INSTITUTO FEDERAL DO ESPÍRITO SANTO
CURSO SUPERIOR DE SISTEMAS DE INFORMAÇÃO

JOSÉ GUILHERME SILVA DE LIMA

APLICAÇÃO WEB PARA GERENCIAMENTO E VENDA DE EXCURSÕES

Serra
2022

JOSÉ GUILHERME SILVA DE LIMA

APLICAÇÃO WEB PARA GERENCIAMENTO E VENDA DE EXCURSÕES

Trabalho de Conclusão de Curso apresentado à Coordenadoria do Curso de Sistemas de Informação do Instituto Federal do Espírito Santo, Campus Serra, como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Me. Daniel Ribeiro Trindade

Serra
2022

Dados Internacionais de Catalogação na Publicação (CIP)

L732a 2022

Lima, José Guilherme Silva de
Aplicação Web para gerenciamento e venda de excursões / José Guilherme Silva de Lima. - 2022.
87 f.; il.; 30 cm

Orientador: Prof. Me. Daniel Ribeiro Trindade.
Monografia (graduação) - Instituto Federal do Espírito Santo, Coordenadoria de Informática, Curso de Bacharelado em Sistemas de Informação, 2022.

1. Engenharia de software. 2. Análise de sistemas. 3. Software - Desenvolvimento. 4. Aplicações Web. 5. Turismo. I. Trindade, Daniel Ribeiro. II. Instituto Federal do Espírito Santo. III. Título.

CDD 005.1

Bibliotecária Rogeria Gomes Belchior - CRB6/ES 417

JOSÉ GUILHERME SILVA DE LIMA

APLICAÇÃO WEB PARA GERENCIAMENTO E VENDA DE EXCURSÕES

Trabalho de Conclusão de Curso apresentado como parte das atividades para obtenção do título de Bacharel em Sistemas de Informação, do curso de Bacharelado em Sistemas de Informação do Instituto Federal do Espírito Santo.

Aprovado em 8 de dezembro de 2022.

COMISSÃO EXAMINADORA

Prof. Me. Daniel Ribeiro Trindade (Orientador)
Instituto Federal do Espírito Santo - Campus Serra

Profª Drª Cristina Klippel Dominicini
Instituto Federal do Espírito Santo - Campus Serra

Prof. Me. Felipe Frechiani de Oliveira
Instituto Federal do Espírito Santo - Campus Serra

MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DO ESPÍRITO SANTO
SISTEMA INTEGRADO DE PATRIMÔNIO, ADMINISTRAÇÃO E CONTRATOS

FOLHA DE ASSINATURAS

Emitido em 08/12/2022

FOLHA DE APROVAÇÃO-TCC Nº 2/2022 - SER - CCTMSI (11.02.32.01.08.02.09)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 15/12/2022 09:06)
CRISTINA KLIPPEL DOMINICINI
PROFESSOR DO ENSINO BASICO TECNICO E TECNOLOGICO
SER-CGEN (11.02.32.01.08.02)
Matrícula: 2154146

(Assinado digitalmente em 14/12/2022 17:49)
DANIEL RIBEIRO TRINDADE
PROFESSOR DO ENSINO BASICO TECNICO E TECNOLOGICO
SER - CCTMSI (11.02.32.01.08.02.09)
Matrícula: 2277933

(Assinado digitalmente em 15/12/2022 17:38)
FELIPE FRECHIANI DE OLIVEIRA
PROFESSOR DO ENSINO BASICO TECNICO E TECNOLOGICO
SER-CGEN (11.02.32.01.08.02)
Matrícula: 2275103

Para verificar a autenticidade deste documento entre em https://sipac.ifes.edu.br/documentos/ informando seu número: **2**, ano: **2022**, tipo: **FOLHA DE APROVAÇÃO-TCC**, data de emissão: **14/12/2022** e o código de verificação: **9a4809d70c**

A minha família
Aos meus amigos
À todos que estiveram comigo durante a minha caminhada

AGRADECIMENTOS

A minha família e amigos por todo apoio e ajuda que me deram, que muito contribuíram e motivaram para a realização deste trabalho.

Ao professor Daniel Ribeiro Trindade, meu orientador, por ter desempenhado tal função com paciência e dedicação.

Por fim, a todos os colegas e professores que estiveram comigo durante essa caminhada, compartilhando tantos momentos e aprendizados.

RESUMO

Segundo um levantamento realizado pelo Fecomercio-SP em 2020 o setor de turismo teve uma redução de faturamento de 33% por conta da pandemia do COVID-19. Esse número esconde o grande crescimento que o setor vinha demonstrando nos períodos anteriores e, com a chegada das vacinas para o COVID-19, é muito provável que este setor volte a crescer. Entretanto, essa retomada pode ser um desafio para as pequenas e médias empresas, pois elas nem sempre possuem recursos financeiros para adquirir ou desenvolver as soluções necessárias para o seu negócio. Neste trabalho, desenvolvemos um sistema web que permite a gestão de excursões por agências de viagens, em especial as empresas de pequeno porte. O sistema recebe as reservas de pacotes de viagens dos passageiros e disponibiliza para a agência de turismos as ferramentas necessárias para gerencia-las, e para os viajantes a possibilidade de criar reservas. Para isso, o sistema permite a criação e gestão de excursões, pacotes de viagens, embarque de passageiros e reservas.Para desenvolver esse projeto, foram utilizadas diversas técnicas e metodologias de análise de requisitos, engenharia de software e desenvolvimento web.Ao final do projeto, avaliou-se que os requisitos levantados durante o processo de análise foram desenvolvidos com sucesso, e as ferramentas escolhidas foram eficientes. A aplicação facilita diversos processos, desde a organização até a reserva de uma excursão, podendo evitar problemas que resultariam na insatisfação dos clientes e auxiliando o pequeno empreendedor em sua recolocação no mercado. Também foram levantadas algumas melhorias futuras. A aplicação foi inicialmente implementada para atender, principalmente, as necessidades relacionadas a gestão de excursões. Porém, foi desenvolvida também uma aplicação que permite a criação de reservas por parte do viajante.

Palavras-chave: Turismo. Excursão. Engenharia de Software. Desenvolvimento Web.

ABSTRACT

According to a survey carried out by Fecomercio-SP in 2020, the tourism sector had a 33% revenue reduction due to the COVID-19 pandemic. This number hides the growth that the sector has been demonstrating in previous periods and, with the arrival of vaccines for COVID-19, it is very likely that this sector will return to grow. However, this can be a challenge for small and medium-sized companies, as they do not always have the financial resources to acquire or develop the necessary solutions for their business. In this work, we developed a web system that allows the management of tours by travel agencies, especially small companies. The system receives reservations for travel packages from passengers and provides the travel agency with the necessary tools to manage them, and for travelers the possibility of creating reservations. For this, the system allows the creation and management of excursions, travel packages, passenger boarding and reservations. To develop this project, several techniques and methodologies were used. At the end of the project, it was evaluated that the requirements raised during the analysis process were successfully developed, and the chosen tools were efficient. The application facilitates several processes and can avoid problems that would result in customer dissatisfaction and help the small entrepreneur. Some future improvements were also raised. The application was initially implemented to meet, mainly, the needs of excursion management, but an application was also developed to be used by travel agencies clients.

Keywords: Tourism. Tour. Software Engineering. Web Development.

LISTA DE FIGURAS

Figura 1 – Como os requisitos evoluem 17
Figura 2 – Diagrama de Caso de Usos 19
Figura 3 – Diagrama de Classes 20
Figura 4 – Diagrama de Fluxo 20
Figura 5 – Arquitetura Cliente Servidor 22
Figura 6 – Estrutura de uma requisição HTTP 22
Figura 7 – Exemplo requisição GET 23
Figura 8 – Exemplo requisição Post 24
Figura 9 – Estrutura de uma resposta HTTP 24
Figura 10 – Tipo *User* 27
Figura 11 – Tipo *Query* 28
Figura 12 – Definição de um resolver 28
Figura 13 – Exemplo de solicitação GraphQL 28
Figura 14 – Exemplo de arquitetura com Hasura 29
Figura 15 – Plataforma da CVC 32
Figura 16 – Como funciona o 123Milhas 33
Figura 17 – Subsistemas 40
Figura 18 – Atores 41
Figura 19 – Interação dos Atores com os Subsistemas 41
Figura 20 – Controle de Agência e Controle de Excursão 42
Figura 21 – Relacionamentos de Controle de Excursão 43
Figura 22 – Casos de Uso do Viajante 43
Figura 23 – Diagrama de Arquitetura 44
Figura 24 – Diagrama de Domínio 45
Figura 25 – Diagrame de Entidades e Relacionamento 48
Figura 26 – Fluxo: Criação de Excursão - Parte 1 50
Figura 27 – Fluxo: Cadastro de Item 51
Figura 28 – Fluxo: Cadastro de Pacote 52
Figura 29 – Fluxo: Solicitação de Reserva 53
Figura 30 – Fluxo: Confirmação de Reserva 54
Figura 31 – API GraphQL 55
Figura 32 – Hasura: Criação de Entidade 56
Figura 33 – Hasura: Permissão Gestor de Agência 57
Figura 34 – Serviço de Actions 59
Figura 35 – Person Service 59
Figura 36 – Criação de Token JWT 60
Figura 37 – Tela de Cadastro 62
Figura 38 – Tela de Login 63

Figura 39 – Tela de Listagem de Excursões 63
Figura 40 – Listagem de Excursões - Mobile 64
Figura 41 – Cadastro de Excursão 65
Figura 42 – Detalhes de Excursão 66
Figura 43 – Itens 66
Figura 44 – Novo Item 67
Figura 45 – Listagem de Pacotes 67
Figura 46 – Novo Pacote 68
Figura 47 – Adicionar Itens no Pacote 68
Figura 48 – Detalhes de um Pacote 69
Figura 49 – Listagem de Embarques 69
Figura 50 – Criação de Embarque 70
Figura 51 – Listagem de Reserva 70
Figura 52 – Detalhes de uma Reserva 71
Figura 53 – Alteração da Situação de uma Reserva 71
Figura 54 – Listagem de Viajantes 72
Figura 55 – Detalhes do Viajante 72
Figura 56 – Tela Principal 73
Figura 57 – Tela Principal - Mobile 74
Figura 58 – Tela Principal - Filtro 75
Figura 59 – Pacotes da Excursão 76
Figura 60 – Pacotes da Excursão - Detalhes 77
Figura 61 – Cadastro de Reserva 78
Figura 62 – Cadastro de Viajante 79
Figura 63 – Cadastro de Reserva Completo 80
Figura 64 – Listagem de Reservas 81
Figura 65 – Detalhes da Reserva 82
Figura 66 – Cancelamento de Reserva 83

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS 14
1.1.1 Objetivo Geral 14
1.1.2 Objetivos Específicos 14
1.2 ESTRUTURA DO TRABALHO 14
2 REFERENCIAL TEÓRICO 16
2.1 ENGENHARIA DE SOFTWARE 16
2.1.1 Gerenciamento de requisitos 16
2.1.2 Unified Modeling Language 18
2.1.3 Diagrama de Classes 19
2.1.4 Diagrama de Fluxo 20
2.1.5 Versionamento 21
2.2 DESENVOLVIMENTO WEB 21
2.2.1 Arquitetura Cliente-Servidor 21
2.2.2 Protocolo HTTP 22
2.2.3 Frontend 25
2.2.3.1 Single Page Application (SPA) 25
2.2.3.2 Vue.js 25
2.2.4 Backend 26
2.2.4.1 GraphQL 26
2.2.4.2 Hasura GraphQL Engine 29
2.2.5 Identificação, Autenticação e Autorização 29
2.2.6 Arquitetura de Micro-serviços 30
2.3 AGÊNCIA DE TURISMO E EXCURSÃO 30
2.4 TRABALHOS CORRELATOS 32
2.4.1 Agências de Viagens Online 32
2.4.2 Paytour 33
3 METODOLOGIA 34
3.1 DESCRIÇÃO GERAL DO SISTEMA 34
3.1.1 Sistema de Gestão de Agência 34
3.1.2 Sistema de Reservas 34
3.2 ANÁLISE DE REQUISITOS 35
3.2.1 Requisitos Funcionais 35
3.2.2 Requisitos Não Funcionais 37
3.2.3 Regras de Negócio 37
3.3 MODELAGEM DO SISTEMA 39
3.3.1 Diagrama de Casos de Uso 40
3.3.2 Diagrama de Arquitetura 44

3.3.3 Diagrama de Domínio . . . 45
3.3.4 Diagrama de Entidades e Relacionamento . . . 47
3.3.5 Diagramas de Fluxo . . . 49
3.3.5.1 Diagrama de Fluxo: Criação de Excursão . . . 49
3.3.5.2 Diagrama de Fluxo: Reserva de Pacote . . . 52
3.4 PROCESSO DE DESENVOLVIMENTO . . . 54
3.4.1 Ferramentas . . . 54
3.4.2 Backend . . . 55
3.4.2.1 API GraphQL . . . 55
3.4.2.2 Serviço de Actions . . . 57
3.4.2.3 Autenticação . . . 59
3.4.3 Frontend . . . 60
4 RESULTADOS . . . 61
4.1 SISTEMA DA AGÊNCIA . . . 61
4.1.1 Cadastro de Agência . . . 61
4.1.2 Login . . . 62
4.1.3 Listagem de Excursões . . . 63
4.1.4 Criação de Reserva . . . 65
4.1.5 Detalhes de Excursão . . . 65
4.1.6 Listagens de Itens . . . 66
4.1.7 Cadastro de Item . . . 66
4.1.8 Listagem de Pacote . . . 67
4.1.9 Novo Pacote . . . 67
4.1.10 Detalhes de Um Pacote . . . 68
4.1.11 Listagem de Embarques . . . 69
4.1.12 Criação de Embarque . . . 69
4.1.13 Listagem de Reservas . . . 70
4.1.14 Detalhes de uma Reserva . . . 70
4.1.15 Alteração da Situação de uma Reserva . . . 71
4.1.16 Listagem de Viajantes . . . 71
4.1.17 Detalhes do Viajante . . . 72
4.2 SISTEMA DO VIAJANTE . . . 72
4.2.1 Tela Principal . . . 72
4.2.2 Pacotes da Excursão . . . 75
4.2.3 Cadastro da Reserva . . . 77
4.2.4 Listagem de Reservas . . . 80
4.2.5 Detalhes da Reserva . . . 81
4.2.6 Cancelamento de Reserva . . . 82
5 CONSIDERAÇÕES FINAIS . . . 84
5.1 CONCLUSÃO . . . 84

5.2 TRABALHOS FUTUROS 84
REFERÊNCIAS 86

1 INTRODUÇÃO

O setor de turismo foi um dos mais afetados pela pandemia do Covid 19 em 2020. Através de um levantamento divulgado pelo Fecomercio (2020), baseado em dados obtidos do Instituto Brasileiro de Geografia e Estatística (IBGE), foi mostrado que o faturamento do setor sofreu uma redução de 33%. Segundo uma reportagem divulgada pelo G1 (2020), a queda de faturamento neste período levou 50 mil empresas do ramo à falência no Brasil. Esses números, entretanto, escondem o grande crescimento que o setor vinha demonstrando nos anos anteriores. Em 2018 foi registrado um crescimento de 26% em números de embarques e 18% em faturamento, que se manteve também em 2019, quando foi observado um aumento de 8.1% de faturamento.

Dentre os diversos tipos de empresas que fazem parte deste setor, as agências e operadoras de viagem e turismo são as que mais se destacam. Segundo a pesquisa "O panorama de agências de viagens e operadores turísticos no Brasil", esse segmento possuía cerca de de 36.000 empresas, das quais 99,52% eram Microempreendedores Individuais e Micro e Pequenas Empresas, empregando 50.510 do total de 69.654 pessoas formalmente ocupadas nesse tipo de negócio. (SEBRAE, 2017)

Segundo o SEBRAE (2017), São consideradas agências de turismo empresas privadas que atuam como intermediárias entre os clientes e fornecedores de serviços turísticos. Mas existe um modelo de negócio que vem crescendo e ganhando destaque, principalmente com o auxílio da facilidade de divulgação de serviços através das redes sociais, que são as agências de excursões. Diferente das organizações já citadas anteriormente, elas atuam oferecendo o serviço turístico diretamente, através da organização de passeios e viagens coletivas. Essas são, normalmente, empresas de pequeno porte, podendo ser profissionais autônomos que lidam com toda a parte de organização, negociação com hotéis, restaurantes e meios de transporte.

Por se tratarem de agências de menor expressão, as organizadoras de excursões geralmente não possuem estrutura nem recursos para se modernizarem, através do desenvolvimento ou da contratação de sistemas que auxiliem no processo de organização e acompanhamento das viagens. Muitas vezes acabam optando por soluções simples, como a utilização de planilhas eletrônicas criadas pelos próprios gestores, de forma amadora, que podem gerar inconsistências e erros que, ao longo da execução do passeio, podem levar à insatisfação dos clientes.

Com base nisso, e com a expectativa do retorno do crescimento do setor de turismo com a chegada das vacinas para o COVID-19, surgiu a ideia de desenvolver um sistema web que permita a gestão de excursões, focado em empresas de pequeno porte, de forma que seja possível organizar e controlar pacotes de viagens, embarque de passageiros,

hospedagem, transporte, cronograma de viagem e pagamento. A utilização da aplicação por organizadores de excursão facilitará diversos processos, desde a organização até a execução da viagem, evitando erros e acelerando os processos operacionais.

Através da oferta dessa ferramenta como forma de atender as necessidades das agências de excursão e atraindo os organizadores para a plataforma, esse sistema pode se integrar com um outro sistema especializado na criação de reservas de excursões por viajantes, finalizando um ciclo completo, da criação da excursão pela agência até a criação do pedido de reserva pelo viajante.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo deste trabalho é desenvolver uma ferramenta que permita que empresas do ramo de excursão possam fazer a gestão de suas excursões e coloca-las a venda, através de uma interface que seja simples, de fácil entendimento, para que pessoas interessadas possam buscar por excursões e fazer reservas de pacotes de viagem.

1.1.2 Objetivos Específicos

Para que a aplicação seja adequadamente desenvolvida,é fundamental que aconteça um bom levantamento de requisitos. Dessa forma, para o projeto, destacam-se os seguintes objetivos específicos:

- Realizar a análise de requisitos, identificando as características fundamentais da aplicação;;
- Utilizar as técnicas de Análise de Sistemas e Engenharia de Software para fazer a modelagem e arquitetura da Aplicação;
- Implementar a aplicação utilizando de ferramentas e tecnologias modernas e que são utilizadas no mercado.

1.2 ESTRUTURA DO TRABALHO

Esse trabalho será dividido em 5 capítulos. No capítulo 1 são apresentados a motivação, objetivos gerais, objetivos específicos e a estrutura do trabalho. No capítulo 2 é apresentado o referencial teórico do trabalho, com uma breve descrição dos conceitos de Análise de Sistemas, Engenharia de Software e Desenvolvimento Web e como o trabalho se encaixa com o conteúdo abordado. No capítulo 3 são apresentadas as tecnologias escolhidas para o desenvolvimento do projeto e a motivação para a escolha de cada uma delas, e também

será apresentada a metodologia escolhida para o desenvolvimento do trabalho, passando pelas etapas de levantamento de requisitos, detalhando cada funcionalidade da aplicação, atributos de qualidade, diagramas utilizados e a implementação da API e da Aplicação Web. No capítulo 4 é apresentado o resultado final do projeto, exibindo o produto final desenvolvido. Finalmente, o capítulo 5 apresenta a conclusão do trabalho, declarando se os objetivos foram alcançados e determinando melhorias futuras.

2 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os conceitos que serão fundamentais para a compreensão do desenvolvimento do trabalho. Por se tratar de um projeto de desenvolvimento de software, serão abordados alguns fundamentos básicos de engenharia de software, focando principalmente no processo de análise de requisitos e modelagem de dados. Para o desenvolvimento do projeto são usadas tecnologias web e mobile e, logo, também serão apresentados os fundamentos básicos de desenvolvimento web e móvel.

2.1 ENGENHARIA DE SOFTWARE

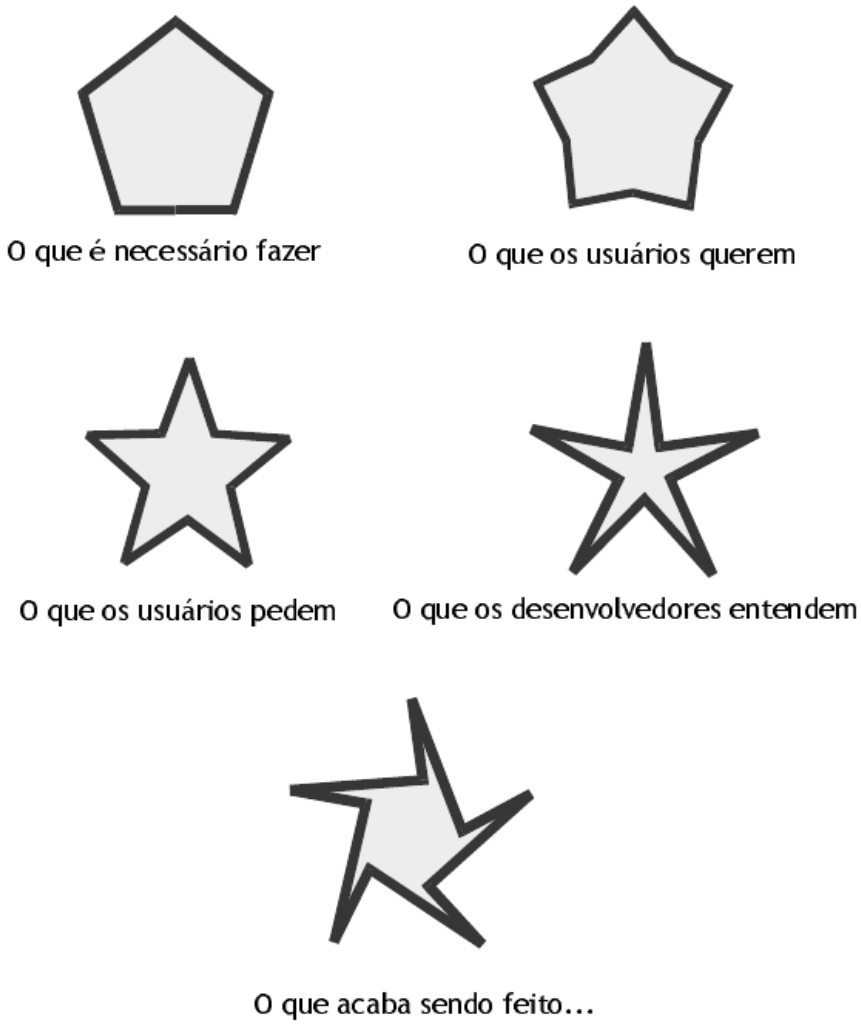
A engenharia de software é um campo da área da computação que tem como objetivo garantir a qualidade do software que será desenvolvido. Ela atua em diversas etapas do processo de desenvolvimento de software, indo desde a especificação de requisitos até a manutenção e documentação.

Quando se trata de softwares desenvolvidos profissionalmente, ou seja, softwares que serão vendidos como produtos finais, ou serão utilizados como parte importante de um negócio, seguir técnicas de engenharia de software se torna primordial para evitar erros e entregar um produto que satisfaça as necessidades do cliente (SOMMERVILE, 2013).

2.1.1 Gerenciamento de requisitos

O gerenciamento de requisitos é um dos processos mais importantes que fazem parte das áreas de atuação da Engenharia de Software. Por mais que existam diversas abordagens a respeito de como garantir a qualidade dentro do contexto de desenvolvimento de um software, a maioria dos autores concordam que garantir que as características, ou requisitos, do software desenvolvido satisfaçam as necessidades dos clientes e/ou usuários é um dos pontos primordiais. Na figura 1, é mostrado de forma descontraída o que pode acontecer quando não é feito um levantamento de requisitos adequado, levando a implementação de uma funcionalidade que é diferente e mais complexa do que seria necessário para o cliente.

Figura 1 – Como os requisitos evoluem



Fonte: Wilson (2000)

De forma mais abrangente, os requisitos de um software podem ser divididos em três grupos: requisitos funcionais, requisitos não funcionais e regras de negócio

Requisitos Funcionais: Os requisitos funcionais descrevem como o sistema deve funcionar e se comportar. Segundo Sommervile (2013), eles podem ser requisitos funcionais de usuário, que devem ser escritos de forma abstrata para que tenham um fácil entendimento, ou requisitos funcionais de sistema, que detalham com precisão as funções do sistema, suas entradas, saídas e possíveis casos de exceção.

Requisitos Não Funcionais: Os requisitos não funcionais são aqueles que descrevem como o sistema deverá entregar as suas funcionalidades, como, por exemplo, requisitos de confiabilidade, desempenho, interface, disponibilidade e outros atributos de qualidade. Esses requisitos tem grande impacto na arquitetura do sistema e nas tecnologias que serão

utilizadas. Se há, por exemplo, o requisito não funcional que defina que o sistema deve ser acessado via web, deverá ser adotada uma arquitetura que permita uma comunicação entre cliente e servidor, e deverão ser utilizadas tecnologias que permitam o desenvolvimento para a web. (SOMMERVILE, 2013),

Regras de Negócio: As regras de negócio podem ser vistas como requisitos do sistema que são orientados a negócio, e que vão indicar como a empresa vai executar suas políticas de negócio e conseguir atingir os seus objetivos. (S.; M.; C., 1997),

Existe uma técnica chamada MoSCoW que é utilizada para classificar os requisitos de um sistema através de sua prioridade. Essa técnica consiste em dividir os requisitos em quatro categorias. (ORACLE, 2000)

Must Have - Tenho que fazer: Requisitos que são fundamentais para o sistema.

Should Have - Devo fazer: Requisitos que não são fundamentais, porém agregam um valor significativo ao sistema.

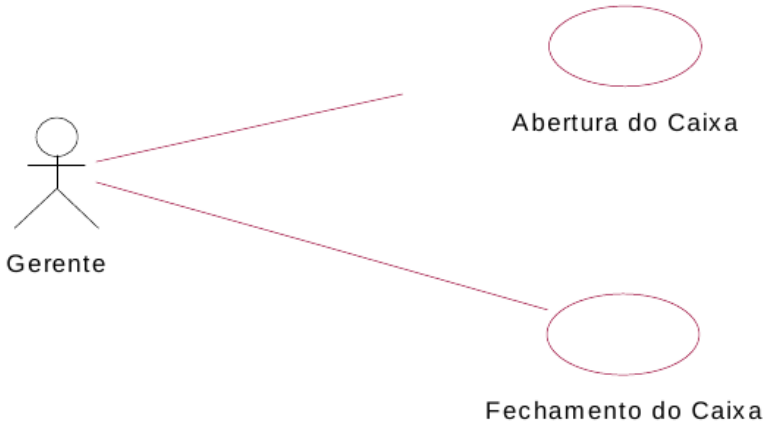
Could Have - Poderia fazer: Requisitos que agregam algum valor ao sistema, mas podem ser facilmente descartados.

Won't Have - Não vou fazer: Requisitos que não são necessários no momento.

