O USO DOS SERVIDORES

O uso de dois arquivos distintos, **servidor.js** e **servicoAPI.js**, define a arquitetura cliente-servidor, que é o padrão ouro na construção de aplicativos.

Aqui está a explicação do papel de cada um:

1. servidor.js: O Lado do Servidor (Backend)

O servidor.js é o seu **Backend** e o coração da sua aplicação.

- **O Que É:** É o programa que roda eternamente no seu computador (ou num servidor na nuvem) e **contém a sua lógica de negócio**. Ele usa a plataforma Node.js (com o Express) para ficar "ouvindo" a porta 8083.
- **O Que Ele Faz:**
    - **Conexão com o Banco de Dados:** É o único responsável por saber as credenciais do MySQL e por executar comandos SQL (INSERT, SELECT, UPDATE, DELETE).
    - **Regras de Negócio:** Aplica validações (ex: "o título não pode ser vazio").
    - **Endpoints (APIs):** Define as "portas" de comunicação. Por exemplo, quando o servidor recebe um POST para /livros, ele sabe que deve incluir um livro no banco.
- **Por Que É Necessário:** O aplicativo móvel (o Cliente) **não pode** se conectar diretamente ao banco de dados por motivos de segurança e eficiência. O servidor age como um **porteiro e tradutor**, protegendo o banco e executando as ações.

Em Resumo: O servidor.js é a **API REST**. Ele decide *como* as operações são feitas.

2. servicoAPI.js: O Lado do Cliente (Frontend)

O servicoAPI.js é um módulo do seu **Frontend** (o aplicativo React Native).

- **O Que É:** É um conjunto de **funções JavaScript** que seu aplicativo usa para se comunicar com o servidor.js.
- **O Que Ele Faz:**
    - **Abstração de Comunicação:** Ele esconde a complexidade de fazer requisições HTTP (fetch) do restante do aplicativo.

    - **Tradução de Ações:** Quando o TelaListaLivros.js precisa dos livros, ele simplesmente chama obterLivros(). O servicoAPI.js traduz isso para uma requisição HTTP GET para http://seu_ip:8083/livros.
    - **Tratamento de Erros:** Ele verifica o código de resposta (ex: 404 ou 500) e lança um erro fácil de entender para a interface.
- **Por Que É Necessário:** Ele centraliza todos os detalhes de comunicação. Se você mudar a porta ou o endereço da API, basta alterar **uma única linha** de código no servicoAPI.js, e o aplicativo inteiro continua funcionando.

Em Resumo: O servicoAPI.js é o **Cliente da API**. Ele decide *o que* deve ser solicitado.

O Fluxo de Comunicação

Para incluir um livro, o fluxo é o seguinte:

1. O aluno clica em **"Criar Livro"** na TelaCriarLivro.js.
2. A tela chama a função criarLivro({ titulo, autor }) dentro do **servicoAPI.js**.
3. O **servicoAPI.js** faz uma requisição HTTP POST para a rota /livros do servidor.
4. O **servidor.js** recebe a requisição, se conecta ao MySQL e executa o INSERT INTO livros....
5. O **servidor.js** envia uma resposta de sucesso (código 201).
6. O **servicoAPI.js** recebe a resposta e a retorna para a TelaCriarLivro.js.

Essa separação é o que permite que seu aplicativo seja seguro, escalável e fácil de manter.