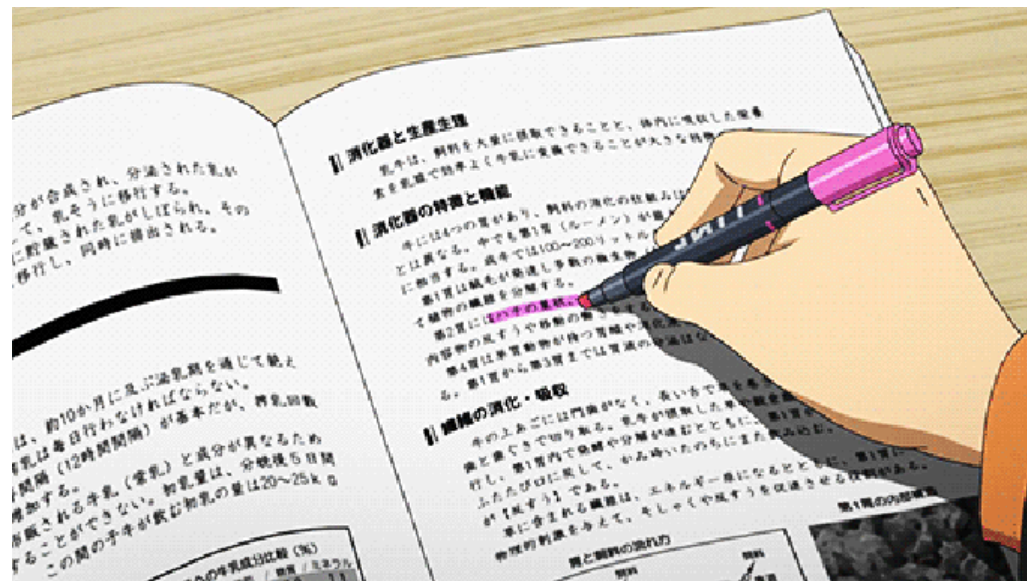




# Auto estudo...

## Até 10h





# DAILY

## A FAZER



## DESENVOLVIMENTO - 5



## TESTE - 3



## CONCLUÍDO



Até 10h15



inteli



## INTRODUÇÃO À COMPUTAÇÃO

# Back-end I - Node.js, models e controllers

Professor Dr. Cristiano Benites





# Agenda do dia



Explicação Teórica



Mão na massa





# QUEM SOU EU...

Dr. CRISTIANO  
BENITES



*Deficiente auditivo \*Grau profundo*







**\*CID H91.8**

*Especialista em leitura labial*

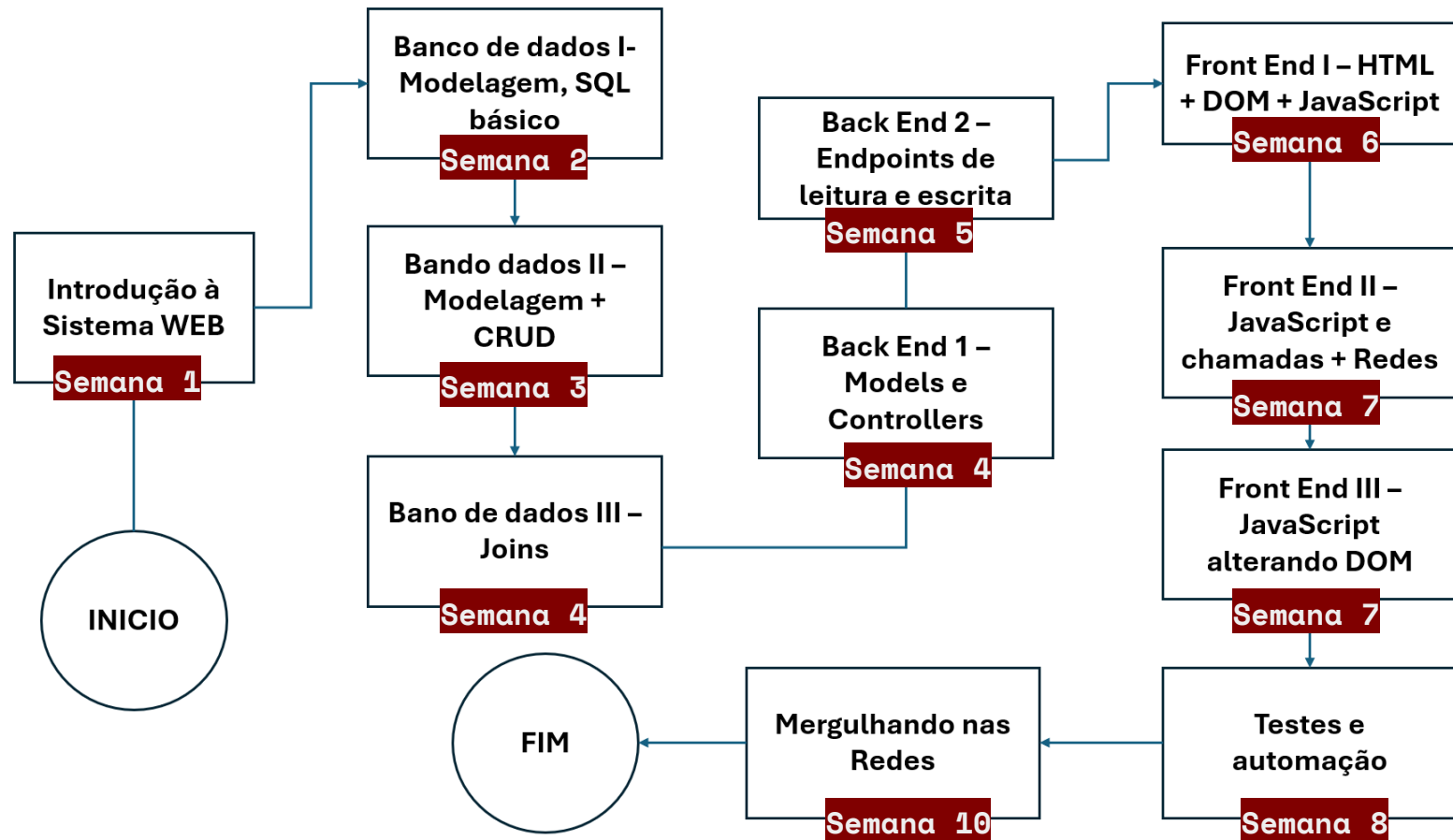




# Combinados

-  Você é responsável pela seu Check in na Adalove
-  Presença na aula das 10h até 12h. "Ausência acarretará em falta"
-  Atividades entregues no prazo
-  Estou sempre disponível no SLACK ou em sala de aula
-  Se planejem sempre de forma antecipada
-  Avaliem nosso encontro na Adalove. Vamos praticar feedback o tempo todo








# Conteúdo do dia



 Back-end I – Node.js, models e controllers


Semana 04

 O que é Node JS?


Semana 04

 Javascript: Fetch DELETE and PUT requests

