Desafio: Sistema de Pagamento Simplificado

Neste desafio, você será responsável por desenvolver um sistema de pagamento simplificado em PHP. O sistema permitirá que usuários comuns e lojistas realizem transferências de dinheiro entre si.

Orientações

- Crie um repositório no GitHub para a resolução do desafio.
- Deixe claro e exemplificado como executar o script/aplicação.
- Pode fazer utilização de um framework.

Objetivo

Desenvolver um sistema de pagamento simplificado que permita a transferência de dinheiro entre usuários comuns e lojistas.

Requisitos

Aqui estão os requisitos principais para o funcionamento do sistema:

- Cadastro de usuários: Ambos os tipos de usuários (comuns e lojistas) devem fornecer nome completo, CPF/CNPJ, e-mail e senha. CPF/CNPJ e e-mails devem ser únicos no sistema, permitindo apenas um cadastro por CPF ou endereço de e-mail.
- Transferências de dinheiro: Usuários comuns podem enviar dinheiro para lojistas e entre si. Lojistas só recebem transferências e não enviam dinheiro para ninguém.
- Validação de saldo: Antes de efetuar uma transferência, o sistema deve validar se o usuário possui saldo suficiente em sua carteira.
- Consulta a serviço externo autorizador: Antes de finalizar uma transferência, o sistema deve consultar um serviço externo autorizador. Utilize este mock para simular a autorização.
- Transações reversíveis: Toda transferência deve ser tratada como uma transação, revertendo em caso de inconsistência e devolvendo o dinheiro para a carteira do usuário remetente.
- Notificação de pagamento: Após o recebimento de um pagamento, tanto o usuário quanto o lojista devem receber uma notificação por e-mail ou SMS. Utilize este mock para simular o envio de notificações.

Avaliação

- Boas práticas: Serão avaliadas habilidades básicas de criação de projetos backend, como conhecimentos sobre REST, uso do Git, capacidade analítica e apresentação de código limpo e organizado.
- Conhecimentos intermediários: Aderência a recomendações de implementação (PSRs), aplicação de SOLID, identificação e aplicação de Design Patterns, documentação e descrição do projeto, implementação e conhecimentos sobre testes de unidade e integração, e boas noções de bancos de dados relacionais.
- Diferenciais: Uso de Docker, cobertura de testes consistente, uso de Design Patterns, proposta de melhoria na arquitetura, consistência nas escolhas e argumentação, domínio das soluções apresentadas, modelagem de dados, manutenibilidade do código, tratamento de erros, uso de container de injeção de dependências cuidados com segurança e arquitetura.

Boas práticas

- Tente seguir as PSRs se estiver utilizando PHP. Caso contrário, siga as boas práticas da comunidade da linguagem ou framework escolhido.

Esse desafio deverá testar suas habilidades de desenvolvimento backend e sua capacidade de criar soluções escaláveis, seguras e bem estruturadas. Boa sorte!