Relatório do Sistema de Gerenciamento de Estacionamento Xulambs Parking

Lucas Valente[1], **Bernardo Ferreira 2**[1], **Thiago Henrique 3**[1]

[1]Instituição de Ensino
Engenharia De Software - PUC MINAS

Abstract. *Este relatório descreve a implementação de um sistema de gerenciamento de estacionamentos para a Xulambs Parking, desenvolvido em Java com padrão MVC e interface gráfica em Java Swing. O sistema atende a requisitos de controle de vagas por tipo, cadastro de clientes e veículos, e cobrança por tempo de uso. São apresentadas as principais funcionalidades, consultas SQL, diagramas de classes e ER, decisões de projeto, relato de experiência e aprendizados da equipe.*

Resumo. *Este relatório apresenta o desenvolvimento de um sistema para controle de estacionamentos da Xulambs Parking, usando Java com MVC e interface Swing. Abordamos: visão geral, consultas SQL, diagramas, decisões de projeto, experiências da equipe, desafios e aprendizados.*

1. Visão Geral das Funcionalidades Implementadas

O sistema Xulambs Parking gerencia múltiplos estacionamentos e permite a alocação de vagas com base em tipos e descontos específicos. Entre as funcionalidades essenciais: - Cadastro de Clientes e Veículos: os clientes podem ser identificados ou anônimos, com opção de múltiplos veículos. Cada cliente é identificado com nome e ID, enquanto veículos são registrados por placa. - Gerenciamento de Vagas por Tipo: a aplicação controla vagas regulares, VIP, para idosos e PCD, com regras distintas de cobrança e alocação conforme o tipo de vaga e desconto correspondente. - Controle de Ocupação de Vagas: implementado com verificação de disponibilidade e bloqueio de vaga ocupada, prevenindo duplicidade. - Cobrança e Tarifação: baseado em frações de tempo (R$4 a cada 15 minutos, limitado a R$50), com aplicação de descontos conforme o tipo de vaga ocupada. - Relatórios Gerenciais: inclui consulta de histórico de uso por cliente, total arrecadado, média de uso, e ranking dos clientes por arrecadação em um período específico.

2. Principais Consultas SQL

As consultas SQL foram projetadas para atender às principais necessidades do sistema de estacionamento:

- Verificar se o veículo está estacionado: `SELECT MAX(PLACA_VEICULO = ? AND SAIDA = 'null') AS 'ESTA_ESTACIONADO' FROM USOVAGA;` Permite identificar se um veículo específico ainda está no estacionamento.

- Histórico de utilização por veículo: `SELECT ID_VAGA, ENTRADA, SAIDA FROM USOVAGA WHERE NOME_ESTACIONAMENTO = ? AND PLACA_VEICULO = ?;` Retorna o histórico de entradas e saídas de um veículo em determinado estacionamento.
- Verificar se o veículo está cadastrado: `SELECT MAX(PLACA = ?) AS 'POSSUI_VEICULO' FROM VEICULOS;` Confirma se o veículo com a placa informada está registrado no sistema.
- Listar todas as vagas de um estacionamento: `SELECT ID, ESTADO FROM VAGAS WHERE NOME_ESTACIONAMENTO = ?;` Retorna o estado atual de todas as vagas de um estacionamento específico.
- Buscar veículos de um cliente: `SELECT PLACA FROM VEICULOS WHERE ID_CLIENTE = ?;` Lista todas as placas de veículos associados a um cliente.
- Verificar a existência de um cliente: `SELECT COUNT(*) FROM CLIENTES WHERE id = ?;` Confirma se um cliente com o ID fornecido está cadastrado.
- Veículo atualmente em uma vaga: `SELECT PLACA, ENTRADA FROM USOVAGA WHERE ID = ? AND NOME_ESTACIONAMENTO = ? AND SAIDA = 'null';` Retorna a placa e a hora de entrada do veículo que está ocupando uma vaga específica.

3. Diagrama de Classes Final

O diagrama de classes ilustra a estrutura e as relações principais do sistema. A classe `Estacionamento` gerencia vagas (`Vaga`), clientes (`Cliente`) e veículos (`Veiculo`), conectando-os às operações (`Utilizacao`). Além disso, subtipos de `Vaga` foram definidos para representar as especificidades dos tipos VIP, PCD e idosos, com diferenciação nas regras de tarifação e alocação.

4. Diagrama Entidade-Relacionamento

Utilizando o dbdiagram, o Diagrama ER mapeia as entidades `Clientes`, `Veiculos`, `Vagas`, e `Utilizacoes`, conectando-as de maneira a refletir as funcionalidades e consultas SQL. Esse diagrama foi desenvolvido para manter a integridade e consistência dos dados com chaves estrangeiras entre as entidades.

