

## **Park'Alot:** Sistema para gerenciamento de estacionamento pagos.

Fernando Camilo Schneider<sup>1</sup>

Jean Carlo Toral<sup>1</sup>

João Ulisses Porto Alegre Ciriaco Teixeira<sup>1</sup>

Otilia Donato Barbosa<sup>2</sup>

Roberson Junior Fernandes Alves<sup>3</sup>

### **Resumo**

O Park'Alot é um sistema de gerenciamento de estacionamento desenvolvido para otimizar a utilização de vagas em ambientes de alta demanda. Criado no contexto do crescimento da frota veicular brasileira, o projeto utiliza Java, *Spring Boot* e *PostgreSQL* para implementar funcionalidades de cadastro, reserva e controle de vagas. A metodologia envolveu a criação de diagramas *UML* para estruturar os requisitos e garantir uma implementação organizada. O sistema propõe-se a melhorar a experiência dos usuários e a eficiência operacional de estacionamento em locais como aeroportos, shoppings e centros comerciais.

Palavras-chave: Gerenciamento de estacionamento; Sistema *web*; Java; *Spring Boot*; Mobilidade urbana.

---

<sup>1</sup> Discentes do Curso de Ciência da Computação  
Unoesc-Campus de São Miguel do Oeste  
Rua Oiapoc, 211. São Miguel do Oeste-SC  
[schneiderfernandocamilo@gmail.com](mailto:schneiderfernandocamilo@gmail.com); [jean.toral@unoesc.edu.br](mailto:jean.toral@unoesc.edu.br);  
[joao.upact@gmail.com](mailto:joao.upact@gmail.com).

<sup>2</sup> Mestre em Informática  
Docente do Curso de Ciência da Computação  
Unoesc-Campus de São Miguel do Oeste  
Rua Oiapoc, 211. São Miguel do Oeste-SC  
[otilia.barbosa@unoesc.edu.br](mailto:otilia.barbosa@unoesc.edu.br).

<sup>3</sup> Mestre em Computação Aplicada  
Docente do Curso de Ciência da Computação  
Unoesc-Campus de São Miguel do Oeste  
Rua Oiapoc, 211. São Miguel do Oeste-SC  
[roberson.alves@unoesc.edu.br](mailto:roberson.alves@unoesc.edu.br).

## 1 INTRODUÇÃO

O tráfego de veículos nas cidades brasileiras é um grande desafio para o cenário urbano devido ao aumento constante da frota veicular, que já ultrapassa 119 milhões de veículos em dezembro de 2023, segundo estudos do Instituto Brasileiro de Planejamento e Tributação (IBPT).

Com base nos dados oficiais da ABRAPARK, o setor de estacionamentos representa um segmento significativo da economia brasileira, contando com aproximadamente 11 mil estabelecimentos comerciais que totalizam mais de 3 milhões de vagas disponíveis em todo país. O setor é responsável pela geração de mais de 300 mil empregos diretos, demonstrando sua relevância não apenas para a mobilidade urbana, mas também para o mercado de trabalho nacional. Este cenário evidencia a necessidade de sistemas eficientes de gerenciamento que possam otimizar a operação destes estabelecimentos, melhorando tanto a experiência dos usuários quanto a gestão dos recursos disponíveis.

Diante deste contexto, este trabalho apresenta o desenvolvimento de um sistema de gerenciamento de estacionamentos, visando otimizar a utilização do espaço e melhorar a experiência dos usuários através de tecnologias modernas que atendam às demandas atuais do setor.

