CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DA PRAIA GRANDE
CURSO SUPERIOR DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

TRAVELER

A secretária de viagens

RELATÓRIO TÉCNICO

Antonio Vinícius Ferreira
Gabriel Benitez Pedutti
Karina Ricci Cinanena Kuhne

PRAIA GRANDE
JUNHO/2025

Antonio Vinícius Ferreira

Gabriel Benitez Pedutti

Karina Ricci Cinanena Kuhne

TRAVELER
A secretária de viagens

Relatório técnico apresentado como requisito parcial para obtenção de aprovação na disciplina Trabalho de Graduação II, no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia de Praia Grande.

Orientador: Prof. Joseffe Barroso de Oliveira

PRAIA GRANDE
JUNHO/2025

Antonio Vinícius Ferreira

Gabriel Benitez Pedutti

Karina Ricci Cinanena Kuhne

TRAVELER
A secretária de viagens

Relatório técnico apresentado como requisito parcial para obtenção de aprovação na disciplina Trabalho de Graduação II, no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia de Praia Grande.

Praia Grande, 27 de junho de 2025

Banca Avaliadora

_____

Nome: Professor Joseffe Barroso de Oliveira - FATEC Praia Grande

_____

Nome: Professor Jonatas Cerqueira Dias - FATEC Praia Grande

_____

Nome: Professora Miriam Vidal Correia Franzese - FATEC Praia Grande

A Deus,

As nossas famílias,

Aos professores e amigos

AGRADECIMENTOS

Agradecemos, primeiramente, a Deus por nos ter dado a vida e a capacidade cognitiva para aprender e produzir, nossos familiares e amigos por todo o apoio e incentivo, a Geovana Vieira e Thiago Sansalone por toda parceria e ajuda com todos os trabalhos durante os semestres e ao professor e nosso orientador Joseffe Barroso por todo direcionamento, apoio e paciência.

"O homem não teria alcançado o possível se, repetidas vezes, não tivesse tentado o impossível."

Max Weber

RESUMO

O mercado de viagens independentes exige um esforço considerável para o viajante, obrigando-o a utilizar diversas plataformas de contratação de transportes, hospedagens, passeios e despesas para se organizar. Pensando nisso o presente trabalho descreve o desenvolvimento do Traveler, um aplicativo Android destinado ao gerenciamento de viagens pessoais e independentes. A solução proposta busca consolidar essas informações em um único lugar, promovendo usabilidade, organização e centralização de dados. O front-end do aplicativo foi desenvolvido em conectado a um back-end em Node.js, que consome um banco de dados PostgreSQL via Prisma ORM, tudo isso em uma infraestrutura de nuvem AWS (EC2, RDS, S3). A metodologia ágil Scrum foi parcialmente adotada com sprints de duas semanas para garantir entregas incrementais e adaptativas. Como estamos lidando com dados pessoais, utilizamos técnicas modernas de segurança, como criptografia de senhas com bcrypt, isolamento em subnets privadas e arquitetura em múltiplas camadas. O projeto visa facilitar o planejamento de viagens, aumentar o controle financeiro dos usuários e reduzir a dependência de agências ou serviços terceirizados.

Palavras-chave: gerenciamento de viagens; React Native; Node.js; PostgreSQL; AWS; aplicativo mobile.

ABSTRACT

The independent travel market demands considerable effort from travelers, requiring them to use multiple platforms to organize transportation, accommodation, activities, and expenses. With this in mind, the present work describes the development of *Traveler*, an Android application designed for managing personal and independent trips. The proposed solution aims to consolidate this information in a single location, promoting usability, organization, and centralized data management. The app's front-end was developed in React Native and connected to a back-end built with Node.js, which communicates with a PostgreSQL database via Prisma ORM, all deployed on an AWS cloud infrastructure (EC2, RDS, S3). The Scrum agile methodology was partially adopted, with two-week sprints to ensure incremental and adaptable deliveries. As the system handles personal data, modern security techniques were implemented, such as password encryption using bcrypt, isolation within private subnets, and a multi-layered architecture. The project seeks to simplify travel planning, enhance users' financial control, and reduce reliance on travel agencies or third-party services.

Keywords: travel management; React Native; Node.js; PostgreSQL; AWS; mobile application.

LISTA DE ABREVIATURAS

API - Application Programming Interface

AWS - Amazon Web Services

CRUD - Create, Read, Update, Delete (Criar, Ler, Atualizar, Deletar)

EAP - Estrutura Analítica do Projeto

EC2 - Elastic Compute Cloud (Serviço de computação da AWS)

HTTP - HyperText Transfer Protocol

ID - Identificador

IOS - iPhone Operating System (Sistema Operacional da Apple)

LGPD - Lei Geral de Proteção de Dados

ORM - Object-Relational Mapping (Mapeamento Objeto-Relacional)

PK - Primary Key (Chave Primária)

PMBOK - Project Management Body of Knowledge

POST - Método HTTP utilizado para envio/criação de dados

RDS - Relational Database Service (Serviço de banco relacional da AWS)

RAM - Random Access Memory

RNF - Requisito Não Funcional

RF - Requisito Funcional

S3 - Simple Storage Service (Serviço de armazenamento da AWS)

SQL - Structured Query Language

UI - User Interface (Interface do Usuário)

VPC - Virtual Private Cloud

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente, dentre os mais de 3,5 milhões de aplicativos mobiles existentes entre os universos Android e IOS, aqueles que possuem como escopo assuntos relacionados a viagens apresentam serviços que diversas vezes são nichados ou resolvem apenas um pequeno problema em comparação ao processo complexo que pode envolver a organização de uma viagem.

Considerando viagens completas e bem planejadas, com todos os serviços adquiridos com antecedência, essa segregação dificulta a organização de toda a jornada do usuário, forçando-o a consultar diversos locais diferentes e organizar essas informações em formatos pouco adaptados ao contexto de viagens como é o caso das planilhas.

O avanço constante da tecnologia e das linguagens de programação permite que os programadores utilizem de uma única tecnologia para diversas plataformas distintas e concorrentes, facilitando a disseminação das aplicações e permitindo um custo e tempo menores.

A falta de uma aplicação centralizada neste ramo de mercado aliada a possibilidade de reutilizar os softwares em diversos sistemas operacionais e as facilidades que os frameworks oferecem aos desenvolvedores, representa uma oportunidade para produzir algo novo e útil para uma grande parcela de usuários.

Em um mercado em que 112,6 milhões de passagens aéreas para voos domésticos, entre nacionais e internacionais, foram comercializadas apenas no ano de 2023 (Tafarelo, 2024), desconsiderando viagens de ônibus e carro, no cenário de um país que possui 249 milhões de smartphones, cerda de 1,2 para cada habitante (FGV, 2024), a possibilidadede consolidar e manipular os dados de um planejamento na palma da mão, consultá-lo e alterá-lo a qualquer momento e compartilhar com outras pessoas cadastradas namesma plataforma tem o poder de facilitar muito a vida dos usuários.

1.1 PROBLEMA

Falta de um aplicativo centralizado para gerir informações, documentos e gastos de uma viagem, a necessidade de gestão financeira e documental integrada.

1.2 OBJETIVO GERAL

Desenvolver um aplicativo de gerenciamento de viagens para facilitar o planejamento e a organização por parte dos utilizadores, consolidando os dados em um único lugar.

O front-end da aplicação será codificado em React Native, consumindo a API do back-end que será desenvolvido utilizando o framework Node.js, que realizará transações em um banco de dados PostgreSQL através do Prisma, um ORM com excelente integração entre Javascript, Typescript e Postgres.

2 TECNOLOGIAS UTILIZADAS

2.1 REACT NATIVE

O React Native é um framework JavaScript de código aberto, criado em 2015 pela empresa Facebook. Ele permite o desenvolvimento de aplicações móveis utilizando uma única base de código, que pode ser portada tanto para Android quanto para iOS, facilitando a criação de aplicativos multiplataforma

Além da compatibilidade com múltiplos sistemas operacionais, o framework possui uma arquitetura modular que favorece o reaproveitamento de código em diversas partes da aplicação por meio do conceito de componentes reutilizáveis (Cunha, 2023).

O Facebook descreve essa abordagem com o lema “aprenda uma vez, escreva em qualquer lugar”, o que sintetiza a proposta do React Native. A tecnologia é baseada no React, e suas principais vantagens são incorporadas ao framework, aplicando-as no contexto de aplicações nativas. A diferença entre ambos está na forma de execução: o React tradicional é renderizado no navegador, enquanto o React Native é interpretado em dispositivos móveis por meio de máquinas JavaScript, como V8 (Android) ou JavaScriptCore (iOS) (Danielsson, 2016).

Entre suas principais vantagens destacam-se a reutilização de código, acelerando o processo de desenvolvimento, e a facilidade de adaptação entre plataformas. Por outro lado, suas principais limitações incluem o desempenho inferior em comparação ao desenvolvimento nativo (que é feito em Java/Kotlin para Android e Swift/Objective-C para IOS), uma vez que o código precisa ser interpretado em tempo de execução, e a complexidade na atualização do framework, o que pode causar atrasos na disponibilização de novos recursos dos sistemas operacionais.

2.2 NODE.JS

O Node.JS é um ambiente de execução JavaScript baseado no motor V8 (máquina virtual JavaScript do Google Chrome).

Ele é um framework de código aberto criado em 2009 por Ryan Dahl que permite a codificação multiplataforma e execução de código JavaScript sem a dependência de um navegador, possibilitando a construção de aplicações server side (Bessa, 2023).

Dentre suas principais características, destacam-se a arquitetura orientada a eventos e o modelo de execução não bloqueante, permitindo múltiplas conexões simultaneamente sem o comprometimento do desempenho do sistema e sendo ideal para sistemas que demandam alta performance e escalabilidade, como API’s, sistemas em tempo real (chats e transmissões de mídia) e arquiteturas baseadas em microsserviços (Corrente, 2025).

2.3 BANCO DE DADOS RELACIONAL

Um banco de dados relacional é um sistema composto por tabelas organizadas em linhas e colunas, que podem ser relacionadas entre si de forma estruturada e gerenciável. Cada tabela representa uma entidade do mundo real, enquanto cada coluna descreve uma característica dessa entidade, e as linhas correspondem às suas ocorrências ou registros.

As relações entre tabelas são estabelecidas por meio de chaves primárias — colunas com valores únicos que identificam cada registro (como um ID) — e chaves estrangeiras, que consistem em valores de chave primária provenientes de outra tabela, permitindo a criação de vínculos entre diferentes entidades do banco de dados (Calanca, 2023).

2.4 POSTGRESQL

O PostgreSQL é um sistema de banco de dados relacional de código aberto que surgiu em meados de 1986 em Berkeley, sendo uma ideia inicial do professor de ciência da computação Michael Stonebraker.

Conhecido por seu desempenho e escalabilidade, um dos seus pontos fortes é o gerenciamento eficiente de acessos simultâneos controlados por meio do uso do MVCC (Multiversion Concurrency Control), um método onde nenhuma transação sobrescreve o banco original, mas sim uma nova instância que é criada por ele e utilizada posteriormente para atualizar a instância original, garantindo assim a consistência dos dados (IBM, [S.D.]).

2.5 PRISMA

Prisma é um ORMs ou Object Relational Mapping, uma forma de alinhar o código com o banco de dados, ajudando desenvolvedor a correlacionar os objetos com as estruturas presentes no banco de dados, permitindo trabalhar com bancos de dados relacionais utilizando objetos de forma a abstrair o código SQL, facilitando a manutenção da aplicação e de uma forma mais global: a comunicação com o banco de dados e o restante da aplicação (TIDB, 2024).

2.6 AWS

A AWS é uma plataforma robusta de serviços de computação em nuvem lançado pela Amazon em 2006 que oferece mais de 240 soluções de TI, entre elas: bancos de dados; machine learning; gerenciamento de aplicativos; segurança; backup e armazenamento (Amazon, [S.D.]).

Os recursos da AWS retiram a necessidade das empresas em manter um servidor, garantem a segurança e a disponibilidade dos dados, mantendo diversos datacenters espalhados por vários nós e realizando o gerenciamento automático da distribuição entre eles.

Devido aos diversos benefícios, como os citados anteriormente, a arquitetura baseada em nuvem esteve entre as 10 principais tendências tecnológicas estratégicas para o ano de 2024, segundo a Gartner (McCartney, 2023).

3 PLANEJAMENTO DO PROJETO

3.1 ESCOPO DO PROJETO

O escopo deste projeto contempla duas funcionalidades principais:

- Gerenciar os itens de uma viagem como transporte, hospedagem e passeios, oferecendo também a possibilidade de upload de documentos para acesso rápido em momentos de embarque e check-in.
- Oferecer um resumo e histórico dos gastos de todas as viagens cadastradas pelo usuário para um melhor controle financeiro.

3.1.1 Cenário Atual

Atualmente as pessoas utilizam vários aplicativos diferentes com a mesma finalidade para realizar compra de passagens, reserva de hospedagens e compra de pacotes turísticos, como Booking, Trivago, Airbnb, Skyscanner e Decolar, sendo necessário além de ter todos instalados, utilizar aplicações diferentes para cada finalidade desejada.

3.1.2 Cenário a ser atingido

O aplicativo procura facilitar a organização de quem viaja, armazenando as informações e documentos de vários aplicativos diferentes em um único lugar, definindo alarmes e gerenciando despesas.

O sistema busca facilitar a organização das viagens para os usuários, diminuindo a quantidade de aplicativos e e-mails que ele utiliza toda vez que precisa confirmar horários, realizar check-ins ou embarcar em transportes, além de permitir uma melhor organização financeira e previsões de gastos caso o usuário pretenda visitar o mesmo lugar novamente em curto prazo.

3.1.3 Premissas

Premissas, de acordo com o guia PMBOK, de maneira resumida são hipóteses assumidas como verdadeiras para a definição do planejamento do projeto (PMI, 2008).

