Banking-api

Usuários:

- Endpoint para criação de usuários com nome,email,cpf,data de nascimento,idade, cidade, email, ativo,login e senha.
- Endpoint para alteração de usuário.
- Endpoint para remoção lógica do usuário, onde o registro não será removido do banco, apenas a coluna ativo deverá ter seu valor alterado.

Autenticação:

- Endpoint para fornecimento de token JWT ao usuário.
- Endpoint para *refresh* do token JWT, recebendo um token válido o retorna outro token válido.
- A API deverá validar a autenticação do token enviado a cada *request*, e retornar o erro http certo, em caso de falha na autenticação.

Contas:

- Endpoint para cadastro de contas, tendo como atributos: número para identificação,saldo, tipo corrente ou poupança, data de criação, data do último depósito, ativo, usuário relacionado.
- Endpoint para cancelamento de contas, executando uma deleção lógica, onde o registro não será removido do banco, apenas a coluna ativo deverá ter seu valor alterado.
- Endpoint listagem de todas as contas ativas.
- Endpoint para busca de conta por id.
- Endpoint para busca de contas por id de usuário.

Operação: (o saldo deverá ser verificado antes de cada operação, em caso de saldo insuficiente retornar um erro, pesquise na internet qual o melhor tipo de erro http para este cenário, crie uma exceção dentro da aplicação e lance em caso de saldo insuficiente)

- Endpoint para depósito por conta.
- Endpoint para saque por conta.
- Endpoint para transferência entre contas.

Rendimento:

- Endpoint que quando chamado, executará a seguinte rotina: recuperar todas as contas ativas do tipo poupanca, e que tiveram seu ultimo depósito a mais de um mês, e incrementar seu saldo em 1%.*(futuramente melhoraremos dois pontos: adicionaremos um framework que invocará*

automaticamente a execução dessa rotina, todo dia às 00:00. E antes de aplicar o rendimento, iremos buscar a taxa de rendimento através de uma integração com a b3 investimentos.)

***OBS**:*

Todo o código deverá ser escrito em inglês.

Lembre de escolher o verbo HTTP certo para cada endpoint.

Lembre de buscar as melhores práticas para caminhos de endpoint na internet.

Lembre de de pesquisar, entender o que é, e mapear os relacionamentos entre as entidades utilizando fetchType = LAZY.

Lembre de tentar utilizar nomes concisos entre as variáveis e métodos.

Lembre de estudar o que são e qual a diferença entre testes unitários e teste de integração ao final da implementação, pois iremos adicionar os testes no final do projeto, utilizando JUnit e Mockito.