EXPRESS

O FRAMEWORK BACKEND

O QUE É NODE?

- **Definição**: Node.js é um ambiente de execução JavaScript no lado do servidor. Ele permite que você execute JavaScript fora do navegador.
- **Uso**: É amplamente utilizado para construir aplicações web, APIs e serviços backend.
- **Principais Características**:
  - Assíncrono e orientado a eventos: Isso significa que as operações de I/O não bloqueiam a execução do código.
  - Escalabilidade: Ideal para aplicações que precisam lidar com um grande número de conexões simultâneas.

SITE OFICIAL

O QUE É NPM?

- **Definição**: NPM (Node Package Manager) é o gerenciador de pacotes padrão para Node.js.
- **Função**: Permite instalar, atualizar e gerenciar bibliotecas (pacotes) que você pode usar em seus projetos Node.js.

SITE OFICIAL

O QUE É NVM?

- **Definição**: NVM (*Node Version Manager*) é uma ferramenta que permite gerenciar múltiplas versões do Node.js em sua máquina.
- **Uso**: É útil para alternar entre versões diferentes do Node.js conforme necessário para diferentes projetos.

SAIBA MAIS

BIBLIOTECA VS FRAMEWORK

BIBLIOTECAS

Uma biblioteca é um conjunto de funcionalidades específicas que você pode usar quando e onde quiser no seu código. Ela oferece ferramentas para resolver problemas pontuais e auxilia em tarefas específicas. No caso de uma biblioteca, como o jQuery, o desenvolvedor chama a biblioteca quando precisa de algo.

- Exemplo: Com o jQuery, você chama funções para manipular o DOM ou eventos. O código principal da aplicação ainda é seu; você apenas "pede ajuda" ao jQuery quando precisa.

FRAMEWORK

Um framework é uma estrutura completa que define e controla o fluxo da aplicação. Ele "chama" o seu código em determinados momentos. O framework estabelece as regras e o modelo que você deve seguir e, por isso, dita como a aplicação deve ser organizada e quais componentes usar.

- Exemplo: Em frameworks como Angular, o código é estruturado dentro de componentes ou módulos definidos pelo framework, que também determina a sequência de execução e como os elementos interagem entre si.

ARQUITETURA EM CAMADAS (LAYERED ARCHITECTURE)

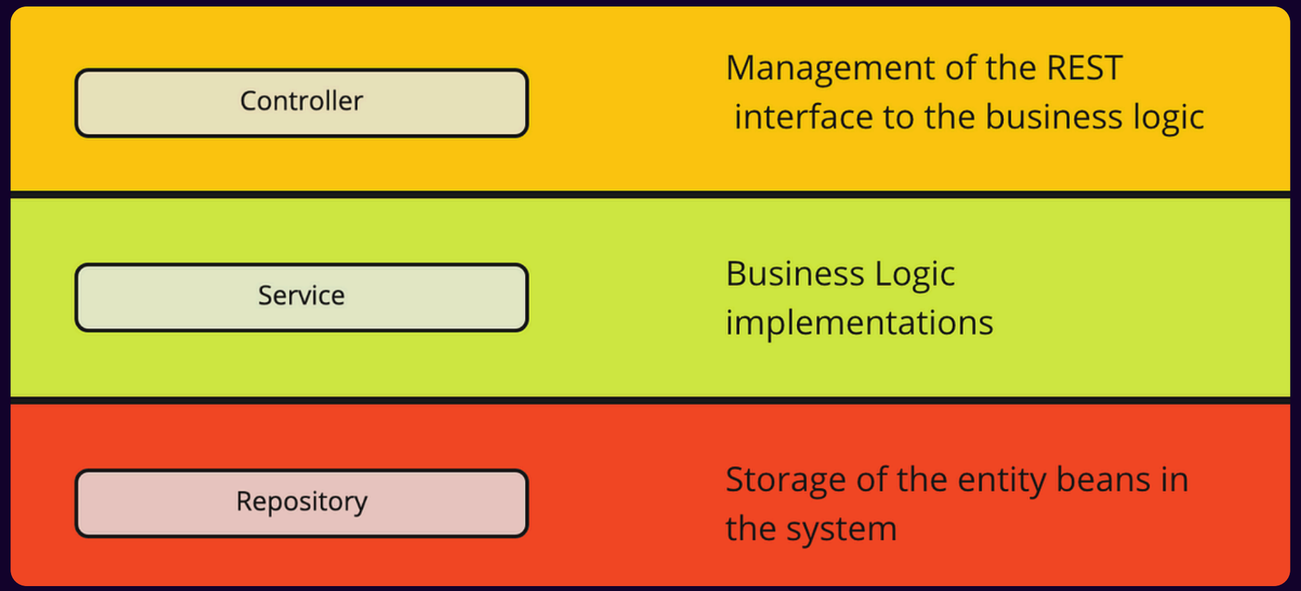
É uma das mais usadas para APIs REST, pois separa bem as responsabilidades, facilitando a manutenção e escalabilidade.