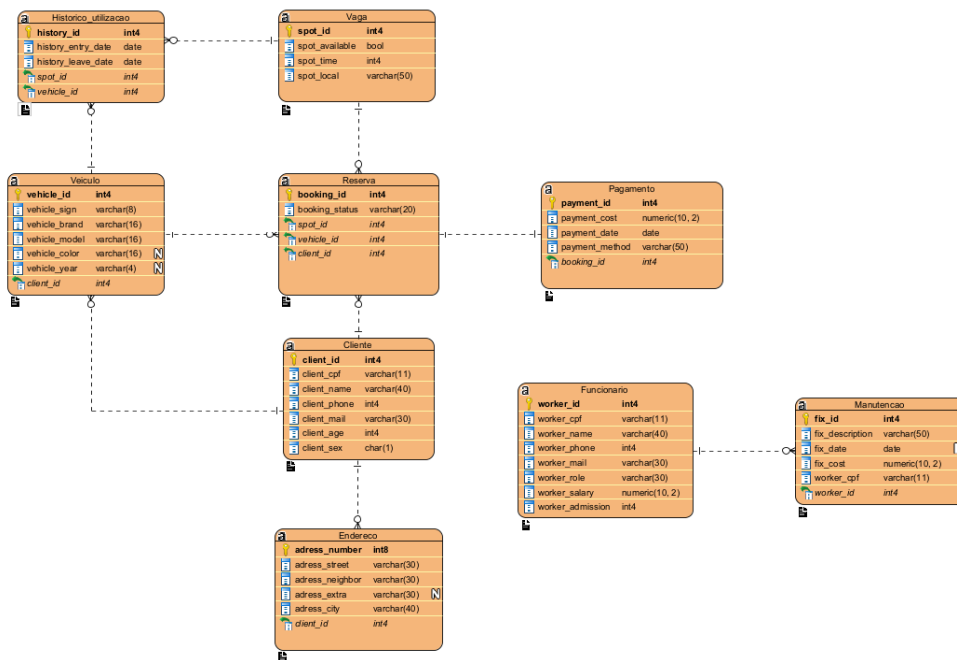
## 2 DESENVOLVIMENTO

Para facilitar o desenvolvimento do projeto, foi utilizado o banco de dados relacional *PostgreSQL*, já desenvolvido anteriormente, juntamente com a ferramenta *Dbeaver*. Na parte de programação foi utilizado a linguagem Java juntamente com a ferramenta *Visual Studio Code* como ambiente de desenvolvimento para fazer a parte do *back-end* com *Spring Boot*, com o intuito de fazer a conexão com o banco de dados. No desenvolvimento dos modelos foram utilizados os requisitos já levantados anteriormente junto com a ferramenta *Visual paradigm*. Para fazer o versionamento do projeto foi utilizado o *GitHub*. No planejamento das atividades e das divisões de tarefas foi utilizado o *Google Drive*.

## 2.1 MODELAGEM

Depois de levantar todos os requisitos necessários para o desenvolvimento do sistema foi iniciada a construção do novo diagrama relacional, seguindo as atualizações feitas no banco de dados como demonstra a figura abaixo.

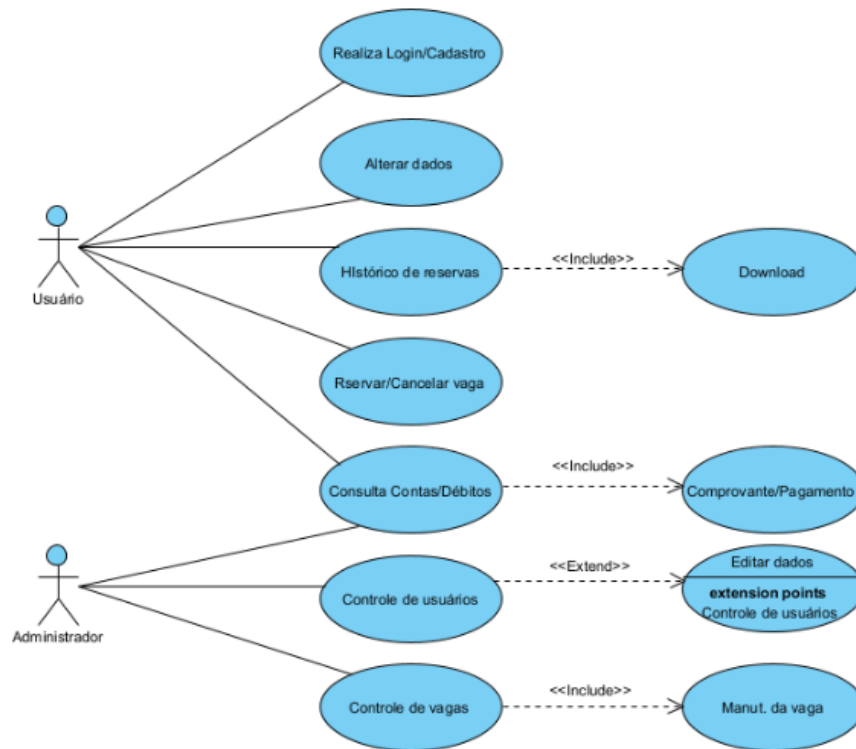
Figura 1: Diagrama Relacional



Fonte: Os autores (2024)

Em seguida foi feito o diagrama de casos de uso, cuja função é descrever como os usuários interagem com o sistema, além disso facilitam a organização dos requisitos de um sistema.

