



## Portfólio

Kauan Vinícius, dev Full-Stack

### 1- Sumário

1. Sumário;
2. Introdução;
3. Objetivo;
4. Tecnologias:
  - 4.1 Front-End;
  - 4.2 Back-End;
  - 4.3 Banco de Dados;
  - 4.4 ORM;
  - 4.5 UI/UX e Protótipo;
  - 4.6 Versionamento;
  - 4.7 Teste de API (Endpoints);
  - 4.8 Repositório e Documentação;
  - 4.9 CI/CD;
  - 4.10 DevOps e Deploy;
  - 4.11 Código Fonte;
  - 4.12 Scripts e Automação;
  - 4.13 Organização.
5. Arquitetura:
  - 5.1 Descrição de Camadas.
6. Banco de Dados:
  - 6.1 Tipo de Banco de Dados;
  - 6.2 Modelagem de Tabelas.

- 7. Endpoints:
  - 7.1 Projects;
  - 7.2 Contacts.
- 8. Documentação da API;
- 9. Fluxo de Funcionamento do Sistema;
- 10. Como Rodar o Projeto:
  - 10.1 Pré-Requisitos;
  - 10.2 Etapas.
- 11. Futuras Evoluções;
- 12. Criador Oficial;
- 13. Considerações.

## **2- Introdução**

Este documento foi desenvolvido com o intuito de descrever tecnicamente o planejamento e desenvolvimento do meu portfólio com Front-End e Back-End integrados, abordando a arquitetura, tecnologias, estruturas, modelagem de banco de dados e o fluxo do projeto.

Criado com o foco em boas práticas de engenharia de software, código limpo e organizado e integração entre camadas, servindo vitrine profissional e base para expansões futuramente.

## **3- Objetivo**

Seu objetivo principal é demonstrar competências práticas em desenvolvimento moderno, com:

- API REST;
- Dados relacionais;

- Organização de código;
- Frameworks e bibliotecas.

## **4- Tecnologias**

### **4.1 Front-End**

- HTML5;
- REACT;
- TYPESCRIPT;
- BOOTSTRAP;
- AXIOS.

### **4.2 Back-End**

- FASTAPI;
- PYTHON;
- REST;
- JSON;
- HTTPS.

### **4.3 Banco de Dados**

- MYSQL;
- SQLITE.

### **4.4 ORM**

- SQLALCHEMY.

### **4.5 UI/UX e Protótipo**

- FIGMA.

### **4.6 Versionamento**

- GIT.

#### **4.7 Teste de API (EndPoints)**

- INSOMNIA.

#### **4.8 Repositório e Documentação**

- GITHUB;
- MARKDOWN;
- GOOGLE DOCS.

#### **4.9 CI/CD**

- GITHUB;
- INSOMNIA.

#### **4.10 DevOps e Deploy**

- DOCKER;
- MICROSOFT AZURE;
- RENDER.

#### **4.11 Código Fonte**

- VISUAL STUDIO CODE.

#### **4.12 Scripts e Automação**

- POWERSHELL.

#### **4.13 Organização**

- GITHUB;
- DOCKER.

### **5- Arquitetura**

O sistema completo segue uma arquitetura baseada na separação das responsabilidades facilitando o desenvolvimento e a manutenção.

## 5.1 Descrição de Camadas

- **CORE:** Configuração de banco e dependências globais;
- **MODELS:** Tabelas do banco de dados;
- **SCHEMAS:** Validação e serialização de dados;
- **CRUD:** Operações de acesso ao banco de dados;
- **MAIN:** Inicialização;
- **ROUTERS:** Endpoints.

## 6- Banco de Dados

### 6.1 Tipo de Banco de Dados

- Relacional;
- SQLite;
- ORM.

OBS: O Banco de Dados é gerado automaticamente ao longo da inicialização da aplicação.

### 6.2 Modelagem de Tabelas

**Tabela:** Projects

CAMPO	TIPO	DESCRIÇÃO
id	Integer	Identificador Único (1, 2, 3...)
title	String	Título do Projeto
description	Text	Descrição do Projeto
technologies	String	Tecnologias Utilizadas
github_url	String	Repositório Onde Está o Projeto

demo_url	String	Demonstração do Projeto
created_at	DateTime	Data de Criação

**Tabela:** Contacts

CAMPO	TIPO	DESCRIÇÃO
id	Integer	Identificador Único
name	String	Nome do Remetente
email	String	E-mail de contato
message	Text	Mensagem Enviada
created_at	DateTime	Data de Envio

## 7- Endpoints

### 7.1 Projects

- **GET**/ Retorna a lista de projetos;
- **POST**/ Cria um novo projeto de banco.

### 7.2 Contacts

- **POST**/ Registra uma nova mensagem de contato.

## 8- Documentação da API

A API nos disponibiliza uma documentação automatizada por meio do Swagger, assim permitindo visualização e testes de endpoints.

Acesse: <https://127.0.0.1:8000/docs>

## 9- Fluxo de Funcionamento do Sistema

1. Usuário acessa o Front-End;

2. O Front-End faz as requisições HTTP;
3. A API processa a requisição;
4. o Banco é consultado ou atualizado;
5. A API retorna os dados em forma de JSON;
6. O Front-End apresenta as informações formatadas ao usuário.

## **10- Como Rodar o Projeto**

### **10.1 Pré-Requisitos**

- Python 3 instalado;
- Ambiente Virtual criado e ativado;
- Node JavaScript instalado;

### **10.2 Etapas**

1. `python -m venv env;`
2. `env\Scripts\activate;`
3. `pip install -r requirements.txt;`
4. `python -m uvicorn app.main:app --reload.`

Aplicação disponível: <https://127.0.0.1:8000>

URL Front-End: <http://localhost:5173/>

## **11- Futuras Evoluções**

- Deploy;
- Testes Automáticos;
- Containerização em Docker.

## **12- Criador Oficial**

**Nome:** Kauan Vinícius;

**Área:** Desenvolvimento de Sistemas;

**Setor:** ETS - DTA;

**Habilidade:** Full-Stack Júnior;

**Cargo:** Jovem Aprendiz - 3o Semestre;

**Empresa:** Robert Bosch;

**Projeto:** Portfólio Full-Stack.

## **13- Considerações**

O projeto demonstra uma base para aplicações modernas, seguindo práticas de desenvolvimento, organização de código e integração entre sistemas. A estrutura permite uma fácil expansão e adaptabilidade para ambientes de produção.