

Case – Engenheiro Suporte de Integração

Wallace Robinson Louza Salles

E-mail: wallace_robinson@hotmail.com

Telefone: (21) 98155-9436

1. Plano de Trabalho:

Foi implementado uma interface web para transferência dos arquivos, conforme informado na case. Acabei substituindo o *JWT* pelo *CSRF Protection*. Falo sobre ele no assunto de Tecnologias utilizadas.

2. Tecnologias utilizadas:

Escolhi o Python como linguagem de programação por ter mais afinidade e trabalhar com ela na maioria do tempo.

No desafio foi utilizado o Python na versão 3.6, junto com o Django 2.2.4 e o uWSGI para uso de escalabilidade da aplicação, contando com 5 workers configuráveis.

Foi utilizado também o fator de autenticação do próprio *Django-Admin* para realizar o login na plataforma. Utilizo também o *CSRF Protection* que possui token de segurança na hora de realizar o upload dos arquivos (Esse token é bastante utilizado em PHP e está presente fortemente na documentação do Django).

Para persistência dos dados, eu preferi trabalhar com o *SQLite3* devido a facilidade da implementação e o projeto não demandar tanto uso de conexão, etc. Entretanto, com o *Django* isso não seria problema. Ele consegue criar a modelagem de dados em diferentes bancos relacionais, por exemplo, *MySQL*, *Postgres*, etc.

3. Embasamento teórico:

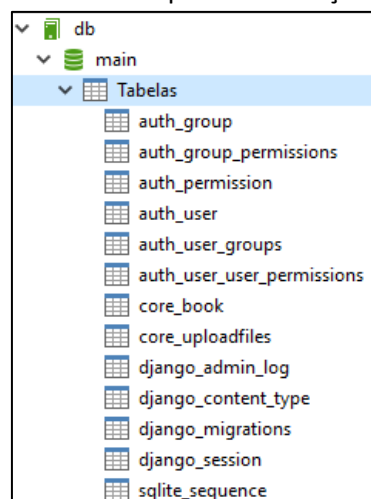
Dado as necessidades, pude concluir que o Django conseguiria por si só, atender a maioria dos pedidos deste case, e não só para esse case, digo em mais de um projeto. De fato, o framework é poderoso.

Quando desenvolvo alguma automação ou API para backend eu prefiro utilizar o *Flask*, que é outro framework do Python. Porém, em cenários que requer mais do frontend, eu acabo optando pelo *Django*.

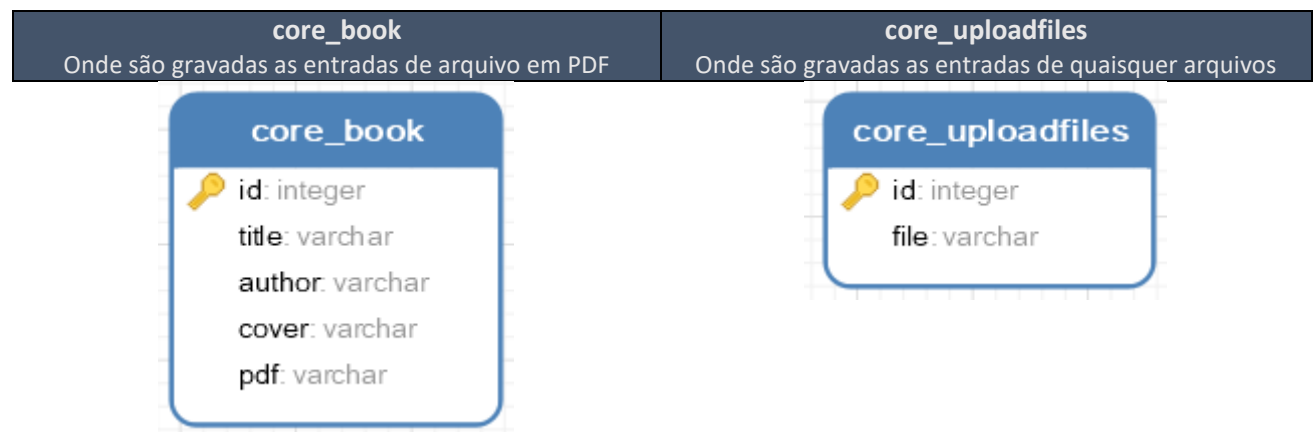
Pensando em uso de escalabilidade em aplicações web, tenho preferido utilizar bastante o *uWSGI*. O mesmo é simples, objetivo, fácil de configurar e poderoso.

