

Curso de Node.JS

Guia do Programador (Victor Lima)

https://www.youtube.com/watch?v=LLqq6FemMNQ&list=PLJ_KhUnIXUPtbtLwaxxUxHqvcNQndmI4B

Resumo do curso feito por Roberto Pinheiro

Aula 01 - O que é o Node.JS

Node.JS é um interpretador javascript que não depende do navegador, ou seja, ele é totalmente desvinculado do navegador.

Os navegadores mais modernos tem um javascript dentro do seu núcleo. A grande sacada do Node foi tirar o javascript dos navegadores, ou seja, agora é possível criar programas javascript fora dos navegadores.

Com a chegada do Node e a possibilidade de executar código javascript fora dos navegadores é possível criar aplicações back-end, aplicações de linha de comando, sistema back-end de jogos, programas para desktop, programas de linha de comando, etc.

As duas bases do Node são:

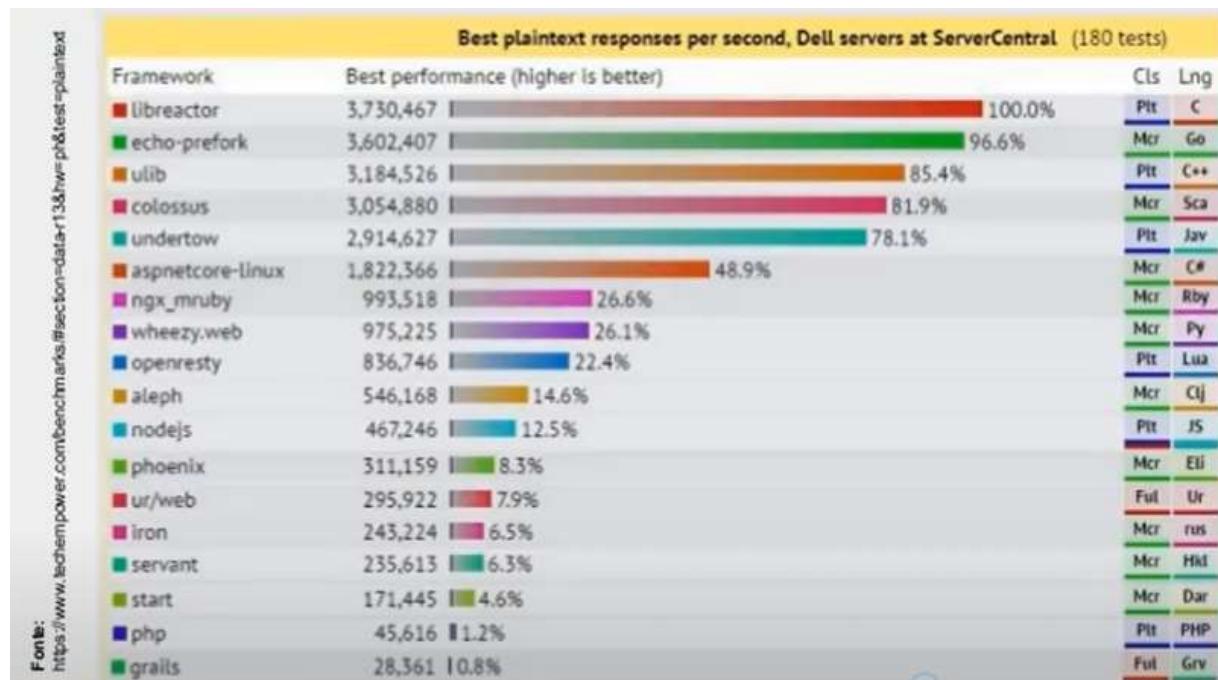
V8: Motor javascript da Google que está presente dentro do Google Chrome. Ele é o responsável por entender o javascript.

libuv: é uma biblioteca que deu características de uma linguagem back-end para o Node. Você consegue se conectar a um banco de dados graças a ela.



Vantagens

- Muito leve, pouco uso de memória Ram, melhor aproveitamento da CPU.
- Utiliza javascript
- Tem um dos maiores ecossistemas de bibliotecas, módulos e plug-ins do mundo.
- Suporta um número imenso de requisições



Aula 02 - Como instalar o Node.JS

Faça o download do Node.JS e instale a versão LTS no computador.

<https://nodejs.org/en/>

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x86)

10.16.0 LTS
Recommended For Most Users

12.7.0 Current
Latest Features

Other Downloads | Changelog | API Docs Other Downloads | Changelog | API Docs

Or have a look at the [Long Term Support \(LTS\) schedule](#).

Sign up for [Node.js Everywhere](#), the official Node.js Monthly Newsletter.

LINUX FOUNDATION COLLABORATIVE PROJECTS

Report Node.js issue | Report website issue | Get Help

© Node.js Foundation. All Rights Reserved. Portions of this site originally © Joyent.

Node.js is a trademark of Joyent, Inc. and is used with its permission. Please review the Trademark Guidelines of the Node.js Foundation.

Linux Foundation is a registered trademark of The Linux Foundation.

Linux is a registered trademark of Linus Torvalds.

```
C:\guia_programador\nodejs>node -v  
v14.15.3  
  
C:\guia_programador\nodejs>npm -v  
6.14.9
```

- Como editor de código utilize o **Visual Studio Code**.

Aula 03 - Node na prática

Você vai entender como utilizar de fato o node.js, criando um exemplo prático, e vendo em detalhes como a plataforma funciona.

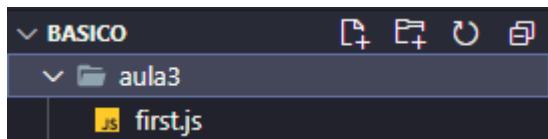
Testes iniciais

- Em C:\guia_programador\nodejs crie a pasta **basico**.

```
cd basico
```

```
code .
```

- Dentro da pasta basico, crie a subpasta **aula3** e dentro dela adicione o arquivo **first.js**



aula3\first.js

```
console.log('Hello World')
```

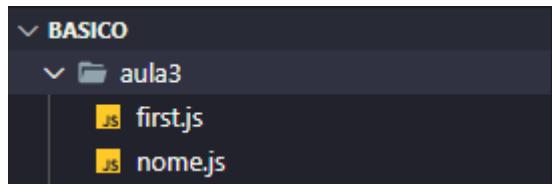
Obs.: No final de cada linha de comando o sinal ; é opcional.

No terminal do Visual Studio Code, na pasta do projeto (basico), entre com o comando:

```
node aula3\first.js
```

```
C:\guia_programador\nodejs\basico\aula3>node first.js
Hello World
```

- Crie o arquivo **nome.js**



aula3\ nome.js

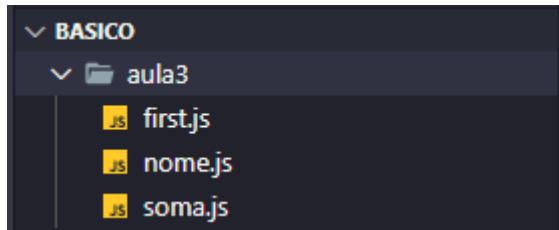
```
var nome = 'Roberto'  
var sobrenome = 'Pinheiro'  
  
console.log(nome + ' ' + sobrenome)
```

- No terminal, entre com o comando:

node nome.js

```
C:\guia_programador\nodejs\basico\aula3>node nome.js  
Roberto Pinheiro
```

- Crie o arquivo **soma.js**



aula3\soma.js

```
var num1 = 15
var num2 = 42

function somar(a, b){
  return a + b
}

console.log(somar(num1, num2))
```

Entre com o comando:

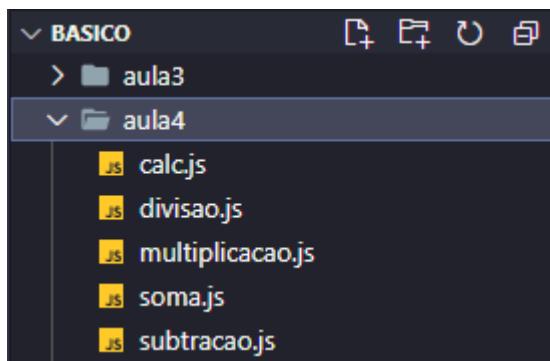
node soma.js

```
C:\guia_programador\nodejs\basico\aula3>node soma.js
57
```

Aula 04 - Módulos

Você vai aprender a como trabalhar com módulos no Node.js, que são uma forma de você dividir toda a lógica da sua aplicação em arquivos diferentes, utilizando a palavra **call module.exports** você consegue exportar objetos de um arquivo, e utilizando a palavra chave **require** ou **import**, você consegue importar coisas para dentro de determinados arquivos. Esse é um dos principais recursos do Node.js e Javascript.

- Crie a pasta **aula4** e dentro dela adicione os arquivos:



aula4\soma.js

```
var soma = function (a,b){  
    return a + b  
}  
  
module.exports = soma;
```

aula4\subtracao.js

```
var subtracao = function (a,b){  
    return a - b  
}  
  
module.exports = subtracao;
```

aula4\multiplicacao.js

```
var multiplicacao = function (a,b){  
    return a * b  
}  
  
module.exports = multiplicacao;
```

aula4\divisao.js

```
var divisao = function (a,b){  
    return a / b  
}  
  
module.exports = divisao;
```

aula4\calc.js

```
var soma = require("./soma")  
var subt = require("./subtracao")  
var multip = require("./multiplicacao")  
var div = require("./divisao")  
  
console.log(soma(1,2))  
console.log(subt(2,4))  
console.log(multip(3,9))  
console.log(div(12,3))
```

- No terminal, na pasta **aula4**, entre com o comando:

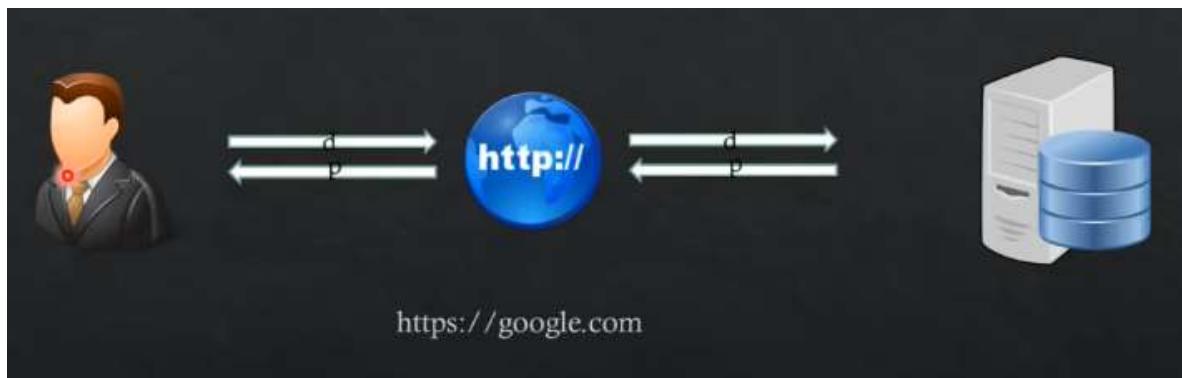
node calc.js

```
C:\guia_programador\nodejs\basico\aula4>node calc.js  
3  
-2  
27  
4
```

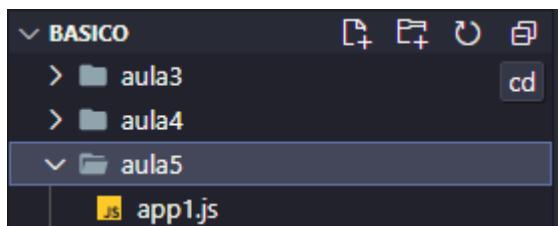
Aula 05 - Protocolo HTTP

Nesta aula você vai entender o que é o protocolo HTTP, que é a entidade responsável por criar toda a comunicação entre servidor e cliente na WEB, como também vai aprender a como criar o seu próprio servidor HTTP com Node.js e Javascript, para já dar os seus primeiros passos com o desenvolvimento web com node.js.