2.1.2 Unified Modeling Language

A UML (Unified Modeling Language) nada mais é do que uma linguagem de notação, ou seja, se trata de uma forma de desenhar e apresentar figuras para o processo de análise e projeto de software. Com ela é possível desenhar os diagramas que definirão a estrutura e o comportamento do sistema. Segundo Graig (2005), saber UML não é tão importante quanto a habilidade do engenheiro de software de saber projetar um software, mas se mostra uma ferramenta extremamente útil no decorrer do processo de desenvolvimento. Na figura 2 é apresentado um diagrama de casos de uso, que é um dos diagramas UML mais conhecidos. Ele é composto pela combinação de atores e casos de usos. Atores são representações dos possíveis papeis adotados pelo usuário ao utilizar o sistema, já casos de usos são as funções que podem ser executadas por um determinado ator. Na figura, um usuário com papel de gerente, pode fazer a abertura e fechamento de caixa através da utilização do sistema. (WILSON, 2000)

Figura 2 – Diagrama de Caso de Usos



Fonte: Wilson (2000)

2.1.3 Diagrama de Classes

O diagrama de classe é um dos tipos de diagramas que podem ser desenvolvidos através da utilização da UML. Este diagrama tem como objetivo identificar os conceitos, ou entidades, que fazem parte do domínio de um problema e como eles se relacionam. Muitas vezes este modelo é utilizado como base para a criação do banco de dados, e as classes acabam se tornando tabelas da base de dados. (WILSON, 2000)

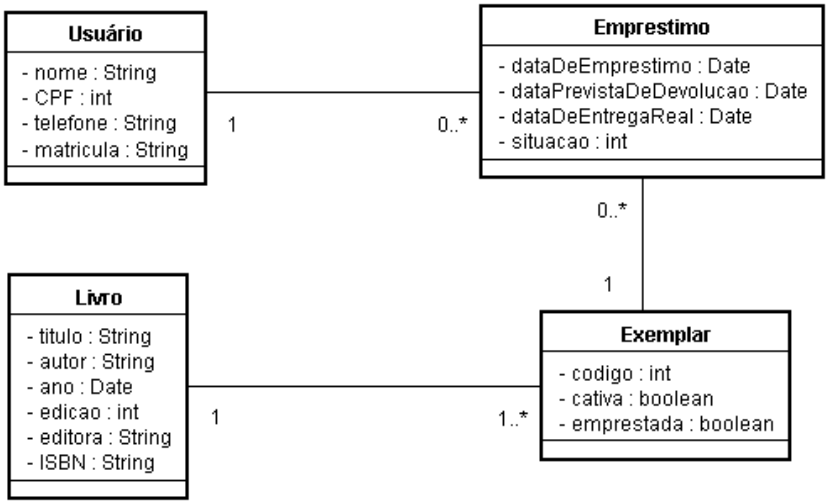
O conceito de classe é bastante abstrato, ela pode se tratar da representação de algo físico do mundo real, algo abstrato, ou um componente de um software. (GRAIG, 2005)

Na UML uma classe é representada por um retângulo que possui o nome da classe, seus atributos e, em alguns casos, os seus métodos (operações que podem ser realizadas por aquela classe). (WILSON, 2000)

Na figura 3, é mostrado como exemplo um diagrama de classes em que foi modelado o domínio de problema de uma biblioteca, onde há as classe Usuário, Empréstimo, Livro e Exemplar. A classe Usuário possui os atributos nome, CPF, telefone e matrícula, e tem uma relação de associação com a classe empréstimo.

A relação de associação representa uma dependência entre duas classes existentes em um diagrama. Isso quer dizer que essas classes sabem da existência uma da outra, de forma que a classe usuário pode possuir nenhum ou vários empréstimos, representado no diagrama pela notação 0...*, e a classe Empréstimo sempre possui um usuário.

Figura 3 – Diagrama de Classes



Fonte: Wilson (2000)

2.1.4 Diagrama de Fluxo

Um diagrama de fluxo permite representar toda a lógica de um sistema e como os dados fluem dentro dele através dos seus processos. (SALVIATI, 1982)

Segundo SALVIATI (1982), em um diagrama de fluxo existem os seguintes símbolos:

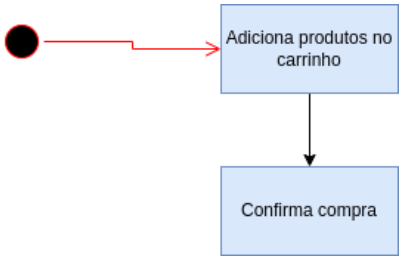
Quadrado: Utilizado para representar entidades externas ao sistema.

Retângulo: Algum processo do sistema.

Seta: Fluxo dos dados.

A figura 4 mostra um exemplo de diagrama de fluxo na prática. É um diagrama simples, que representa de forma resumida como é feita uma compra em uma loja virtual. Os retângulos representam dois processos, que é o de adicionar produtos no carrinho de compras e o de confirmar a compra, e as setas mostram o fluxo de dados entre os processos.

Figura 4 – Diagrama de Fluxo



Fonte: Criação própria

2.1.5 Versionamento

Na Engenhara de Software, o controle de versões tem o papel de fazer o gerenciamento das mudanças que ocorrem no software ao longo do seu desenvolvimento e na sua manutenção, visto que durante o processo de desenvolvimento diversas funcionalidades são acrescentadas, melhorias são implementadas e erros são corrigidos, fazendo com que seja importante que os desenvolvedores saibam o que aconteceu em cada alteração e quais são os impactos dessas mudanças no funcionamento do sistema.

Além disso, durante o processo de planejamento muitas vezes são adotadas estratégias de entrega de funcionalidades de forma incremental, onde partes do projeto são adicionadas aos poucos, e a cada entrega existe uma versão do produto que pode ser utilizada pelo cliente, necessitando de um controle mais cuidadoso em relação as versões que estão sendo publicadas e permitindo que o cliente esteja ciente do que foi alterado em cada atualização (SOMMERVILE, 2013).

Existem muitas ferramentas de controle de versão que são utilizadas no mercado, algumas até antigas, como o CVS (Concurrent Versions System), criado em 1986. Porém para o desenvolvimento desse trabalho optou-se por utilizar o Git, uma das ferramentas mais utilizadas no mercado atual.

O Git é um sistema de controle de versão *open source*. Através dele é possível criar repositórios de código e controlar todo o histórico de alterações do software que está sendo desenvolvido. Para o desenvolvimento desse projeto, o Git foi utilizado juntamente com o GitHub. O Github é uma plataforma que permite a hospedagem de repositórios Git na nuvem, além de possuir diversos *plugins* que auxiliam no processo de desenvolvimento.

2.2 DESENVOLVIMENTO WEB

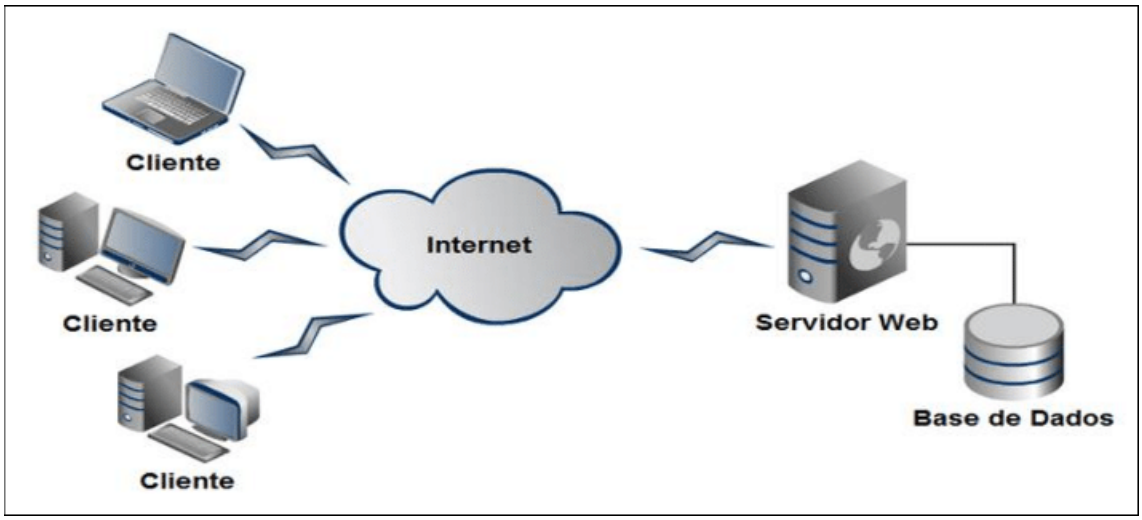
O desenvolvimento web é um ramo da área de desenvolvimento de software voltado para o desenvolvimento de softwares que serão acessados via rede. Existe uma variedade imensa de tipos de aplicações que podem ser desenvolvidas com essa finalidade, desde sites institucionais até sistemas complexos, que podem ser utilizados somente em redes internas, como o caso de sistemas administrativos de empresas, ou aplicações que são acessadas via internet, essas últimas as que mais utilizamos em nosso dia a dia.

2.2.1 Arquitetura Cliente-Servidor

As aplicações desenvolvidas para web funcionam através da arquitetura Cliente-Servidor, onde há a combinação de dois módulos:o Cliente (ou *frontend*) e o servidor (ou *backend*). A módulo do Cliente, na maioria das vezes, é executada no dispositivo do usuário, podendo ser um computador ou outro dispositivo, como um smartfone, e a camada do

Servidor é executada em servidores, acessados através da rede. Essas dois módulos se comunicam, seguindo um protocolo de comunicação específico que possibilita que o cliente faça solicitações ao módulo servidor, o qual as processa e responde ao cliente com dados de resposta. Essa arquitetura permite que várias aplicações sejam executadas ao mesmo tempo em diferentes *hosts*, sempre consultando uma mesma fonte de recursos, como é mostrado na figura 23, onde há 3 aplicações *frontend* em execução em dispositivos isolados, que se comunicam com o servidor através da internet. (KALAKOTA, 1997).

Figura 5 – Arquitetura Cliente Servidor



Fonte: Seruffo (2013)

2.2.2 Protocolo HTTP

O *Hyper Text Transport Protocol*, ou HTTP, é o protocolo de comunicação utilizado em aplicações web que permite a comunicação entre a aplicação que está no lado cliente com a aplicação que esta no lado servidor. O funcionamento geral do protocolo é baseado no fluxo de solicitação e resposta.

Um navegador web ao tentar acessar um site através de uma URL digitada pelo usuário, esta no papel de cliente. O navegador fará uma solicitação (*request*) através do protocolo http para uma aplicação que está sendo executada em um servidor.

Como mostrado na figura 6, a estrutura de uma requisição é dividida em verbo, URL, versão do HTTP, cabeçalho e corpo da requisição.

Figura 6 – Estrutura de uma requisição HTTP

Fonte: Ferreira (2019)

Na figura 7 é mostrado um exemplo de requisição HTTP. Todas as requisições são legíveis para o ser humano. Neste caso temos uma requisição com o método GET, solicitando um recurso que esta em um servidor que possui a URL www.exemplo.com. A versão do HTTP é a 1.1, e as demais informações fazem parte do cabeçalho da requisição, como o User-agent, que mostra detalhes sobre o software que realizou a requisição.

Figura 7 – Exemplo requisição GET

```
GET /hello.txt HTTP/1.1
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
Host: www.example.com
Accept-Language: en, mi
```

Fonte: w3big (2019)

Através da URL também podem ser enviados dados adicionais a requisição, chamados de string de consulta (*query string*). A *string* de consulta é representada pelo caractere ? e pode ser concatenada com outras strings de consultas através do caractere &. Para fazer uma requisição e adicionar duas strings de consulta, buscando um usuário de nome "José", com idade "25", seria utilizada a seguinte URL: www.exemplo.com.br/?nome=jose&idade=25.

Na requisição HTTP, o método é um verbo que representa qual é a operação que o cliente deseja realizar, podendo ser GET, POST, DELETE ou PUT.

GET: Método utilizado para fazer a solicitação de algum recurso. Na maioria das vezes não são enviados dados no corpo da requisição.

POST: Método utilizado quando se deseja enviar dados ao servidor, com intuito de executar ações ou modificar o seu estado interno. Autenticação, envio de formulários e criação de novas entradas no banco de dados são exemplos de utilizações deste método.

No POST, os dados podem ser enviados através do corpo da requisição, ou como *string* de consulta na URL. Na figura 8 é mostrado como exemplo uma requisição realizada para o servidor que possui a URL foo.com, em que os dados são enviados no corpo da requisição. O cabeçalho Content-Type mostra o formato que o corpo possui, que neste caso é application/x-www-form-urlencoded, um tipo de codificação. No corpo da requisição é enviado o conteúdo "say=Hito=Mom".

Figura 8 – Exemplo requisição Post

```
POST / HTTP/1.1
Host: foo.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

say=Hi&to=Mom
```

Fonte: w3big (2019)

- **PUT:** Da mesma forma que o POST, esse método envia dados no corpo da requisição, mas com a intenção de fazer a atualização de um recurso.

- **DELETE:** Método utilizado para fazer a remoção de um recurso.

A aplicação do lado servidor ira receber essas requisições, fazer o processamento e mandar uma resposta que também contem um cabeçalho. Este contém algumas informações importantes, como o código de status para aquela requisição, e o corpo da resposta em texto. O header Content-Type identifica o formato do corpo da resposta, que pode ser desde uma imagem até uma pagina HTML. A figura 9 mostra a estrutura de uma resposta enviada por um servidor.

Figura 9 – Estrutura de uma resposta HTTP

Fonte: Ferreira (2019)

O código de resposta de uma requisição HTTP tem a finalidade de indicar se a requisição foi corretamente concluída, ou se houve algum tipo de falha. Esses são alguns exemplos de código de resposta:

200 - Ok: Houve sucesso na requisição.

400 - Bad Request: O servidor não entendeu a sua requisição. Algum dado foi enviado de forma incorreta.

401 - Unauthorized: A aplicação que fez a requisição não esta autorizada a acessar o recurso desejado.

404 - Not found: O servidor não encontrou o recurso que foi solicitado na requisição.

500 - Internal Server Error: Houve um erro interno na aplicação do lado servidor, ou ela não soube como lidar com a requisição.

2.2.3 Frontend

De acordo com Kalakota (1997), a camada de *frontend* é responsável pela exibição, ou seja, por toda a parte em que ocorrerá a interação do usuário com a aplicação. A aplicação *frontend* é executada no navegador web, que é um programa que faz a rendenrização da informação através da linguagem de marcação *Hypertext Markup Language* (HTML) e do CSS (*Cascading Style Sheets*), que é responsável por adicionar estilo e animações simples ao documento web.

A combinação de CSS com HTML é muito poderosa. Porém, para adicionar mais dinâmica as aplicações web o navegador permite a utilização de código Javascript, que é uma linguagem de programação interpretada que consegue tornar as páginas mais interativas.

2.2.3.1 Single Page Application (SPA)

Uma *Single Page Application* (Aplicação de Página Única) é uma aplicação em que todo o documento web, ou pelo menos a maior parte deles, é carregado uma única vez pelo navegador e a partir dai os demais recursos são solicitados por demanda dinamicamente ao servidor. Diferente de uma aplicação tradicional, *Server Side Rendering*, em que ao fazer a navegação para outra seção da aplicação seria realizada uma nova *request* ao servidor para obter a página desejada, em uma SPA a alteração para a próxima pagina seria realizada dinamicamente, através da manipulação do documento web com Javascript, e seriam requisitados ao servidor somente os dados dinâmicos necessários, que retornariam em arquivos XML ou JSON. A utilização de SPAs torna a experiência do usuário muito mais agradável, trás economia de banda e permite a implementação de interfaces web muito mais semelhantes a interfaces *desktop* e móvel (MARCONDES, 2013).

Existe uma grande variedade de *frameworks* para desenvolvimento de SPAs, como o Vue.JS, Angular e React.Js. Para o desenvolvimento desse trabalho foi optado pelo Vue.Js.

2.2.3.2 Vue.js

O Vue.Js é um *framework* Javascript *open source* (código aberto) utilizado para o desenvolvimento de interfaces de usuário. Ele foi projetado para ser adotado de forma

incremental. Seu ecossistema possibilita a criação de SPAs complexas, de forma modular e com certa facilidade de aprendizado (VUE, 2022).

2.2.4 Backend

A aplicação que é desenvolvida para ser utilizada no lado servidor, também é chamada de *Backend*, e é responsável, como o nome já diz, pelo o que está por trás da tela. É nesta aplicação onde é implementada toda a lógica de negócio complexa do sistema. No *backend* é onde também se lida com a persistência de dados, já que muitas aplicações precisam armazenar informações (KALAKOTA, 1997).

Existem diversas linguagens que podem ser utilizadas para o desenvolvimento *backend*, como Java, PHP, Python, C, Ruby e Javascript. Apesar de Javascript ser uma linguagem originalmente projetada para ser executada somente no lado do frontend, é possível utilizá-la também no backend quando utilizado o módulo Node.js no servidor. O Node.js pode ser visto como um ambiente de execução de javascript que permite a utilização da linguagem no lado servidor, fora de um navegador.

Com a crescente utilização de aplicações web, foram surgindo também diversos *frameworks* que vem com a proposta de facilitar o desenvolvimento web. Entre os *frameworks backend* mais utilizados podemos listar:

Frameworks baseados em PHP: Laravel, Zend framework e CakePHP;

Frameworks baseados em Java: Spring;

Frameworks baseados em Python: Django;

Frameworks baseados em Javascript: Express, Adonis.js e Nest.js;

Para o desenvolvimento deste projeto, boa parte do *backend* será desenvolvido com o auxilio do Hasura, uma ferramenta implementada em Haskel, que será melhor abordada nas proximas seções, e através da implementação de serviços com responsabilidades específicas, como lidar com autenticação de usuário, utilizando Javascript e Express.

2.2.4.1 GraphQL

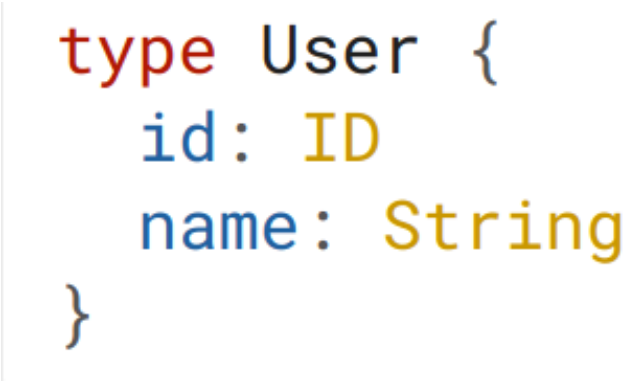
O Graphql, é uma linguagem de consultas para APIs que foi implementada com a ideia de permitir que o cliente sempre obtenha em suas requisições somente aquilo que necessita, fazendo com que a aplicação que utilize uma API GraphQl possa ter total controle sobre os dados que está consumindo. (GRAPHQL, 2022)

Em um servidor que utiliza GraphQL, todas as requisições são enviadas para um único ponto de entrada, e no corpo da requisição é enviado uma estrutura em texto em forma de grafo, que representa os dados que são requisitados pela aplicação cliente. (GRAPHQL, 2022)

O GraphQL possui os seguintes conceitos básicos.

Schema: Tudo em GraphQL são tipos. Existem os tipos escalares (String, Int, Boolean, ID), e os tipos que podem ser criados pelo desenvolvedor como, por exemplo, um tipo Usuário. O Schema é um arquivo que define tudo que existe na API e que pode ser consultado pelo *client*. Na figura 10 é mostrado como exemplo a definição de um Schema que possui um tipo *User* (usuário), que possui dois campos escalares, o campo do tipo ID, chamado id, e um campo do tipo String, chamado *name*.

Figura 10 – Tipo *User*

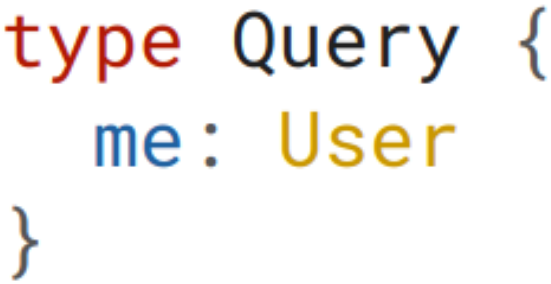


Fonte: GraphQL (2022)

Query e Mutation: Existem dois tipos base que devem ser criados em um Schema GraphQL, que são o tipo Query e o tipo Mutation. o tipo Query contem como campos todas as consultas de leitura que podem ser realizadas na API, e o tipo Mutation todas consultas de criação e atualização de dados. O tipo Query pode ser visto também como a "mãe"de todas as consultas, de forma que toda consulta feita a uma API GraphQL deva começar através de uma query.

Na figura 11 é mostrado um exemplo de definição de um tipo Query em um Schema GraphQL. No exemplo existe o campo *me* que representa uma consulta que pode ser realizada nesta API. O campo *me* é do tipo User, mostrado anteriormente na figura 10. Na prática, esse exemplo mostra que ao fazer a consulta por *me* na API, será retornado um usuário.

Figura 11 – Tipo *Query*



Fonte: GraphQL (2022)

Resolvers: Os resolvers são as funções que resolvem cada solicitação recebida. Cada tipo no Schema pode ter um resolver que diz como aquele campo deve ser resolvido.

Uma requisição a uma API GraphQL deve especificar no teu corpo um texto que define qual a query ou mutation que deve ser realizada e quais campos devem ser retornados para o *client*. Os resolvers serão executados e o servidor retornará uma resposta. Na figura 12 é mostrado como exemplo a definição de um resolver em javascript para a query *me*.

Figura 12 – Definição de um resolver

```
{
    resolvers: {
      Query: {
        me: function (request) {
          return getUser(request.auth.user)
        }
      },
    }
}
```

A figura 16 mostra o exemplo de uma requisição GraphQL. Nesse exemplo é realizada uma query de "hero", e consultando os seus campos filhos, "name"e "appearsIn". Na figura, o lado esquerdo é a requisição e o direito é resposta, retornando um objeto "hero", com os seus atributos.

Figura 13 – Exemplo de solicitação GraphQL

```
{
  hero {
    name
    appearsIn
  }
}
```

```
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "appearsIn": [
        "NEWHOPE",
        "EMPIRE",
        "JEDI"
```

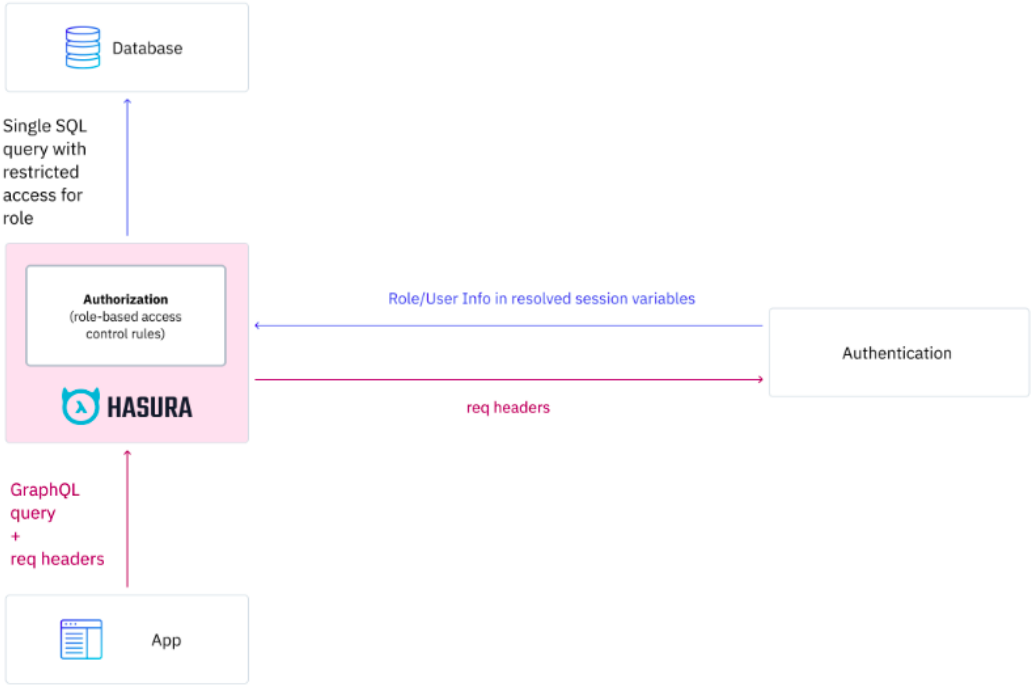
Fonte: GraphQL (2022)

2.2.4.2 Hasura GraphQL Engine

O Hasura Graphql Engine é uma ferramenta que é capaz de criar um servidor GraphQL com todas as funcionalidades de CRUD (Create, Read, Update, Delete), baseado em um banco de dados (Postgre ou MySQL). Além de criar um CRUD completo, o Hasura também permite o gerenciamento de permissões de usuário, consultas em tempo real e gatilhos de webhook para eventos de criação e atualização de dados (HASURA, 2022).

A figura 14 mostra como pode ficar a arquitetura de um sistema web que utiliza o Hasura. Ele pode ser configurado para ficar entre sua aplicação *client* e o seu banco de dados, e se integrar com serviços de autenticação e outros servidores GraphQL remotos.

Figura 14 – Exemplo de arquitetura com Hasura



Fonte: Hasura (2022)

2.2.5 Identificação, Autenticação e Autorização

Quando se trata de aplicações para a web, muitas vezes há a necessidade de se preocupar com a garantia da privacidade do usuário e com a segurança. Segundo Rahman, Shuva e Ali (2016), existem três fatores que são os pilares dessas questões, que são a identificação, a autenticação e a autorização.

A identificação é o que permite o reconhecimento de uma entidade que está fazendo uma requisição. A Autenticação é a forma na qual a entidade que está fazendo a solicitação comprova a sua identidade. Por ultimo, a autorização determina a que recursos essa entidade já identificada e autenticada terá acesso. (RAHMAN; SHUVA; ALI, 2016)

Existe um padrão que é utilizado para lidar com autenticação e autorização, chamado, JWT (*JSON Web Token*), que permite uma comunicação mais segura entre duas aplicações, através de *tokens*. Os *tokens* são objetos JSON em código Base64 assinados digitalmente através de segredos ou com pares de chaves pública/privada. Por serem assinados, os *tokens* conseguem garantir que a entidade que está fazendo uma requisição realmente é quem diz ser.

Um *token* JWT é dividido em três partes: cabeçalho, carga útil e assinatura.

Cabeçalho: O cabeçalho de um *token* JWT armazena dois campos. No campo "alg", é informado qual foi o algoritmo utilizado para fazer a assinatura, e no campo "typ"o tipo de *token*.

2.2.6 Arquitetura de Micro-serviços

Uma aplicação em Micro-serviços adota uma estrutura em que os seus componentes, chamados serviços, são divididos em pequenas aplicações independentes com responsabilidades específicas, diferente de uma aplicação chamada monolítica, em que toda a aplicação é implementada em uma única unidade. (LEWIS, 2014).

Para ficar mais claro, pode-se imaginar uma aplicação web qualquer. Por mais que que exista um módulo que seja executado no lado cliente e outro que seja executado no lado servidor, caso a aplicação que esta no lado servidor seja monolítica, no momento do processamento de uma requisição, todo o processo será executado em uma única unidade, compartilhando recursos de processamento e memória, e qualquer mudança que seja feita no sistema resultará na alteração e na implantação dessa unidade como um todo. Porém, quando se trata de uma aplicação em micro serviços, a lógica de negócio é divida em serviços especializados, que são executados em unidades separadas, em processos diferentes, que podem estar inclusive em máquinas diferentes se comunicando através do protocolo HTTP. Através dessa arquitetura, uma aplicação de vendas pode ter, por exemplo, um serviço de autenticação e um serviço de pagamento, de modo que fique muito mais seguro fazer modificações em um desses módulos sem que outros sejam diretamente afetados.

2.3 AGÊNCIA DE TURISMO E EXCURSÃO

Serviços turísticos são quaisquer serviços que são oferecidos para melhorar a qualidade de uma viagem ou passeio. Esses serviços podem abranger diversos segmentos, como hospedagem, alimentação e transporte, e são oferecidos por diversos tipos de empresas, como hotéis e empresas aéreas. As agências de turismo são empresas privadas que atuam como intermediárias entre os clientes e fornecedores de serviços turísticos, de modo que

consigam fornecer para os consumidores produtos e serviços com preços mais atrativos do que eles conseguiriam ao se dirigir diretamente aos fornecedores. (SEBRAE, 2017)

Existem muitas empresas grandes e já consolidadas que atuam com esse papel, como a CVC e a Decolar. Essas duas empresas são exemplos de organizações do setor que se modernizaram com a utilização de plataformas online para possibilitar que o cliente consiga ter acesso aos seus serviços sem precisar sair de casa.

Segundo SEBRAE (2017), podemos classificar as agências de turismo através das seguintes categorias:

Agências de viagens detalhistas: São agências que não elaboram os seus próprios produtos, tendo como foco a comercialização de viagens com roteiros pré-estabelecidos adquiridos por agências maioristas ou operadoras de turismo. Essas são as agências em maior número no mercado.