Semana 04

 Use JavaScript to execute an API DELETE request

Semana 04

 Como sair do zero em Node.js em apenas uma aula

Semana 04

 Projeto Individual (Parte 2) – Conectando Banco de Dados e Servidor + Arquitetura MVC

Semana 04





Node.js devalbont  
development on  
environment

```

Tatabesczf{eo}:
  fit +|unge-i|eetrit(--clee)
}fabvecleraiert
  stungctia|ctc1p
  p1anyoe-ticnt
  sconeot||cotuscu{||
    arosf2fd| aaiogrliit
    l iurcogeri_lts{
    steaasefuse|}}
    e uosice|ctuae|e|>
    ohml11|. etä_||asseern>
    onesttt..naboeucot?
    neofidizirevtrn>liianeta>
    oacttle.dztiinoi..cotes>
    fzzi)
    oshth|zsutuect-)
    oetsaconeste>
  c(eurint-)
  tzt
  ly zta- Yono
  o yome=urhere-Ulrigno-
  9 Acogcãonuit>
    02
    Tocamnon/>
    e._oul..scçwiscilngc>
    0.130
    llos=0--debugcãutst>
  Ditent.
  01..cõnestoconec>
  0..la it--Tãgestajje>
  
```

# Back-end I - Node.js, Models e Controllers

Aplicações back-end com Node.js, implementando modelos e controladores para criar APIs RESTful completas.

