Teste técnico

Captalys — BackEnd Developer

Na Captalys, nós acreditamos que o **crédito é um bem comum**, e assim, uma força para o desenvolvimento social e econômico.

Por isso, investimos nosso capital financeiro, humano e intelectual para desenvolver novas tecnologias que tornem esta visão uma realidade.

Nossa ambição é empoderar qualquer pessoa ou organização a investir em, ofertar, e ter acesso a crédito para que possa criar prosperidade para si, suas comunidades, e nosso futuro coletivo. **Focamos em criar impacto, agindo sempre com integridade.**

Gerimos aproximadamente R$ 7 bilhões em fundos de investimento em crédito privado, oferecemos soluções de financiamento para empresas e seus clientes, prestamos serviços de originação e gestão de carteiras de crédito para bancos, investidores, marcas, *fintechs* e *marketplaces*, visando **aumentar o acesso e eficiência de todo o ecossistema de crédito privado.**

Desafio tecnico

O objeto deste desafio e criar uma API REST que se comunica com a API REST oficial do Github: https://api.github.com/

A aplicacao devera ter:

- Um endpoint para listar os repositorios publicos de um usuario;
    - `GET /repositories/?username=<username>`
    - Este endpoint deverá ter um parâmetro opcional booleano `from_local` que, caso verdadeiro, deverá buscar os repositórios do usuário na base local e retorná-los na seguinte descrita abaixo. Quando `from_local` for falso, o endpoint deve ir diretamente na api do github e buscar as informações. O retorno deve conter, no mínimo, uma lista com o nome de cada repositório.

```
{
   “user_id”: 1,
   “username”: “mrdev”,
   “repositories”: [ lista dos repositorios ]
 }
```

- Um endpoint para mostrar os detalhes de um repositorio especifico;
    - `GET /repositories/<repository_name>`
    - Devem ser retornadas as informações: url, nome, tipo de acesso, data de criação, data da ultima atualização, tamanho, stars e watchers.
    - Este endpoint deve ter um parâmetro booleano obrigatório `save_data` que, caso verdadeiro, salvará os detalhes do repositório em uma tabela.

Persistência

- O processo de persistência dos dados deverá conter apenas duas tabelas: `user` e `repository` e esta última deverá ter uma FK para o id do usuário dono do repo na tabela `user`.

Tecnologias

- Python 3+
- Flask + (FlaskRestplus/FlaskRestX) ou Starlette ou FastAPI;
- Requests ou Aiohttp;

Restricoes

- Escrever no arquivo README.md as intrucoes para subir a API em outro computador.

- Criar um repositorio publico no Github, Bitbucket ou Gitlab para armazenar o projeto.
- O projeto deve ser executado dentro de um container Docker.

Diferenciais

- Pytest e coverage: Escrever testes para os endpoints e garantir uma cobertura superior a 80%.
- Docker-composer: Adicionar um container com postgres para persistência de dados.
- Alembic: Adicionar migrations para criar a estrutura do db.