

Avaliação Técnica - Trackfy

Igor Dantas Gomes Franca

Contexto

Foi criada uma aplicação que permite administrar tarefas, no formato *To-Do-List*, em que é possível anotar novas tarefas, mudar seu status ("Em progresso" ou "Pronto"), filtrar as tarefas por status e apagá-las.

Solução

Front-end:

HTML | CSS |
JavaScript

Back-end:

Python
→ Flask (API)
→ SQLAlchemy
(ORM)
PostgreSQL (Vercel
Storage)


Deployment:

Vercel (Front e Back)





 todolist

-  main-page → Front-end
 -  index.html
 -  css
 -  style.css
 -  js
 -  script.js

| Deploy **todo-site** no Vercel

-  python-api → Back-end (API)

Deploy **todo-api** no Vercel

-  app.py
-  config.py
-  models.py
-  views.py

Para permitir a persistência dos itens da To-Do-List, com o auxílio de uma RESTful API em Flask (Python) para integração com o banco de dados (PostgreSQL), são acionados *scripts* no código JavaScript para quase todas as funções da página (exceto filtrar), enviando requisições para essa API.

A plataforma Vercel foi escolhida para deployment, pois além de não trazer custo, permite hospedar tanto a API quanto a aplicação visual. Também é possível manter um banco de dados na plataforma, integrando com a própria aplicação e facilitando questões de segurança, como variáveis de ambiente do banco para o sistema.

No "BD" PostgreSQL, foi utilizado apenas uma simples tabela, com colunas ID (Serial), Mensagem e Status, considerando que a plataforma seria de uso pessoal.

Front-end

Para uma aplicação aparentemente simples, a ausência de frameworks não atrapalhou no desenvolvimento em geral.

A composição da única tela da aplicação contém:

1. Um campo de inserção de dados (texto), tornando-se posteriormente um item na lista To-Do
2. Um campo de filtro por status de cada tarefa ("Em progresso" ou "Pronto"), com botões selecionados de forma alternada e que mostram apenas as tarefas com o status selecionado
3. Uma lista To-Do com seus respectivos itens, cada um podendo ter seu status alternado continuamente, ou então ser deletado

Cada um dos comandos, exceto o filtro, executa funções *fetch*, enviando requisições para a API. No mesmo instante em que a página é carregada, um método "GET" já obtém todos os itens no BD para seu display na lista.

Back-end






A API desenvolvida no framework Flask, na linguagem Python, permitiu criar uma aplicação simples, com consideravelmente pouca tipagem e complexidade, para poucos métodos exigidos pelo front-end, que foram:

1. Seleção de todos itens (GET)
2. Adicionar nova mensagem (POST)
3. Atualizar status de um item (PUT)
4. Deletar um item (DELETE)

O mapeamento para a tabela "todo", no Postgres hospedado no Vercel, foi feito através da ORM SQLAlchemy, sendo também responsável por fazer a conexão com o banco de dados. Enquanto isso, o Flask é responsável pela troca desses dados com o cliente.

O teste de cada método foi auxiliado com o uso do software Postman.

Abaixo, segue a organização dos arquivos e suas funções:

-  python-api
 -  app.py → arquivo para execução da API
 -  config.py → configurações da aplicação (IP, CORS, Var. Ambiente)
 -  models.py → ORM para a tabela **todo** no banco de dados
 -  views.py → métodos e rotas (GET, POST...)

Execução

A aplicação encontra-se disponível para uso em <https://todo-site-chi.vercel.app/>, hospedada na plataforma Vercel. (obs.: a aplicação pode

trazer um pouco de atraso durante a realização das requisições à API, por tratar-se de um recurso gratuito e limitado, embora não apresente erro nas suas funções).

A aplicação também pode ser executada localmente, seguindo os seguintes passos:

Execução local do front:

1. Nas primeiras linhas do arquivo `./main-page/js/scripts.js`, substitua a variável `API_PI` pela que está comentada logo abaixo, substituindo a porta (8000) conforme necessário
2. Feito isso, basta executar um servidor local à partir da página `./main-page`, com o auxílio da extensão *LiveServer* no VSCode, ou através do comando **`python3 -m http.server`** no console.

Execução local do back:

1. No arquivo `./python-api/config.py`, substitua o valor da variável `"PASSWORD"` pela string `"efsDQy8Mt9uA"`. Isso dará acesso ao banco de dados hospedado no Vercel
2. No arquivo `./python-api/app.py`, adicione o atributo `port=8000` dentro do método `app.run()`, dessa forma: `app.run(port=8000)`
3. Por fim, execute o arquivo `app.py`