Requisições DELETE e PUT usando JavaScript.

# O Que é Node.js?



## JavaScript no Servidor

Executa código JS fora do navegador



## Motor V8

Mesma engine do Google Chrome



## NPM

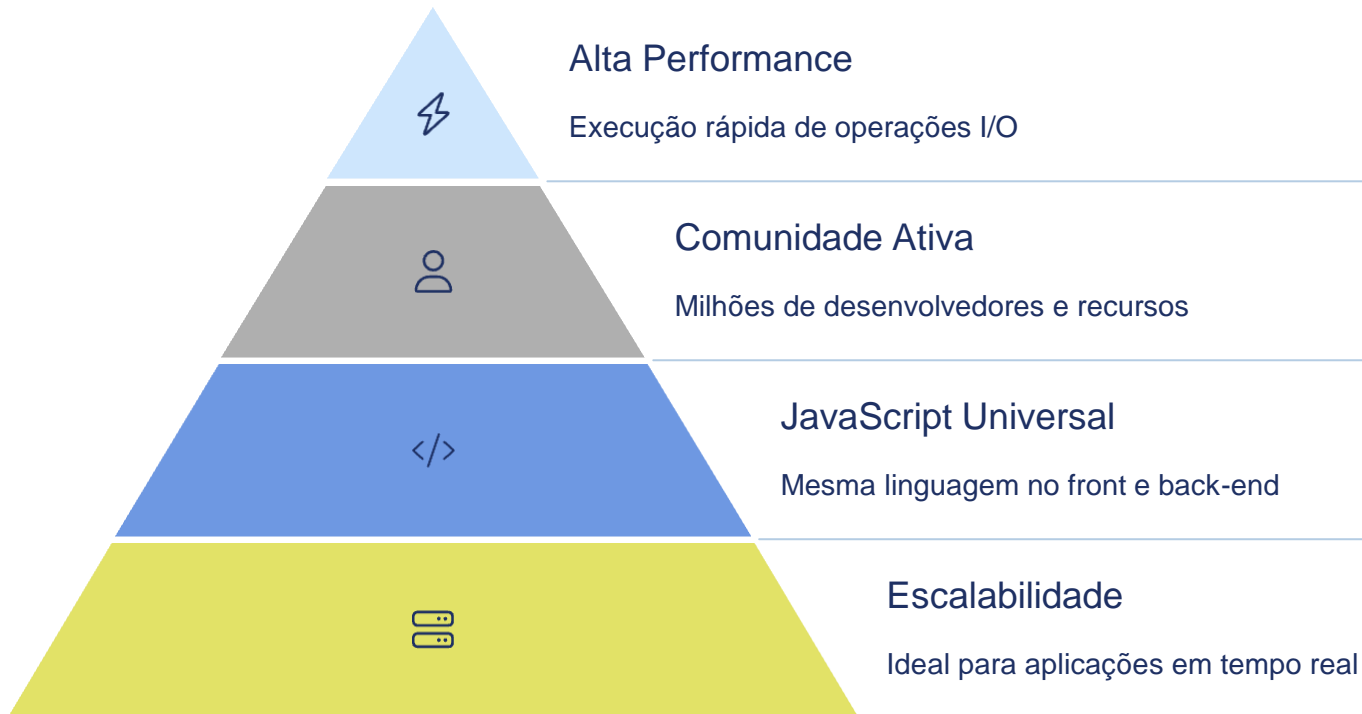
+1 milhão de pacotes prontos para uso



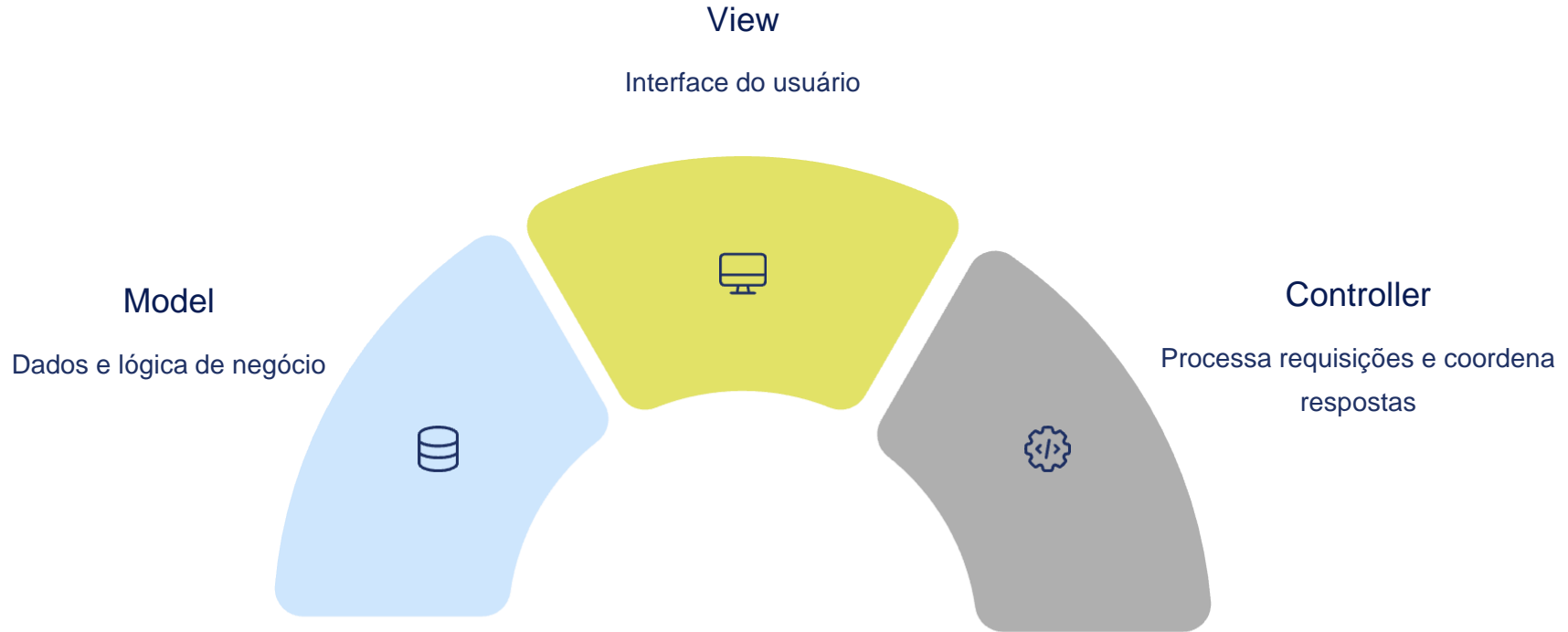
## Assíncrono

I/O não bloqueante para alta performance

