Desafio Técnico - Pleno ZapSign

Atores:

- **Desenvolvedor:** O desenvolvedor é responsável por implementar uma aplicação CRUD (Create, Read, Update, Delete) usando Python/Django para o backend, PostgreSQL para o banco de dados e Angular/TypeScript para o frontend.
- **API ZapSign (Sandbox):** A API externa utilizada para criar e gerenciar assinaturas eletrônicas para documentos.

Pré-condições:

- O desenvolvedor deve criar uma conta de acesso no ambiente de sandbox da Zapsign. https://sandbox.app.zapsign.com.br/
- Deve ser criado um banco de dados PostgreSQL com as tabelas necessárias: `Company`, `Documents` e `Signers`. Verificar o Diagrama do Banco de dados nas últimas páginas.
- Deve ser gerado o token da API para a ZapSign e armazenar na tabela `Company`, coluna api_token.

Pós-condições:

- Documentos criados gerenciados com sucesso usando as operações CRUD.
- Documentos criados via API ZapSign são armazenados na tabela `Documents`.
- Signatários associados a cada documento são armazenados na tabela `Signers`.
- As operações do CRUD (UPDATE e DELETE) são realizadas diretamente no banco de dados local. Somente para a operação de CREATE é enviado uma request para o ambiente de Sandbox da Zapsign para realizar a criação do documento no site e salvar o retorno no banco de dados, tais como, open_id, token do documento criado, etc.

Observações:

1. Documentação da API: https://docs.zapsign.com.br/
2. O design não será avaliado (pode utilizar qualquer bootstrap/material UI).
3. Devem ser realizados testes unitários em pelo menos uma das aplicações (Django e Angular).
4. **No final do documento possui os diagramas de fluxo, classes, e banco de dados.**

Bônus:

1. Utilizar Code Clean, Design Patterns, Padrões de projeto, etc.
2. Rodar o projeto no Docker Compose.

Fluxo Principal:

1. **Criação de Conta no Sandbox da ZapSign:**
   O desenvolvedor deve começar criando uma conta no ambiente de sandbox da ZapSign através do link https://sandbox.app.zapsign.com.br/.
2. **Estrutura do Banco de Dados:**
   Em seguida, o desenvolvedor deve criar a estrutura do banco de dados conforme o diagrama fornecido, incluindo as tabelas `Company`, `Document`, e `Signers`.
3. **Coleta do API Token:**
   Após criar a conta, o desenvolvedor deve coletar o `api_token` gerado na nova conta ZapSign e salvá-lo na tabela `Company`, na coluna `api_token`.
4. **Implementação do CRUD para Documentos:**
   O desenvolvedor deve implementar a funcionalidade CRUD (Create, Read, Update, Delete) para documentos utilizando Angular/TypeScript. Isso inclui a criação de um componente no frontend que exibirá todos os documentos armazenados no banco de dados.
5. **Formulário de Criação de Documento:**
   Além disso, o desenvolvedor deve implementar um componente separado responsável por criar um formulário no frontend, onde os usuários poderão inserir detalhes sobre o documento. O formulário deve incluir campos para o nome do documento, detalhes do signatário (nome e email), e uma URL para um arquivo PDF.
6. **Envio de Dados para a API ZapSign:**
   Ao enviar o formulário, as informações devem ser enviadas para a API ZapSign no endpoint https://sandbox.api.zapsign.com.br/api/v1/docs/.
7. **Resposta da API e Salvamento no Banco de Dados:**
   A API responderá com um objeto JSON contendo detalhes como `open_id`, `token` e o status do documento. Esses detalhes devem ser salvos no banco de dados.
8. **Atualização do Frontend:**
   Após o salvamento dos dados, o frontend deve exibir uma mensagem de sucesso e recarregar o componente do CRUD (Não a página inteira).

Operações CRUD:

1. **Create (Criar):** A aplicação permite que os usuários criem novos documentos e signatários, enviando as informações necessárias para a API ZapSign.
2. **Read (Ler):** A aplicação exibe uma lista de todos os documentos armazenados na tabela `Documents`, junto com seus signatários associados da tabela `Signers`.
3. **Update (Atualizar):** Os usuários podem atualizar os detalhes de um documento ou signatário, através do CRUD.
4. **Delete (Excluir):** Os usuários podem excluir um documento, o que também excluirá os signatários associados. As informações do documento e do signatário são removidas do banco de dados local.

Gerenciamento do Banco de Dados:

1. A tabela `Company` armazena informações sobre a empresa, incluindo o token da API necessário para interagir com a API ZapSign.
2. A tabela `Documents` armazena informações sobre cada documento criado, incluindo o `open_id`, `token`, `name`, `status`, data de criação e o usuário que criou o documento.
3. A tabela `Signers` armazena informações sobre cada signatário associado a um documento, incluindo o nome do signatário, email, status e o token associado ao documento.

Integração com o Frontend:

1. O frontend Angular/TypeScript interage com o backend Django via APIs RESTful.
2. Quando um documento é criado com sucesso, o frontend deve atualizar dinamicamente a lista de documentos sem precisar recarregar a página inteira, proporcionando uma experiência de usuário contínua.

Diagrama de Fluxo

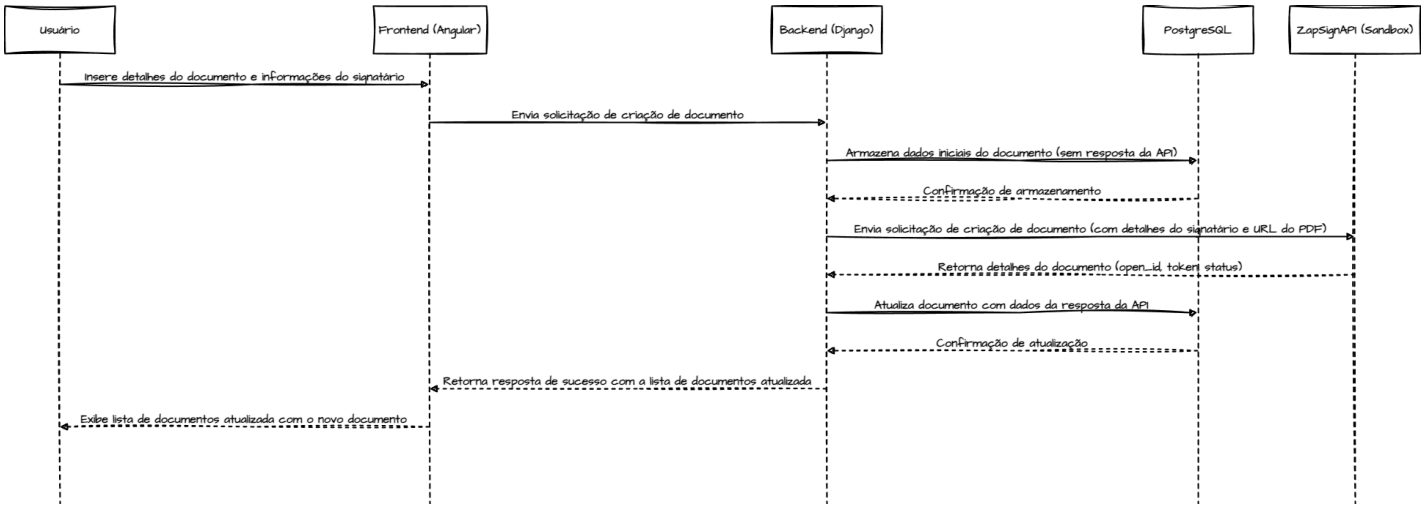

Diagrama de Classe

Diagrama Banco de Dados do PostgreSQL