UNIVERSIDADE SALVADOR – UNIFACS

Ciência da Computação

Equipe:

Nicolas Batista Lima - RA: 1272217454,

Raphael Freitas Oliveira - RA: 1272211685,

João Luccas Lordelo Marques - RA: 1272212801

Data: 27/11/2023

RELATÓRIO DE SISTEMAS DISTRIBUÍDOS

Salvador, BA

2023

Descrição dos requerimentos de softwares

Tecnologias necessárias para execução do projeto:

- Python3 com Pip
- Node 18+

Tecnologias utilizadas:

Servidor:

- FastAPI (API Rest)
- Jinja2 (Templates dos relatórios)
- Weasyprint (Geração de PDFs)
- Pydantic (Validação)

Cliente:

- ReactJs
- MaterialUI (Componentes e estilização)
- DateFNS (Manipulação de datas)
- Vite (Build)

Banco de dados:

- Sqlite3

A justificativa da escolha da tecnologia

Escolhemos utilizar o Python com o Framework FastAPI no backend, pois é uma combinação que possibilita a criação de APIs rest de forma expressiva, limpa e de fácil compreensão utilizando JSON como formato de comunicação. O Python também já fornece em sua biblioteca nativa um pacote para conexão com banco de dados Sqlite3, o que se encaixou muito bem com os requisitos desse projeto.

O cliente desenvolvido foi um Front End web construído em ReactJs. O React é uma biblioteca robusta para construção de interfaces de usuário em uma velocidade incrível, principalmente quando aliada a um framework de componentes como o MaterialUI. Com a utilização dessas duas ferramentas, pudemos focar os esforços de desenvolvimento na comunicação cliente - servidor.

Instruções para instalação e execução da aplicação

1. **Instalar o pip:**

   python -m pip install --upgrade pip

2. **Clonar o projeto com o git e usar na sua IDE de preferência.**

   git clone https://github.com/nicklima-amigos/EntregaFinalA3-SD.git

3. **Criar ambiente virtual do Python:**

   Linux:

   cd ./src/api && python3 -m venv .venv && source ./.venv/bin/activate && cd ../../

   Windows (PowerShell):

   cd .\src\api && python -m venv .venv && .\.venv\Scripts\Activate && cd ..\..\

4. **De dentro da pasta raiz do projeto, rodar o comando para instalar as dependências**

   npm install && npm run install:all

5. **Executar server e client**

   npm run start:all

6. **Acessar a documentação da API no Swagger no seu navegador**

   http://localhost:8000/docs

Apresentação e detalhamento sobre a arquitetura, estratégia e algoritmos utilizados.

A utilização do FastAPI para declaração das rotas permitiu uma organização modular dos arquivos, e a integração do framework com a biblioteca Pydantic permitiu a utilização de classes Python para validação das informações recebidas pelo servidor.

O FastAPI também possui um sistema intuitivo de tratamento de erros, permitindo a configuração dos erros corretos em cada uma das situações específicas.

Finalmente, utilizamos a biblioteca Jinja para criar templates HTML utilizados como esqueleto para os relatórios. O Jinja fica responsável pela renderização desses templates com as informações corretas quando o relatório é solicitado pelo cliente. Após a renderização do relatório, utilizamos a biblioteca Weasyprint para gerar arquivos PDF a partir do HTML renderizado. Esses arquivos são finalmente enviados para o cliente em formato de arquivo (download).

Para fazer a comunicação entre cliente e servidor, utilizamos a API fetch nativa do navegador, que foi abstraída por nós em serviços específicos responsáveis por fazer as requisições para os endpoints necessários do servidor nos momentos necessários.