

## NÍVEL BÁSICO



PROJETO 09

(CONTEÚDO DISPONÍVEL) {  
**PROJETO;**  
**CASE DE;**  
**ESTUDO;**  
(end);  
})();

#PORTFÓLIOBOOSTPROGRAM

**CONHECIMENTOS REQUIRIDOS:**



**FULL-STACK**

# WIREFRAME

Lista de projetos em thumb (miniatura)

Nome do projeto

descrição

feature 1

feature 2

feature 3

stack de tecnologia e como foi feito

stack de tecnologia e como foi feito

stack de tecnologia e como foi feito

Imagem

# PROJETO CASE DE ESTUDO

Crie uma **lista de projetos** que inclua **estudos de caso** e **detalhes sobre o projeto**.

## TECH STACK

- ➔ React
- ➔ NodeJS
- ➔ Prisma
- ➔ BaaS(Back-as-a-service)
- ➔ Conhecimento Básico de serverless



## LIBRARIES

- ➔ React-router



## BRIEFING

Ser capaz de mostrar seu trabalho é uma ótima maneira de ajudar outras pessoas a entender como você pensa e aprender sobre as experiências que teve ao longo do caminho.

Isso é especialmente bom para candidatos a emprego que precisam se destacar entre os outros.

## NÍVEL 1

Mostrar os vários projetos em que você trabalhou ajuda um possível empregador ou colaborador a saber em quais tipos de projetos você trabalhou.

Crie uma página que inclua uma lista de alguns dos melhores projetos em que você trabalhou.

## NÍVEL 2

Saber no que você trabalhou é útil, mas ser capaz de ajudar alguém a entender o contexto é ainda melhor.

Adicione imagens e detalhes do projeto para cada projeto.

## NÍVEL 3

Para entender completamente o seu processo de pensamento, fornecer estudos de caso pode ajudar a orientar alguém em suas decisões e desafios que o levaram ao projeto finalizado.

Adicione estudos de caso para cada projeto que detalha o processo e os desafios do projeto.





## REQUISITOS DETALHADOS

### Back-end:

- ➡ Crie um **schema de banco de dados** para armazenar dados do projeto. O código de **back-end** precisará criar um schema de banco de dados para armazenar dados do projeto.
- ➡ O esquema do banco de dados pode ser criado usando uma ferramenta como o **postgreSQL**. O esquema do banco de dados precisará incluir **tabelas** para armazenar **títulos de projetos, descrições, imagens, estudos de caso e outros metadados**.
- ➡ Para armazenar o banco de dados **postgreSQL** via plataformas **serverless** (algo que você verá muito por aqui) eu recomendo utilizar os serviços

RAILWAY

ou

VERCEL



- ➡ Assim você terá seus bancos na **cloud**, diminuindo o tempo de **configuração e aumentando a escalabilidade**.



Para **ORM**, utilize [Prisma](#) para poder integrar com seu **BaaS** ter toda a **infra-estrutura serverless**. Veja maiores informações sobre prisma [aqui](#):

## PRISMA



Crie uma **API REST** para expor os dados do projeto ao **front-end**. A **API REST** pode ser criada usando uma estrutura como **Express**.



A **API REST** precisará fornecer métodos para buscar uma lista de projetos, buscar um projeto específico e adicionar um novo projeto.

### Front-end:



Busque dados da **API REST**. A **API REST** pode ser usada para buscar uma lista de projetos, buscar um projeto específico e adicionar um novo projeto.



Crie uma lista de projetos. O **código frontend** precisará criar uma lista de projetos. A lista de projetos pode ser criada usando um **componente React**.



O **componente React** precisará buscar dados da **API REST** e exibir os projetos em uma lista.



Uma dica técnica aqui é fazer o **fetch da lista dos projetos** e otimizar utilizando os **hooks useState** para fazer **fetch da API** assim que o componente for renderizado. Dependendo do tamanho da lista talvez seja interessante um [useCallback](#).

- ➡ **Adicione detalhes do projeto.** O código **front-end** precisará adicionar detalhes do projeto. Os detalhes do projeto podem ser adicionados usando um **componente React**.
- ➡ O componente React precisará buscar dados da **API REST** e exibir os detalhes do projeto, como **título, descrição, imagens e estudos de caso**.
- ➡ **Adicione imagens do projeto.** O código do **front-end** precisará adicionar imagens do projeto. As imagens do projeto podem ser adicionadas usando uma **biblioteca de imagens de terceiros**, como **Cloudinary** ou **fileStack**.
- ➡ A biblioteca de imagens de terceiros precisará ser configurada para **armazenar e servir** as imagens do projeto.
- ➡ **Adicione estudos de caso.** O código **front-end** precisará adicionar estudos de caso. Os estudos de caso podem ser adicionados usando um **componente React**.
- ➡ O componente React precisará buscar dados da **API REST** e exibir os **estudos de caso**.

#### Vale lembrar que:

- ➡ As duas partes devem ser **dissociadas** para que possam ser desenvolvidas de forma independente.

#### O que isso quer dizer?

- ➡ Separe a lógica em **folders** que tenham nomes bem **semânticos e distintos** de acordo com a sua responsabilidade. **Exemplo: Web para Front e API para back.**

