Documentação Técnica do Projeto

Nome do projeto: **TechChallengeFase2**

**Introdução**

Este projeto é parte do Tech Challenge Fase 2, desenvolvido pelo grupo 28

**Objeto**

Disponibilizar uma plataforma onde professores possam realizar postagem de suas aulas e elas serem consumidas pelos seus alunos.

**Instalação – Pré Requisitos**

-Node.js

-NPM

-Docker desktop

-Git

**Passos Iniciais**

1.Realizar o clone do repositório [*https://github.com/techchallengegroup28/TechChallengeFase2.git*](https://github.com/techchallengegroup28/TechChallengeFase2.git)

2.Abrir o projeto com sua IDE de desenvolvimento, navegar até o diretório TechChallengeFase2

3.No terminal de comando da sua IDE executar o comando ***npm install*** para baixar as dependências do projeto.

4.No terminal executar o comando ***docker compose build***este comando é responsável por construir as imagens do Docker, serão criadas duas imagens uma contendo o serviço web com a api e outra contendo o banco de dados postgres com o database FIAP e a tabela Post.

5.No terminal executar o comando ***docker-compose up -d*** este comando é responsável por iniciar os containers da aplicação e banco de dados.

6.Abrir o navegado se sua escolha e acessar o endereço <http://localhost:3000/api-docs/> será exibida a página contendo a documentação da api, isso indica que a api está disponível para consulta, inclusão, delete e atualização através dos verbos get,post,put e delete.

**Arquitetura do Projeto**

Desenvolvido na arquitetura node.js pois oferece várias vantagens, especialmente para aplicações que exigem alta escalabilidade e desempenho.

1. Servidor e Roteamento
   * Express: Utilizado para criar o servidor e gerenciar as rotas da aplicação.
   * Morgan: Middleware para logging de requisições HTTP.
   * Cookie-Parser: Middleware para parsear cookies.
2. Configuração e Variáveis de Ambiente
   * Dotenv: Utilizado para carregar variáveis de ambiente a partir de um arquivo [.env](vscode-file://vscode-app/c:/Users/Marcelo/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).
3. Banco de Dados
   * pg: Drivers para conexão com bancos de dados PostgreSQL.
   * Sequelize: ORM (Object-Relational Mapping) para interagir com o banco de dados de forma mais simples e estruturada.
4. Testes
   * Jest: Framework de testes para JavaScript.
   * Supertest: Utilizado para testar endpoints HTTP.
5. Documentação da API
   * Swagger-jsdoc: Utilizado para gerar a documentação da API a partir de comentários no código.
   * Swagger-ui-express: Middleware para servir a documentação da API gerada pelo Swagger.

**Estrutura**

TechChallengeFase2/

├── .github/

│ └── workflows/

│ └── main.yml # Configuração de integração contínua

├── bin/

│ └── www # Ponto de entrada do servidor

├── controllers/ # Controladores para gerenciar a lógica de negócios

│ └── indexController.js

│ └── postController.js

├── databases/

│ └── database.js # Conectar com o banco de dados

├── models/

│ └── post.js

├── public/

│ └── stylesheets/

│ └── style.css # Configuração de integração contínua

│ index.html

├── routes/ # Definição das rotas da aplicação

│ ├── index.js

│ └── post.js

├── tests/ # Testes unitários e de integração

│ ├── post.test.js

├── docs/ # Documentação da API gerada pelo Swagger

│ └── swagger.json

├── views/ # Templates de visualização (se aplicável)

│ └── index.js

├── .env # Variáveis de ambiente

├── .env.docker #

├── .gitignore # Arquivos e diretórios a serem ignorados pelo Git

├── Dockerfile # Arquivo de configuração do Docker

├── docker-compose.yml

├── Dockerfile.db

├── FIAP.sql # Contêm as tabelas e insert do banco de dados

├── package.json # Dependências e scripts do projeto

├── swaggerConfig.js

├── README.md # Documentação do projeto

└── app.js # Arquivo principal da aplicação

**Testes**

Como rodar os testes

Ferramentas de teste utilizadas

**Deploy**

O Deploy é realizado quando fazemos um ***push*** no repositório remoto, automaticamente o git actions é ativado e realizar o build na aplicação, ao finalizar o processo envia a imagem criada para o dockerHub

As ferramentas utilizadas para Deploy são o git, git actions, Docker e dockerHub