Relatório Final Java Parking

Áulus Arcanjo Alves Batisa[1], **Henrique Leite Najar**[1],
Kayque Allan Ribeiro Freitas[1],**Rafael de Oliveira Caldeira Lopes**[1]

[1]Instituto de Ciências Exatas e Informática - PUC MINAS

Abstract. *The increasing scarcity of parking spaces in Brazilian urban centers has proven to be a promising opportunity for innovative companies. In this context, Xulambs plans to launch Xulambs Parking, an integrated system for managing parking facilities in major cities. Xulambs Parking will feature a network of strategically distributed parking lots, offering a practical and functional solution that prioritizes customer comfort and ease of use. Moreover, all data, such as customer records, vehicle information, and parking space occupancy, will be securely stored in a database, ensuring operational efficiency and reliability.*

Resumo. *A crescente escassez de vagas de estacionamento nos centros urbanos brasileiros tem se mostrado uma oportunidade promissora para empresas inovadoras. Nesse cenário, a Xulambs planeja lançar o Xulambs Parking, um sistema integrado para a gestão de estacionamentos em grandes cidades. O Xulambs Parking contará com uma rede de estacionamentos distribuídos estrategicamente, oferecendo uma solução prática e funcional que prioriza o conforto e a facilidade de uso para os clientes. Além disso, todos os dados, como registros de clientes, veículos e ocupação de vagas, serão armazenados de forma segura em um banco de dados, garantindo a eficiência e confiabilidade da operação.*

1. Visão Geral das Principais Funcionalidades

O sistema de estacionamento Xulambs foi desenvolvido para fornecer uma solução integrada e eficiente para a gestão de estacionamento nas grandes cidades, atendendo aos requisitos estabelecidos em cada sprint de desenvolvimento.

2. Gestão de Estacionamentos

- Cadastro e gerenciamento de múltiplos estacionamentos, cada um com um número fixo de vagas identificadas alfanumericamente (ex.: vaga A02).
- Classificação das vagas em categorias específicas: Regular, Idoso, PCD e VIP, cada uma com regras e características particulares, como descontos ou tarifas diferenciadas.

2.1. Controle de Vagas e Veículos

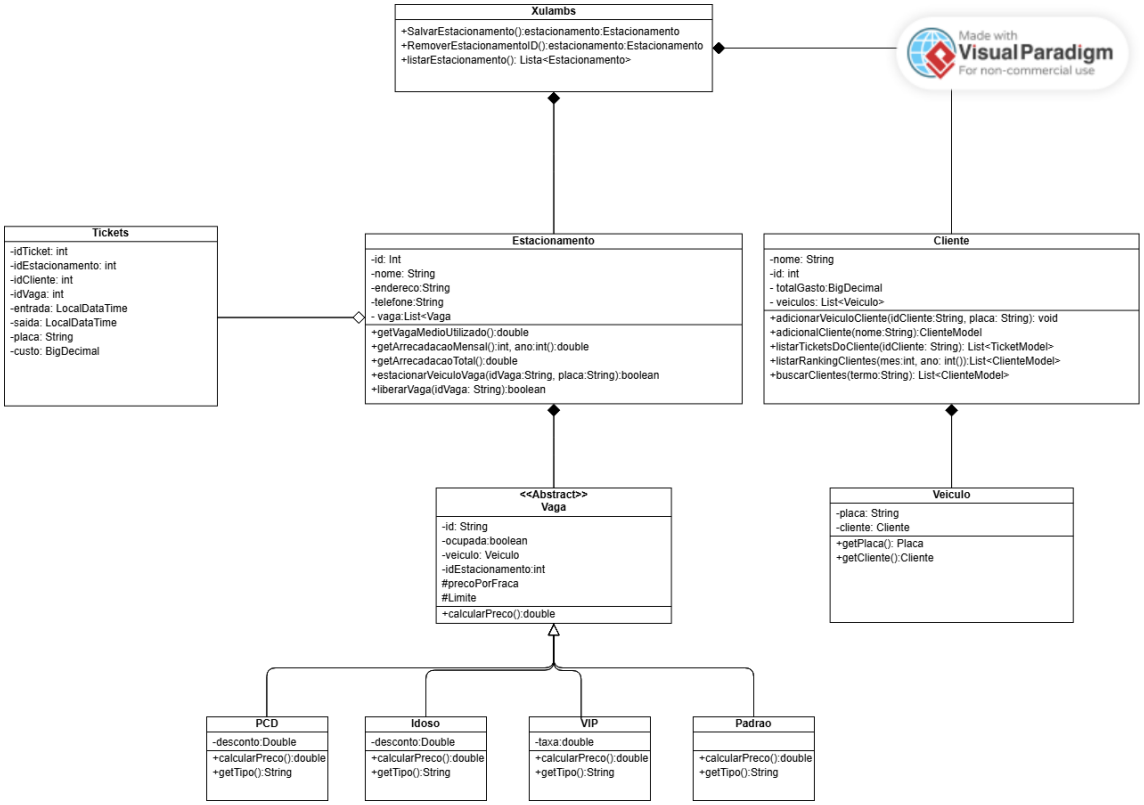
- Registro de automóveis por placa, associados a clientes cadastrados ou não identificados.
- Garantia de que cada vaga será utilizada exclusivamente por um único cliente, evitando conflitos.
- Possibilidade de vincular múltiplos veículos a um mesmo cliente.

