Teste de Codificação de Desenvolvedor Back-End

Descrição do Problema

A Lu Estilo é uma empresa de confecção que está buscando novas oportunidades de negócio, mas o time comercial não possui nenhuma ferramenta que facilite novos canais de vendas.

Solução

Para ajudar o time comercial, você deve desenvolver uma API RESTful utilizando FastAPI que forneça dados e funcionalidades para facilitar a comunicação entre o time comercial, os clientes e a empresa. Essa API deve ser consumida por uma interface Front-End, que será desenvolvida por outro time.

Desafio Extra – Integração com WhatsApp

Dando continuidade à solução para o time comercial da Lu Estilo, agora será necessário integrar a API com o WhatsApp, permitindo que a equipe envie mensagens automáticas para clientes a partir de eventos comerciais, como novos pedidos, envio de orçamentos ou promoções.

Desafio

Implemente uma funcionalidade adicional na API que permita o envio de mensagens de WhatsApp para clientes utilizando WhatsApp API.

Requisitos

1. Linguagens/estruturas:
    a. Python, FastAPI, Pytest para testes.
2. Endpoints:
    a. Autenticação:
        i. POST /auth/login: Autenticação de usuário.
        ii. POST /auth/register: Registro de novo usuário.
        iii. POST /auth/refresh-token: Refresh de token JWT.
    b. Clientes:
        i. GET /clients: Listar todos os clientes, com suporte a paginação e filtro por nome e email.

ii. POST /clients: Criar um novo cliente, validando email e CPF únicos.

iii. GET /clients/{id}: Obter informações de um cliente específico.

iv. PUT /clients/{id}: Atualizar informações de um cliente específico.

v. DELETE /clients/{id}: Excluir um cliente.

c. Produtos:

i. GET /products: Listar todos os produtos, com suporte a paginação e filtros por categoria, preço e disponibilidade.

ii. POST /products: Criar um novo produto, contendo os seguintes atributos: descrição, valor de venda, código de barras, seção, estoque inicial, e data de validade (quando aplicável) e imagens.

iii. GET /products/{id}: Obter informações de um produto específico.

iv. PUT /products/{id}: Atualizar informações de um produto específico.

v. DELETE /products/{id}: Excluir um produto.

d. Pedidos:

i. GET /orders: Listar todos os pedidos, incluindo os seguintes filtros: período, seção dos produtos, id_pedido, status do pedido e cliente.

ii. POST /orders: Criar um novo pedido contendo múltiplos produtos, validando estoque disponível.

iii. GET /orders/{id}: Obter informações de um pedido específico.

iv. PUT /orders/{id}: Atualizar informações de um pedido específico, incluindo status do pedido.

v. DELETE /orders/{id}: Excluir um pedido.

3. Autenticação e Autorização:

a. Utilize JWT (JSON Web Token) para autenticação.

b. Proteja as rotas de clientes, produtos e pedidos para que apenas usuários autenticados possam acessá-las.

c. Implemente níveis de acesso: admin e usuário regular, restringindo ações específicas a cada nível.

4. Validação e Tratamento de Erros:

   a. Implemente validações adequadas para todos os endpoints.

   b. Garanta que respostas de erro sejam informativas e sigam um padrão consistente.

   c. Registre erros críticos em um sistema de monitoramento, como Sentry.

5. Banco de Dados:

   a. Utilize um banco de dados relacional como PostgreSQL.

   b. Implemente migrações de banco de dados para facilitar a configuração do ambiente.

   c. Utilize índices adequados para melhorar a performance das consultas.

6. Documentação da API:

   a. Utilize o sistema de documentação automática do FastAPI (Swagger).

   b. Inclua exemplos de requisições e respostas para cada endpoint.

   c. Adicione seções de descrição detalhada para cada endpoint, explicando regras de negócio e casos de uso.

7. Testes:

   a. Implemente testes unitários e de integração.

   b. Utilize pytest para os testes.

Recomendações

- Boas práticas:
  1. Preocupe-se com técnicas adequadas de programação e arquitetura.
  2. Divida suas alterações em commits pequenos e bem descritos.
  3. Realize deploy da aplicação na plataforma de sua preferência utilizando Docker.

Entrega

Disponibilize o código através do GitHub. O teste deve ser entregue até o dia 26/05/2025.

Critérios de Avaliação

1. Funcionalidade: A API cumpre todos os requisitos especificados.
2. Código: Qualidade do código, incluindo a organização, legibilidade e boas práticas.
3. Documentação: Documentação clara e completa da API.
4. Teste: Cobertura de testes e qualidade dos mesmos.
5. Deploy: Capacidade de deploy utilizando Docker e clareza das instruções para execução do projeto.

Notas

Adapte os endpoints e funcionalidades conforme necessário, adicionando ou removendo recursos opcionais que possam demonstrar suas habilidades. Use seu tempo com sabedoria para encontrar a melhor solução possível que possa ser entregue dentro do prazo.

Links Úteis

- FastAPI Documentation: https://fastapi.tiangolo.com/
- Pytest Documentation: https://docs.pytest.org/
- Docker Documentation: https://docs.docker.com/
- GitHub: https://github.com/