HTTP é um protocolo que permite que um cliente se comunique com um servidor.



- Crie a pasta **aula5** e dentro dela adicione o arquivo **app1.js**



aula5\app1.js

```
var http = require('http');

http.createServer().listen(8081);

console.log('Servidor rodando!');
```

Esse é o **servidor nativo do Node.JS**

- Entre com o comando:

```
node app1.js
```

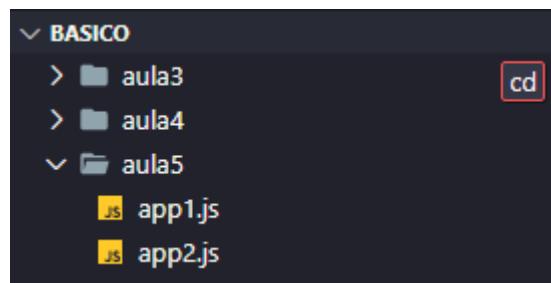
```
C:\guia_programador\nodejs\basico\aula5>node app1.js
Servidor rodando!
```

- Em um navegador, entre com a URL:

localhost:8081

Nada irá acontecer porque nada foi pedido ao servidor.

Fazendo uma solicitação ao servidor



aula5\app2.js

```
var http = require('http');

http.createServer(function(req, res){
  res.end("Hello World! Welcome to my website!");
}).listen(8081);

console.log('Servidor rodando!');
```

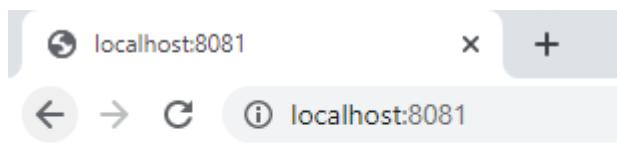
Para que as alterações tenham efeito é necessário primeiramente fechar o servidor (com CTRL-C) e em seguida abri-lo novamente.

```
node app2.js
```

```
C:\guia_programador\nodejs\basico\aula5>node app2.js  
Servidor rodando!
```

- Em um navegador, entre com a URL:

localhost:8081



Hello World! Welcome to my website!

Aula 06 - Introdução ao Express

Nesta aula você vai ser introduzido ao principal framework de desenvolvimento web com node.js, o Express ou Express.js, com ele é possível criar aplicações back-end http com javascript.

Express é o principal framework e a principal ferramenta usada para criar aplicações back-end utilizando o Node.

Instalando o Express

Vamos criar um novo projeto. Dentro de `C:\guia_programador\nodejs` crie uma pasta chamada `app`.

```
C:\guia_programador\nodejs>dir
 0 volume na unidade C é W10-195
 0 Número de Série do Volume é 6047-7D0C

  Pasta de C:\guia_programador\nodejs

30/04/2021  08:37    <DIR>          .
30/04/2021  08:37    <DIR>          ..
30/04/2021  08:37    <DIR>          app
30/04/2021  08:35    <DIR>          basico
                  0 arquivo(s)           0 bytes
                  4 pasta(s)   114.924.855.296 bytes disponíveis
```

```
cd app
code .
```

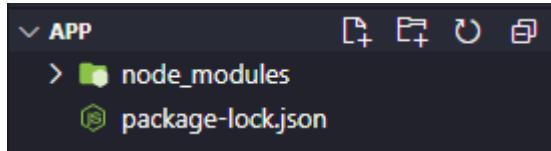
Dentro da pasta do projeto, entre com o comando:

```
npm install express --save
```

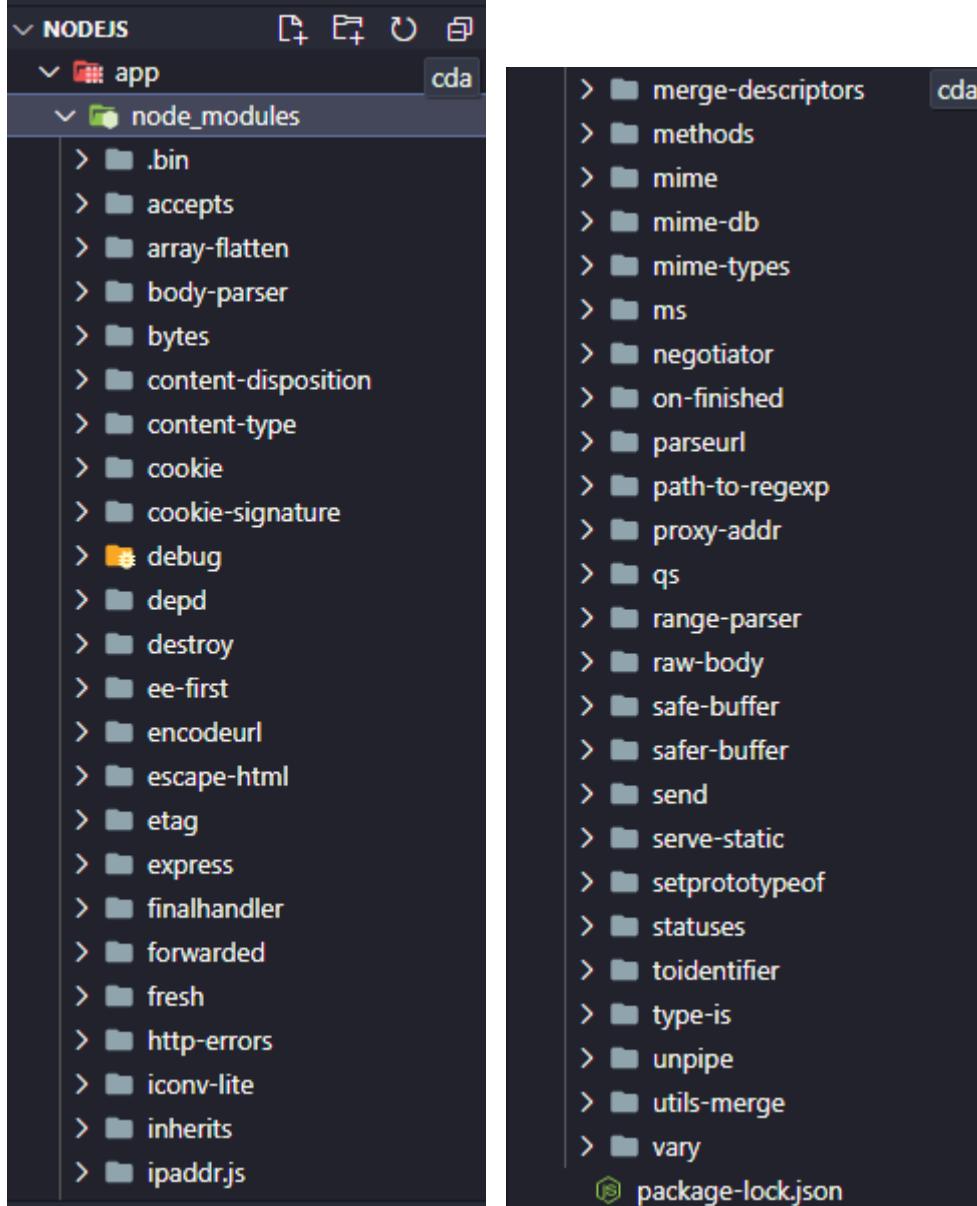
```
C:\guia_programador\nodejs\app>npm install express --save
npm [WARN] saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\app\package.json'
npm [notice] created a lockfile as package-lock.json. You should commit this file.
npm [WARN] enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\app\package.json'
npm [WARN] app No description
npm [WARN] app No repository field.
npm [WARN] app No README data
npm [WARN] app No license field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 24.003s
found 0 vulnerabilities
```

Dentro da pasta `app` é criada a subpasta `node_modules` com todas as dependências do projeto.



- Quando você baixa um pacote diretamente do **npm** pela primeira vez, é criado um arquivo chamado **package-lock.json**
- Este arquivo guarda informações a respeito de todos os módulos que existem dentro do seu projeto. É um arquivo de configuração do **npm** e, a princípio, não deve ser alterado.



Aula 7 - Rotas

Você vai aprender a como trabalhar com rotas no framework Express.js, que é uma forma de definir as url e caminhos da sua aplicação Node.js.

Função de callback é uma função que é executada sempre que algum evento acontece.

Uma rota é um caminho para a sua aplicação. O Express é um framework orientado a rotas.

app\aula7.js

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

app.get("/", function(req, res){
  res.send("Seja bem-vindo ao meu app!");
});

app.get("/sobre", function(req, res){
  res.send("Minha página sobre");
});

app.get("/blog", function(req, res){
  res.send("Bem-vindo ao meu blog!");
});

// abrindo o servidor com Express (essa deve ser a última função do código)

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

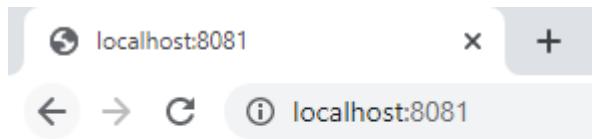
- Para que as alterações tenham efeito é necessário primeiramente fechar o servidor (com CTRL-C) e em seguida abri-lo novamente.

node aula7.js

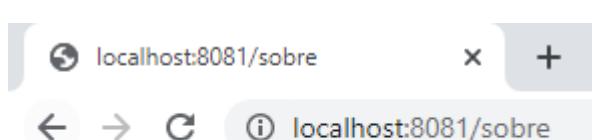
```
C:\guia_programador\nodejs\app>node aula7.js
Servidor rodando na URL: http://localhost:8081
```

No browser:

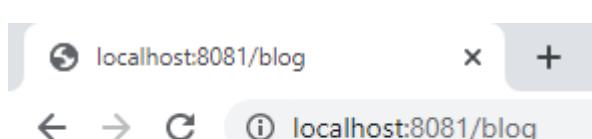
<http://localhost:8081/>



<http://localhost:8081/sobre>



<http://localhost:8081/blog>



Aula 8 - Parâmetros

Você vai aprender a como trabalhar com parâmetros em rotas utilizando o framework **Express.js** do Node.js para desenvolver sua aplicação web back-end utilizando Javascript.

app\aula8a.js

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

app.get("/", function(req, res){
  res.send("Seja bem-vindo ao meu app!");
});

app.get("/sobre", function(req, res){
  res.send("Minha página sobre");
});

app.get("/blog", function(req, res){
  res.send("Bem-vindo ao meu blog!");
});

app.get("/ola/:nome/:cargo/:cor", function(req, res){
  res.send(req.params);
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

