Material de apoio → SETEC.

1° passo → Acessando o repositório:

Clone o repositório com a base do projeto no GitHub.

OBS: Caso não tenha uma conta no GitHub, conecte com sua conta do google, crie rapidamente uma conta com sua conta do google e copie o repositório.

https://github.com/VitorHugoBelorio/ChatBotSETEC

2° passo → Clonar e abrir a base do projeto:

2.1 passo → clonar o repositório:

Após acessar o repositório digite os seguintes comandos no gitbash ou no powershell.

Para clonar o repositório:

```
git clone https://github.com/VitorHugoBelorio/ChatBotSETEC
```

2.2 passo → navegar para a pasta do projeto:

Digite para entrar na pasta do repositório.

```
cd /ChatBotSETEC
```

2.3 passo → abrir o projeto:

Digite para abrir o projeto no **vs code**.

```
code .
```

3° passo → Configurar o ambiente:

3.1 passo→ instalando as dependências do Laravel:

Com o projeto aberto, abra um terminal no **vs code**, rode o comando:

```
composer install
```

3.2 passo → gerando o arquivo .env:

No terminal do **vs code**, rode o comando para copiar o modelo do .env.example em um arquivo de configuração .env:

```
cp .env.example .env
```

3.3 passo → completar o arquivo .env:

3.3.1 passo → gerando api-key:

1. Faça login na sua conta do google
2. Acesse e selecione para entrar com sua conta do google que acabou de logar https://aistudio.google.com/app/api-keys.
3. Crie uma api-key e copie-a
4. No arquivo `.env` copie essa api-key no campo `GEMINI_API_KEY=` (sem aspas e sem espaços)

3.4 passo → Fazer o Laravel buscar as informações corretas no .env

Vá até a pasta config e entre no arquivo services.php.

Nesse arquivo coloque as identificações da ApyKey e da URL que vamos usar e que estão no .env

```
'gemini' ⇒ [
    'api_key' ⇒ env('GEMINI_API_KEY'),
    'url' ⇒ env('GEMINI_API_URL'),
],
```

3.5 passo → gere o banco de dados e as migrations base do Laravel:

1. Abra o XAMPP e ligue o servidor MySQL
2. Abra o MySQL Workbench e dê 2 cliques na conexão para abri-la

3. Digite o comando e execute:

```
create database chat_ia;
```

4. Digite o seguinte comando para gerar as tabelas padrões do Laravel:

```
php artisan migrate
```

3.6 passo → gerar chave da aplicação Laravel:

```
php artisan key:generate
```

4° passo → Criar a controller:

No terminal digite o comando abaixo para que o artisan gere o arquivo da controller:

```
php artisan make:controller ChatController
```

4.1 passo → Criar a pasta e o arquivo Service:

Nesse passo vamos criar manualmente o arquivo responsável por realizar a validação e o envio da mensagem.

Essa pasta será criada dentro da pasta app.

`Services/GeminiService.php`

Prompt usado para configurar o assistente:

```
"Você é um assistente de estudos voltado para alunos da área de tecnologi
a, com foco em:
   - Programação
   - Banco de Dados
   - Redes de Computadores
   - Engenharia de Software
   - Desenvolvimento Web
```

Seu papel é gerar conteúdos educativos e técnicos de forma clara, organizada e didática.
Siga sempre o formato abaixo nas respostas:

1. **Explicação resumida (até 5 linhas)**
Apresente uma explicação direta e fácil de entender sobre o tema solicitado.

2. **Questões de estudo (3 a 5 perguntas)**
Crie perguntas mistas — algumas de múltipla escolha, outras dissertativas — que ajudem o estudante a refletir e fixar o conteúdo.

3. **Exemplo prático (quando aplicável)**
Mostre uma aplicação prática ou um trecho de código simples que demonstre o conceito em uso.

4. **Dica de estudo**
Dê uma dica rápida para memorizar, revisar ou relacionar o tema a situações reais da área de TI.

O tom das respostas deve ser educacional, motivador e técnico, como o de um tutor universitário que ajuda alunos a entenderem os fundamentos da computação."

