O que é o Back-end?

Backend é responsável por gerenciar e processar as solicitações feitas pelo cliente (navegador web, por exemplo) e enviar as respostas correspondentes.

O backend geralmente consiste em um servidor e um banco de dados conversando.

O servidor é responsável por receber as solicitações do cliente, processá-las e retornar as respostas.

A aplicação, muitas vezes desenvolvido em linguagens como Python, Java, PHP, Ruby, Node.js, entre outras, contém a lógica de negócios e as regras de funcionamento do sistema. O banco de dados é utilizado para armazenar e recuperar informações conforme necessário pelo aplicativo.

Algumas das tarefas comuns realizadas pelo backend incluem:

- autenticação de usuários
- processamento de formulários
- manipulação de dados
- gerenciamento de sessões
- entre outras

Em resumo, o backend é responsável por tornar possível a funcionalidade de um sistema de software, enquanto o frontend é responsável pela interface do usuário e pela interação direta com o usuário final.

Monolito:

Um monolito é uma aplicação ou sistema de software que é **desenvolvido como uma única unidade**.

Isso significa que todas as funcionalidades, componentes e partes do sistema são agrupados e interligados em um único código base. Em termos práticos, um monolito é um aplicativo completo que pode ser implantado e executado em um único ambiente.

API (Interface de Programação de Aplicações):

Em um contexto de desenvolvimento de software, uma API é um conjunto de definições de interface e métodos que permitem que outros aplicativos interajam com o software principal.

Uma API pode ser usada para acessar funcionalidades específicas de um aplicativo ou para permitir a integração entre diferentes sistemas.

> A principal diferença entre um monolito e uma API é a abordagem de arquitetura de software. Um monolito é um sistema completo e autocontido, onde todas as partes estão interligadas e dependem uma das outras. Por outro lado, uma API é um conjunto de interfaces e métodos que permitem a comunicação entre diferentes sistemas ou componentes de um sistema maior.

Rotas em uma API:

Rotas em uma API são **URLs que direcionam requisições HTTP para as funções ou métodos adequados que irão processar essas requisições** e retornar as respostas apropriadas. As rotas definem como os clientes podem interagir com a API, especificando quais endpoints estão disponíveis e quais ações podem ser realizadas em cada um deles.

Exemplo em Laravel:

No framework **Laravel**, as rotas são definidas no arquivo `routes/web.php` ou `routes/api.php`, dependendo se são rotas web (monolíto) ou rotas da API. Abaixo está um exemplo simples de definição de rota em uma API usando Laravel:

```
// routes/api.php

use Illuminate\Support\Facades\Route;

Route::get('/usuarios', 'UsuarioController@index');
Route::get('/usuarios/{id}', 'UsuarioController@show');
Route::post('/usuarios', 'UsuarioController@store');
Route::put('/usuarios/{id}', 'UsuarioController@update');
Route::delete('/usuarios/{id}', 'UsuarioController@destroy');
```

Neste exemplo, temos cinco rotas para operações CRUD básicas em uma entidade "usuário".

Cada rota define um endpoint da API (/usuarios para listar todos os usuários, /usuarios/{id} para mostrar, atualizar ou excluir um usuário específico) e associa a rota a um método do controlador UsuarioController responsável por processar a requisição.

Por exemplo, a rota GET /usuarios acima direciona requisições GET para /usuarios para o método index do UsuarioController, que pode retornar uma lista de usuários em formato JSON.

A rota POST /usuarios direciona requisições POST para /usuarios para o método store do UsuarioController, que pode criar um novo usuário com base nos dados recebidos na

requisição.

```
public function index()
{
    $usuarios = Usuario::all();

    if ($usuarios->isEmpty()) {
        return response()->json(['message' => 'Não temos usuários cadastrados'], 200)
    }

    return response()->json($usuarios, 200);
}
```