As premissas identificadas para o projeto foram:

- O usuário possui acesso à internet para baixar o aplicativo.
- Todos os dados inseridos pelo usuário serão considerados confiáveis, não havendo validação de autenticidade para despesas, documentos ou informações pessoais.
- O aplicativo será utilizado em dispositivos móveis Android.
- O back-end estará disponível em um ambiente com alta disponibilidade.
- O aplicativo assumirá que uma viagem só pode ser cadastrada por um único usuário responsável, ainda que futuramente haja a possibilidade de desenvolver a função de compartilhamento.
- O sistema deverá possuir os dados geográficos (país, estado, município) previamente carregados via serviço externo ou base local.
- O aplicativo sincronizará os dados com o servidor sempre que for identificada uma conexão com a internet, mas as funcionalidades estarão disponíveis offline.
- O projeto será desenvolvido por um time acadêmico, com papéis definidos e prazos estabelecidos pelo cronograma do curso.
- O escopo não contempla a venda de pacotes turísticos ou passagens diretamente pelo app, apesar de possibilitar o desenvolvimento dessa funcionalidade futuramente.

3.1.4 Riscos

Riscos em projetos são **eventos ou condições incertas** que, caso ocorram, podem afetar negativamente ou positivamente os objetivos do projeto, e gerenciar os riscos é fundamental para planejar uma resposta em prol de mitigá-los ou até evitá-los. Os riscos levantados para o projeto foram:

Tabela 1: Levantamento de riscos

ID	Descrição do Risco	Probabilidade	Impacto	Severidade	Plano de Resposta	Responsável
R01	Atraso na entrega do aplicativo	Alta	Alta	Crítica	Reforçar cronograma	Todos os desenvolvedores
R02	Falta de conhecimento em React Native	Média	Média	Moderada	Realizar capacitações no início do projeto	Desenvolvedor front-end
R03	Perda de dados por erro de versão ou commit	Baixa	Média	Moderada	Utilizar Git com controle de branches e CI/CD	Todos os desenvolvedores
R04	Mudança nos requisitos durante o projeto	Média	Alta	Alta	Manter escopo fechado e validar requisitos com o orientador	Analista funcional
R05	Dificuldade na integração da API com o app mobile	Média	Média	Moderada	Definir contrato de API claro e testar com Postman	Back-end e front-end

Fonte: elaborado pelos autores (2024)

3.1.5 Restrições

De acordo com o guia PMBOK, restrições são limitações internas e externas específicas que interferem na execução do projeto (PMI, 2008). As restrições levantadas para o projeto foram:

- Os integrantes da equipe só poderão atuar no projeto em períodos extra aula e trabalho.
- A entrega do projeto não pode ultrapassar a data estipulada pela instituição de ensino.

3.2 METODOLOGIA

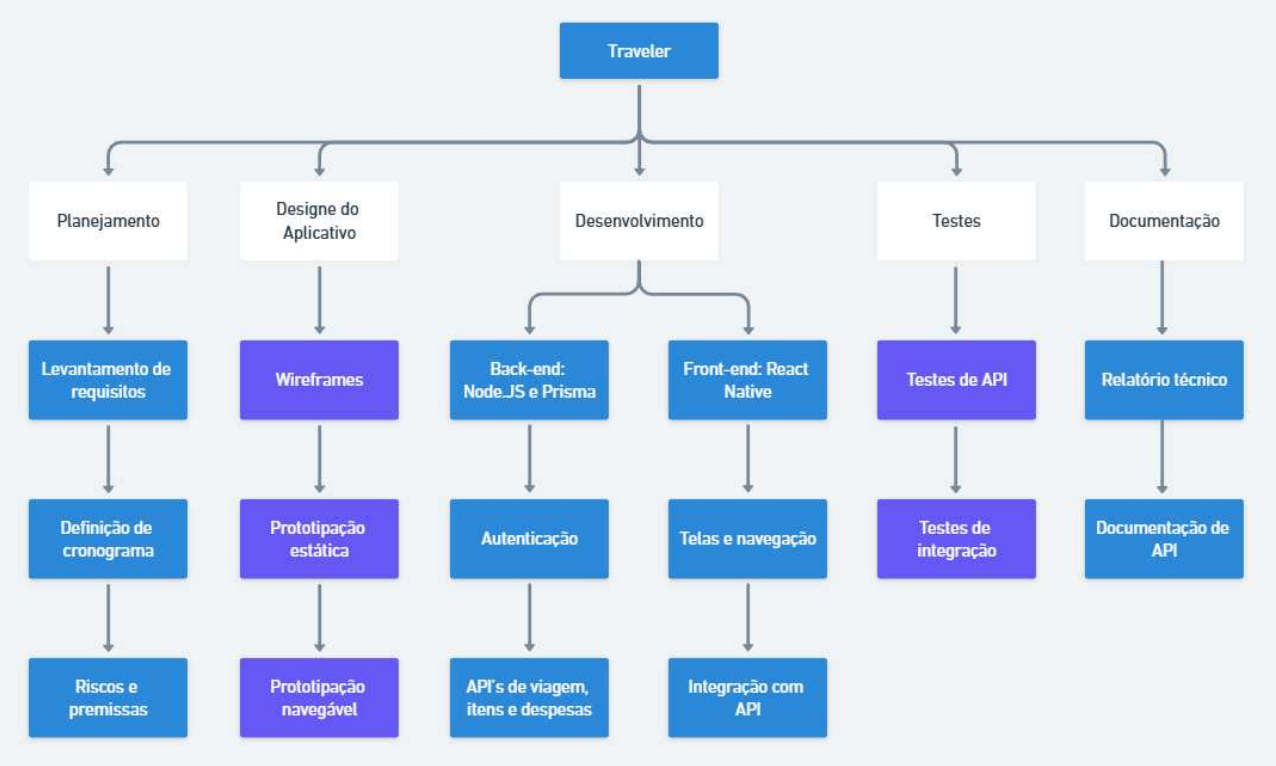
O desenvolvimento do aplicativo será realizado de forma incremental (com várias entregas pequenas que complementarão o desenvolvimento continuamente), adotando parcialmente a metodologia ágil Scrum. Para definição das entregas, será realizada a cerimônia de Sprint Planning, que, segundo Schwaber e Sutherland (2020), “é o evento no qual é planejado o trabalho a ser realizado durante a sprint, resultando em um plano coerente criado por toda a equipe Scrum”. Nessa reunião, são definidos o objetivo da sprint, os itens do backlog priorizados e a estratégia para sua implementação.

Foi definido que cada sprint terá duração de duas semanas, possibilitando entregas ágeis e validação progressiva dos requisitos. Esse modelo permite um ciclo de desenvolvimento estruturado e incremental, além de facilitar possíveis adaptações na solução.

3.3 ESTRUTURA ANALÍTICA DO PROJETO

A EAP (Estrutura Analítica do Projeto) é uma ferramenta de gestão de projetos que serve para dividir o escopo do projeto em partes menores, mais gerenciáveis, de forma a facilitar o controle de prazos, recursos, riscos e entregas. A EAP do projeto Traveler foi definida da seguinte forma:

Figura 1: EAP - Estrutura Analítica do Projeto



Fonte: elaborado pelos autores (2024)

3.4 CRONOGRAMA

Figura 2: Cronograma

Funcionalidade	Cronograma Traveler									
Sprint	Janeiro		Fevereiro		Março		Abril		Maio	
	Sprint 1	Sprint 2	Sprint 1	Sprint 2	Sprint 1	Sprint 2	Sprint 1	Sprint 2	Sprint 1	Sprint 2
Infra e Banco de Dados										
Back End										
Tela de Cadastro e Login										
Menus laterais										
Tela de perfil										
Telas de cadastro e exibição de viagem										
Telas de cadastro e exibição de hospedagem										
Telas de cadastro e exibição de transporte										
Telas de cadastro e exibição de passeio										
Telas de cadastro e exibição de despesas										

Fonte: elaborado pelos autores (2024)

3.5 PAPÉIS E RESPONSABILIDADES

O Traveler foi desenvolvido por uma equipe com 3 integrantes:

- Antonio Vinícius Ferreira: Analista funcional. Responsável por desenvolver a documentação, requisitos, regras de negócio e prototipação.

- Gabriel Benitez Pedutti: Desenvolvedor front-end. Responsável por desenvolver o front-end utilizando o React Native e integrando-o ao back-end.
- Karina Ricci Cinanena Kuhne: Desenvolvedora back-end e devops. Responsável por desenvolver o back-end, projetar e implementar o banco de dados e conectá-lo ao back-end.

3.6 PROTÓTIPOS DO PROJETO

Figura 3: Tela de carregamento

Fonte: elaborado pelos autores (2024)

Tela de carregamento que será exibida assim que o usuário acessar o aplicativo.

Figura 4: Tela de log-in

Fonte: elaborado pelos autores (2024)

Tela de login com funcionalidades de cadastro, login com e-mail e senha com opção de lembrar os dados do usuário ou login com redes sociais.

Figura 5: Tela inicial

Fonte: elaborado pelos autores (2024)

Tela inicial do aplicativo com os botões “Menu” e “Usuário” na parte superior, “Cadastre uma nova viagem” e um resumo da próxima viagem do usuário com a possibilidade de acessar a tela de resumo da viagem a partir do botão “Mais informações”.

Figura 6: Tela de menu

Fonte: elaborado pelos autores (2024)

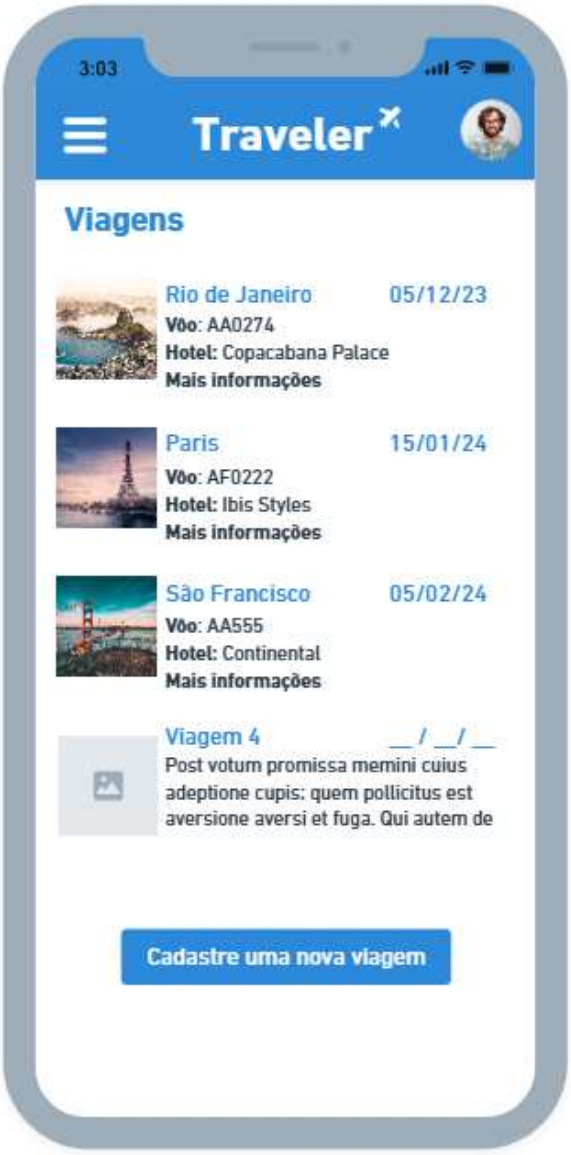
Tela de Menu com as opções de navegar por todos os módulos do aplicativo (resumo de viagens, resumo de transportes, resumo de hospedagens, resumo de passeios turísticos, resumo financeiro, sobre, configurações e sair).

Figura 7: Tela de perfil

Fonte: elaborado pelos autores (2024)

Tela de perfil com as informações pessoais do usuário (nome, sexo, data de nascimento e e-mail), além da foto e a possibilidade de editar a imagem.

Figura 8: Tela de resumo de viagens



Fonte: elaborado pelos autores (2024)

Tela de resumo de viagens com o histórico das últimas viagens do usuário com a data e as últimas informações de transporte e hospedagem respectivamente, caso a viagem já tenha ocorrido ou dos próximos transportes e hospedagens caso a viagem esteja ocorrendo ou seja no futuro, a imagem selecionada pelo usuário para a viagem e a possibilidade de efetuar o cadastro de uma nova viagem.

Figura 9: Tela de informações da viagem

Fonte: elaborado pelos autores (2024)

Tela de resumo da viagem com a imagem selecionada pelo usuário, o título, a data, as informações do próximo ou do último transporte, hospedagem e passeio e um total dos gastos até a data atual, além da meta de gastos definida pelo usuário e a possibilidade de cadastrar novos itens nessa mesma viagem.

Figura 10: Tela de informações do transporte

Fonte: elaborado pelos autores (2024)

Tela de transporte com as informações do transporte selecionado (a partir da tela de viagem sendo o próximo ou último transporte da viagem ou da tela de transportes, sendo o item selecionado pelo usuário), com as informações de destino, se tem ou não escala, valor pago, status do transporte e detalhes, além da opção de edição do item.

Figura 11: Tela de informações de hospedagem

Fonte: elaborado pelos autores (2024)

Tela de hospedagem com as informações da hospedagem selecionada (a partir da tela de viagem sendo a próxima ou última hospedagem da viagem ou da tela de hospedagens, sendo o item selecionado pelo usuário), com as informações de endereço, data e horário de check-in e check-out, valor pago e observações, além da opção de edição do item.

Figura 12: Tela de informações de passeio turístico

Fonte: elaborado pelos autores (2024)

Tela de passeio turístico com as informações do passeio turístico selecionado (a partir da tela de viagem sendo o próximo ou último passeio turístico da viagem ou da tela de passeios turísticos, sendo o item selecionado pelo usuário), com as informações de ponto de encontro, data, valor e observações, além da opção de edição do item.

