

NÍVEL BÁSICO



PROJETO 03

```
(CONTEÚDO DISPONÍVEL) {  
GERENCIADOR;  
DE;  
ORÇAMENTO;  
(end);  
})();
```

#PORTFÓLIOBOOSTPROGRAM

CONHECIMENTOS REQUIRIDOS:




FRONT-END

WIREFRAME

Orçamento

intervalo de data até



Todos os custos

Compra	Categoria	Data	Custo

GERENCIADOR DE ORÇAMENTO

Crie um **gerenciador de orçamento** para rastrear e gerenciar finanças e despesas.

TECH STACK

- ➔ React
- ➔ TailwindCSS
- ➔ ViteJS
- ➔ Vercel (conhecimento básico de serverless)



LIBRARIES

- ➔ Google Sheets API
- ➔ Sheet2API



BRIEFING

Gerenciar finanças é difícil. Acompanhar tudo o que você comprou inclui muita matemática que ninguém quer fazer.

Além disso, você pode não estar disposto a fornecer a uma empresa da web todos os seus dados financeiros para fazer isso por você.

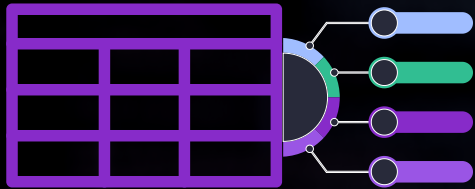


Esse projeto pode ser o projeto que irá fazer você brilhar no **front-end**, usaremos somente recursos FE neste projeto então capriche nas transições, animações e estilos!

NÍVEL 1

A forma mais comum de olhar para os dados financeiros é uma **tabela**. É uma boa maneira de classificar visualmente grandes quantidades de dados usando colunas para somar números.

Crie uma **tabela editável** na qual você pode adicionar despesas como linhas.



NÍVEL 2

Como qualquer outra coisa, ter um site estático onde gerenciemos recursos no código é útil e eficiente, mas talvez não seja a maneira mais fácil de gerenciá-lo.

Adicione seu orçamento ao **Planilhas Google** e use a **Sheets2API** para acessar os dados.

NÍVEL 3

O uso de tabelas é uma maneira comum de examinar os dados, mas às vezes você deseja uma visão geral simples, em vez de ter que escolher cada linha e descobrir por conta própria.

Crie um **painel de estatísticas** e informações simples que você precisaria para uma rápida olhada. **Como desafio extra:** crie um **date-picker** para filtrar por datas a seleção de orçamento

REQUISITOS DETALHADOS

Yuri, porque ViteJS no Lugar de NextJS neste projeto?

Boa parte dos projetos que estamos desenvolvendo no guia são **Full-stack**. Esse é somente **front-end**.

Logo não há necessidade de aumentar a complexidade dele adicionando **NextJS**. Vamos de **ViteJS**.

A primeira coisa que você deve fazer é seguir a documentação oficial do **viteJS** para instalação dele em seu novo projeto em:

VITEJS.DEV



Após instalação siga o manual para as seguintes etapas.

Criar tabela

- ➔ Crie um componente **React** chamado **BudgetTable**.
- ➔ No componente **BudgetTable**, crie uma tabela com duas colunas: **Categoria** e **Quantidade**.
- ➔ No componente crie uma função chamada **addRow** que recebe uma categoria e uma quantidade como argumentos.
- ➔ Na função **addRow**, adicione uma nova linha à tabela com a **categoria** e a **quantidade** especificadas.
- ➔ Renderize o componente **BudgetTable** no componente **App**.

Adicionar linhas de dados

- ➡ Crie uma variável de estado chamada `budgetData` no componente App.
- ➡ No componente App, adicione uma função chamada `loadBudgetData` que carrega os dados do orçamento de uma API, no nosso caso vamos usar:

SHEET2API



Que vai converter seu `spreadsheet` numa `API` para consumo e inserção de dados.

- ➡ Na função `loadBudgetData`, retorne um `array de objetos`, cada um com uma propriedade `categoria` e `quantidade`.
- ➡ No componente App, use o `hook useState` para definir a variável de estado `budgetData` para o valor retornado pela função `loadBudgetData`.
- ➡ No componente `BudgetTable`, use a variável de estado `budgetData` para preencher a tabela com os dados.

Adicionar formulário para adicionar linhas

- ➡ Crie um componente `React` chamado `BudgetForm`.
- ➡ No componente `BudgetForm`, crie um formulário com dois campos de entrada: `categoria` e `quantidade`.
- ➡ No componente `BudgetForm`, crie um botão chamado `Adicionar linha` e crie as funções necessárias para adicionar uma linha.

Adicionar a capacidade de remover linha

- ➔ No componente `BudgetTable`, adicione um botão a cada linha chamado `Remover linha`.
- ➔ No botão `Remover linha`, chame a função `removeRow` no componente `BudgetTable` com o índice da linha a ser removida.

Adicionar dados ao Sheets

- ➔ Crie uma planilha do `Google Sheets`.
- ➔ Na planilha do `Google Sheets`, crie uma planilha chamada `Orçamento`.
- ➔ Na planilha `Orçamento`, crie duas colunas: `Categoria` e `Quantidade`.
- ➔ No componente `BudgetTable`, chame a função `addRowToSheets` com os valores dos campos de `entrada categoria` e `quantidade`.
- ➔ A função `addRowToSheets` deve pegar os valores dos campos de entrada `categoria` e `quantidade` e adicioná-los à planilha `Budget` no `Google Sheets`.

Obter dados da API

Vamos utilizar o:

SHEET2API



Ele irá servir como nosso `database`.

Use o Google OAuth (opcional)

- ➔ Crie um cliente **Google OAuth**.
- ➔ No componente **App**, chame a função **authenticateUser** para autenticar o usuário com o **Google OAuth**.
- ➔ A função **authenticateUser** deve chamar o **cliente Google OAuth** para autenticar o usuário e retornar o **ID do usuário**.
- ➔ No componente **BudgetTable**, use o **ID do usuário** para obter os dados do orçamento do usuário da **API**.

Adicionar painel de estatísticas

- ➔ Crie um componente **React** chamado **BudgetStatsDashboard**.
- ➔ No componente **BudgetStatsDashboard**, crie um **gráfico** que mostra o orçamento **total, gasto e restante**.
- ➔ No componente **BudgetStatsDashboard**, use a variável de estado **budgetData** para preencher o gráfico com os dados.