Agências de viagens maioristas: São agências que elaboram os roteiros de viagem e comercializam com as agências detalhistas. Raramente esse tipo de agência promove a venda direto ao consumidor final.

Agências de viagens receptivas: Esse tipo de agência presta serviço para as demais agências de turismo, oferecendo ao turista uma variedade de serviços, como passeios e assessoria ao turista durante a viagem. Também podem comercializar produtos a parte do pacote turístico adquirido pelo cliente.

Agências de viagens consolidadoras: Nem toda agência de turismo pode fazer a emissão de passagens diretamente com as companhias aéreas, pois precisam de um registro no IATA (*International Air Transport Association*). Por isso as agências consolidadoras atuam como distribuidoras de passagens aéreas para as agências que não possuem tal registro.

Além das empresas que atuam como intermediárias entre cliente e fornecedor, existe um modelo de negócio que vem crescendo e ganhando destaque, principalmente com o auxílio da facilidade de divulgação de serviços através das redes sociais, que são as agências de excursões. Diferente das organizações já citadas anteriormente, elas atuam oferecendo o serviço turístico diretamente, através da organização de passeios e viagens coletivas. Essas são, normalmente, empresas de pequeno porte, podendo ser profissionais autônomos que lidam com toda a parte de organização, negociação com hotéis, restaurantes e meios de transporte. (SEBRAE, 2017)

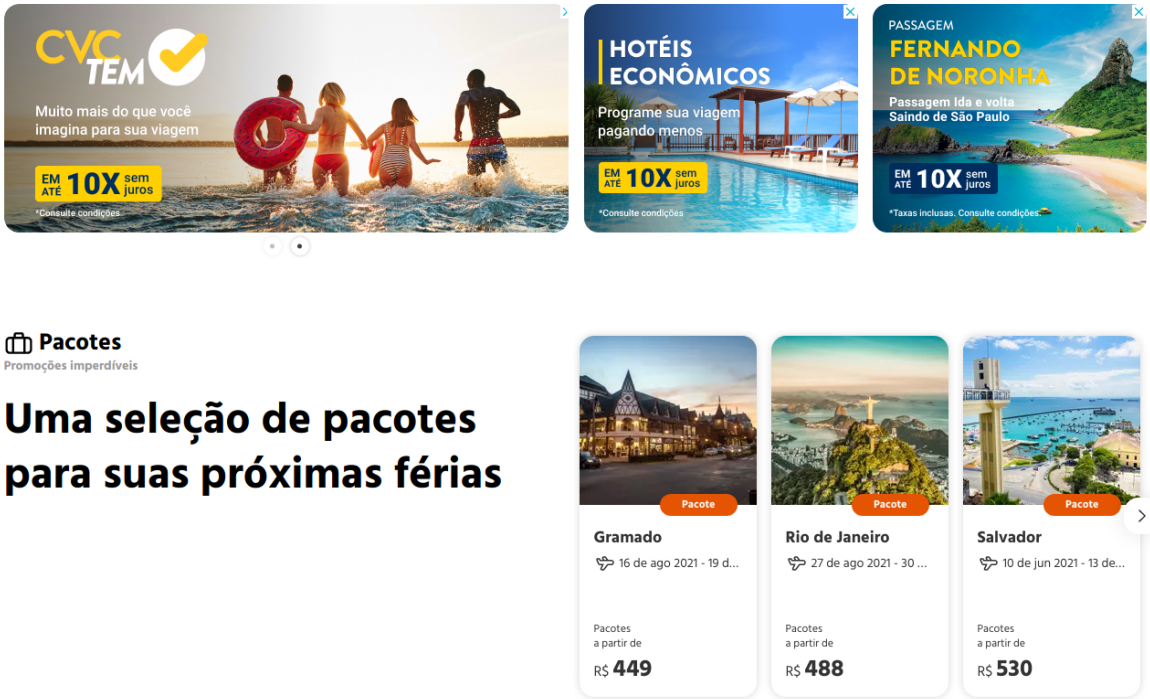
2.4 TRABALHOS CORRELATOS

Nessa seção seram expostos alguns sites e sistemas que apresentam soluções para o setor de turismo. Não é facil encontrar informações detalhadas sobre plataformas que trabalham com a gestão de viagens e excursões, pelo fato de serem produtos comerciais voltados para outras empresas e, por isso, para ter acesso a essas plataformas é necessário fazer solicitações de demonstração e ser uma pessoa jurídica.

2.4.1 Agências de Viagens Online

As OTAs, ou Agências de Viagens Online, são sites focados na venda de serviços relacionados a viagens para os consumidores. Existem uma série de plataformas já conhecidas e consolidadas nesse ramo, como a Decolar, 123Milhas, Expedia, Booking e a CVC.

Figura 15 – Plataforma da CVC



Fonte: https://www.cvc.com.br/

Esses sites fazem a venda tanto de produtos/serviços individuais, como passagens aéreas, hospedagem em hotel, aluguel de carros e seguros, quanto pacotes completos e customizados de acordo com os desejos dos clientes. Através deles o cliente pode procurar por destinos ou datas específicas, procurando por pacotes de viagens que atendam suas necessidades ou encontrar promoções, porém não trabalham com excursões coletivas, que são o foco desse trabalho, e não oferecem uma plataforma de gestão para o organizador de viagens e passeios.

Figura 16 – Como funciona o 123Milhas

Fonte: https://123milhas.com/pacotes-de-viagem

2.4.2 Paytour

O Paytour é uma plataforma especializada em otimizar e impulsionar as vendas de empresas organizadoras de excursões. Ela oferece a agência um sistema de gestão, onde é possível gerenciar os seus passeios, clientes, reservas, vendas e financeiro, além de monitorar o desempenho da organização através de relatórios. Também é oferecido ao organizador a possibilidade de criar um canal de venda online, com interface customizada, integrado com diversas formas de pagamentos, como Rede, Paypal, Pagseguro e Cielo. Para adquirir a ferramenta é necessário fazer a assinatura de planos, em que os valores não foram disponibilizados no site oficial.

A plataforma que será desenvolvida nesse trabalho se difere da Paytour na forma de divulgação e venda das excursões. A Paytour cria para a agência um canal de venda exclusivo e customizado, agregando valor através da exclusividade e personalização, já a plataforma que será desenvolvida, fornece uma única plataforma onde serão disponibilizadas todas as excursões, de diversas agências.

3 METODOLOGIA

Para o desenvolvimento deste trabalho, foram utilizadas diversas metodologias e conceitos de desenvolvimento web, análise de requisitos e engenharia de software. As ferramentas foram escolhidas primeiramente focando em atender aos requisitos do sistema, mas também na utilização de tecnologias modernas, que são amplamente utilizadas no mercado atual.

3.1 DESCRIÇÃO GERAL DO SISTEMA

O sistema proposto neste trabalho tem o objetivo de auxiliar as pequenas agências de excursão em duas frentes: a gestão de excursões e pacotes de viagens; e a divulgação de excursões e controle reservas. Para isso, foram desenvolvidas duas aplicações *frontend*, com responsabilidades diferentes, mas que consomem o mesmo *backend*. Essas aplicações serão descritas nas próximas seções como dois sistemas, o Sistema de Gestão da Agência e o Sistema de Reservas.

3.1.1 Sistema de Gestão de Agência

Com foco na gestão de agências e excursões, essa aplicação permite que uma agência de excursão se cadastre e consiga gerenciar diversas excursões. Pelo sistema é possível adicionar uma nova excursão, seus locais de embarque e fazer o controle de todos os seus pacotes de viagem, viajantes, reservas e itens inclusos.

Uma excursão no sistema é simplesmente a representação de uma excursão no mundo real. Pode ser vista como um passeio que tem uma data de inicio e uma data de fim, e possui diversos viajantes como participantes.

Cada excursão pode ter diversos pacotes de viagem. Um pacote de viagem é a entidade que pode ser reservada por um viajante. Basicamente ele é um conjunto de itens que fazem parte da excursão, como café da manhã ou um passeio de lancha. Através do sistema, o gestor pode definir quais itens fazem parte de cada pacote, e criar pacotes totalmente customizados, com preços diferentes, quantidades diferentes e para quantidade de pessoas diferentes, desde pacotes individuais até pacotes para um grupo de viajantes.

Toda solicitação de reserva realizada pelo sistema de reservas pode ser gerenciada pelo sistema de gestão da agência. O gestor poderá confirmar pagamentos de reservas, visualizar as reservas que já estão confirmadas e também cancelá-las.

3.1.2 Sistema de Reservas

Focando na busca por excursões e realização de reservas, foi criada uma aplicação que permite que viajantes interessados se cadastrem, encontrem excursões criadas pelas agências

de viagem, façam solicitações de reservas e adicionem todos os viajantes que farão parte do passeio.

Ao acessar o sistema, a primeira coisa que o usuário encontrará será a tela de busca por excursões. Nessa tela ele verá uma lista de excursões, e poderá fazer uma filtragem por origem, destino, data e quantidade de pessoas por pacote.

Para realizar uma reserva, o usuário deverá se cadastrar no sistema. Ao efetivar uma nova reserva, o viajante poderá adicionar mais de um viajante que fará parte do passeio, bastando adicionar nome, telefone e documento para cada um deles. Esses outros viajantes não precisam ser usuários cadastrados no sistema.

Por esse sistema o viajante pode acompanhar a situação da sua reserva, saber se ela ja foi confirmada pela agência e também cancelá-la. Nesse primeiro momento, o sistema não terá controle de pagamento, por isso todo contexto de pagamento deverá ocorrer por fora da plataforma, através do contato entre o viajante solicitante da reserva e o gestor da agência.

3.2 ANÁLISE DE REQUISITOS

A primeira etapa do processo de desenvolvimento do projeto foi fazer a análise de requisitos do sistema. Para isso foi usada a técnica *MoSCoW*.

3.2.1 Requisitos Funcionais

Aplicando-se a regra de *MoSCoW*, foram levantados os seguintes requisitos funcionais, presentes na tabela 1.

Tabela 1 – Requisitos funcionais

Requisito	Descrição	Prioridade
(RF01) Controle de Agência	CRUD	Must
(RF02) Controle de Excursão	CRUD	Must
(RF03) Controle de Embarques	CRUD	Should
(RF04) Controle de Pacote	CRUD	Must
(RF05) Controle de Itens de Pacote	CRUD	Must
(RF06) Controle de Reserva	CRUD	Must

(RF07) Cancelamento Automático de Reserva	O sistema deve cancelar automaticamente reservas em que a data de pagamento foi expirada.	Must
(RF08) Login	O sistema deve permitir que os usuários se autentiquem	Must
(RF09)Busca por Excursões	O sistema deve permitir que viajantes busquem excursões.	Must
(RF010) Solicitação de Reserva	O sistema deve permitir que viajantes solicitem a reserva de uma excursão	Must
(RF011) Cancelamento de Reserva	O sistema deve permitir que viajantes e gestores de agência cancelem uma reserva	Must
(RF012) Controle de Guias de Excursão	CRUD	Won't
(RF013) Pagamento de reserva	O sistema deve permitir que viajantes façam o pagamento de uma reserva	Won't
(RF014) Cadastro de Viajante	O sistema deve permitir que novos viajantes se cadastrem	Must

Fonte: Criação própria

Para tornar o desenvolvimento mais simples, foi decidido que em uma fase inicial o sistema não incluiria alguns requisitos.

O *backend* do sistema foi modelado para permitir que uma agência pudesse fazer o cadastro de usuários com a função de guia de excursão. Esse tipo de usuário poderia acompanhar uma excursão, porém não teria permissão para fazer alterações nas informações a respeito dela. Esse seria o Requisito de Controle de Guias de Excursão. Porém, para tornar o desenvolvimento do *frontend* mais simples, nessa fase inicial foi optado por não adicionar essa funcionalidade.

Outro requisito que foi levantado foi o de Pagamento de Reserva. Esse requisito envolveria a integração com algum serviço de pagamento, e faria todo o controle do fluxo do pagamento de uma reserva de pacote. Para tornar o desenvolvimento inicial mais simples, também foi optado por não desenvolver esse requisito na primeira fase do sistema.

3.2.2 Requisitos Não Funcionais

Aplicando-se a regra de *MoSCoW*, foram levantados os seguintes requisitos não funcionais, presentes na tabela 2

Tabela 2 – Requisitos não funcionais

Requisito	Categoria	Prioridade
(RNF01) O sistema será construído para rodar em ambiente web, e deverá ter um design responsivo, para que possa ser utilizado tanto através do *desktop* quando através de dispositivos móveis	Usabilidade	Must
(RNF02) A interface do sistema deve se comportar adequadamente independente do navegador utilizado pelo usuário	Usabilidade	Must
(RNF03) Para a autenticação, cada usuário do sistema se autenticar através de login e senha. A senha deverá ser criptografada no banco de dados	Segurança	Must
(RNF04) A API do sistema será pública, mas alguns recursos só poderão ser acessados com a identificação do usuário através da autenticação com JWT	Segurança	Must
(RNF05) A API do sistema deverá fazer o controle de permissões de usuário. Um viajante não poderá, por exemplo, ter acesso a reservas criadas por outros viajantes.	Segurança	Must

Fonte: Criação própria

3.2.3 Regras de Negócio

Aplicando-se a regra de *MoSCoW*, foram levantados as seguintes regras de negócio, presentes na tabela 3

Tabela 3 – Regras de negócio

Regra de Negócio	Requisitos Relacionados	Prioridade

(RN01) Toda agência cadastrada deverá ter um documento vinculado. Este documento poderá ser CPF ou CNPJ	RF01	Must
(RN02) Uma excursão pode ter vários locais de embarque. Caso o viajante faça uma reserva e adicione acompanhantes, estes acompanhantes poderão ser cadastrados com locais de embarque diferentes.	RF03, RF010	Must
(RN04) Pacotes só podem estar disponíveis para vendas até a data de inicio da excursão..	RF04, RF09, RF010	Must
(RN05) O gestor da agência poderá definir um período de tempo máximo em que uma nova solicitação de reserva deve ser paga. Caso esse limite seja extrapolado, a solicitação de reserva deverá ser cancelada..	RF04, RF07	Must
(RN06) O viajante poderá cancelar a reserva de uma excursão que não foi concluída ou não esta em andamento a qualquer momento. Mas o sistema ainda não se responsabiliza pelo reembolso do pagamento.	RF011	Must
(RN07) O gestor da agência poderá cancelar a reserva de uma excursão que não foi concluída ou não esta em andamento a qualquer momento. Mas o sistema ainda não se responsabiliza pelo reembolso do pagamento.	RF011, RF06	Must
(RN08) Ao buscar por novas excursões, o viajante só poderá visualizar excursões que estejam disponíveis e que tenham pacotes a venda e disponíveis	RF09, RF04, RF02	Must

(RN09) O gestor da agência poderá tornar uma excursão ou um pacote indisponíveis a qualquer momento, porém isso não vai afetar a reservas que já foram realizadas.	RF09, RF04, RF02	Must
(RN10) O gestor da agência pode alterar o valor de um pacote a qualquer momento, mas isso não deve afetar a reservas que já foram solicitadas ou que já foram pagas.	RF09, RF04, RF02	Must
(RN11) Não deve ser permitido criar novas solicitações de reservas para pacotes que já tenham a soma do número de reservas em estado de solicitada, aguardando pagamento e confirmada igual a quantidade disponível do pacote.	RF09, RF04, RF02	Must

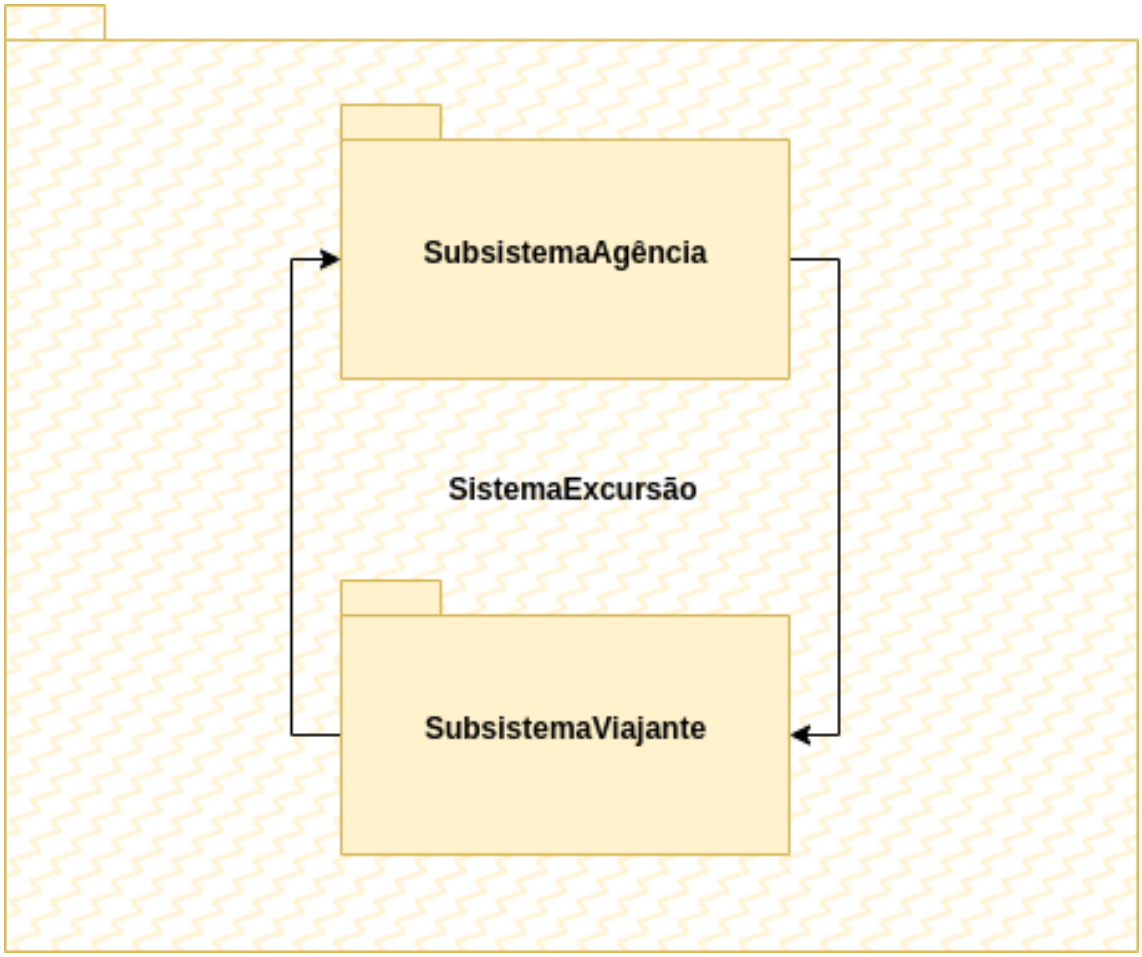
Fonte: Criação própria

3.3 MODELAGEM DO SISTEMA

Para auxiliar no desenvolvimento do sistema, houve uma etapa de modelagem onde foram utilizadas diversas técnicas de projeto de sistemas e engenharia de software. Alguns diagramas foram criados com intuito de promover tanto uma visão mais macro quanto uma visão mais detalhada da arquitetura do sistema e do seu funcionamento.

Através da análise dos requisitos e das regras de negócio, foi observado que é possível dividir o sistema em dois subsistemas focando na separação do domínio do sistema em subdomínios, como mostra a figura 17. Esses dois subsistemas dependem um do ofutro, e juntos formam o domínio do sistema de excursões. O subsistema de agência, foca em todo o domínio de negócio voltado para o gestor da gerência de excursões, e é onde acontecerá todo o controle de excursões, pacotes, embarques e reservas, sendo assim denominado SubsistemaAgência. Já o subsistema de reservas foca em todo o domínio de negócio voltado para o viajante, e é onde ele poderá buscar por excursões e criar reservas de pacotes, sendo assim denominado SubsistemaViajante.

Figura 17 – Subsistemas

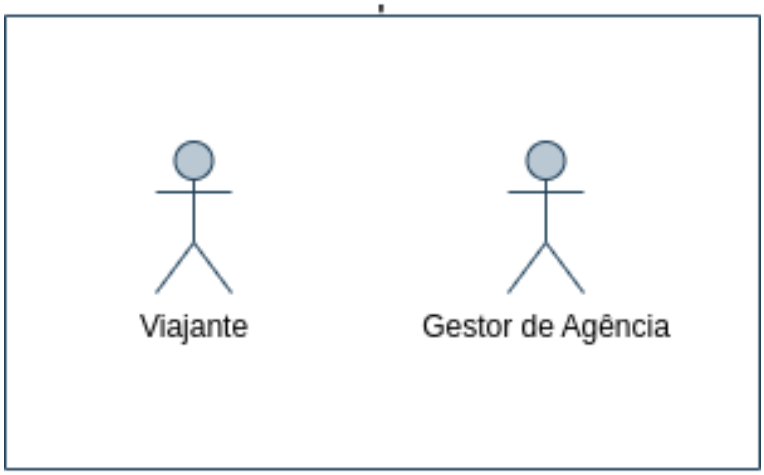


Fonte: Criação própria

3.3.1 Diagrama de Casos de Uso

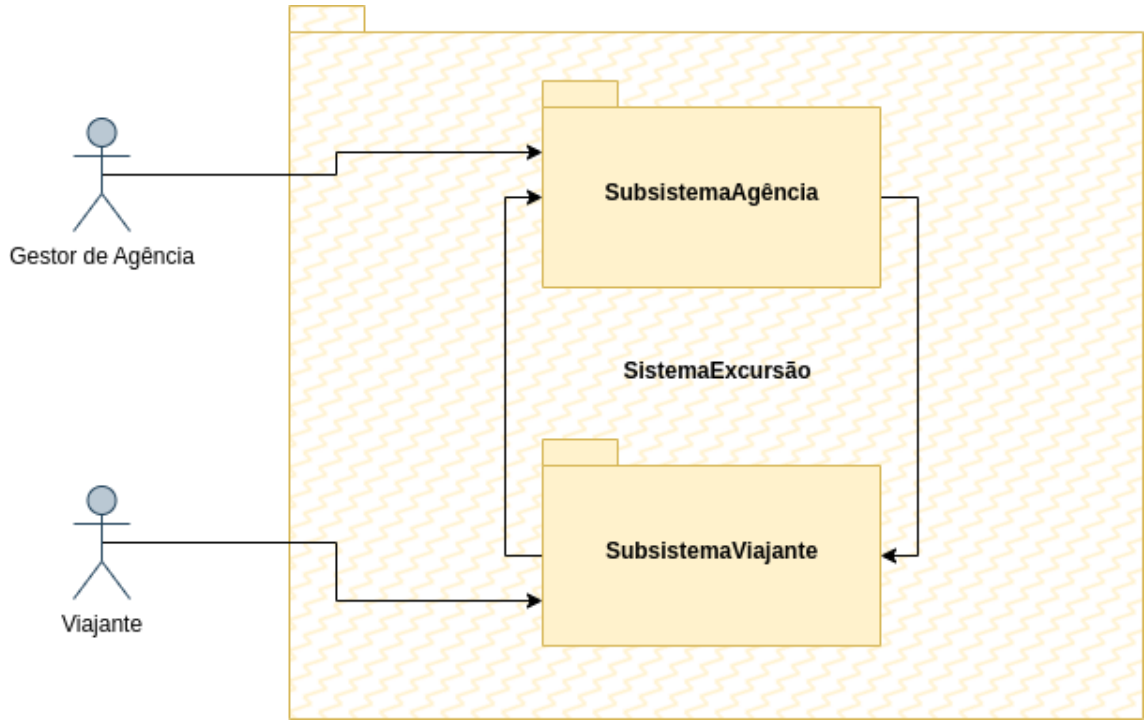
A partir da definição dos subsistemas, foi possível definir quais serão os atores que irão interagir com cada um deles. Como mostra a figura 18, na primeira fase do sistema deverão existir somente 2 atores, sendo eles o Gestor da Agência e o Viajante. O Gestor da Agência interage somente com o SubsistemaAgência e o Viajante somente com o SubsistemaViajante. A figura 19 mostra a interação dos atores com seus respectivos subsistemas.

Figura 18 – Atores



Fonte: Criação própria

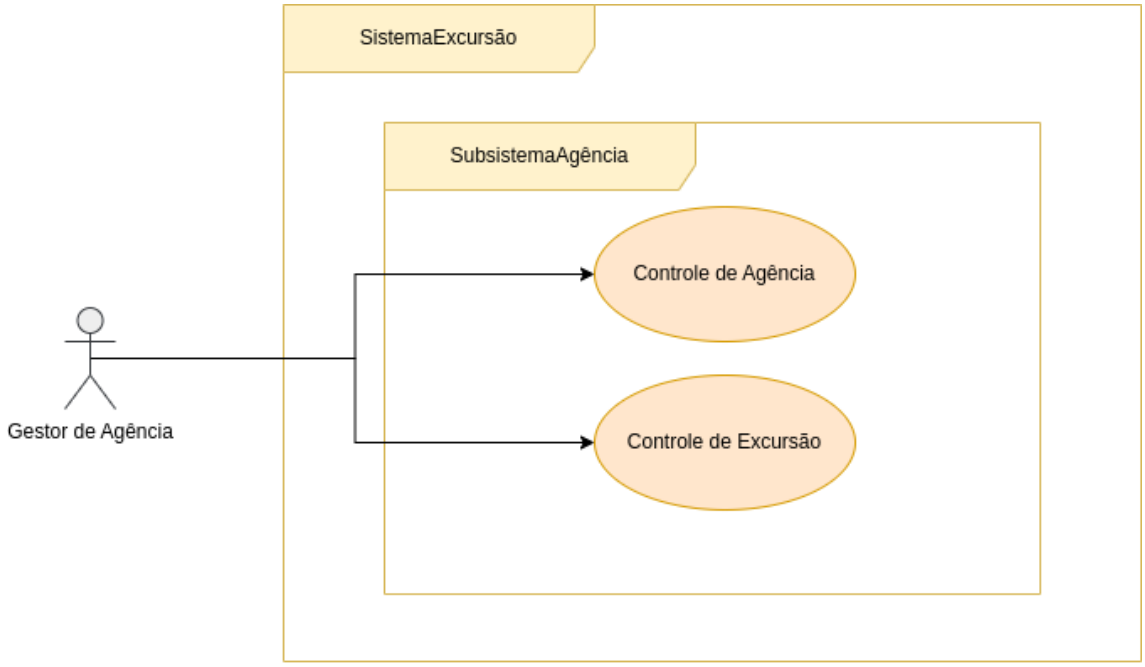
Figura 19 – Interação dos Atores com os Subsistemas



Fonte: Criação própria

Quando o Gestor da Agência interage com o subsistema de Agência, existem dois casos de uso que podem ser considerados os mais importantes e que se sobressaem em relação aos demais, sendo esses o caso de uso de Controle de Agência e o Controle de Excursão, como demonstrado na figura 20.