2.2. Histórico e Relatórios de Uso

- Acesso ao histórico de utilização do estacionamento para clientes, com filtros por período de tempo.
- Relatórios administrativos que incluem:
  - Receita total gerada pelo estacionamento;
  - Ganhos por mês específico;
  - Valor médio por utilização;
  - Lista de clientes que mais contribuíram para a receita mensal.

3. Principais Consultas SQL

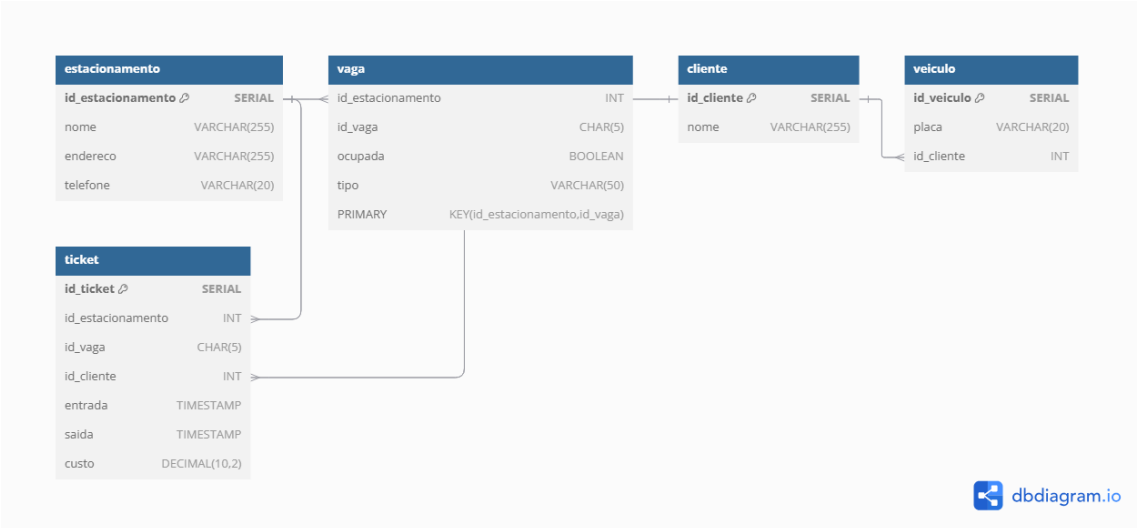
- **AdicionarEstacionamento:** Salva um novo estacionamento no banco de dados.
- **ExibirEstacionamentos:** Lista todos os estacionamentos cadastrados.
- **ExcluirEstacionamentoPorId:** Remove um estacionamento e suas dependências, como as vagas associadas, usando o ID do estacionamento.
- **RegistrarVagas:** Salva uma lista de vagas em um estacionamento específico.
- **LocalizarEstacionamentoPorId:** Busca um estacionamento pelo seu ID.
- **ListarVagasPorEstacionamento:** Lista todas as vagas de um estacionamento específico.
- **ConsultarRankingEstacionamentos:** Retorna o ranking de estacionamentos baseado no total faturado por cada um.
- **RankingPorUtilizacao:** Retorna o ranking dos estacionamentos com base na quantidade de utilizações.
- **ObterArrecadacaoTotal:** Calcula o total arrecadado em um estacionamento específico.
- **CalcularMediaUtilizacao:** Calcula o valor médio das utilizações de um estacionamento.
- **ArrecadacaoMensal:** Calcula o total arrecadado mensalmente em um estacionamento.
- **CriarTicket:** Salva um novo ticket no banco de dados, registrando a entrada de um veículo.
- **ObterTicketsAtivos:** Busca todos os tickets ativos (sem saída registrada) de um estacionamento.
- **ConsultarVeiculoPorVaga:** Busca a placa do veículo que está em uma vaga específica.
- **RegistrarSaidaVeiculo:** Registra a saída de um veículo de uma vaga e calcula o custo do ticket.
- **ObterTicketsFechados:** Busca todos os tickets fechados (com saída registrada) de um estacionamento.
- **TicketPorVaga:** Busca o ticket associado a uma vaga específica.
- **VerificarTicketAberto:** Verifica se há algum ticket aberto (sem saída registrada) para um veículo.



