

Introdução ao Backend

Desvendando o Backend

A força invisível que sustenta as aplicações digitais

Back End
DEVELOPMENT



Processamento

Lógica de negócios e processamento de dados



Armazenamento

Gerenciamento e persistência de dados



Segurança

Proteção de dados e autenticação

O backend é responsável por tudo o que acontece nos bastidores das aplicações web e móveis. Enquanto o frontend é o que o usuário vê e interage, o backend processa dados, gerencia bancos de dados, implementa regras de negócio e garante a segurança das informações.

Fundamentos Inegociáveis

Antes de mergulhar nas linguagens e ferramentas, é essencial compreender os conceitos fundamentais que sustentam todo o desenvolvimento backend.

Comunicação Cliente-Servidor

O modelo onde o navegador (cliente) faz solicitações ao servidor, que processa e retorna respostas. É a base de toda a web.

Protocolos HTTP/HTTPS

Regras que definem como as informações são transmitidas na web. HTTPS adiciona uma camada de segurança com criptografia.

Lógica de Programação

A arte de instruir o computador a realizar tarefas de forma eficiente, usando algoritmos, estruturas condicionais e laços de repetição.



Algoritmos

Sequências lógicas de passos para resolver problemas



Estruturas de Dados

Listas, filas, pilhas e árvores para organizar informações



Front End

- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation



Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

Infraestrutura Web

DNS e Domínios



Sistema que traduz nomes de domínio (como google.com) para endereços IP numéricos



Servidores

Computadores potentes que armazenam arquivos e código das aplicações



Hospedagem

Serviços que disponibilizam espaço em servidores para suas aplicações

Arquiteturas do Backend

As arquiteturas de backend definem como os sistemas são estruturados, escalados e mantidos. Cada abordagem tem suas vantagens e desafios específicos.

Monolítica

Todos os componentes da aplicação estão integrados em uma única base de código.

Vantagens

- Simplicidade no desenvolvimento
- Facilidade na implantação

Desvantagens

- Difícil de escalar partes específicas
- Maior risco de falhas

Microserviços

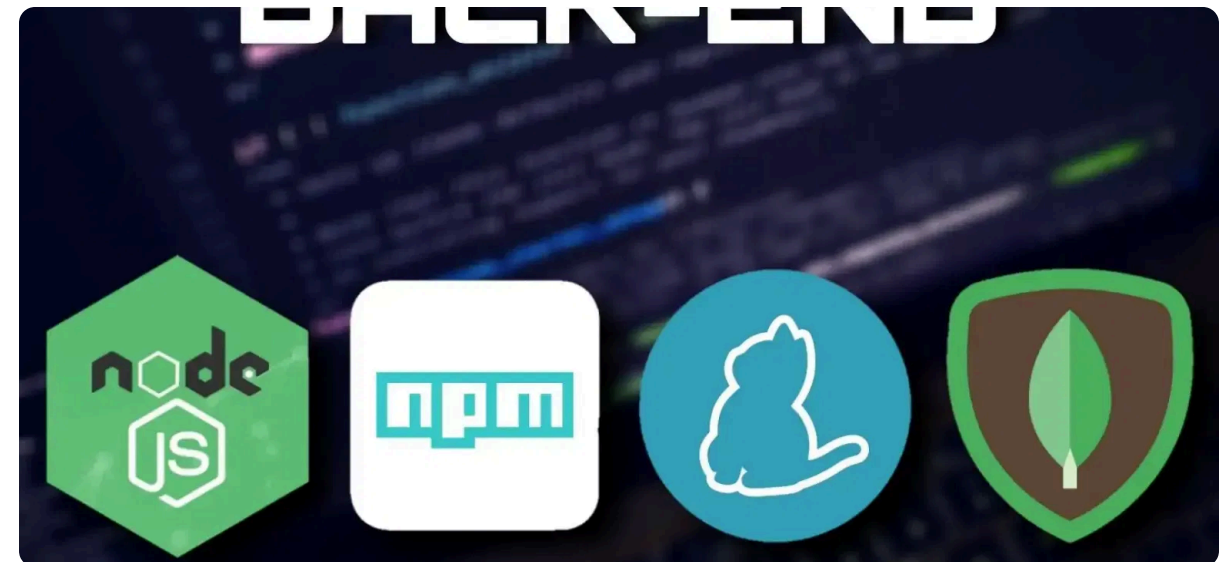
Divide a aplicação em pequenos serviços independentes, cada um com função específica.