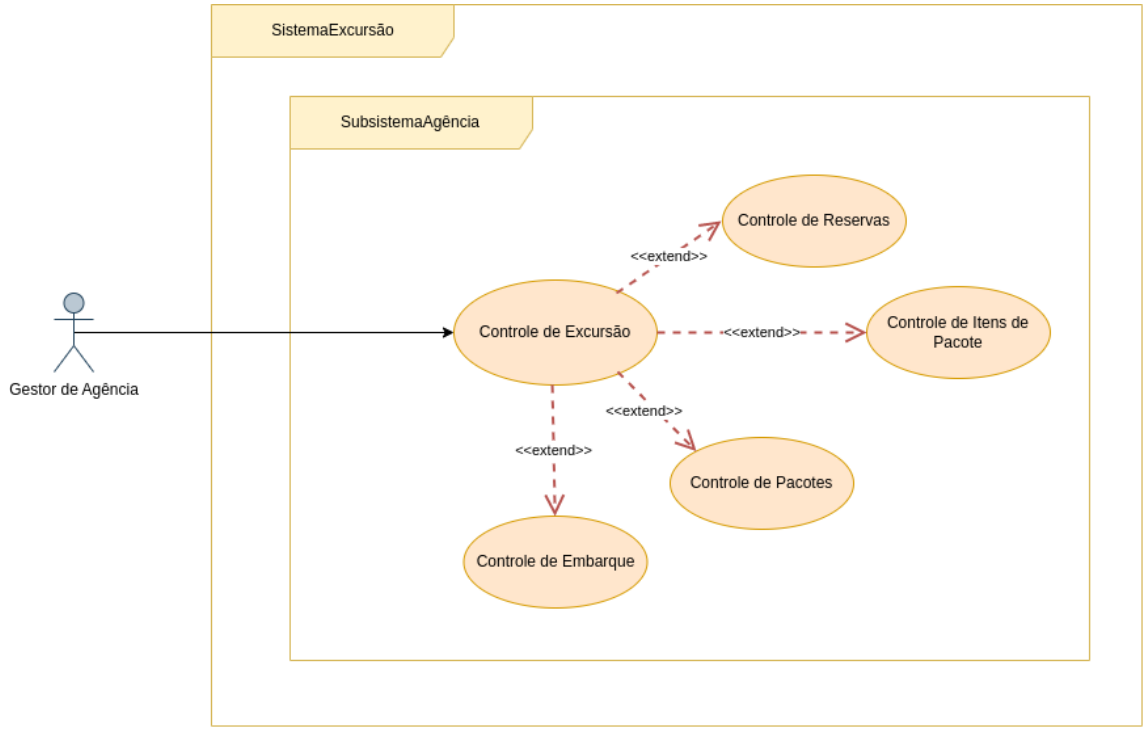
Figura 20 – Controle de Agência e Controle de Excursão



Fonte: Criação própria

O caso de uso Controle de Excursão também pode ser visto como porta de entrada para outros casos de uso. Quando um gestor faz o controle de uma excursão ele pode, ou não, estar interagindo com os casos de uso de Controle de Embarque, Controle de Reserva, Controle de Pacotes e Controle de Itens de Pacote. Essa relação entre os casos de uso é representado pelo «extend», como mostra a figura 21.

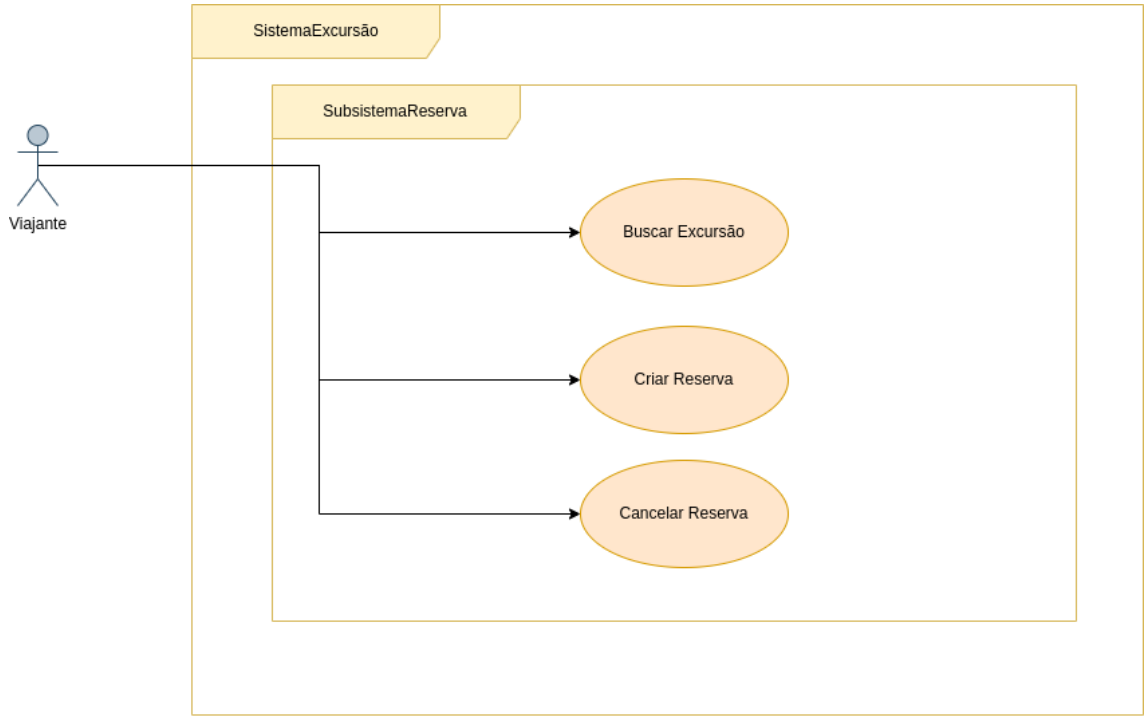
Figura 21 – Relacionamentos de Controle de Excursão



Fonte: Criação própria

Os principais casos de uso relacionados com o Viajante são o de buscar excursão, criar reserva e cancelar reserva, como mostra a figura 22.

Figura 22 – Casos de Uso do Viajante



Fonte: Criação própria

3.3.2 Diagrama de Arquitetura

A arquitetura do sistema foi pensada de uma forma que levasse ao desacoplamento dos subsistemas. Por isso duas aplicações *frontend* foram desenvolvidas, uma focando nos casos de uso do Viajante e outra nos casos de uso do Gestor de Agência.

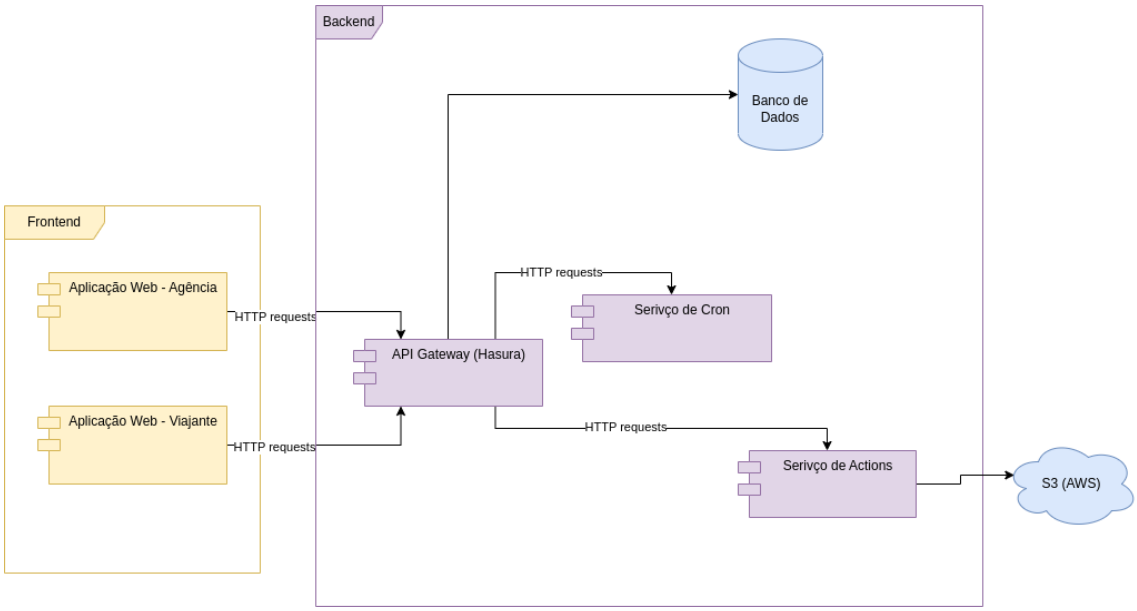
O *backend* do sistema foi modelado adotando uma arquitetura de micro-serviços e buscando utilizar da melhor forma os recursos disponibilizados pela ferramenta Hasura GraphQL Engine. Desta forma, todas as funcionalidades simples, que lidam somente com CRUD, ficam sob responsabilidade do Hasura, que também adota um papel de API *gateway*, se tornando um único ponto de entrada para os demais micro-serviços, que são o Serviço de Actions e o Serviço de Cron. O Hasura e os micro-serviços são executados em *containers* isolados. Cada micro-serviço foi desenvolvido para atuar de forma independente.

Para fazer o armazenamento dos arquivos de mídia, como fotos de capa de uma excursão, foi utilizado um serviço chamado *Simple Storage Service* (S3), que é um serviço de armazenamento da AWS (*Amazon Web Services*) que fornece escalabilidade, disponibilidade e segurança. Dessa forma, todos os arquivos de mídia ficam armazenados S3, e no banco de dados salvamos somente uma URL para esse arquivo.

Somente o Hasura se comunica diretamente com o banco de dados. Os demais serviços se comunicam com o Hasura quando precisam de fazer operações que envolvam a base de dados.

A figura 23 mostra o resultado final da arquitetura do sistema.

Figura 23 – Diagrama de Arquitetura



Fonte: Criação própria

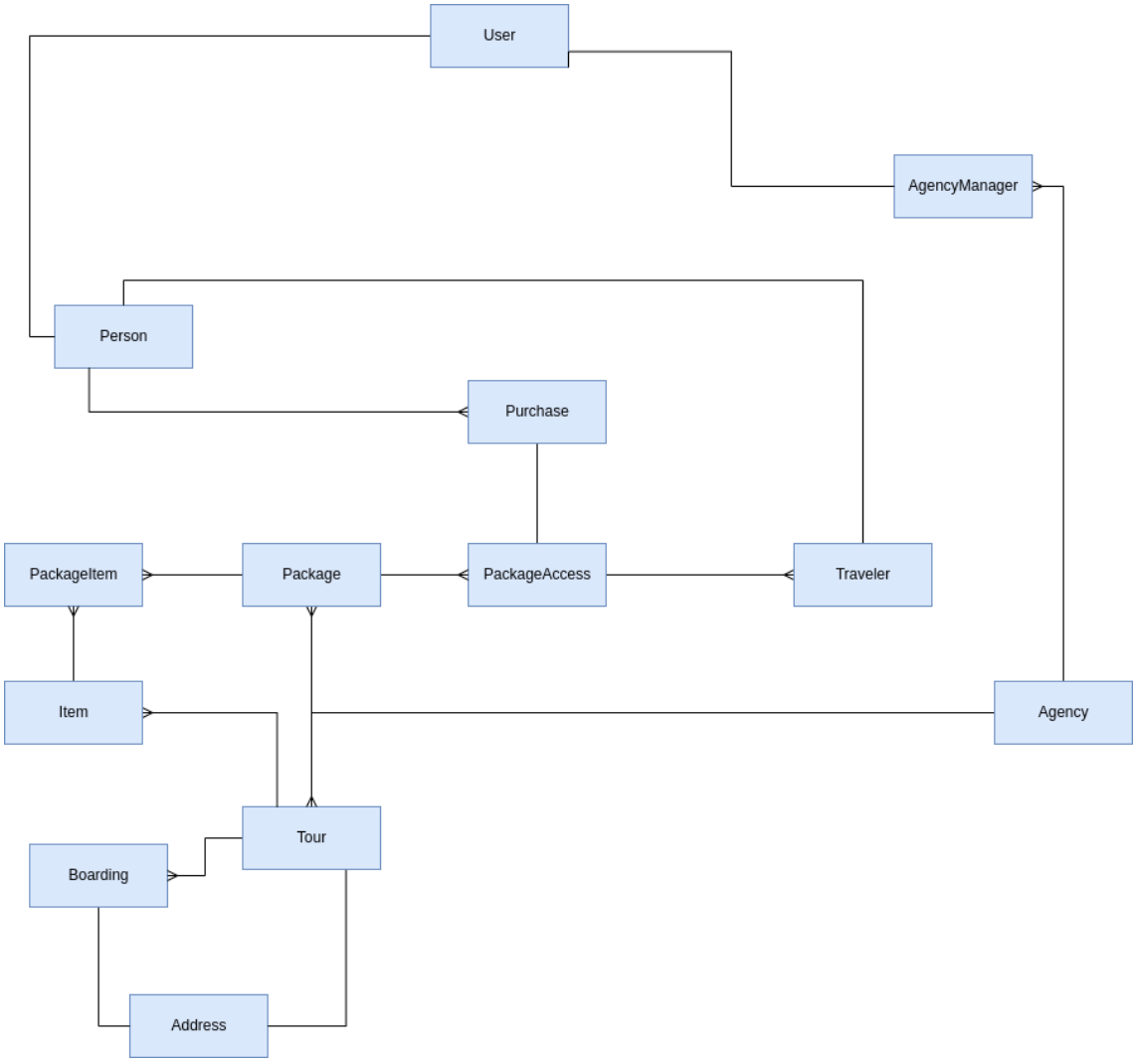
O hasura cria automaticamente uma API GraphQL que lida com CRUD, porém funcionalidades que necessitam de validações mais complexas precisam de tratamento especial. O Serviço de Actions foi criado para tratar tais casos, tendo a responsabilidade de lidar com as funcionalidades mais complexas. Essas funcionalidades serão explicadas em mais detalhes na seção que fala exclusivamente sobre o serviço de actions.

O Serviço de Cron também é um serviço que atua dando suporte ao Hasura, porém a sua responsabilidade é lidar com as rotinas automáticas, como o cancelamento automático de solicitações de reservas com data de pagamento expirada.

3.3.3 Diagrama de Domínio

Através da análise dos requisitos e das regras de negócio, foram levantadas as entidades que fariam parte do sistema. De forma mais macro, sem levar em consideração atributos e aspectos de implementação, foi possível chegar ao diagrama de domínio de negócio, representado pela figura 24.

Figura 24 – Diagrama de Domínio



Fonte: Criação própria

- **User:** É um usuário do sistema. Essa entidade possui um atributo chamado role, que representa a função deste usuário. As possíveis roles são *traveler* (viajante) e *manager* (gestor de agência).

- **Person:** É a entidade que representa uma pessoa. Essa pessoa pode ser relacionada com um usuário ou ser um viajante, que não é um usuário, mas foi adicionado como acompanhante durante a reserva de algum pacote.

- **Agency:** É uma agência de excursão.

- **AgencyManager:** É um gestor de uma agência. Toda agência possui um gestor, ele possui um usuário e é quem tem a permissão de cadastrar pacotes de viagem.

- **Tour:** É a entidade que representa uma excursão criada por uma agência

- **Boarding:** É a entidade que relaciona um embarque. Uma excursão pode ter mais do que um local de embarque.

- **Address:** É a entidade que relaciona um endereço. Um local de embarque possui um endereço, e uma excursão também possui um endereço de destino.

- **Package:** É um pacote que pertence a uma agência. Essa é a entidade que é colocada disponível para ser reservada. Uma agência pode ter diversos pacotes, com diversos preços e pode estar disponível para quantidades de pessoas variadas. Exemplo: Pacote para casal (duas pessoas) ou pacote individual (uma pessoa).

- **Item:** É um item que faz parte de uma excursão. Os itens devem ser adicionados em pacotes, que serão colocados a venda.

- **PackageItem:** É um item que está presente em um pacote da excursão.

- **Purchase:** É a entidade que representa uma reserva de um pacote. A reserva é realizada por uma pessoa.

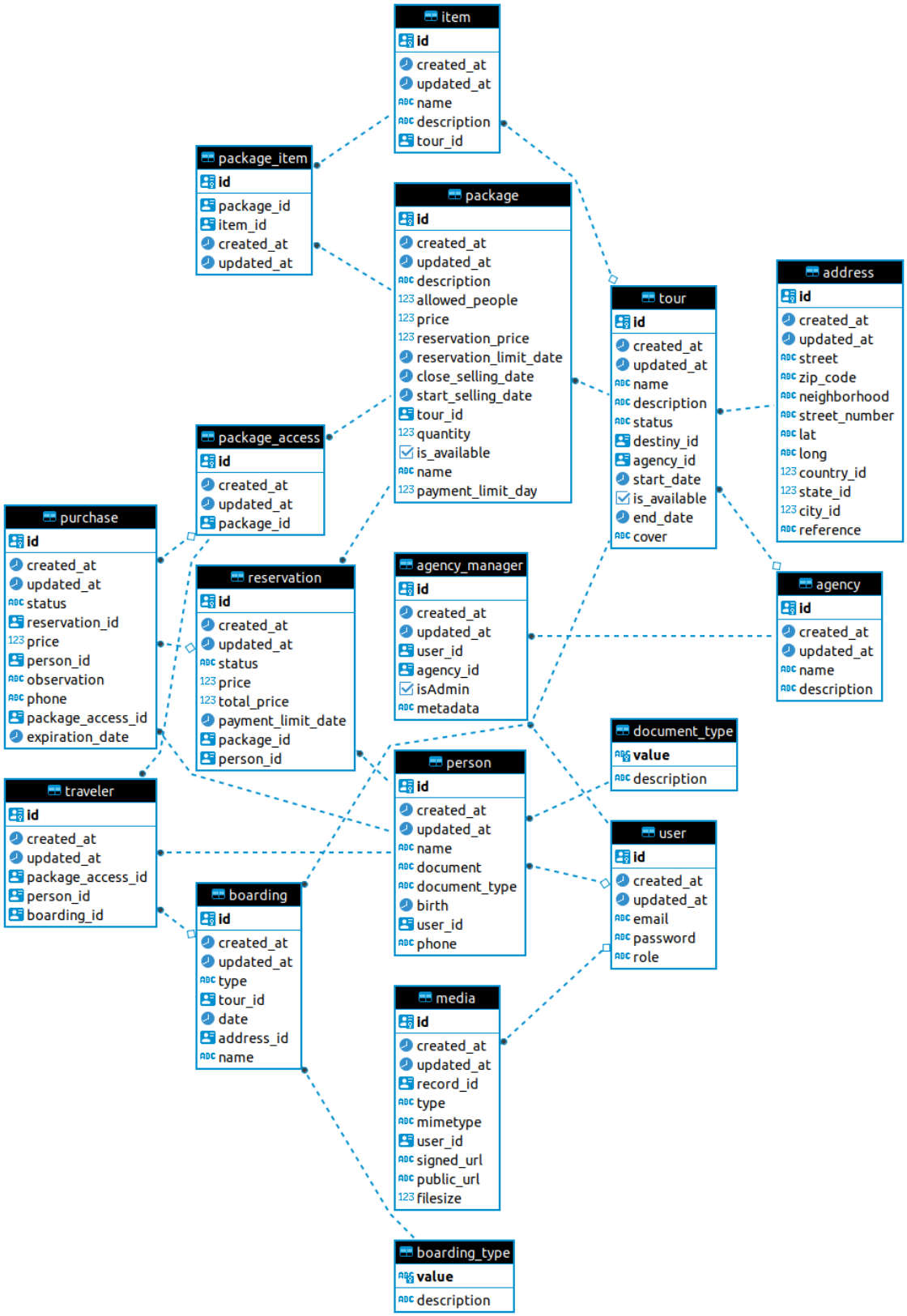
- **PackageAccess:** É o acesso a um pacote. Uma Purchase não se relaciona diretamente com um Package, o que ela cria é o acesso a um pacote para os viajantes. Um PackageAccess pode estar relacionado com vários viajantes, e isso quer dizer que os viajantes terão acesso aquele pacote em conjunto, vindo da mesma reserva, utilizando as mesmas configurações.

- **Traveler:** É um viajante que possui acesso a algum pacote. Ele pode ser a mesma pessoa que realizou a reserva ou algum acompanhante.

3.3.4 Diagrama de Entidades e Relacionamento

O diagrama de entidade e relacionamento mostra como foram modeladas as entidades do sistema no banco de dados.(Figura 25)

Figura 25 – Diagrame de Entidades e Relacionamento



Fonte: Criação própria

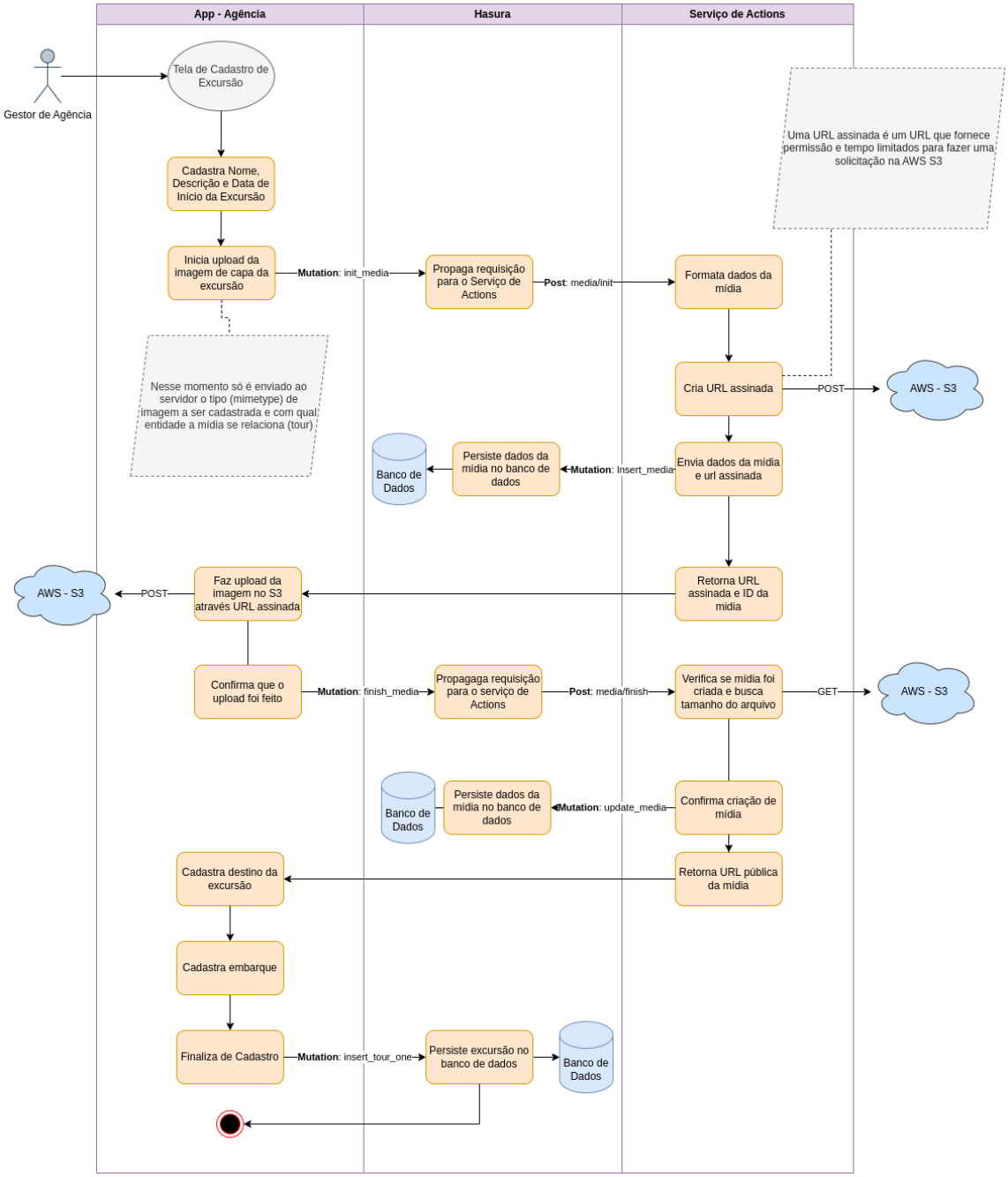
3.3.5 Diagramas de Fluxo

Para modelar os principais processos do sistema foram criados dois diagramas de fluxo. Estes diagramas tem o intuito de tornar mais claro o fluxo da informação dentro do sistema e podem ser utilizados para encontrar gargalos do sistema e pensar em melhorias futuras.

3.3.5.1 Diagrama de Fluxo: Criação de Excursão

Um dos fluxos mais complexos do sistema é o fluxo de criação de excursão, pois esse fluxo só é considerado como finalizado quando a excursão está preparada para receber solicitações de reserva, e isso só acontece após a criação e liberação de algum pacote de viagem para a excursão. A figura 26 mostra o início deste fluxo.

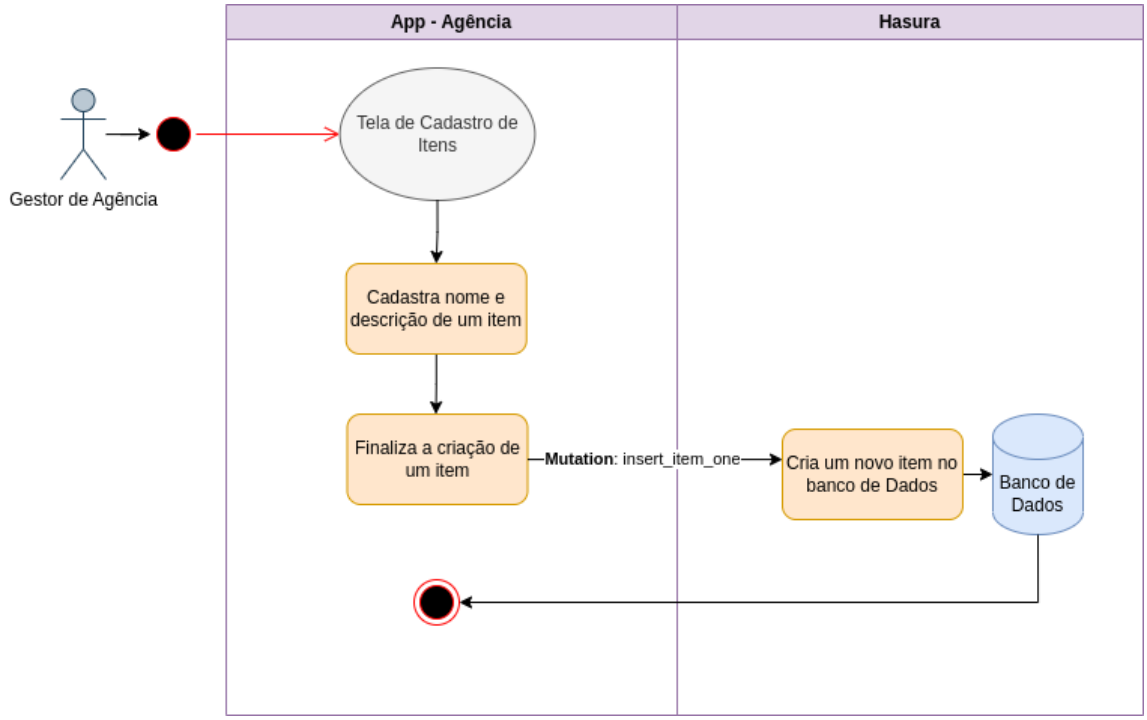
Figura 26 – Fluxo: Criação de Excursão - Parte 1



Fonte: Criação própria

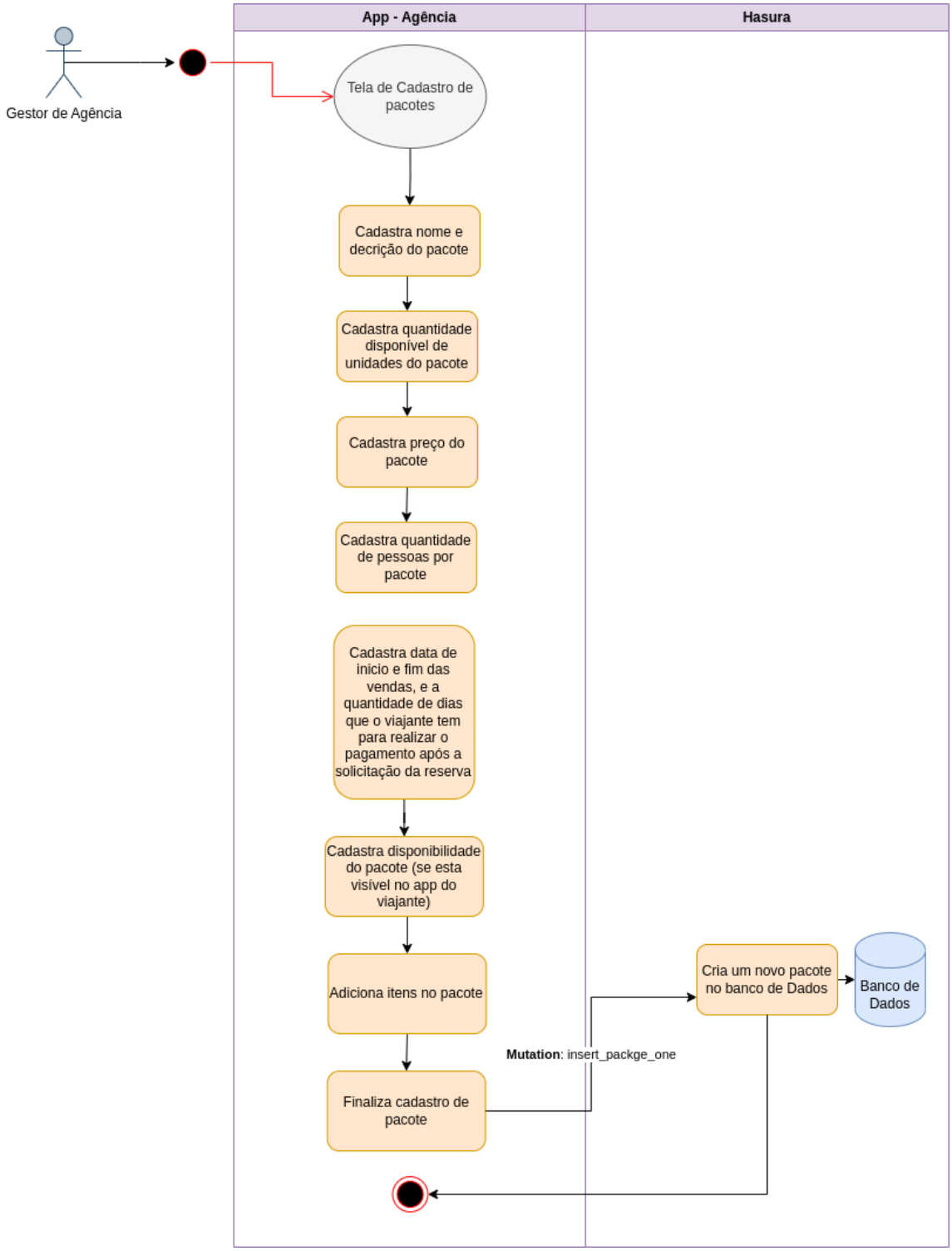
Após a sua criação, a excursão ainda não está disponível para reservas. Para isso, o gestor da agência precisa passar pelo fluxo de criação de pacotes. Os pacotes podem ser cadastrados com itens inclusos ou sem itens inclusos. A figura 27 demonstra o fluxo de cadastro de itens e a figura 28 o fluxo de cadastro de pacotes.