4. Modelagem da base de dados:

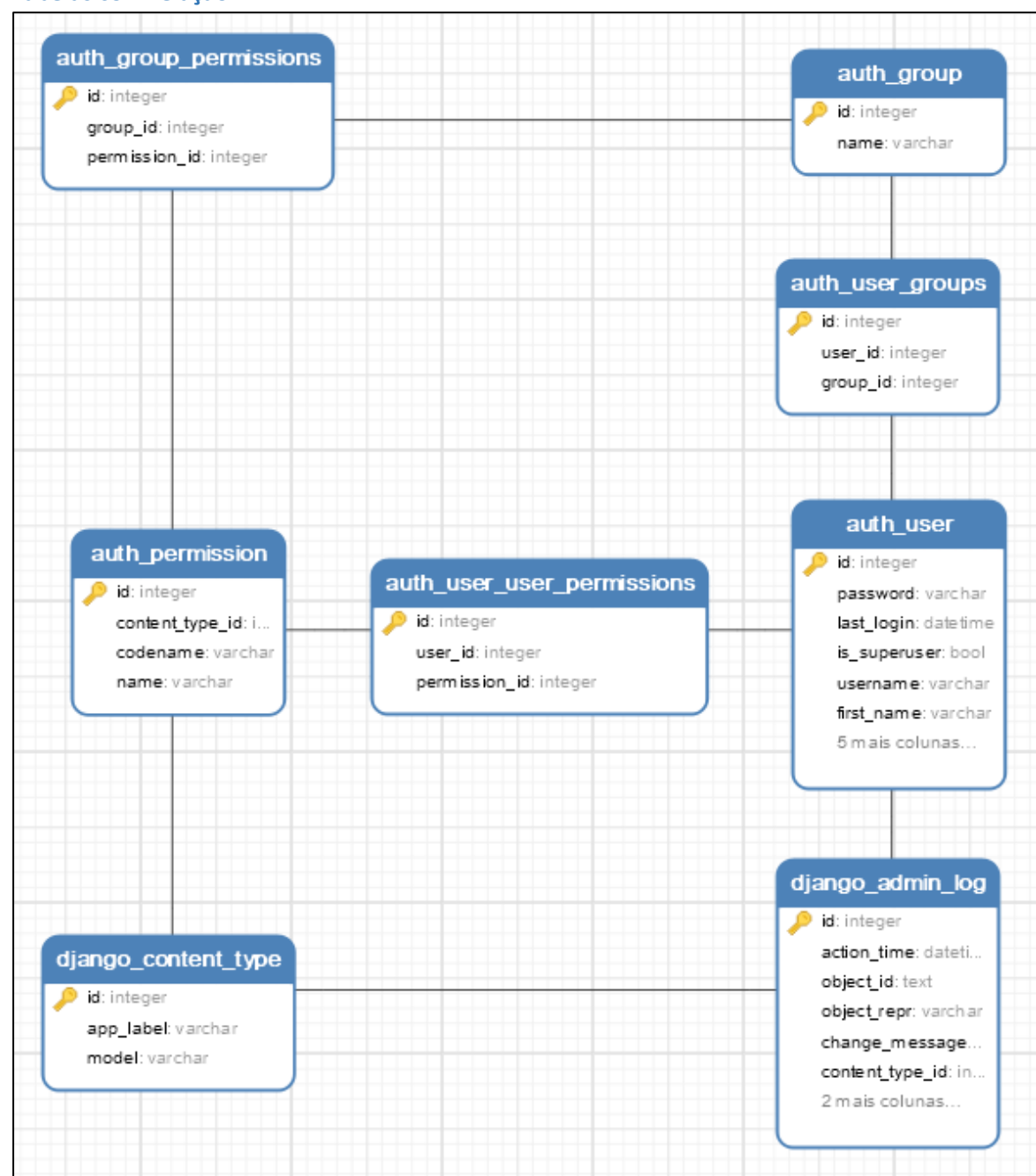
Segue abaixo, telas capturadas do software *Navicat* para visualização das tabelas criadas pelo *Django-Admin*:



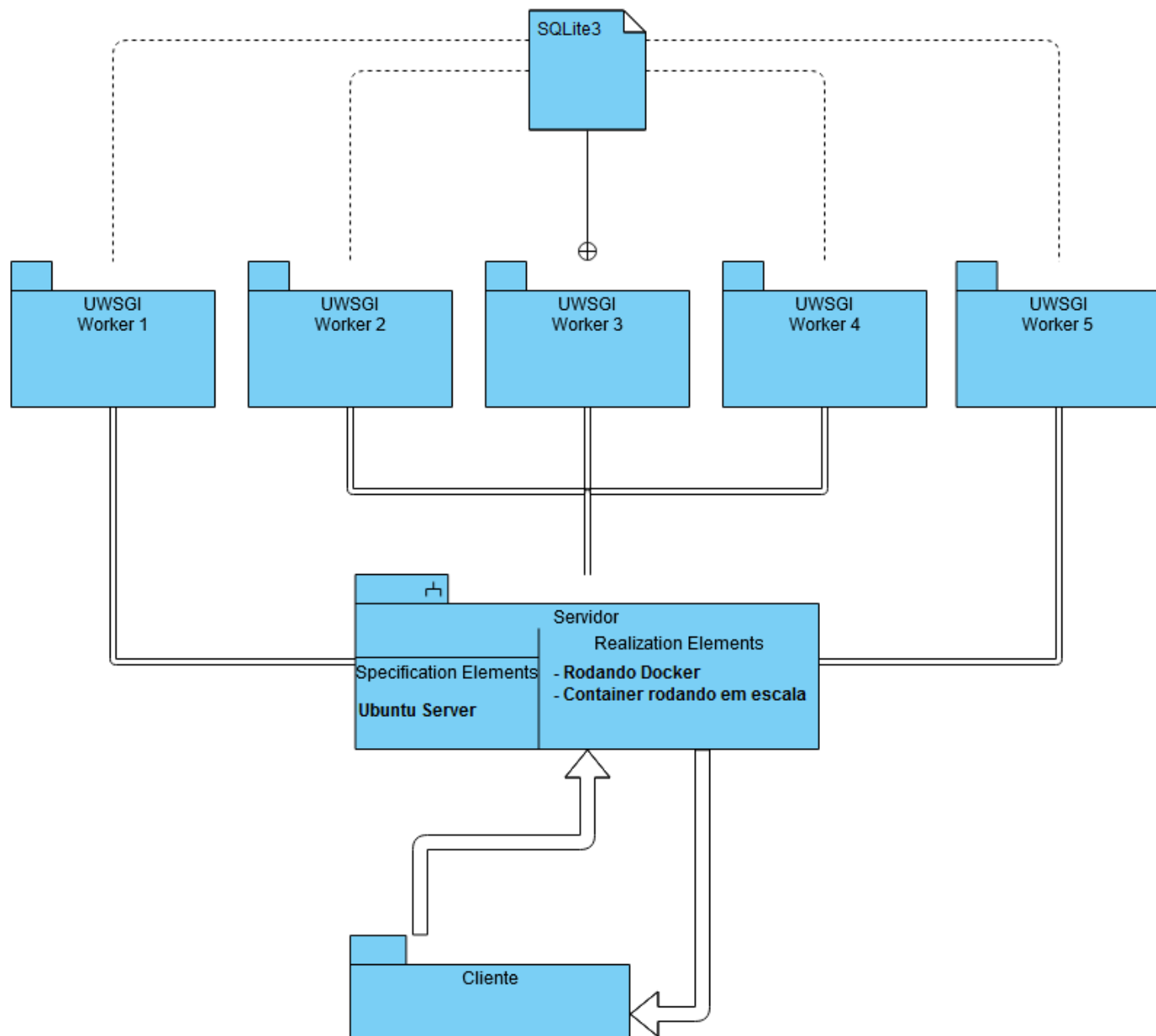
Porém, as duas principais do projeto são: **core_book** e **core_uploadfiles**



Tabelas com relação:



5. Diagrama de implantação da solução:



6. Código publicado no GitHub:

<https://github.com/wsalles/itau-file-transfer>

7. Referências utilizadas:

<https://docs.djangoproject.com/en/2.2/>

<https://docs.djangoproject.com/en/2.2/ref/csrf/>

<https://django-crispy-forms.readthedocs.io/en/latest/>

8. Testes (TDD) e Metodologia Ágil:

Sobre os testes, eu costumo implementar TDD em todos eles, entretanto, nesse projeto eu não tive tempo de fazer. Era entregar o projeto funcional ou entregar somente os testes.

Eu sigo a metodologia ágil e entendo que um sistema sem testes, é um sistema legado. Porém, espero que não fiquem chateado com a falta do TDD.

9. **Imagem do Docker publicada no Docker Hub:**

<https://hub.docker.com/r/wsalles/itau-file-transfer>