POS TECH

TRABALHO SUBSTITUTIVO DE TECH CHALLENGE

TRABALHO SUB TECH CHALLENGE CURSO SOAT – PÓSTECH

FASE 2

Uma empresa de revenda de veículos automotores nos contratou pois quer implantar uma plataforma que funcione na internet, sendo assim, temos que criar a plataforma. O time de UX já está criando os designs, e ficou sob sua responsabilidade criar a API. O desenho da solução envolve as seguintes necessidades do negócio:

- Cadastrar um veículo para venda (Marca, modelo, ano, cor, preço);
- Editar os dados do veículo;
- Efetuar a venda de um veículo (CPF da pessoa que comprou, data da venda);
- Listagem de veículos à venda, ordenada por preço, do mais barato para o mais caro;
- Listagem de veículos vendidos, ordenada por preço, do mais barato para o mais caro;
- Disponibilizar um endpoint (webhook) para que a entidade que processa o pagamento consiga, a partir do código do pagamento, informar se o pagamento foi efetuado ou cancelado.

  Importante: nem todos os campos e funcionalidades necessárias para atender os requisitos estão descritos acima, por isso a modelagem é fundamental para entender como resolver o problema e entender o que precisa ser feito para que a solução funcione.

Outro ponto importante é fazer a documentação dos endpoints de forma adequada para que o time de frontend possa fazer a integração de forma correta. O padrão a ser usado é o OpenAPI/Swagger.

Além disso, como estamos com um novo time responsável pela infraestrutura, será necessário a implantação de Kubernetes. Foi solicitado que toda a descrição dos serviços a serem publicados sejam feitas neste padrão, usando deployment, configmap, secrets e services.

Entregáveis

- PDF contendo os links de acesso aos itens abaixo:
  - Repositório com o código-fonte do software (ver próximo item);
  - Vídeo demonstrando a solução funcionando, tanto na implementação da aplicação quanto na infraestrutura Kubernetes.
- Conteúdo do Repositório:
  - Arquivo Readme.md que explique o que é o projeto, como foi implementado, como usar localmente e como testar;

- Código-fonte de software que funcione corretamente, implemente **todas** as necessidades acima descritas e implemente os conceitos SOLID e Clean Architecture **de forma prescritiva**;
- Todos os arquivos “manifesto” Kubernetes para a implementação da solução em um cluster, o Dockerfile para o build da aplicação e o arquivo de definição docker-compose que descreva todos os componentes necessários para que a aplicação funcione corretamente e seja possível subir a aplicação localmente usando **apenas** o comando “docker compose up”.