




Material de Estudo: Configuração do Projeto e Ambiente Node.js

 **Mentor:** Jeremias O Nunes

 **Disciplina:** Desenvolvedor Web

 **Aula 6** – Ambientes Node.js + Express + API REST

 **Carga Horária:** 36h da UC | 1 aula de 4h

Objetivo Geral da Aula

Capacitar você, aluno desenvolvedor, a **criar do zero um servidor web moderno com Node.js**, utilizando **npm para gerenciar pacotes**, organizando o projeto com **MVC simplificado** e criando suas primeiras **rotas REST com Express.js**.

1. Node.js: Rodando JavaScript no Lado do Servidor

O que é Node.js?

Node.js é um **ambiente de execução JavaScript** fora do navegador. Ele permite usar a mesma linguagem do front-end para **criar servidores e serviços back-end**.

Analogia:

Imagine o JavaScript como um guitarrista que só tocava em casa (navegador). O Node.js é o palco profissional que o leva ao mundo exterior: shows, palcos, servidores — **ele agora toca em qualquer lugar**.

Por que é importante?

- Leve e rápido (usa o motor V8 do Chrome)
- Permite aplicações escaláveis (chats, APIs, sistemas reais)
- Base de grandes empresas: Netflix, PayPal, LinkedIn

Como funciona?

- Baseado em **event loop**
- Lida com **operações assíncronas** (como banco de dados, arquivos, rede)

- Ideal para APIs RESTful e microserviços

◇ 2. npm: O Gerenciador de Ferramentas do Projeto

🔑 O que é o npm?

É o **gerenciador de pacotes do Node.js**. Permite instalar bibliotecas prontas, scripts, plugins e automatizar tarefas.

🧠 Analogia:

Pense no npm como o “**mercado online de ferramentas para desenvolvedores**”. Nele você encontra martelos (Express), furadeiras (bcrypt), planilhas (Sequelize) e instala tudo com um simples comando.

🔧 Comandos essenciais:

bash

CopiarEditar

```
npm init -y          # Cria o package.json
npm install express   # Instala o Express
npm install bcryptjs  # Instala biblioteca de criptografia
npm install sequelize # ORM para bancos relacionais
```

■ O que é o package.json?

É o **documento oficial do seu projeto**, onde estão:

- Nome, versão
- Dependências instaladas
- Scripts de execução (start, dev, etc)

◇ 3. Git: Controle de Versão Profissional

🔑 O que é o Git?

Sistema de **controle de versão**. Salva as versões do seu projeto, permite colaborar com outras pessoas e voltar no tempo, se necessário.

🌐 Analogia:

Git é como um **diário de bordo de um navio**. Registra todas as alterações e te permite navegar de volta se algo der errado.

🌐 GitHub:

Plataforma online para **armazenar e compartilhar** seus repositórios. Ideal para colaboração e versionamento em equipe.

📄 .gitignore:

Arquivo que lista **pastas e arquivos que não devem ser versionados**, como:

```
node_modules/  
.env  
logs/
```

◇ 4. Organizando seu Projeto com MVC Simplificado

🔗 Por que organizar?

Projetos mal organizados são difíceis de manter. O padrão **MVC (Model - View - Controller)** ajuda a dividir o código em responsabilidades.

📁 Estrutura:

```
meu-backend/  
├── controllers/    # Lógica da aplicação  
├── routes/         # Caminhos e endpoints da API  
├── models/         # Representações dos dados  
├── config/         # Conexões e arquivos de configuração  
├── server.js       # Início da aplicação  
├── .gitignore  
└── package.json
```

🔗 Função de cada pasta:

| Pasta | Finalidade |
|-------------|---|
| controllers | Manipula requisições e respostas da API |
| routes | Define os caminhos da API (ex: GET /produtos) |
| models | Define as estruturas dos dados (ex: Produto) |
| config | Configurações do banco de dados |
| server.js | Ponto inicial da aplicação |

◇ 5. Instalando as Dependências Essenciais

📦 Instalação:

```
npm install express bcryptjs sequelize
```

🔍 Para que serve:

- **Express:** Criação de rotas e controle do servidor
- **bcryptjs:** Criptografia de senhas e dados sensíveis
- **sequelize:** ORM (Object-Relational Mapping) que conecta seu código JS ao banco de dados

◇ 6. Criando o Servidor com Express.js

📄 Código base: server.js

```
const express = require('express');
const app = express();
const PORT = 3000;

// Middleware para entender JSON
app.use(express.json());

// Rota inicial
app.get('/', (req, res) => {
  res.json({ mensagem: "Servidor funcionando!" });
});

// Nova rota personalizada
app.get('/teste', (req, res) => {
  res.json({ aluno: "Jeremias" });
});

// Iniciar servidor
app.listen(PORT, () => {
  console.log(`Servidor rodando em 
```

Teste no navegador:

- Acesse: <http://localhost:3000/>
- Veja o JSON com a mensagem
- Acesse: /teste para retornar seu nome

7. API REST: Comunicação Organizada

O que é REST?

REST é um padrão de comunicação entre sistemas. Utiliza **URLs + métodos HTTP** para executar ações.

| Método | Ação | Exemplo |
|--------|-----------|----------------|
| GET | Listar | /usuario |
| POST | Criar | / usuario |
| PUT | Atualizar | /usuario /1 |
| DELETE | Remover | /usuario /1 |

Exemplo prático:

```
app.get('/hello', (req, res) => {  
  const nome = req.query.name || "visitante";  
  res.json({ mensagem: `Olá, ${nome}!` });  
});
```

URL Teste:

GET /hello?name=Lucas

Retorno:

```
{ "mensagem": "Olá, Lucas!" }
```

◇ 8. bcryptjs e sequelize: Segurança e Banco

🔒 bcryptjs – Criptografia de Senhas

```
const bcrypt = require('bcryptjs');
const senha = '123456';
const hash = bcrypt.hashSync(senha, 10); // gera hash
console.log(hash);
```

🔒 Nunca salve senhas em texto puro no banco!

🐘 Sequelize – ORM

```
const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('nome_banco', 'usuario', 'senha', {
  host: 'localhost',
  dialect: 'mysql' // ou postgres, sqlite
});

// Exemplo de model:
const Produto = sequelize.define('Produto', {
  nome: Sequelize.STRING,
  preco: Sequelize.FLOAT
});
```

Avaliação dos Aprendizados:

| Critério | Evidência Esperada |
|-----------------------------------|--|
| Projeto inicializado corretamente | package.json criado, .gitignore funcional |
| Pacotes instalados com sucesso | express, bcryptjs, sequelize no package.json |
| Estrutura organizada | Pastas MVC criadas conforme modelo |
| Servidor Express funcional | Rota GET / funcionando e testável |
| Participação ativa | Discussões, perguntas, ajuda a colegas |

Recursos para Estudo:

- Documentação Node.js
- Documentação Express.js
- [Guia Rápido npm](#)
- [Documentação Sequelize](#)
- [bcryptjs no npm](#)

Conclusão Final da Aula

Ao final dessa jornada, você **entende como montar um back-end moderno com JavaScript**, como estruturar, proteger e expandir seu projeto com **boas práticas**. Está pronto para **construir APIs REST e conectar com front-ends**.