartilhar dados de localização em tempo real durante viagens em grupo, enfrentando desafios de conexão, segurança e privacidade. O trabalho está dividido em seis capítulos: **Introdução**(1), que apresenta a problemática abordada pelo trabalho de conclusão, assim como seu escopo; **Fundamentação Teórica**(2), que apresenta referencias e recursos de comparação.; **Objetivos**(3), que apresenta os objetivos do trabalho de conclusão e do presente volume; **Solução**(4), que apresenta a solução proposta para atingir os objetivos propostos; **Planejamento de Continuidade**(5), que apresenta as próximas etapas de desenvolvimento e implementação do trabalho de conclusão; **Considerações Finais**(6), que conclui o volume, recapitulando o que foi apresentado e fazendo considerações para a próxima etapa do trabalho.

2 Fundamentação Teórica

Este capítulo aborda informações para subsidiar a proposta apresentada. As informações são divididas nas seguintes subseções: **Aplicações de viagens e compartilhamento de localização**, que abordam aplicações atuais destinadas a auxiliar viajantes, assim como aplicações de uso cotidiano que utilizam o compartilhamento de localização. A **Privacidade no compartilhamento de posição** aborda aspectos de preocupação com a privacidade que surgem com o compartilhamento de localização, tanto do ponto de vista técnico quanto por parte dos usuários. A **Cobertura de rede móvel de telefonia** aborda as limitações de cobertura e disponibilidade da rede móvel de telefonia de acordo com a localidade e as operadoras telefônicas. Por fim, **Comparativo com a solução proposta** descreve uma análise contrastando as diferenças entre as soluções existentes e a solução proposta neste trabalho. A solução proposta será abordada em mais detalhes no capítulo 4.

2.1 Aplicações de viagens e compartilhamento de localização

Existem diversas aplicações no mercado para auxílio no planejamento e acompanhamento de viagens, com funcionalidades variadas. Planejadores de viagens como Lambus[23], Stippl[34], TripIt[17] e Wanderlog[36] auxiliam na criação de itinerários, na organização de passagens e outros documentos. Aplicações como Visited Travel[35] e Pin Traveler[29] permitem ao usuário criar mapas de lugares visitados, podendo compartilhá-los com familiares e amigos. Já aplicações como Roadie[27] e OsmAnd[28] auxiliam na navegação em situações variadas.

Contudo, embora diversos aplicativos de viagens consumam dados de localização do usuário para acompanhar seu trajeto ou itinerário, nem todos permitem que o usuário compartilhe sua localização com outros usuários. Durante o levantamento de aplicações do segmento, FindPenguins[20] e Polarsteps[30] foram os aplicativos encontrados, voltados para viagens, com a funcionalidade de compartilhar a localização do usuário em tempo real. Ambos oferecem diversas outras funcionalidades, apresentando-se como opções bastante completas para registrar e compartilhar viagens e trajetos, mas focados na experiência individual, ou ao menos em um perfil único, sem a possibilidade de funcionalidades colaborativas entre usuários.

Existem outras aplicações que oferecem compartilhamento de localização com enfoques que não são viagens. Life360[24], por exemplo, é um aplicativo focado em compartilhamento de localização cotidiana e permite a criação de grupos onde todos os usuários podem visualizar a localização uns dos outros, além de funcionalidades como histórico de localização e notificações das atividades dos usuários. Opções como ShareLocation[33] e Google Maps[22] permitem compartilhar a localização em tempo real por um período ou

trajeto limitado, podendo incluir funcionalidades como acompanhar a previsão de chegada e gerar rotas entre usuários a fim de se encontrarem. Porém, embora essas aplicações recebam informações de mais de um usuário ou aparelho, essas informações não são tratadas de forma colaborativa.

2.2 Privacidade no compartilhamento de posição

A privacidade e a segurança dos dados dos usuários devem ser tópicos abordados com cuidado, especialmente quando envolvem localização e outros dados pessoais. Ao criar grupos de usuários ou aparelhos que recebem informações uns dos outros, corremos o risco de que, com um aparelho comprometido, sejam expostas informações de todos os membros do grupo. O trabalho de *Aagard et al. (2023)*[1] demonstra este risco com o aplicativo Life360[24], mencionado também no capítulo anterior, podendo haver exposição de mensagens, endereços de e-mails, nomes, entre outras informações.

Contudo, apesar dos riscos, a percepção dos usuários é variada. O trabalho de *Boutet; Morel (2025)*[16] coletou dados qualitativos e quantitativos de 99 jovens adultos, nativos digitais, entre 20 e 26 anos. O resultado descreve que o grupo subestima os riscos relacionados à privacidade, além de demonstrar práticas arriscadas em relação à essa privacidade. Já o trabalho de *Belligoni et al. (2023)*[15] coletou dados de 1016 estudantes universitários por meio de um questionário, levantando questionamentos sobre o compartilhamento de localização, incluindo privacidade e segurança. O estudo mostrou que a maioria dos participantes utilizava o compartilhamento de localização, mas apresentava preocupações com segurança e privacidade, sendo a privacidade estatisticamente mais preocupante para os participantes.

2.3 Cobertura de rede móvel de telefonia

A comunicabilidade pode se tornar um desafio para compartilhar a localização em tempo real. No caso de smartphones, é necessário ter acesso a rede móvel de telefonia, que pode ser instável, indisponível ou disponível somente para clientes de uma determinada operadora telefônica. De acordo com a Agência Nacional de Telecomunicações (ANATEL)[2], existem oito operadoras presentes no Brasil que, em conjunto, oferecem cobertura de rede móvel para 92,56% da população. Entretanto, a cobertura está mais concentrada em áreas urbanas, com moradores urbanos tendo 99,45% de cobertura, enquanto moradores rurais atingem somente 44,72%. A ANATEL também disponibiliza dados de cobertura por localidade, que indicam que o território nacional possui apenas 15,31% de cobertura de rede móvel 3G e 14,17% de rede móvel 4G. Para o usuário, essa cobertura se apresenta menor, uma vez que nenhuma operadora telefônica possui cobertura de rede móvel 3G ou 4G superior a 11%.

2.4 Comparativo com a solução proposta

Esta seção compara as funcionalidades das aplicações de viagens e compartilhamento de localização presentes no mercado com a solução proposta. O comparativo está ilustrado por meio da tabela na figura 1.

Solução	Planejamento	Registro	Navegação	Compartilhar Localização	Função colaborativa	Integração com aparelhos diversos
Lambus, Stippl, TripIt	X					
Visited Travel, Pin Traveler		X				
OsmAnd, Roadie			X			
FindPenguins, Polarsteps		X		X		
Life360, ShareLocation, Google Maps				X		
Solução proposta		X		X	X	X

Figura 1: Comparativo entre aplicações e solução proposta

Os planejadores de viagem, como Lambus, Stippl e TripIt, concentram-se na organização de itinerários, passagens e reservas, mas não registram o percurso nem possibilitam o compartilhamento de localização em tempo real. Os diários de viagem, como Visited Travel e Pin Traveler, permitem marcar lugares visitados e compartilhar mapas ao final da jornada, porém, não oferecem navegação nem interação ao vivo. Aplicativos de navegação, como OsmAnd e Roadie, traçam rotas e orientam o usuário, mas carecem de funcionalidades de registro de viagem e de compartilhamento com terceiros.

As aplicações de rastreamento individual, como FindPenguins e Polarsteps, combinam o registro do percurso e o compartilhamento de localização em tempo real, mas a experiência é centrada no viajante: cada usuário cria seu próprio diário, sem agregação de dados entre diferentes participantes. Por sua vez, as aplicações de compartilhamento cotidiano, como Life360, ShareLocation e o recurso de compartilhamento de localização do Google Maps, permitem que familiares ou amigos vejam a localização uns dos outros, mas não contemplam planejamento de roteiros, registro de trajetos anteriores ou integração colaborativa de dados. Nessas aplicações, cada usuário gerencia sua lista de contatos, e as informações permanecem dispersas.

A solução proposta busca preencher essas lacunas ao reunir, em um único sistema, o registro de trajetos e o compartilhamento em tempo real, acrescidos de uma função colaborativa que agrega as localizações de todos os membros do grupo em um rastro coletivo de viagem. Além disso, o sistema é projetado para integrar dados provenientes de diferentes aparelhos, conferindo maior resiliência em locais com cobertura de rede heterogênea. Dessa forma, a proposta combina funcionalidades que, nas soluções existentes, aparecem isoladas, oferecendo uma visão conjunta e mais completa da viagem para viajantes e seus acompanhantes.

3 Objetivos

O presente trabalho de conclusão tem como objetivo principal desenvolver um sistema de viagens que permita o compartilhamento e o processamento de dados de localização em um perfil único e colaborativo. Os objetivos específicos estabelecidos para atingir o objetivo principal foram: Desenvolver uma aplicação móvel que permita compartilhar dados de localização e cadastrar um aparelho transmissor de localização; desenvolver uma API para receber dados de localização de diversos aparelhos transmissores de localização pré-cadastrados e correlacioná-los como um perfil único e colaborativo.

Este volume descreverá a solução proposta em detalhes, assim como os próximos passos para sua implementação. As tecnologias, o ferramental e os serviços a serem utilizados foram escolhidos levando em consideração a utilização no mercado, assim como a familiaridade do autor. Também foi dada preferencia em utilizar a mesma linguagem de programação para as aplicações *mobile* e de servidor, conforme será exposto no capítulo 4, visando reduzir possíveis confusões durante o desenvolvimento. Havia sido planejado o desenvolvimento de provas de conceito para este volume, com o objetivo de validar determinados fluxos da aplicação. Estas provas de conceito não foram realizadas devido à imprevistos pessoais na agenda do autor ao longo deste período. Porém, sua ausência não irá impactar a implementação da solução nas próximas etapas do trabalho de conclusão. Em contrapartida, foram estabelecidos a modelagem do banco de dados e o mapeamento de rotas da API do servidor, tarefas que não estavam presentes no escopo inicial deste volume.

4 Solução

O presente trabalho de conclusão propõe a aplicação *Safe Travels*, um sistema que permitirá receber dados de localização de diversos perfis e aparelhos, processando-os de forma colaborativa, a fim de gerar um perfil único agregado de localização. O *Safe Travels* será composto por uma aplicação *mobile* e um servidor *cloud*, seguindo o modelo de aplicação cliente-servidor. O servidor *cloud* será capaz de receber dados de localização oriundos de diversos aparelhos e usuários, além de processa-los para fornecer as informações requeridas através de rotas de API. A aplicação *mobile* será a interface pela qual o usuário irá interagir com a aplicação, podendo compartilhar seus dados, selecionar preferencias de privacidade, além de acompanhar as viagens de outros usuários e grupos.

Este capítulo aborda os componentes da solução proposta e é composto por quatro seções. A seção “Tecnologias” descreve linguagens de programação, *frameworks*, bibliotecas e outras ferramentas utilizadas para desenvolver a solução. A seção “Fluxo de desenvolvimento” descreve a estrutura utilizada para apoiar e guiar o desenvolvimento da solução, incluindo a organização de repositórios, automações e testes. A seção “API”. descreve a aplicação de servidor da solução proposta, incluindo a arquitetura, a modelagem do banco de dados e o mapeamento de rotas da API. A seção “Aplicação Mobile” descreve a aplicação cliente da solução proposta, incluindo a arquitetura e a prototipação de telas.