Figura 27 – Fluxo: Cadastro de Item



Fonte: Criação própria

Figura 28 – Fluxo: Cadastro de Pacote

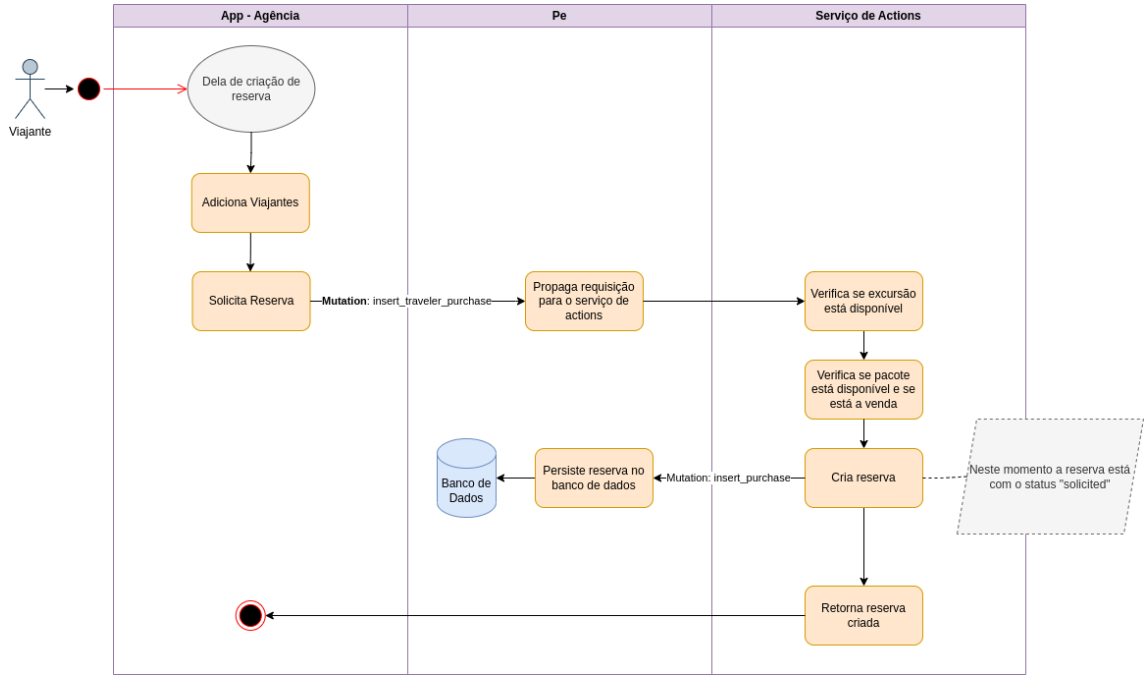


Fonte: Criação própria

3.3.5.2 Diagrama de Fluxo: Reserva de Pacote

O fluxo de reserva de pacote se inicia a partir do momento em que o viajante cria uma nova solicitação de reserva pelo app de reservas. A figura 29 mostra como é realizado esse processo.

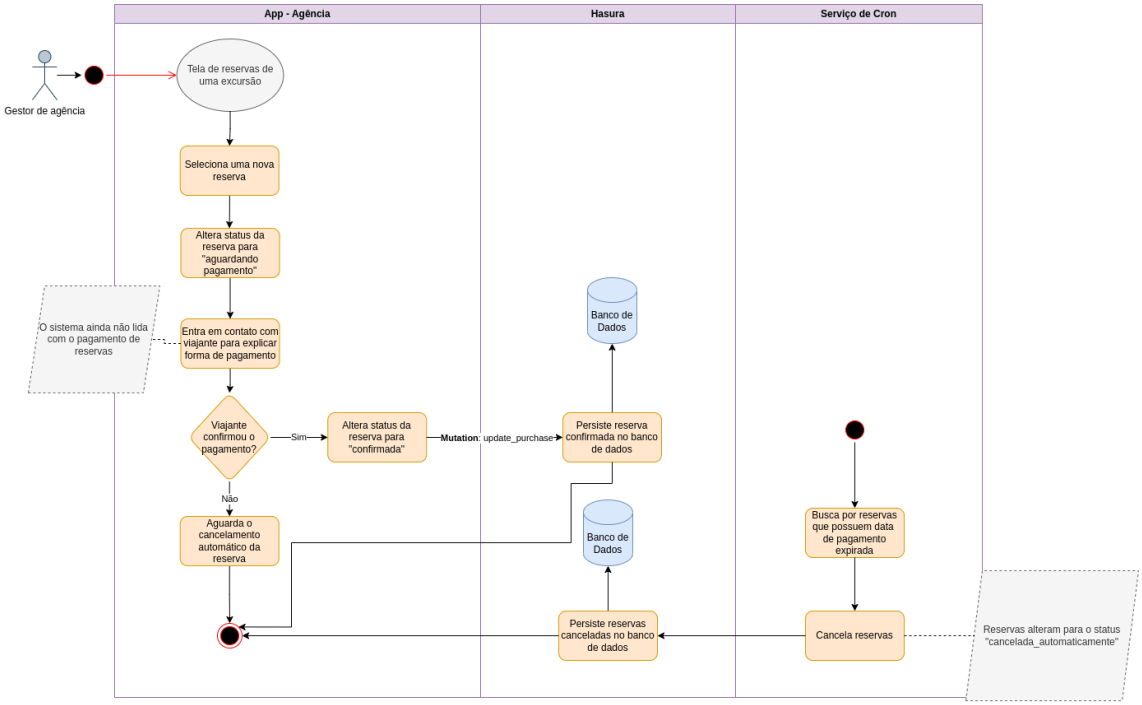
Figura 29 – Fluxo: Solicitação de Reserva



Fonte: Criação própria

Após o viajante solicitar uma nova reserva, o gestor da agência precisa confirmá-la. Como mostra na figura 30, o gestor acessa a tela de reservas de uma excursão e seleciona uma reserva solicitada por algum viajante. A partir dai ele pode alterar o estado dela para *waiting payment* (aguardando pagamento) e iniciar o contato com o viajante da forma que preferir. Caso o gestor não confirme a reserva até a data limite de pagamento, o serviço de cron irá cancelar a solicitação automaticamente.

Figura 30 – Fluxo: Confirmação de Reserva



Fonte: Criação própria

3.4 PROCESSO DE DESENVOLVIMENTO

Durante o processo de desenvolvimento do sistema foram utilizadas diversas ferramentas. Esta seção vai apresentá-las e explicar como foram utilizadas no contexto de desenvolvimento *backend*, *frontend* e aplicação móvel.

3.4.1 Ferramentas

Esta seção mostra quais foram as ferramentas utilizadas no desenvolvimento do sistema, e explica brevemente a motivação para a escolha de cada uma delas.

- **Hasura GraphQL Engine:** Ferramenta que se tornou o ponto central do *backend*. Foi escolhida por ser uma ferramenta que acelera o processo de desenvolvimento, com a criação automática de uma API GraphQL praticamente completa e por lidar com permissionamento.

- **Node:** Foi utilizado para o desenvolvimento dos serviços de Actions e Cron. Foi escolhido principalmente pela facilidade de desenvolvimento e por contar com diversas bibliotecas que ajudam no desenvolvimento.

- **Vue JS:** *Framework* escolhido para o desenvolvimento do *frontend*, principalmente por ser focado na criação de *Single Page Applications* e por lidar com componentização.

- **CORE UI:** Ferramenta utilizada para facilitar o desenvolvimento do frontend. As duas aplicações *frontend* foram desenvolvidas com *templates* do CORE UI. O Core UI não é só uma mistura de componentes de interface, é um dos únicos *templates* de painel de administração de código aberto que é aprovado e apoiado por profissionais do mercado. (COREUI, 2022)

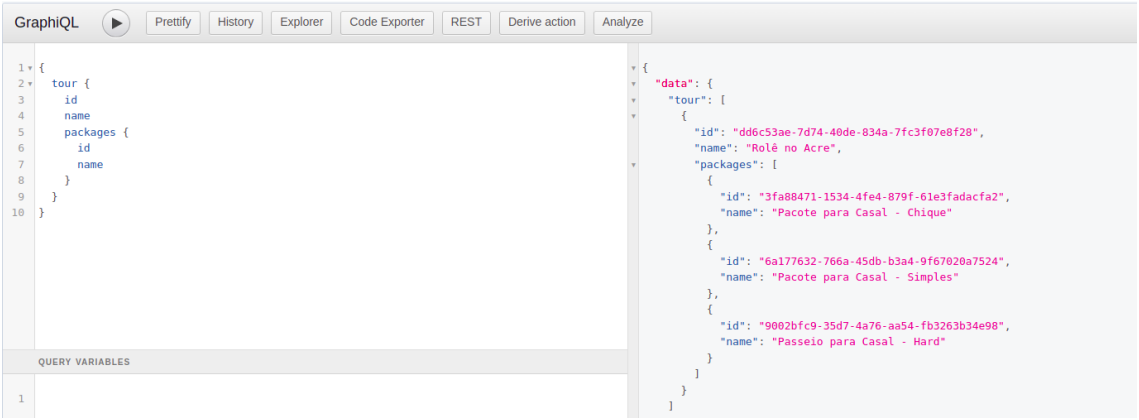
3.4.2 Backend

O *backend* do sistema é composto pela API GraphQL gerada pelo Hasura e pelos serviços auxiliares. Nesta seção será explicado melhor como eles foram implementados e como funciona essa integração.

3.4.2.1 API GraphQL

O Hasura é o responsável por gerar a API GraphQL. A figura 31 mostra a interface disponibilizada pelo Hasura para que seja possível fazer consultas a sua API. Do lado esquerdo temos a requisição realizada e do lado direito a reposta.

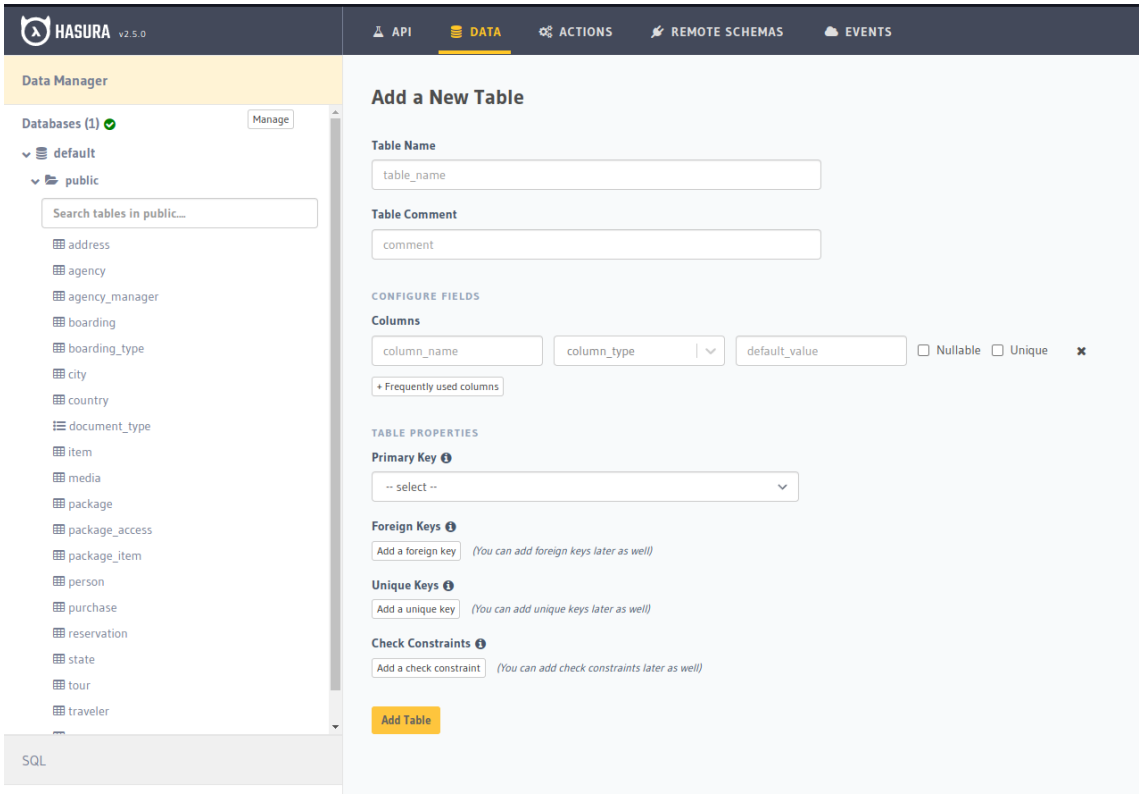
Figura 31 – API GraphQL



Fonte: Criação própria

Através da plataforma do Hasura foram criadas todas as entidades que dão origem as tabelas de banco de dados, para que a partir dai a ferramenta criasse a API GraphQL que permite a criação, deleção, atualização e atualização de cada uma dessas entidades. A Figura 32 mostra a plataforma e a criação de uma entidade.

Figura 32 – Hasura: Criação de Entidade



Fonte: Criação própria

Utilizando do recurso de permissionamento do Hasura, foram criadas permissões de forma personalizada e baseada em variáveis de sessão. Por exemplo, na figura 33, mostra a definição de uma permissão através do Hasura. Nessa permissão nós estamos restringindo a leitura da entidade de agência para os usuários com função de gestor de agência, de forma que eles só consigam ler agências cujo ID seja igual a variável de sessão *hasura-agency-id*, que deve estar presente no *token* JWT do usuário.

Figura 33 – Hasura: Permissão Gestor de Agência

Fonte: Criação própria

Para lidar com casos de uso mais complexos, foi utilizado o recurso de actions do Hasura, que são maneiras de estender o schema GraphQL gerado pela ferramenta, criando mutations e queries personalizadas que, ao invés de passarem pelo Hasura e se comunicarem diretamente com o banco de dados, são enviadas para um serviço serviço auxiliar, que foi desenvolvido em Node e é executado separadamente do Hasura.

3.4.2.2 Serviço de Actions

Como foi dito na seção que fala sobre a arquitetura do sistema, o Hasura cria automaticamente uma API GraphQL com as funcionalidades de CRUD. Porém, existem algumas funcionalidades que precisam de um processamento mais complexo, por isso é necessário o Serviço de Actions para atuar como um serviço auxiliar.

Quando lidamos com a criação de um novo usuário, por exemplo, existe uma estratégia de criptografia das informações do usuário a ser feita antes de persistir os dados no banco. Por isso, a criação de usuário passa pelo serviço de actions.

Outra funcionalidade que passa pelo serviço de actions é a criação de arquivos de mídia (para a foto de capa de uma excursão, por exemplo). Existe uma action que antes de persistir os dados da nova mídia no banco, faz alguns processamentos no serviço S3 da AWS, que é onde as imagens são salvas.

O serviço de Actions foi totalmente implementado em Node com Typescript. Por ser um

serviço simples e com poucas funcionalidades, foi adotada uma arquitetura interna simples.

A Figura 34 mostra qual foi a arquitetura interna adotada para o serviço.

Quando a aplicação *frontend* faz uma requisição para o Hasura que é redirecionada para o serviço de actions, primeiramente a requisição passa pelos *middlewares*, que irão fazer algumas verificações inciais, como verificar as variáveis de sessão.

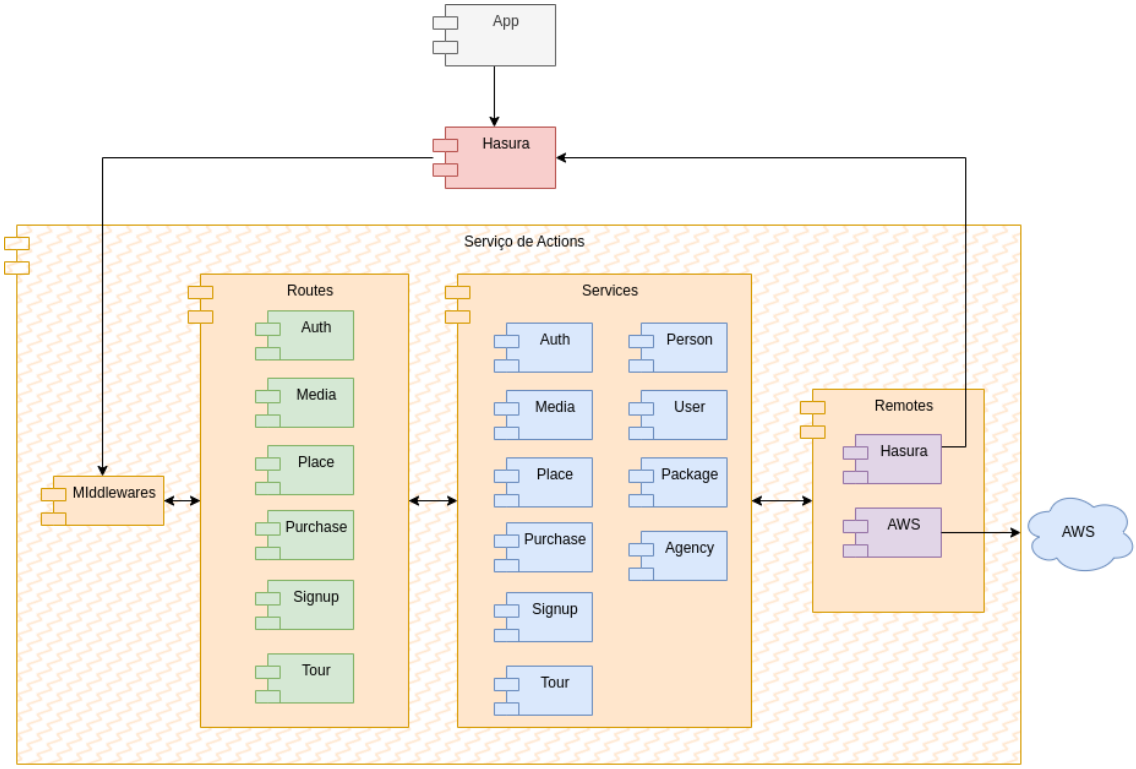
Cada action configurada no hasura, envia uma requisição para uma rota especifica do serviço de Actions. A action de login, por exemplo, envia uma notificação para a rota /auth/login. Por isso, como mostra a figura 34, após passar pelos middlewares, requisições passam por roteadores especializados para cada tipo de requisição. No caso do login, o roteador especializado é o Auth, que vai definir como vai ser processada a requisição realizada na rota /auth/login.

Os serviços são os responsáveis pela implementação das regras de negócio do sistema. Os roteadores se comunicam com o serviços para que eles façam o processamento das lógicas de negócio relacionadas com as entidades do sistema.

Por ultimo, existe a comunicação com os sistemas e serviços remotos, que são externos ao sistema. No momento atual só há a comunicação com o próprio Hasura e com a AWS.

O Serviço de CRON utiliza a mesma arquitetura, por isso não foi criada uma seção para falar sobre ele.

Figura 34 – Serviço de Actions



Fonte: Criação própria

A Figura 35 mostra um trecho de código da aplicação. No exemplo é apresentado o *service* de person no serviço de actions. Esse serviço só tem um método, responsável pela criação da entidade Person. Esse método se utiliza do *remote* hasura para se comunicar com o Hasura e inserir um novo registro na tabela Person.

Figura 35 – Person Service

```
import { Person as PersonModel } from '../models'
import { HasuraService } from '../remotes/hasura'
import { INSERT_PERSON } from '../remotes/hasura/graphql'

export class Person {
    static async create(person: PersonModel): Promise<PersonModel> {
        const response = await HasuraService.mutate({
            mutation: INSERT_PERSON,
            variables: person
        })

        return response.data.insert_person_one as PersonModel
    }
}
```

Fonte: Criação própria

3.4.2.3 Autenticação

O Hasura consegue lidar com autorização através de variaveis de sessão em tokens JWT, porém não sabe lidar com autenticação. Por isso, essa funcionalidade ficou por conta do serviço de actions.

Os usuários do sistema se autenticam por email e senha através de uma action de login. Essa action valida o email e a senha do usuário, levando em consideração que a senha que está persistida no banco de dados é criptografada, e retorna um token JWT.

A Figura 36 mostra como é criado o token JWT dentro da aplicação. O token deve ter as seguintes informações:

- **x-hasura-allowed-roles:** Todas as funções que aquele usuário pode exercer (viajante ou gestor de agência).
- **x-hasura-default-role:** Função padrão daquele viajante (viajante ou gestor de agência).
- **x-hasura-user-id:** ID do usuário
- **x-hasura-agency-id:** ID da agência do usuário (se o usuário for um gestor de agência).

Figura 36 – Criação de Token JWT

```
public static async generateJwtToken(user: UserModel): Promise<string> {
  const { id, role } = user

  const agencyId = user.agency_manager?.agency?.id ? user.agency_manager?.agency?.id : null

  const userTokenData = {
    user_id: id,
    hasura: pickBy({
      'x-hasura-allowed-roles': [role],
      'x-hasura-default-role': role,
      'x-hasura-user-id': user.id?.toString(),
      'x-hasura-agency-id': agencyId
    }),
  }

  return jwt.sign({ uuid: user.id, data: userTokenData }, privateKeyBuffer, {
    algorithm: 'RS256',
    // expiresIn: JWT_EXPIRE || 86400,
    keyid: JWT_KEY_ID,
  })
}
```

Fonte: Criação própria

3.4.3 Frontend

Todo o *frontend* da aplicação foi desenvolvido com Vue.JS. Uma das estratégias adotadas foi utilizar o máximo possível dos componentes prontos do Core Ui, para facilitar no desenvolvimento e manter uma identidade visual.

Para lidar com segurança, após o login o *token* JWT do usuário é salvo no *local storage* do navegador, e é enviado no *header* das requisições HTTP que necessitam de autenticação.

4 RESULTADOS

Neste capítulo serão apresentados os resultados obtidos após o processo de desenvolvimento do sistema. De modo geral, os resultados serão divididos em duas seções, uma para apresentar os resultados do sistema da agência e outra para os resultados do sistema do viajante.

4.1 SISTEMA DA AGÊNCIA

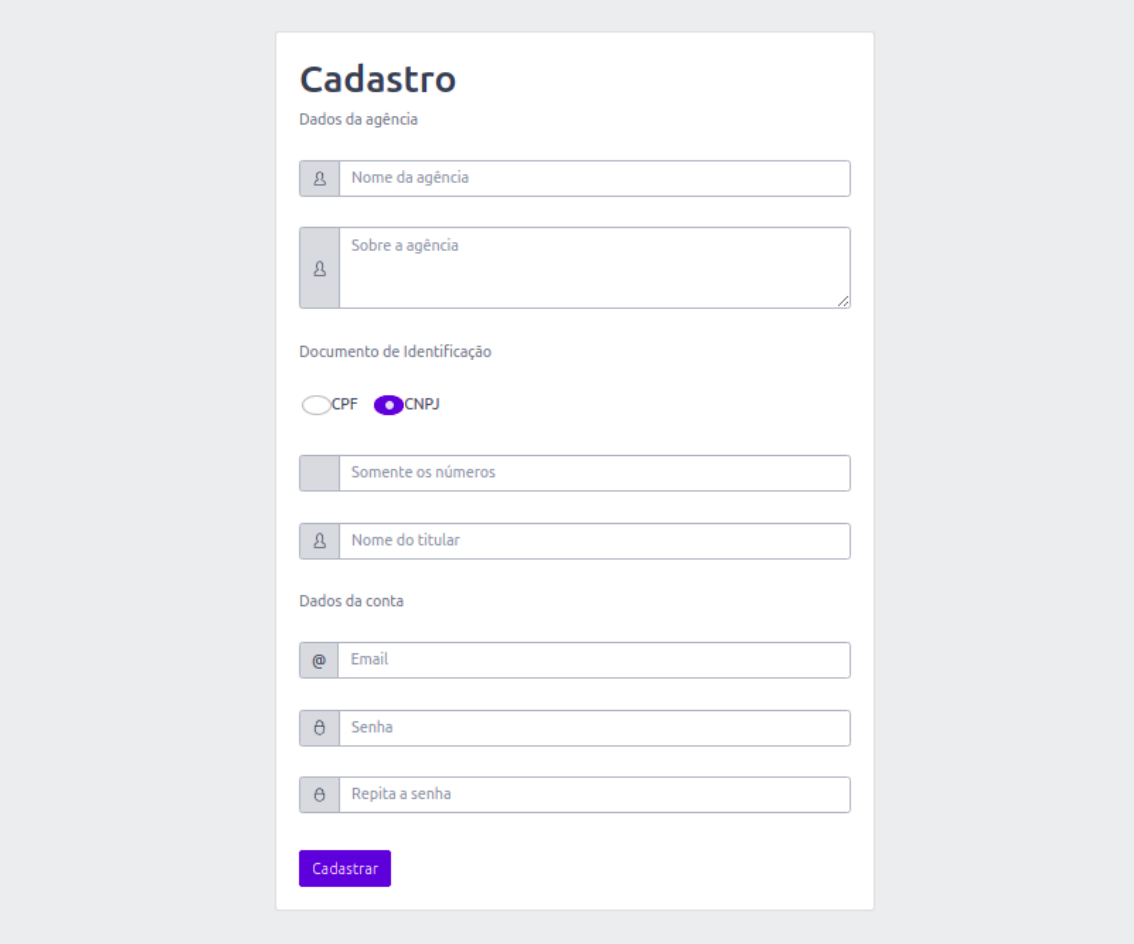
Nessa seção serão apresentados os resultados obtidos no desenvolvimento do sistema da agência.

4.1.1 Cadastro de Agência

A primeira coisa que um gestor de agência precisa fazer para ter acesso ao sistema de gestão de agência é realizar o cadastro da sua agência. Como mostra a figura 37, para realizar o cadastro é necessário o registro de algumas informações a respeito da agência, como o noma da agência e uma breve descrição. Também é necessário o registro de algum documento de idêntificação, podendo ser um CPF ou um CNPJ.

