Introdução à Engenharia de Software e Desenvolvimento Back-End com Node.js

Autenticação com JWT, Padrão MVC e Banco de Dados Relacional

Objetivos da Aula

- Visualizar o plano de ensino da UC
- Compreender fundamentos da engenharia de software
- Diferenciar requisitos funcionais e não funcionais
- Introduzir metodologias ágeis e melhoria contínua
- Apresentar PostgreSQL e modelagem de dados
- Entender arquitetura MVC
- Compreender autenticação via JWT

O que é Engenharia de Software?

> "Aplicação sistemática de princípios para desenvolver software confiável e eficiente."

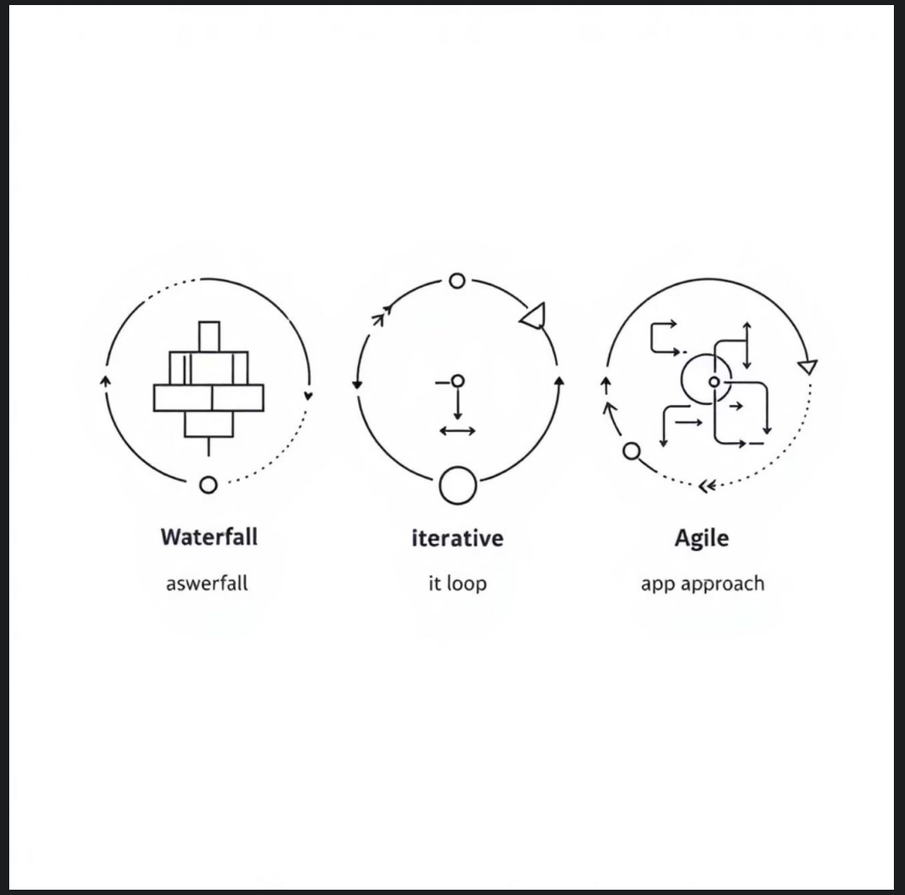
📘 Referência: Sommerville

Importância:

- Reduz falhas
- Traz previsibilidade
- Facilita manutenção

🧠 **Pergunta para reflexão:** Você já participou de algum projeto? Como ele foi organizado?

Ciclo de Vida de Software



- Cascata (linear, sequencial)
- Iterativo
- Ágil (incremental, adaptativo)

Qualidade de Software (ISO/IEC 25010)

Atributos essenciais para garantir a excelência no desenvolvimento de software

Funcionalidade

O software atende às necessidades declaradas e implícitas dos usuários

Usabilidade

Facilidade de uso e aprendizado do sistema pelos usuários

Eficiência

Desempenho em relação aos recursos utilizados

Segurança

Proteção de informações e dados contra acesso não autorizado

Manutenibilidade

Facilidade de modificação e evolução do software

Requisitos Funcionais vs Não Funcionais

1

Requisitos Funcionais (RF)

Descrevem o que o sistema deve fazer

Exemplo: "Usuário pode fazer login"

2

Requisitos Não Funcionais (RNF)

Descrevem como o sistema deve se comportar

Exemplo: "A resposta deve ocorrer em até 2s"