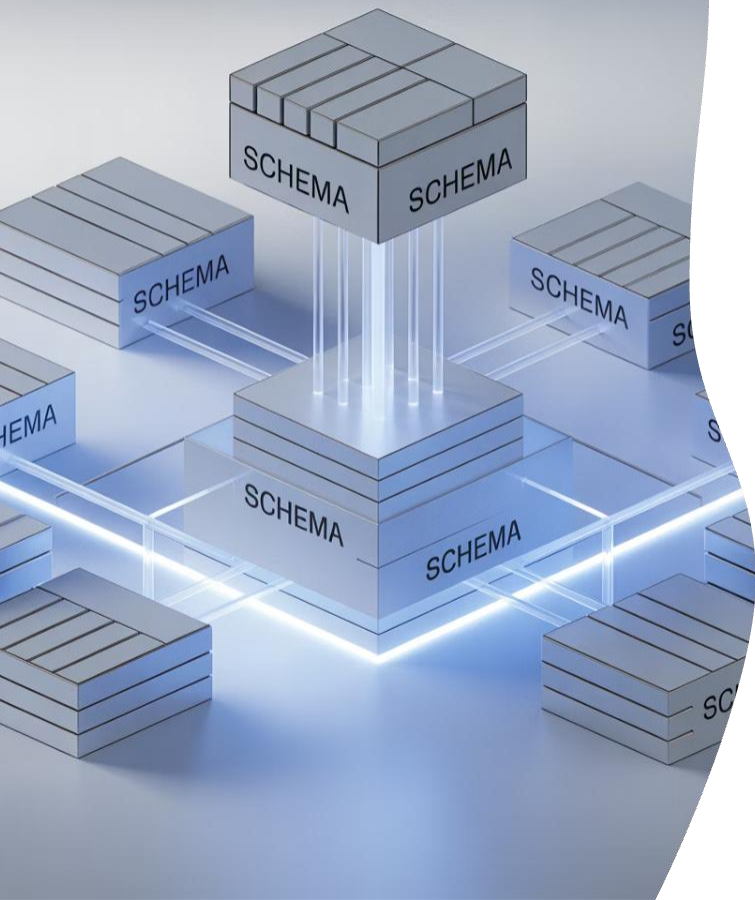
# Vantagens do Node.js



# Arquitetura MVC no Node.js



# Models no Node.js



## Representação de Entidades

Definem estrutura dos dados no sistema



## Lógica de Negócio

Encapsulam regras específicas



## Validação de Dados

Garantem integridade das informações



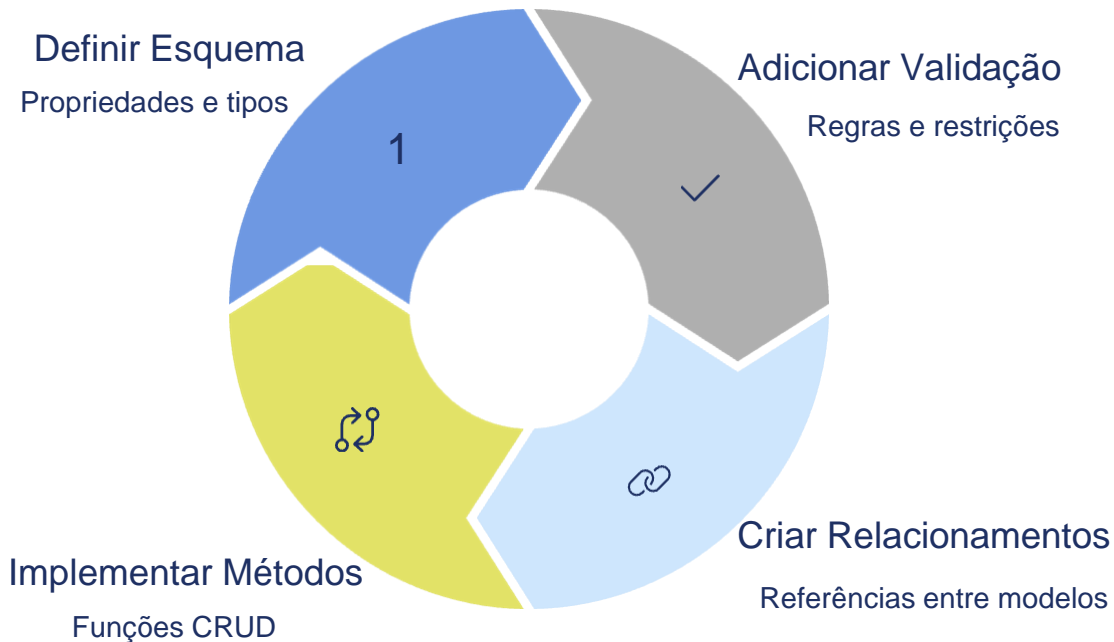