O gestor da agência também precisa criar uma senha para autênticação.

Figura 37 – Tela de Cadastro

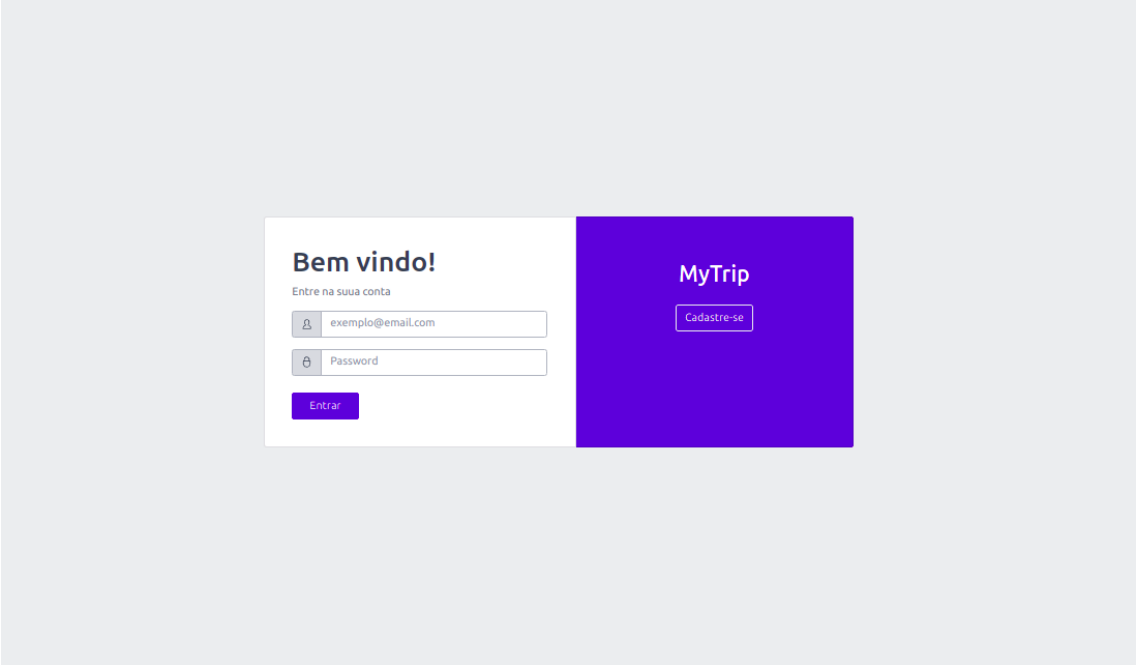


Fonte: Criação própria

4.1.2 Login

Após a finalização do cadastro, o gestor da agência ja vai possuir um usuário no sistema, por isso ele sera redirecionado para a tela de login, onde deve preencher o seu email e a senha para acessar o sistema, como mostra a figura 38

Figura 38 – Tela de Login

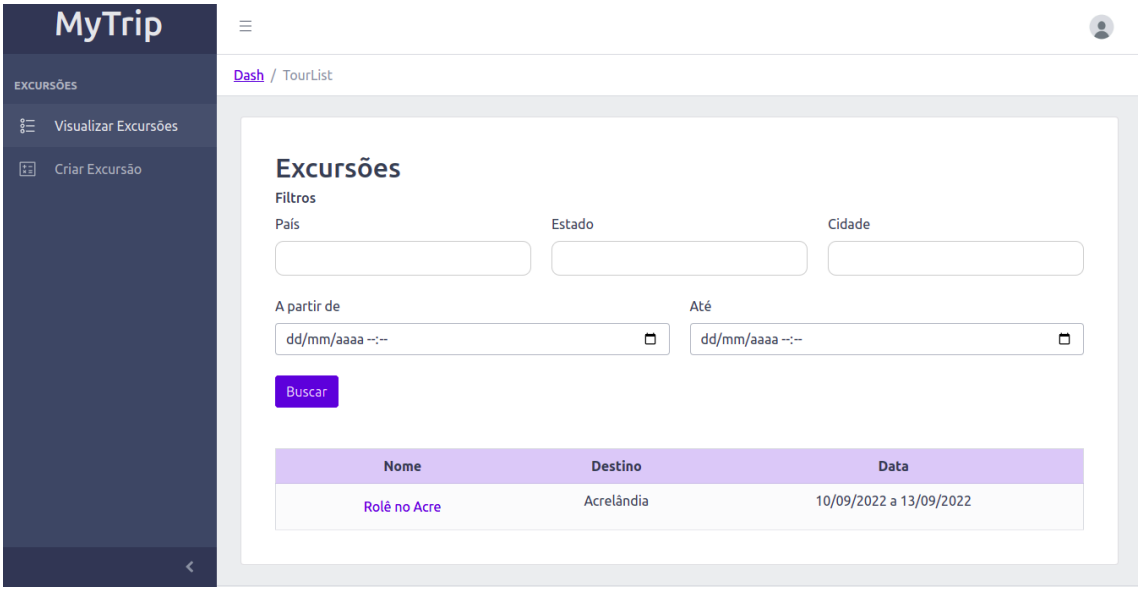


Fonte: Criação própria

4.1.3 Listagem de Excursões

Após o login, o gestor da agência será redirecionado para a tela principal do sistema, que é a tela de listagem de excursão (figura 40). Nessa tela o gestor poderá ver todas as excursões que já foram criadas pela agência, e filtrar por destino e data.

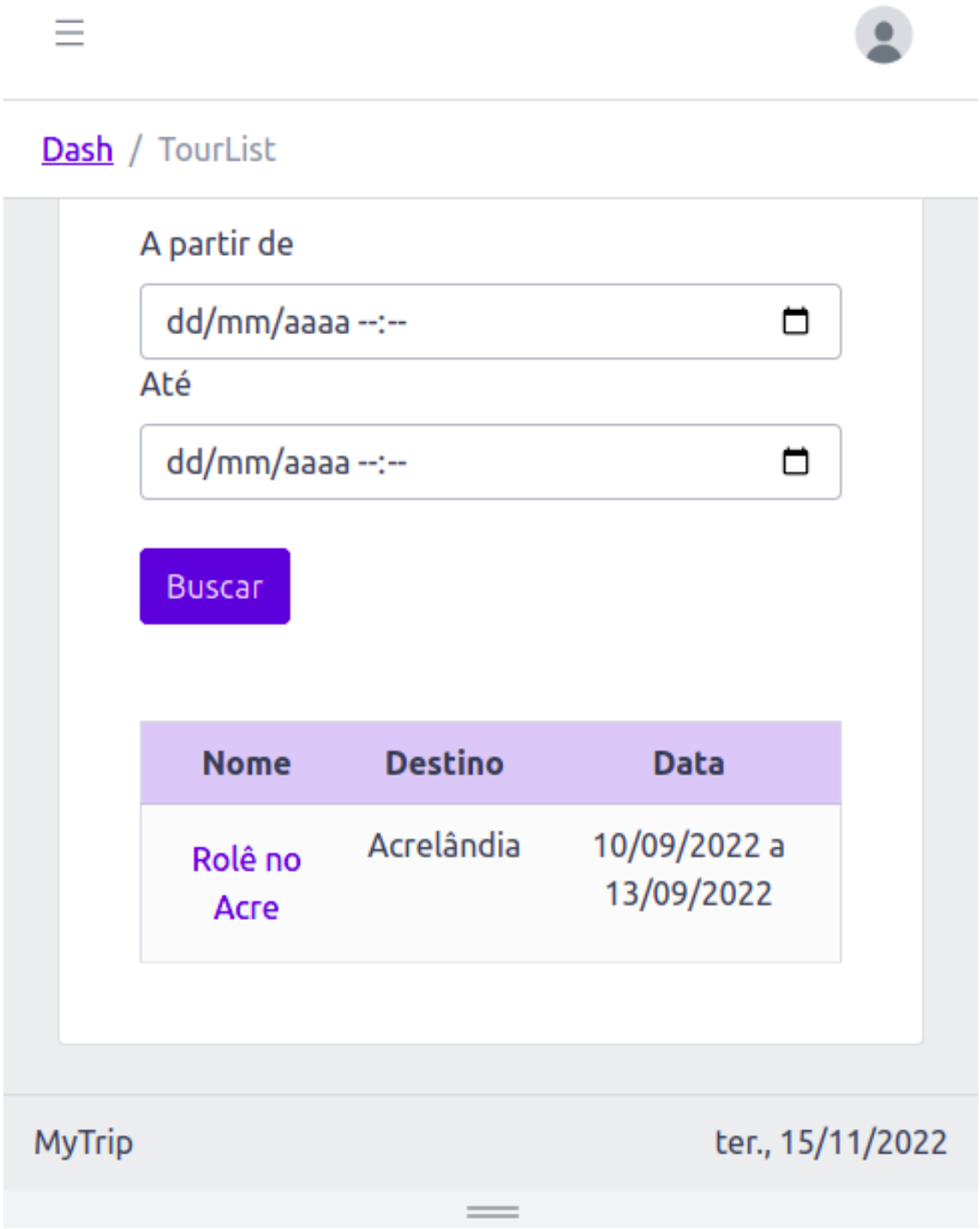
Figura 39 – Tela de Listagem de Excursões



Fonte: Criação própria

A figura 40 mostra como a tela de listagem de excursão fica quando acessada por um dispositivo móvel.

Figura 40 – Listagem de Excursões - Mobile



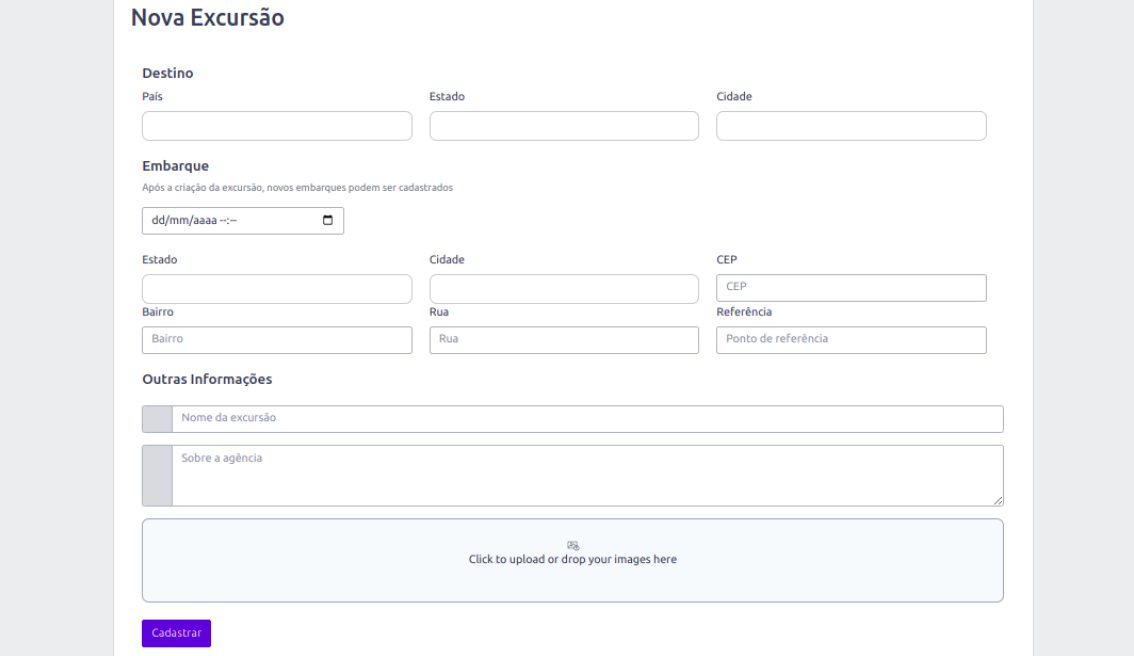
Fonte: Criação própria

A partir dessa tela o gestor da agência pode clicar no nome de uma das excursões para acessar os detalhes dessa excursão (figura 42), ou pode clicar no "Criar Excursão"no menu lateral, para fazer o cadastro de uma nova excursão (figura 41).

4.1.4 Criação de Reserva

Ao clicar em "Criar Excursão", o gestor tem acesso a tela de cadastro de uma nova excursão, como mostra figura 41. Nesta tela o gestor da agência deverá preencher o nome, embarque e destino da excursão, além de adicionar uma imagem para ser a sua capa.

Figura 41 – Cadastro de Excursão

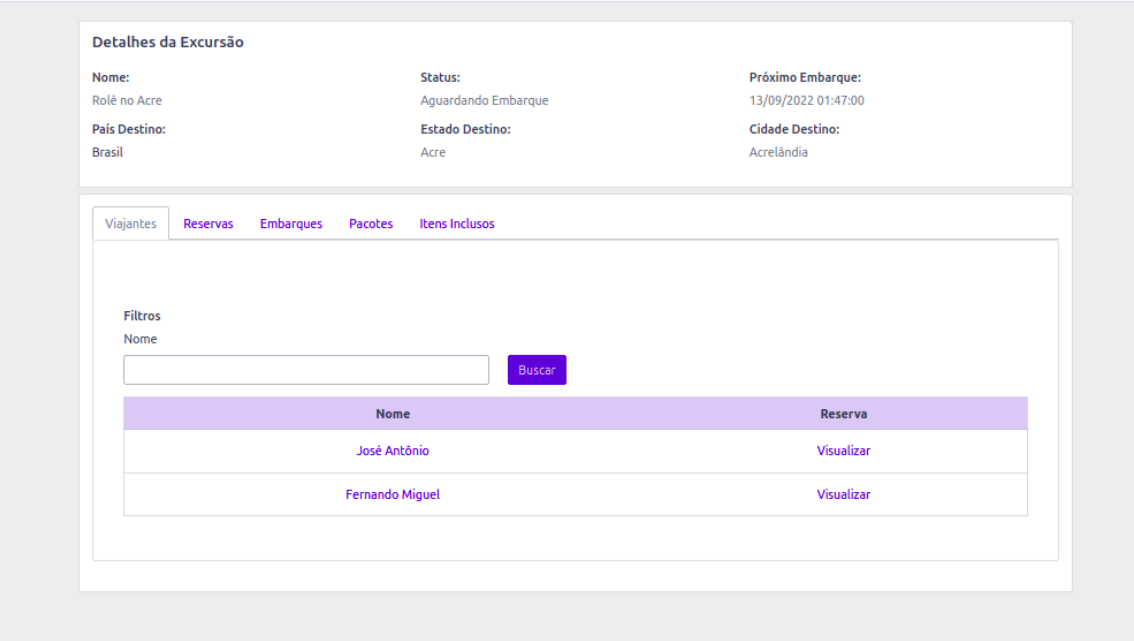


Fonte: Criação própria

4.1.5 Detalhes de Excursão

Na tela de detalhes de uma excursão, o gestor da agência terá acesso a todas as informações de uma excursão e poderá navegar por diversas seções com mais detalhes sobre essa excursão. As seções que podem ser acessadas são as de viajantes, reservas, embarques, pacotes e intens inclusos, como mostra a figura 42.

Figura 42 – Detalhes de Excursão

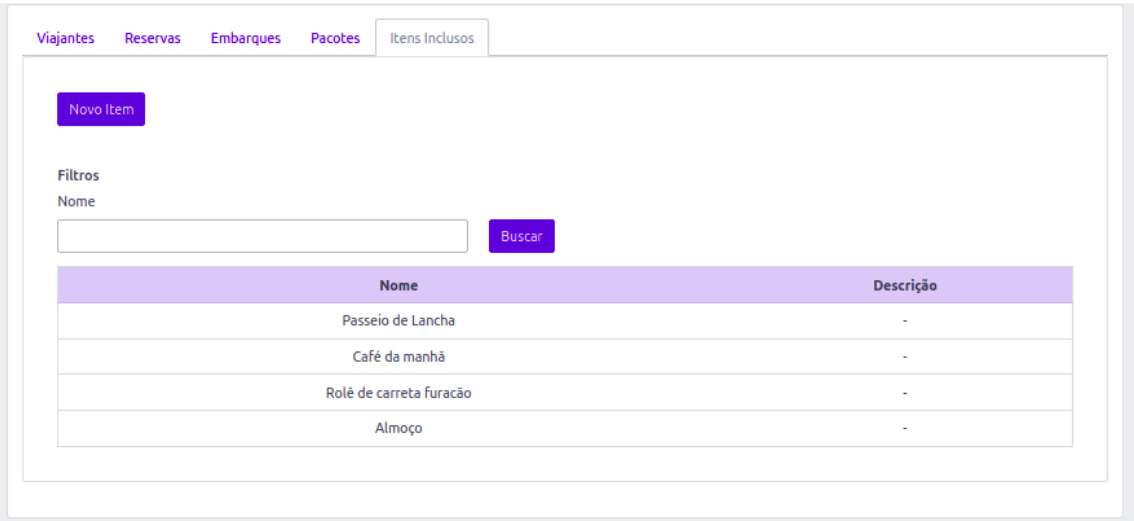


Fonte: Criação própria

4.1.6 Listagens de Itens

Nos detalhes da excursão, o gestor da agência poderá visualizar os itens inclusos nessa excursão ao clicar em "Itens Inclusos"na barra de navegação, como mostra a figura 43. Nessa tela o gestor também pode clicar em "Novo Item"para fazer o cadastro de um novo item.

Figura 43 – Itens

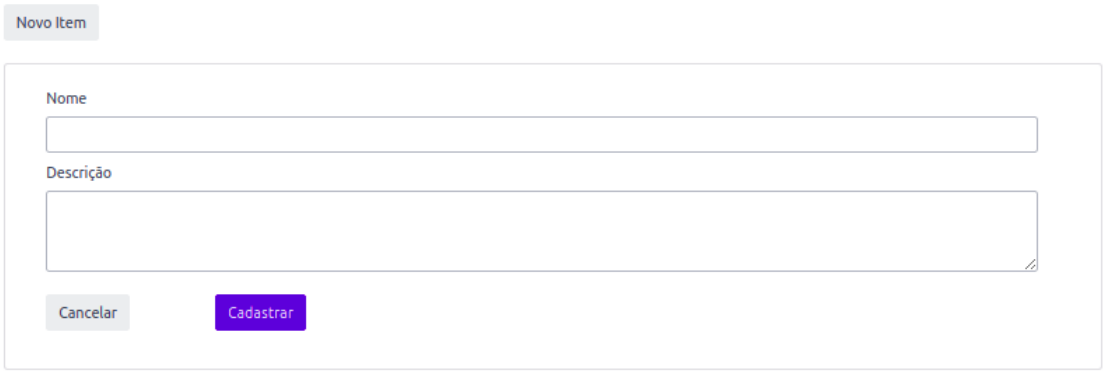


Fonte: Criação própria

4.1.7 Cadastro de Item

Ao clicar em "Novo Item", o gestor poderá fazer o cadastro de um novo item, adicionando o seu nome e sua descrição, como mostra a figura 44.

Figura 44 – Novo Item

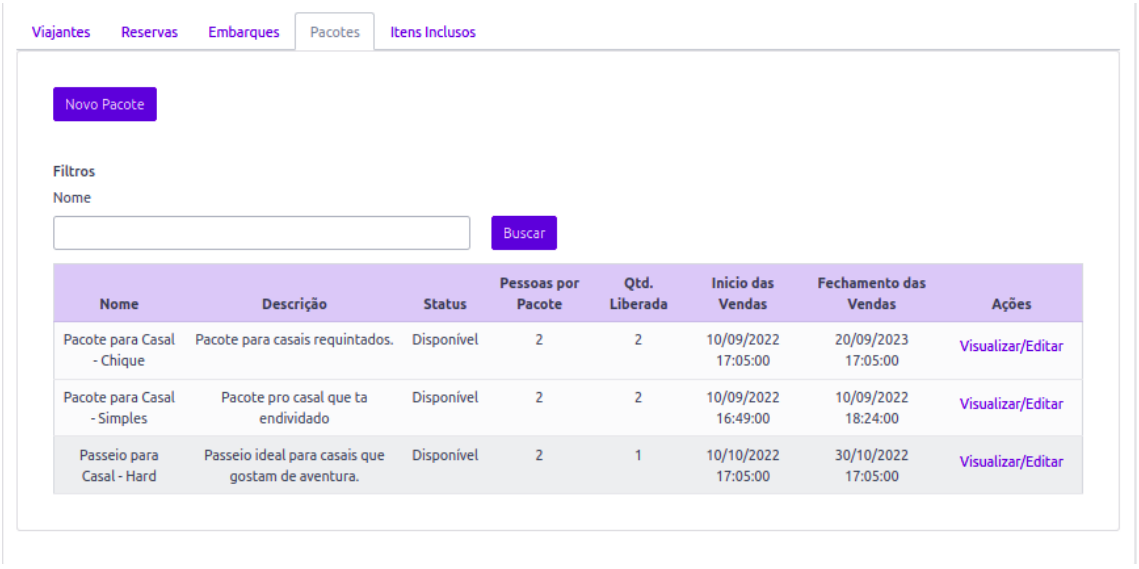


Fonte: Criação própria

4.1.8 Listagem de Pacote

Na seção de pacotes, o gestor irá visualizar a listagem de todos os pacotes cadastrados nessa excursão. Nessa tela ele também poderá buscar os pacotes por nome e ao clicar em "Novo Pacote", poderá fazer o cadastro de um novo pacote. A figura 45 mostra a tela de listagem de pacotes. Como mostra a figura 45, o gestor também pode clicar em "Visualizar/Editar"para visualizar e alterar os dados dos pacotes cadastrados.

Figura 45 – Listagem de Pacotes

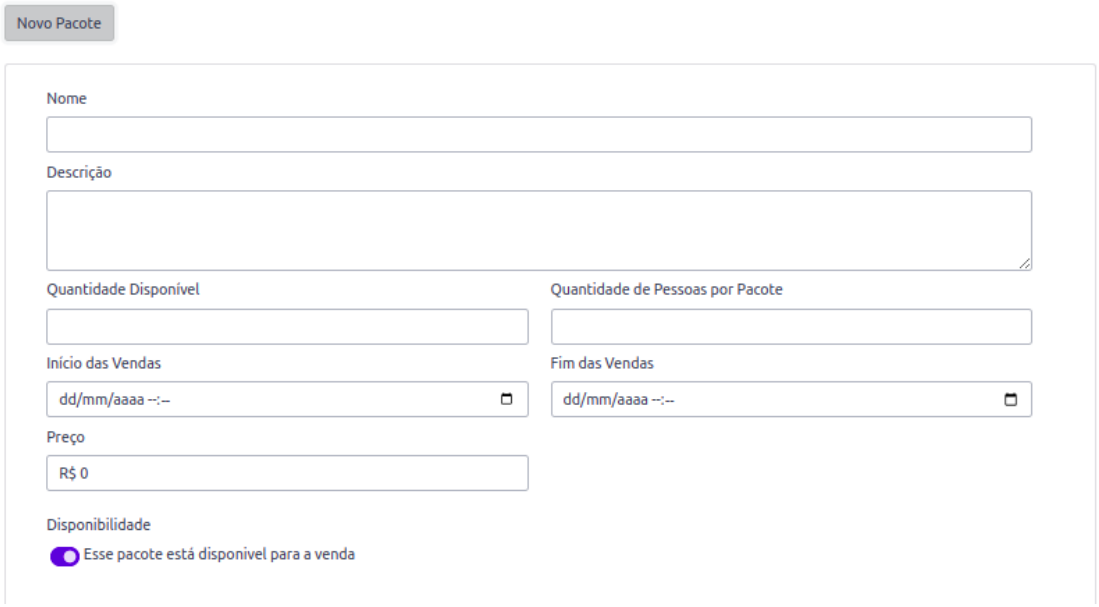


Fonte: Criação própria

4.1.9 Novo Pacote

Ao clicar em "Novo Pacote", o gestor poderá fazer o cadastro de um novo pacote. Nessa tela ele poderá fazer o registro de todos os dados de um pacote, incluindo a quantidade disponível desse pacote, para quantas pessoas são, dade de inicio e fim das vendas, preço e se o pacote deve estar disponível para reserva pelo sistema do viajante. A figura 46 mostra a tela de cadastro de pacote.

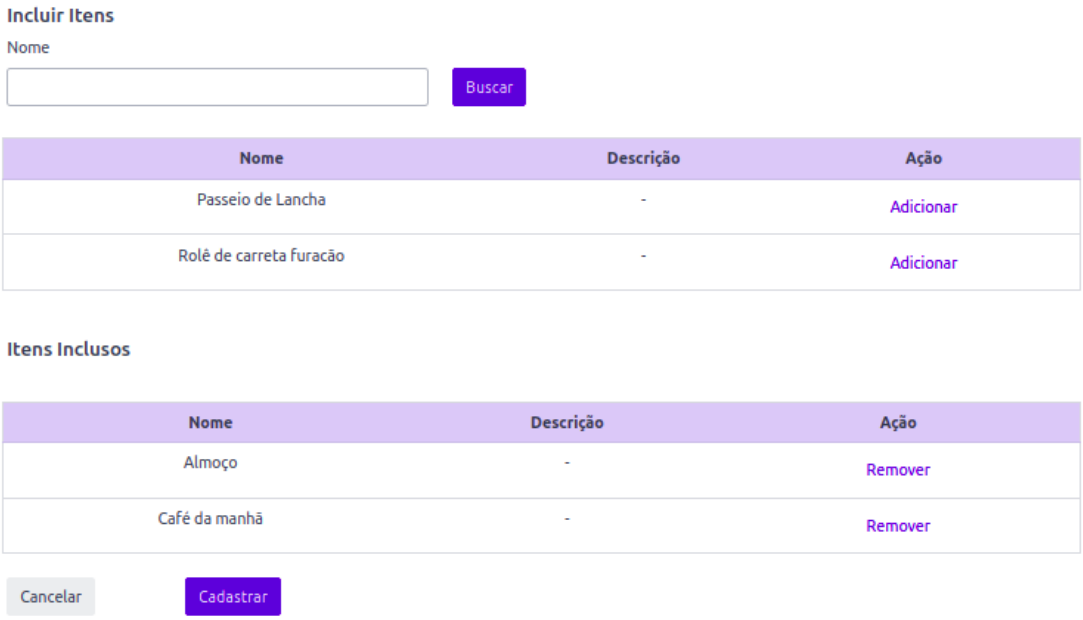
Figura 46 – Novo Pacote



Fonte: Criação própria

Na tela de cadastro de pacote também é possível adicionar e remover itens para esse pacote. A figura 47 mostra a tela em que o gestor pode adicionar e remover itens em um pacote.

Figura 47 – Adicionar Itens no Pacote

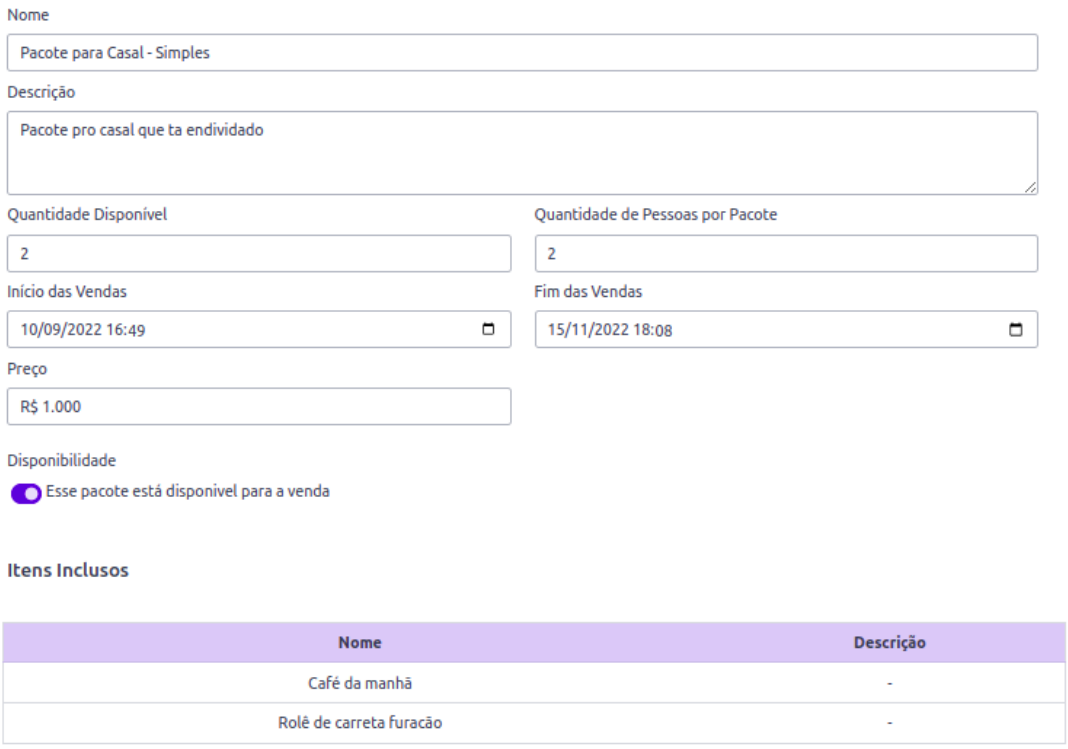


Fonte: Criação própria

4.1.10 Detalhes de Um Pacote

Ao clicar em "Visualizar/editar"um pacote, o gestor poderá editar os dados deste pacote. Porém, não é possível alterar os itens inclusos. A figura 48 mostra a tela de detalhes de um pacote.

