**Spring Boot - AWS - Ago/2024**

**Descrição do Desafio**

Desenvolva um sistema de gestão de estacionamento para um shopping, utilizando apenas Java e MySQL. O sistema deve gerenciar o número de vagas disponíveis, registrar a entrada e saída de veículos por diferentes cancelas, calcular o valor a ser pago pelo tempo de permanência e garantir que as regras específicas de entrada e saída sejam respeitadas. O sistema também deve permitir o cadastro de veículos mensalistas e caminhões de entrega, enquanto veículos avulsos e de serviço público não precisam ser previamente cadastrados.

**Requisitos do Sistema:**

1. **Cadastro de Veículos e Categorias:**
   * O sistema deve suportar o cadastro de veículos nas seguintes categorias:
     + **Mensalista:** Veículo com um plano de estacionamento recorrente. As placas dos veículos mensalistas devem ser registradas previamente para abrir automaticamente as cancelas.
     + **Caminhões de Entrega:** Veículos de carga, como caminhões de entrega e vans. As placas desses veículos devem ser registradas previamente para abrir automaticamente as cancelas.
   * **Serviço Público:** Veículos de serviço público, como ambulâncias e viaturas, que têm acesso livre, não ocupam vagas e são isentos de cobrança. Esses veículos não precisam ser cadastrados.
   * **Avulso:** Veículos que não precisam de cadastro prévio. Recebem um ticket na entrada, que será utilizado para registrar a saída e calcular o valor a ser pago.
2. **Tipos de Veículos:**
   * **Carros de Passeio**
   * **Motos**
   * **Caminhões de Entrega**
   * **Veículos de Serviço Público**
3. **Gestão das Cancelas:**
   * Cancelas de entrada são numeradas de **1 a 5**.
   * Cancelas de saída são numeradas de **6 a 10**.
   * **Mensalistas e Avulsos** podem entrar por qualquer cancela de entrada (1 a 5) e sair por qualquer cancela de saída (6 a 10).
   * **Caminhões de Entrega** só podem entrar pela **cancela 1** e sair por qualquer cancela de saída (6 a 10).
   * **Motos** só podem entrar pela **cancela 5** e sair pela **cancela 10**.
   * **Veículos de Serviço Público** têm acesso livre em todas as cancelas sem cobrança.
4. **Gestão de Vagas:**
   * O sistema deve controlar o número total de vagas disponíveis, numerando-as para facilitar o gerenciamento.
   * Defina O limite máximo de vagas disponíveis no estacionamento é 500 vagas.
   * Reserve 200 vagas para os mensalistas.
   * **Moto** ocupa 1 vaga, **Carro de Passeio** 2 vagas, **Caminhão de Entrega** 4 vagas.
5. **Registro de Entrada e Saída através de Tickets:**
   * **Mensalistas e Caminhões de Entrega:** A entrada e saída são registradas automaticamente com base na placa do veículo. Não é necessário ticket.
   * **Avulsos:** Recebem um **ticket** ao entrar no estacionamento, que registra:
     + A **hora de entrada**.
     + A **hora de saída** (registrada no momento da saída).
     + A **cancela** de entrada e saída.
     + A **vaga** ocupada (número da vaga).
     + O **valor a ser pago** (calculado no momento da saída).
   * Os dados do ticket devem ser persistidos no banco de dados MySQL.
   * **Veículos de Serviço Público** não recebem **ticket**.
6. **Cobrança:**
   * Implementar um sistema de cálculo de valor baseado no tempo que o veículo permaneceu no estacionamento, por vaga ocupada.
   * Valor por minuto estacionado: R$ 0,10
   * Cobrança mínima de R$ 5,00 para veículos avulsos.
   * **Mensalistas** pagam uma taxa fixa mensal de R$ 250,00.
   * **Veículos de Serviço Público** são isentos de cobrança.

**Critérios de Avaliação:**

1. **Corretude:**
   * Verificar se o sistema respeita as regras de negócio para cada tipo de veículo e cancela.
   * Conferir se o cálculo do valor devido é feito corretamente, incluindo a aplicação da cobrança mínima para veículos avulsos.
2. **Estrutura do Código:**
   * Avaliar se o código está bem organizado, com classes bem definidas e métodos que seguem boas práticas de programação.
3. **Persistência:**
   * Verificar se os dados são corretamente persistidos no banco de dados e se as operações de leitura e escrita estão funcionando conforme esperado.
4. **Documentação:**
   * A documentação do código deve ser clara, com comentários que expliquem a lógica de cada parte do sistema.
5. **Teste:**
   * Testar se o sistema funciona corretamente para diferentes cenários, como entrada e saída de veículos em diferentes ordens e situações, incluindo o gerenciamento de vagas numeradas e limitadas.