Como funciona a Arquitetura em Camadas?
Ela organiza a aplicação em três camadas principais:
1️⃣ **Controller** → Recebe requisições e retorna respostas.
2️⃣ **Service** (ou Use Case) → Contém a lógica de negócio.
3️⃣ **Repository** (ou Data Access Layer) → Se comunica com o banco de dados.

ARQUITETURA EM CAMADAS (LAYERED ARCHITECTURE)

Como essas camadas interagem?

- O Controller não faz lógica pesada, ele só chama o Service.
- O Service executa as regras de negócio e chama o Repository.
- O Repository acessa o banco de dados.



Vantagens dessa arquitetura

Código mais organizado e modular
Facilita testes unitários (podemos testar cada camada separadamente)
Torna a aplicação mais escalável e fácil de manter
Respeita SOLID (especialmente o princípio da responsabilidade única - SRP)

Essa arquitetura tem outro nome?

Além de Arquitetura em Camadas, ela também pode ser chamada de:

- *Pattern Controller-Service-Repository* (quando explicamos pelo nome das camadas)
- *Arquitetura Hexagonal* (Ports and Adapters) → Uma versão mais avançada, desacoplando mais as dependências
- *Clean Architecture* (Robert C. Martin - Uncle Bob) → Parecida, mas adiciona mais camadas como Application e Domain

O QUE É O EXPRESS?

O QUE É EXPRESS.JS?

O **Express.js** é um framework minimalista para Node.js que facilita a criação de servidores web e APIs. Ele abstrai funcionalidades complexas do Node.js, permitindo que os desenvolvedores criem aplicações de forma rápida e organizada.

POR QUE USAR O EXPRESS.JS?

- **Simplicidade**: Código limpo e intuitivo.
- **Rapidez**: Facilita a criação de servidores e rotas.
- **Flexibilidade**: Permite integração com diversos bancos de dados e bibliotecas.
- **Ampla comunidade**: Possui suporte de uma grande comunidade e documentação extensa.

COMO FUNCIONA O EXPRESS.JS?

O Express funciona como um middleware que gerencia requisições HTTP e respostas. Ele permite definir rotas, lidar com múltiplos tipos de requisição (GET, POST, PUT, DELETE) e integrar-se com outras bibliotecas para funcionalidades adicionais.

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

// Definindo uma rota
app.get("/", (req, res) => {
  res.send("Olá, mundo!");
});

// Iniciando o servidor
app.listen(3000, () => {
  console.log("Servidor rodando em http://localhost: 3000");
});
```

RECURSOS PRINCIPAIS DO EXPRESS.JS

1. **Roteamento**: Permite definir caminhos para acessar diferentes partes da aplicação.
2. **Middlewares**: Funções que manipulam requisições antes de chegar à rota final.
3. **Integração com Banco de Dados**: Facilita conexões com MongoDB, MySQL, PostgreSQL, etc.
4. **Gerenciamento de Sessão e Autenticação**: Suporte para autenticação de usuários via JWT ou cookies.
5. **Suporte a APIs REST**: Criação de APIs de forma estruturada e eficiente.

CONCLUSÃO

O Express.js é uma ferramenta **essencial** para desenvolvedores Node.js, pois permite criar servidores e APIs de forma **simples**, **eficiente** e **escalável**. Seu uso é amplamente recomendado para projetos que necessitam de **desempenho** e flexibilidade.

THANK YOU

@wallace027dev