Figura 48 – Detalhes de um Pacote

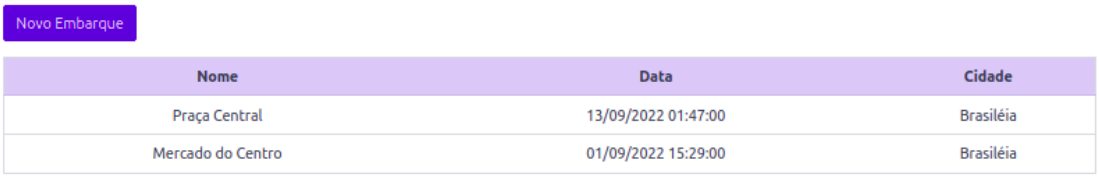


Fonte: Criação própria

4.1.11 Listagem de Embarques

Na seção de listagem de embarques, como mostra a figura 49 o gestor da agência poderá visualizar todas os embarques cadastrados para essa excursão, e poderá criar um novo embarque clicando em "Novo Embarque".

Figura 49 – Listagem de Embarques

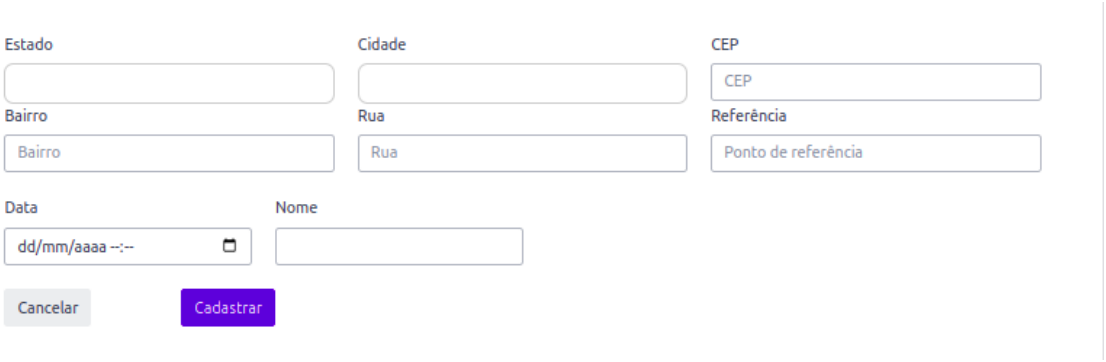


Fonte: Criação própria

4.1.12 Criação de Embarque

Ao clicar em "Novo Embarque"o gestor da agência pode cadastrar um novo embarque para a excursão, cadastrando um nome, dados de endereço e data. A figura 50 mostra a tela de criação de um embarque.

Figura 50 – Criação de Embarque

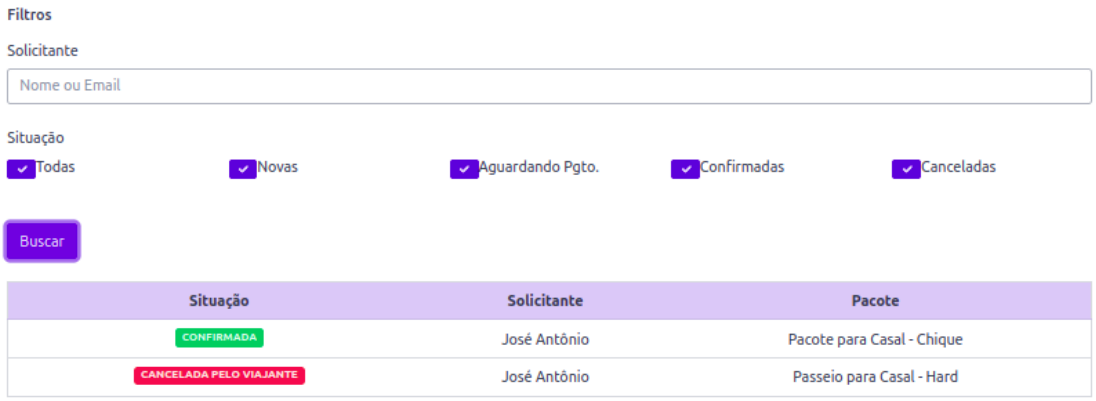


Fonte: Criação própria

4.1.13 Listagem de Reservas

Na seção de listagem de reservas o gestor poderá visualizar todas as reservas realizadas para essa excursão. Como mostra a figura 51, ele poderá filtrar pela situação da reserva que pode ser cancelada, confirmada ou ser uma nova solicitação de resserva. Ao clicar em uma reserva, o gestor poderá visualizar os detalhes daquela reserva e alterar a sua situação.

Figura 51 – Listagem de Reserva

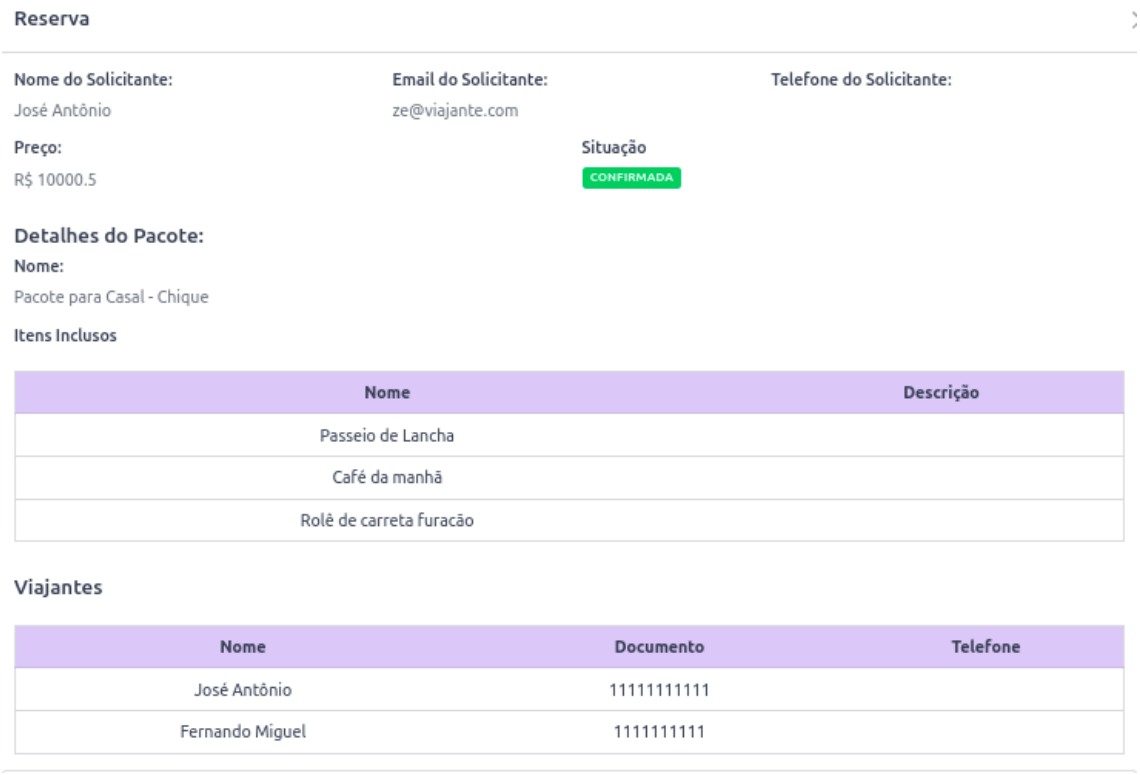


Fonte: Criação própria

4.1.14 Detalhes de uma Reserva

Na tela de detalhes da reserva o gestor visualizará todas as informações a respeito daquela reserva. Poderá ver mais detalhes sobre o solicitante e quem serão os viajantes que farão parte dessa reserva. A figura 52 mostra a tela de detalhes de uma reserva.

Figura 52 – Detalhes de uma Reserva

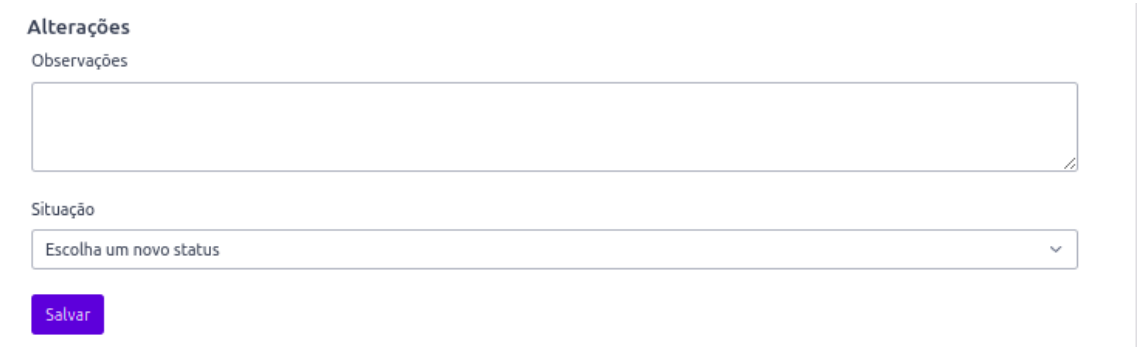


Fonte: Criação própria

4.1.15 Alteração da Situação de uma Reserva

Ainda na tela de detalhes de uma reserva, como mostra a figura 53, o gestor da agência poderá adicionar uma observação a respeito daquela reserva e poderá também alterar a sua situação. É importante ressaltar que as reservas canceladas não podem ter sua situação alterada.

Figura 53 – Alteração da Situação de uma Reserva



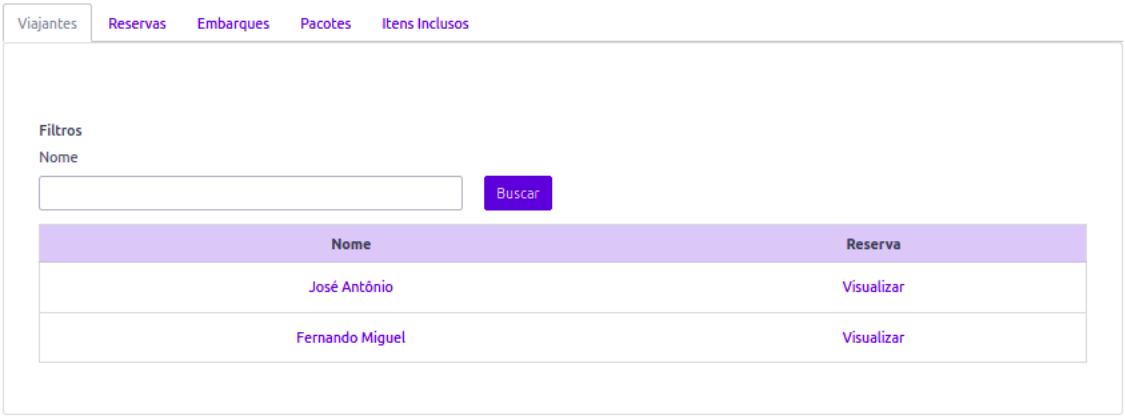
Fonte: Criação própria

4.1.16 Listagem de Viajantes

Na tela de listagem de viajantes, que é exibida na figura 54, o gestor visualizará todos os viajantes que vão participar dessa excursão. Nessa tela o gestor poderá clicar no nome do

viajante para ver os detalhes desse viajante ou ou visualizar a reserva daquele viajante. É importante ressaltar que vários viajantes podem estar relacionados com a mesma reserva.

Figura 54 – Listagem de Viajantes



Fonte: Criação própria

4.1.17 Detalhes do Viajante

Na tela de detalhes do viajante, como mostra a figura 55, são exibidos os dados daquele viajante, como seu nome, seu documento e número de telefone.

Figura 55 – Detalhes do Viajante

Fonte: Criação própria

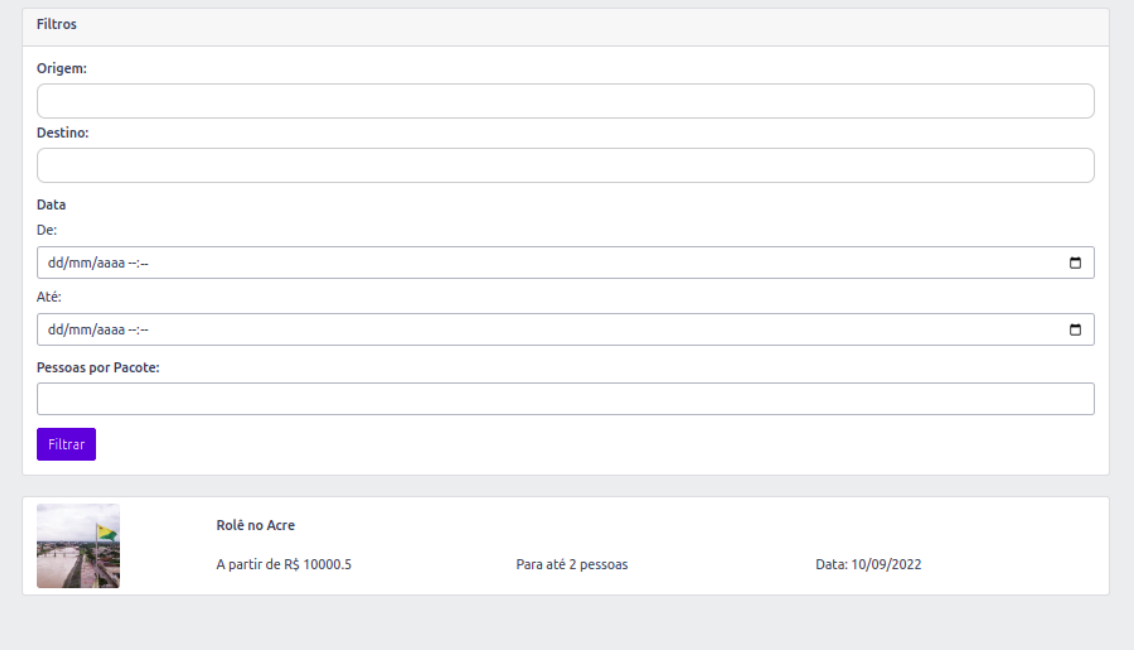
4.2 SISTEMA DO VIAJANTE

Nessa seção serão apresentados os resultados obtidos no desenvolvimento do sistema do viajante.

4.2.1 Tela Principal

Quando um usuário acessa o sistema do viajante, a primeira coisa que ele vê são as excursões disponíveis para serem reservadas. Como mostra a figura 56, nessa tela são exibidas as principais informações de cada excursão que está disponível, como seu nome, preço e sua data. Nessa tela o usuário também pode filtrar por origem, destino, quantidade de pessoas por pacote e data.

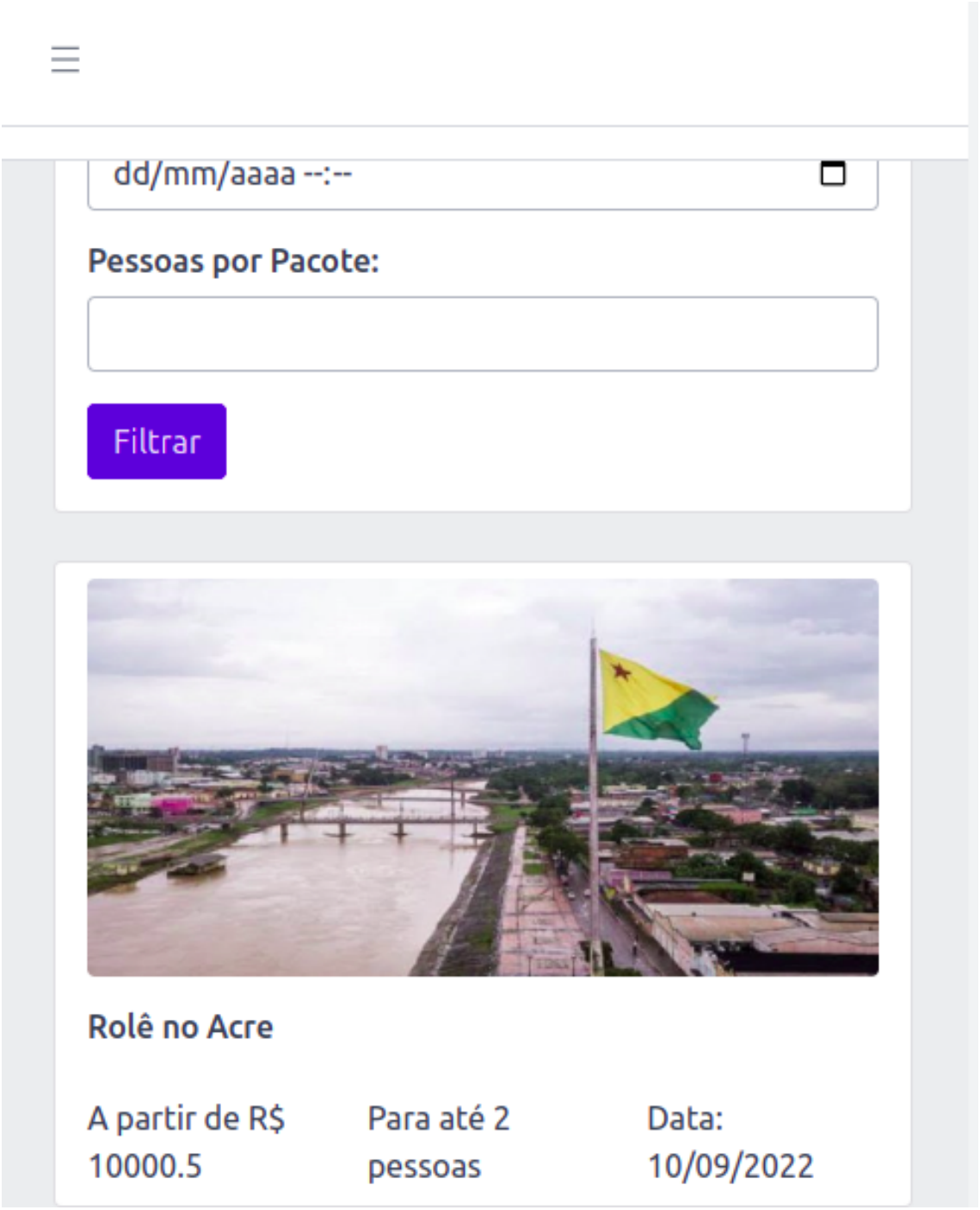
Figura 56 – Tela Principal



Fonte: Criação própria

A figura 57 mostra como a tela principal fica se for acessada através de um dispositivo móvel. Todas as telas desse sistema foram implementadas focando principalmente no acesso através de dispositivos móveis.

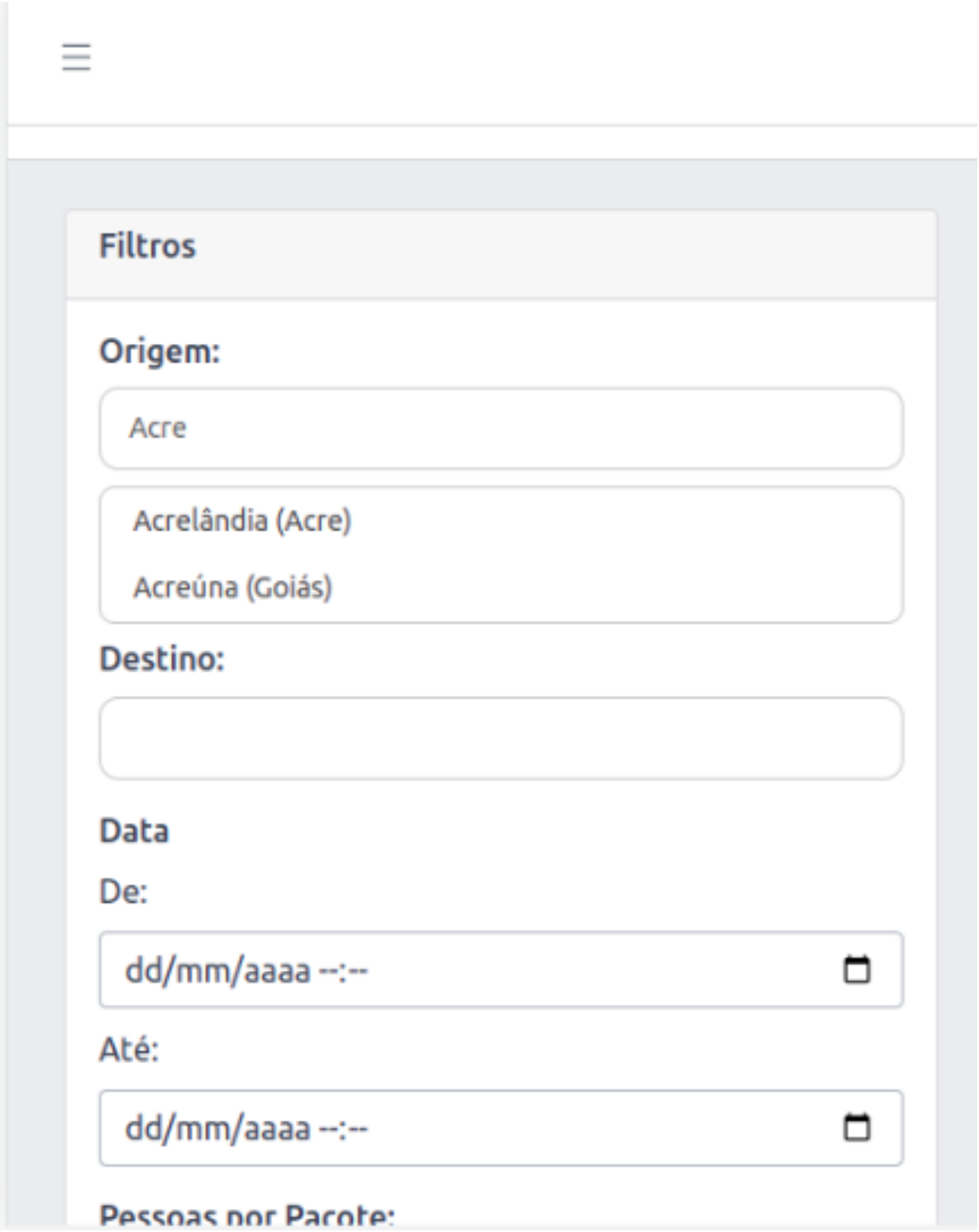
Figura 57 – Tela Principal - Mobile



Fonte: Criação própria

A figura 58 mostra como foi feito o filtro por origem e destino na tela principal.

Figura 58 – Tela Principal - Filtro



Fonte: Criação própria

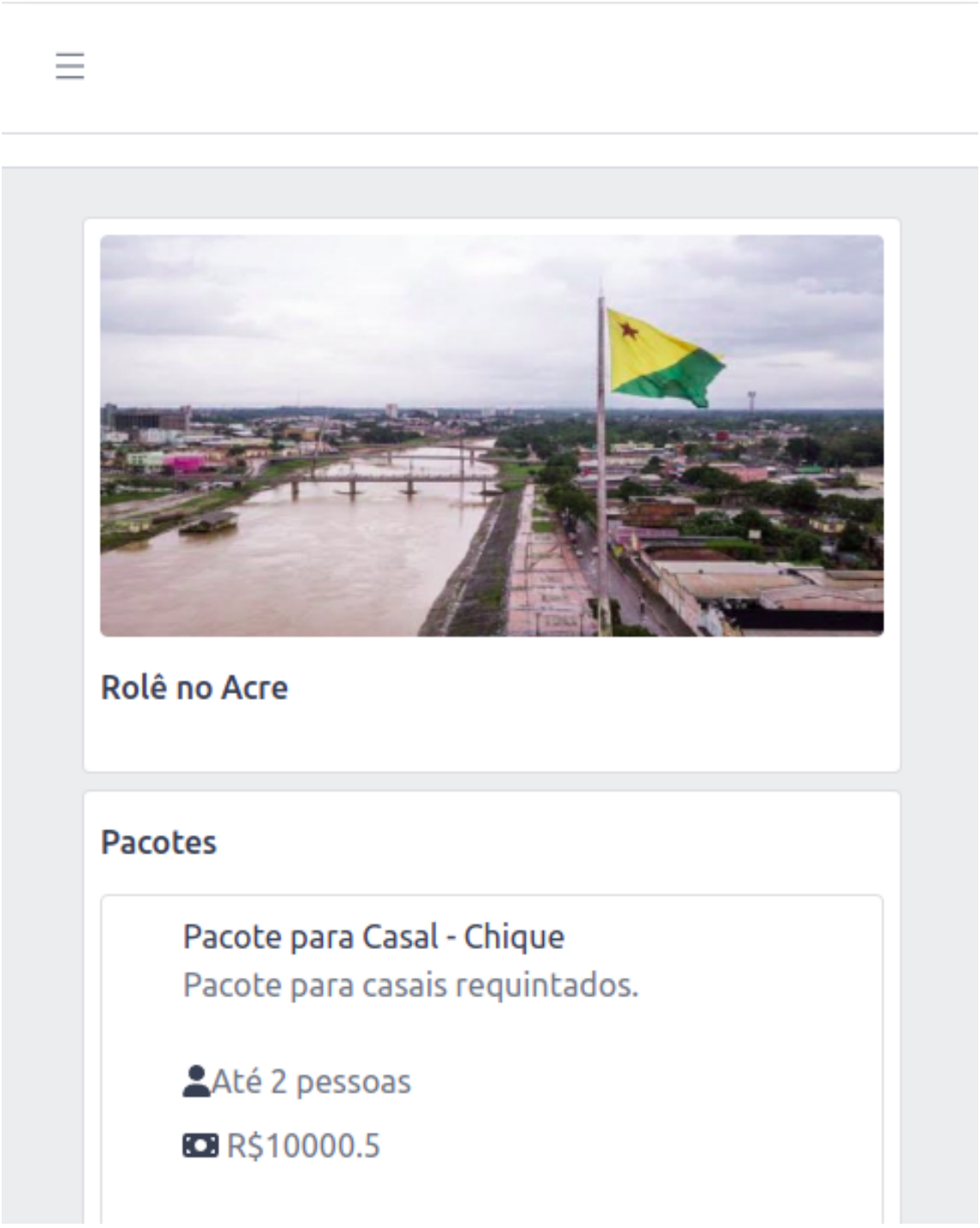
Quando um usuário clica em uma excursão, ele será direcionado para a tela de pacotes da excursão.

4.2.2 Pacotes da Excursão

Na tela de pacotes de excursão, o usuário visualizará a foto de capa da excursão, o nome da excursão e a lista de pacotes que estão disponíveis para serem reservados. A figura 59

mostra a primeira coisa que o usuário visualiza assim que clica em uma excursão.