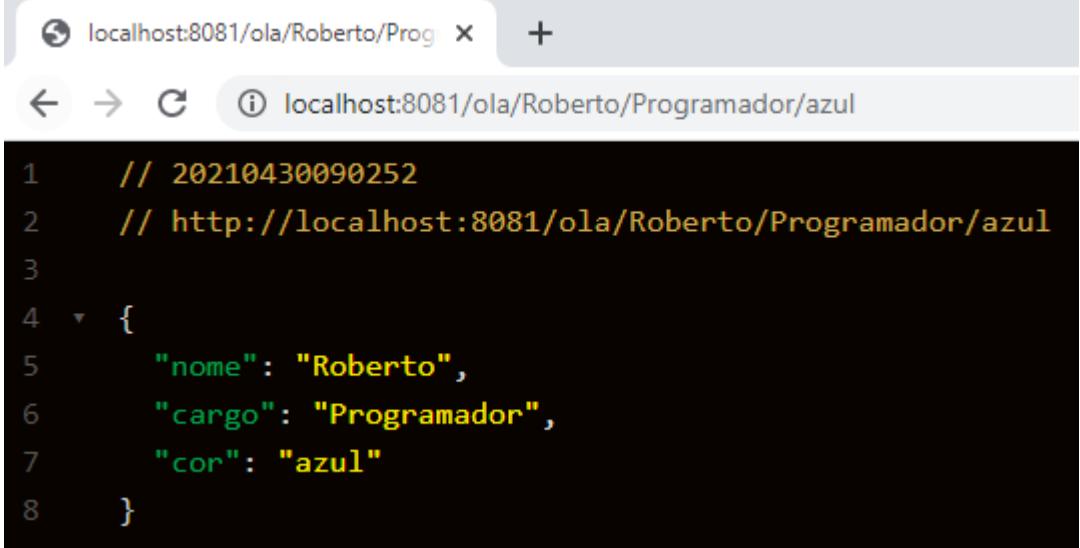
- Para que as alterações tenham efeito é necessário primeiramente fechar o servidor (com CTRL-C) e em seguida abri-lo novamente.

node aula8a.js

```
C:\guia_programador\nodejs\app>node aula8a.js
Servidor rodando na URL: http://localhost:8081
```

No browser:

localhost:8081/ola/Roberto/Programador/azul



A screenshot of a web browser window. The address bar shows "localhost:8081/ola/Roberto/Programador/azul". The main content area displays the following JSON data:

```
1 // 20210430090252
2 // http://localhost:8081/ola/Roberto/Programador/azul
3
4 {
5     "nome": "Roberto",
6     "cargo": "Programador",
7     "cor": "azul"
8 }
```

app\aula8b.js

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

app.get("/", function(req, res){
  res.send("Seja bem-vindo ao meu app!");
});

app.get("/sobre", function(req, res){
  res.send("Minha página sobre");
});

app.get("/blog", function(req, res){
  res.send("Bem-vindo ao meu blog!");
});

app.get("/ola/:nome/:cargo/:cor", function(req, res){
  res.send("<h1>Olá " + req.params.nome + "</h1>" + "<h2>Seu cargo é " + req.params.cargo +
  "</h2>" + "<h3>Sua cor favorita é " + req.params.cor + "</h3>");
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

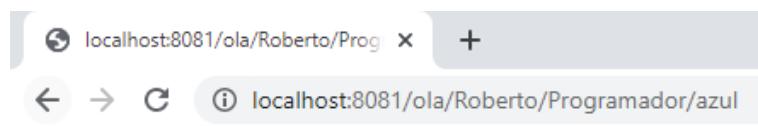
- Para que as alterações tenham efeito é necessário primeiramente fechar o servidor (com CTRL-C) e em seguida abri-lo novamente.

node testes\aula8b.js

```
C:\guia_programador\nodejs\app>node aula8b.js
Servidor rodando na URL: http://localhost:8081
```

No browser:

localhost:8081/ola/Roberto/Programador/azul



Aula 9 - Nodemon

Você vai aprender a como utilizar a ferramenta Nodemon para poder executar os seus programas Node.js de uma forma continua sem a necessidade de ficar toda hora recarregando o Node.js para desenvolver suas aplicações Web back-end utilizando o framework express.js e o Javascript.

Instalando o Nodemon

```
npm install nodemon -g
```

```
C:\guia_programador\nodejs\app>npm install nodemon -g
C:\Users\beto1\AppData\Roaming\npm\nodemon -> C:\Users\beto1\AppData\Roaming\npm\node_modules\nodemon\bin\nodemon.js
> nodemon@2.0.7 postinstall C:\Users\beto1\AppData\Roaming\npm\node_modules\nodemon
> node bin/postinstall || exit 0

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.3.1 (node_modules\nodemon\node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ nodemon@2.0.7
updated 1 package in 37.958s
```

Rodando o servidor com o nodemon

Entre com o comando:

```
nodemon aula8b.js
```

```
C:\guia_programador\nodejs\app>nodemon aula8b.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node aula8b.js`
Servidor rodando na URL: http://localhost:8081
|
```

Faça uma alteração no arquivo:

app\testes\aula8b.js

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

app.get("/", function(req, res){
  res.send("Bem-vindo ao meu app!");
});

app.get("/sobre", function(req, res){
  res.send("Minha página sobre");
});

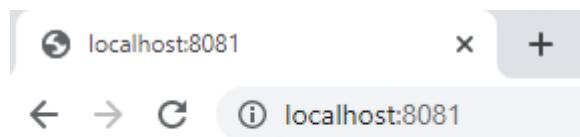
app.get("/blog", function(req, res){
  res.send("Bem-vindo ao meu blog!");
});

app.get("/ola/:nome/:cargo/:cor", function(req, res){
  res.send("<h1>Olá " + req.params.nome + "</h1>" + "<h2>Seu cargo é " + req.params.cargo +
"</h2>" + "<h3>Sua cor favorita é " + req.params.cor + "</h3>");
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

A alteração será detectada pelo nodemon:

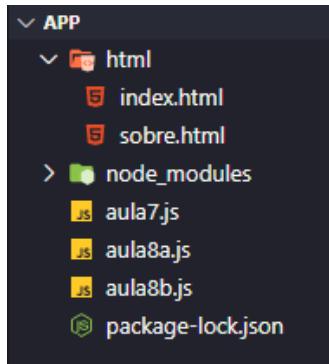
```
C:\guia_programador\nodejs\app>nodemon aula8b.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node aula8b.js`
Servidor rodando na URL: http://localhost:8081
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node aula8b.js`
Servidor rodando na URL: http://localhost:8081
[]
```



Bem-vindo ao meu app!

Aula 10 - Exibindo HTML

Nesta aula você vai aprender a como exibir arquivos HTML, ou seja, como trabalhar com views no framework Express.js do Node.js para desenvolvimento de aplicações Web com Javascript



app\html\index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Meu App!</title>
</head>
<body>
<h1>Olá! Seja bem-vindo ao meu app!</h1>
<h2>Meu nome é Roberto Pinheiro</h2>
</body>
</html>
```

app\html\sobre.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Sobre</title>
</head>
<body>
<h1>Sobre Nós</h1>
</body>
</html>
```

app\aula10.js

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

app.get("/", function(req, res){
  res.sendFile(__dirname + "/html/index.html");
});

app.get("/sobre", function(req, res){
  res.sendFile(__dirname + "/html/sobre.html");
});

app.get("/blog", function(req, res){
  res.send("Bem-vindo ao meu blog!");
});

app.get("/ola/:nome/:cargo/:cor", function(req, res){
  res.send("<h1>Olá " + req.params.nome + "</h1>" + "<h2>Seu cargo é " + req.params.cargo +
  "</h2>" + "<h3>Sua cor favorita é " + req.params.cor + "</h3>");
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

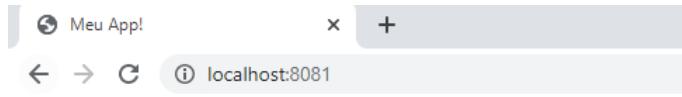
Rodando o servidor

nodemon aula10.js

```
C:\guia_programador\nodejs\app>nodemon aula10.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node aula10.js`
Servidor rodando na URL: http://localhost:8081
|
```

- No browser:

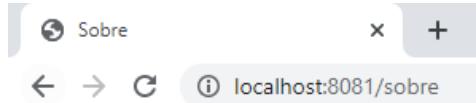
localhost:8081



Olá! Seja bem-vindo ao meu app!

Meu nome é Roberto Pinheiro

localhost:8081/sobre



Sobre Nós

Aula 11 - Instalando o MySQL

Nesta aula você vai aprender a como instalar o MySQL, que será o nosso sistema de banco de dados enquanto estivermos trabalhando com desenvolvimento web com Node.js e Javascript.

- Faça o download do MySQL (a versão MySQL Community Edition)

<https://www.mysql.com/>

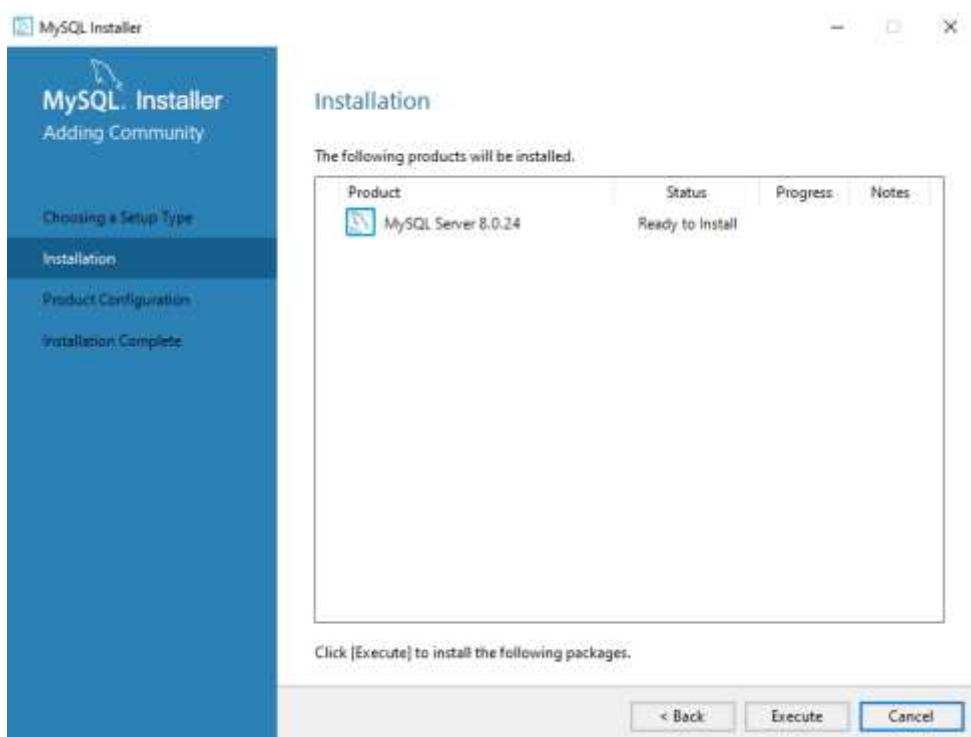
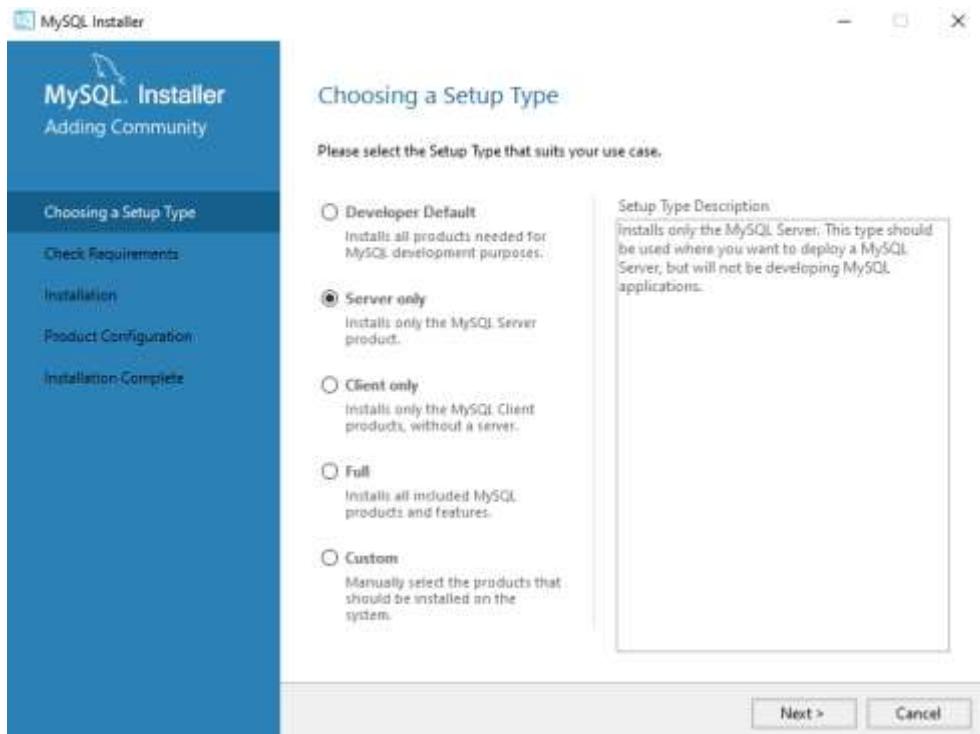


