Desafio: Sistema de Gerenciamento Bancário Colaborativo   
Objetivo: Desenvolver um sistema bancário colaborativo onde os usuários possam criar   
contas bancárias, realizar depósitos, saques, transferências entre contas e consultar   
extratos. O sistema terá funcionalidades básicas de controle de acesso e um serviço de   
API RESTful para integração com o frontend. Além disso, será necessário implementar   
autenticação e autorização de usuários, com foco na segurança das transações.   
Trello: https://trello.com/b/cCVMHBfh/sistema-bancario   
   
Requisitos:   
Funcionalidades:   
1. Autenticação de Usuários:   
o Registro de novos usuários.   
o Login e logout de usuários.   
o Recuperação de senha via e-mail.   
2. Gestão de Contas Bancárias:   
o Listar contas bancárias do usuário logado.   
o Criar novas contas bancárias.   
o Editar informações de contas (ex: titular).   
o Consultar saldo e extrato bancário.   
o Realizar depósitos e saques.   
o Realizar transferências entre contas (dentro do próprio banco).   
3. Integração com Django Rest Framework (DRF):   
o Implementar uma API para criação, edição e visualização de contas   
bancárias.   
o Implementar endpoints para depósitos, saques e transferências.   
o Implementar autenticação por token para acesso à API, garantindo que   
apenas usuários autenticados possam realizar transações.   
4. Controle de Transações:   
o Registrar todas as transações (depósitos, saques, transferências) com data   
e valor.   
o Verificar se o saldo é suficiente antes de realizar um saque ou   
transferência.   
o Exibir um histórico completo de transações realizadas por cada usuário.   
5. Filtros e Busca:   
o Buscar transações por tipo (depósito, saque, transferência).   
o Filtrar transações por data (última semana, mês, ano).   
o Buscar contas bancárias do usuário por nome do titular.   
6. Interface Web Responsiva:   
o Design simples e responsivo utilizando Django Templates.   
o Interface para visualização do saldo e extrato bancário.   
o Interface para realizar depósitos, saques e transferências.

Modelos:   
1. Usuário (User):   
o Nome: CharField.   
o E-mail: EmailField.   
o Senha: CharField (utilizar o modelo de autenticação padrão do Django).   
2. Conta Bancária (BankAccount):   
o Número da conta: CharField.   
o Titular: ForeignKey para o modelo User.   
o Saldo: DecimalField (com duas casas decimais).   
o Data de criação: DateTimeField.   
3. Transação (Transaction):   
o Tipo: CharField (depósito, saque, transferência).   
o Valor: DecimalField.   
o Data: DateTimeField.   
o Conta de origem: ForeignKey para o modelo BankAccount.   
o Conta de destino (somente para transferências): ForeignKey para o   
modelo BankAccount (opcional).   
o Status: CharField (concluída, pendente, falha).   
   
Endpoints da API:   
1. /api/accounts/ - GET, POST   
(Listagem de contas bancárias do usuário logado, criação de novas contas).   
2. /api/accounts/{id}/ - GET, PUT   
(Visualização e edição de informações de uma conta específica).   
3. /api/deposit/ - POST   
(Realizar depósito em uma conta bancária).   
4. /api/withdraw/ - POST   
(Realizar saque de uma conta bancária).   
5. /api/transfer/ - POST   
(Realizar transferência entre contas bancárias).   
6. /api/transactions/ - GET   
(Listar transações do usuário logado com filtros por tipo e data).   
   
Desafios Técnicos:   
1. Controle de Acesso:   
o Implementar controle para garantir que os usuários só possam acessar e   
manipular suas próprias contas e transações.   
o Impedir que um usuário saque ou transfira mais do que o saldo   
disponível na conta.   
2. Transações Seguras:

o Assegurar que as transações (depósitos, saques, transferências) sejam   
realizadas de forma atômica e que o sistema não permita inconsistências   
(por exemplo, debitar e creditar valores errados em casos de falhas).   
3. Autenticação Segura:   
o Utilizar autenticação via tokens para a API, garantindo que somente   
usuários autenticados possam realizar operações nas contas.   
o Implementar a criptografia das senhas utilizando o padrão do Django.   
   
Critérios de Avaliação:   
1. Funcionalidade:   
o Todas as funcionalidades descritas devem estar implementadas e   
funcionando corretamente.   
2. Código Limpo:   
o O código deve ser bem organizado, seguindo boas práticas do Django e   
boas práticas de desenvolvimento (como separação de responsabilidades,   
nomeação clara de funções, etc.).   
3. Autenticação e Segurança:   
o A implementação de autenticação e autorização deve ser segura,   
utilizando tokens para a API e garantindo a proteção das informações   
sensíveis (como senhas).   
4. Documentação:   
o O projeto deve ter uma documentação clara e simples, explicando como   
rodar a aplicação, como utilizar a API, e como fazer as transações.   
5. Interface do Usuário:   
o A interface web deve ser simples, funcional e responsiva, permitindo ao   
usuário realizar todas as operações necessárias de forma intuitiva.   
   
Entregáveis:   
•   
Código-fonte do projeto Django no github:   
https://github.com/carloscelestino1/banco\_turma\_noite\_senac   
•   
Instruções para rodar a aplicação:   
o Instalação de dependências.   
o Setup inicial (criação de banco de dados, configuração do Django).   
o Comandos para migração do banco de dados e rodar o servidor.   
o Arquivo requirements.txt   
o Redme informativo   
•   
Testes :   
o Testes unitários e de integração, especialmente para as funções que   
envolvem transações financeiras, como depósitos, saques e   
transferências. (opcional)   
o Testes de conexão da API com postman(prints)