ubiwhere

Exercício de Python & Django

Destinatário:
Tomás Batista

Python & Django

Abstract: Pretende-se uma API REST que permita a gestão de ocorrências em ambiente urbano. As ocorrências devem ter **descrição**, uma **localização geográfica**, **autor**, **data de criação**, **data de actualização**, **estado** (por validar, validado, resolvido) e pertencer a uma das seguintes categorias:

- CONSTRUCTION: Eventos planeados de obras nas estradas;
- SPECIAL_EVENT: eventos especiais (concertos, feiras, etc);
- INCIDENT: acidentes ou outros eventos inesperados;
- WEATHER_CONDITION: eventos meteorológicos que afetam as estradas;r
- ROAD_CONDITION: estados das estradas que afetem quem circula nestas (piso degradado, buracos, etc).

1) API REST
1.1) Tem de permitir a adição de ocorrências (com a localização geográfica, e autor associados). Nota: O estado *default* será sempre **por validar** quando estas são criadas e o autor deve ser um utilizador registado;
1.2) Tem de permitir a actualização de ocorrências (para mudar o estado das mesmas para "validadas" por um administrador do sistema);
1.3) Tem de permitir a filtragem de ocorrências por autor, por categoria e por localização (raio de alcance).

Notas:
A) A componente da API REST deve ser desenvolvida com Django (para expor uma API a partir de um projeto Django e recomenda-se a utilização do package Django Rest Framework); Ajuda se partilhares connosco uma collection Postman/Insomnia para conseguirmos testar rapidamente os endpoints que implementaste;
B) Deves permitir o registo de utilizadores. Este pode ser feito recorrendo ao painel do Django Admin, mas caso pretendas podes implementar também os endpoints necessários para o registo de utilizadores;
C) Deves fornecer instruções de como inicializar o projeto Django com a API. Caso tenhas conhecimentos em docker, podes fazê-lo através de um docker container (usando docker-compose). O mesmo se aplica para outros serviços, como a base de dados, que utilizes para estruturar o teu projeto;
D) A BD deixamos ao teu critério (SQLite, PostgresQL, MySQL,…) - mas recomendamos PostgreSQL com PostGIS para a parte geográfica;
E) Será também valorizado caso coloques o código implementado no Github, juntamente com as instruções para o inicializar.

Não espero um resultado completo, perfeitamente documentado e com todo o código coberto por testes, porém se conseguires fazê-lo, melhor!

Sugestões:

- Caso tenhas poucos conhecimentos em Django, sugerimos que explores a documentação em https://www.djangoproject.com/start/;
- No que diz respeito aos filtros podes encontrar mais informação na documentação em https://django-filter.readthedocs.io/en/master/;
- Relativamente ao docker, caso não tenhas conhecimentos e queiras disponibilizar o teu projeto utilizando-o sugerimos que vejas a documentação em https://docs.docker.com/get-started/;
- Relativamente ao Django Rest Framework podes explorar a documentação disponível em https://www.django-rest-framework.org/;
- Relativamente à localização geográfica podes verificar a documentação em django-rest-framework-gis/README.rst at master · openwisp/django-rest-framework-gis;

- Para facilitar a utilização do PostGIS, sugerimos que utilizes uma imagem Docker (https://hub.docker.com/r/mdillon/postgis/) para iniciar o serviço. Caso tenhas alguma dificuldade a iniciar o PostGIS, não hesites em partilhar conosco, estamos disponíveis para te esclarecer e ajudar em qualquer dúvida.

Critérios de avaliação principais:

- Qualidade do código - Não inventes, alguém vai ter que manter o teu código daqui a 10 anos!
- Arquitectura da solução - A aplicação é simples, mas a separação de responsabilidades é essencial à medida que os projectos vão crescendo. (Se te batesse um change request a meio da implementação, quão difícil será adaptar o que já tens feito?).
- Arquitectura dos endpoints - Os endpoints representam bem o conceito? Estão separados do modelo de dados? A tua solução segue os conceitos REST? Fazes bom uso dos verbos e códigos de resposta?
- Documentação - Pydoc, Readmes, Swagger, Redoc, etc... - Não é quantidade é qualidade. Preciso de saber acima de tudo como fazer deploy e correr o serviço. Preciso também de uma lista de endpoints e de como os invocar.
- Familiaridade com as API usadas - Só depois de olhar para tudo o que está acima é que vou ver ser usaste bem o que quer que seja que tentaste usar!

Todas as dúvidas que surjam podes partilhar connosco, afinal de contas, caso venhas para a Ubiwhere, estarás a trabalhar em equipa e não sozinho.

Boa sorte!