Projeto de criação de um sitema de Estacionamentos Interface Gráfica Java

Diogo Caribe[1], **Lucas Ferreira**[2], **Paulo Assis**[3], **Pedro Duarte**[4], **Pedro Maia**[5]

[1]Instituto de Informática – Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Campus Coração Eucarístico – 30535-901 – Belo Horizonte – MG – Brasil

Abstract. *This project was developed as part of the Modular Programming course, with the aim of creating a Parking Management System for the company Xumelabs. The "Dog of XumeLabs" team developed the system using Java, with data storage in MySQL. The system allows for the registration of parking lots, customers and vehicles, as well as managing parking spaces and usage history. Its aim is to optimize parking lot management and improve the user experience through an intuitive graphical interface.*

Resumo. *Este projeto foi desenvolvido na disciplina de Programação Modular, com o objetivo de criar um Sistema de Gerenciamento de Estacionamentos para a empresa Xumelabs. A equipe "Cão do XumeLabs" desenvolveu o sistema utilizando Java, com armazenamento de dados em MySQL. O sistema permite o cadastro de estacionamentos, clientes e veículos, além de gerenciar vagas de estacionamento e histórico de utilização. Seu objetivo é otimizar a gestão de estacionamentos e melhorar a experiência dos usuários por meio de uma interface gráfica intuitiva.*

1. Evolução e Desenvolvimento de um Sistema de Gerenciamento de Estacionamentos

Este projeto foi desenvolvido por alunos do segundo período de Engenharia de Software da PUC Minas, na disciplina de Laboratório de Programação Modular. O objetivo foi criar um Sistema de Gerenciamento de Estacionamentos, evoluindo ao longo de quatro sprints. O sistema iniciou com funcionalidades básicas no terminal e, com o tempo, incorporou o padrão MVC para organização do código, interface gráfica em Java e armazenamento de dados em MySQL via JDBC. Entre as funcionalidades implementadas, destacam-se o cadastro de clientes e veículos, cálculo de valores arrecadados, ranking de clientes, e análise de ocupação de vagas. O projeto também incluiu diagramas de classe, testes unitários e tratamento de exceções, garantindo robustez e escalabilidade ao sistema.

2. Funcionalidades do Sistema

O sistema desenvolvido apresenta diversas funcionalidades que permitem uma gestão eficiente do estacionamento. A seguir, descrevem-se as principais funcionalidades implementadas:

- **Cadastro de Estacionamento**: Permite registrar um novo estacionamento, informando seu nome e o número de vagas disponíveis.
- **Cadastro de Cliente**: Possibilita o cadastro de clientes no sistema, onde devem ser informados o nome e o CPF.

- **Cadastro de Veículo**: Permite registrar veículos no sistema, associando-os ao CPF do proprietário e à placa do veículo.
- **Estacionar Veículo**: O usuário pode selecionar uma vaga disponível e informar a placa do veículo. Caso o veículo não esteja associado a um cliente, é possível estacioná-lo de forma anônima.
- **Desocupar Vaga**: Exibe os usos de vaga em aberto para que o usuário escolha qual deseja desocupar. Após a escolha, o pagamento é realizado e a vaga é liberada.
- **Histórico de Uso de Vagas**: A funcionalidade permite consultar os usos de vaga de um cliente em um período específico. O usuário deve informar o CPF do cliente, a data de início e a data de fim para obter o histórico referente ao período.
- **Valor Arrecadado**: Gera relatórios financeiros baseados em um mês e ano selecionados. O sistema exibe o valor arrecadado no mês, o total acumulado e a média arrecadada de um estacionamento específico.
- **Ranking de Clientes**: Apresenta, de forma ordenada, os clientes que mais gastaram em um mês específico. O usuário deve informar o mês e o ano para gerar o ranking.
- **Vagas Utilizadas por Período do Dia**: Permite selecionar um período do dia (manhã, tarde ou noite), além do mês e ano, para visualizar as vagas utilizadas. O sistema exibe o tipo de vaga (idoso, PCD, VIP ou padrão), a quantidade de vezes que cada tipo foi utilizada e o valor arrecadado correspondente.
- **Taxa de Ocupação por Tipo de Vaga**: Disponibiliza informações detalhadas sobre a utilização de vagas por tipo. Após selecionar o mês e o ano, o sistema exibe o tipo de vaga, a quantidade utilizada, a porcentagem de ocupação e o valor arrecadado. Essa funcionalidade oferece uma visão geral do desempenho do negócio.