O diagrama Entidade-Relacionamento (DER) modela o banco de dados usado pelo sistema. Ele organiza dados em tabelas como Estacionamento, Vaga, Cliente, Veículo e Ticket. O DER define relacionamentos claros, como a associação entre clientes e veículos, e entre tickets e vagas, garantindo integridade e consistência nos dados. Chaves primárias e estrangeiras são utilizadas para estruturar consultas eficientes, permitindo rastrear ocupação de vagas, histórico de uso e receitas.

4. Diagrama de Classes Final

5. Diagrama Entidade-Relacionamento (DER)



O diagrama de classes representa a estrutura do sistema para gerenciar um estacionamento, destacando as principais entidades: Estacionamento, Vaga, Cliente, Veículo e *Ticket*. Ele detalha as relações entre essas classes, bem como os atributos e métodos necessários para o funcionamento do sistema.

6. Decisões do Projeto

- Utilização do MySQL como banco de dados relacional, devido à sua confiabilidade em transações e desempenho robusto.
- Implementação do padrão MVC para garantir maior organização e facilidade na manutenção do sistema.
- Emprego de estruturas de dados em Java, como `ArrayList`, para permitir manipulação eficiente e dinâmica de informações.
- Criação de exceções personalizadas, como `ClienteDAOException` e `EstacionamentoDAOException`, para melhorar o tratamento de erros no sistema.

7. Relatos Pessoais

Durante o desenvolvimento do projeto, tive a oportunidade de aplicar diversos conceitos que foram discutidos ao longo da disciplina, especialmente em relação ao uso de banco de dados e organização de sistemas. Este foi um processo desafiador, mas extremamente enriquecedor, no qual aprendi a lidar com várias dificuldades que surgiram ao longo do caminho.

7.1. Primeiro Desafio: Organização do Código e Aplicação do Padrão MVC

Inicialmente, uma das minhas maiores dificuldades foi organizar o código de forma eficiente. Percebi que, ao não utilizar corretamente o padrão **MVC (Model-View-Controller)**, o sistema se tornava difícil de manter e a comunicação entre as camadas era um tanto confusa. Após revisar os conceitos e reorganizar o código, a aplicação do padrão MVC não só melhorou a clareza do sistema, mas também facilitou a implementação de novos recursos e a manutenção do código.