Vantagens

- Escalabilidade individual
- Desenvolvimento independente

Desvantagens

- Complexidade de gerenciamento
- Requer habilidades em DevOps



Serverless

Permite executar funções sem se preocupar com a infraestrutura subjacente.

Vantagens

- Escalabilidade automática
- Custos reduzidos (pague pelo uso)

Desvantagens

- Dependência do provedor de nuvem
- Cold starts em funções pouco usadas

Orientada a Eventos

Os componentes se comunicam por meio de eventos, permitindo maior desacoplamento.

Vantagens

- Alta reatividade e escalabilidade
- Ideal para processamento em tempo real

Desvantagens

- Complexidade de rastreamento
- Desafios de consistência de dados

Linguagens de Programação

A escolha da linguagem de programação é um passo crucial no desenvolvimento backend. Cada linguagem tem características específicas que a tornam mais adequada para determinados tipos de projetos.




Python

Sintaxe clara e legível, ideal para iniciantes. Frameworks robustos como Django e Flask.

Web

IA

Análise de Dados



JavaScript (Node.js)

Permite desenvolvimento full-stack com uma única linguagem. Excelente para aplicações em tempo real.

APIs

Tempo Real

Full-Stack



Java

Robusta e escalável, amplamente usada em empresas. Framework Spring Boot para desenvolvimento rápido.

Empresarial

Alta Demanda

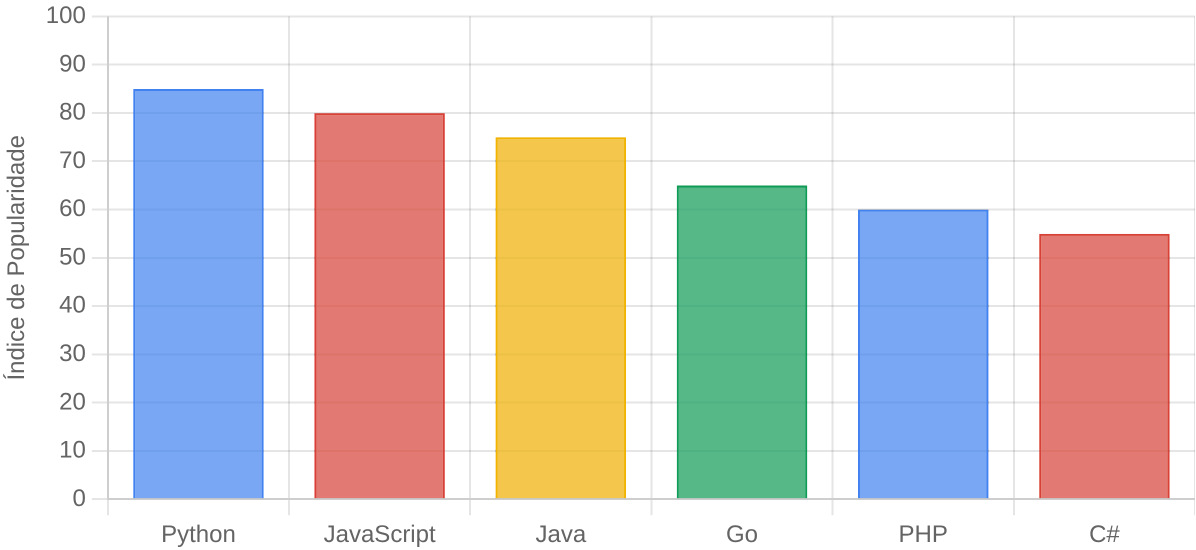
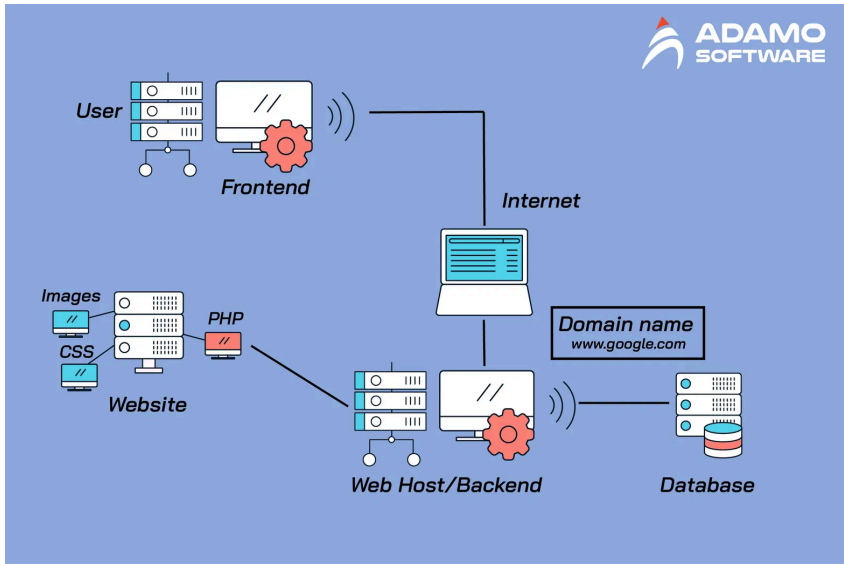