4.1 Tecnologias

4.1.1 AWS

A AWS é a líder entre os provedores de computação em nuvem[13]. Seus serviços permitem a criação de sistemas *cloud native* resilientes, escaláveis e com alcance global em instantes, sem a necessidade de gerenciar infraestrutura. Cada serviço possui formatos de cobrança próprios. Porém, diversos contam com uma categoria de uso gratuita. Os serviços selecionados foram escolhidos tendo em vista a possibilidade de desenvolver e a infraestrutura requerida de forma gratuita durante o período de desenvolvimento do trabalho. A seguir, serão descritos os serviços ofertados pela AWS que serão utilizados para compor o servidor da solução.

4.1.1.1 API Gateway

Serviço para criação e gerenciamento de APIs *REST* e *WebSocket* de forma escalável e segura. Nos 12 primeiros meses da conta AWS, é possível realizar 1 milhão de chamadas de API por mês de forma gratuita[3].

4.1.1.2 Cognito

Fornece autenticação e gerenciamento de usuários, com suporte para *login* social e MFA. Permite gerenciar até 50 mil usuários ativos mensais de forma gratuita[4].

4.1.1.3 DynamoDB

Banco de dados NoSQL totalmente gerenciado, com escalabilidade automática e baixa latência. Permite 25 GB de armazenamento, 25 unidades de leitura e 25 unidades de escrita por mês de forma gratuita[5].

4.1.1.4 Identity and Access Management - IAM

Gerencia identidades, papéis e permissões de acesso a recursos da AWS. Seu uso é totalmente gratuito[11].

4.1.1.5 Kinesis Data Streams - KDS

Captura e processa dados em tempo real para análise contínua. Permite um fluxo de 1 MB/s de entrada e saída sem custo nos primeiros 12 meses [6].

4.1.1.6 Lambda

Executa código sob demanda sem necessidade de provisionar servidores. Permite 1 milhão de execuções e 400 mil GB/s por mês de forma gratuita[12].

4.1.1.7 Relational Database Service - RDS

Banco de dados relacional gerenciado, compatível com PostgreSQL, MySQL e outros. Permite a execução de instancias db.t4g.micro (2 vCPU, 1 GiB) por 750 horas por mês e 20 GB de armazenamento até 12 meses[7].

4.1.1.8 Parameter Store

Serviço do AWS Systems Manager usado para armazenar parâmetros de configuração e segredos de forma segura. Não há cobrança para até 10 mil parâmetros do tipo Standard [14].

4.1.1.9 Simple Email Service - SES

Serviço gerenciado para envio e recebimento de *e-mails* em escala, com suporte a autenticação, análise e integração com aplicações. A categoria gratuita permite o envio ou a recepção de até 3 mil mensagens por mês[8].

4.1.1.10 Simple Queue Service - SQS

Serviço de filas de mensagens para integração assíncrona entre componentes. Permite 1 milhão de solicitações por mês sem custo[9].

4.1.1.11 Simple Storage Service - S3

Armazenamento de objetos com alta durabilidade e disponibilidade global. A categoria gratuita permite 5 GB de armazenamento, com 20 mil solicitações de GET e 2 mil de PUT por mês, até 12 meses[10].

4.1.2 TypeScript

O TypeScript[26] e o JavaScript, linguagem na qual o TypeScript se baseia, são amplamente utilizados para aplicações web e móveis[32]. Diferentemente do JavaScript, o TypeScript adiciona tipagem estática e recursos de orientação a objetos, permitindo detectar erros ainda na fase de desenvolvimento e tornando o código mais seguro e previsível[18].

A linguagem é transpilada para JavaScript comum, o que garante compatibilidade com navegadores, servidores e *frameworks* existentes. Sua ampla adoção na indústria e o suporte oficial por parte de ferramentas modernas fazem dela uma escolha consolidada para projetos de médio e grande porte.

No contexto deste trabalho, o TypeScript é utilizado tanto na aplicação móvel quanto na API, promovendo a reutilização de estruturas de dados, a uniformidade entre camadas e uma melhor manutenção do código ao longo do tempo.

4.1.3 React Native

O React Native[25] é um *framework* de código aberto que permite desenvolver aplicativos móveis nativos utilizando as linguagens JavaScript ou TypeScript. Ele possibilita que uma única base de código gere versões para Android e iOS, com desempenho próximo ao de aplicativos desenvolvidos nativamente em cada sistema.

A tecnologia é baseada em componentes reutilizáveis e em uma arquitetura declarativa, o que simplifica a criação de interfaces de usuário dinâmicas e responsivas. Além disso, integra-se de forma natural com APIs nativas dos dispositivos, como GPS, câmera e notificações.

A escolha pelo React Native deve-se à sua eficiência no desenvolvimento multiplataforma, à integração direta com TypeScript e à grande comunidade de suporte, o que reduz o tempo de implementação e facilita futuras manutenções[31].

4.1.4 Figma

O Figma é uma ferramenta de *design* de interfaces e prototipagem. Ele permite criar *wireframes*, fluxos de navegação e protótipos interativos, centralizando arquivos e facilitando o compartilhamento de *feedback* em um único espaço de trabalho[19].

Neste trabalho, o Figma é utilizado para projetar as telas da aplicação móvel, definindo a identidade visual, os componentes de interface e os fluxos de navegação. Os protótipos gerados servem como referência visual para a implementação, reduzindo ambiguidades e alinhando expectativas entre a arquitetura proposta e a experiência do usuário.

4.1.5 GitHub

O GitHub é uma plataforma de desenvolvimento colaborativo baseada em Git, utilizada para hospedagem de código-fonte e controle de versão. A ferramenta oferece recursos como *issues*, *pull requests*, gerenciamento de *branches* e *wikis*, além de integrações com uma ampla variedade de serviços externos[21].

No contexto deste trabalho, o GitHub é utilizado como repositório do código da solução proposta e como base para automações de desenvolvimento contínuo. Por meio de *workflows* de integração e entrega contínuas, a plataforma apoia a execução de testes automatizados, a verificação da qualidade e o empacotamento da aplicação, contribuindo para o ciclo de desenvolvimento.

4.2 Fluxo de desenvolvimento

O desenvolvimento da solução será conduzido com um conjunto de práticas e ferramentas para auxiliar o processo. O projeto será mantido no GitHub em três repositórios com objetivos distintos. Dois deles serão destinados ao desenvolvimento da solução em si: um para a API e outro para a aplicação *mobile*. O terceiro repositório será utilizado como *Wiki*, centralizando informações do projeto como um todo. Todos os repositórios permanecerão públicos para acesso.

Os repositórios da API e da aplicação *mobile* irão dispor de um conjunto de ferramentas automatizadas para auxiliar o desenvolvimento, buscando minimizar erros humanos e agregar valor à solução. Os repositórios serão configurados para não aceitarem *commits* de código novo diretamente na *branch* principal, exigindo a realização de *pull requests*. A cada *pull request*, será executada uma pipeline automatizada contendo checagem de lint, *build* de código e, no caso do repositório da API, o *deploy* da solução. Além disso, será utilizada a funcionalidade de *code review* automatizado pelo GitHub Copilot, agente de IA do GitHub, para trazer uma checagem extra ao código. Estes processos serão assegurados pelas regras dos repositórios do próprio GitHub.

4.3 Servidor *Cloud*

A solução proposta contará com um servidor *cloud native* com o qual a aplicação mobile irá se comunicar para armazenar, processar e administrar seus dados, viagens e interações com outros usuários. O servidor também permitirá que o usuário configure e adicione dispositivos transmissores de localização alternativos, de forma que ele possa contar com múltiplas fontes de dados, mesmo que esteja viajando sozinho.

A comunicação entre o servidor e a aplicação *mobile* será majoritariamente por requisições HTTPs para a API, seguindo conceitos *REST*. A adoção dessas características busca tornar as rotas mais claras, melhorando o entendimento do estado atual da API. Isso também contribui para o desenvolvimento de novas plataformas e para a manutenção da aplicação *mobile*. A API foi idealizada para utilizar recursos com categorias gratuitas durante o desenvolvimento, mas com potencial de escalabilidade para acompanhar o aumento no número de usuários.

O código fonte da aplicação servidor será desenvolvido em TypeScript, buscando aproximar o desenvolvimento da aplicação servidor ao desenvolvimento da aplicação *mobile*. O código será organizado em módulos, buscando auxiliar na manutenção do código, assim como facilitar a inserção ou remoção de funcionalidades em módulos isolados. O uso de módulos também permite utilizar a funcionalidade de *layers* das funções Lambda, onde o código será inserido. A funcionalidade de *layers* divide o código em camadas, podendo reutiliza-las em diferentes funções, reduzindo a quantidade de dados armazenados na *cloud*, assim como o tempo de *deploy* da *stack* como um todo. A seguir, serão apresentados mais detalhes do planejamento do servidor *cloud*, incluindo a arquitetura, a modelagem de banco de dados e o mapeamento de rotas.

4.3.1 Arquitetura

A arquitetura do servidor foi desenvolvida para ser *cloud native* utilizando serviços da AWS. Esta seção irá expor a arquitetura de serviços *cloud*, assim como as arquiteturas de software a serem utilizadas.

4.3.1.1 Arquitetura de serviços *cloud*

O servidor foi planejado com serviços AWS, levando em conta as opções disponibilizadas de forma gratuita, mas que ainda contém escalabilidade para acomodar o aumento no número de usuários e requisições. A arquitetura de serviços foi desenvolvida conforme demonstrado na figura 2, e será descrita ao longo desta seção.