Essas funcionalidades foram projetadas para atender às principais necessidades de um sistema de gestão de estacionamentos, proporcionando eficiência operacional e facilidade de uso.

3. Principais Consultas SQL e Suas Finalidades

Segue o detalhamento de duas consultas principais implementadas no sistema:

- **Taxa de Ocupação por Tipo de Vaga:**

```
SELECT v.tipo_vaga, COUNT(uv.numeroVaga) AS vagas_usadas,
       (COUNT(uv.numeroVaga) * 100.0 /
        (SELECT COUNT(uv2.numeroVaga)
        FROM usodevaga uv2
        WHERE uv2.status = ? AND
        uv2.nome_estacionamento = uv.nome_estacionamento))
       AS porcentagem_utilizacao,
       SUM(uv.valoraPagar) AS valor_total_arrecadado
FROM usodevaga uv
JOIN vaga v ON uv.numeroVaga = v.numeroVaga
WHERE uv.nome_estacionamento = ? AND
```

```
        v.nome_estacionamento = ? AND
        uv.status = ? AND
        MONTH(uv.data) = ? AND YEAR(uv.data) = ?
GROUP BY v.tipo_vaga
ORDER BY porcentagem_utilizacao DESC;
```

Finalidade: Calcula a taxa de utilização de cada tipo de vaga em um estacionamento, retornando quantidade de vagas usadas, porcentagem de ocupação e valor total arrecadado no período.

- **Vagas Utilizadas por Período do Dia:**

```
SELECT v.tipo_vaga, COUNT(uv.numeroVaga) AS vagas_usadas,
       SUM(uv.valoraPagar) AS valor_arrecadado
FROM usodevaga uv
JOIN vaga v ON uv.numeroVaga = v.numeroVaga
WHERE uv.nome_estacionamento = ? AND
      v.nome_estacionamento = ? AND
      uv.status = ? AND
      HOUR(uv.horachegada) BETWEEN ? AND ? AND
      HOUR(uv.horasaida) BETWEEN ? AND ? AND
      MONTH(uv.data) = ? AND YEAR(uv.data) = ?
GROUP BY v.tipo_vaga;
```

Finalidade: Retorna o uso de vagas em períodos específicos do dia (manhã, tarde, noite), listando tipo de vaga, quantidade ocupada e valor arrecadado.

4. Diagrama de Classe Final

O diagrama de classes evoluiu ao longo do desenvolvimento e é composto pela classe principal `Vaga`, com os atributos `disponível` e `numeroVaga`, e o método `alterarDisponibilidade`. As classes filhas `VagaVip`, `VagaDefault`, `VagaIdoso` e `VagaPCD` herdam de `Vaga`, aplicando variações nas taxas e descontos. A classe `Estacionamento` gerencia o nome e o número de vagas, e a classe `UsoDeVaga` associa `Vaga` e `Veiculo`. A classe `Veiculo` contém o atributo `placa` e pode estar vinculada a um cliente, que possui um CPF e nome, podendo ter múltiplos veículos.

5. Discussão sobre as Escolhas de Implementação

Diversas escolhas técnicas foram feitas para garantir o desempenho, escalabilidade e facilidade de manutenção do sistema de gerenciamento de estacionamentos.

5.1. Escolha do SGBD: MySQL

A escolha do MySQL se deu pela sua robustez, alta disponibilidade e integração com Java, permitindo realizar consultas complexas e armazenar dados relacionais de forma eficiente, como o cálculo de arrecadação de estacionamentos.

5.2. Padrão Arquitetural MVC

Adotamos o padrão MVC (Modelo-Visão-Controlador) para separar claramente as responsabilidades, facilitando a manutenção e escalabilidade do sistema. O Modelo gerencia dados, a Visão a interface e o Controlador interage entre eles.