Go (Golang)

Desenvolvida pelo Google, foca em performance e concorrência. Ideal para microsserviços.

Performance

Microsserviços



Outras linguagens populares incluem PHP (desenvolvimento web e CMS), Ruby (framework Ruby on Rails) e C# (ecossistema Microsoft .NET).

A escolha da linguagem deve considerar fatores como tipo de projeto, ecossistema, mercado de trabalho e curva de aprendizado.

APIs: A Ponte entre Sistemas

As APIs (Application Programming Interfaces) são como garçons em um restaurante: recebem pedidos de um sistema (cliente), os entregam a outro sistema (servidor), que processa e devolve a resposta.

↔ RESTful APIs

O padrão mais utilizado para comunicação entre sistemas na web, baseado nos métodos HTTP.

GET

/posts (Listar posts)

GET

/posts/123 (Obter post)

POST

/posts (Criar post)

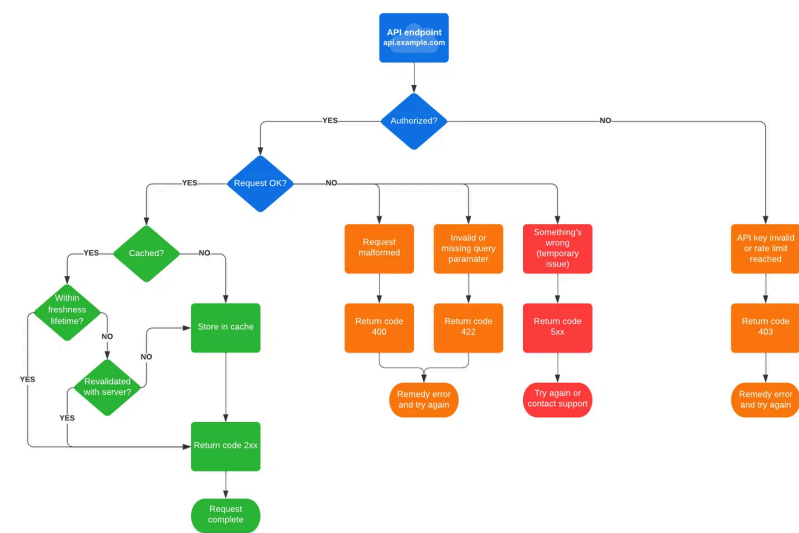
PUT

/posts/123 (Atualizar)

DELETE

/posts/123 (Excluir)

- ✔ **Simplicidade e Compatibilidade**
APIs RESTful são leves e compatíveis com praticamente todas as linguagens.
- ✔ **Stateless (Sem Estado)**
Cada requisição contém todas as informações necessárias.



🔗 GraphQL

Uma alternativa mais recente ao REST que permite aos clientes solicitar exatamente os dados que precisam.

```
query {
  post(id: "123") {
    title
    content
    author { name, email }
  }
}
```

- + **Vantagens**
Reduz requisições, melhora performance e facilita evolução da API.
- **Desafios**
Curva de aprendizado mais íngreme e configuração mais complexa.

Bancos de Dados

Os bancos de dados são o coração de qualquer aplicação backend, responsáveis por armazenar, organizar e recuperar dados de forma eficiente.

Bancos de Dados Relacionais (SQL)

Organizam dados em tabelas com linhas e colunas, com relações definidas por chaves. Ideais para dados estruturados.