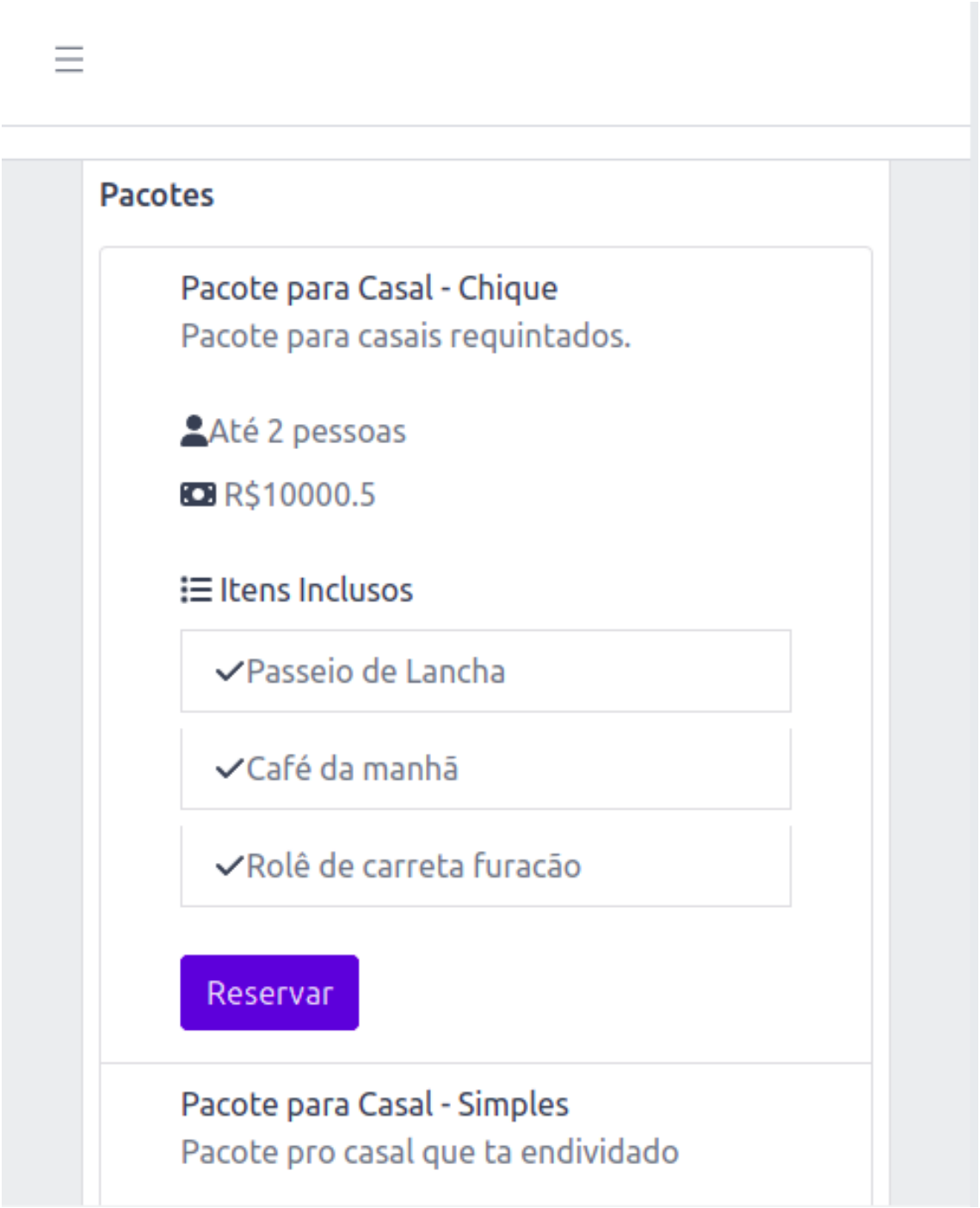
Figura 59 – Pacotes da Excursão



Fonte: Criação própria

Na listagem de pacotes, o usuário poderá clicar em "Itens Inclusos"para visualizar todos os itens inclusos naquele pacote e poderá clicar em "Reservar"para ser direcionado para a tela de reserva de um pacote. A figura 60 mostra os detalhes da listagem de pacotes.

Figura 60 – Pacotes da Excursão - Detalhes



Fonte: Criação própria

4.2.3 Cadastro da Reserva

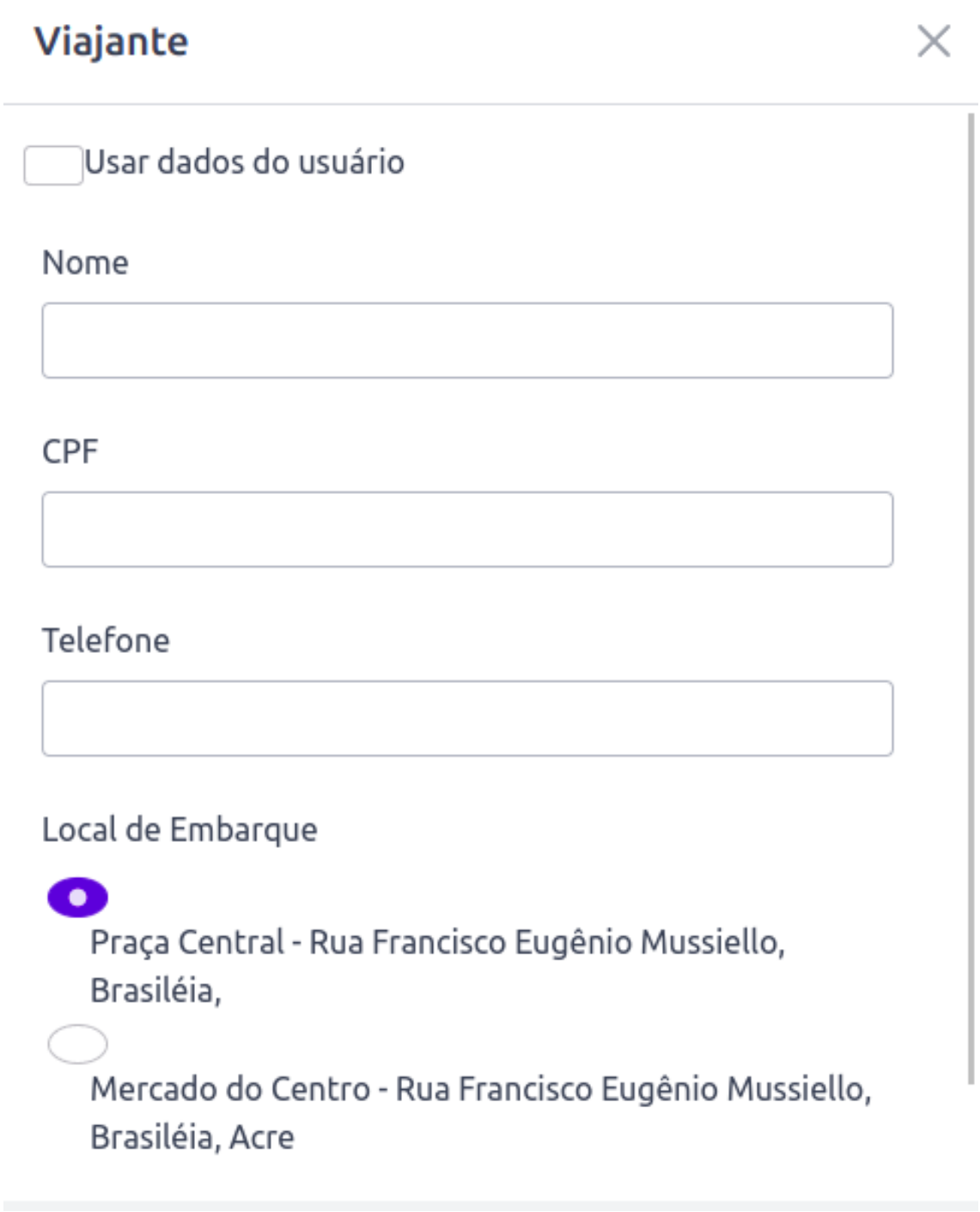
Ao clicar em "Reservar", caso o usuário ainda não esteja logado no sistema, ele será redirecionado para a tela de login. Caso o usuário já esteja logado no sistema, ele será redirecionado para a tela de cadastro de reserva. Como mostra figura 61, nessa tela o usuário verá novamente os detalhes do pacote que ele deseja reservar, mas também poderá adicionar os viajantes que farão parte dessa reserva, ou seja, quem serão os viajantes que irão para a excursão.

Figura 61 – Cadastro de Reserva

Fonte: Criação própria

Ao clicar em "Novo Viajante", o usuário poderá cadastrar um viajante que fará parte da excursão. Nessa tela o usuário terá a opção de adicionar manualmente os dados do viajante ou clicar na opção "Usar dados do usuário". Essa opção vai preencher automaticamente todos os campos do formulário com os dados do usuário que está logado no sistema. Nessa tela o usuário também deve escolher onde o viajante fará o embarque. A figura 62 mostra a tela de cadastro de um novo viajante.

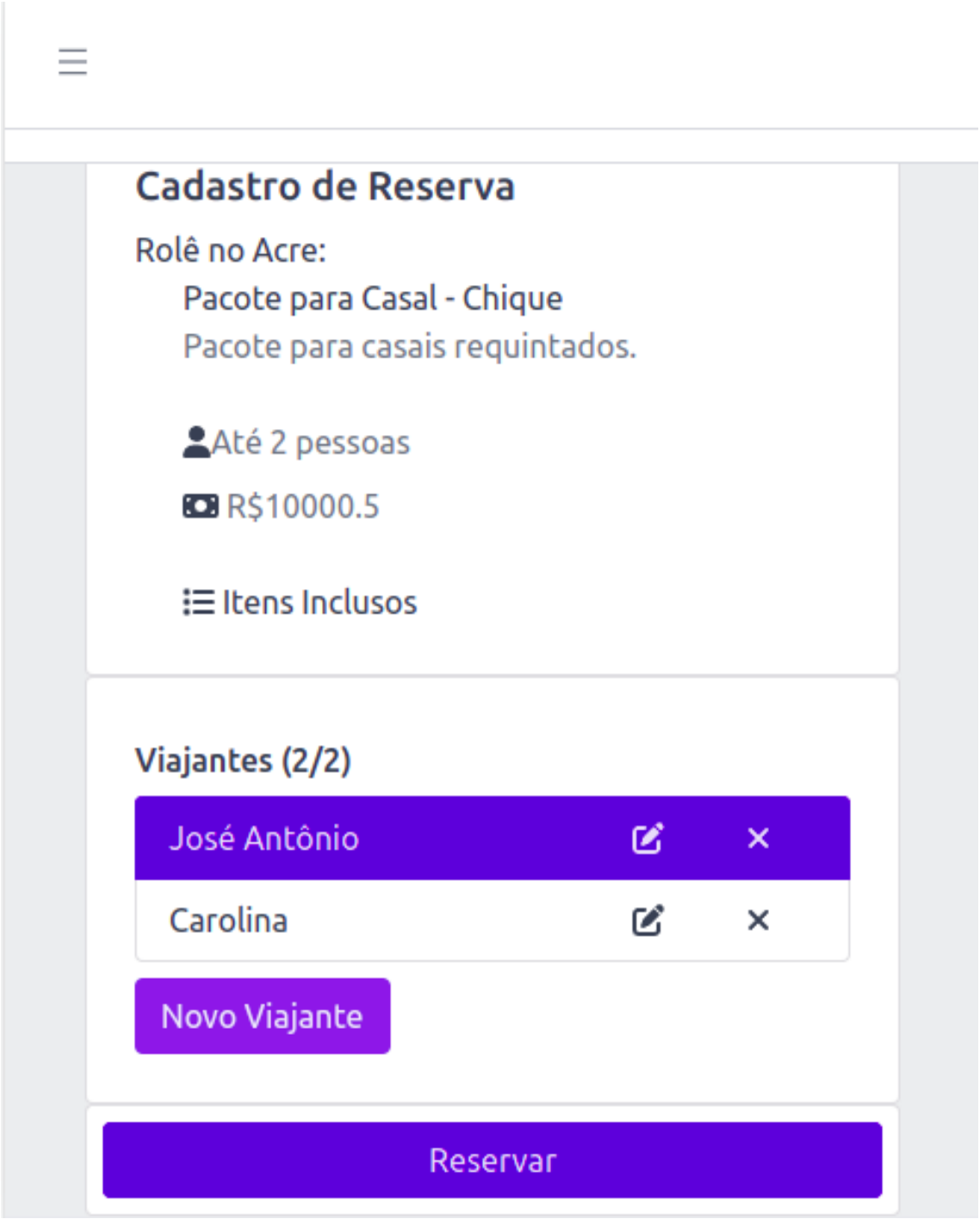
Figura 62 – Cadastro de Viajante



Fonte: Criação própria

A figura 63 mostra a tela de cadastro de reserva após a criação dos viajantes. Quando um dos viajantes cadastrados é o usuário logado, o viajante é exibido com um destaque na cor azul.

Figura 63 – Cadastro de Reserva Completo

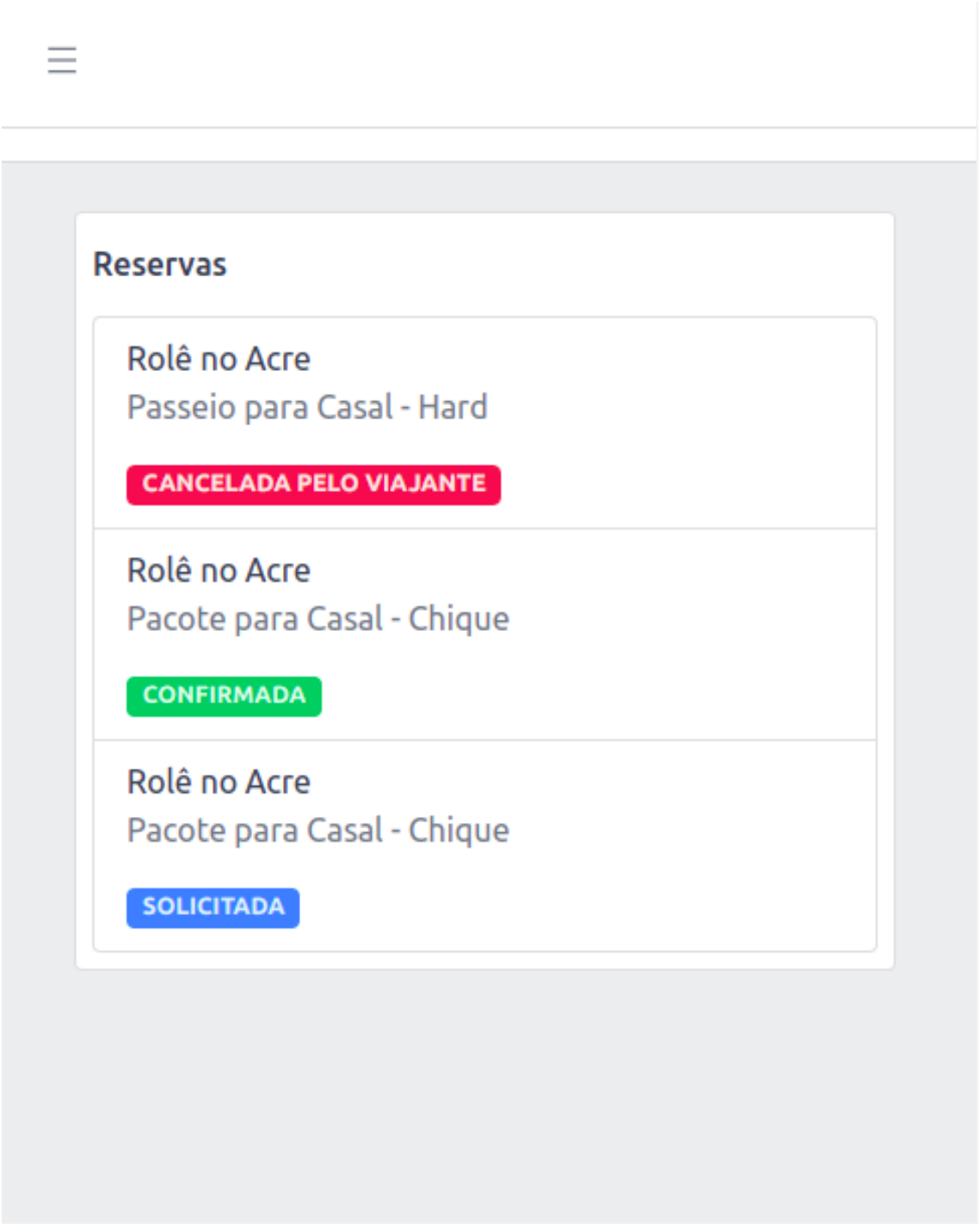


Fonte: Criação própria

4.2.4 Listagem de Reservas

Após a solicitação de uma reserva, o usuário pode acessar a tela de "Listagem de Reservas". Nessa tela o usuário visualizará todas as reservas e a situação em que cada uma das reservas se encontra. A figura 64 mostra a tela de listagem de reservas.

Figura 64 – Listagem de Reservas

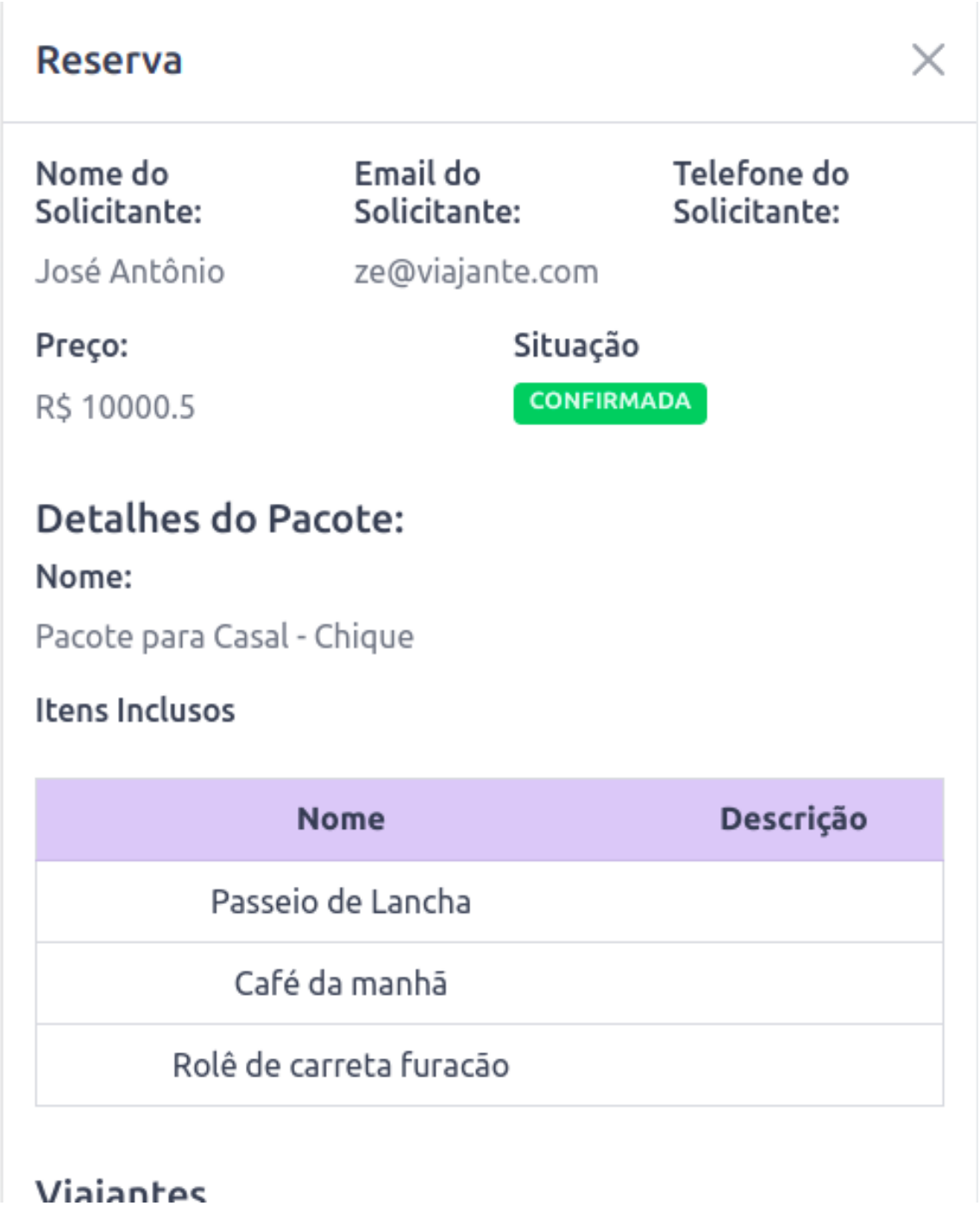


Fonte: Criação própria

4.2.5 Detalhes da Reserva

Ao clicar em uma das reservas, o usuário terá acesso aos detalhes dessa reserva. A figura 65 mostra a tela de detalhes da reserva.

Figura 65 – Detalhes da Reserva



Fonte: Criação própria

4.2.6 Cancelamento de Reserva

Ainda na tela de detalhes de reserva, o usuário pode clicar no botão "Cancelar Reserva"para solicitar o cancelamento da sua reserva. A figura 66 mostra o botão de cancelamento de reserva.

Figura 66 – Cancelamento de Reserva

Fonte: Criação própria

5 CONSIDERAÇÕES FINAIS

Nesta seção será apresentada a conclusão a respeito do desenvolvimento do sistema , as principais dificuldades encontradas durante o processo e as melhorias futuras.

5.1 CONCLUSÃO

Considerando todos os requisitos levantados durante o processo de análise, todos os requisitos considerados importantes foram implementados com sucesso.

Foi possível comprovar a importância da utilização de técnicas de análise de requisitos e engenharia de software aprendidas durante o curso no processo de desenvolvimento. O levantamento de requisitos ajudou, inclusive, na definição do que seria suficiente para uma fase inicial do sistema, de modo que fosse desenvolvido um sistema funcional, que soluciona um problema real e que pode gerar um impacto econômico.

Também é possível concluir que as tecnologias utilizadas para o desenvolvimento do sistema foram boas escolhas, visto que tudo que foi planejado foi entregue. Isso mostra como atualmente existem ferramentas que se tornaram grandes aliadas dos desenvolvedores, facilitando o processo de criação de software.

Durante o desenvolvimento do sistema algumas etapas foram mais dolorosas, como, por exemplo, a implementação do *frontend*, principalmente na criação das interfaces, que é um processo que necessita também de um pouco de criatividade. As ferramentas escolhidas se mostraram uteis também nessa etapa, visto que foram utilizados alguns componentes e *templates* prontos da ferramenta CORE UI, que facilitaram o desenvolvimento.

5.2 TRABALHOS FUTUROS

Após a conclusão do desenvolvimento do sistema e da análise de tudo que foi entregue, foi observado que mesmo tendo resultados positivos, são necessárias algumas melhorias para que o sistema desenvolvido esteja pronto para ser colocado no mercado.

Um dos pontos mais importantes a serem melhorados tem relação com as interfaces dos dois sistemas. Principalmente em se tratando do sistema voltado para o viajante, seria interessante o uso de componentes mais amigáveis e que chamem mais atenção.

Para que o sistema seja capaz de fechar todo o ciclo de reserva de um pacote, também será importante que seja desenvolvida a funcionalidade de pagamento de uma reserva. Essa funcionalidade deverá ser integrada com algum serviço de pagamento.

Outro ponto que deve ser considerado é que seria interessante que o sistema da agência tivesse a capacidade de funcionar mesmo sem conexão com a internet, caso o gestor da

agência resolva acessar o sistema durante um passeio que esteja ocorrendo em algum lugar que não possua acesso a rede.

Também seria interessante que o sistema desenvolvido fosse analisado por um guia de excursão ou por uma agência de excursão, para que as funcionalidades desenvolvidas sejam validadas por alguém que atue nesse mercado.s

REFERÊNCIAS

COREUI. *Core UI*. 2022. Disponível em: <https://coreui.io/>. Acesso em: 13 april. 2022.

FECOMERCIO. *Faturamento do turismo já encolheu 33,6pesquisa*. 2020. Disponível em: <https://www.fecomercio.com.br/noticia/faturamento-do-turismo-ja-encolheu-33-6-em-2020-mostra-pesquisa>. Acesso em: 04 abr. 2021.

FERREIRA, Gabs. *HTTP: Desmistificando o protocolo da Web*. 2019. Disponível em: <https://www.alura.com.br/artigos/desmistificando-o-protocolo-http-parte-1>. Acesso em: 02 abr. 2021.

G1. *Quase 50 mil estabelecimentos do setor do turismo fecharam no país em 6 meses, diz CNC*. 2020. Disponível em: <https://g1.globo.com/economia/noticia/2020/10/05/quase-50-mil-estabelecimentos-do-setor-do-turismo-fecharam-no-pais-em-6-meses-diz-cnc.ght>. Acesso em: 04 abr. 2021.

GRAIG, Larman. *Utilizando UML e padrões - Uma introdução a análise e ao projeto orientados a objeto e ao desenvolvimento interativo*. [S.l.]: "Bookman", 2005.

GRAPHQL. *Get Started*. 2022. Disponível em: <https://graphql.org/learn/>. Acesso em: 04 abr. 2021.

HASURA. *Hasura - Get Started*. 2022. Disponível em: <https://hasura.io/docs/latest/getting-started/index/>. Acesso em: 13 april. 2022.

KALAKOTA, Ravi. *FAQ - Client-Server*. 1997. Disponível em: <https://nostalgie.gershtein.net/1996/www/books/faq/client-server/client-server3.html>. Acesso em: 04 abr. 2021.

LEWIS, Martin Fowler James. *Microservices - a definition of this new architectural term*. 2014. Disponível em: <https://martinfowler.com/articles/microservices.html>. Acesso em: 04 abr. 2021.

MARCONDES, Alexandre Rocha. *Single Page Application e Outras Maravilhas da Web Moderna*. 2013. Disponível em: <https://imasters.com.br/desenvolvimento/single-page-applications-e-outras-maravilhas-da-web-moderna>. Acesso em: 04 abr. 2021.

ORACLE. *Custom Development Method Fast Track (CDM Fast Track)*. [S.l.: s.n.], 2000.

RAHMAN, Md. Tanvir; SHUVA, Taslima Ferdaus; ALI, K. M. Akkas. Trusted device along with trusted location and biometry based authentication method. 2016. Disponível em: <https://www.ijcaonline.org/archives/volume150/number4/rahman-2016-ijca-911490.pdf>. Acesso em: 27 mar. 2022.

S., ROSCA D. GREENSPAN; M., FEBLOWITZ; C., WILD. A decision making methodology in support of business rules lifecycle. 1997. Disponível em: <https://ieeexplore.ieee.org/document/566874>. Acesso em: 27 mar. 2022.

SALVIATI, M. E. Diagrama de fluxo de dados: um novo instrumento para representação gráfica de sistemas. 1982. Disponível em: <http://hdl.handle.net/20.500.11959/brapci/70873>. Acesso em: 27 mar. 2022.

SEBRAE. *Quase 50 mil estabelecimentos do setor do turismo fecharam no país em 6 meses, diz CNC*. 2017. Disponível em: <https://www.sebrae.com.br/sites/PortalSebrae/bis/o-panorama-de-agencias-de-viagens-e-operadores-turisticos-no-brasil>. Acesso em: 04 abr. 2021.

SERUFFO, Marcos César da Rocha. *Cenário proposto: arquitetura cliente-servidor*. 2013. Disponível em: <https://www.researchgate.net/figure/Figura-2-Cenario-proposto-arquitetura-cliente-servidor_fig2_337991059>. Acesso em: 02 abr. 2021.

SOMMERVILE, Ian. *Engenharia de Software*. [S.l.]: "PEARSON", 2013.

VUE. *Get Started*. 2022. Disponível em: <https://vuejs.org/v2/guide/>. Acesso em: 04 abr. 2021.

W3BIG. *estrutura de mensagem HTTP*. 2019. Disponível em: <https://www.alura.com.br/artigos/desmistificando-o-protocolo-http-parte-1>. Acesso em: 24 jun. 2021.

WILSON, Paula Filho. *Engenharia de Software: fundamentos, métodos e padrões*. [S.l.]: "LTC", 2000.