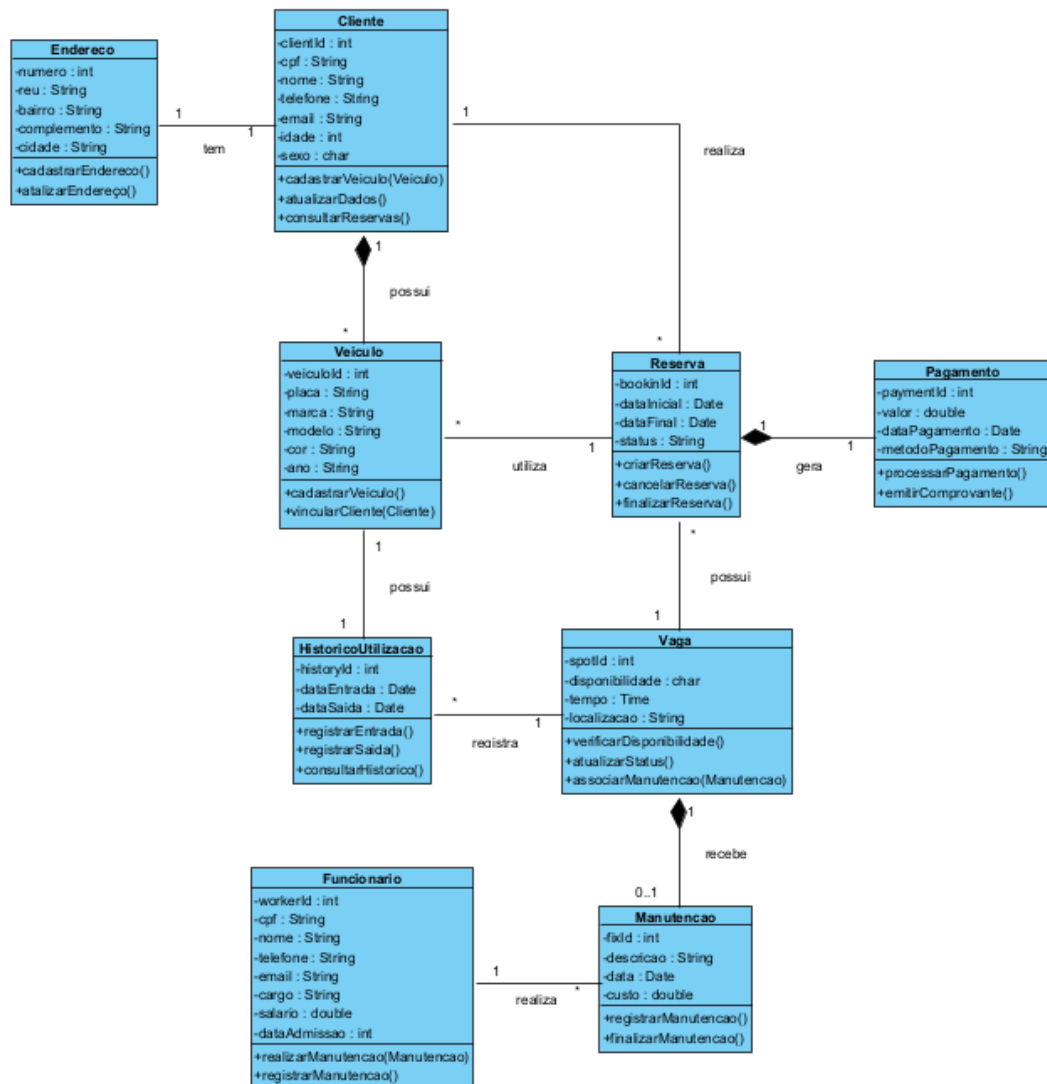
Figura 2: Diagrama de Casos de Uso



Fonte: Os autores (2024)

Para ter uma visão da estrutura do sistema de forma estática, e visualizar as interfaces e seus relacionamentos foi utilizado um diagrama de classes, exposto abaixo, modelado utilizando o *Visual Paradigm*, onde tem a possibilidade de verificar as ligações das tabelas e demais informações importantes.