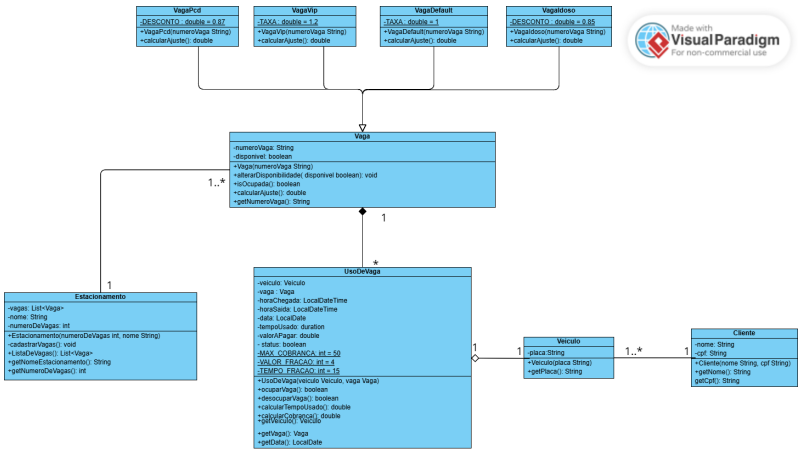

Figure 1. Diagrama de Classe Final

5.3. Uso de Collections: Map, List e ArrayList

Utilizamos o `Map` para associar identificadores a valores, como placas de veículos e status de vagas, e `List` e `ArrayList` para armazenar dados dinâmicos, como clientes e veículos.

5.4. Tratamento de Exceções

A implementação de `try-catch` garantiu a robustez do sistema ao capturar erros, como falhas no banco de dados ou entradas inválidas, além de criar exceções personalizadas para uma gestão de fluxo mais eficiente.

6. Diagrama de Entidade e Relacionamento

O Diagrama de Entidade e Relacionamento (DER) a seguir ilustra as entidades e seus relacionamentos no sistema de gerenciamento de estacionamentos.

Figure 2. Diagrama de Entidade e Relacionamento

7. Principais Problemas Enfrentados

Durante o desenvolvimento do projeto, os principais desafios foram a adoção da programação orientada a objetos, que ainda não havia sido utilizada, e a criação das classes controladoras, um conceito novo para a equipe. Além disso, a modelagem do projeto exigiu cuidado para garantir a separação adequada das responsabilidades entre as classes, evitando o acoplamento excessivo.

8. Principais Aprendizados

Entre os principais aprendizados, destacam-se o uso do JDBC para integração com o banco de dados, o desenvolvimento de projetos com interface gráfica, a implementação de persistência de dados em bancos de dados, a prática de trabalho em equipe e a modelagem de dados, que contribuíram significativamente para o aprimoramento das habilidades de desenvolvimento.

9. Sugestões de Mudança no Diagrama da Solução

9.1. Integração com API de Notificação

Uma possível melhoria seria integrar o sistema com uma **API de notificações em tempo real**, como **Firebase Cloud Messaging (FCM)**. Isso permitiria enviar atualizações automáticas sobre o status das vagas, como mudanças de disponibilidade ou alertas para clientes, sem necessidade de acesso constante ao sistema.

9.2. Sistema de Recomendação de Vagas

Outra sugestão seria implementar um **sistema de recomendação** de vagas, utilizando **aprendizado de máquina**. O sistema poderia sugerir a vaga mais adequada ao cliente com base em seu histórico e preferências, otimizando a alocação de vagas e melhorando a experiência do usuário.

10. Relatos Pessoais

Lucas Ferreira: Durante o desenvolvimento, pude trocar conhecimentos com meu grupo, onde cada um contribuiu de forma significativa. Enfrentamos diversos problemas que me ajudaram a encontrar soluções práticas, o que foi valioso para minha preparação no mercado de trabalho, fornecendo uma visão realista dos desafios no desenvolvimento de sistemas.

Paulo Assis: Este projeto me proporcionou uma evolução significativa na programação e na compreensão da arquitetura de sistemas. Os desafios enfrentados contribuíram para o meu crescimento pessoal e profissional, e as habilidades adquiridas serão essenciais para minha trajetória profissional, agregando valor à minha formação.

Pedro Duarte: Foi muito legal passar por situações com o grupo onde achávamos que o trabalho estaria perdido, mas que no final deu tudo certo e que todos puderam cooperar para terminarmos.

Diogo Caribé: O desenvolvimento do JavaParking foi uma experiência desafiadora, com altos e baixos. Em alguns momentos, precisei seguir em frente com a implementação sem entender todos os detalhes, o que me ensinou a resolver problemas de forma prática. Apesar das dificuldades, o projeto foi importante para o meu aprendizado.

Pedro Maia: O Trabalho foi uma experiência muito proveitosa, consegui aprender bastante a matéria, além de aprender a trabalhar em equipe com meu grupo, o resultado ficou muito satisfatório!