

## NÍVEL BÁSICO



### PROJETO 11

(CONTEÚDO DISPONÍVEL) {  
**GERENCIADOR;**  
**DE;**  
**METAS;**  
(end);  
})();

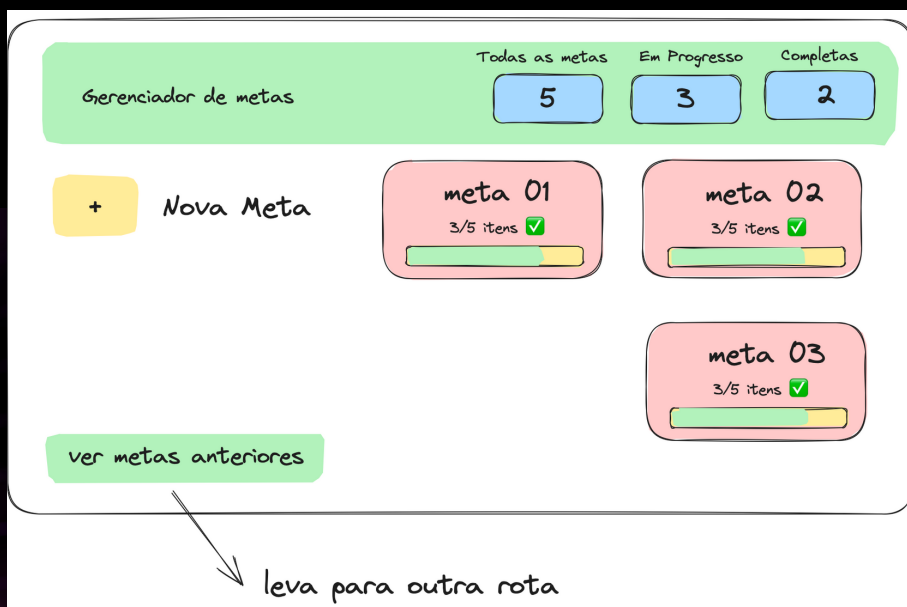
#PORTFÓLIOBOOSTPROGRAM

### CONHECIMENTOS REQUIRIDOS:



**FULL-STACK**

# WIREFRAME



# GERENCIADOR DE METAS

Crie um painel de **gerenciamento de metas** para **adicionar, rastrear e modificar** suas metas.



## TECH STACK

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



## BRIEFING

As metas são uma ótima maneira de nos ajudar a definir um desafio que queremos nos esforçar para alcançar. Além disso, tornar essas metas públicas é uma forma de ajudar a prestar contas a outras pessoas e obter o apoio da comunidade.



## NÍVEL 1

Para começar com metas, você precisa tê-las. Se você ainda não tem, reserve um tempo para pensar sobre isso, mas, no final das contas, esses serão seus objetivos públicos.

Crie uma lista de metas e coloque-as em seu site.

## NÍVEL 2

Seus objetivos são públicos, mas é difícil acompanhá-los. Cada atualização de progresso requer uma alteração de código e atualização manual da interface do usuário.

Nessa etapa adicionemos as metas a uma **API** e mensure uma maneira de quebrar ela em entregáveis menor para termos um **%** de quanto foi executado.

## NÍVEL 3

Altere a Interface para que **mostre o %** de quantas tarefas suas foram concluídas. Baseadas no % relativo de cada uma das tarefas.

Ex: 2 tarefas em 50% de andamento e uma em 75% significa que você tem 66,67% de suas tarefas, o que é um pouco mais de 65%.

50 %





# REQUISITOS DETALHADOS

## Back-end:

- ➔ Crie um novo aplicativo **Next.js**, iremos utilizar muitas vezes o NextJS por aqui. O nextJS é um **framework full-stack** que facilita nossa interação nas duas camadas da aplicação seja no front ou no back. Veja mais em:

## NEXTJS

- ➔ Instale um banco de dados. Por exemplo, você pode usar **MySQL**, **Postgres** ou **MongoDB**.
- ➔ Crie um **schema de banco de dados** para armazenar cada meta.
- ➔ O schema do banco de dados deve incluir **tabelas** para armazenar o **título da meta**, **descrição**, **data de vencimento** e **progresso**.
- ➔ Crie uma **API REST** para expor os dados de metas ao **front-end**. A API REST deve fornecer métodos para **criar**, **ler**, **atualizar** e **excluir metas**.

## Front-end:

- ➔ Instale o **React Router**.
- ➔ Busque dados da **API REST**. O código front-end precisará buscar os dados de metas da API REST e armazená-los em uma variável de estado.

- ➡ Crie uma **lista de metas**. O código do **front-end** precisará criar uma lista de metas com base nos dados armazenados na variável de estado.
- ➡ Adicione **campos de progresso** com base nas caixas de seleção dentro de cada meta. O código **front-end** precisará adicionar campos de progresso a cada meta com base no número de caixas de seleção marcadas.
- ➡ Pense da seguinte maneira, ao clicar na meta você será redirecionado para uma rota aonde exibirá em formato de **checklist**.

```
(input type="checkbox").
```

- ➡ O calculo percentual de cada uma das metas para ser considerado será o número checkpoints finalizados / numero total de checkpoints.
- ➡ Mostra o **progresso** na **interface do usuário** com base em valores relativos. O código front-end precisará mostrar o progresso de cada meta na interface do usuário, com base nos valores relativos das **caixas de seleção**.
- ➡ Vale estratégias de **z-index** e duas **divs** intercaladas uma como **background** total e outra preenchendo a posição de acordo com o **percentual**.
- ➡ Crie uma função em **javascript** para definir o percentual concluído por aproximação. Um **map** fazendo a atribuição desses valores em **JS** para **props** em **CSS** para interface seria uma ótima forma de desenvolver isso.
- ➡ Ao clicar em "**ver metas anteriores**" devemos direcionar para uma nova rota aonde iremos exibir **SOMENTE** as metas já **completas**.