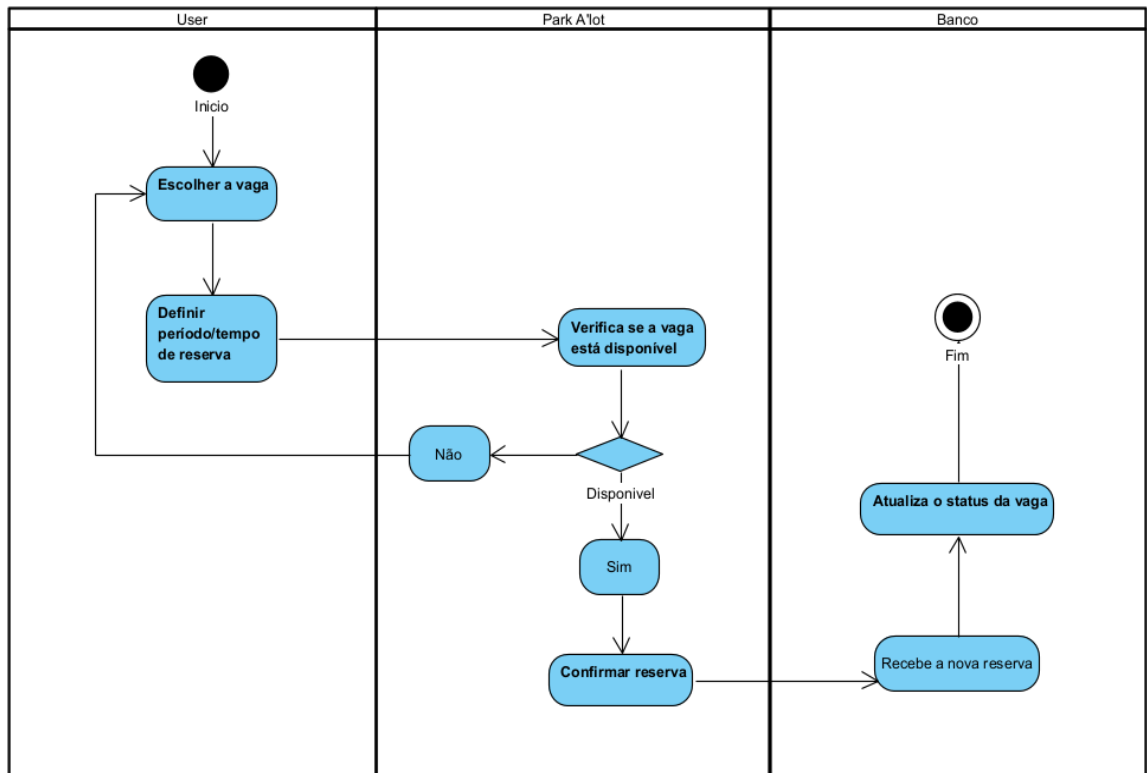
Figura 3: Diagrama de Classes



Fonte: Os autores (2024)

Para descrever os passos a serem percorridos para a conclusão de um método ou algoritmo específico, neste caso a reserva de uma vaga, foi feito um diagrama de atividades conforme demonstra a figura abaixo.