Figura 13: Tela de resumo de despesas

Fonte: elaborado pelos autores (2024)

Tela de despesas com o resumo do total de gastos e a meta definida na parte superior e as últimas despesas cadastradas na parte inferior, com os dados de título, valor, data, horário, categoria e forma de pagamento, além das opções de editar e excluir e a possibilidade de adicionar uma nova despesa dentro da viagem por onde a tela foi acessada.

Figura 14: Tela de cadastro de despesa

Fonte: elaborado pelos autores (2024)

Tela de cadastro de despesa com as opções de título, valor (com máscara de moeda padrão real), forma de pagamento (lista suspensa com as opções de dinheiro, crédito, débito e pix), categoria (lista suspensa com as categorias alimentação, transporte, passeio e hospedagem), data (com máscara de data e hora) e observações (caixa de digitação de texto sem formato), além do botão de cadastrar, que salva a nova despesa e voltar, que cancela o que o usuário está editando e retorna para a tela anterior.

Figura 15: Tela de cadastro de transporte

Fonte: elaborado pelos autores (2024)

Tela de cadastro de transporte com as opções de tipo (lista suspensa com as opções avião, ônibus e navio), origem, destino, valor (com máscara de moeda padrão real), forma de pagamento (lista suspensa com as opções de dinheiro, crédito, débito e pix), data (com máscara de data e hora), data da escala (com máscara de data e hora), check box para cadastrar a despesa automaticamente, observações e opção de definir um alarme, além do botão cadastrar, que salva o novo transporte e voltar, que cancela o que o usuário está editando e retorna para a tela anterior.

Figura 16: Tela de cadastro de hospedagem

Fonte: elaborado pelos autores (2024)

Tela de cadastro de hospedagem com as opções de tipo (lista suspensa com as opções hotel, casa e hostel), endereço, valor (com máscara de moeda padrão real), forma de pagamento (lista suspensa com as opções de dinheiro, crédito, débito e pix), check-in (com máscara de data e hora), check-out (com máscara de data e hora), check box para cadastrar a despesa automaticamente, observações e opção de definir um alarme, além do botão cadastrar, que salva a nova hospedagem e voltar, que cancela o que o usuário está editando e retorna para a tela anterior.

Figura 17: Tela de cadastro de passeio turístico

Fonte: elaborado pelos autores (2024)

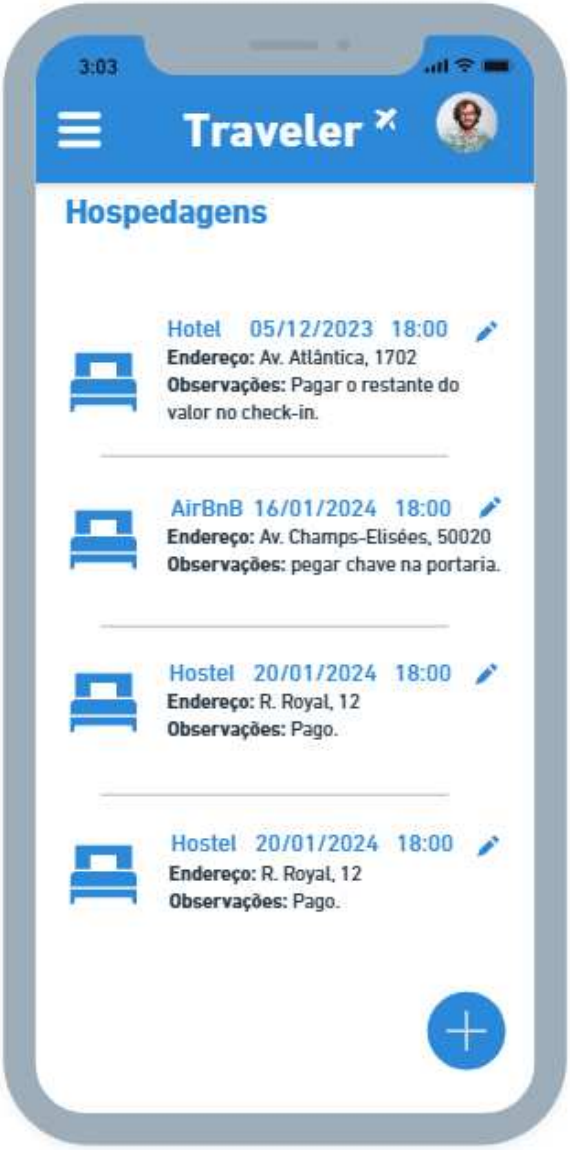
Tela de cadastro de passeio turístico com as opções de nome, endereço, data (com máscara de data e hora), valor (com máscara de moeda padrão real), forma de pagamento (lista suspensa com as opções de dinheiro, crédito, débito e pix), check box para cadastrar a despesa automaticamente, observações e opção de definir um alarme, além do botão cadastrar, que salva o novo passeio turístico e voltar, que cancela o que o usuário está editando e retorna para a tela anterior.

Figura 18: Tela de histórico de transportes

Fonte: elaborado pelos autores (2024)

Tela de transportes com o histórico dos últimos transportes cadastrados pelo usuário, apresentando no resumo a data, o horário, a origem, o destino, o status e a opção de editar, além da possibilidade de cadastrar um novo transporte.

Figura 19: Tela de histórico de hospedagens



Fonte: elaborado pelos autores (2024)

Tela de hospedagens com o histórico das últimas hospedagens cadastradas pelo usuário, apresentando no resumo a categoria, a data, o horário, o endereço e um resumo das observações, além da possibilidade de cadastrar uma nova hospedagem.

Figura 20: Tela de histórico de passeios turísticos

Fonte: elaborado pelos autores (2024)

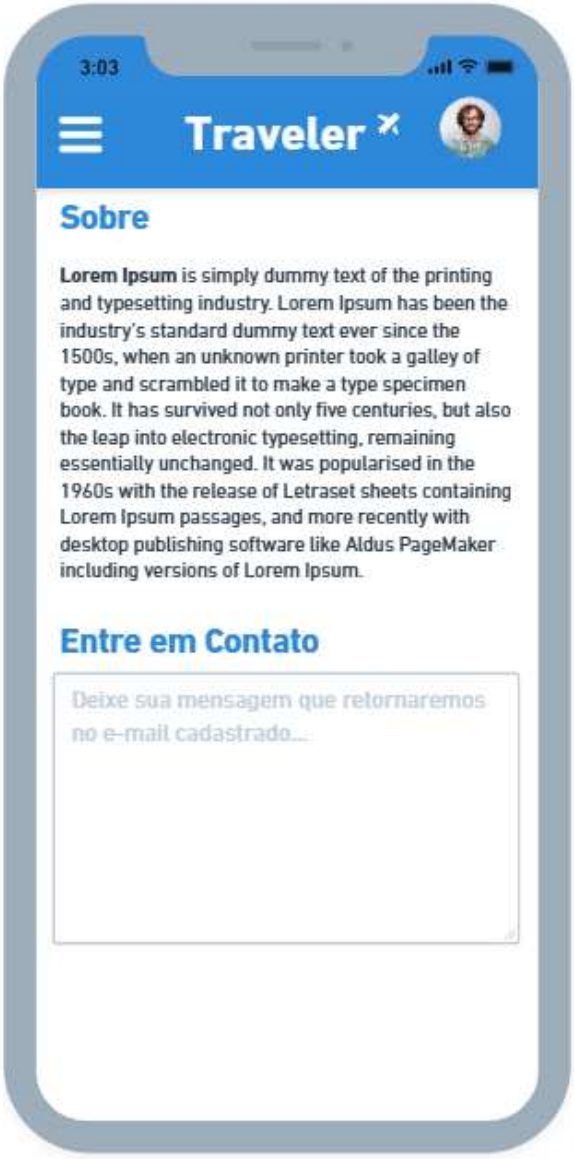
Tela de passeios turísticos com o histórico dos últimos passeios cadastrados pelo usuário, apresentando no resumo o nome, a data, o horário, e um resumo das observações, além da possibilidade de cadastrar um novo passeio.

Figura 21: Tela de histórico financeiro

Fonte: elaborado pelos autores (2024)

Tela do histórico financeiro com um resumo do histórico financeiro das últimas viagens cadastradas pelo usuário, apresentando no resumo a imagem definida para a viagem, o título da viagem, a data, o total de gastos e a meta de gastos.

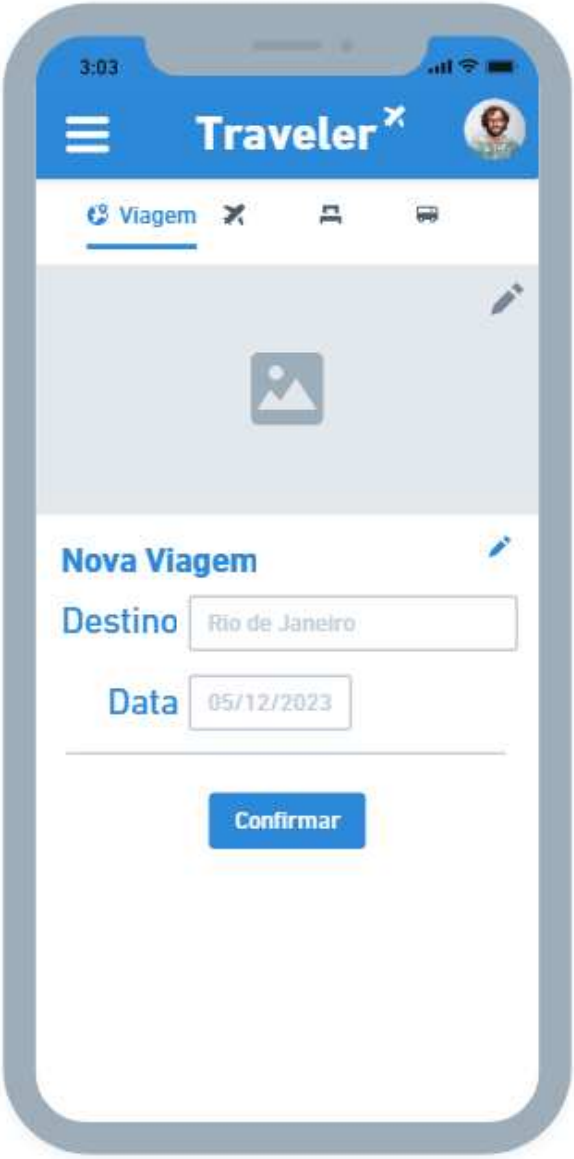
Figura 22: Tela sobre



Fonte: elaborado pelos autores (2024)

Tela sobre apresentando informações sobre os desenvolvedores do aplicativo e a possibilidade de encontrar em contato com o suporte.

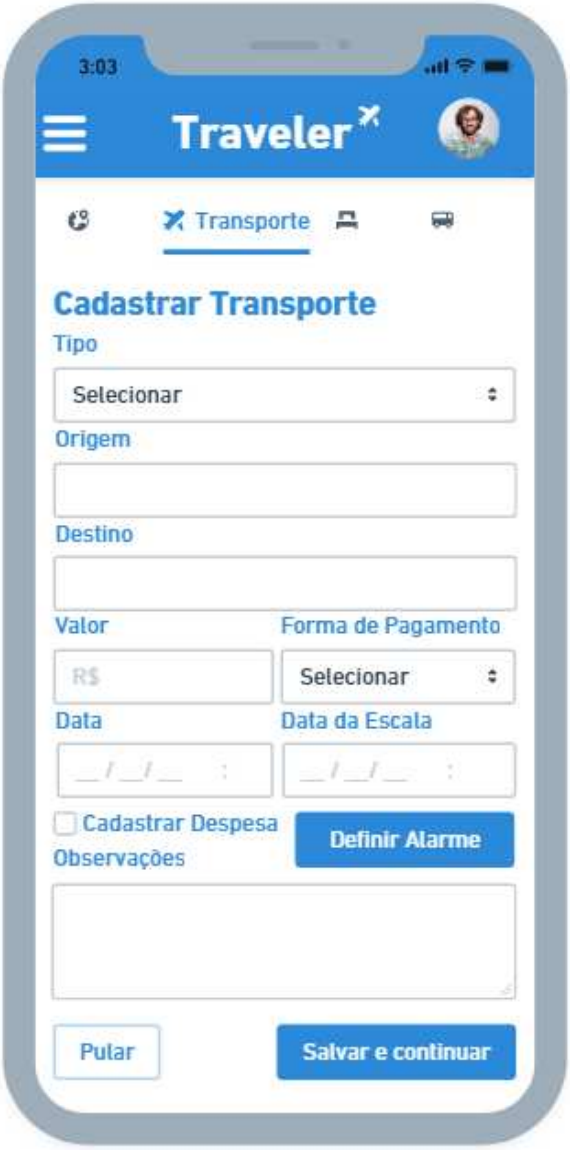
Figura 23: Tela de cadastro de viagem



Fonte: elaborado pelos autores (2024)

Tela de cadastro de viagem com a opção de selecionar uma imagem, definir o destino e a data.