## Operações CRUD

Gerenciam acesso ao banco de dados

NodeJCpewdiner CHDMr  
SchemaFlow

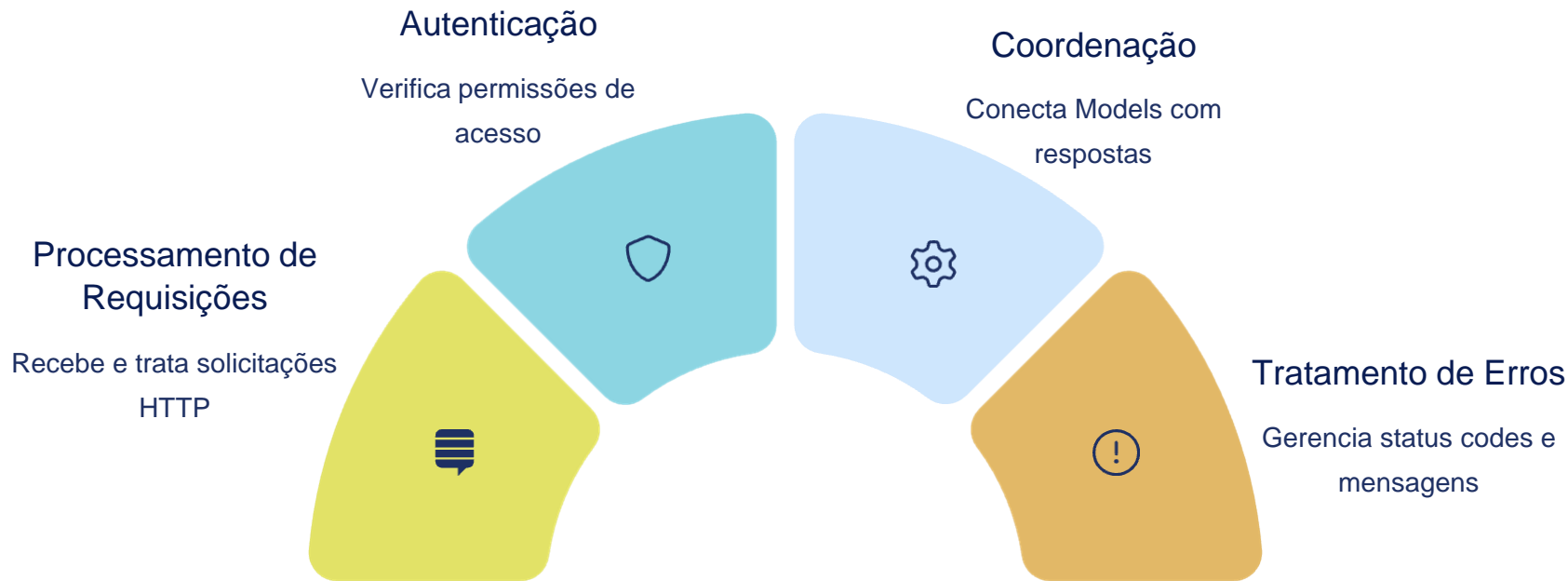
Try it free

# Exemplo Prático: Criando um Model





# Controllers no Node.js



# Exemplo Prático: Criando um Controller

1

## Estrutura Básica

Crie um arquivo usuarioController.js e importe o model

2