Conforme mencionado anteriormente, o servidor da solução proposta visa receber dados oriundos da aplicação *mobile*(1), assim como de dispositivos transmissores de localização

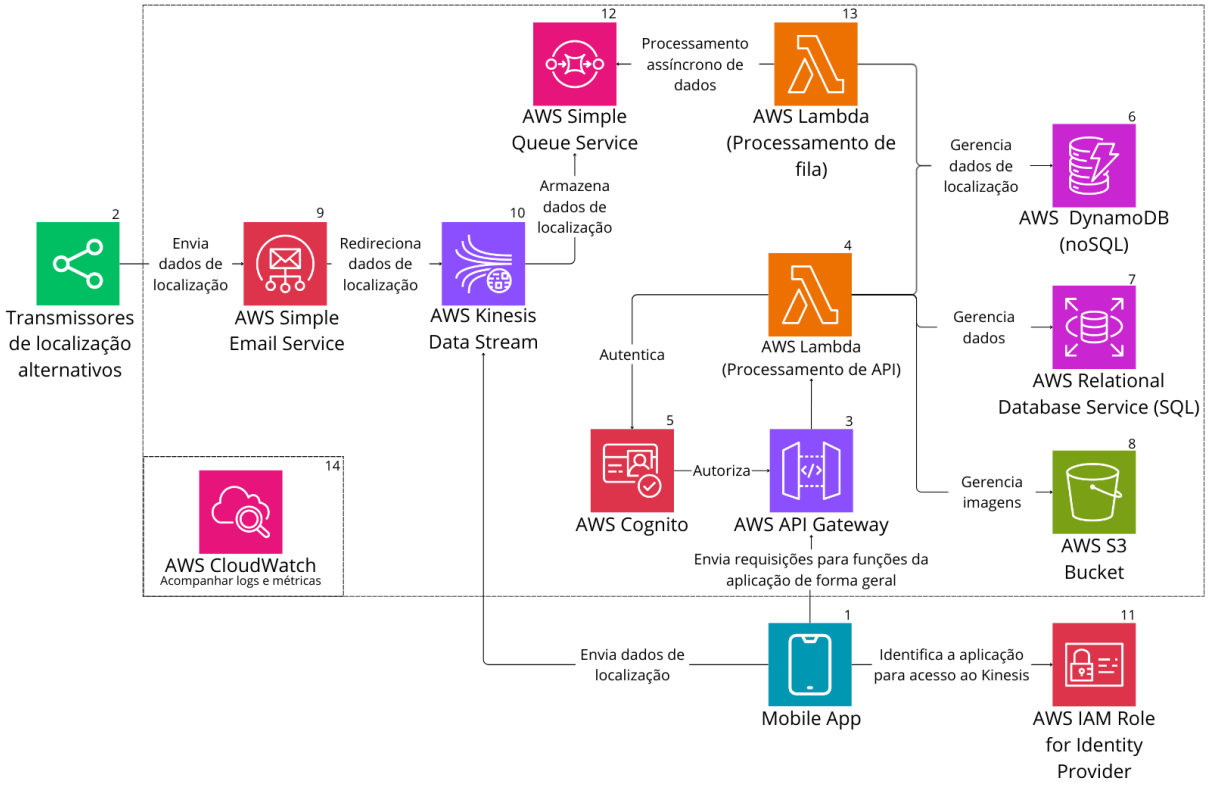

Figura 2: Diagrama de arquitetura de serviços da API

alternativos(2). A aplicação *mobile*(1) terá duas formas diferentes de interagir com o servidor *cloud*: via requisições HTTPS no AWS API Gateway(3), para gerenciar a maioria das interações do usuário na aplicação, e através do Kinesis Data Stream(10), para envio de grandes quantidades de dados de localização.

A API Gateway contará com rotas públicas e privadas, divididas em subgrupos de acordo com suas funcionalidades, conforme descrito na seção 4.3.3. O AWS Cognito(5) será utilizado para autenticação das rotas privadas, obrigando que as requisições possuam um *bearer token* válido no *header*. Quando o API Gateway receber uma requisição válida, ela será redirecionada para uma função AWS Lambda para o processamento da API(4), que tem seu funcionamento interno descrito na seção 4.3.1.2.

O Kinesis Data Stream(10) será o principal destino dos dados de localização da aplicação *mobile*, sendo capaz de lidar com a ingestão de grandes volumes de dados. O objetivo de separar a ingestão de dados de localização das demais requisições é reduzir o fluxo de requisições no API Gateway(3), o que também diminui a chance de ocorrer *throttling*, tornando a API mais resiliente ao aumento de requisições e dados dos usuários. Diferente do API Gateway, o Kinesis Data Stream poderá receber dados provenientes da aplicação *mobile* ou de transmissores de localização alternativos. Os dados da aplicação *mobile* serão enviados diretamente da aplicação para o Kinesis Data Stream, utilizando IAM Roles(11) para permitir acesso ao recurso. Porém, não é viável contar com a possibilidade

de que os transmissores de localização alternativos enviem dados diretamente ao Kinesis Data Stream, visto que estes possuem plataformas e comunicações próprias para cada aparelho.

Para receber dados de transmissores genéricos, o servidor terá a capacidade de receber dados por *e-mail* através do SES(9). Diversos transmissores, inclusive o que será utilizado durante o desenvolvimento desta aplicação, enviam dados de localização através de *e-mails* genéricos, os quais podem ser processados e ingeridos pela aplicação com facilidade. Para isso, o usuário da aplicação *mobile* deverá adicionar e configurar seus transmissores de localização alternativos para que o servidor possa ingerir seus dados. Ao receber um e-mail no SES, o servidor irá repassar a mensagem para o Kinesis Data Stream para ser processada posteriormente.

Os dados enviados ao Kinesis Data Stream são enviados ao SQS(12) para processamento assíncrono, o que ativa a Lambda de processamento de fila(13). Esta Lambda, então, é responsável por lidar com os dados, extraindo informações das mensagens recebidas do SES e armazenando todos os dados válidos na base de dados. O funcionamento interno da Lambda de processamento de fila é descrito em mais detalhes na seção 4.3.1.3.

A base de dados dos servidores será composta por um DynamoDB(6) para armazenamento não-relacional e um RDS(7) para armazenamento relacional. O objetivo da adoção de dois modelos diferentes de bancos de dados é equilibrar as vantagens e desvantagens de cada um, considerando a performance e o custo de manutenção. Além disso, será utilizado também um S3 Bucket(8) como armazenamento auxiliar para manter imagens com maior facilidade, podendo inclusive disponibilizar um *link* para a aplicação busca-las diretamente do *bucket*. A base de dados será descrita em mais detalhes na seção 4.3.2.

Por fim, serão utilizados serviços do CloudWatch(14) para acessar *logs* de execução de software e para acompanhar métricas de desempenho de recursos.

4.3.1.2 Arquitetura de software - Processamento de API

A Lambda de Processamento de API será responsável por processar as requisições feitas ao API Gateway(3). Sua arquitetura interna, demonstrada na figura 3, divide o código em módulos de acordo com as funcionalidades.

Quando uma nova requisição chega ao API Gateway(3), esta requisição é enviada como evento para a Lambda de Processamento de API(4). O ponto de entrada da Lambda é o módulo Gerente de API(4.01), que determina o caminho interno a ser seguido de acordo com a rota da requisição que foi feita. Requisições de autenticação serão redirecionadas para o módulo de autenticação(4.02), que chamará funções do Cognito(5) para gerenciar lógicas de *login*, registro e gerenciamento de senhas de usuários. Embora o destino principal dos dados de localização seja o Kinesis Data Stream, a Lambda de processamento

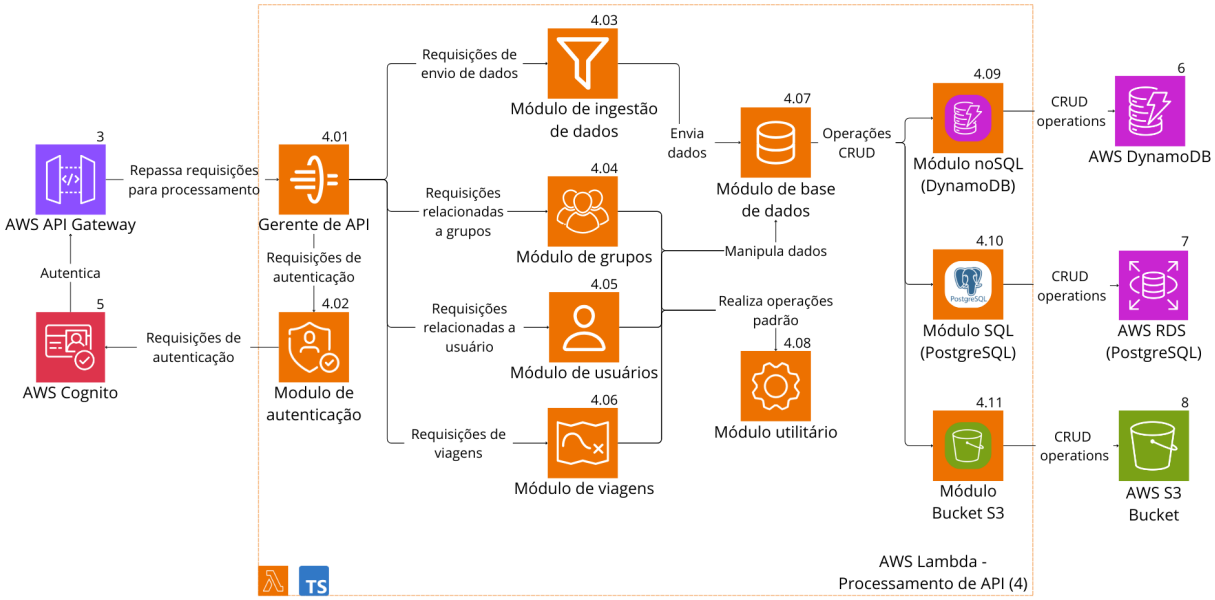

Figura 3: Diagrama de arquitetura da Lambda de processamento da API

da API também terá um módulo de ingestão de dados(4.03), funcionando como reserva, caso haja algum problema com o Kinesis Data Stream ou sua conexão. Por fim, demais requisições serão enviadas aos módulos de grupos(4.04), usuários(4.05) e viagens(4.06) de acordo com o alvo da ação a ser realizada pela rota.

Os módulos citados acima serão responsáveis por processar as requisições. Porém, terão acesso ao módulo de base de dados(4.07) e ao módulo utilitário(4.08) como ferramentas para auxiliar e abstrair partes do processamento. O módulo utilitário irá conter funções comuns entre os módulos de usuários, grupos e viagens, permitindo que reutilizem funções sem a necessidade de reescreve-las. O módulo de base de dados servirá como intermediário aos módulos noSQL(4.09), SQL(4.10) e *Bucket* S3(4.11), visando abstrair a interação com a base de dados, assim como as estruturas que estão em uso. Desta forma, caso seja necessário realizar alterações na base de dados, não será necessário alterar os módulos principais. Por fim, os módulos noSQL, SQL e *Bucket* S3 se comunicarão com seus respectivos bancos ou estruturas de dados para realizar operações *CRUD*. O módulo noSQL irá se comunicar com o DynamoDB(6), o módulo SQL irá se comunicar com o RDS(7) e o módulo *Bucket* S3 irá se comunicar com o S3 Bucket(8).

4.3.1.3 Arquitetura de software - Processamento de fila

A Lambda de Processamento de fila será responsável por processar os dados depositados no SQS(12) pelo Kinesis Data Stream. Sua arquitetura interna, demonstrada na figura 4, divide o código em módulos, visando reutilizar o código desenvolvido para a Lambda de processamento de API, descrita na seção 4.3.1.2.

A Lambda de Processamento de fila é ativada por novos dados que chegam ao SQS(12)

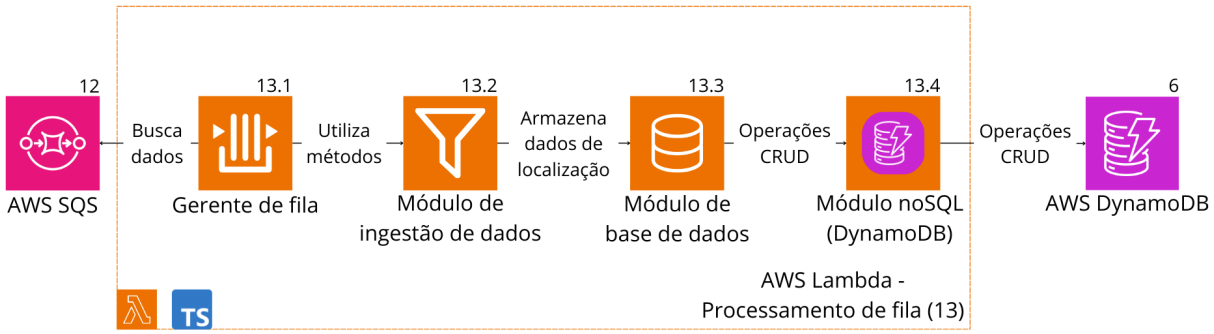

Figura 4: Diagrama de arquitetura da Lambda de processamento da fila

e recepciona esses dados através do módulo gerente de fila(13.1). Os dados são então redirecionados ao módulo de ingestão de dados(13.2) para serem tratados e padronizados. Por fim, o módulo de ingestão de dados utiliza o módulo de base de dados(13.3), que servirá como abstração para o módulo noSQL(13.4). Diferente da Lambda de processamento de API, que se comunica com toda a base de dados, o único componente da base de dados com o qual a Lambda de processamento de fila irá se comunicar será o DynamoDB(6). Isso ocorrerá porque os dados de localização serão armazenados somente no DynamoDB, não havendo necessidade de conceder acessos adicionais a esta Lambda.