🧠 **Atividade rápida:** Crie RF/RNF para um aplicativo de tarefas

Metodologias Ágeis e Scrum

Manifesto Ágil

- Indivíduos e interações > processos e ferramentas
- Software funcionando > documentação abrangente
- Colaboração com o cliente > negociação de contratos
- Responder a mudanças > seguir um plano fixo

Fluxo Scrum

Entrega e Melhoria Contínua

Desenvolvimento

Escrita de código e testes unitários

Integração Contínua

Commits frequentes e testes automáticos

Entrega Contínua

Deploys frequentes em produção

Testes

Verificação de qualidade e funcionalidade

Ferramentas

- Git + GitHub Actions
- Jenkins
- Docker
- Railway/Render

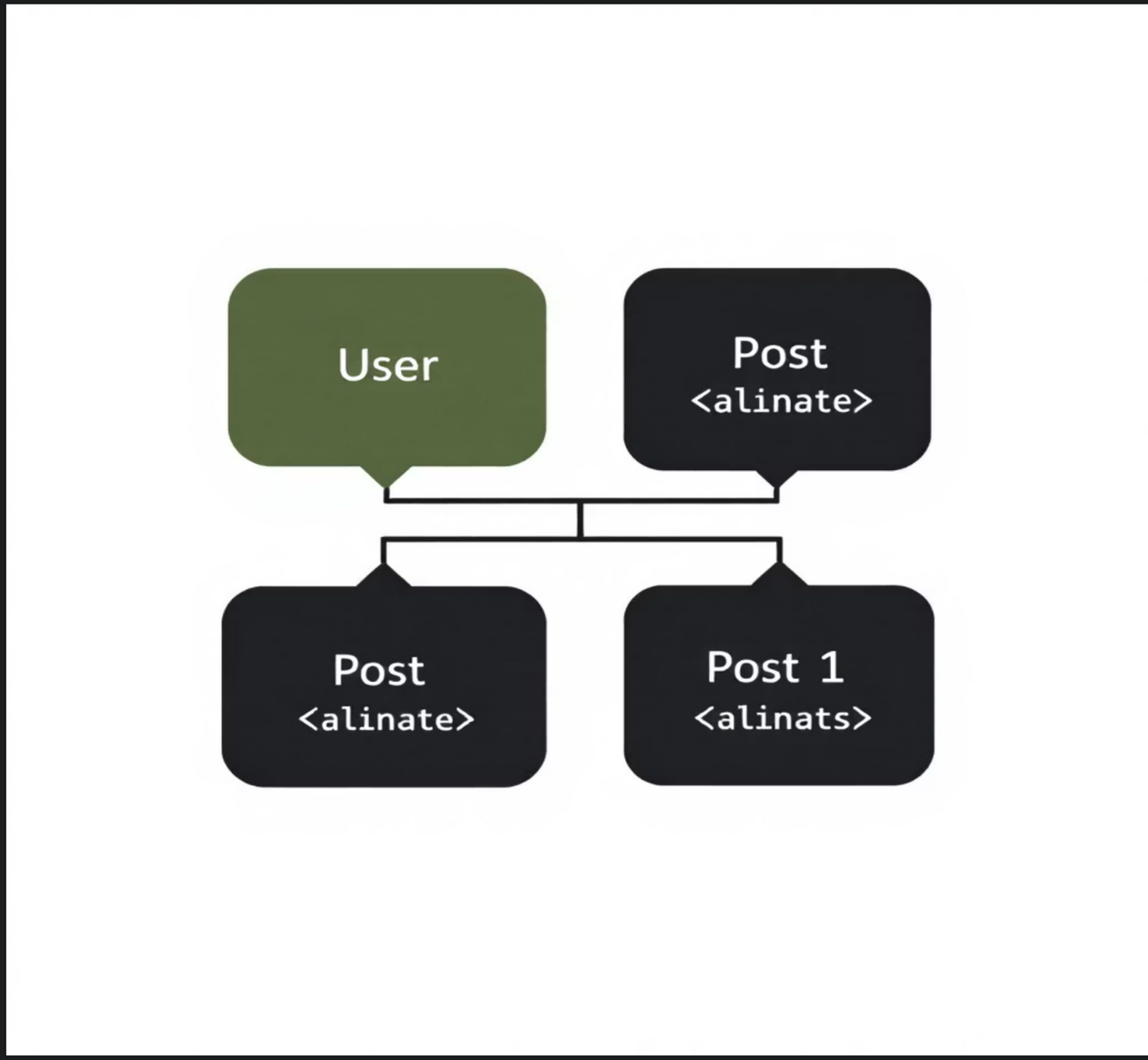
Banco de Dados Relacional (PostgreSQL)

