**Desafio – Teste Prático para Desenvolvedor Java**

**1. Objetivos do Teste**

* **Avaliar competências técnicas:** Verificar o domínio do Java (versão 8 +) ou qualquer versão acima, uso de frameworks (por exemplo, Spring Boot ou Java EE) e manipulação de dados via JPA/Hibernate, ou o que quiser utilizar.
* **Validação de regras de negócio:** Simular um cenário real, onde o candidato deve implementar um sistema que gerencie “vagas de estacionamento, permita reservas, libere vagas e calcule o custo com base no tempo de locação”, ou o tema que quiser trabalhar, aqui de novo foi só um exemplo.
* **Organização e documentação:** Será analisado se o código é modular, limpo e bem documentado (README com instruções de build, execução e testes).

**“ O que eu preciso fazer? “**

* 1. Faça o download do *Teste-Pratico-Desenvolvedor-Java.zip* com o projeto base para seu desenvolvimento. “Más pode fazer um do zero, este é para adiantar”.
* 2. Implemente o desafio conforme abaixo.
* 3. Quando finalizar seu projeto, faça um push para um repositório de sua conta no GitHub.
* 4. Grave um vídeo explicando sobre as decisões que você tomou para implementação deste desafio.
* 5. Envie um e-mail avisando que finalizou o desafio com a url do seu repositório, link do vídeo que foi gravado e aguarde nosso contato.

**2. Escopo e Funcionalidades**

**“Sistema de Locação de Vagas de Estacionamento”.**

Este é só um tema sugerido por nós, podes usar o tema que quiser, é só utilizar os exemplos de requisitos abaixo.

**Funcionalidades Principais:**

1. **Gerenciamento de Vagas de Estacionamento:**
   * Cadastro de vagas: Cada vaga deve ter identificador, número ou código, tipo (ex.: comum, VIP), valor da hora e status (disponível, reservada ou ocupada).
   * Consulta de vagas disponíveis.
2. **Reserva e Locação:**
   * Permitir que um usuário (pode ser simplificado sem autenticação) reserve uma vaga disponível.
   * Registrar o início da locação (data/hora de início).
   * Permitir o encerramento da locação, onde o sistema calculará o valor total com base na duração e no valor da hora da vaga.
3. **Regras de Negócio:**
   * Impedir que uma vaga reservada seja reservada novamente.
   * Validar que, ao encerrar a locação, o tempo mínimo ou as condições necessárias sejam atendidos para o cálculo correto do custo.
   * Atualizar o status da vaga automaticamente após a liberação (retornando a “disponível”).

**Funcionalidades Extras (Opcional)**

* Implementação de testes unitários (usando JUnit e, se necessário, Mockito).
* Criação de uma pequena interface (pode ser via endpoints REST) que permita a interação com o sistema.
* Documentação das decisões arquiteturais e dos desafios encontrados.

**Se por acaso preferir fazer de outro tema, escolha um tema com um escopo de funcionalidades igual ou maior que o designado neste documento!**

**3. Tecnologias Sugeridas**

* **Linguagem e Frameworks:**
  + Java (versão 8 ou superior)
  + Spring Boot (ou Java EE com JAX-RS, conforme orientação)
* **Persistência:**
  + Hibernate/JPA para mapeamento objeto-relacional
  + Banco de dados H2 (para facilitar a execução local) ou outro BD de sua escolha
* **Gerenciamento de Dependências:**
  + Maven ou Gradle
* **Testes:**
  + JUnit (e Mockito, se aplicável)
* **Versionamento:**
  + Git (preferencialmente com repositório no GitHub)

Mas esteja livre para utilizar outras tecnologias, só não esqueça de explanar seu funcionamento no projeto.

**4. Modelagem e Estrutura de Dados**

**Entidades Sugeridas**

1. **ParkingSpot (Vaga de Estacionamento):**
   * **id:** Identificador único (gerado automaticamente)
   * **numero:** Número ou código da vaga
   * **tipo:** Tipo da vaga (ex.: “comum”, “VIP”)
   * **valorPorHora:** Valor da locação por hora
   * **status:** Enum com valores (DISPONÍVEL, RESERVADA, OCUPADA)
2. **Reservation (Reserva/Locação):**
   * **id:** Identificador único
   * **parkingSpot:** Relação com a vaga reservada
   * **dataInicio:** Data e hora de início da locação
   * **dataFim:** Data e hora de término (inicialmente nula e definida no encerramento)
   * **valorTotal:** Valor calculado com base no tempo de locação

*(Opcional)* **Cliente:** Se desejar simular o cadastro de clientes, pode incluir nome, CPF, etc.