4.3.2 Banco de dados

A base de dados do servidor deve ser capaz de realizar consultas relacionando dados de usuários, grupos e viagens, mas também deve lidar com grandes volumes de dados de localização. A necessidade de consultas torna uma base de dados SQL ideal para o armazenamento de dados, permitindo a indexação de determinados dados, além de gerenciar o relacionamento entre tabelas e otimizar as buscas. Contudo, visto que o armazenamento alocado tem um impacto considerável no custo da infraestrutura do banco de dados em nuvem, especialmente para estruturas relacionais, é interessante explorar a possibilidade de utilizar opções mais baratas quando possível.

No contexto do servidor do Safe Travels, existe um tipo de dado que pode ser bastante volumoso e que possui relação somente com outra entidade ou tabela: Os dados de localização. Cada usuário, grupo ou viagem terá a necessidade de lidar com seus dados de localização, que podem se tornar volumosos rapidamente, considerando o compartilhamento de localização em tempo real. Para lidar com isso, também será utilizado um banco de dados não relacional para armazenar os dados de localização, sem a necessidade de estruturas relacionais e com custos potencialmente inferiores. O banco não relacional escolhido foi o DynamoDB, que é nativo da AWS e permite armazenar os dados em tabelas, mantendo a possibilidade de indexações simples. A última estrutura a ser utilizada na base de dados é um *bucket* S3 para o armazenamento de imagens. Dessa forma, será

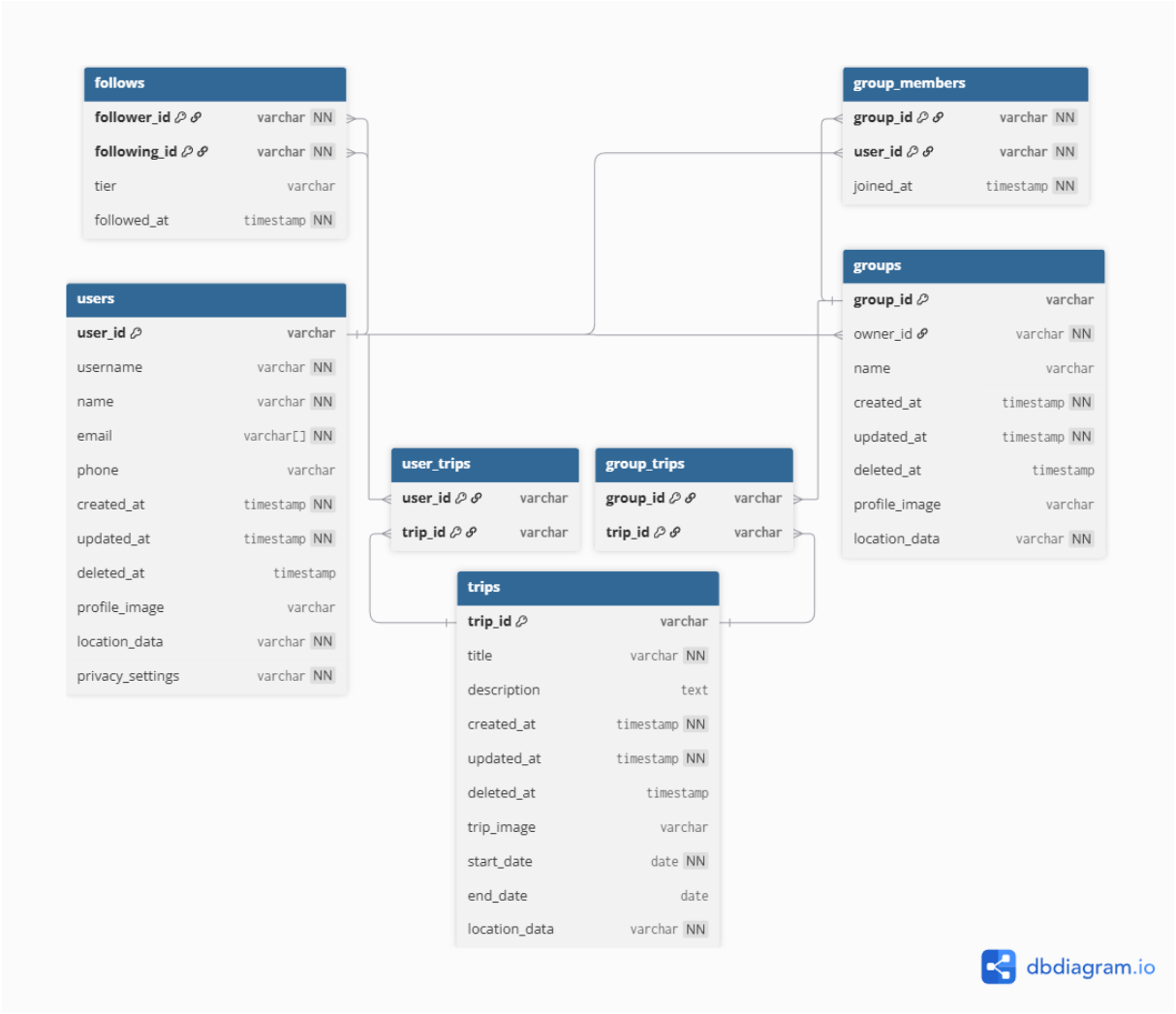

Figura 5: Diagrama de modelagem do banco de dados SQL

possível fornecer um link temporário em cada requisição da aplicação *mobile* que aponte diretamente para a imagem no *bucket*.

O banco de dados relacional terá referências para os dados necessários nas demais bases, de forma que será possível obter a referência para, então, buscar a informação no DynamoDB ou no *bucket* S3. O banco de dados relacional escolhido para a aplicação foi o PostgreSQL devido a sua integração com o RDS e a possibilidade nativa de utilizar listas como tipo de dados. A modelagem do banco relacional, conforme mostrado na figura 5, foi feita considerando três entidades principais e quatro entidades associativas, nomeadas em inglês para padronização internacional.

As entidades principais são ***users***, ***groups*** e ***trips***. Todas as entidades primárias terão UUIDs independentes de seus demais dados, como chaves primárias, exclusão lógica, imagem de perfil e uma tabela própria para os seus dados de localização no DynamoDB. A exclusão lógica se dará com o preenchimento do campo *deleted_at*. A tabela de dados no DynamoDB será referenciada no campo *location_data*, enquanto a imagem de perfil

terá seu *path* no *bucket* S3 no campo *profile_image*.

Users irão se relacionar consigo mesmos através da entidade associativa ***follows***, que conterá *follower_id*, para indicar a chave primária do usuário seguindo, e *following_id*, indicando a chave primária do usuário sendo seguido. A entidade *follows* também conterá o campo *tier* para indicar o grau de permissão que o seguidor tem. *Groups* irão se relacionar com *users* de duas formas. A primeira será através do campo *owner_id* de *groups*, indicando qual *user* é o administrador do *group*. A segunda será através da entidade associativa ***group_member***, que indica que um *user* é participante de um *group*. Por fim, *trips* irá se relacionar tanto com *users* quanto com *groups*, porém somente com uma das entidades por vez. Isso será feito com as entidades associativas ***user_trips*** e ***group_trips*** que irão indicar quais *trips* pertencem a cada *user* ou *group*. Mais detalhes sobre o mapeamento do banco de dados relacional podem ser encontrados no apendice C.

4.3.3 Mapeamento de Rotas

As rotas da API foram mapeadas para prever as necessidades das aplicações *mobile* e de servidor, e podem ser lidas no apêndice B. As requisições foram agrupadas de acordo com o módulo da Lambda de processamento de API que irá processa-las. Para indicar esse agrupamento, as requisições possuem os prefixos ***auth***, ***user***, ***group***, ***trip*** ou ***location***. A maioria das requisições necessita de *tokens* de autorização no *header*, com exceção de algumas requisições relacionadas a autenticação de usuários. As requisições retornarão respostas com códigos específicos de acordo com o resultado da operação. Os códigos de resposta mais comuns no mapeamento das rotas são 200, indicando sucesso na operação; 400, indicando requisição inválida; 401, indicando que o usuário não está autenticado; 429, indicando que foram feitas muitas requisições; e 500, indicando um erro inesperado na aplicação por parte do servidor.

4.4 Aplicação *Mobile*

A solução proposta conta com uma aplicação *mobile* para arrecadar dados de localização a serem enviados ao servidor, assim como possibilitar que os usuários interajam com outros usuários e grupos, visualizando viagens, selecionando configurações de privacidade, além de outras funcionalidades. A aplicação mobile será desenvolvida com o *framework* React Native, visando a possibilidade de desenvolver para múltiplas plataformas com um único código fonte, garantindo uniformidade de funcionalidades entre dispositivos e agilizando o desenvolvimento. A presente seção irá detalhar a arquitetura planejada para a aplicação *mobile*, assim como descrever a prototipação de telas realizada.

4.4.1 Arquitetura

A aplicação *mobile* foi planejada para utilizar uma arquitetura modular, facilitando a manutenção e a adição de funcionalidades. Além disso, a aplicação contará com três camadas: Interface, Lógica de Negócio e Serviços. A camada de Interface será responsável por gerenciar telas, componentes visuais e fluxos de uso que serão utilizados pelo usuário. A camada de Interface se comunicará com a camada de Lógica de Negócio, que gerenciará as funcionalidades principais da aplicação, como compartilhamento de localização, gerenciamento de perfil, grupos e viagens, alertas, entre outros. Por fim, a camada de Lógica de Negócio irá se comunicar com a camada de Serviços para acessar recursos do dispositivo, como GPS e armazenamento interno, e realizar chamadas para o servidor *cloud*.

A comunicação entre as camadas de Lógica de Negócio e Serviços será essencial para o serviço de localização, especialmente em segundo plano. A aplicação deverá ser capaz de coletar dados de localização mesmo quando a tela do aparelho estiver desligada ou quando o usuário estiver utilizando outra aplicação. Para isso, serão utilizadas APIs nativas de geolocalização e tarefas de *background* agendadas, buscando manter um consumo eficiente da bateria. O usuário deverá poder configurar o intervalo de atualizações para ajustar suas preferências, podendo priorizar uma maior quantidade de dados ou um menor impacto energético.