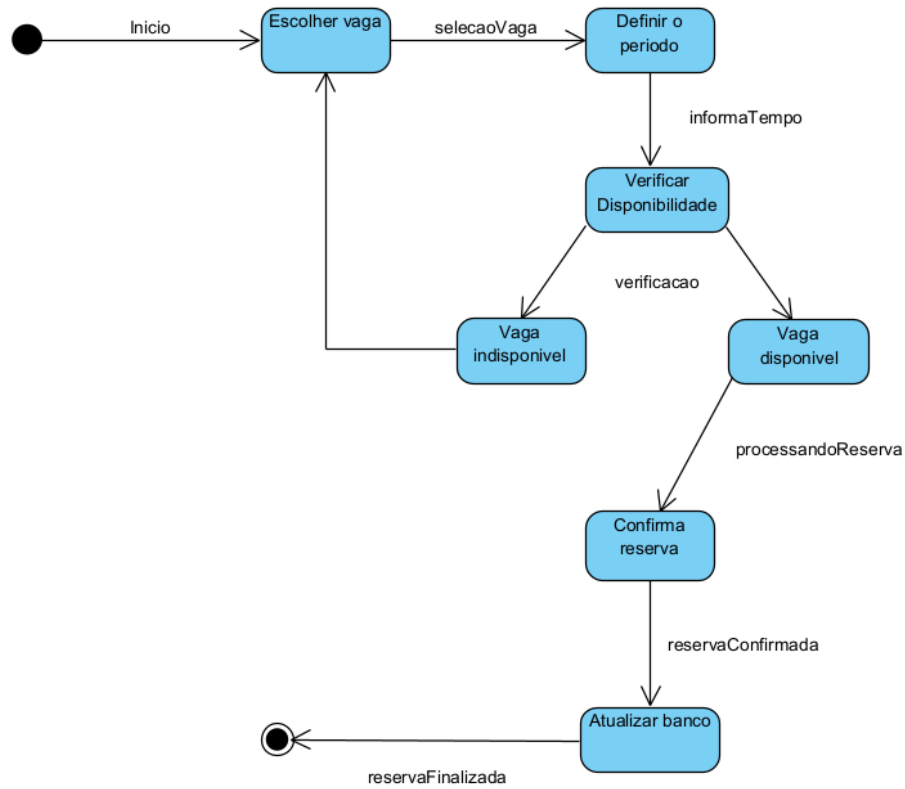
Figura 4: Diagrama de Atividades



Fonte: Os autores (2024)

Em seguida foi feito o Diagrama de Estados, que é utilizado para acompanhar os estados por que passa uma instância de uma classe, ou representar os estados de um Caso de Uso ou mesmo de um subsistema ou sistema completo.

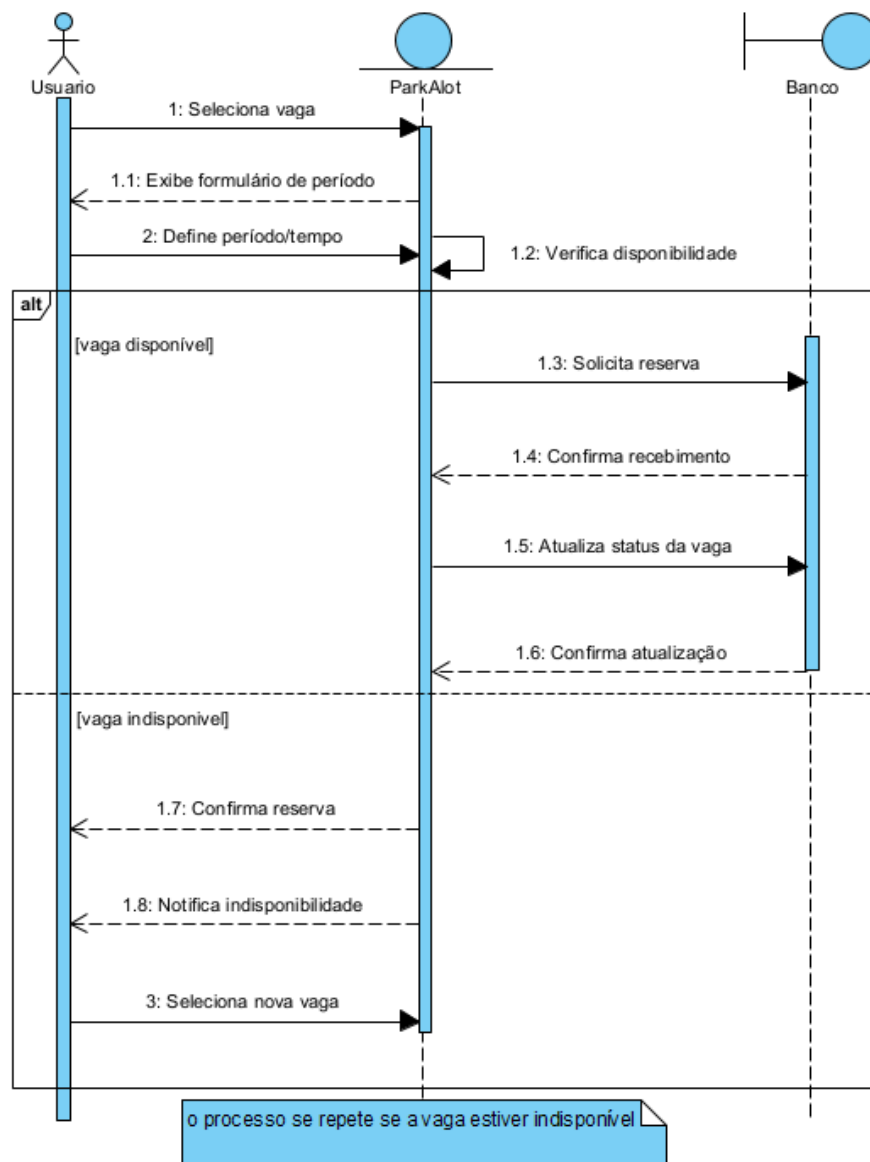
Figura 5: Diagrama de Estados



Fonte: Os autores (2024)

Para concluir a parte da modelagem foi feito o Diagrama de Sequência que enfatiza a ordenação das mensagens trocadas entre os objetos.