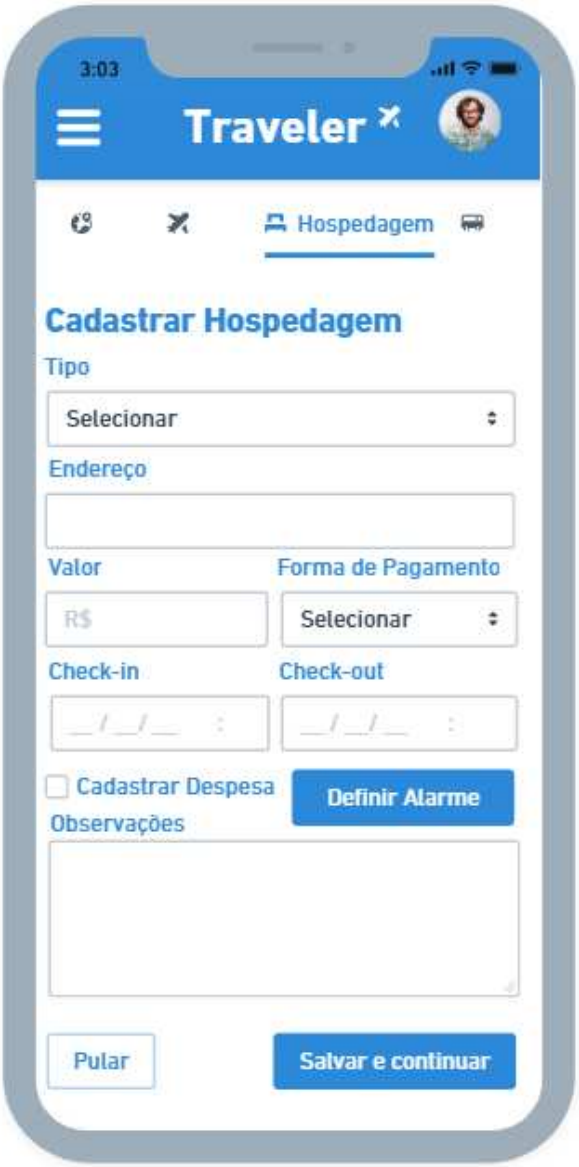
Figura 24: Tela de cadastro de transporte a partir do cadastro da viagem



Fonte: elaborado pelos autores (2024)

Tela de cadastro de transporte com as mesmas opções da tela de cadastro de transporte a partir da tela de transportes, porém com um carrossel na parte superior indicando que a próxima tela será o cadastro da hospedagem e apresentando a opção de pular.

Figura 25: Tela de cadastro de hospedagem a partir do cadastro da viagem



Fonte: elaborado pelos autores (2024)

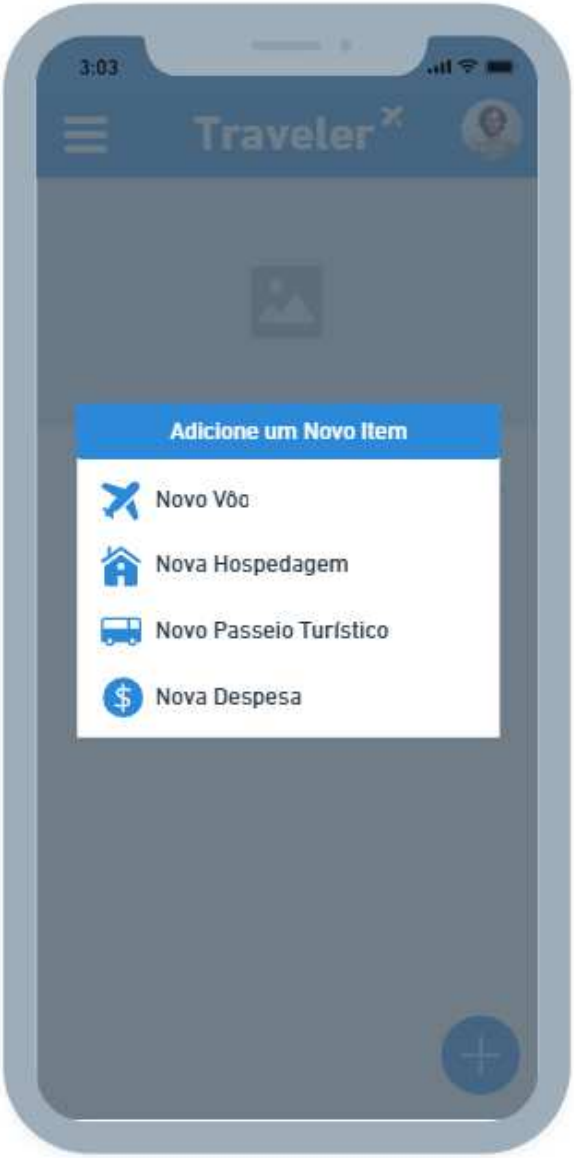
Tela de cadastro de hospedagem com as mesmas opções da tela de cadastro de hospedagem a partir da tela de hospedagens, porém com um carrossel na parte superior indicando que a próxima tela será o cadastro do passeio turístico e apresentando a opção de pular.

Figura 26: Tela de cadastro de passeio turístico a partir do cadastro da viagem

Fonte: elaborado pelos autores (2024)

Tela de cadastro de passeio turístico com as mesmas opções da tela de cadastro de passeio a partir da tela de passeios, porém com um carrossel na parte superior indicando que essa é a última tela e apresentando a opção de pular e finalizar.

Figura 27: Tela de adição de item



Fonte: elaborado pelos autores (2024)

Pop-up de cadastro de novo item a partir da tela da viagem, com as opções de transporte, hospedagem, passeio turístico e despesa.

4 DESENVOLVIMENTO DO APLICATIVO

4.1 LEVANTAMENTO DE REQUISITOS

4.1.1 Requisitos funcionais

Tabela 2: Requisitos funcionais

Nº	Requisitos	Notas
RF01	O aplicativo deve oferecer uma tela inicial com as opções de cadastro de novo usuário ou login utilizando aplicativos de terceiros, ao realizar o primeiro acesso.	Caso o usuário realize o cadastro, serão solicitadas informações como: Nome; e-mail; data de nascimento; telefone e sexo. Ao realizar o login utilizando aplicativos de terceiros (Google, Facebook etc...), o aplicativo fará um pré-cadastro do usuário.
RF02	O aplicativo deve oferecer a possibilidade de login com o endereço de e-mail e senhas cadastrados na tela inicial.	A senha digitada deverá ser ocultada, substituindo os caracteres digitados por asteriscos.
RF03	O aplicativo deve oferecer a funcionalidade de recuperar a senha, na tela inicial.	Uma mensagem será enviada ao e-mail ou telefone cadastrado com um número de verificação.
RF04	O aplicativo deve oferecer a opção de salvar o e-mail de acesso do cliente na tela inicial.	Caso o usuário realize o logout, ao reabrir o aplicativo ele deverá apresentar o e-mail do cliente.
RF05	O aplicativo deve oferecer a opção de visualizar a senha digitada.	Quando o cliente pressionar a opção de visualização, a senha deve aparecer por 5 segundo e voltar ao formato oculto.
RF06	O aplicativo deve apresentar um resumo com todas as viagens cadastradas pelo usuário na tela principal.	As viagens devem ser ordenadas pela data mais recente e com informações resumidas de data, endereço e observações.

RF07	O aplicativo deverá exibir uma tela inicial com as últimas viagens pendentes, além de permitir o cadastro de uma nova viagem a partir da tela principal.	As viagens deverão ser apresentadas com uma imagem definida pelo usuário e um resumo da próxima hospedagem, transporte e passeio.
RF08	O aplicativo deve apresentar uma barra de menu, que poderá ser acessada a partir de qualquer tela.	O menu deve ser apresentado de maneira representada na parte superior da tela, de forma expansível, possibilitando a navegação pelas funções: Viagens; Transportes; Hospedagens; Passeios; Financeiro e Ajuda.
RF09	O aplicativo deve apresentar um ícone para acessar as configurações de perfil do usuário.	O ícone deverá ficar na parte superior da tela, ao lado oposto do menu e deverá apresentar um ícone de usuário ou a foto carregada.
RF10	O aplicativo deve apresentar uma tela de resumo do perfil do usuário.	Ao clicar no ícone de perfil, o usuário deverá ser direcionado para uma tela resumida com foto, nome e possibilidade de navegar para a tela de edição.
RF11	O aplicativo deverá permitir a edição das informações do usuário.	A tela deverá possuir os campos: Nome; Email; Data de Nascimento; Telefone; Sexo e Nacionalidade.
RF12	O aplicativo deve possuir uma tela com o resumo de cada item cadastrado dentro de uma viagem.	A tela de resumo poderá ser acessada a partir da tela principal ou da tela de Viagens, a partir do menu, e deverá exibir o próximo compromisso de cada categoria.

RF13	O aplicativo deverá permitir que o usuário visualize os detalhes de qualquer item cadastrado na viagem a partir da tela de resumo.	Ao clicar em cima do item resumido apresentado na tela da viagem, o usuário deverá ser direcionado para uma tela que apresentará as informações completas daquele compromisso, permitindo sua edição.
RF14	O aplicativo deverá permitir que o usuário cadastre um novo compromisso a partir da tela de detalhes do item acessado (transporte, hospedagem ou passeio).	O compromisso deverá ser cadastrado dentro da viagem do item original.
RF15	O aplicativo deverá apresentar um resumo do custo da viagem.	O resumo deverá ser apresentado na tela de detalhes da viagem.
RF16	O aplicativo deverá permitir cadastrar uma meta de gastos para cada viagem	A meta de gastos deverá ser apresentada junto ao resumo de custo, com o intuito de comparação.
RF17	O aplicativo deverá permitir cadastrar as despesas da viagem de acordo com categorias.	O sistema deverá apresentar categorias pré-definidas e permitir que o usuário cadastre suas próprias categorias.
RF18	O aplicativo deverá permitir que o usuário cadastre alarmes para cada compromisso (transporte, hospedagem ou passeio).	Os alarmes poderão ser sonoros ou silenciosos, via notificação.
RF19	O aplicativo deverá apresentar uma tela de resumo para cada tipo de item (transporte, hospedagem ou passeio).	Os itens devem ser ordenados por data e serão acessados quando o usuário clicar sobre o nome do item ao expandir o menu.
RF20	O aplicativo deverá possuir uma tela com as informações básicas do desenvolvedor e uma caixa de mensagens para que o usuário entre em contato.	As mensagens disparadas pelo usuário deverão ser encaminhadas na caixa de e-mail da aplicação.

Fonte: elaborado pelo autor (2024)

4.1.2 Requisitos não funcionais

Tabela 3: Requisitos não funcionais

Nº	Requisitos	Notas
RNF01	Android versão 5 ou superior.	Versão mínima suportada pela Play Store.
RNF02	Mínimo de 1Gb de Memória RAM para versões de Android inferiores ao Android 8, mínimo de 2Gb de Memória RAM para versões superiores ao Android 8.	Recomendado 4Gb de Memória RAM.
RNF03	Mínimo de 512Mb de armazenamento interno disponível.	Recomendado 1Gb de memória disponível.
RNF04	O aplicativo deverá ser desenvolvido em React Native.	
RNF05	O aplicativo deverá possuir integração com um banco de dados PostgreSQL.	

Fonte: elaborado pelos autores (2024)

4.1.3 Regras de negócio

Tabela 4: Regras de negócio

Nº	**Nome**	**Descrição**
RN01	Cadastro de usuário	O aplicativo apresentará a tela de cadastro de usuário com os campos: nome; e-mail; senha; data de nascimento; telefone; sexo e nacionalidade (com as opções brasileiro e estrangeiro).
RN02	Cadastro de viagem	O aplicativo apresentará a opção de cadastrar uma nova viagem com os campos: destino e data.
RN03	Cadastro de transporte	Dentro da tela de resumo de cada viagem criada o aplicativo apresentará a opção de cadastrar um novo transporte, com os campos: tipo; origem;

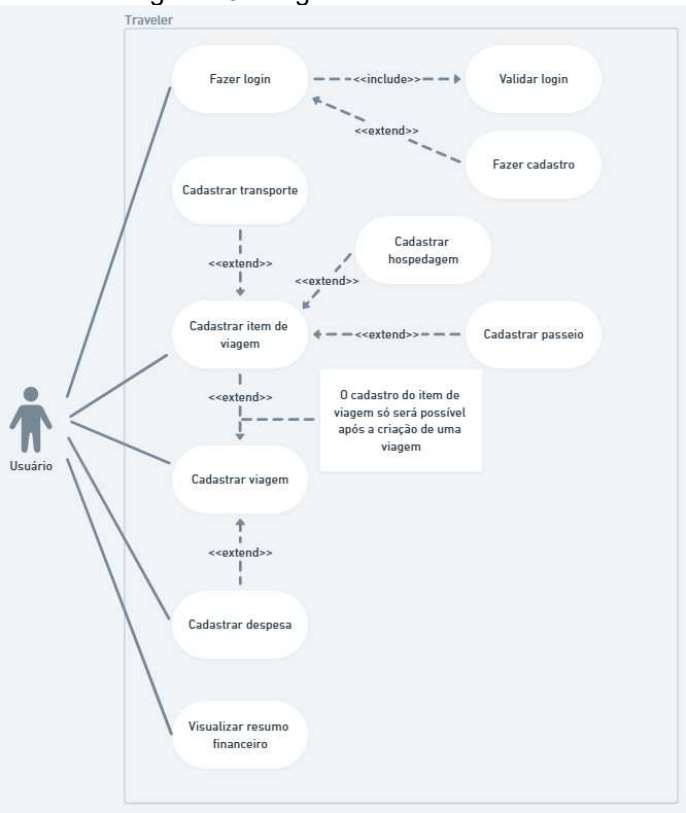
		destino; valor; data e data da escala, além da possibilidade de definição de alarme e cadastro automático nas despesas.
RN04	Cadastro de hospedagem	Dentro da tela de resumo de cada viagem criada o aplicativo apresentará a opção de cadastrar uma nova hospedagem, com os campos: tipo; endereço; valor; check-in e check-out, além da possibilidade de definição de alarme e cadastro automático nas despesas.
RN05	Cadastro de passeio	Dentro da tela de resumo de cada viagem criada o aplicativo apresentará a opção de cadastrar um novo passeio turístico, com os campos: nome; endereço; data e valor, além da possibilidade de definição de alarme e cadastro automático nas despesas.
RN06	Cadastro de despesa	Dentro da tela de resumo de cada viagem criada o aplicativo apresentará a opção de cadastrar uma nova despesa, com os campos: título; valor; forma de pagamento; data; categoria e observações.
RN07	Edição de viagem	O usuário terá a opção de editar as informações da viagem sem restrições.
RN08	Edição de transporte	O usuário terá a opção de editar as informações do transporte sem restrições.
RN09	Edição de hospedagem	O usuário terá a opção de editar as informações da hospedagem sem restrições.
RN10	Edição de passeio	O usuário terá a opção de editar as informações do passeio sem restrições.
RN11	Edição de despesa	O usuário terá a opção de editar as informações da despesa sem restrições.
RN12	Visualização de resumo financeiro	O usuário terá disponível um resumo financeiro de suas últimas viagens.