Os dados de localização obtidos deverão conter latitude, longitude, data e hora, e poderão ser mantidos no dispositivo temporariamente enquanto não for possível realizar a transmissão para o servidor. O seu envio será realizado para o Kinesis Data Stream no servidor *cloud*, o que não inclui a mesma verificação de autenticação utilizada nas requisições para a API Gateway. Por conta disso, cada pacote de dados de localização enviado deve conter também informações sobre o usuário que está enviando os dados, assim como utilizar uma IAM Role pré definida para ter permissão de envio ao serviço do servidor.

4.4.2 Prototipação de Telas

Além de oferecer as funcionalidades propostas na solução, a aplicação *mobile* deve apresentar uma interface que auxilie na navegação e na utilização pelo usuário. Para buscar atingir este objetivo, foram desenvolvidas prototipação de telas para as principais funcionalidades da aplicação *mobile*, demonstradas por completo no apêndice A. A prototipação foi feita utilizando o Figma e buscou-se seguir um *design* simples que facilitasse a visualização das informações.

Foi adicionada uma barra de navegação na parte inferior das telas para acessar as principais funcionalidades: *Home*, *Trips*, *Maps*, *Groups* e *Profile*. *Home* irá conter pequenos *cards* com marcos de viagens de outros usuários e grupos, como a chegada a uma nova cidade, determinada distância percorrida, etc. *Trips* irá permitir que o usuário visualize

suas viagens, incluindo anteriores e atuais. Também será possível acessar uma página específica de cada viagem que irá exibir mais informações sobre a mesma, conforme demonstrado na figura 9. *Maps* irá mostrar um mapa com a localização mais recente dos usuários e grupos seguidos, conforme demonstrado na figura 10. *Groups* irá listar os grupos aos quais o usuário faz parte, permitindo visualizar cada um em uma página com informações completas. Está página irá informar os participantes, indicar quem é o administrador e também exibir viagens deste grupo, conforme demonstrado na figura 11. Por fim, *Profile* irá conter o perfil do usuário, permitindo visualizar e editar informações, além de selecionar preferências de privacidade.

5 Planejamento de Continuidade

O trabalho desenvolvido para o presente volume do trabalho de conclusão resultou na arquitetura de serviços e software das aplicações *mobile* e de servidor, além da modelagem de banco de dados relacional, mapeamento de rotas e API, entre outras estruturas e processos definidos. O que foi desenvolvido neste semestre servirá de alicerce para o trabalho a ser feito no próximo, que terá o objetivo de desenvolver e implementar a aplicação *mobile* e o servidor *cloud* propostos para o *Safe Travels*.

Ao longo do próximo semestre, a solução será implementada de forma gradual, iniciando com a funcionalidade de compartilhamento de localização, a principal da solução proposta, e adicionando outras funcionalidades de acordo com sua importância para validar a solução proposta. Planeja-se medir a performance e a eficiência da solução desenvolvida por meio de métricas como o consumo de bateria do celular durante o uso da aplicação *mobile*, o sucesso no envio de dados de localização e a taxa de sucesso de processamento de dados de localização com transmissores de localização no servidor *cloud*. Ao final do próximo semestre, espera-se ter desenvolvido um MVP que valide o funcionamento da solução proposta neste trabalho de conclusão.

6 Considerações Finais

O presente volume do trabalho de conclusão apresenta o *Safe Travels* como solução proposta, sendo esta uma plataforma colaborativa de compartilhamento de dados de localização voltada para viagens individuais e em grupo. Este volume definiu os objetivos da solução e detalhou sua arquitetura. Foram apresentados o conjunto de tecnologias e ferramentas selecionadas, o fluxo de desenvolvimento adotado e os principais componentes da aplicação de servidor e da aplicação *mobile*. A arquitetura proposta utiliza serviços da AWS dentro das categorias gratuitas, permitindo desenvolvimento e testes sem custos significativos, mas mantendo o potencial de escalabilidade. A modelagem do banco de dados, o mapeamento das rotas da API e a prototipação das telas da aplicação *mobile* compõem o núcleo estrutural desenvolvido neste semestre.

As definições apresentadas aqui servirão como alicerce para o próximo volume do trabalho, no qual será realizada a implementação efetiva da solução. O planejamento inclui a priorização da funcionalidade central, o compartilhamento de localização, seguido pela integração dos demais recursos necessários para validar o conceito da plataforma. Ao final do próximo semestre, espera-se alcançar um MVP funcional que permita avaliar a eficiência, o fluxo de dados e a viabilidade da solução proposta, incluindo métricas como consumo de bateria, disponibilidade de envio e taxa de sucesso de processamento no servidor.

Assim, este volume cumpre o objetivo de estabelecer uma base sólida para o desenvolvimento da solução *Safe Travels*, consolidando decisões técnicas, estruturando o projeto e preparando o caminho para sua implementação prática.