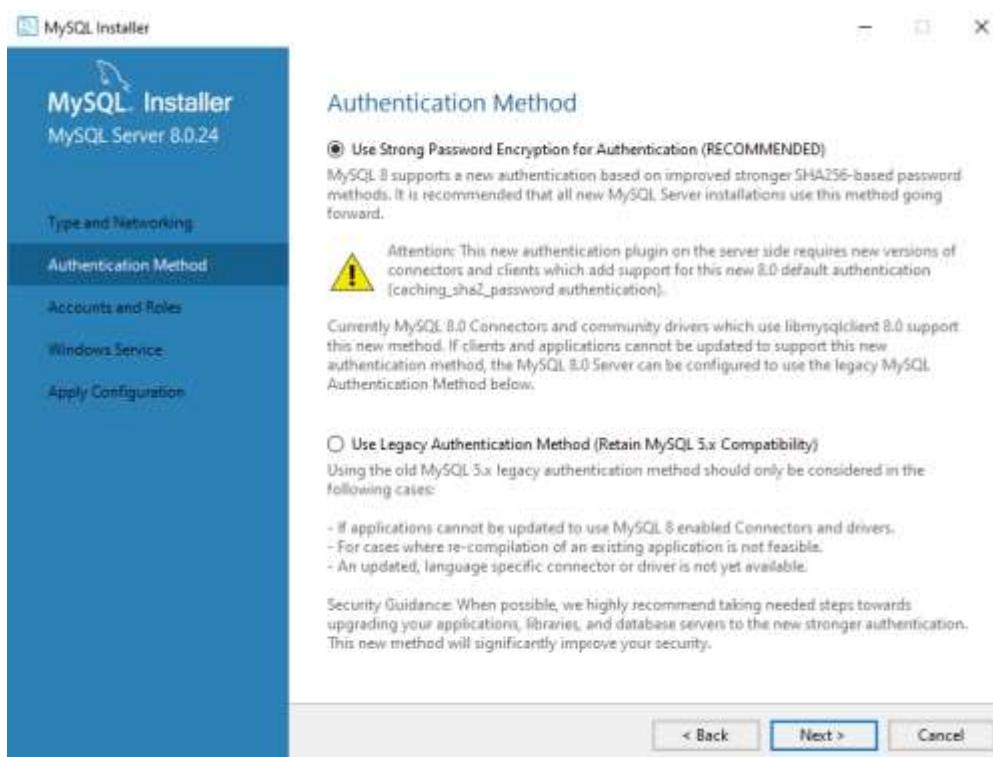
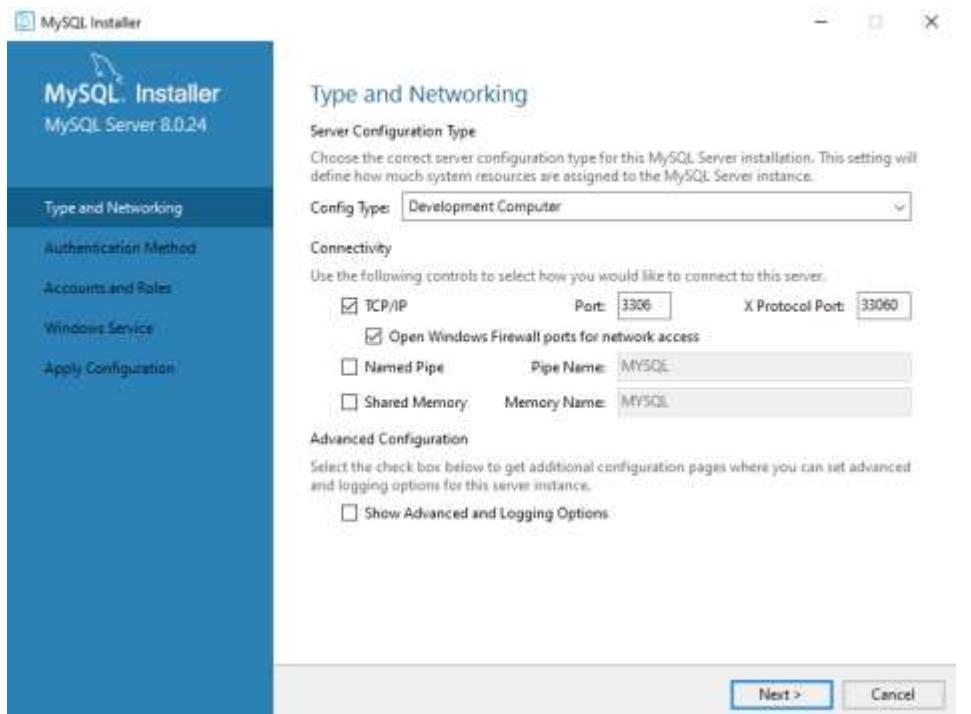
<https://dev.mysql.com/downloads/>

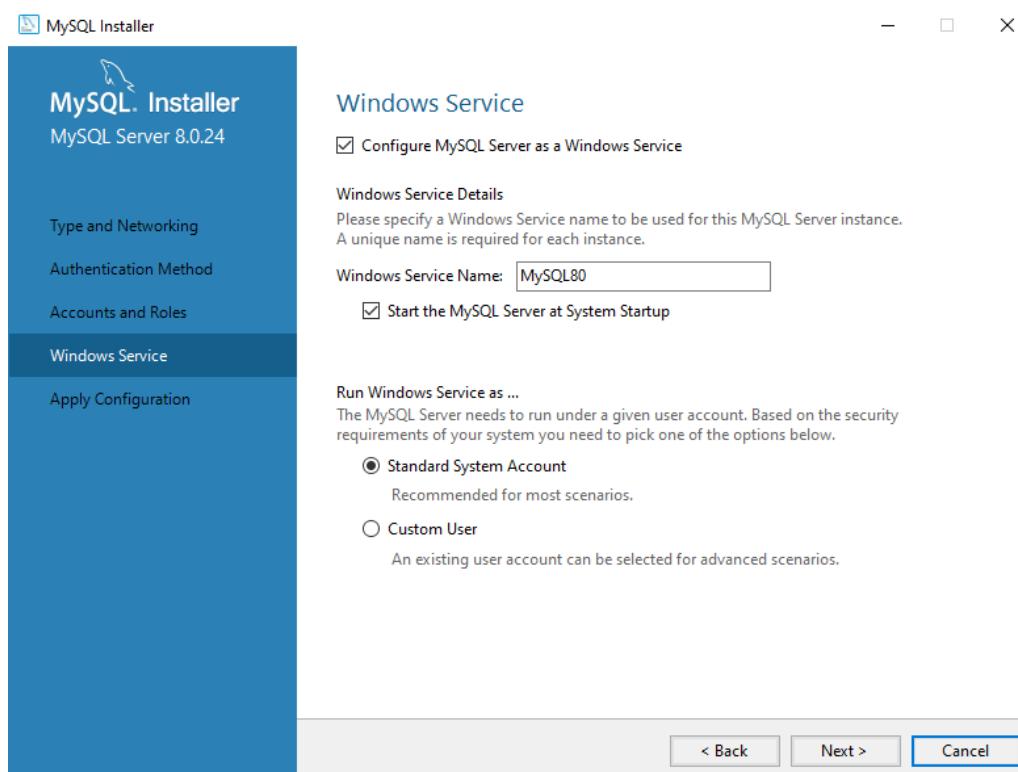
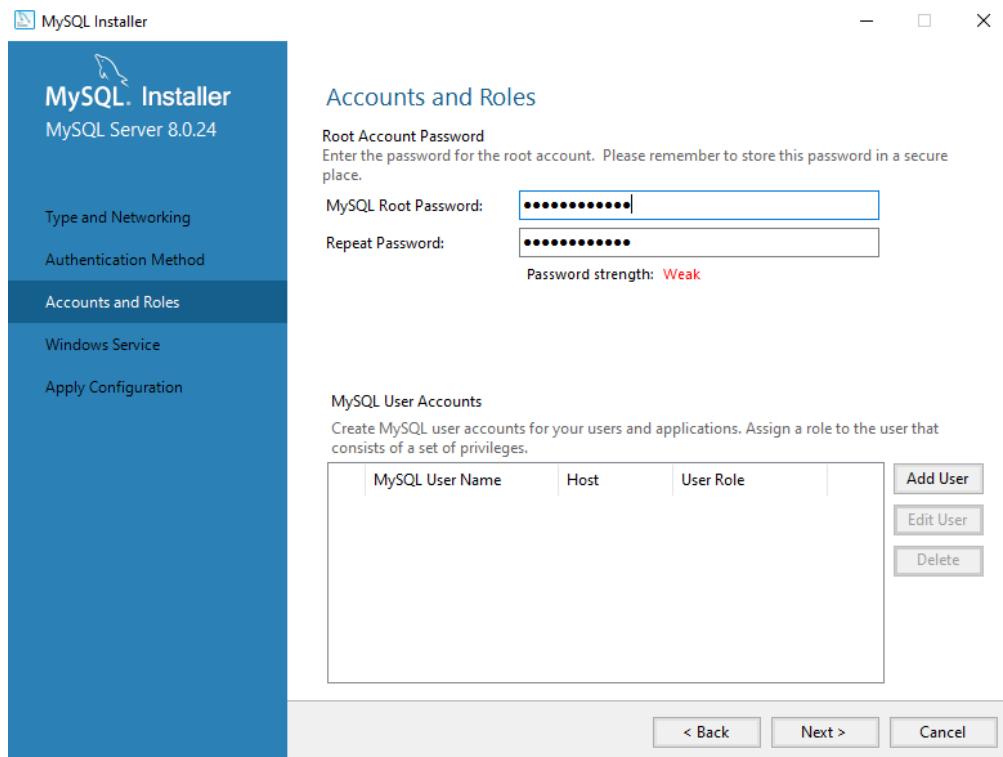
- Baixe e instale o MySQL Community Server