Fonte: elaborado pelos autores (2024)

4.1.4 Diagramas

4.1.4.1 Diagrama de Caso de Uso

Figura 28: Diagrama de caso de uso



Fonte: elaborado pelos autores (2025)

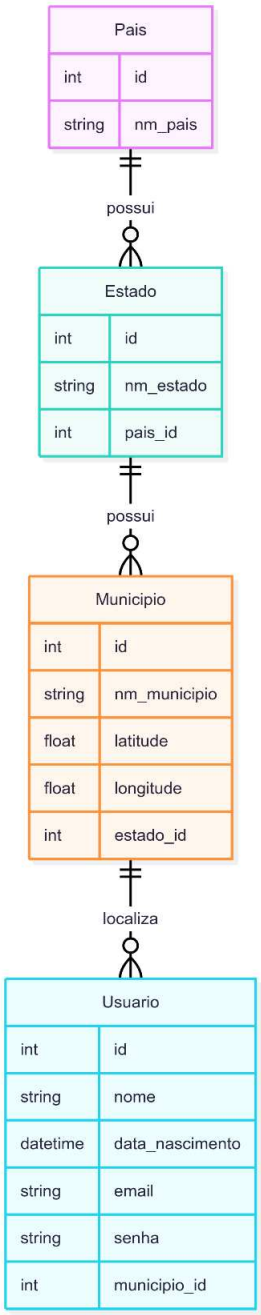
4.2 PROJETO DE BANCO DE DADOS

Será utilizado o banco de dados relacional PostgreSQL conectado ao backend via utilizando o ORM Prisma.

4.2.1 Modelo de Entidade Relacional (MER)

4.2.1.1 Módulo usuário

Figura 29: Diagrama de entidade e relacionamento - módulo de usuário



Fonte: elaborado pelos autores (2024)

Tabela 5: Dicionário de dados País

Atributo	**Tipo**	**Restrições**	**Descrição**
id	Int	PK, NOT NULL	Identificador único do país
nm_pais	String	NOT NULL	Nome do país
nm_pais_ascii	String	NULLABLE	Nome sem acentuação

Fonte: elaborado pelos autores (2024)

Tabela 6: Dicionário de dados Estado

Atributo	Tipo	Restrições	Descrição
id	Int	PK, NOT NULL	Identificador único do estado
nm_estado	String	NOT NULL	Nome do estado
nm_estado_ascii	String	NULLABLE	Nome sem acentuação
pais_id	Int	FK → pais.id, NOT NULL	País ao qual o estado pertence

Fonte: elaborado pelos autores (2024)

Tabela 7: Dicionário de dados Município

Atributo	Tipo	Restrições	Descrição
id	Int	PK, NOT NULL	Identificador único
nm_municipio	String	NOT NULL	Nome do município
nm_municipio_ascii	String	NULLABLE	Nome sem acentuação
latitude	Float	NOT NULL	Latitude geográfica
longitude	Float	NOT NULL	Longitude geográfica
estado_id	Int	FK → estado.id, NOT NULL	Estado ao qual pertence

Fonte: elaborado pelos autores (2024)

Tabela 8: Dicionário de dados Usuários

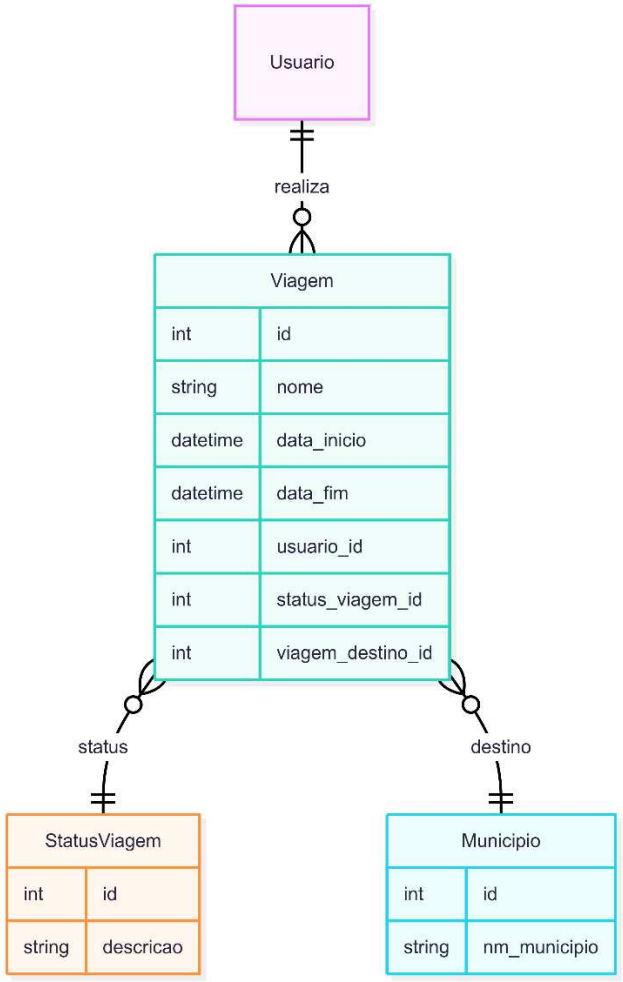
Atributo	Tipo	Restrições	Descrição
id	Int	PK, NOT NULL	Identificador único do usuário
nome	String	NOT NULL	Nome completo
data_nascimento	DateTime	NOT NULL	Data de nascimento
email	String	UNIQUE, NOT NULL	E-mail do usuário
senha	String	NOT NULL	Senha (criptografada)
municipio_id	Int	FK → municipio.id, NOT NULL	Município de origem
created_at	DateTime	NOT NULL, DEFAULT: now()	Data de criação
updated_at	DateTime	NOT NULL, Atualizado	Última atualização

		automaticamente	

Fonte: elaborado pelos autores (2024)

4.2.1.2 Módulo viagem

Figura 30: Diagrama de entidade e relacionamento - módulo de viagem



Fonte: elaborado pelos autores (2024)

Tabela 9: Dicionário de dados Viagem

Atributo	**Tipo**	**Restrições**	**Descrição**
id	Int	PK, NOT NULL	Identificador único
nome	String	NOT NULL	Nome da viagem
data_inicio	DateTime	NOT NULL	Início da viagem
data_fim	DateTime	NOT NULL	Fim da viagem
usuario_id	Int	FK → usuarios.id, NOT NULL	Criador da viagem
status_viagem_id	Int	FK → status_viagem.id, NOT NULL	Status atual
viagem_destino_id	Int	FK → municipio.id, NOT NULL	Município de

			destino
created_at	DateTime	NOT NULL, DEFAULT: now()	Criação
updated_at	DateTime	NOT NULL, Atualizado automaticamente	Última atualização

Fonte: elaborado pelos autores (2024)

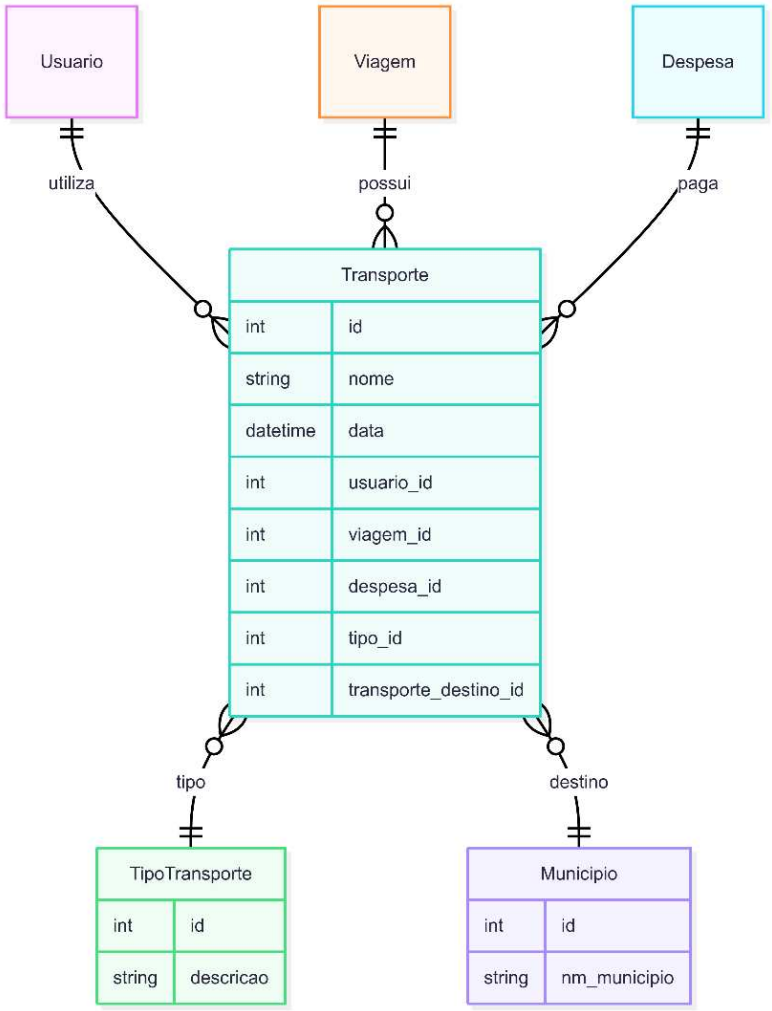
Tabela 10: Dicionário de dados StatusViagem

Atributo	**Tipo**	**Restrições**	**Descrição**
id	Int	PK, NOT NULL	Identificador
descricao	String	UNIQUE, NOT NULL	Nome do status (Ex: Planejada, Concluída)

Fonte: elaborado pelos autores (2024)

4.2.1.3 Módulo Transporte

Figura 31: Diagrama de entidade e relacionamento - módulo de transporte



Fonte: elaborado pelos autores (2024)

Tabela 11: Dicionário de dados Transporte

Atributo	**Tipo**	**Restrições**	**Descrição**
id	Int	PK, NOT NULL	Identificador
nome	String	NOT NULL	Nome do transporte
tipo_id	Int	FK → tipo_transporte.id, NOT NULL	Tipo
data	DateTime	NOT NULL	Data do deslocamento
despesa_id	Int	FK → despesas.id, NOT NULL	Despesa associada
viagem_id	Int	FK → viagens.id, NOT NULL	Viagem relacionada
transporte_destino_id	Int	FK → municipio.id, NOT NULL	Município de destino
usuario_id	Int	FK → usuarios.id, NOT NULL	Usuário que realizou
documento_anexo	String	NULLABLE	Documento (ex: recibo)
created_at	DateTime	NOT NULL, DEFAULT:	Criação

		now()	
updated_at	DateTime	NOT NULL, Atualizado automaticamente	Última atualização

Fonte: elaborado pelos autores (2024)

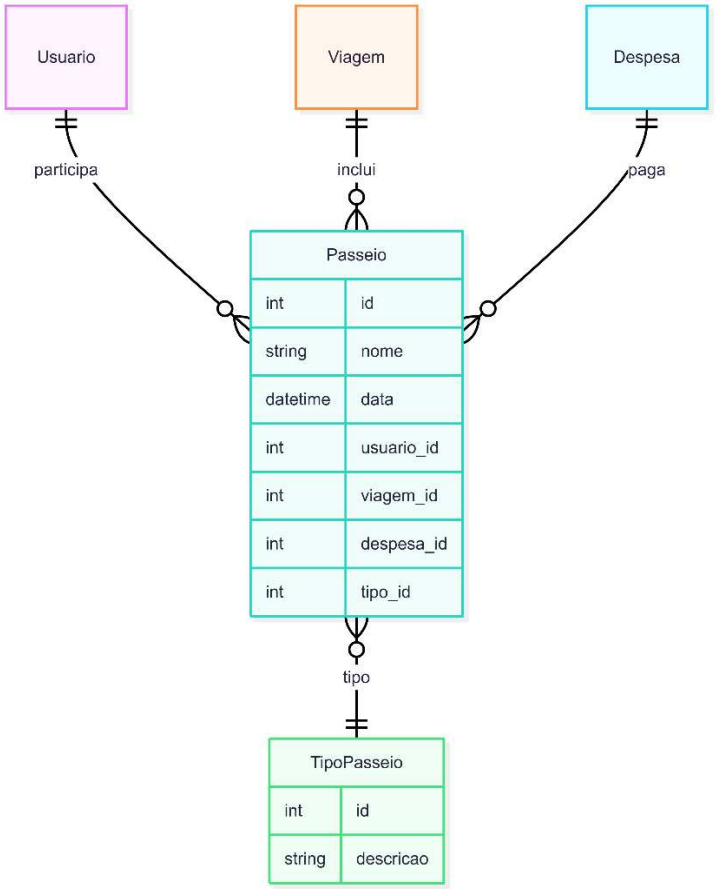
Tabela 12: Dicionário de dados TipoTransporte

Atributo	**Tipo**	**Restrições**	**Descrição**
id	Int	PK, NOT NULL	Identificador
descricao	String	UNIQUE, NOT NULL	Tipo do transporte (avião, ônibus etc.)