5. Decisões de Projeto

Para a implementação, utilizamos:

- Tratamento de erros e exceções com mensagens personalizadas. Quando um erro ou exceção é identificado (por exemplo, ao tentar acessar um arquivo inexistente ou conectar-se a um banco de dados), um tratamento de erro é acionado para lidar com a falha. Com mensagens personalizadas, podemos fornecer informações mais claras e específicas sobre o erro, em vez de mensagens genéricas ou técnicas.
- Collections: foram utilizados `ArrayList` para manipulação de listas temporárias de clientes e vagas, e `HashMap` para consultas rápidas por ID.
- Padrão Singleton: para validação de ocupação de vagas e duplicidade de veículos, melhorando o controle de erros e a segurança da aplicação.

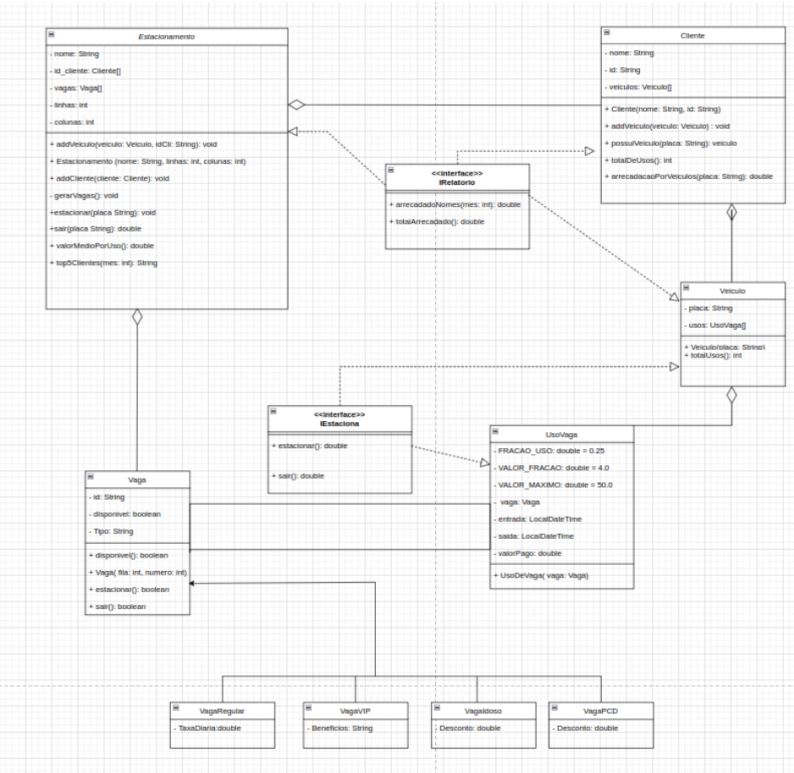

Figure 1. Diagrama de Classes UML

```java
public boolean registrar(String placa, String idVaga) {  1 usage
    if(new VeiculoDAO().veiculoRegistrado(placa)) {
        if (!new UsoDeVagaDAO().estaEstacionado(placa)) {
            new UsoDeVagaDAO().cadastrarUsoDeVaga(estacionamentoAtual.estacionar(placa, idVaga), estacionamentoAtual.getNome(), placa);
            new VagaDAO().atualizarEstadoVaga(idVaga, estacionamentoAtual.getNome());
            return true;
        } else {
            view.exibeMensagem("Esse veículo já está estacionado! Tente outro veículo.");
            view.getTextFieldPlaca().setText("");
            return false;
        }
    } else {
        if(view.exibeDialogo( mensagem: "Placa não encontrada. Deseja registrar um novo veículo?") == JOptionPane.YES_OPTION) {
            new RegistrarVeiculos(estacionamentoAtual, placa).setVisible(true);
        }
        return false;
    }
```