## Método de Busca

Implemente função para listar todos os usuários

3

## Método de Criação

Adicione validação e persistência de novos dados

4

## Métodos PUT/DELETE

Implemente funções para atualizar e remover registros

hboard

Dashboard

API Docs

Tutorials

Pitfalls



## HTTP Response

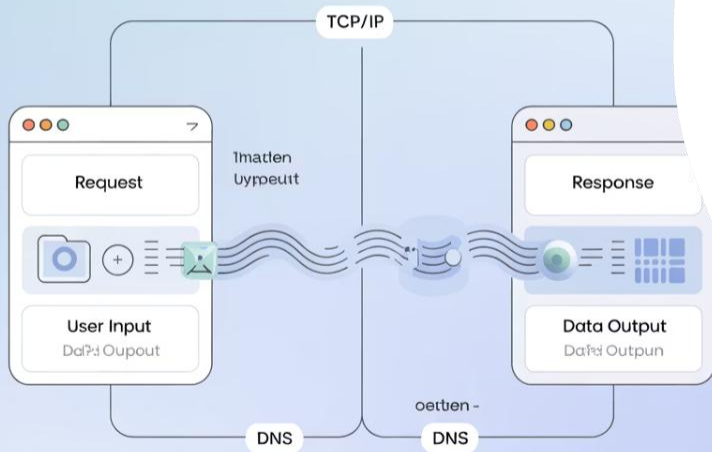
Overview

Protocols

Security

Security

FAQ