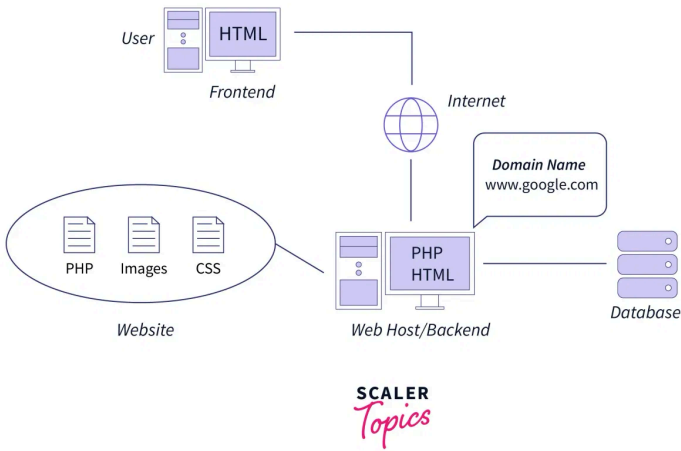
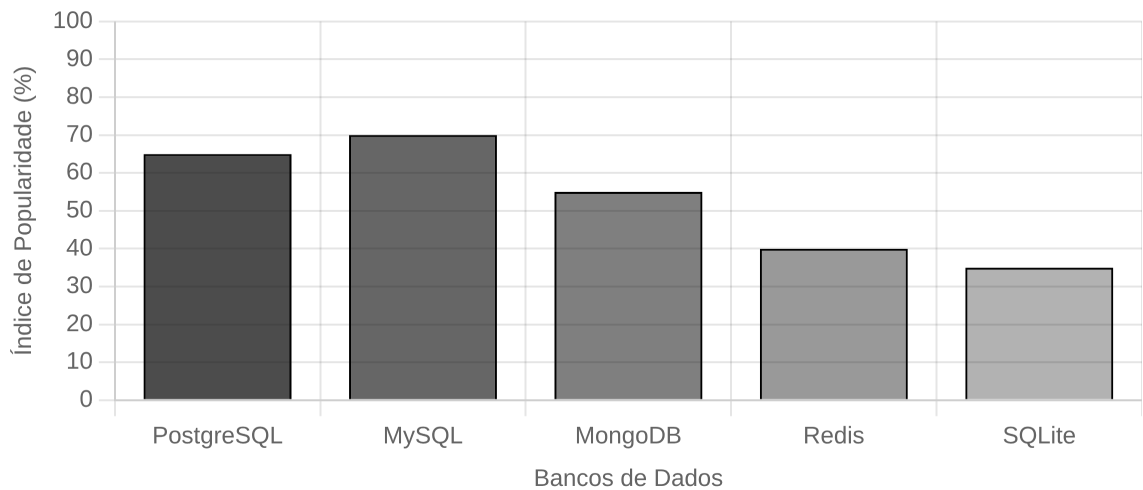
- ✓ PostgreSQL

✓ MySQL
- ✓ SQL Server

✓ Oracle

Vantagens:

- Consistência e integridade dos dados
- Consultas complexas com SQL
- Transações ACID



Bancos de Dados NoSQL

Mais flexíveis e escaláveis, ideais para dados não estruturados e aplicações que exigem alta disponibilidade.

- ✓ Documento: MongoDB

✓ Chave-Valor: Redis
- ✓ Coluna: Cassandra

✓ Grafo: Neo4j

Vantagens:

- Escalabilidade horizontal facilitada
- Flexibilidade para diferentes tipos de dados
- Alta performance para grandes volumes

Escolhendo o Banco de Dados Ideal





A escolha entre SQL e NoSQL depende das necessidades específicas do seu projeto:

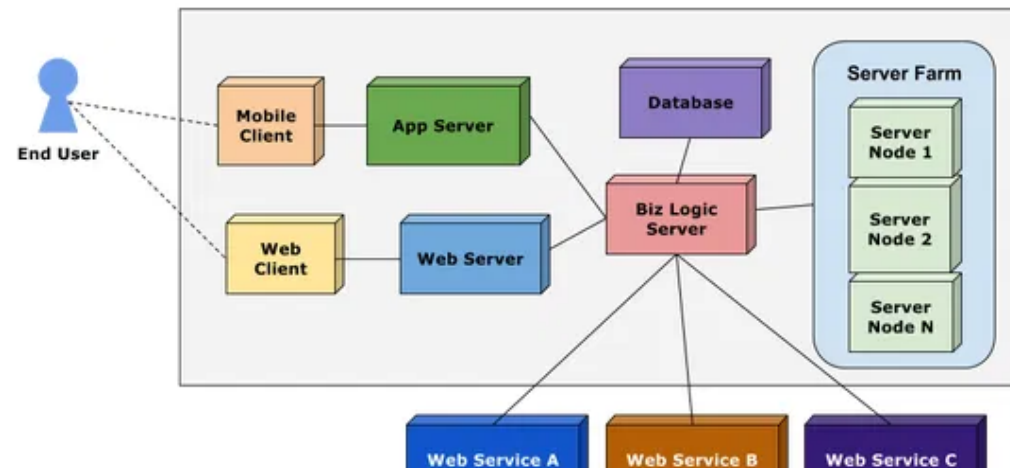
- Use SQL para dados estruturados e relacionamentos complexos
- Use NoSQL para escalabilidade e flexibilidade
- Considere abordagens híbridas quando necessário

Otimização e Segurança

A performance e a segurança são aspectos críticos no desenvolvimento backend. Sistemas otimizados proporcionam melhor experiência ao usuário, enquanto práticas de segurança protegem dados sensíveis.

Otimização de Performance

-  **Gerenciamento de Estado**
Armazena informações sobre a interação do usuário entre requisições.
-  **Cache**
Armazena dados frequentemente acessados para acesso rápido.
-  **Otimização de Consultas**
Melhora a eficiência das consultas com índices otimizados.
-  **Balanceamento de Carga**
Distribui o tráfego entre múltiplos servidores.



Segurança no Backend

Injeção de SQL

Ataques que exploram vulnerabilidades em consultas SQL. **Proteção:** Use consultas parametrizadas e ORM.

Cross-Site Scripting (XSS)

Injeção de scripts maliciosos em páginas web. **Proteção:** Valide e sanitize todas as entradas de usuário.

Autenticação e Autorização

Falhas que permitem acesso não autorizado. **Proteção:** Autenticação de dois fatores, tokens JWT e controle de acesso.

Ferramentas e Próximos Passos

Para se tornar um desenvolvedor backend eficiente, é essencial dominar um conjunto de ferramentas que facilitam o desenvolvimento, teste e implantação de aplicações.

🔑 Controle de Versão

🔑 Git

🔑 GitHub

🔑 GitLab

💻 Ambiente de Desenvolvimento

</> VS Code

</> IntelliJ IDEA

🚢 Docker

🔧 Testes

Jest

Pytest

JUnit

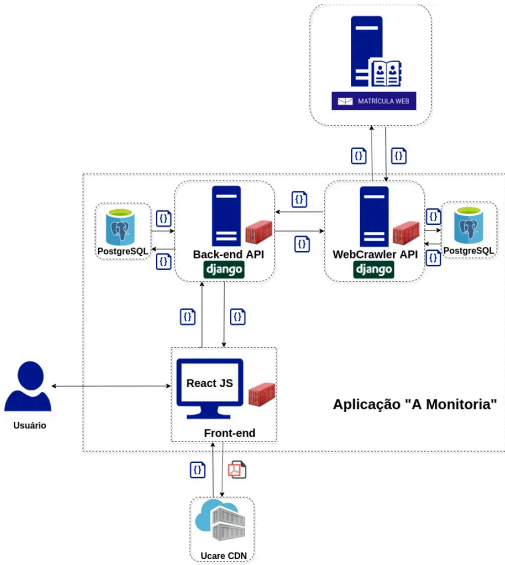
Postman

📈 Monitoramento

Prometheus

Grafana

ELK Stack



Recursos para Aprendizado

- 📖 Plataformas de Ensino**
Rocketseat, Alura, Udemy, Coursera, freeCodeCamp
- 📄 Documentações**
MDN Web Docs, documentações oficiais das linguagens
- 👥 Comunidades**
Stack Overflow, GitHub, Dev.to, fóruns específicos

Próximos Passos para Evoluir

- 1

Construa Projetos Práticos
Desenvolva aplicações reais para aplicar conhecimentos.
- 2

Contribua com Open Source
Participe de projetos de código aberto.
- 3

Aprenda DevOps
Estude CI/CD e containerização.
- 4

Mantenha-se Atualizado
Acompanhe as tendências do mercado.