**5. Endpoints (API REST)**

Implemente uma API RESTful com os seguintes endpoints (ou similar):

* **POST /api/vagas:** Cadastrar uma nova vaga.
* **GET /api/vagas/disponiveis:** Listar todas as vagas com status DISPONÍVEL.
* **POST /api/reservas:** Criar uma reserva para uma vaga disponível, registrando o início da locação e alterando o status da vaga para RESERVADA.
* **PUT /api/reservas/{id}/encerrar:** Encerrar a reserva (ou locação), registrando a data/hora final, calculando o custo total e atualizando o status da vaga para DISPONÍVEL.

**6. Critérios de Avaliação**

* **Qualidade do Código:** Clareza, organização, arquitetura e boas práticas de desenvolvimento.
* **Funcionalidades:** Implementação correta das operações de cadastro, reserva, encerramento e cálculo do valor.
* **Testes Automatizados:** Existência e cobertura de testes unitários para métodos e serviços críticos.
* **Documentação:** README completo com:
  + Descrição do projeto.
  + Instruções de configuração e build.
  + Descreva também os passos para execução de sua aplicação.
  + Orientações sobre como rodar os testes.
  + Explicação breve das decisões arquiteturais.

**7. Instruções de Entrega**

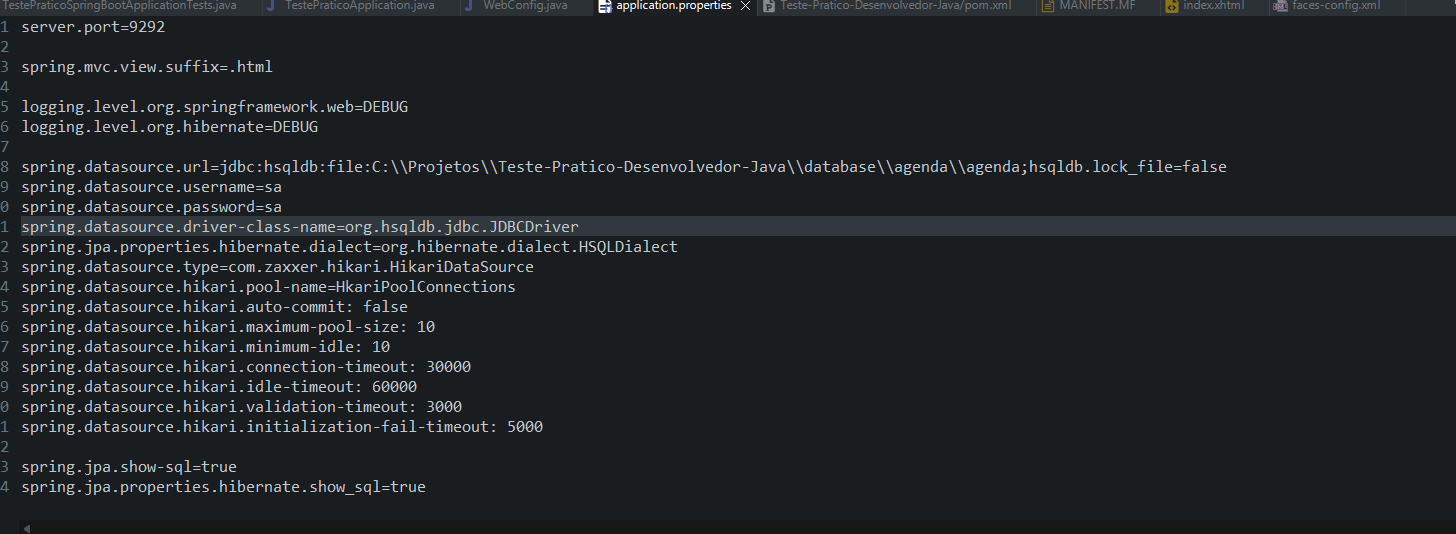
* **Prazo:** Um prazo será Definido para a entrega do teste, de acordo com o acordado pelo recrutador. (por exemplo, 3 a 5 dias).
* **Repositório:** Quando finalizar seu projeto, faça um push para um repositório de sua conta no GitHub. Git (ex.: GitHub) e compartilhe o link.
* **Apresentação das Decisões:** Grave um vídeo explicando sobre as decisões que você tomou para implementação deste desafio.
* **Feedback:** Caso não consiga entregar todas as funcionalidades extras, o que for implementado deve ser funcional e bem documentado. Informe que o foco está na qualidade do código e na clareza das soluções apresentadas.
* **Envio do Projeto:**  Envie um e-mail avisando que finalizou o desafio com a url do seu repositório, e o link do vídeo que foi gravado e aguarde nosso contato.