7.2. Segundo Desafio: Trabalhando com Banco de Dados

Outro desafio que enfrentei foi relacionado às **consultas SQL**. Ao lidar com um volume maior de dados, percebi que algumas consultas estavam lentas e afetando a performance do sistema. Após estudar mais sobre **índices** e **joins eficientes**, consegui otimizar as consultas, tornando o sistema mais rápido e eficaz. A aplicação de boas práticas no banco de dados, como normalização e o uso adequado de chaves primárias e estrangeiras, também contribuiu para a integridade e consistência dos dados.

7.3. Resultados Alcançados

Após implementar as mudanças sugeridas, o sistema se tornou muito mais robusto e eficiente. A organização do código facilitou a adição de novas funcionalidades, enquanto a otimização do banco de dados garantiu uma performance mais rápida, mesmo com o aumento do número de usuários. A implementação de práticas de segurança ajudou a proteger os dados e a evitar vulnerabilidades, um aspecto essencial em qualquer sistema.

8. Principais Dificuldades

- Otimização de Consultas SQL: Desafios em escrever consultas SQL eficientes para grandes volumes de dados, garantindo um bom desempenho nas operações de leitura e escrita.

- Gerenciamento de Conexões com o Banco de Dados: Problemas relacionados ao gerenciamento de conexões simultâneas no MySQL, especialmente em sistemas com muitos usuários ou alto volume de transações.
- Escalabilidade do Banco de Dados: Dificuldade em planejar e implementar uma arquitetura escalável para o banco de dados, especialmente ao considerar a necessidade de crescimento do sistema no futuro.
- Garantir a Segurança dos Dados: Implementação de mecanismos de segurança, como criptografia de dados sensíveis e prevenção contra ataques de SQL injection, para garantir a integridade e a privacidade das informações.
- Sincronização de Dados entre Ambientes: Dificuldade em manter a consistência dos dados entre diferentes ambientes (desenvolvimento, homologação e produção), especialmente em cenários de múltiplos desenvolvedores trabalhando simultaneamente.
- Gerenciamento de Transações Complexas:Desafios ao lidar com transações longas ou complexas, que exigem controle preciso sobre o início, meio e término da operação, evitando problemas de deadlock ou inconsistências.
- Backup e Recuperação de Dados: Garantir que os dados sejam regularmente backupados e que um plano de recuperação esteja em vigor, para prevenir perdas de dados em caso de falhas no sistema ou corrupção de dados.

9. Principais Aprendizados

- Compreender a importância de um modelo de dados bem estruturado e as melhores práticas para definição de chaves e relacionamentos entre as tabelas do banco de dados.
- Aprender a configurar e gerenciar transações no MySQL, garantindo a integridade dos dados em situações de falhas e mantendo a confiabilidade do sistema.
- Adquirir experiência na aplicação de boas práticas de programação, como o uso do padrão MVC, para garantir que o código seja modular, reutilizável e fácil de manter.
- Perceber a relevância de testar e validar todas as operações de banco de dados antes da integração final, a fim de evitar inconsistências ou erros durante a execução do sistema.

10. Sugestões De Mudança

Sugere-se aprimorar a organização do código, reforçando a aplicação do padrão **MVC** para garantir uma maior separação de responsabilidades, facilitando a manutenção e a escalabilidade do sistema. Além disso, recomenda-se otimizar as consultas ao banco de dados, utilizando técnicas como índices, **joins eficientes** e análise de performance para tornar as operações mais rápidas e eficazes. A implementação de transações bem estruturadas também pode contribuir para garantir a integridade dos dados, especialmente em operações que envolvem múltiplas etapas. Por fim, a aplicação de boas práticas de **segurança**, como controle de acesso e prevenção contra injeções **SQL**, é essencial para proteger o sistema e os dados dos usuários.