Apêndice A Prototipação de telas

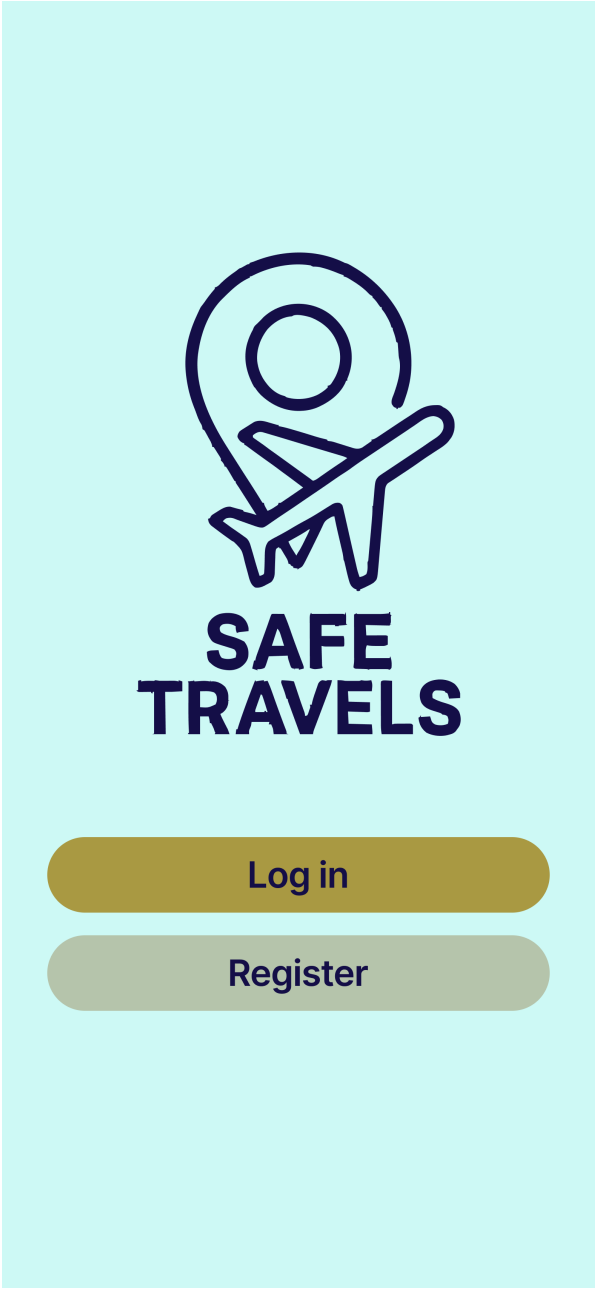

Figura 6: Splash screen

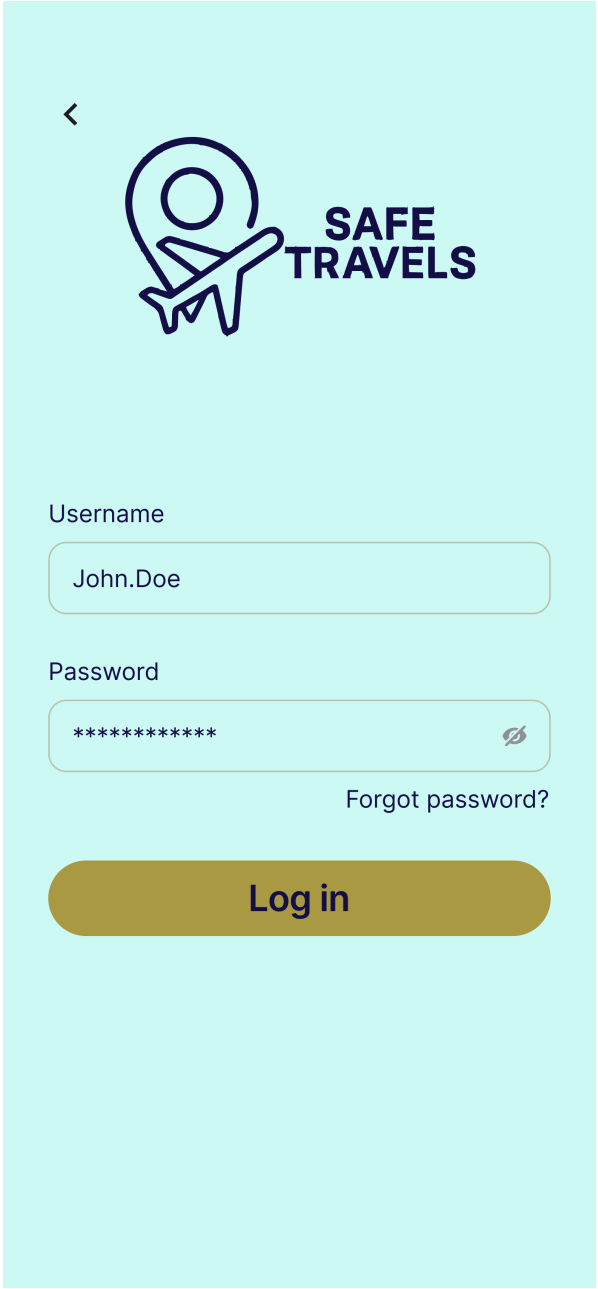

Figura 7: Página de login

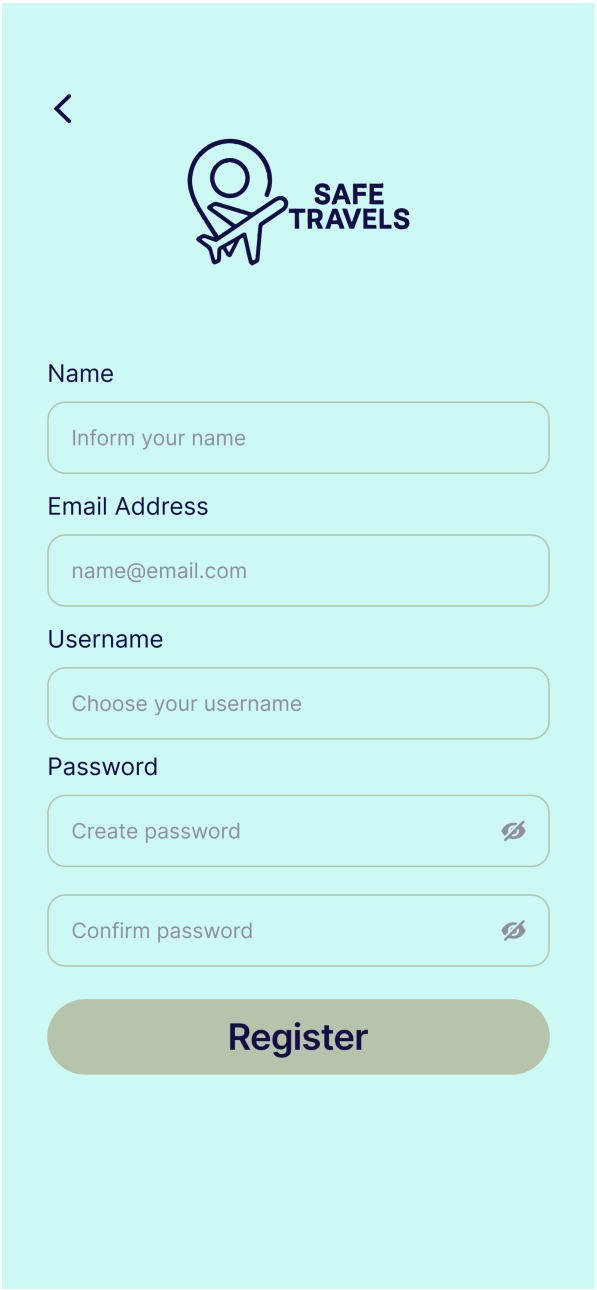

Figura 8: Página de cadastro

Figura 9: Página de viagens

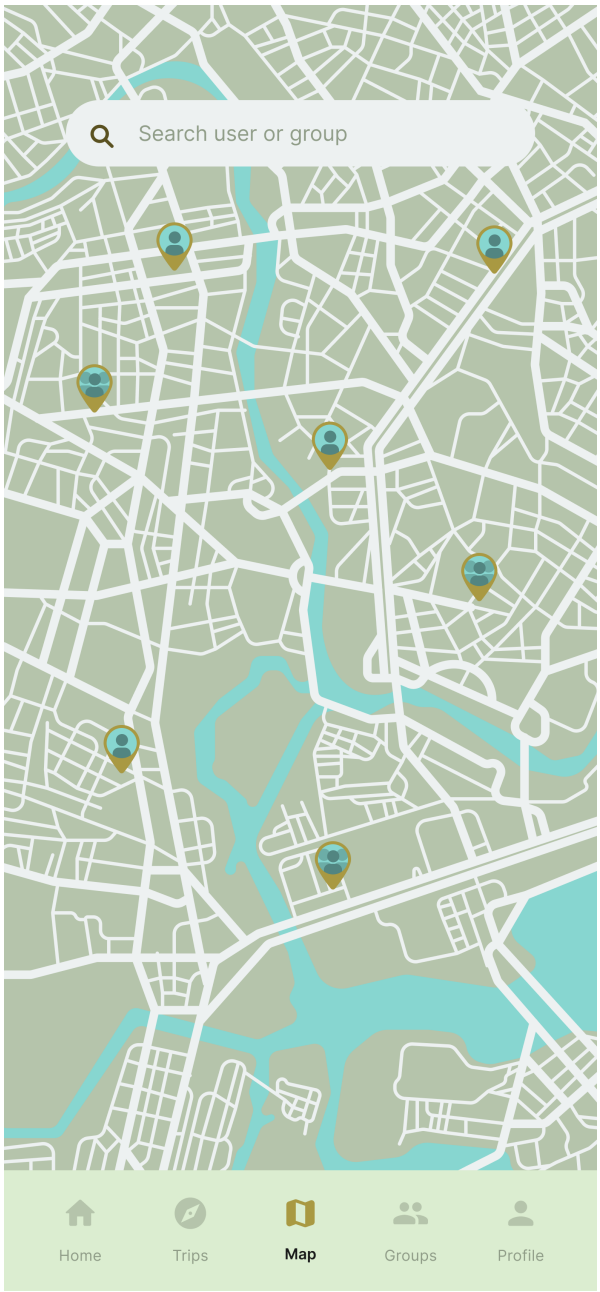

Figura 10: Página de mapa

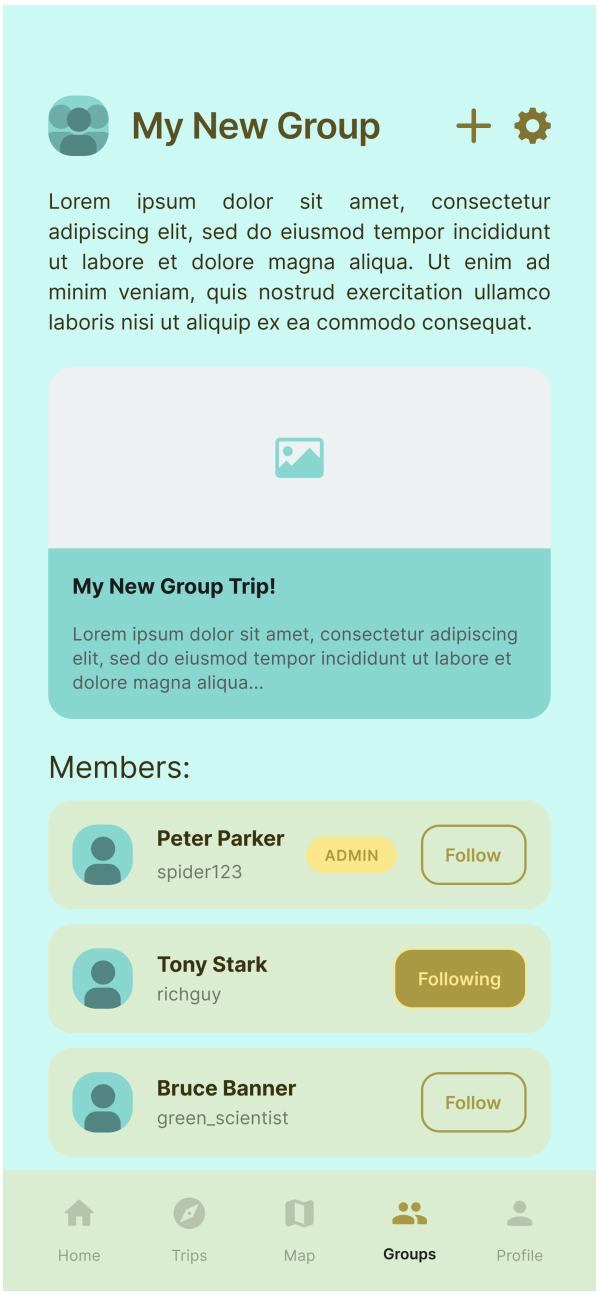

Figura 11: Página de grupo

Fonte: Elaborado pelo autor

Apêndice B Mapeamento de rotas da API

```yaml
openapi: "3.0.3"
info:
  title: Safe Travels' Endpoints
  version: "1.0.0"

components:
  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT

servers:
  - url: https://api.server.test/v1 # TODO: Replace with actual server url

paths:
  /auth/login:
    post:
      summary: Authenticate user to access the API
      security: []
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required: [username, password]
              properties:
                username:
                  type: string
                  example: john-doe
                password:
                  type: string
                  format: password
                  example: MyPassword@123
      responses:
        "200":
          description: Login was successful
          content:
            application/json:
              schema:
                type: object
                properties:
                  access_token:
                    type: string
```

```yaml
                  example: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
                expires_in:
                  type: integer
                  example: 3600
      "401":
        description: Invalid credentials
      "429":
        description: Too many requests
      "500":
        description: Internal server error

/auth/refresh:
  post:
    summary: Use refresh token to revalidate authentication
    security: []
    requestBody:
      content:
        application/json:
          schema:
            type: object
            required: [refreshToken]
            properties:
              refreshToken:
                type: string
                example: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
    responses:
      "200":
        description: Session refreshed
        content:
          application/json:
            schema:
              type: object
              properties:
                access_token:
                  type: string
                  example: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
                expires_in:
                  type: integer
                  example: 3600
      "401":
        description: unauthorized
      "403":
        description: Forbidden. User logged out
      "410":
        description: Expired refresh token
      "429":
        description: Too many requests
      "500":
```

```yaml
          description: Internal server error

  /auth/logout:
    post:
      summary: Revokes the user authentication
      security: [bearerAuth: []]
      responses:
        "204":
          description: Logout was successful
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /auth/register:
    post:
      summary: Register a new user
      security: []
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required: [name, username, email, password]
              properties:
                name:
                  type: string
                  example: "John Doe"
                username:
                  type: string
                  example: john-doe
                email:
                  type: string
                  format: email
                  example: john.doe@example.com
                password:
                  type: string
                  format: password
                  example: MyPassword@123
      responses:
        "201":
          description: User registered
        "400":
          description: Invalid credentials
        "409":
          description: There is another user with these credentials
        "429":
```

```yaml
          description: Too many requests
        "500":
          description: Internal server error

  /auth/forgot-password:
    post:
      summary: User requests code to change password
      security: []
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required: [username, email]
              properties:
                username:
                  type: string
                  example: john-doe
                email:
                  type: string
                  format: email
                  example: john.doe@example.com
      responses:
        "200":
          description: If there is an account with this username and e-mail, a code
          ↪  will be sent
        "400":
          description: Invalid credentials
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /auth/reset-password:
    post:
      summary: User sends it's reset code and new password to confirm password change
      security: []
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required: [username, email, password, resetCode]
              properties:
                username:
                  type: string
```

```
                  example: john-doe
                email: # usar aqui???
                  type: string
                  format: email
                  example: john.doe@example.com
                password:
                  type: string
                  format: password
                  example: MyPassword@123
                resetCode:
                  type: string
                  example: "123456"
    responses:
      "200":
        description: Password reseted successfully
      "400":
        description: Invalid credentials
      "401":
        description: Invalid code
      "410":
        description: Expired code
      "429":
        description: Too many requests
      "500":
        description: Internal server error

/auth/change-password:
  post:
    summary: Authenticated user changes it's password
    security: [bearerAuth: []]
    requestBody:
      required: true
      content:
        application/json:
          schema:
            type: object
            required: [newPassword]
            properties:
              newPassword:
                type: string
                format: password
                example: MyPassword@123
    responses:
      "200":
        description: Password changed successfully
      "400":
        description: Invalid password
      "401":
```

```yaml
          description: User unauthorized
        "409":
          description: New password is the same as the old one
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /auth/verify-email:
    post:
      summary: User verifies it's e-mail confirming given code
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required: [email, code]
              properties:
                email:
                  type: string
                  format: email
                  example: john.doe@example.com
                code:
                  type: string
                  example: "123456"
      responses:
        "200":
          description: E-mail verified
        "400":
          description: Invalid credentials
        "410":
          description: Expired code
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /auth/verify-phone:
    post:
      summary: User verifies it's phone confirming given code
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
```

```yaml
              type: object
              required: [phone, code]
              properties:
                phone:
                  type: string
                  pattern: '^\+[1-9]\d{1,14}$'
                  example: "+5551987654321"
                code:
                  type: string
                  example: "123456"
      responses:
        "200":
          description: Phone verified
        "400":
          description: Invalid credentials
        "410":
          description: Expired code
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /auth/register-device:
    post:
      summary: Register a new location device to a user
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required:
                - type
                - value
              properties:
                type:
                  type: string
                  enum: [email, phone]
                  example: email
                value:
                  type: string
                  description: E-mail or phone in internacional pattern (E.164)
                  example: "+55 51 99999-9999"
      responses:
        "201":
          description: Device successfully registered
          content:
```

```yaml
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: string
                    format: uuid
                    example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
                  type:
                    type: string
                  value:
                    type: string
        "400":
          description: Invalid data
        "409":
          description: Device already registered
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /user/get-profile/{user_id}:
    get:
      summary: Get a user's profile information
      security: [bearerAuth: []]
      parameters:
        - name: user_id
          in: path
          required: true
          description: UUID of the user
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Successful operation
          content:
            application/json:
              schema:
                type: object
                properties:
                  user_id:
                    type: string
                    format: uuid
                    example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
                  username:
                    type: string
```

```yaml
                    example: john-doe
                  name:
                    type: string
                    example: "John Doe"
                  profile_image:
                    type: string
                    format: uri
                    example: "https://example.com/uploads/profile-picture.png"
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: Forbidden. User can't fetch this information
        "404":
          description: User not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /user/set-privacy:
    post:
      summary: Defines privacy settings
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
      responses:
        "200":
          description: privacy settings updated successfully
        "400":
          description: Invalid privacy settings
        "401":
          description: unauthorized
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /user/ask-to-follow/{user_id}:
    get:
      summary: Ask to follow another user
      security: [bearerAuth: []]
      parameters:
```

```yaml
      - name: user_id
        in: path
        required: true
        description: UUID of the user
        schema:
          type: string
          format: uuid
          example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
    responses:
      "200":
        description: Successful operation
      "400":
        description: Invalid id
      "401":
        description: unauthorized
      "404":
        description: User not found.
      "429":
        description: Too many requests
      "500":
        description: Internal server error

/user/accept-follower/{user_id}:
  get:
    summary: Accept another user's request to follow
    security: [bearerAuth: []]
    parameters:
      - name: user_id
        in: path
        required: true
        description: UUID of the user
        schema:
          type: string
          format: uuid
          example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
    responses:
      "200":
        description: Successful operation
      "400":
        description: Invalid id
      "401":
        description: unauthorized
      "404":
        description: User or request not found.
      "429":
        description: Too many requests
      "500":
        description: Internal server error
```

```yaml

  /user/change-follower-tier:
    post:
      summary: Change the follower category
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
      responses:
        "200":
          description: Successful operation
        "400":
          description: invalid payload
        "401":
          description: unauthorized
        "404":
          description: Follower not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /user/remove-follower/{user_id}:
    delete:
      summary: Remove another user from the list of followers of the user who made the
      ↪ request
      security: [bearerAuth: []]
      parameters:
        - name: user_id
          in: path
          required: true
          description: UUID of the user to be removed
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Follower removed successfully
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "404":
          description: Follower not found.
```

```yaml
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /user/stop_following/{user_id}:
    delete:
      summary: Stop following another user
      security: [bearerAuth: []]
      parameters:
        - name: user_id
          in: path
          required: true
          description: UUID of the user
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "404":
          description: User not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /user/get-location/{user_id}:
    get:
      summary: Get a user's current location. Granularity depends on the following