Conceitos Fundamentais

- Entidades e Relacionamentos
- Chaves Primárias e Estrangeiras
- Tabelas normalizadas

📘 Ferramenta para diagrama: https://dbdiagram.io

Exemplo de Modelo ER



Arquitetura MVC

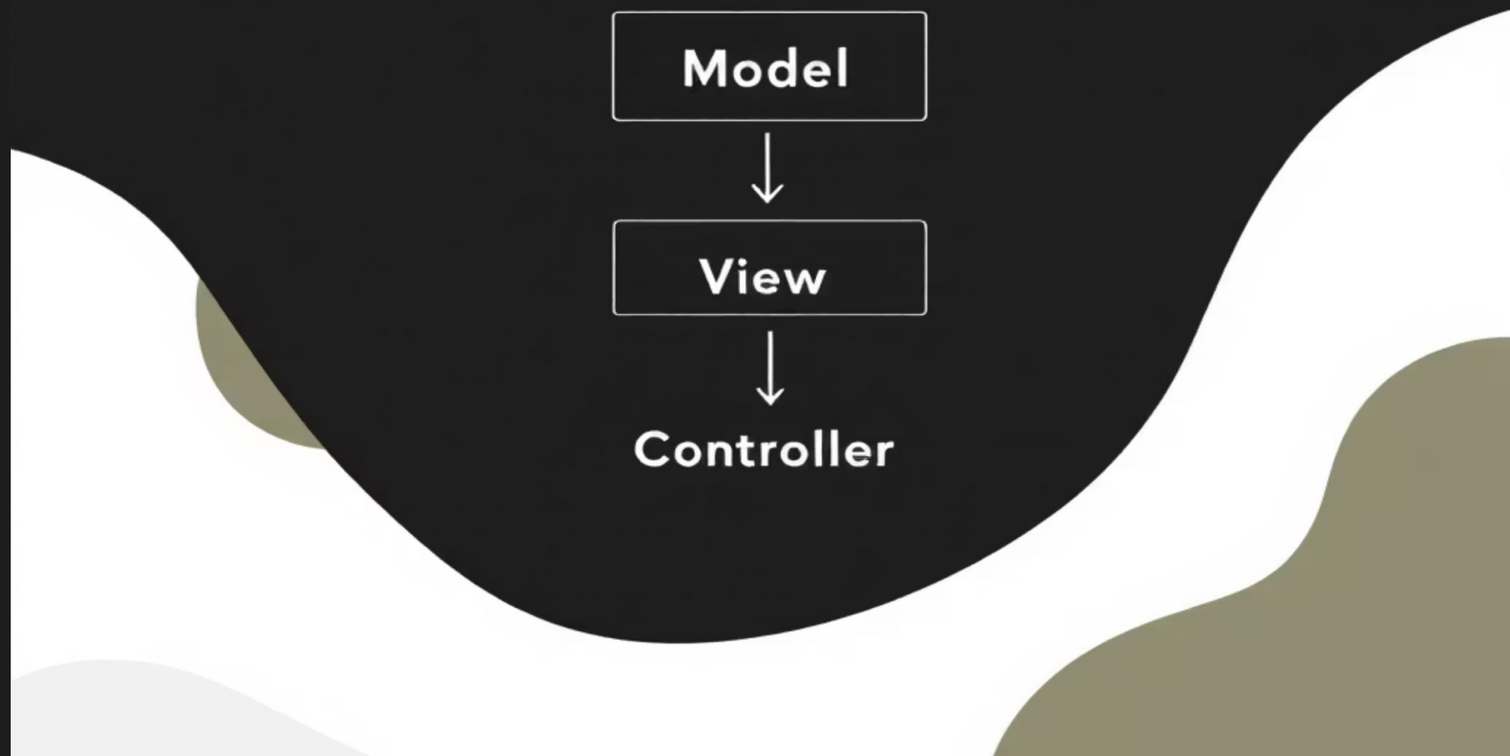
Estrutura típica

```
/controllers/userController.js/models/User.js/routes/userRoutes
.js
```

Benefícios

- Separação de responsabilidades
- Código modular e testável
- Facilita manutenção