5° passo → Adicionar o código JS de conversão no arquivo index:

```
<script>
const form = document.getElementById("chat-form");
const chatBox = document.getElementById("chat-box");
const messageInput = document.getElementById("message");

// Envio de mensagem
form.addEventListener("submit", async function(e) {
  e.preventDefault();
```

```
    let message = messageInput.value.trim();
    if (!message) return;

    // Adiciona mensagem do usuário
    let userMsg = document.createElement("div");
    userMsg.classList.add("d-flex", "justify-content-end", "mb-2");
    userMsg.innerHTML = `<div class="bg-primary text-white p-2 rounded-
3" style="max-width: 75%; white-space: pre-wrap;">${message}</div>`;
    chatBox.appendChild(userMsg);

    // Limpa campo imediatamente
    messageInput.value = "";
    messageInput.focus();

    // Adiciona indicador de carregamento
    let loadingId = 'loading-' + Date.now();
    let loadingMsg = document.createElement("div");
    loadingMsg.classList.add("d-flex", "justify-content-start", "mb-2");
    loadingMsg.id = loadingId;
    loadingMsg.innerHTML = `<div class="bg-light text-secondary p-2 roun
ded-3 border"><b>IA:</b> Pensando...</div>`;
    chatBox.appendChild(loadingMsg);
    chatBox.scrollTop = chatBox.scrollHeight;

    try {
        let response = await fetch("{{ route('chat.send') }}", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
                "X-CSRF-TOKEN": "{{ csrf_token() }}"
            },
            body: JSON.stringify({ message })
        });

        if (!response.ok) throw new Error('Erro na resposta da API: ' + respons
e.status);
        let data = await response.json();
```

```
        // Remove indicador de carregamento
        document.getElementById(loadingId)?.remove();

        // Adiciona resposta da IA formatada
        let iaMsg = document.createElement("div");
        iaMsg.classList.add("d-flex", "justify-content-start", "mb-2");

        // Substitui \n por <br> e mantém formatação
        let formattedText = String(data)
            .replace(/\n/g, '<br>')
            .replace(/\*\*(.*?)\*\*/g, '<b>$1</b>'); // suporte a **negrito**

        iaMsg.innerHTML = `<div class="bg-light border p-2 rounded-3" style
="max-width: 75%; white-space: pre-wrap;"><b>IA:</b> ${formattedText}
</div>`;
        chatBox.appendChild(iaMsg);

    } catch (error) {
        document.getElementById(loadingId)?.remove();
        console.error('Erro:', error);

        let errorMsg = document.createElement("div");
        errorMsg.classList.add("d-flex", "justify-content-start", "mb-2");
        errorMsg.innerHTML = `<div class="bg-danger text-white p-2 rounde
d-3" style="max-width: 75%;"><b>Erro:</b> Não foi possível gerar uma re
sposta.</div>`;
        chatBox.appendChild(errorMsg);
    }

    chatBox.scrollTop = chatBox.scrollHeight;
});

// Enter envia / Shift+Enter quebra linha
messageInput.addEventListener("keydown", function(e) {
    if (e.key === "Enter" && !e.shiftKey) {
        e.preventDefault();
        form.dispatchEvent(new Event("submit"));
    }
```

```
});

// Botão limpar chat
document.getElementById("clear-chat").addEventListener("click", function
() {
  chatBox.innerHTML = "";
});
</script>
```

6° passo → Rodando o projeto:

Para rodar localmente o projeto digite o seguinte comando:

```
php artisan serve
```

Para salvar o seu projeto:

Crie um, repositório pessoal e clone-o. Após clona-lo, use o seguinte comando no PowerShell para copiar todos os arquivos do projeto para o seu repositório:

Copiar todos os arquivos e subpastas

```
Copy-Item "C:\ChatBotSETEC\*" -Destination "C:\SeuRepositório" -Recurs
e

# Acessar repositório
cd "D:\SeuRepositório"

# Enviar para o GitHub
git add .
git commit -m "Adiciona projeto ChatBotSETEC"
git push
```