Desafio Técnico: API de Gerenciamento de Tarefas

Objetivo: Criar uma API para gerenciamento de tarefas (To-Do List) utilizando FastAPI. A API deve permitir a criação, leitura, atualização e exclusão de tarefas.

Requisitos:

- Utilizar FastAPI para criar a API.
- A API deve ter endpoints para:
    - Criar uma nova tarefa.
    - Listar todas as tarefas.
    - Atualizar uma tarefa existente.
    - Deletar uma tarefa.
    - Visualizar uma tarefa específica pelo seu ID.

Especificações:

- Cada tarefa deve ter os seguintes campos:
    - **id** (inteiro, autoincrementado)
    - **titulo** (string, obrigatório)
    - **descricao** (string, opcional)
    - **estado** (string, obrigatório, valores possíveis: "pendente", "em andamento", "concluída")
    - **data_criacao** (datetime, gerado automaticamente)
    - **data_atualizacao** (datetime, atualizado automaticamente)

Tópicos para Orientar o Desenvolvimento:

Requisitos Obrigatórios:

1. **Configuração Básica**:
    - Inicializar um projeto FastAPI.
2. **Teste Unitário**:
    - Escrever testes unitários para os endpoints utilizando Pytest.
3. **CRUD de Tarefas**:
    - Implementar os endpoints para criar, listar, atualizar, visualizar e deletar tarefas.
    - Validar os dados de entrada utilizando Pydantic.
4. **Autenticação**:
    - Implementar autenticação básica para os endpoints.
    - Utilizar JWT para a autenticação.
5. **Documentação da API**:
    - A API deve ser autodocumentada utilizando o Swagger (disponível em **/docs**).

Requisitos Opcionais:

1. **Banco de Dados**:
    - Utilizar SQLite ou outro banco de dados relacional.
    - Utilizar SQLAlchemy (SQLMODEL) para gerenciar o ORM (Object-Relational Mapping).
2. **Migrações de Banco de Dados com Alembic**:
    - Configurar o Alembic para gerenciar migrações de esquema do banco de dados.
    - Criar migrações para a criação e modificação das tabelas de tarefas.
3. **Filtros e Paginação**:
    - Adicionar a possibilidade de filtrar tarefas pelo estado.
    - Implementar paginação na listagem de tarefas.
4. **Dockerização**:
    - Criar um Dockerfile para containerizar a aplicação.
    - Criar um docker-compose.yml para facilitar o desenvolvimento e a execução da aplicação.
5. **Crawler (Opcional)**:
    - Implementar um crawler simples que busca tarefas de uma fonte externa e as adiciona ao banco de dados da aplicação.
    - A fonte externa pode ser qualquer site público de gerenciamento de tarefas (por exemplo, uma lista de exemplos de tarefas).
6. **Desempenho**:
    - Implementar caching para melhorar o desempenho dos endpoints de leitura.

Instruções de Submissão:

- Crie um repositório no GitHub para o projeto.
- Faça commits regulares com mensagens claras e significativas.
- Inclua um README.md com instruções claras sobre como configurar e executar o projeto localmente, incluindo comandos para rodar a aplicação e os testes.
- Envie o link do repositório GitHub ao final do desafio.

Observações:

- Este desafio é realizável em um único dia, dependendo do nível de experiência do desenvolvedor.
- Os primeiros itens são bem simples, mas há espaço para um desenvolvedor avançado demonstrar suas habilidades através dos itens opcionais.

Espero que este desafio atenda suas necessidades! Se precisar de ajustes ou tiver alguma dúvida, estou à disposição.