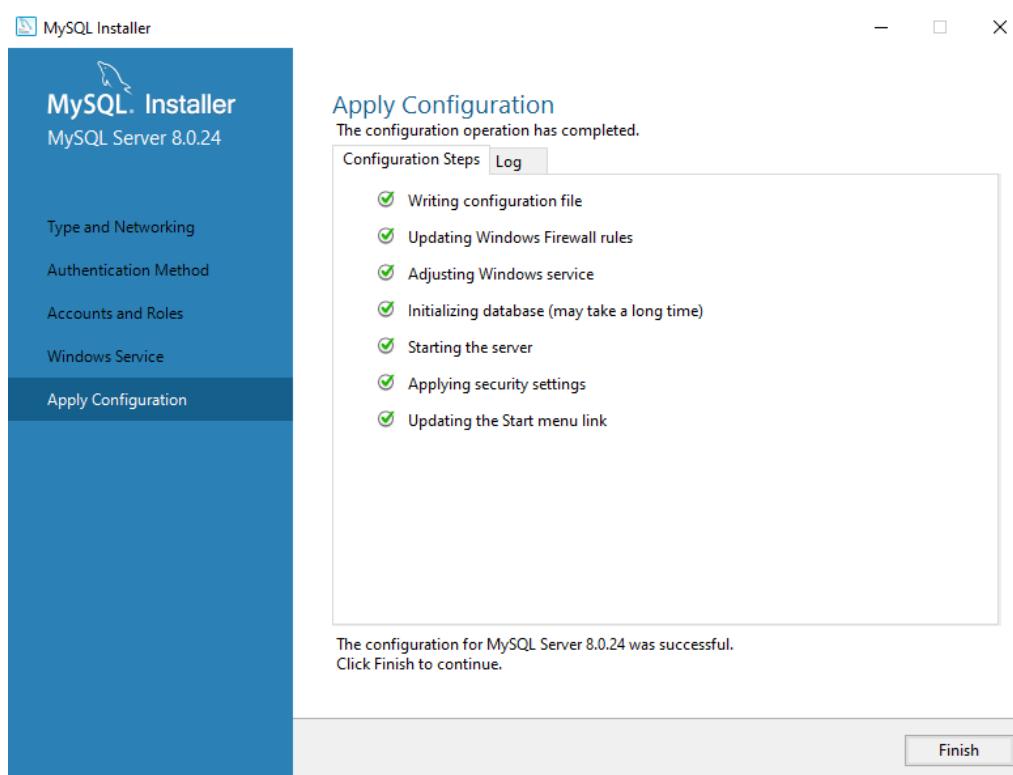
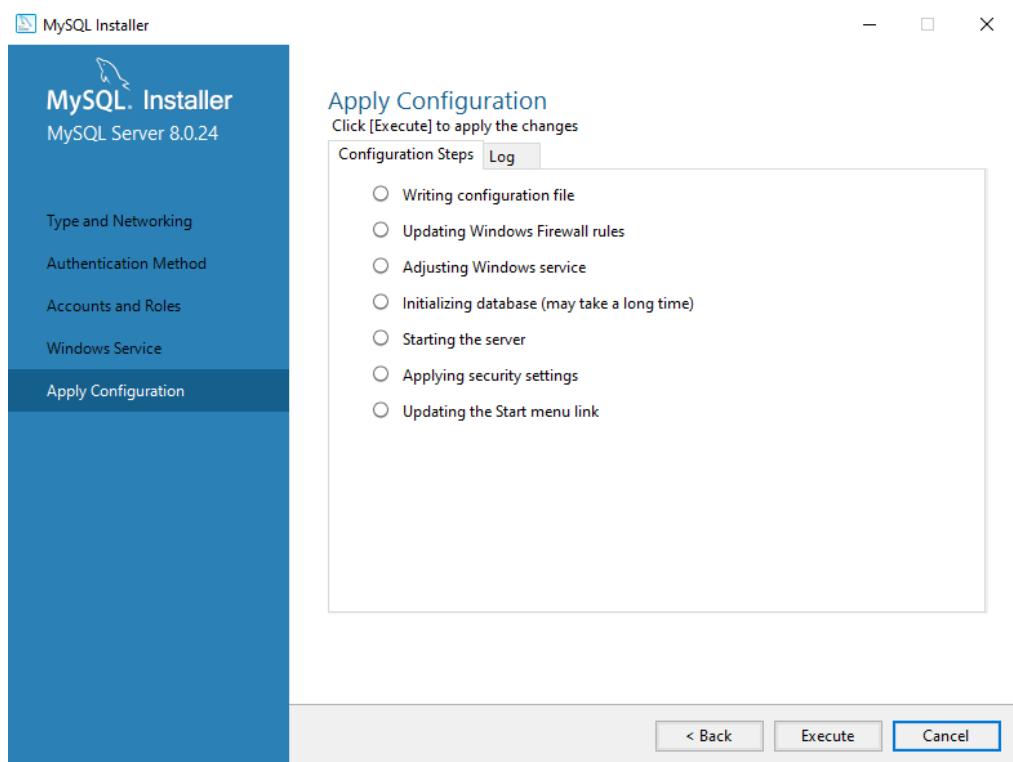
① MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL, Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench
- MySQL Installer for Windows
- MySQL for Visual Studio
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/.NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP
- MySQL Benchmark Tool
- Time zone description tables
- Download Archives





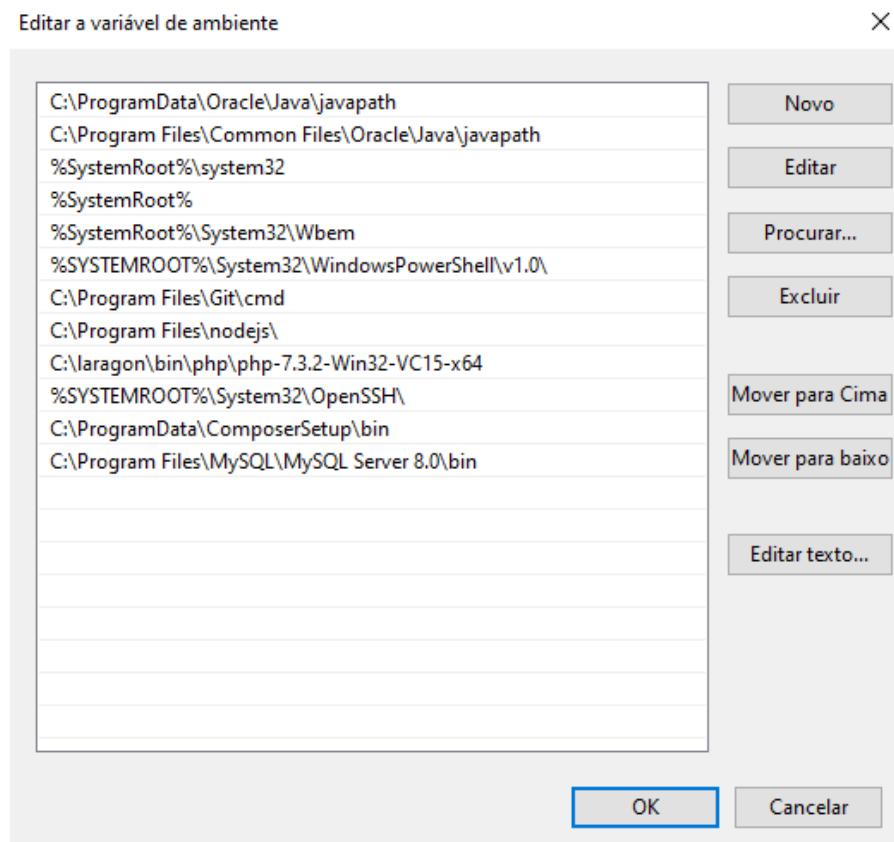




Inserir o path em variáveis de ambiente:

- Clique com o botão direito do mouse sobre o ícone Computador. Selecione Propriedades.
- Selecione a opção Configurações Avançadas do Sistema.
- Clique no botão Variáveis de Ambiente.
- Edite o path, inserindo o caminho da pasta bin do MySQL.

C:\Program Files\MySQL\MySQL Server 8.0\bin



- Reinicie o computador.
- No terminal, entre com o comando:

`mysql -u root -p`

E insira a senha.

```
C:\Users\beto1>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.24 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Ou:

- Clique no aplicativo instalado, chamado:

MySQL 5.7 Command Line Client

E entre com a senha.

```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.24 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

- Em caso de problemas: Instalar o pacote Microsoft Visual C++ 2015 e refazer a instalação.



Trocando a senha do MySQL

- Inicialmente, acesse com a senha inicial:

```
mysql -u root -p
```

Uma maneira simples de alterar a senha do root para versões recentes do MySQL é usando o comando ALTER USER. No entanto, esse comando não funcionará agora porque as tabelas de permissões não estão carregadas.

Vamos dizer ao servidor de banco de dados para recarregar as tabelas de permissões emitindo o comando FLUSH PRIVILEGES.

```
FLUSH PRIVILEGES;
```

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.16 sec)
```

Alterando a senha para 123456

Para o MySQL 5.7.6 e mais recentes, bem como o MariaDB 10.1.20 e mais recentes, use o seguinte comando:

```
ALTER USER 'root'@'localhost' IDENTIFIED BY '123456';
```

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY '123456';
Query OK, 0 rows affected (1.11 sec)
```

Aula 12 - Como criar tabelas no MySQL

Nesta aula, continuando nossa jornada no mundo do desenvolvimento web com javascript e node.js, você vai aprender a como criar tabelas no MySQL.

```
mysql -h localhost -u root -p
```

```
C:\Users\beto1>mysql -h localhost -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.24 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
SHOW DATABASES;
```

```
mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sys            |
+-----+
4 rows in set (1.97 sec)

mysql>
```

Criando o banco de dados

```
CREATE DATABASE sistema_cadastro;
```

```
mysql> CREATE DATABASE sistema_cadastro;
Query OK, 1 row affected (1.39 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sistema_cadastro |
| sys |
+-----+
5 rows in set (0.02 sec)
```

Selecionando o banco de dados

```
use sistema_cadastro;
```

```
mysql> use sistema_cadastro;
Database changed
```

Criando a tabela usuarios

```
CREATE TABLE usuarios (
    nome VARCHAR(50),
    email VARCHAR(100),
    idade INT
);
```

- Copie e cole o código na linha de comando do MySQL:

```
mysql> CREATE TABLE usuarios (
    ->     nome VARCHAR(50),
    ->     email VARCHAR(100),
    ->     idade INT
    -> );
Query OK, 0 rows affected (3.00 sec)
```