Fonte: elaborado pelos autores (2024)

4.2.1.4 Módulo passeio

Figura 32: Diagrama de entidade e relacionamento - módulo de passeio



Fonte: elaborado pelos autores (2024)

Tabela 13: Dicionário de dados Passeio

Atributo	**Tipo**	**Restrições**	**Descrição**
id	Int	PK, NOT NULL	Identificador
nome	String	NOT NULL	Nome do passeio

tipo_id	Int	FK → tipo_passeio.id, NOT NULL	Tipo do passeio
data	DateTime	NOT NULL	Data de realização
despesa_id	Int	FK → despesas.id, NOT NULL	Despesa associada
viagem_id	Int	FK → viagens.id, NOT NULL	Viagem associada
usuario_id	Int	FK → usuarios.id, NOT NULL	Usuário responsável
documento_anexo	String	NULLABLE	Arquivo (foto, recibo etc.)
created_at	DateTime	NOT NULL, DEFAULT: now()	Criação
updated_at	DateTime	NOT NULL, Atualizado automaticamente	Última modificação

Fonte: elaborado pelos autores (2024)

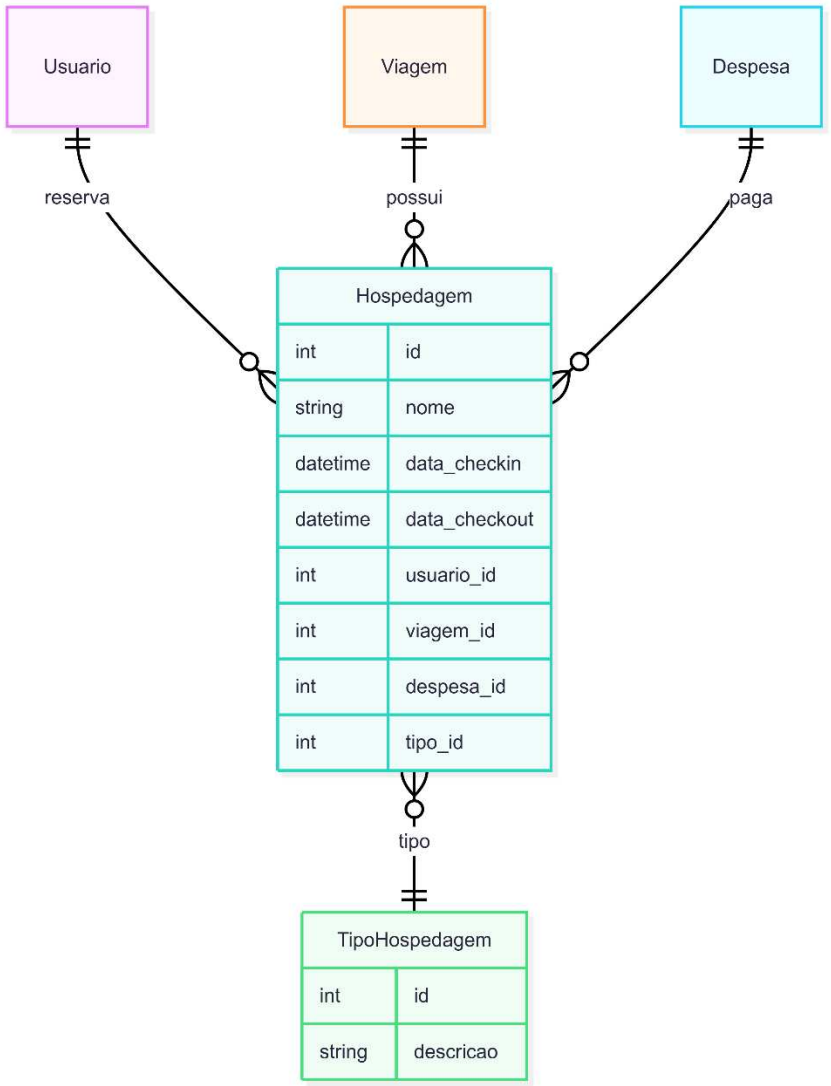
Tabela 14: Dicionário de dados TipoPasseio

Atributo	**Tipo**	**Restrições**	**Descrição**
id	Int	PK, NOT NULL	Identificador
descricao	String	UNIQUE, NOT NULL	Descrição (turístico, cultural…)

Fonte: elaborado pelos autores (2024)

4.2.1.5 Módulo hospedagem

Figura 33: Diagrama de entidade e relacionamento - módulo de hospedagem



Fonte: elaborado pelos autores (2024)

Tabela 15: Dicionário de dados Hospedagem

Atributo	**Tipo**	**Restrições**	**Descrição**
id	Int	PK, NOT NULL	Identificador
nome	String	NOT NULL	Nome do hotel/pousada
tipo_id	Int	FK → tipo_hospedagem.id, NOT NULL	Tipo da hospedagem
data_checkin	DateTime	NOT NULL	Data de entrada
data_checkout	DateTime	NOT NULL	Data de saída
despesa_id	Int	FK → despesas.id, NOT NULL	Despesa associada
viagem_id	Int	FK → viagens.id, NOT NULL	Viagem associada
usuario_id	Int	FK → usuarios.id, NOT NULL	Criador do registro

endereco	String	NULLABLE	Endereço do local
documento_anexo	String	NULLABLE	Comprovante (ex: PDF)
created_at	DateTime	NOT NULL, DEFAULT: now()	Data de criação
updated_at	DateTime	NOT NULL, Atualizado automaticamente	Última atualização

Fonte: elaborado pelos autores (2024)

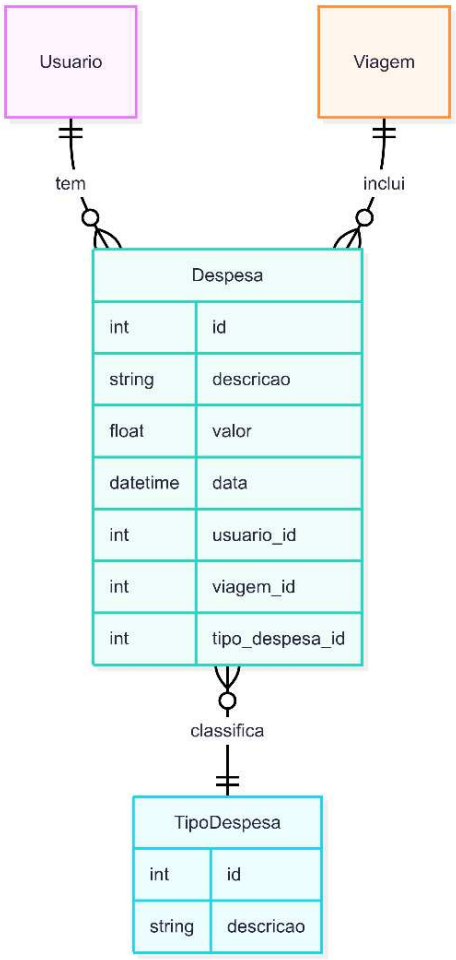
Tabela 16: Dicionário de dados Hospedagem

Atributo	**Tipo**	**Restrições**	**Descrição**
id	Int	PK, NOT NULL	Identificador
descricao	String	UNIQUE, NOT NULL	Descrição (hotel, hostel...)

Fonte: elaborado pelos autores (2024)

4.2.1.6 Módulo despesa

Figura 34: Diagrama de entidade e relacionamento - módulo de despesa



Fonte: elaborado pelos autores (2024)

Tabela 17: Dicionário de dados Despesas

Atributo	**Tipo**	**Restrições**	**Descrição**
id	Int	PK, NOT NULL	Identificador
descricao	String	NOT NULL	Descrição textual
valor	Float	NOT NULL	Valor gasto
data	DateTime	NOT NULL	Data do gasto
viagem_id	Int	FK → viagens.id, NOT NULL	Viagem associada
tipo_despesa_id	Int	FK → tipo_despesa.id, NOT NULL	Tipo da despesa
usuario_id	Int	FK → usuarios.id, NOT NULL	Usuário responsável
created_at	DateTime	NOT NULL, DEFAULT: now()	Data de criação
updated_at	DateTime	NOT NULL, Atualizado automaticamente	Última modificação

Fonte: elaborado pelos autores (2024)

Tabela 18: Dicionário de dados TipoDespesa

Atributo	Tipo	Restrições	Descrição
id	Int	PK, NOT NULL	Identificador
descricao	String	UNIQUE, NOT NULL	Nome do tipo (Ex: Alimentação)

Fonte: elaborado pelos autores (2024)

4.3 PROJETO TÉCNICO

4.3.1 Servidor

O back-end do projeto será desenvolvido utilizando o framework Node JS, criado em 2009 e definido como “um ambiente de execução do código JavaScript do lado servidor (server side)” (Bessa, 2023), permitindo o desenvolvimento de aplicações JavaScript independentes de um navegador para serem executadas, como API’s Rest.

Essas API’s se comunicarão com um banco de dados relacional PostgreSQL, um projeto de código aberto iniciado em 1986 e posteriormente complementado conhecido por sua confiabilidade e flexibilidade. (IBM, [S.D.]).

Essa integração ocorrerá com a adoção do Prisma, um ORM com excelente integração com Javascript, Typescript e Postgres.

Os ORMs ou Object Relational Mapping, são uma forma de alinhar o código com o banco de dados, ajudando o desenvolvedor a correlacionar os objetos com as estruturas presentes na base, facilitando a manutenção da aplicação de uma forma mais global: a comunicação com o banco de dados e o restante da aplicação (Abba, 2022).

4.3.2 Sistema Mobile

O front-end do aplicativo será desenvolvido para Android utilizando o framework React Native, um framework lançado em 2015 pelo Facebook.

Seu principal propósito enquanto framework é trazer facilidade ao desenvolvedor, pois desenvolvendo app em React Native, há compatibilidade com ambos os sistemas operacionais móveis, sendo assim será possível portar a aplicação para o sistema IOS no futuro.

4.3.3 Ferramentas

A principal ferramenta de desenvolvimento será o Visual Studio Code, um poderoso editor de texto que permite produzir códigos em várias linguagens e conta com várias extensões para customizar e facilitar o trabalho do desenvolvedor.

Para os testes, serão utilizados o Android Studio, ambiente de desenvolvimento criado pelo Google que oferece emuladores Android, e o Expo Go, um ambiente isolado e controlado que permite testar códigos desenvolvidos em React Native diretamente em um smartphone físico.

4.4 ARQUITETURA

4.4.1 Metas e restrições de arquitetura

- O sistema Traveler deve ser disponível para dispositivos mobile com iOS e Android se utilizando da nuvem Aws para armazenamento de dados e serviços.
- A funcionalidade de visualização de informações de viagens deve estar disponível offline.
- Todos os requisitos de desempenho e carregamento devem ser observados na confecção da arquitetura.

4.4.2 Visão Geral

A visão lógica está organizada em cinco pacotes principais que representam as classes de maior importância, assim como os pacotes e subpacotes de serviço que possuam importância arquitetônica significante para o projeto e serão descritos a seguir.

4.4.2.1 Pacote Apresentação

Responsável pela interface do usuário e interação com o sistema. Este pacote está dividido em dois subpacotes:

- **Views**: Contém as classes que gerenciam as diferentes páginas de visualização da aplicação.
- **Forms**: Inclui os formulários usados para capturar dados do usuário.

4.4.2.2 Pacote Aplicação

Contém a lógica central da aplicação que coordena a interação entre a interface do usuário e os serviços. Contém um subpacote:

- **Controllers**: Contém as classes que gerenciam a lógica do sistema

4.4.2.3 Pacote Serviços

Oferece serviços que encapsulam a lógica de negócios do sistema.

- **ServiceUser**: Serviço responsável por operações relacionadas a usuários.
- **ServiceViagem** (subpacote): Conjunto de serviços específicos para operações relacionadas a viagens.

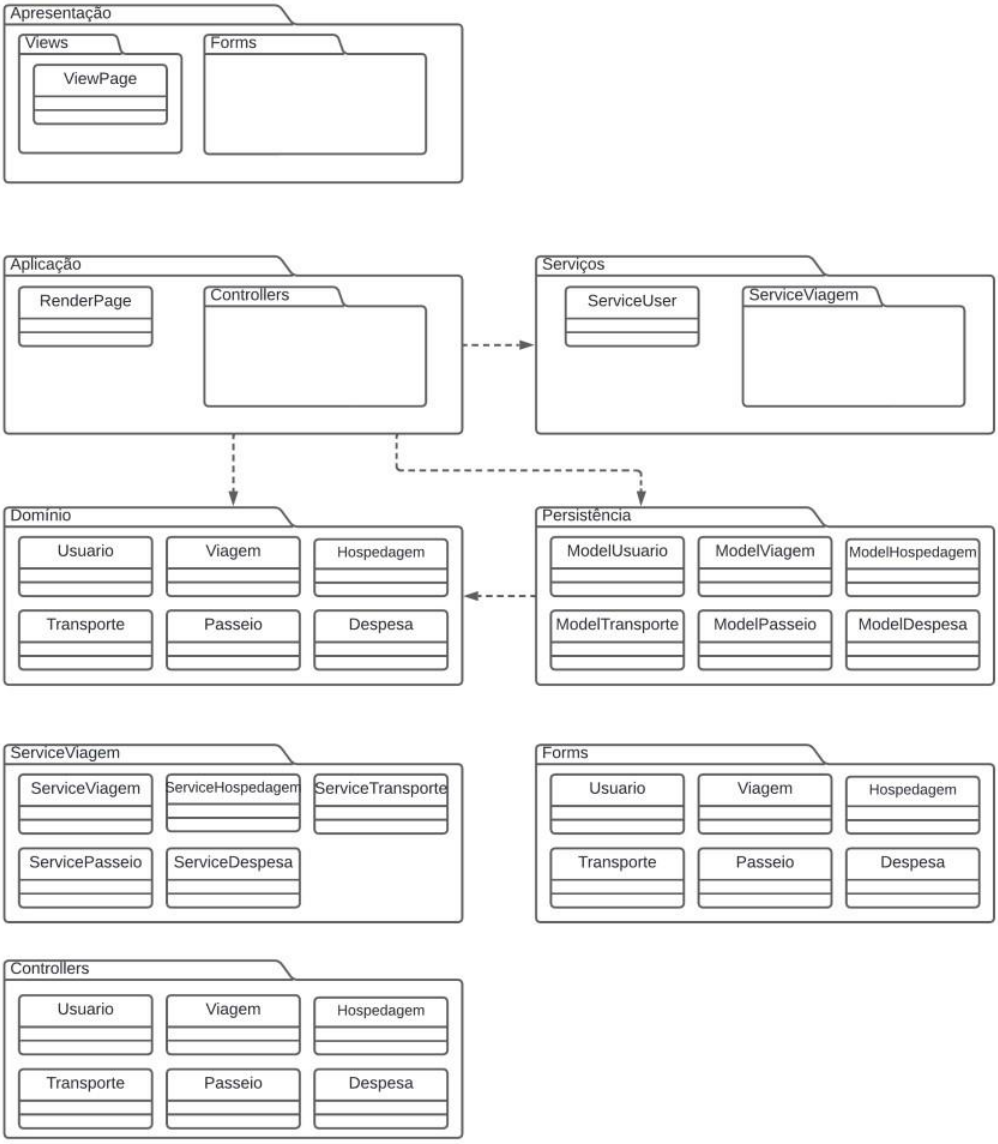
4.4.2.4 Pacote Domínio

Define as entidades principais do sistema, que são utilizadas por outros pacotes.

