Seazone Code Challenge - APIs Back End

O desafio consiste em criar três **APIs REST**.
Leia todo o documento antes de começar a desenvolver o desafio.

Tecnologias a ser utilizadas:

- ➔ Python
- ➔ Django e Django Rest Framework
- ➔ Banco de dados SQL

Contexto:

A empresa Khanto começou a desenvolver um novo sistema e precisa criar um banco de dados com 3 entidades: Imóveis, Anúncios e Reservas. Um imóvel pode ter diversos anúncios, mas um anúncio é referente apenas a um imóvel. Um anúncio pode ter várias reservas, mas uma reserva se refere a apenas um anúncio.

A tabela de imóveis deve conter: o código do imóvel, o limite de hóspedes, a quantidade de banheiros, se aceita animais de estimação, o valor de limpeza, a data de ativação, a data e horário de criação e a data e horaŕio de atualização.

A tabela de anúncios deve conter: a qual imóvel o anúncio pertence, nome da plataforma em que o anúncio foi publicado (ex: AirBnb) , a taxa da plataforma, a data e horário de criação e a data e horário de atualização.

A tabela de reservas deve conter: o código da reserva (gerado aleatoriamente), a qual anúncio a reserva pertence, data de check-in, data de check-out, preço total, campo de comentário, número de hóspedes, data e horário de criação e data e horário de atualização.

Datas e horários de criação e de atualização devem ser inseridos automaticamente no sistema, e não manualmente.

Requisitos

API 1 - Imóveis

- Deve ser possível buscar uma lista de imóveis, buscar um imóvel individual, criar, alterar e deletar um imóvel;

- Deve-se criar uma fixture (seeder) para alimentar o banco de dados com pelo menos 5 imóveis.

API 2 - Anúncios:

- Deve ser possível buscar uma lista de anúncios, buscar um anúncio individual, criar um anúncio e alterar um anúncio, mas não apagá-lo.

- Deve-se criar uma fixture (seeder) para alimentar o banco de dados com pelo menos 3 anúncios.

API 3 - Reservas:

- Deve ser possível buscar uma lista de reservas, buscar uma reserva individual, criar uma reserva e deletar uma reserva, mas não alterá-la;

- Deve-se validar que não é possível criar uma reserva com a data de check-in posterior à data de check-out;

- Deve-se criar uma fixture (seeder) para alimentar o banco de dados com pelo menos 8 reservas.

Observações importantes:

- ➔ A qualidade do Readme e a organização do repositório é um ponto analisado;
- ➔ Ao fazer buscas, todos os campos de cada tabela devem aparecer em cada API;
- ➔ Para fazer uma busca individual, deve-se passar um parâmetro na URL contendo o id da entidade, caso esse parâmetro não seja passado, a lista será buscada;
- ➔ Todas as tabelas e códigos relacionados ao banco de dados devem ser construídos através do ORM nativo do Django.
- ➔ Criar testes unitários relevantes

Entrega:

Para entregar o projeto, você deverá criar um repositório no seu perfil pessoal do Github (criá-lo caso ainda não tenha), e deixá-lo em modo público.