SHOW TABLES;

```
mysql> SHOW TABLES;
+-----+
| Tables_in_sistema_cadastro |
+-----+
| usuarios
+-----+
1 row in set (0.44 sec)
```

Visualizando a estrutura da tabela

DESCRIBE usuarios;

```
mysql> DESCRIBE usuarios;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nome  | varchar(50) | YES  |     | NULL    |       |
| email | varchar(100) | YES  |     | NULL    |       |
| idade | int        | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.11 sec)
```

Aula 13 - Manipulando o MySQL

Nesta aula você vai aprender a como trabalhar com os comandos insert, select e where, para manipular os seus bancos de dados para desenvolver web com Node.js

Inserindo dados na tabela

```
INSERT INTO usuarios(nome, email, idade) VALUES (
    "Roberto",
    "roberto@gmail.com",
    58
);
```

```
mysql> INSERT INTO usuarios(nome, email, idade) VALUES (
    ->     "Roberto",
    ->     "roberto@gmail.com",
    ->     58
    -> );
Query OK, 1 row affected (0.42 sec)
```

```
select * from usuarios;
```

```
mysql> select * from usuarios;
+-----+-----+-----+
| nome | email        | idade |
+-----+-----+-----+
| Roberto | roberto@gmail.com |      58 |
+-----+-----+-----+
1 row in set (0.04 sec)
```

```
INSERT INTO usuarios(nome, email, idade) VALUES ("Roberto", "roberto@gmail.com", 58);
INSERT INTO usuarios(nome, email, idade) VALUES ("Luis Silva", "luis_silva@gmail.com", 16);
INSERT INTO usuarios(nome, email, idade) VALUES ("Maria Ribeiro", "maria_ribeiro@gmail.com", 45);
INSERT INTO usuarios(nome, email, idade) VALUES ("Rita Santos", "rita_santos@gmail.com", 26);
```

- Copie e cole no MySQL:

```
mysql> INSERT INTO usuarios(nome, email, idade) VALUES ("Luis Silva", "luis_silva@gmail.com", 16);
Query OK, 1 row affected (0.30 sec)

mysql> INSERT INTO usuarios(nome, email, idade) VALUES ("Maria Ribeiro", "maria_ribeiro@gmail.com", 45);
Query OK, 1 row affected (0.18 sec)

mysql> INSERT INTO usuarios(nome, email, idade) VALUES ("Rita Santos", "rita_santos@gmail.com", 26);
Query OK, 1 row affected (0.08 sec)
```

`select * from usuarios;`

```
mysql> select * from usuarios;
+-----+-----+-----+
| nome | email | idade |
+-----+-----+-----+
| Roberto | roberto@gmail.com | 58 |
| Luis Silva | luis_silva@gmail.com | 16 |
| Maria Ribeiro | maria_ribeiro@gmail.com | 45 |
| Rita Santos | rita_santos@gmail.com | 26 |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

`select * from usuarios where idade >= 45;`

```
mysql> select * from usuarios where idade >= 45;
+-----+-----+-----+
| nome | email | idade |
+-----+-----+-----+
| Roberto | roberto@gmail.com | 58 |
| Maria Ribeiro | maria_ribeiro@gmail.com | 45 |
+-----+-----+-----+
2 rows in set (0.12 sec)
```

Aula 14 - Deletando registros no MySQL

Nesta aula você vai aprender a como deletar registros em seu banco de dados MySQL no nosso curso de desenvolvimento web com Node.js e Javascript.

```
delete from usuarios where nome = "Maria Ribeiro";
```

```
mysql> delete from usuarios where nome = "Maria Ribeiro";
Query OK, 1 row affected (0.88 sec)
```

```
select * from usuarios;
```

```
mysql> select * from usuarios;
+-----+-----+-----+
| nome | email | idade |
+-----+-----+-----+
| Roberto | roberto@gmail.com | 58 |
| Luis Silva | luis_silva@gmail.com | 16 |
| Rita Santos | rita_santos@gmail.com | 26 |
+-----+-----+-----+
3 rows in set (0.02 sec)
```

Aula 15 - Atualizando registros no MySQL

Nesta aula você vai aprender a como utilizar a operação update de atualização do Mysql para desenvolvermos nossos projetos web utilizando Node.js e Javascript.

```
update usuarios set nome = "Roberto Pinheiro" where nome = "Roberto";
```

```
mysql> update usuarios set nome = "Roberto Pinheiro" where nome = "Roberto";
Query OK, 1 row affected (0.28 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
select * from usuarios;
```

```
mysql> select * from usuarios;
+-----+-----+-----+
| nome | email | idade |
+-----+-----+-----+
| Roberto Pinheiro | roberto@gmail.com | 58 |
| Luis Silva | luis_silva@gmail.com | 16 |
| Rita Santos | rita_santos@gmail.com | 26 |
+-----+-----+-----+
3 rows in set (0.09 sec)
```

Aula 16 - Sequelize

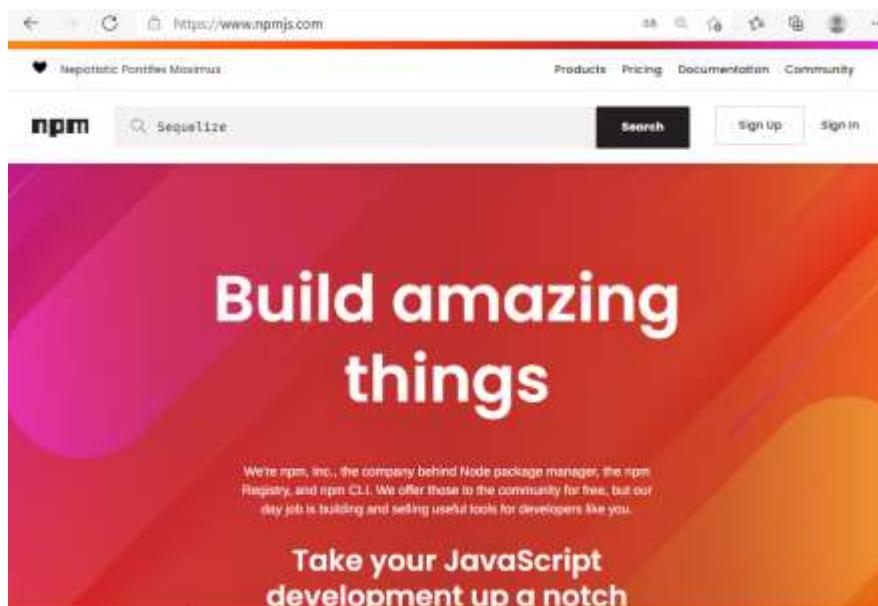
Nesta aula você vai conhecer o ORM Sequelize, que serve para abstrair toda a manipulação de bancos de dados SQL(MySQL, Postgres, etc...) para o Node.js. Juntos, vamos criar uma aplicação Node.js com Javascript e Sequelize.

Instalando o Sequelize

- Acesse a URL:

<https://www.npmjs.com/>

- E procure por **Sequelize**.



No terminal, na pasta do projeto, entre com o comando:

npm install sequelize --save

```
C:\guia_programador\nodejs\app>npm install sequelize --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\app\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\app\package.json'
npm WARN app No description
npm WARN app No repository field.
npm WARN app No README data
npm WARN app No license field.

+ sequelize@6.6.2
added 19 packages from 80 contributors and audited 107 packages in 39.604s
found 0 vulnerabilities
```

Instalando módulo para MySQL

```
npm install mysql2 --save
```

```
C:\guia_programador\nodejs\app>npm install mysql2 --save
npm [WARN] saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\app\package.json'
npm [WARN] enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\app\package.json'
npm [WARN] app No description
npm [WARN] app No repository field.
npm [WARN] app No README data
npm [WARN] app No license field.

+ mysql2@2.2.5
added 12 packages from 18 contributors and audited 125 packages in 8.755s
found 0 vulnerabilities
```

Aula 17 - Como se conectar ao MySQL

Nesta aula você vai aprender a como fazer uma conexão entre o Node.js com o Express.js ao MySQL, utilizando o Sequelize, o mesmo processo pode ser usado com outros SGDBs como Postgres e SQLServer.

Conectando com o banco de dados

app\conexao.js

```
const Sequelize = require('sequelize');
const sequelize = new Sequelize('sistema_cadastro', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

sequelize.authenticate().then(function(){
  console.log("Conectado com sucesso!");
}).catch(function(erro){
  console.log("Falha ao se conectar: " + erro);
});
```

nodemon conexao.js

```
C:\guia_programador\nodejs\app>nodemon conexao.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node conexao.js`
Executing (default): SELECT 1+1 AS result
Conectado com sucesso!
```

Configurando senha errada

app\conexao.js

```
const Sequelize = require('sequelize');
const sequelize = new Sequelize('sistema_cadastro', 'root', '123', {
  host: "localhost",
  dialect: "mysql"
});

sequelize.authenticate().then(function(){
  console.log("Conectado com sucesso!");
}).catch(function(error){
  console.log("Falha ao se conectar: " + error);
});
```

nodemon conexao.js

```
C:\guia_programador\nodejs\app>nodemon conexao.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node conexao.js`
Falha ao se conectar: SequelizeAccessDeniedError: Access denied for user 'root'@'localhost' (using password: YES)
[nodemon] clean exit - waiting for changes before restart
|
```

Aula 18 - Models no Sequelize

Nesta aula você vai entender o que é um model, e como criar um model para representar a sua tabela SQL no Node.js utilizando o sequelize.

```
mysql -h localhost -u root -p
```

```
C:\Users\beto1>mysql -h localhost -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 30
Server version: 8.0.24 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Criando o banco de dados

- Crie um banco de dados chamado **test**

```
create database test;
```

```
mysql> create database test;
Query OK, 1 row affected (0.37 sec)

mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sistema_cadastro |
| sys            |
| test           |
+-----+
6 rows in set (0.00 sec)
```

Criando as tabelas com o Sequelize

app\cria_tabela_postagens.js

```
const Sequelize = require('sequelize');
const sequelize = new Sequelize('test', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

sequelize.authenticate().then(function(){
  console.log("Conectado com sucesso!");
}).catch(function(erro){
  console.log("Falha ao se conectar: " + erro);
});

//Criando a tabela

const Postagem = sequelize.define('postagens', {
  titulo: {
    type: Sequelize.STRING
  },
  conteudo: {
    type: Sequelize.TEXT
  }
});

Postagem.sync({force:true});
```

app\cria_tabela_usuarios.js

```
const Sequelize = require('sequelize');
const sequelize = new Sequelize('test', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

sequelize.authenticate().then(function(){
  console.log("Conectado com sucesso!");
}).catch(function(erro){
  console.log("Falha ao se conectar: " + erro);
});

//Criando a tabela

const Usuario = sequelize.define('usuarios', {
  nome: {
    type: Sequelize.STRING
  },
  sobrenome: {
    type: Sequelize.STRING
  },
  idade: {
    type: Sequelize.INTEGER
  },
  email: {
    type: Sequelize.STRING
  }
});

Usuario.sync({force:true});
```