Figura 6: Diagrama de Sequência



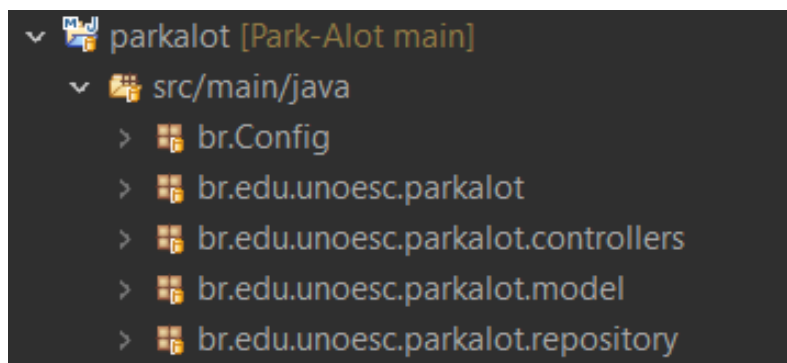
Fonte: Os autores (2024)

## 2.2 BACKEND

Para o início do desenvolvimento do *backend*, foram utilizadas as ferramentas Java e *Spring Boot*. A estrutura da aplicação foi projetada de forma simplificada, priorizando maior legibilidade e facilidade para possíveis correções futuras. A organização do projeto segue uma separação clara de arquivos em pastas específicas, como ilustrado na figura 7.



Figura 7: Organização de arquivos



Fonte: Os autores (2024)

No pacote **br.edu.unoesc.parkalot**, encontra-se a classe principal do projeto. Já na pasta *Controllers*, estão todas as classes responsáveis pelo controle do sistema, incluindo as de Cliente, Veículo, Vaga, Reserva e uma específica para o relacionamento entre Veículo e Vaga. Na pasta *Models*, encontram-se os scripts para a criação da base de dados. Apesar de o projeto incluir todos os campos necessários para a execução completa, a versão final utilizou apenas as tabelas de Cliente, Veículo, Vaga e Reserva. A configuração da conexão com o banco de dados foi realizada no arquivo de propriedades ***application.properties***, conforme apresentado na figura 8.

Figura 8: Arquivo de propriedades

```
1 spring.application.name=parkalot
2
3 server.port=8080
4 server.address=0.0.0.0
5 spring.h2.console.enabled=true
6 spring.h2.console.path=/h2-console
7 spring.datasource.url=jdbc:postgresql://localhost:5433/pparkalot
8 spring.datasource.username=postgres
9 spring.datasource.password=postgres
10 spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
11
12 spring.jpa.properties.hibernate.id.new_generator_mappings=true
13 spring.jpa.hibernate.ddl-auto=update
14
```

Fonte: Os autores (2024)

Foram implementadas diversas requisições para garantir o funcionamento da aplicação:

- **Tabela Cliente:** além das operações básicas de *CRUD* (*Create, Read, Update, Delete*), foram desenvolvidas requisições para verificação de CPF e verificação de e-mail — utilizadas na área de cadastro —, além da requisição de *login*, que valida os dados inseridos pelo usuário e verifica sua existência na base.
- **Tabela Veículo:** foram realizadas apenas as requisições básicas de *CRUD*.
- **Tabela Vaga:** foram criadas requisições de listagem, incluindo uma para exibir todas as vagas e outra específica para vagas disponíveis.
- **Tabela Reserva:** além das operações *CRUD*, implementou-se uma requisição para relacionar veículo e vaga, que altera automaticamente o status da vaga para "Reservada". Também foi criada uma requisição para listar todas as reservas associadas ao cliente logado no sistema.

## 2.3 FRONTEND


Para testar todas essas requisições, utilizou-se a ferramenta Insomnia. A parte do frontend foi desenvolvida utilizando HTML, CSS e JavaScript, como demonstrado nas próximas imagens.

Figura 9: Página de cadastro



The image shows a registration form titled "Cadastro-se" for the application "PARK'AL'T". The form is centered on a light blue background. At the top, there is a logo consisting of a car icon inside a blue octagon, followed by the text "PARK'AL'T" in a bold, sans-serif font. Below the logo, the form contains several input fields: "Informe o nome", "Informe o CPF", "Informe o email", "Informe o telefone", "Selecione o sexo" (with a dropdown arrow), and "dd/mm/aaaa" (with a calendar icon). At the bottom of the form, there is a "Cadastrar" button and a link that says "Já é cadastrado? Entre".