4.4.2.5 Pacote Persistência

Gerencia a persistência dos dados da aplicação, interagindo com o banco de dados.

Figura 35: Diagrama de pacote de persistência

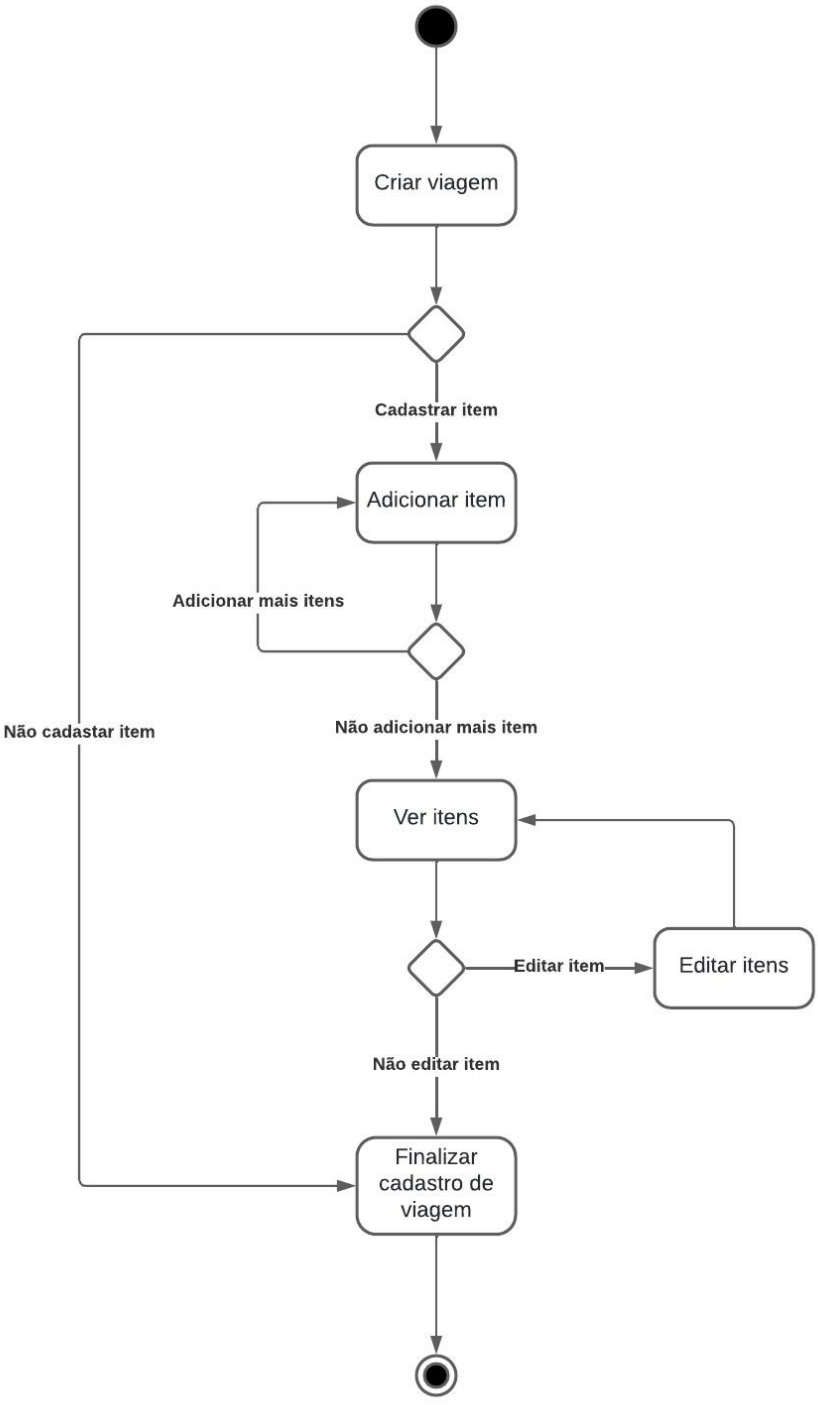


Fonte: elaborado pelos autores (2024)

4.4.3 Visão de processos

4.4.3.1 Diagrama de atividades relacionadas a processos em itens de viagem

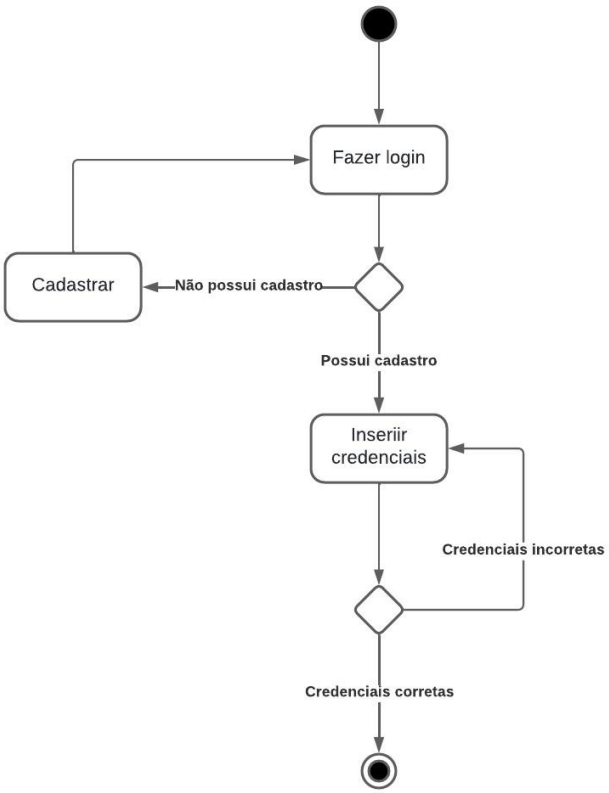
Figura 36: Diagrama de processos – viagens



Fonte: elaborado pelos autores (2024)

4.4.3.2 Diagrama de atividades relacionado a processos em usuário

Figura 37: Diagrama de processos – usuários



Fonte: elaborado pelos autores (2024)

4.4.4 Visão de implantação

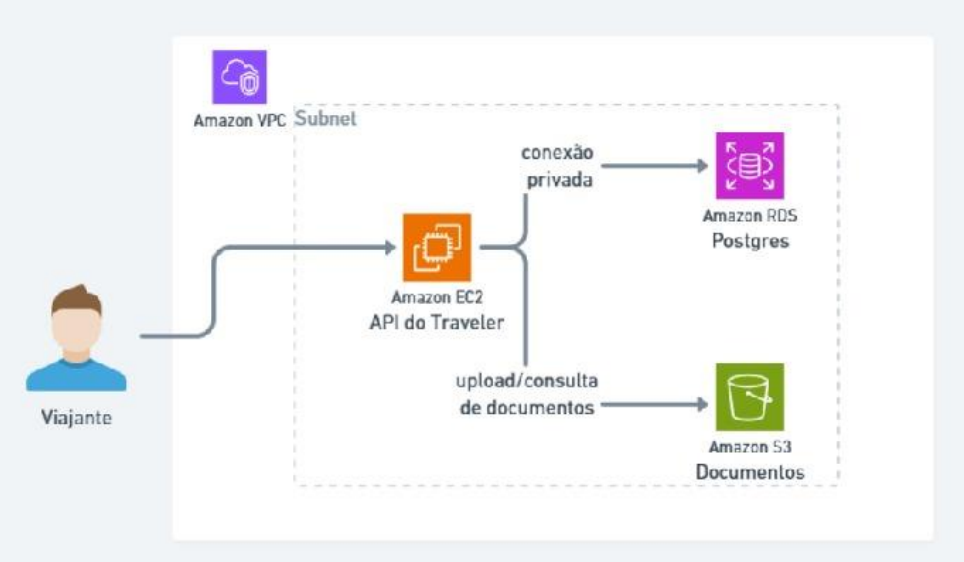
O nó cliente corresponde ao dispositivo móvel utilizado pelo usuário final, responsável por realizar requisições diretamente à camada de API hospedada em uma instância EC2 na AWS. Essa camada de API gerencia tanto a autenticação dos usuários quanto as operações relacionadas às informações das viagens, incluindo criação, consulta, atualização e exclusão de dados.

Toda a infraestrutura está contida dentro de uma Virtual Private Cloud (VPC) específica na AWS, garantindo isolamento e controle sobre o ambiente. Tanto a instância EC2, que hospeda a API, quanto o banco de dados relacional hospedado em uma instância RDS estão configurados dentro da mesma subnet privada, permitindo uma comunicação segura e direta entre ambos os componentes, sem exposição pública dos dados sensíveis.

O banco de dados relacional em RDS é utilizado para armazenar informações estruturadas referentes às viagens, usuários e autenticação, acessado exclusivamente pela instância EC2 de forma privada dentro da subnet.

Adicionalmente, um bucket no Amazon S3 é utilizado especificamente para armazenar documentos associados às viagens nos formatos PDF e JPG. Esses arquivos são acessados e gerenciados pela API na EC2, proporcionando um armazenamento escalável e seguro dos documentos.

Figura 38: Diagrama de arquitetura



Fonte: elaborado pelos autores (2024)

4.4.5 Visão de implementação

4.4.5.1 Visão Geral

A arquitetura do aplicativo Traveler é composta pelas seguintes camadas: camada de Aplicação do Usuário e Camada de Serviços e Banco de Dados que serão explicadas nos próximos itens.

4.4.5.2 Camada de Aplicação do Usuário:

Responsável pela interface do usuário e pela comunicação com o usuário final, sua principal função é fornecer uma interface amigável e intuitiva para o usuário interagir com o aplicativo. Será desenvolvida em React Native.

Operações relacionadas:

- Exibir informações e dados relevantes para o usuário.
- Permitir que o usuário realize ações relacionadas aos itens de viagem.
- Coletar dados do usuário e validá-los.
- Comunicar-se com a camada de serviços e banco de dados para obter e enviar dados.

4.4.5.3 Camada de Serviços

Responsável pela implementação das regras de negócio e processamento dos dados. Será desenvolvida utilizando as tecnologias: Node.js, Prisma e AWS Lambda. Operações relacionadas:

- Processar requisições do usuário.
- Transformar dados para que sejam armazenados no banco de dados.
- Coletar dados do usuário e validá-los.
- Comunicar-se com a camada de serviços e banco de dados para obter e enviar dados.

4.4.5.4 Camada de Banco de Dados

- Responsável pela manipulação e armazenamento de dados relacional. Serão utilizadas as tecnologias: AWS DynamoDB e MariaDB.

  Operações relacionadas:

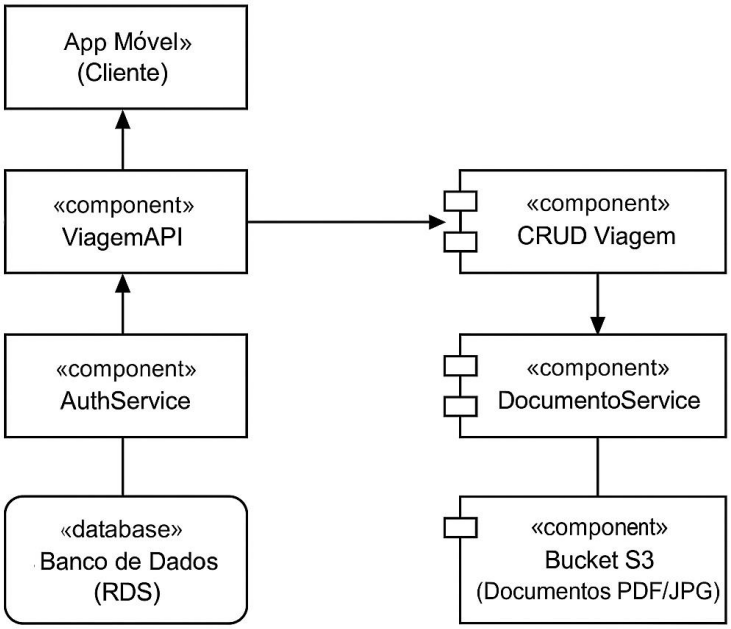
- Realizar a manipulação e armazenamento no banco de dados.

4.4.6 Camadas

O aplicativo Traveler possui as seguintes camadas: a da aplicação do usuário, a de serviços para processamento de serviços e AWS DynamoDB de banco de dados.

4.4.6.1 Diagrama de componentes

Figura 39: Diagrama de componentes



Fonte: elaborado pelos autores (2024)

4.4.7 Tamanho e desempenho

O dimensionamento do software no cliente deve ser o menor possível para permitir um manejo de espaço adequado para dispositivos mais antigos. A performance deve ser excelente levando em consideração o uso de nuvem, porém ele deve ser restrito ao uso gratuito das ferramentas da AWS.

4.4.8 Qualidade

A arquitetura de cloud contribui no quesito extensibilidade devido ao manejo escalonável, portável e seguro de que a Amazon AWS permite em seus serviços cloud-native. Utilizaremos os serviços Amazon RDS (Relational Database Service), que é um serviço que facilita a criação, operação e escalabilidade de bancos de dados relacionais na nuvem e o Amazon EC2 (Elastic Compute Cloud), fundamental para disponibilizar ambientes de execução sob demanda, permitindo a criação de instâncias escaláveis e controladas para testes, homologações e eventual produção.