- No terminal, na pasta de trabalho, entre com o comando:

node cria_tabela_postagens.js

```
C:\guia_programador\nodejs\app>node cria_tabela_postagens.js
Executing (default): SELECT 1+1 AS result
Executing (default): DROP TABLE IF EXISTS `postagens`;
Conectado com sucesso!
Executing (default): CREATE TABLE IF NOT EXISTS `postagens` (`id` INTEGER NOT NULL auto_increment , `titulo` VARCHAR(255) , `conteudo` TEXT , `createdAt` DATETIME NOT NULL , `updatedAt` DATETIME NOT NULL , PRIMARY KEY (`id`)) ENGINE=InnoDB;
Executing (default): SHOW INDEX FROM `postagens`
```

- E em seguida, entre com o comando:

node cria_tabela_usuarios.js

```
C:\guia_programador\nodejs\app>node cria_tabela_usuarios.js
Executing (default): SELECT 1+1 AS result
Executing (default): DROP TABLE IF EXISTS `usuarios`;
Conectado com sucesso!
Executing (default): CREATE TABLE IF NOT EXISTS `usuarios` (`id` INTEGER NOT NULL auto_increment , `nome` VARCHAR(255) , `sobrenome` VARCHAR(255) , `idade` INTEGER , `email` VARCHAR(255) , `createdAt` DATETIME NOT NULL , `updatedAt` DATETIME NOT NULL , PRIMARY KEY (`id`)) ENGINE=InnoDB;
Executing (default): SHOW INDEX FROM `usuarios`
```

- No MySQL:

show databases;

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sistema_cadastro |
| sys |
| test |
+-----+
6 rows in set (0.13 sec)

mysql> use test;
Database changed
```

```
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| postagens      |
| usuarios       |
+-----+
2 rows in set (0.07 sec)
```

```
mysql> describe postagens;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id   | int  | NO   | PRI | NULL    | auto_increment |
| titulo | varchar(255) | YES | YES | NULL    | |
| conteudo | text | YES | YES | NULL    | |
| createdAt | datetime | NO | NO | NULL    | |
| updatedAt | datetime | NO | NO | NULL    | |
+-----+-----+-----+-----+-----+
5 rows in set (0.17 sec)

mysql> describe usuarios;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id   | int  | NO   | PRI | NULL    | auto_increment |
| nome | varchar(255) | YES | YES | NULL    | |
| sobrenome | varchar(255) | YES | YES | NULL    | |
| idade | int  | YES | YES | NULL    | |
| email | varchar(255) | YES | YES | NULL    | |
| createdAt | datetime | NO | NO | NULL    | |
| updatedAt | datetime | NO | NO | NULL    | |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

Inserindo dados na tabela postagens com o Sequelize

app\insere_dados_postagens.js

```
const Sequelize = require('sequelize');
const sequelize = new Sequelize('test', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

sequelize.authenticate().then(function(){
  console.log("Conectado com sucesso!");
}).catch(function(error){
  console.log("Falha ao se conectar: " + error);
});

const Postagem = sequelize.define('postagens', {
  titulo: {
    type: Sequelize.STRING
  },
  conteudo: {
    type: Sequelize.TEXT
  }
});

Postagem.create({
  titulo: "Título da primeira postagem",
  conteudo: "Descrição ou conteúdo da primeira postagem"
});
```

No terminal, na pasta de trabalho, entre com o comando:

`node insere_dados_postagens.js`

```
C:\guia_programador\nodejs\app>node insere_dados_postagens.js
Executing (default): SELECT 1+1 AS result
Executing (default): INSERT INTO `postagens` (`id`, `titulo`, `conteudo`, `createdAt`, `updatedAt`) VALUES (DEFAULT,?,?,?);
Conectado com sucesso!
```

```
mysql> select * from postagens;
+----+-----+-----+-----+-----+
| id | titulo | conteudo | createdAt | updatedAt |
+----+-----+-----+-----+-----+
| 1 | Título da primeira postagem | Descrição ou conteúdo da primeira postagem | 2021-04-28 15:53:06 | 2021-04-28 15:53:06 |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Inserindo dados na tabela usuarios com o Sequelize

app\insere_dados_usuarios.js

```
const Sequelize = require('sequelize');
const sequelize = new Sequelize('test', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

sequelize.authenticate().then(function(){
  console.log("Conectado com sucesso!");
}).catch(function(erro){
  console.log("Falha ao se conectar: " + erro);
});

const Usuario = sequelize.define('usuarios', {
  nome: {
    type: Sequelize.STRING
  },
  sobrenome: {
    type: Sequelize.STRING
  },
  idade: {
    type: Sequelize.INTEGER
  },
  email: {
    type: Sequelize.STRING
  }
});

Usuario.create({
  nome: "Victor",
  sobrenome: "Lima",
  idade: 20,
  email: 'victor_lima@gmail.com'
});

Usuario.create({
  nome: "Roberto",
  sobrenome: "Pinheiro",
  idade: 58,
  email: 'roberto@gmail.com'
});

Usuario.create({
  nome: "Ana Regina",
  sobrenome: "Lopes",
  idade: 38,
  email: 'ana_lopes@gmail.com'
});
```

No terminal, na pasta de trabalho, entre com o comando:

`node insere_dados_usuarios.js`

```
C:\guia_programador\nodejs\app>node insere_dados_usuarios.js
Executing (default): SELECT 1+1 AS result
Executing (default): INSERT INTO `usuarios` (`id`, `nome`, `sobrenome`, `idade`, `email`, `createdAt`, `updatedAt`) VALUES (DEFAULT ,?, ?, ?, ?, ?, ?)
Conectado com sucesso!
Executing (default): INSERT INTO `usuarios` (`id`, `nome`, `sobrenome`, `idade`, `email`, `createdAt`, `updatedAt`) VALUES (DEFAULT ,?, ?, ?, ?, ?, ?)
Executing (default): INSERT INTO `usuarios` (`id`, `nome`, `sobrenome`, `idade`, `email`, `createdAt`, `updatedAt`) VALUES (DEFAULT ,?, ?, ?, ?, ?, ?);
```

```
mysql> select * from usuarios;
+----+-----+-----+-----+-----+-----+-----+
| id | nome | sobrenome | idade | email           | createdAt        | updatedAt        |
+----+-----+-----+-----+-----+-----+-----+
| 1  | Victor | Lima      | 20   | victor_lima@gmail.com | 2021-04-28 15:55:42 | 2021-04-28 15:55:42 |
| 2  | Roberto | Pinheiro   | 58   | roberto@gmail.com    | 2021-04-28 15:55:42 | 2021-04-28 15:55:42 |
| 3  | Ana Regina | Lopes     | 38   | Ana_lopes@gmail.com | 2021-04-28 15:55:42 | 2021-04-28 15:55:42 |
+----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Aula 19 - Handlebars

Nesta aula você vai aprender a como criar um projeto Node.js utilizando a template engine Handlebars, que serve para exibir informações do back-end de uma forma robusta no HTML.

O Handlebars é um template engine. Ele fornece muitas funcionalidades ao HTML.

Instalação e configuração do Handlebar

Na pasta do projeto, entre com o comando:

`npm install express-handlebars --save`

```
C:\guia_programador\nodejs\app>npm install express-handlebars --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\app\package.json'
npm ERR! enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\app\package.json'
npm ERR! app No description
npm ERR! app No repository field.
npm ERR! app No README data
npm ERR! app No license field.

+ express-handlebars@5.3.0
added 18 packages from 43 contributors and audited 148 packages in 15.824s

1 package is looking for funding
  run `npm fund` for details

Found 0 vulnerabilities
```

app\index.js

```
const express = require("express");
const app = express();
const handlebars = require('express-handlebars');
const Sequelize = require('sequelize');

// Config
// Template Engine
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Conexão com o banco de dados MySQL
const sequelize = new Sequelize('test', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

app/views/layouts/main.handlebars

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Postagens Node.JS</title>
</head>
<body>
  {{body}}
</body>
</html>
```

Aula 20 - Como criar um formulário

Nesta aula você vai aprender a como criar um formulários com Node.js para poder fazer o envio de dados para o seu back-end, junto com isso vamos desenvolver a nossa aplicação prática com Javascript e Node.js

app\index.js

```
const express = require("express");
const app = express();
const handlebars = require('express-handlebars');
const Sequelize = require('sequelize');

// Config

// Template Engine
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Conexão com o banco de dados MySQL
const sequelize = new Sequelize('test', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

// Rotas
app.get('/cad', function(req, res){
  res.send('ROTA DE CADASTRO DE POSTS');
});

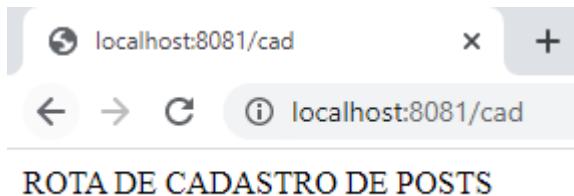
app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

- No terminal:

nodemon index.js

```
C:\guias_programador\nodejs\app>nodemon index.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Servidor rodando na URL: http://localhost:8081
```

- No browser:



app\views\formulario.handlebars

```
<div class="container mt-5">
  <h1 class="text-center">Cadastro de Postagens</h1>

  <form action="" method="">
    <div class="form-group">
      <label for="titulo">Título: </label>
      <input class="form-control" type="text" name="titulo">
    </div>

    <div class="form-group">
      <label for="conteudo">Conteúdo: </label>
      <textarea class="form-control" name="conteudo" id="conteudo" cols="30" rows="10"></textarea>
    </div>

    <button class="btn btn-success" type="submit">Cadastrar</button>
  </form>
</div>
```

app\index.js

```
const express = require("express");
const app = express();
const handlebars = require('express-handlebars');
const Sequelize = require('sequelize');

// Config
// Template Engine
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Conexão com o banco de dados MySQL
const sequelize = new Sequelize('test', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

// Rotas
app.get('/cad', function(req, res){
  res.send('ROTA DE CADASTRO DE POSTS');
});

app.get('/create_post', function(req, res){
  res.render('formulario');
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

app\views\layouts\main.handlebars

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Postagens Node.JS</title>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
  {{body}}
</body>
</html>
```

- Na pasta de projetos do terminal:

node index.js

- No browser:

http://localhost:8081/create_post

The screenshot shows a web browser window titled "Postagens NodeJS". The address bar displays "localhost:8081/create_post". The main content area is titled "Cadastro de Postagens". It contains two input fields: one for "Título" (Title) and one for "Conteúdo" (Content). A green button labeled "Cadastrar" (Create) is located at the bottom left of the form.

Cadastro de Postagens

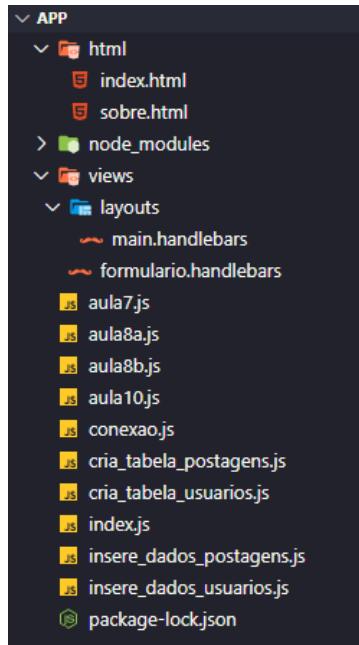
Título:

Conteúdo:

Cadastrar

Aula 21 - Como enviar dados do formulário

Nesta aula você vai entender como enviar dados de um formulário para o seu back-end Node.js, com isso você vai poder receber os dados enviados pelo usuário por um formulário em seu sistema Node.js feito com Javascript.



app\views\formulario.handlebars