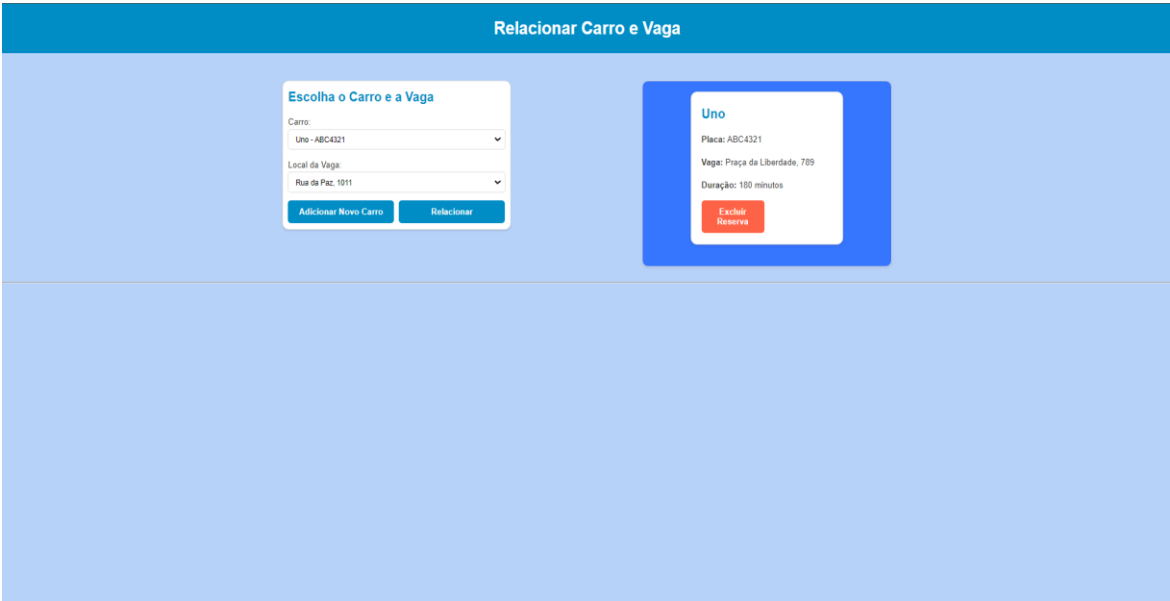
Fonte: Os autores (2024)

Figura 10: Página de *login*

The image shows a login form titled "LOGIN" for the application "PARK'AL'T". The form is centered on a light blue background. At the top, there is a logo consisting of a car icon inside a blue octagon, followed by the text "PARK'AL'T" in a bold, sans-serif font. Below the logo, the form contains two input fields: "Email" and "CPF". At the bottom of the form, there are two buttons: "ENTRAR" and "CADASTRAR".

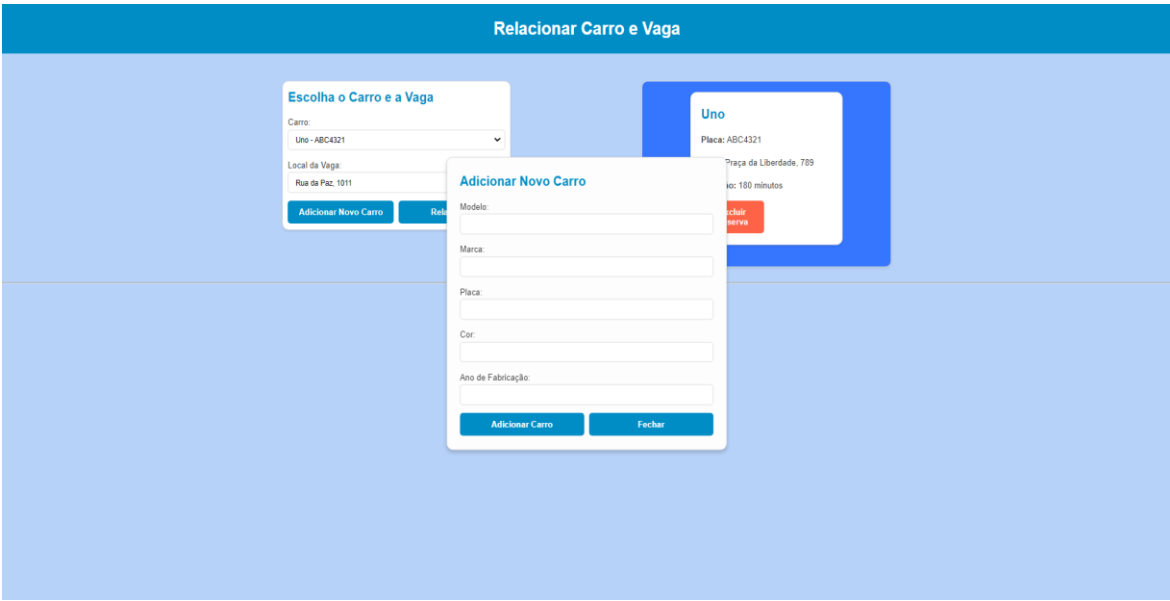
Fonte: Os autores (2024)