A manutenibilidade do código será mantida devido ao sólido projeto inicial e

manutenção de uma boa documentação, quesitos essenciais ao se tratar de uma aplicação desta escala.

4.4.9 Segurança

Para garantir a segurança dos dados no aplicativo Traveler e cumprir com a Lei Geral de Proteção de Dados (LGPD), adotaremos várias medidas essenciais. Primeiramente, controlaremos rigorosamente o acesso aos dados usando subnets privadas, garantindo que apenas pessoas autorizadas possam acessá-los.

Além disso, protegeremos todas as informações transmitidas e armazenadas usando técnicas de criptografia como o bcrypt, um algoritmo adaptativo que permite ajustar o custo computacional do processo de hash, tornando o algoritmo resistente a ataques de força bruta, assegurando que os dados dos usuários estejam sempre seguros.

Para assegurar a conformidade com a LGPD, seremos transparentes sobre como usamos os dados dos nossos usuários. Pediremos consentimento explícito antes de coletar qualquer informação pessoal e ofereceremos a opção de visualizar, corrigir ou excluir esses dados conforme desejado.

4.5 IMPLANTAÇÃO

Inicialmente, o aplicativo será executado em um servidor local com o objetivo de realizar testes exploratórios e validar as funcionalidades implementadas. Posteriormente, toda a estrutura será migrada para os serviços da Amazon AWS conforme descrito no item 4.4, possibilitando sua distribuição a um grupo selecionado de usuários para a realização de testes com pessoas reais e a coleta de feedbacks. Por fim, o aplicativo será disponibilizado nas lojas oficiais para download.

5 IMPLEMENTAÇÕES FUTURAS

Como possíveis implementações futuras temos:

- Integração com o open finance para registrar automaticamente os gastos da viagem de acordo com o app do banco do usuário.
- Integração com outros aplicativos de turismo como Airbnb, Booking, Trivago e Skyscanner.
- Venda de passagens, hospedagens e passeios diretamente no aplicativo.
- Apresentação de propagandas direcionadas de acordo com os destinos dos usuários.
- Implementar o armazenamento de documentos para possibilitar que o usuário escaneie recibos e bilhetes.
- Compartilhamento de viagens entre usuários, permitindo que mais de uma pessoa possa editar a mesma viagem e que todas as edições reflitam em todas as contas vinculadas.

6 CONSIDERAÇÔES FINAIS

O aplicativo Traveler se propõe a ser uma solução centralizada para o gerenciamento de viagens independentes, visando servir como uma agenda virtual, uma secretária que auxiliará o usuário a planejar suas viagens sem depender de um serviço de terceiros.

A escolha de tecnologias robustas e modernas, como React Native, Node.js, PostgreSQL e os serviços da AWS permitiu a criação de um projeto sólido e escalável. Além disso, a adoção do Scrum com sprints de duas semanas favoreceu entregas incrementais e adaptativas, permitindo maior flexibilidade e resposta às mudanças nos requisitos.

No que diz respeito à segurança, o projeto incorpora técnicas modernas como a criptografia de senhas com bcrypt, isolamento em subnets privadas e uma arquitetura em múltiplas camadas, garantindo a proteção dos dados pessoais dos usuários e a conformidade com a Lei Geral de Proteção de Dados (LGPD). A implantação na nuvem AWS, com serviços como RDS e EC2, reforça a disponibilidade e a escalabilidade do sistema.

Devido aos custos por requisições em API’s privadas e a necessidade de uma estrutura que algumas delas exigem por lidar com dados sensíveis, como API’s de open finance, não foi possível implementar algumas funcionalidades conforme planejado no início do projeto, deste modo deixamos features como integração com a conta bancária, outros aplicativos de viagens e login via redes sociais como melhorias futuras. Essas funcionalidades aprimorariam a experiência do usuário e geram a oportunidade de captação de receita no aplicativo.

Em suma, o Traveler se destaca como uma proposta inovadora e bem fundamentada para o mercado de viagens independentes, com uma arquitetura tecnológica robusta, foco em segurança e um claro roadmap de futuras expansões. A consolidação de informações em um único local, a facilidade de planejamento e o controle financeiro aprimorado representam um valor significativo para os viajantes.

APÊNDICE A — DOCUMENTAÇÃO DA API TRAVELER

1. VISÃO GERAL

A API *Traveler* implementa uma arquitetura RESTful com o propósito de fornecer recursos para que usuários planejem viagens, gerenciem despesas, hospedagens, passeios e transportes, de maneira independente e flexível. Os dados trafegam em JSON e todas as operações seguem convenções HTTP padrão.

- **Base da API**: http://localhost:3333
- **Formato de dados**: JSON (application/json)
- **Padrões de autenticação**: Sem autenticação na versão de testes.
- **Estado da API**: Estágio de desenvolvimento local

2. ENTIDADES PRINCIPAIS

Entidade	**Descrição**
Usuário	Representa a pessoa que cria viagens e adiciona conteúdo
Viagem	Centraliza uma viagem e vincula todas as despesas e eventos
Despesa	Gastos relacionados a uma viagem, podem estar vinculados a outros itens
Transporte / Hospedagem / Passeio	Itens específicos adicionados à viagem
Tipo	Categorias de cada item, ex.: “hotel”, “carro”, “passeio histórico”
Documento	Anexos vinculados a qualquer entidade (ex: PDF de reserva)
Localização	Nomes e códigos de países, estados e municípios

3. PADRÕES DE RESPOSTA HTTP

Status	**Significado**
200 OK	Requisição bem-sucedida

201 Created	Recurso criado com sucesso
204 No Content	Operação sem retorno (ex: delete)
400 Bad Request	Erro de validação nos dados enviados
404 Not Found	Recurso não localizado
500 Internal Server Error	Erro inesperado no servidor

4. ENDPOINTS POR ENTIDADE

4.1 USUÁRIO

- POST /usuarios — Cria novo usuário.

Exemplo de requisição:

```
{
  "nome": "Karina",
  "email": "karina@fatec.com",
  "senha": "1234",
  "municipio_id": 8076,
  "data_nascimento": "1990-05-15"
}
```

- POST /usuarios/login — Realiza login (futuro uso de token JWT)
- GET /usuarios — Lista todos os usuários
- PUT /usuarios/{id}/update — Atualiza nome/email
- DELETE /usuarios/{id}/delete — Remove usuário

4.2 VIAGEM

- POST /viagem/ — Cria viagem

Exemplo de requisição:

```
{
  "nome": "Viagem para Salvador",
  "data_inicio": "2024-11-28",
  "data_fim": "2024-11-30",
  "usuario_id": 1,
  "viagem_destino_id": 155
}
```


- GET /viagem/usuario/{id} — Lista viagens de um usuário
- PUT /viagem/{id}/update — Edita dados da viagem
- DELETE /viagem/{id}/delete — Remove a viagem

4.3 DESPESA

- POST /despesa/ — Cria nova despesa

Exemplo de requisição:


```
{
   "descricao": "Café da manhã",
   "valor": 25.5,
   "data": "2024-11-29T00:00:00.000Z",
   "usuario_id": 1,
   "viagem_id": 1,
   "tipo_id": 2
}
```


- GET /despesa/ — Lista geral
- GET /despesa/usuario/{id} — Por usuário
- GET /despesa/viagem/{id} — Por viagem
- GET /despesa/{id} — Por ID específico
- PUT /despesa/{id}/update — Atualiza valores ou tipo
- DELETE /despesa/{id}/delete — Remove

4.4 HOSPEDAGEM

- POST /hospedagem/

Exemplo de requisição:


```
{
  "nome": "Hotel da Kah",
  "tipo_id": 13,
  "data_checkin": "2024-11-28",
  "data_checkout": "2024-11-30",
  "viagem_id": 1,
  "valor": 300.00,
  "documento_anexo": "./hotelkah.pdf",
  "endereco": "Av Presidente Costa e Silva, 5000",
  "municipio_id": 1116
}
```


- GET /hospedagem/viagem/{id}
- GET /hospedagem
- PUT /hospedagem/{id}/update
- DELETE /hospedagem/delete

4.5 TRANSPORTE

- POST /transporte/

Exemplo de requisição:


```
{
  "nome": "Transporte da Karina",
  "tipo_id": 2,
```

```
  "data": "2024-11-28",
  "viagem_id": 1,
  "valor": 456.00,
  "transporte_destino_id": 5,
  "documento_anexo": "./arquivo.pdf"
}
```


- GET /transporte/viagem/{id}
- GET /transporte/usuario/{id}
- PUT /transporte/{id}/update
- DELETE /transporte/{id}/delete

4.6 PASSEIO

- POST /passeio/

Exemplo de requisição:


```
{
  "nome": "Passeio no Cristo Redentor",
  "tipo_id": 3,
  "data": "2024-12-01",
  "viagem_id": 1,
  "valor": 150.50,
  "documento_anexo": "./documentos/cristo.pdf"
}
```


- GET /passeio/viagem/{id}
- GET /passeio/usuario/{id}
- GET /passeio
- PUT /passeio/{id}/update
- DELETE /passeio/{id}/delete

4.7 TIPOS DE CATEGORIA

Utilizados para preencher campos tipo_id nas entidades anteriores.

- GET /tipo-transporte
- GET /tipo-hospedagem
- GET /tipo-despesa
- GET /tipo-passeio

4.8 LOCALIZAÇÕES

Para vincular a viagens ou perfis de usuário:

- GET /locations/paises
- GET /locations/estados?idPais=29
- GET /locations/municipios?idEstado=907
- GET /locations/{id}

5. CONSIDERAÇÕES FINAIS

A API do sistema Traveler é modular e extensível, permitindo novas funcionalidades como:

- Autenticação JWT e refresh tokens.
- Exportação de itinerário em PDF.
- Envio de notificações e alarmes.
- Dashboard de estatísticas financeiras.

REFERÊNCIAS BIBLIOGRÁFICAS

ABBA, Ihechikara. **What is an ORM – The Meaning of Object Relational Mapping Database Tools**. [S.I.]: FreeCodeCamp, 2022. Disponível em: https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/. Acesso em: 19 set. 2024.

AMAZON. **Computação em nuvem com a AWS**. Seattle: Amazon Web Services, [S.D.]. Disponível em: https://aws.amazon.com/pt/what-is-aws/. Acesso em: 21 jun. 2025.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). **NBR 10719: Informação e documentação — Relatório técnico e/ou científico — Apresentação**. Rio de Janeiro: ABNT, 2011.

BESSA, André. **O que é Node.js?** São Paulo: Alura, 2023. Disponível em: https://www.alura.com.br/artigos/node-js. Acesso em: 3 maio 2025.

CALANCA, Paulo. **SQL e NoSQL: entenda bancos relacionais e não relacionais**. São Paulo: Alura, 2023. Disponível em: https://www.alura.com.br/artigos/sql-nosql-bancos-relacionais-nao-relacionais. Acesso em: 20 jun. 2025.

CORRENTE, Daiane. **Node.js: guia completo para pessoas desenvolvedoras**. Porto Alegre: KingHost, 2025. Disponível em: https://king.host/blog/tecnologia/o-que-e-nodejs/. Acesso em: 20 jun. 2025.

CUNHA, André. **React Native: o que é e tudo sobre o framework**. São Paulo: Alura, 2023. Disponível em: https://www.alura.com.br/artigos/react-native. Acesso em: 17 abr. 2024.

DANIELSSON, William. **React Native application development – A comparison between native Android and React Native**. 2016. Trabalho de Conclusão de Curso (Mestrado) – Linköpings Universitet, Institutionen för datavetenskap. Disponível em: https://liu.diva-portal.org/smash/get/diva2:960828/FULLTEXT01.pdf. Acesso em: 5 jun. 2024.

FUNDAÇÃO GETÚLIO VARGAS (FGV). **Uso de TI no Brasil: país tem mais de dois dispositivos digitais por habitante, revela pesquisa**. Rio de Janeiro: FGV, 2024. Disponível em: https://portal.fgv.br/noticias/uso-ti-brasil-pais-tem-mais-dois-dispositivos-digitais-habitante-revela-pesquisa. Acesso em: 9 maio 2024.

IBM. **O que é PostgreSQL?** Armonk: IBM, [S.D.]. Disponível em: https://www.ibm.com/br-pt/topics/postgresql. Acesso em: 3 maio 2025.

MCCARTNEY, Ava. **As 10 principais tendências tecnológicas estratégicas da Gartner para 2024**. Stamford: Gartner, 2023. Disponível em: https://www.gartner.com.br/pt-br/artigos/as-10-tendencias-tecnologicas-estrategicas-gartner-2024. Acesso em: 20 maio 2024.

PMI – PROJECT MANAGEMENT INSTITUTE. **Um guia do conhecimento em gerenciamento de projetos (Guia PMBOK®)**. 4. ed. Newtown Square: PMI, 2008. Disponível em: https://www.cin.ufpe.br/~if717/slides/PMBOK.pdf. Acesso em: 19 set. 2024.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do Scrum: o guia definitivo para o Scrum – as regras do jogo**. [S.l.]: Scrum Guides, 2020. Disponível em: https://scrumguides.org. Acesso em: 19 set. 2024.

TAFARELO, Saulo. **Brasil registra 112,6 milhões de passageiros e aumento de voos em 2023**. São Paulo: CNN Brasil, 2024. Disponível em: https://www.cnnbrasil.com.br/viagem-e-gastronomia/viagem/brasil-registra-1126-milhoes-de-passageiros-e-aumento-de-voos-em-2023/. Acesso em: 30 jun. 2025.

TIDB. **Understanding Prisma ORM**. [S.l.]: PingCAP, 2024. Disponível em: https://www.pingcap.com/article/understanding-prisma-orm/. Acesso em: 1 jun. 2024.