# JavaScript: Requisições HTTP

# 4

Métodos Principais

GET, POST, PUT, DELETE

# 200

Sucesso

Código para resposta bem-sucedida

# 404

Não Encontrado

Recurso inexistente

# 500

Erro Servidor

Falha interna na aplicação

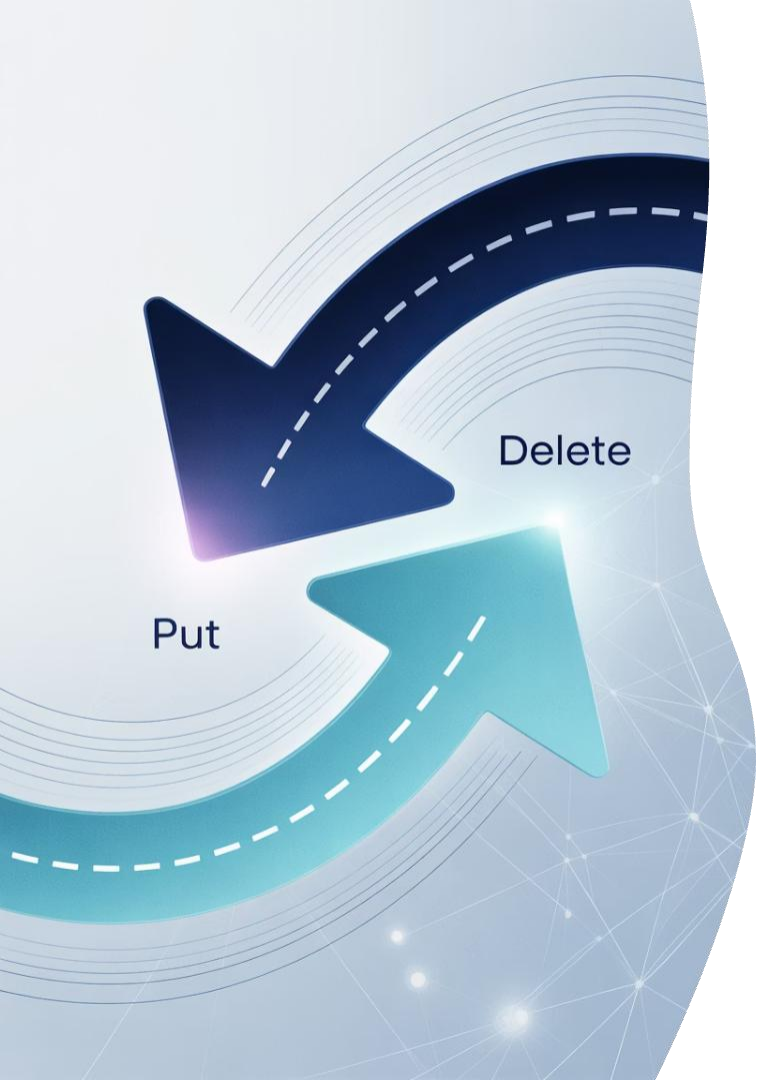
## Métodos HTTP

- GET: busca dados
- POST: cria recursos
- PUT: atualiza recursos
- DELETE: remove recursos