Figura 11: Página para reservar vagas



Fonte: Os autores (2024)

Figura 12: Modal para adicionar novo carro



Fonte: Os autores (2024)

### 3 CONCLUSÃO

O desenvolvimento deste sistema de gerenciamento de estacionamento demonstrou o impacto positivo que as tecnologias podem trazer ao setor, além disso foi pensado para atender ambientes de alta demanda e fluxo intenso, como aeroportos, shoppings, estádios, e centros comerciais. Pelo fato destes locais apresentarem desafios específicos devido aos picos de utilização em determinados horários, causado pelo fluxo de veículos e pelas necessidades distintas que cada ambiente possui, é essencial implementar soluções ágeis que facilitem o acesso dos usuários e reduzam o tempo de espera, garantindo assim uma experiência gratificante.

A estrutura do projeto foi realizada em camadas bem definidas, separando claramente o backend e o frontend, permitindo um desenvolvimento organizado e com fácil manutenção. Utilizando *Spring Boot* com Java no *backend*, junto com o banco de dados *PostgreSQL*, proporcionou uma base sólida para o sistema, enquanto a interface desenvolvida em HTML, CSS e JavaScript oferece uma experiência intuitiva para os usuários.

Na modelagem foi realizada a criação de diversos diagramas , para garantir que o sistema fosse implementado de uma forma estruturada e alinhada com os requisitos coletados. As funcionalidades que foram implementadas no sistema atendem às necessidades básicas de um estacionamento eficiente e moderno.

Este sistema contribui para a melhoria da mobilidade urbana e do gerenciamento de estacionamento, com potencial de ampliação para atender a demandas futuras. Por ter uma estrutura escalável ele pode ser adaptado para diversos cenários específicos, para que não contribua apenas para melhorar a mobilidade e a experiência dos usuários, mas também para aumentar a eficiência operacional dos estacionamento.

Acredita-se que a expansão futura do projeto, com a integração de novas tecnologias e funcionalidades específicas, fortalecerá ainda mais sua aplicabilidade em diferentes setores e contextos.

***Park'Alot: System for managing paid parking lots.***

**Abstract**

*Park'Alot is a parking management system developed to optimize parking space utilization in high-demand environments. Created in the context of the Brazilian vehicle fleet growth, the project uses Java, Spring Boot, and PostgreSQL to implement registration, reservation, and parking space control functionalities. The methodology involved creating UML diagrams to structure requirements and ensure an organized implementation. The system aims to improve user experience and operational efficiency in parking spaces at locations such as airports, shopping malls, and commercial centers.*

*Keywords: Parking management; Web system; Java; Spring Boot; Urban mobility.*

## REFERÊNCIAS

ROVER, Ardinete; MELLO, Regina Oneda. **Normas da ABNT: Orientações para a produção científica**. 2. ed. Joaçaba: Editora Unoesc, 2024.

O TEMPO. Frota brasileira fecha 2023 em 119.227.657, um veículo para cada 1,7 habitante. **O Tempo**, 2024. Disponível em: <https://www.otempo.com.br/economia/frota-brasileira-fecha-2023-em-119-227-657-um-veiculo-para-cada-1-7-habitante-1.3346324>. Acesso em: 20 de nov. de 2024.

ALVES, Roberson J. F. **Modelagem UML: Diagrama de Classes**. São Miguel do Oeste: Unoesc, 2024. 26 slides. Apresentação de slides.

ALVES, Roberson J. F. **Análise e Projeto OO com UML: Introdução e Casos de Uso**. São Miguel do Oeste: Unoesc, 2024. 68 slides. Apresentação de slides.

ALVES, Roberson J. F. **Modelagem UML: Diagrama de Atividades e de Estados**. São Miguel do Oeste: Unoesc, 2024. 36 slides. Apresentação de slides.

ALVES, Roberson J. F. **Modelagem UML: Diagrama de Sequência**. São Miguel do Oeste: Unoesc, 2024. 27 slides. Apresentação de slides.