Figure 2. Tratamento de erros e exceções

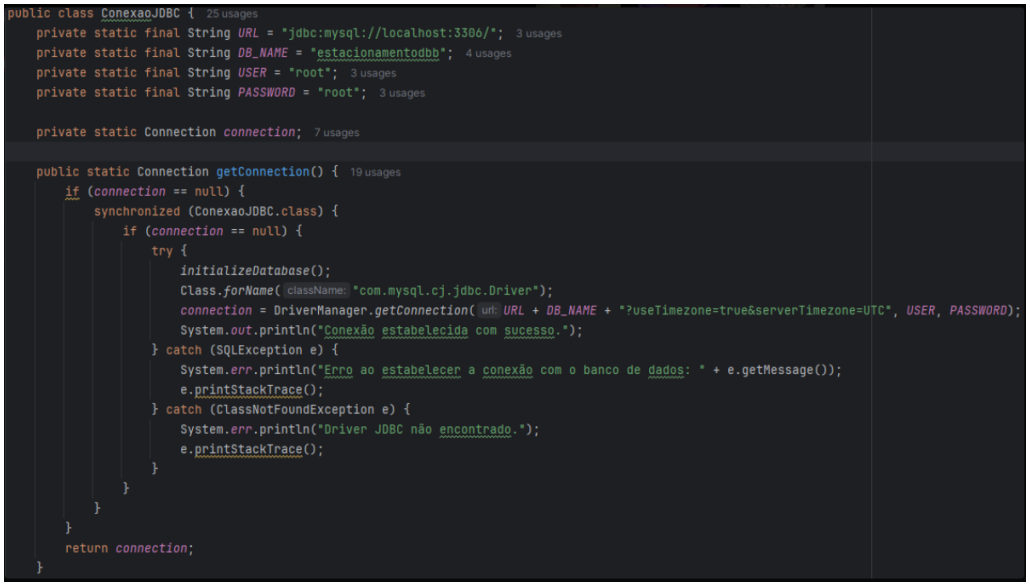

Figure 3. Padrão Singleton

6. Relato de Experiência Pessoal

O desenvolvimento deste sistema foi uma experiência desafiadora, mas extremamente gratificante. Tivemos a oportunidade de consolidar conhecimentos importantes sobre a arquitetura MVC, que nos ajudou a estruturar o projeto de forma organizada e eficiente. Trabalhar com o Swing para criar a interface gráfica foi uma jornada de aprendizado prático, onde enfrentamos dificuldades para alinhar usabilidade e funcionalidade, mas o resultado nos deixou muito orgulhosos.

Um dos maiores desafios foi modelar o sistema para atender aos requisitos do negócio, o que exigiu bastante reflexão e ajustes ao longo do caminho. Essa etapa nos ensinou muito sobre como traduzir demandas reais em soluções de software e como a persistência de dados é essencial para garantir que tudo funcione sem problemas.

Foi um processo que nos fez crescer tanto tecnicamente quanto pessoalmente, reforçando habilidades como trabalho em equipe, resolução de problemas e atenção aos detalhes. O mais recompensador foi ver a ideia sair do papel e se transformar em algo funcional e útil, capaz de resolver problemas reais.

7. Principais Dificuldades

Dentre os desafios: - Implementação de Regras de Tarifação e Descontos: O principal desafio nessa área foi lidar com a diversidade de regras específicas para diferentes tipos de vagas, como vagas VIP, comuns, para deficientes, entre outras. Cada tipo possuía tarifas distintas e condições específicas de descontos, como gratuidades em períodos curtos ou valores reduzidos para determinados perfis de clientes. Foi necessário implementar uma lógica robusta que parametrizasse essas regras de forma flexível, mas ao mesmo tempo validasse as condições corretamente para evitar erros de cobrança. Além disso, garantir que essas regras fossem facilmente atualizáveis sem afetar o funcionamento do sistema foi um aprendizado importante.

- Concorrência de Vagas: A gestão da concorrência foi desafiadora, pois pre-

cisávamos garantir que uma vaga não fosse alocada simultaneamente para mais de um veículo. Para isso, desenvolvemos um sistema de bloqueio que validava a disponibilidade antes de registrar uma nova ocupação. Além disso, foi necessário implementar testes rigorosos para situações de alta demanda, como horários de pico, para evitar inconsistências causadas por operações concorrentes. Este desafio nos proporcionou um aprendizado valioso sobre controle de transações e uso de locks no banco de dados.

- Configurar Driver do MYSQL: Houve dificuldades técnicas na configuração do driver do MySQL, especialmente devido a conflitos entre diferentes versões do driver e do banco de dados. Esses conflitos geraram problemas de compatibilidade, que resultaram em erros de conexão e instabilidade no ambiente de desenvolvimento. Para superar isso, realizamos uma pesquisa detalhada sobre versões compatíveis e configuramos o ambiente de forma a padronizar tanto o driver quanto o banco de dados. Além disso, a experiência nos ajudou a entender a importância de documentar e replicar configurações para evitar problemas futuros.

Esses desafios não só enriqueceram nossa experiência técnica, mas também nos ensinaram a importância de planejar, testar e documentar cada etapa do desenvolvimento.

8. Principais Aprendizados

O projeto permitiu um aprendizado prático sobre: - MVC: organizar o código em camadas facilitou o desenvolvimento e a manutenção. - Consultas SQL Otimizadas: desenvolver consultas eficazes para garantir desempenho. - Java Swing: gerenciar eventos e atualizações da interface foi uma habilidade valiosa adquirida durante o desenvolvimento.

9. Sugestões de Melhorias no Diagrama da Solução

Recomenda-se criar subclasses para os diferentes tipos de vagas, facilitando a inclusão de regras específicas e uma estrutura mais organizada para expansão futura. Adicionalmente, uma entidade `HistoricoCliente` pode melhorar a eficiência na consulta e no controle dos registros de uso.

References