Fluxo MVC



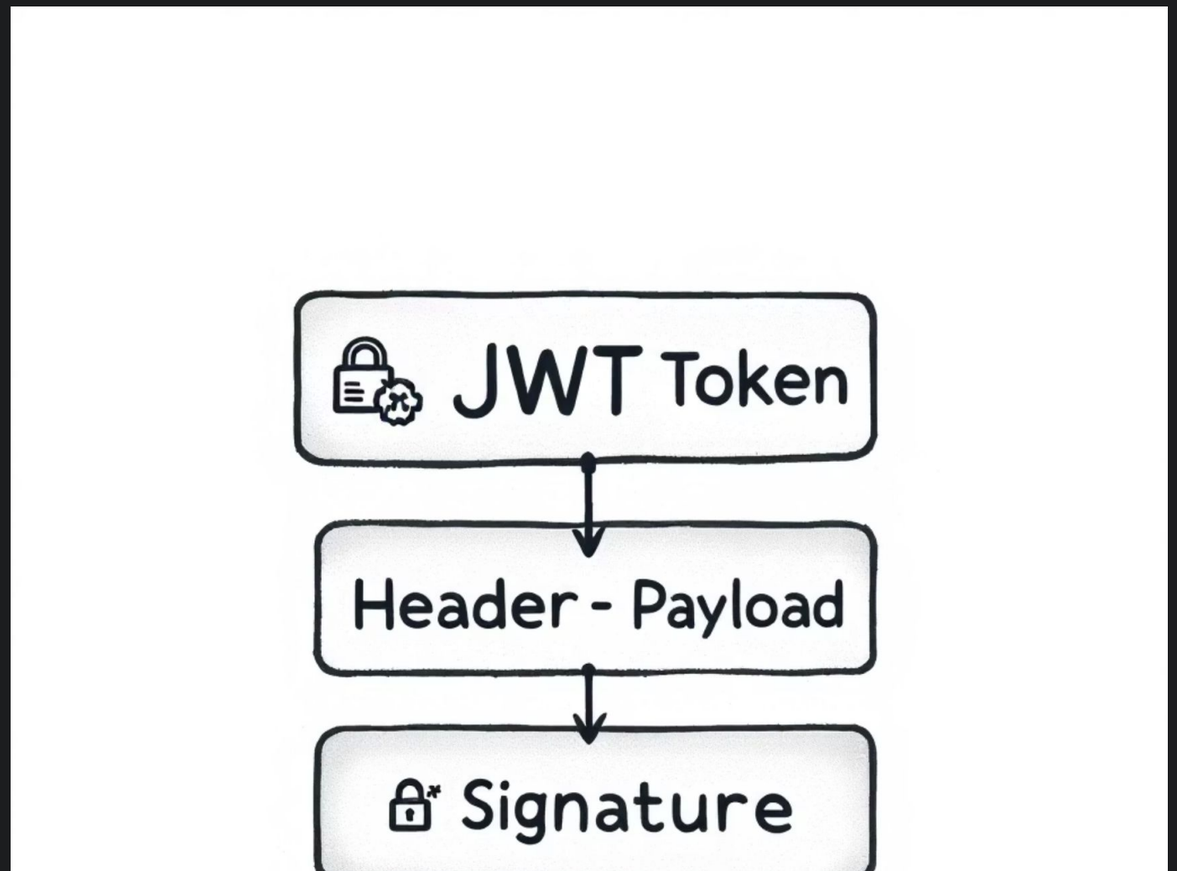
Autenticação com JWT

O que é JWT?

- Token assinado com chave secreta
- Carrega informações do usuário
- Stateless (sem sessão no servidor)

Estrutura

`Header.Payload.Signature`



Fluxo de Autenticação

1. Cliente envia login/senha
2. Servidor valida e retorna JWT
3. Cliente guarda token
4. Futuras requisições enviam Bearer <token>
5. Middleware valida

Armazenamento sugerido

- App Web: Cookie HTTP Only
- App Mobile: Secure Storage

Atividades Sugeridas e Encerramento

Atividade Prática

- Rascunho de modelo ER

Leitura Complementar

- jwt.io/introduction
- expressjs.com
- Sommerville – Engenharia de Software

Para Reflexão

- Como requisitos bem definidos evitam retrabalho?
- O que torna um sistema "seguro"?
- Por que separar responsabilidades (MVC) facilita o projeto?