      ↪  tier.
      security: [bearerAuth: []]
      parameters:
        - name: user_id
          in: path
          required: true
          description: UUID of the user
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
```

```yaml
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: Doesn't have permission to get the user location
        "404":
          description: User not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /user/send-location:
    post:
      summary: Send location information to the API. May be single data or batch.
      ↪ Asynchronous processing for faster response.
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
      responses:
        "200":
          description: Successful operation
        "400":
          description: invalid payload
        "401":
          description: unauthorized
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/create:
    post:
      summary: Creates a new group_id
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
      responses:
```

```
        "200":
          description: Group created
        "400":
          description: Invalid payload
        "401":
          description: unauthorized
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/invite:
    post:
      summary: Invite an user to be part of a group
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
      responses:
        "200":
          description: Invite sent
        "400":
          description: invalid payload
        "401":
          description: unauthorized
        "403":
          description: User is not allowed to invite to this group
        "404":
          description: User or group not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/respond-invite:
    post:
      summary: Respond an invite to join a group
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
      responses:
```

```
        "200":
          description: Response sent (accept or deny)
        "400":
          description: invalid payload
        "401":
          description: unauthorized
        "404":
          description: Group or invite not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/remove-member/{group_id}/{user_id}:
    delete:
      summary: Remove another user from the list of members of a group who made the
      ↪ request
      security: [bearerAuth: []]
      parameters:
        - name: group_id
          in: path
          required: true
          description: UUID of the group
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
        - name: user_id
          in: path
          required: true
          description: UUID of the user to be removed
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Member removed successfully
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: User doesn't have permission to remove members from this group
        "404":
          description: Member or group not found.
        "429":
          description: Too many requests
```

```yaml
        "500":
          description: Internal server error

  /group/leave/{group_id}:
    delete:
      summary: Leave a group
      security: [bearerAuth: []]
      parameters:
        - name: group_id
          in: path
          required: true
          description: UUID of the group
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: User is no longer member of the group
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: User isn't part of the group
        "404":
          description: Group not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/set-group-privacy:
    post:
      summary: Defines the group privacy settings
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
      responses:
        "200":
          description: group privacy settings updated successfully
        "400":
          description: Invalid privacy settings
        "401":
```

```yaml
          description: unauthorized
        "403":
          description: User doesn't have permission to change the group privacy
          ↪  settings (isn't admin/owner)
        "404":
          description: Group not found
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/set-user-privacy:
    post:
      summary: Defines a user privacy setting regarding a specific group.
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
      responses:
        "200":
          description: member privacy settings updated successfully
        "400":
          description: Invalid privacy settings
        "401":
          description: unauthorized
        "403":
          description: User isn't a group member.
        "404":
          description: Group not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/ask-to-follow/{group_id}:
    get:
      summary: Ask to follow a group
      security: [bearerAuth:[]]
      parameters:
        - name: group_id
          in: path
          required: true
          description: UUID of the group
          schema:
            type: string
```

```yaml
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: Unauthorized
        "404":
          description: Group not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/accept-follower/{user_id}:
    get:
      summary: Accept another user's request to follow
      security: [bearerAuth: []]
      parameters:
        - name: user_id
          in: path
          required: true
          description: UUID of the user
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "404":
          description: User or request not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/remove-follower/{group_id}/{user_id}:
    delete:
      summary: Remove an user from the list of followers of the group
      security: [bearerAuth: []]
      parameters:
```

```yaml
        - name: group_id
          in: path
          required: true
          description: UUID of the group
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
        - name: user_id
          in: path
          required: true
          description: UUID of the user to be removed
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Follower removed successfully
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: User doesn't have permission to remove followers from this
          ↪ group
        "404":
          description: Follower not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/stop_following/{group_id}:
    delete:
      summary: Stop following a group
      security: [bearerAuth: []]
      parameters:
        - name: user_id
          in: path
          required: true
          description: UUID of the user
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
```

```yaml
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "404":
          description: Group not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /group/get-location/{group_id}:
    get:
      summary: Get a group's current location. Granularity depends on the following
      ↪ tier.
      security: [bearerAuth: []]
      parameters:
        - name: group_id
          in: path
          required: true
          description: UUID of the group
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: Doesn't have permission to get the group location
        "404":
          description: Group not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /trip/get-trip-sumary/{trip_id}:
    get:
      summary: Get a trip's basic informaiton
      security: [bearerAuth: []]
      parameters:
        - name: trip_id
```

```yaml
          in: path
          required: true
          description: UUID of the trip
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: Doesn't have permission to view this trip
        "404":
          description: Trip not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /trip/get-trip-details/{trip_id}:
    get:
      summary: Get a trip's complete informaiton
      security: [bearerAuth: []]
      parameters:
        - name: trip_id
          in: path
          required: true
          description: UUID of the trip
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: Doesn't have permission to view this trip
        "404":
          description: Trip not found.
        "429":
```

```yaml
          description: Too many requests
        "500":
          description: Internal server error

  /trip/list-trips/user/{user_id}:
    get:
      summary: List a user's trips
      security: [bearerAuth: []]
      parameters:
        - name: user_id
          in: path
          required: true
          description: UUID of the user
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: Doesn't have permission to list this user's trip
        "404":
          description: User not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /trip/list-trips/group/{group_id}:
    get:
      summary: List a groups's trips
      security: [bearerAuth: []]
      parameters:
        - name: group_id
          in: path
          required: true
          description: UUID of the group
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
```

```yaml
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: Doesn't have permission to list this group's trip
        "404":
          description: User not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /trip/get-user-current-trip/{user_id}:
    get:
      summary: Fetch the current trip from a user
      security: [bearerAuth: []]
      parameters:
        - name: user_id
          in: path
          required: true
          description: UUID of the user
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: Doesn't have permission to get this user's trip
        "404":
          description: User not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /trip/get-group-current-trip/{group_id}:
    get:
      summary: Fetch the current trip from a group
      security: [bearerAuth: []]
      parameters:
```

```yaml
        - name: group_id
          in: path
          required: true
          description: UUID of the group
          schema:
            type: string
            format: uuid
            example: "c21f969b-5f03-4333-9a9d-6a5566e8a5e7"
      responses:
        "200":
          description: Successful operation
        "400":
          description: Invalid id
        "401":
          description: unauthorized
        "403":
          description: Doesn't have permission to get this group's trip
        "404":
          description: Group not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error

  /location/send-location:
    post:
      summary: Fallback endpoint to send location data if Kinesis fails.
      security: [bearerAuth: []]
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
      responses:
        "200":
          description: Location data received successfully
        "400":
          description: Invalid data
        "401":
          description: unauthorized
        "404":
          description: user not found.
        "429":
          description: Too many requests
        "500":
          description: Internal server error
```

Apêndice C Modelagem do banco de dados relacional

```
1
2  Table users {
3    user_id varchar [default: `gen_random_uuid()`]
4    username varchar [not null]
5    name varchar [not null]
6    email varchar[] [not null]
7    phone varchar
8    created_at timestamp [not null, default: `now()`]
9    updated_at timestamp [not null, default: `now()`]
10   deleted_at timestamp
11   profile_image varchar
12   location_data varchar [not null, default: "", note: "Default is user_id"]
13   privacy_settings varchar [not null, default: "default"]
14
15   indexes {
16     user_id [pk]
17     username
18     email
19     phone
20   }
21 }
22
23 Table follows {
24   follower_id varchar [not null, ref: > users.user_id]
25   following_id varchar [not null, ref: > users.user_id]
26   tier varchar [default: 'regular']
27   followed_at timestamp [not null, default: `now()`]
28
29   indexes {
30     (follower_id, following_id) [pk]
31   }
32 }
33
34 Table groups {
35   group_id varchar [default: `gen_random_uuid()`]
36   owner_id varchar [not null, ref: > users.user_id]
37   name varchar
38   created_at timestamp [not null, default: `now()`]
39   updated_at timestamp [not null, default: `now()`]
40   deleted_at timestamp
41   profile_image varchar
42   location_data varchar [not null, default: "", note: "Default is group_id"]
43
44   indexes {
45     group_id [pk]
```

```
  }
}

Table group_members {
  group_id varchar [not null, ref: > groups.group_id]
  user_id varchar [not null, ref: > users.user_id]
  joined_at timestamp [not null, default: `now()`]

  indexes {
    (group_id, user_id) [pk]
  }
}

Table trips {
  trip_id varchar [default: `gen_random_uuid()`]
  title varchar [not null]
  description text
  created_at timestamp [not null, default: `now()`]
  updated_at timestamp [not null, default: `now()`]
  deleted_at timestamp
  trip_image varchar
  start_date date [not null, default: `now()`]
  end_date date
  location_data varchar [not null, default: "", note: "Default is trip_id"]

  indexes {
    trip_id [pk]
  }
}

Table user_trips {
  user_id varchar [ref: > users.user_id]
  trip_id varchar [ref: > trips.trip_id]

  indexes {
    (user_id, trip_id) [pk]
  }
}

Table group_trips {
  group_id varchar [ref: > groups.group_id]
  trip_id varchar [ref: > trips.trip_id]

  indexes {
    (group_id, trip_id) [pk]
  }
}
```

Referências

[1] Posie Aagaard, Bijan Dinyarian, Omar Abduljabbar, and Kim-Kwang Raymond Choo. Family locating sharing app forensics: Life360 as a case study. *Forensic Science International: Digital Investigation*, 44:301478, 2023.

