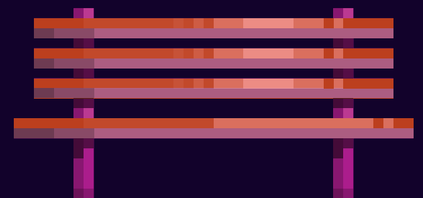

EXERCÍCIO

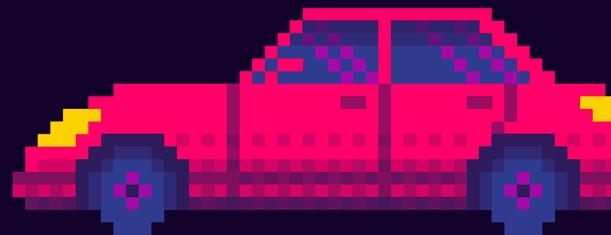
API COM NODE.JS, JAVASCRIPT E EXPRESS

Parabéns! Chegamos à conclusão do nosso módulo de desenvolvimento de APIs usando Node.js, JavaScript e Express. Neste material didático, vamos consolidar o conhecimento criando uma API de lista de tarefas. A API permitirá criar, ler, atualizar e excluir tarefas, com rotas específicas para cada funcionalidade.

Arquivo base de uma API:

```
1 const express = require("express");
2
3 const servidor = express();
4
5 servidor.listen("3000", () ⇒ console.log("Olá mundo"));
```





ESTRUTURA DA TAREFA

Cada tarefa terá os seguintes atributos:

- id: Número inteiro único que identifica a tarefa.
- descricao: Texto que descreve a tarefa.
- finalizada: Um valor booleano (true ou false) indicando se a tarefa foi concluída.
- criadaEm: Data de criação da tarefa.

Attributes

- id: int
- descricao: string
- finalizada: boolean
- criadaEm: Date

ROTAS DA API

```js
const express = require("express");

const router = express.Router();

// Buscar todas as tarefas.
router.get("/tarefas", (request, response) => {});
// Filtrar tarefas pela descrição.
router.get("/tarefas/:descricao", (request, response) => {});
// Filtrar tarefas por data de criação.
router.get("/tarefas/data/:data", (request, response) => {});
// Criar uma nova tarefa.
router.post("/tarefas", (request, response) => {});
// Editar a descrição de uma tarefa.
router.put("/tarefas/:id", (request, response) => {});
//  Finalizar uma tarefa (mudar para true).
router.put("/tarefas/:id/finalizar", (request, response) => {});
// Deletar uma tarefa.
router.delete("/tarefas/:id", (request, response) => {});

```

CONFIGURAÇÃO INICIAL DO PROJETO PELO TERMINAL

Inicie um novo projeto Node.js:

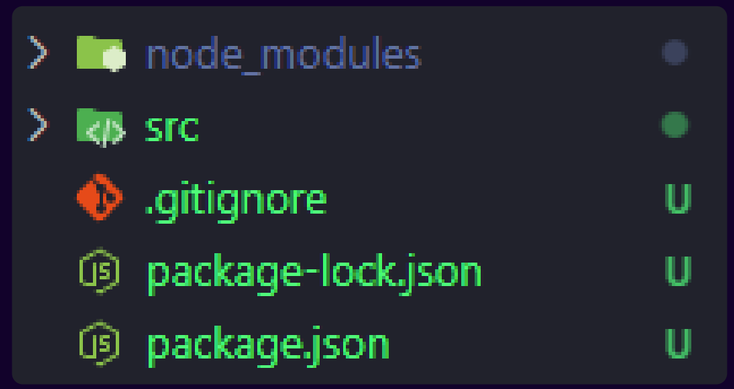
- npm init -y

Instale o Express:

- npm install express

Crie a estrutura do projeto:

- mkdir routes
- touch server.js routes/tarefas.js



Não esquecer de configurar o arquivo **.gitignore** *para não salvar o* **node_modules** *para o git*

ERROS COMUNS

01

CORS: Algumas APIs podem bloquear requisições vindas de domínios diferentes (cross-origin). Isso é chamado de erro de CORS. Esse tipo de erro não pode ser resolvido no front-end, é preciso que o servidor permita a requisição.