**8. Dicas para o Candidato:**

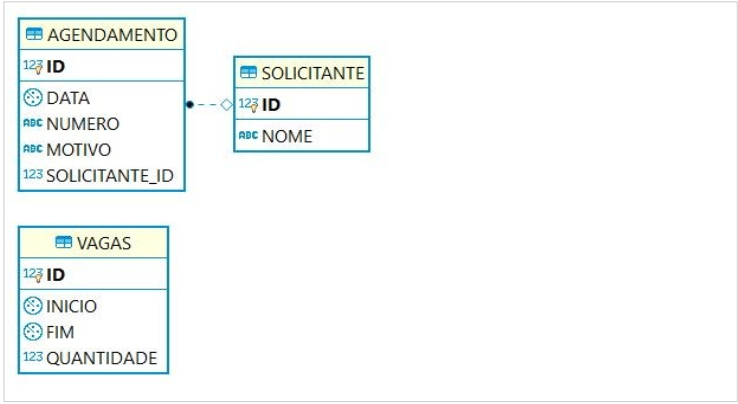
* Divida o problema em partes: Comece modelando as entidades, crie os serviços (business layer) e depois exponha a API.
* Priorize a implementação dos requisitos mínimos antes de tentar as funcionalidades extras.
* Invista tempo na escrita de testes unitários, pois eles são um diferencial importante.
* Documente brevemente suas escolhas e quaisquer desafios encontrados durante o desenvolvimento.
* Não se preocupe com autenticação ou multitenancy
* Fique à vontade para utilizar seus temas e layout de telas
* No arquivo README do projeto explique um pouco do funcionamento e a arquitetura que
* você adotou em sua implementação
* Descreva também os passos para execução de sua aplicação

**9. Esqueleto de projeto com banco HSQLB pré-configurado:**

* Com este documento vem acompanhando uma carcaça de projeto para ser implementada pelo candidato, este mesmo é para adiantar
* Ele já vem com um banco HSQLDB pré-configurado, é um banco de dados relacional escrito em Java que pode ser executado em modo de memória ou em modo de arquivo, para não precisar ficar instalando banco de dados.

****

* **Ele já vai vir com umas tabelas de exemplo.**

****

* **Configurações para Funcionamento:**
* No arquivo "application.properties" informe em *%CAMINHO\_BANCO\_HSQL\_LOCAL%* o caminho completo utilizado para acesso ao banco de dados: spring.datasource.url=jdbc:hsqldb:file:%CAMINHO\_BANCO\_HSQL\_LOCAL%;hsqldb.lock\_file=false
  + *Por exemplo: C:\\Projetos\\Teste-Pratico-Desenvolvedor-Java\\banco\\agenda*
* **Utilize o comando abaixo para compilação do projeto:**
  + mvn install -DskipTests
* **Para iniciá-lo, utilize o comando abaixo:** 
  + java -jar -Dserver.port=9494 target/Teste-Pratico-Desenvolvedor-Java-0.0.2-SNAPSHOT.jar
* **Abra o negador e acesse:**
  + http://localhost:9494

**Lembrando que o candidato pode montar o ambiente de acordo com sua estratégia.**

**Desde já Boa Sorte!**