[2] Agência Nacional de Telecomunicações – ANATEL. Painel cobertura móvel. `https://informacoes.anatel.gov.br/paineis/infraestrutura/cobertura-movel`. Painel interativo com mapas de cobertura móvel por município e tecnologia (3G, 4G). Dados baseados em licenciamento e estimativas regulatórias. Acesso em: 03 set. 2025.

[3] Amazon Web Services. Amazon api gateway. `https://aws.amazon.com/api-gateway/`, 2025. Serviço gerenciado para criação e gerenciamento de APIs REST e WebSocket. Free Tier: 1 milhão de chamadas/mês por 12 meses.

[4] Amazon Web Services. Amazon cognito. `https://aws.amazon.com/cognito/`, 2025. Autenticação e gerenciamento de usuários. Free Tier: 50 mil MAUs mensais, sempre gratuito.

[5] Amazon Web Services. Amazon dynamodb. `https://aws.amazon.com/dynamodb/`, 2025. Banco de dados NoSQL escalável e de baixa latência. Free Tier: 25 GB + 25 unidades de leitura e escrita/mês, sempre gratuito.

[6] Amazon Web Services. Amazon kinesis data streams. `https://aws.amazon.com/kinesis/data-streams/`, 2025. Captura e processamento de dados em tempo real. Free Tier: 1 MB/s entrada e saída por 12 meses.

[7] Amazon Web Services. Amazon relational database service (rds). `https://aws.amazon.com/rds/`, 2025. Banco de dados relacional gerenciado. Free Tier: 750 horas/mês + 20 GB de armazenamento por 12 meses.

[8] Amazon Web Services. Amazon simple email service (ses). `https://aws.amazon.com/ses/`, 2025. Envio e recebimento de e-mails em escala. Free Tier: 62 mil e-mails/mês via EC2.

[9] Amazon Web Services. Amazon simple queue service (sqs). `https://aws.amazon.com/sqs/`, 2025. Filas de mensagens totalmente gerenciadas. Free Tier: 1 milhão de solicitações/mês, sempre gratuito.

[10] Amazon Web Services. Amazon simple storage service (s3). `https://aws.amazon.com/s3/`, 2025. Armazenamento de objetos com alta durabilidade e disponibilidade. Free Tier: 5 GB + 20 mil GET + 2 mil PUT/mês por 12 meses.

[11] Amazon Web Services. Aws identity and access management (iam). https://aws.amazon.com/iam/, 2025. Controle de acesso a recursos AWS. Uso gratuito e ilimitado.

[12] Amazon Web Services. Aws lambda. https://aws.amazon.com/lambda/, 2025. Execução de código sob demanda sem servidores. Free Tier: 1 milhão de execuções e 400 mil GB-segundos/mês, sempre gratuito.

[13] Amazon Web Services. Aws named as a leader in the 2025 gartner magic quadrant for strategic cloud platform services. https://aws.amazon.com/pt/resources/analyst-reports/gartner/magic-quadrant-for-strategic-cloud-platform-services-mq/, 2025. A AWS ocupa a mais alta posição em "capacidade de execução" segundo a Gartner.

[14] Amazon Web Services. Aws systems manager parameter store. https://aws.amazon.com/systems-manager/features/#Parameter_Store, 2025. Armazenamento seguro de parâmetros e segredos com criptografia KMS. Free Tier: até 10.000 parâmetros Standard, incluindo SecureString.

[15] S. Belligoni, K. A. Stevens, S. Hasan, and H. Yu. Privacy and security concerns with passively collected location data for digital contact tracing among u.s. college students. *PLOS ONE*, 18(11):e0294419, 2023. Published: Nov 22, 2023.

[16] Antoine Boutet and Victor Morel. "i'm not for sale" – perceptions and limited awareness of privacy risks by digital natives about location data. *arXiv preprint arXiv:2502.11658v3*, 2025.

[17] Concur Technologies, Inc. Tripit – trip planner & flight tracker. https://www.tripit.com/web. Aplicativo/plataforma que organiza automaticamente seus planos de viagem ao encaminhar confirmações por e-mail; cria um itinerário completo, sincroniza com calendário, oferece acesso multiplataforma e funcionalidades Pro como alertas de voos em tempo real, mapas interativos de aeroportos, acompanhamento de tarifas, opções de transporte e estatísticas de viagem. Acesso em: 03 set. 2025.

[18] Bruno Couriol. Airbnb Releases Tool to Convert Large Codebases to TypeScript. InfoQ (Online), 2020.

[19] Figma, Inc. Figma – collaborative interface design tool. https://www.figma.com/, 2025. Ferramenta online de design de interfaces, prototipação e colaboração em tempo real baseada em nuvem.

[20] FindPenguins. Findpenguins – travel tracker app. https://findpenguins.com/. Aplicativo gratuito para planejar, rastrear e compartilhar viagens; gera vídeos 3D do itinerário (flyover), permite criar livros fotográficos personalizados, registrar "Footprints"

(postagens de viagem), explorar roteiros de outros viajantes e funciona offline com baixo consumo de bateria. Acesso em: 03 set. 2025.

[21] GitHub, Inc. Github – the complete developer platform. https://github.com/, 2025. Plataforma de hospedagem de código-fonte e colaboração baseada em Git, com suporte a issues, pull requests e automações de CI/CD.

[22] Google LLC. Google maps. https://www.google.com/maps. Plataforma de mapas online com rotas para carro, transporte público, bicicleta e a pé; imagens de satélite, vistas panorâmicas 360° (Street View), informações de trânsito em tempo real, exploração de locais e listagem de estabelecimentos. Acesso em: 03 set. 2025.

[23] Lambus GmbH. Lambus – all-in-one travel planner. https://www.lambus.com/. Plataforma para planejamento completo de viagens — permite criar itinerários com múltiplas paradas, enviar confirmações por e-mail, importar arquivos GPX, gerenciar documentos, despesas, notas e fotos; recursos colaborativos em grupo e funcionalidades PRO como cálculo de duração e distância entre paradas, alertas de portão, clima local, exportação de dados, modo offline e muito mais. Acesso em: 03 set. 2025.

[24] Life360, Inc. Life360 – family safety and location-sharing app. https://intl.life360.com/. Aplicativo de segurança familiar com compartilhamento de localização, detecção de acidentes, assistência rodoviária e integração com rastreadores Tile. Acesso em: 03 set. 2025.

[25] Meta Platforms, Inc. React native — learn once, write anywhere. https://reactnative.dev/, 2025. Framework multiplataforma para desenvolvimento móvel com um único código-base para Android e iOS.

[26] Microsoft Corporation. Typescript — javascript with syntax for types. https://www.typescriptlang.org/, 2025. Linguagem que adiciona tipagem estática e ferramentas para escalar desenvolvimento JavaScript.

[27] Ori App Studio GmbH. Roadie – trip planner road trip itinerary builder. https://roadietripplanner.com/. Portal para planejamento de viagens rodoviárias com múltiplos pontos; permite criação de roteiros em mapa interativo com cálculo de distâncias e tempo de trajeto; busca de atrações, restaurantes e hospedagem ao longo da rota; inspiração com roteiros prontos; compartilhamento colaborativo e exportação de itinerários. Acesso em: 03 set. 2025.

[28] OsmAnd BV. Osmand – offline maps and navigation. https://osmand.net/pt/. Aplicativo de mapeamento e navegação completamente offline; oferece mapas vetoriais detalhados, navegação por voz com recálculo automático, rotas para carro, bicicleta, pedestre; suporte a pontos de interesse (POI), modos especializados (como trilhas,

navegação pública), registro de rotas GPX, plugins como curvas de nível e integração com Wikipedia/Wikivoyage. Acesso em: 03 set. 2025.

[29] Pin Traveler LLC. Pin traveler – travel tracker & world travel map. https://pintraveler.net/. Aplicativo e plataforma web para rastreamento e registro de viagens; permite criar um mapa personalizado com "pins" de locais visitados, exportar e compartilhar mapas, registrar fotos, itinerários e estatísticas de viagem. Acesso em: 03 set. 2025.

[30] Polarsteps. Polarsteps. https://www.polarsteps.com. Aplicativo de planejamento e registro de viagens. Acesso em: 03 set. 2025.

[31] Naveen C. Ramachandrappa. A Comparative Analysis of Native vs React Native Mobile App Development. *Int. Journal of Computer Trends and Technology*, 72(9):57–62, 2024.

[32] Joshua D. Scarsbrook, Mark Utting, and Ryan K. L. Ko. Typescript's evolution: An analysis of feature adoption over time, 2023.

[33] Share Location. Share location – aplicativo de compartilhamento de localização em tempo real. https://sharelocation.app/. Aplicativo para compartilhar sua localização em tempo real com amigos e familiares; sem necessidade de cadastro ou instalação por parte dos receptores; inclui histórico de localização, alertas de chegada/saída (Place Alerts), monitoramento de bateria, velocidade de condução e condições climáticas. Acesso em: 03 set. 2025.

[34] Stippl B.V. Stippl – the all-in-one travel planner. https://www.stippl.io/. Plataforma que integra planejamento de itinerários, orçamento de viagem, listas de embalagem, registro de memórias e funcionalidades baseadas em IA para criação de roteiros; permite gestão de transporte, hospedagem, atividades, despesas, geração de vídeos em 3D e colaboração com amigos. Acesso em: 03 set. 2025.

[35] Visited. Visited app. https://visitedapp.com/. Aplicativo para mapear viagens, descobrir destinos, planejar itinerários e visualizar estatísticas de viagem. Acesso em: 03 set. 2025.

[36] Wanderlog, Inc. Wanderlog – free travel road trip planner. https://wanderlog.com/. Aplicativo/plataforma para planejar viagens e road trips gratuitamente; permite colaboração em tempo real, criação de itinerários com visualização em mapa, importação automática de reservas via e-mail, rastreamento de orçamento, listas de embalagem, otimização de rota, recomendações inteligentes e funcionalidades Pro como acesso offline, exportar para Google Maps e integração com Gmail. Acesso em: 03 set. 2025.