02

Verificação da resposta: Nem toda resposta HTTP com status 200 significa sucesso total. Verifique sempre se os dados são os esperados.

ENTENDA MAIS

01 Buscar todas as tarefas (GET /tarefas): Retorna a lista completa de tarefas.

02 Filtrar pela descrição (GET /tarefas/:descricao): Busca tarefas cuja descrição contenha o termo fornecido.

03 Filtrar por data de criação (GET /tarefas/data/:data): Retorna as tarefas criadas na data especificada (formato YYYY-MM-DD).

04 Criar nova tarefa (POST /tarefas): Cria uma nova tarefa com uma descrição fornecida no corpo da requisição.

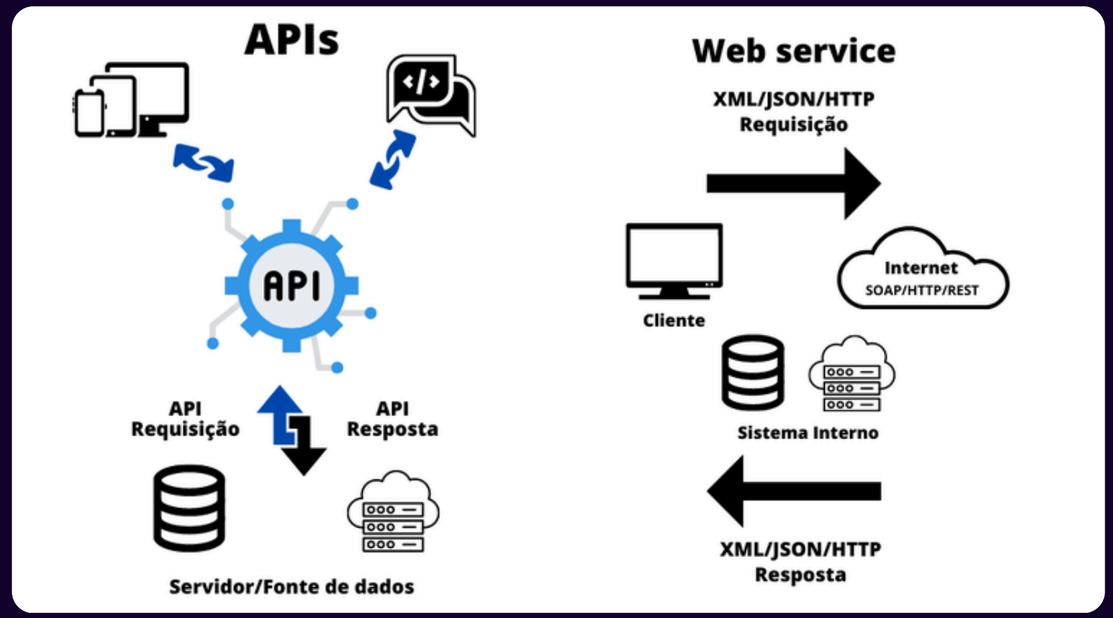
05 Editar a descrição (PUT /tarefas/:id): Atualiza a descrição da tarefa com o ID especificado.

06 Finalizar uma tarefa (PUT /tarefas/:id/finalizar): Marca a tarefa como finalizada (finalizada: true).

07 Deletar tarefa (DELETE /tarefas/:id): Remove a tarefa da lista pelo ID.

BOA SORTE!

Este exercício consolida os conceitos de RESTful APIs que estudamos durante o módulo. A API de tarefas que criamos é um ótimo exemplo de como podemos utilizar Node.js e Express para lidar com dados de forma organizada e eficiente. Teste cada rota, entenda como as requisições HTTP funcionam, e continue praticando!




THANK
YOU