## APIs JavaScript

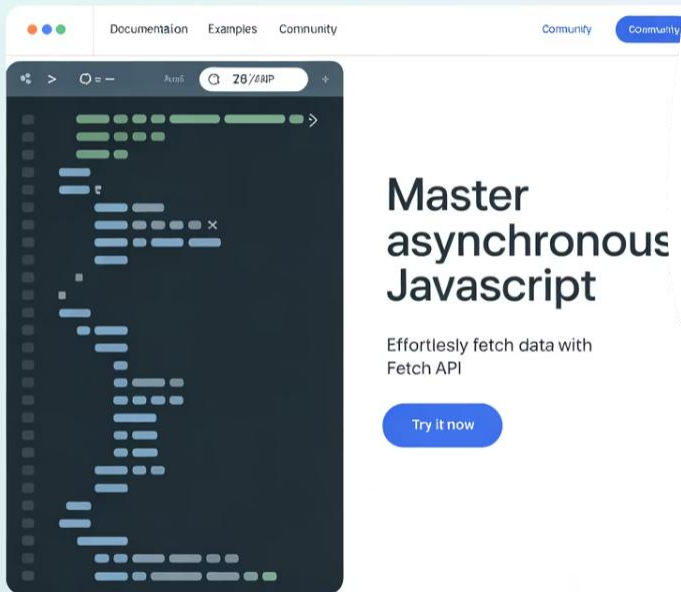
- Fetch: nativa moderna
- Axios: biblioteca popular
- jQuery AJAX: compatibilidade

# Entendendo Requisições PUT e DELETE



Método	Função	Característica	Status
PUT	Atualiza recursos	Idempotente	200, 204
DELETE	Remove recursos	Irreversível	200, 204
Segurança	Autenticação	Tokens JWT	403, 401

# Fetch API: Visão Geral



## Enviar Requisição

Configurar método, cabeçalhos e corpo



## Aguardar Promise

Operação assíncrona com `.then()` ou `await`



## Processar Resposta

Converter para JSON ou outro formato



## Tratar Erros

Capturar falhas com `.catch()` ou `try/catch`

BORA FAZER UMA  
ATIVIDADE PARA  
FIXAR OS CONCEITOS!

MÃO NA  
MASSA





[https://github.com/cristianobenites/Materiais\\_Aulas/blob/main/Back\\_End/Modulo2/aula\\_4/README.md](https://github.com/cristianobenites/Materiais_Aulas/blob/main/Back_End/Modulo2/aula_4/README.md)



# Avalie nossa aula de hoje na Adalove!



Aluno



Professor

MUITO OBRIGADO!



# Formações Acadêmicas

## Graduações

*Enfermagem*

*Ciência da Computação*

*Tecnologia em Redes de Computadores*

*Programação de Computadores*

*Sistema de Informação*

## Pós-Graduações

*Mestre em Engenharia Elétrica e da Computação*

*Doutor em Engenharia Elétrica e da Computação*

*MBA Em Data Center e Computação em Nuvem*

*MBA em Gestão da Tecnologia da Informação*

*MBA Engenharia de Software*



## Certificações

*DELL EMC PROFISSIONAL – Cloud Infrastructure and Services;*

*DELL PROVEN PROFISSIONAL – Information Storage and Management.*

*ISO 27001*

*MCSA Windows SERVER - 410, 411 e 412;*

*Linux – Lpic 1 e Lpic 2;*

*Microsoft Certified Professional;*

*Symantec Backup Exec;*

*Symantec Endpoint Protection,*

*Cobit;*

*ITIL;*

Professor Dr. Cristiano Benites



OBRIGADO  
**E LET'S BORA!**