**DOCUMENTAÇÃO DA PLATAFORMA DE FEEDBACKS COM RECOMPENSAS EM TOKENS (TRANSFERO ACADEMY)**

**1. VISÃO GERAL DO PROJETO**

Este repositório contém o código-fonte do backend da "Plataforma de Feedbacks com Recompensas em Tokens", desenvolvida pela Transfero Academy. O objetivo principal do projeto é conectar empresas que buscam feedbacks qualificados com usuários comuns, que são recompensados em tokens por suas contribuições. A plataforma também envolve analistas para avaliação de feedbacks e um administrador para gerenciamento geral. A branch dev é utilizada para o desenvolvimento contínuo.

**2. PAPEIS E ATORES NO SISTEMA**

A plataforma atende a diferentes tipos de usuários e suas respectivas funções:

* **Empresas:** Desejam receber feedbacks qualificados para aprimorar seus produtos ou serviços.
* **Usuários Comuns:** São recompensados em tokens ao completarem missões ou responderem questionários, fornecendo feedbacks valiosos.
* **Analistas:** Responsáveis por avaliar a qualidade e relevância dos feedbacks submetidos.
* **Administrador:** Gerencia planos, contratos e todas as transações da plataforma, incluindo a estrutura de tokens.

**3. ESTRUTURA DE TOKENS E BLOCKCHAIN**

Os tokens da plataforma simulam uma estrutura blockchain. Há suporte real a carteiras via BNB Chain ou Polygon para produção e ETH-Sepolia para o ambiente de desenvolvimento, indicando uma robusta integração com tecnologias blockchain.

**4. TECNOLOGIAS PRINCIPAIS**

A plataforma é construída sobre um conjunto de tecnologias modernas para garantir escalabilidade e eficiência:

* **Backend:** Desenvolvido em Django com Django REST Framework, focado em APIs robustas e performáticas.
* **Blockchain:** Integração com Smart Contracts (Solidity) para a gestão e emissão dos tokens.
* **Banco de Dados:** Utiliza PostgreSQL para o ambiente de produção, garantindo alta performance e confiabilidade, e SQLite para o desenvolvimento local, facilitando a prototipagem.
* **Autenticação:** Implementa JWT (JSON Web Tokens) para um sistema de autenticação seguro e stateless.
* **Frontend:** O frontend é desenvolvido em React e reside em um repositório separado, indicando uma arquitetura desacoplada.

**5. REQUISITOS PARA RODAR LOCALMENTE**

Para configurar e rodar o projeto localmente, são necessários os seguintes softwares:

* **Python:** Versão 3.10 ou superior.
* **pip:** Gerenciador de pacotes do Python.
* **Poetry (opcional) ou Venv:** Ferramentas para gerenciamento de ambientes virtuais Python, sendo Poetry uma alternativa mais robusta.
* **Git:** Sistema de controle de versão.

**6. COMO CONTRIBUIR PARA O REPOSITÓRIO**

Este tutorial detalha o fluxo de trabalho ideal para contribuições, garantindo que as alterações sejam integradas de forma eficiente ao repositório principal.

**6.1 Instalação e Setup Local**

Siga estas etapas para preparar seu ambiente de desenvolvimento:

1. **Faça o Fork do Repositório:**
   * Acesse o repositório de desenvolvimento: [Repositório de desenvolvimento](https://github.com/TransferoNovaIguacu/feedback-company-backend.git).
   * Clique no botão "Fork" no canto superior direito do GitHub para criar uma cópia do repositório em sua conta.
2. **Clone o Repositório Forkado:**
   * Utilize o seguinte comando, substituindo SEU-USUARIO pelo seu nome de usuário do GitHub:

Bash

git clone --branch dev --single-branch https://github.com/SEU-USUARIO/feedback-company-backend.git

1. **Configure o Ambiente Local:**
   * **Com venv:**

Bash

python -m venv venv

venv\Scripts\activate

* + **Com poetry:**

Bash

poetry install

poetry shell

1. **Instale as Dependências do Projeto:**
   * Execute o comando para instalar as bibliotecas necessárias:

Bash

pip install -r requirements-dev.txt

1. **Crie o Arquivo .env:**
   * Na pasta raiz do projeto, crie um arquivo .env com as seguintes variáveis de ambiente:
   * SECRET\_KEY=teste123
   * DEBUG=True
   * ALLOWED\_HOSTS=127.0.0.1
2. **Execute as Migrações e Crie um Superusuário:**
   * Aplique as migrações do banco de dados e crie uma conta de superusuário para acesso administrativo:

Bash

python manage.py makemigrations

python manage.py migrate

python manage.py createsuperuser

1. **Rode o Servidor Local:**
   * Inicie o servidor de desenvolvimento:

Bash

python manage.py runserver

**6.2 Fluxo de Contribuição**

As etapas a seguir devem ser realizadas após a configuração inicial do ambiente.

1. **Sincronizando seu Repositório com o Principal:**
   * Adicione o repositório principal como upstream para sincronizar suas alterações:

Bash

git remote add upstream https://github.com/TransferoNovaIguacu/feedback-company-backend.git

* + Atualize seu repositório local:

Bash

git fetch upstream

git checkout dev

git merge upstream/dev

1. **Criação de Branch:**
   * Crie uma nova branch para suas alterações, seguindo o padrão de nomenclatura tipo/descricao-da-tarefa (ex: feature/login-system):

Bash

git checkout -b <tipo>/descricao-da-tarefa

* + *Nota:* Consulte [Conventional Commits](https://www.conventionalcommits.org/en/v1.0.0/) para entender o padrão de nomenclatura recomendado.

1. **Fazendo Alterações e Commitando:**
   * Adicione os arquivos alterados ao *staging area* e crie um *commit* com uma mensagem descritiva:

Bash

git add .

git commit -m "feat: add user login system"

1. **Subindo as Alterações para o Seu Repositório Forkado:**
   * Envie suas alterações para a sua branch no repositório *forkado*:

Bash

git push origin <nome-da-branch>

1. **Criando o Pull Request:**
   * Acesse seu repositório no GitHub.
   * Selecione a branch que você acabou de enviar.
   * Clique em "Compare & pull request".
   * Descreva as mudanças realizadas e clique em "Create pull request".
   * ***Atenção:*** Certifique-se de que o Pull Request está sendo enviado para a branch dev do repositório principal.
2. **Aguardando Revisão:**
   * Aguarde a revisão e aprovação do seu Pull Request. Caso receba feedbacks, realize os ajustes necessários e envie novamente.

**7. PLANEJAMENTO E DOCUMENTAÇÃO ADICIONAL**

A pasta /docs do repositório contém informações adicionais e mais aprofundadas sobre o projeto, incluindo:

* Requisitos funcionais e não funcionais.
* Regras de negócio e fluxos do sistema.
* Diagramas de classes UML.
* Backlog de Tarefas.

**8. FLUXO COMPLETO DE CONTRIBUIÇÃO (RESUMO)**

Para recapitular, o fluxo de contribuição recomendado é:

1. Fork e Clone do Repositório Principal.
2. Sincronização com o Repositório Principal (upstream).
3. Criação de Branch para Tarefa.
4. Realização de Alterações, Commit e Push.
5. Criação do Pull Request.
6. Aguardando Revisão e Aprovação.