```
<div class="container mt-5">
  <h1 class="text-center">Cadastro de Postagens</h1>

  <form action="/store_post" method="POST">
    <div class="form-group">
      <label for="titulo">Título:</label>
      <input class="form-control" type="text" name="titulo">
    </div>

    <div class="form-group">
      <label for="conteudo">Conteúdo:</label>
      <textarea class="form-control" name="conteudo" id="conteudo" cols="30"
rows="10"></textarea>
    </div>

    <button class="btn btn-success" type="submit">Cadastrar</button>
  </form>
</div>
```

app/index.js

```
const express = require("express");
const app = express();
const handlebars = require('express-handlebars');
const Sequelize = require('sequelize');

// Config
// Template Engine
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Conexão com o banco de dados MySQL
const sequelize = new Sequelize('test', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

// Rotas
app.get('/cad', function(req, res){
  res.send('ROTA DE CADASTRO DE POSTS');
});

app.get('/create_post', function(req, res){
  res.render('formulario');
});

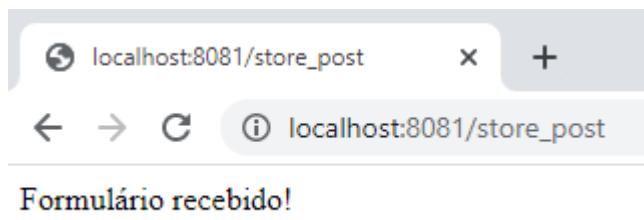
app.post('/store_post', function(req, res){
  res.send('Formulário recebido!');
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

http://localhost:8081/create_post

The screenshot shows a web browser window with the title bar "Postagens Node.js". The address bar displays the URL "localhost:8081/create_post". The main content area is titled "Cadastro de Postagens". It contains two input fields: "Título:" with the value "Primeira postagem" and "Conteúdo:" with the value "Olá mundo!". Below the inputs is a green button labeled "Cadastrar".

Clicando no botão Cadastrar:



Aula 22 - Body parser

Nesta aula você vai ser apresentando a biblioteca do Node.js **Body parser**, que serve para permitir com que clientes externos possam enviar informação para sua aplicação Node.js feita com Javascript, com o body parser você vai poder receber dados de formulários, e/ou dados XML e Json de requisições HTTP.

Instalando o Body parser

Na pasta do seu projeto, entre com o comando:

```
npm install body-parser --save
```

```
C:\guia_programador\nodejs\app>npm install body-parser --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\app\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\app\package.json'
npm WARN app No description
npm WARN app No repository field.
npm WARN app No README data
npm WARN app No license field.

+ body-parser@1.19.0
updated 1 package and audited 163 packages in 4.804s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

app\index.js

```
const express = require("express");
const app = express();
const handlebars = require('express-handlebars');
const bodyParser = require('body-parser');
const Sequelize = require('sequelize');

// Config

// Template Engine
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Body Parser
app.use(bodyParser.urlencoded({extend:false}));
app.use(bodyParser.json());

// Conexão com o banco de dados MySQL
const sequelize = new Sequelize('test', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

// Rotas
app.get('/create_post', function(req, res){
  res.render('formulario');
});

app.post('/store_post', function(req, res){
  res.send("Título: " + req.body.titulo + "<br>Conteúdo: " + req.body.conteudo);
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

nodemon index.js

```
C:\guia_programador\nodejs\app>nodemon index.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
body-parser deprecated undefined extended: provide extended option index.js:14:20
Servidor rodando na URL: http://localhost:8081
```

Postagens Node.js

localhost:8081/create_post

Cadastro de Postagens

Título:

Conteúdo:

Olá mundo!

Cadastrar

- Clique no botão Cadastrar:

localhost:8081/store_post

Título: Primeira postagem
Conteúdo: Olá mundo!

Aula 23 - Estruturando o banco de dados

Nesta aula vamos estruturar nosso banco de dados para começar a criar nossa primeira aplicação Node.js

Criando o banco de dados da aplicação

mysql -h localhost -u root -p

```
C:\Users\beto1>mysql -h localhost -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 8.0.24 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

- Crie o banco de dados com nome **postapp**:

create database postapp;

```
mysql> create database postapp;
Query OK, 1 row affected (1.21 sec)
```

show databases;

```
mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| postapp        |
| sistema_cadastro |
| sys            |
| test           |
+-----+
7 rows in set (3.01 sec)
```

- Selecione o banco de dados **postapp**:

```
use postapp;
```

```
mysql> use postapp;
Database changed
```

```
show tables;
```

```
mysql> use postapp;
Database changed
mysql> show tables;
Empty set (2.13 sec)
```

Criando um novo projeto chamado postapp

- Na pasta **C:\guia_programador\nodejs** crie uma pasta chamada **postapp**
- Dentro desta pasta iremos criar o nosso aplicativo, do zero.

```
C:\guia_programador\nodejs>dir
0 volume na unidade C é W10-195
0 Número de Série do Volume é 6047-7D0C

Pasta de C:\guia_programador\nodejs

30/04/2021 10:16    <DIR>        .
30/04/2021 10:16    <DIR>        ..
30/04/2021 10:07    <DIR>        app
30/04/2021 08:40    <DIR>        basico
30/04/2021 10:16    <DIR>        postapp
              0 arquivo(s)          0 bytes
              5 pasta(s)   114.890.706.944 bytes disponíveis
```

```
cd postapp
code .
```

```
C:\guia_programador\nodejs>cd postapp
C:\guia_programador\nodejs\postapp>code .
```

Instalando as dependências do projeto

- Dentro da pasta **postapp** instale os seguintes pacotes:

npm install express --save

```
C:\guia_programador\nodejs\postapp>npm install express --save
npm [WARN] saveError: ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\postapp\package.json'
npm [notice] created a lockfile as package-lock.json. You should commit this file.
npm [WARN] enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\postapp\package.json'
npm [WARN] postapp No description
npm [WARN] postapp No repository field.
npm [WARN] postapp No README data
npm [WARN] postapp No license field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 14.532s
found 0 vulnerabilities
```

npm install sequelize --save

```
C:\guia_programador\nodejs\postapp>npm install sequelize --save
npm [WARN] saveError: ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\postapp\package.json'
npm [WARN] enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\postapp\package.json'
npm [WARN] postapp No description
npm [WARN] postapp No repository field.
npm [WARN] postapp No README data
npm [WARN] postapp No license field.

+ sequelize@6.6.2
added 19 packages from 80 contributors and audited 107 packages in 14.418s
found 0 vulnerabilities
```

npm install mysql2 --save

```
C:\guia_programador\nodejs\postapp>npm install mysql2 --save
npm [WARN] saveError: ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\postapp\package.json'
npm [WARN] enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\postapp\package.json'
npm [WARN] postapp No description
npm [WARN] postapp No repository field.
npm [WARN] postapp No README data
npm [WARN] postapp No license field.

+ mysql2@2.2.5
added 12 packages from 18 contributors and audited 125 packages in 4.816s
found 0 vulnerabilities
```

npm install body-parser --save

```
C:\guia_programador\nodejs\postapp>npm install body-parser --save
npm [WARN] saveError: ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\postapp\package.json'
npm [WARN] enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\postapp\package.json'
npm [WARN] postapp No description
npm [WARN] postapp No repository field.
npm [WARN] postapp No README data
npm [WARN] postapp No license field.

+ body-parser@1.19.0
updated 1 package and audited 130 packages in 3.123s
found 0 vulnerabilities
```

```
npm install express-handlebars --save
```

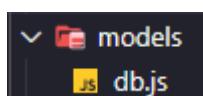
```
C:\guia_programador\nodejs\postapp>npm install express-handlebars --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\postapp\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\postapp\package.json'
npm WARN postapp No description
npm WARN postapp No repository field.
npm WARN postapp No README data
npm WARN postapp No license field.

+ express-handlebars@5.3.0
added 18 packages from 43 contributors and audited 148 packages in 8.134s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

- Na pasta raiz da aplicação, crie uma pasta com o nome de **models**.
- Dentro dela, crie o arquivo **db.js**.



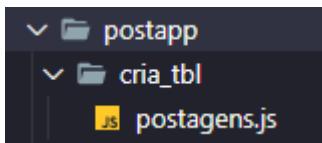
postapp\models\db.js

```
const Sequelize = require('sequelize');
const sequelize = new Sequelize('postapp', 'root', '123456', {
  host: "localhost",
  dialect: "mysql"
});

sequelize.authenticate().then(function(){
  console.log("Conectado com sucesso!");
}).catch(function(erro){
  console.log("Falha ao se conectar: " + erro);
});

module.exports = {
  Sequelize: Sequelize,
  sequelize: sequelize
};
```

- No diretório raiz da aplicação, crie uma pasta chamada **cria_tbl**.
- Dentro dela adicione o arquivo **postagens.js**



Criando a tabela postagens

- No diretório raiz da aplicação, execute o comando:

`node cria_tbl\postagens.js`

```
C:\guia_programador\nodejs\postapp>node cria_tbl\postagens.js
Executing (default): SELECT 1+1 AS result
Executing (default): DROP TABLE IF EXISTS `postagens`;
Conectado com sucesso!
Executing (default): CREATE TABLE IF NOT EXISTS `postagens` (`id` INTEGER NOT NULL auto_increment , `titulo` VARCHAR(255) , `conteudo` TEXT , `createdAt` DATETIME NOT NULL , `updatedAt` DATETIME NOT NULL , PRIMARY KEY (`id`)) ENGINE=InnoDB;
Executing (default): SHOW INDEX FROM `postagens`
```

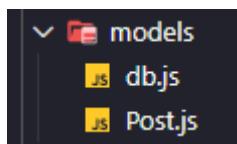
`show tables;`

```
mysql> show tables;
+-----+
| Tables_in_postapp |
+-----+
| postagens         |
+-----+
1 row in set (0.02 sec)
```

`describe postagens;`

```
mysql> describe postagens;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| id    | int    | NO  | PRI | NULL    | auto_increment |
| titulo | varchar(255) | YES | PRI | NULL    |                |
| conteudo | text   | YES |     | NULL    |                |
| createdAt | datetime | NO  |     | NULL    |                |
| updatedAt | datetime | NO  |     | NULL    |                |
+-----+-----+-----+-----+-----+
5 rows in set (0.12 sec)
```

- Dentro da pasta **models**, crie o arquivo **Post.js**.



postapp/models/Post.js

```
const db = require('./db');

const Post = db.sequelize.define('postagens', {
  titulo: {
    type: db.Sequelize.STRING
  },
  conteudo: {
    type: db.Sequelize.TEXT
  }
});

module.exports = Post;
```

Aula 24 - Pegando dados do formulário

Nesta aula vamos entender como pegar dados de um formulário em nossa aplicação Node.js feita com MySQL e Sequelize com ajuda da biblioteca body parser.

postapp\index.js

```
const express = require("express");
const app = express();
const handlebars = require('express-handlebars');
const bodyParser = require('body-parser');
const Sequelize = require('sequelize');
const Post = require('./models/Post');

// Config

// Template Engine
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Body Parser
app.use(bodyParser.urlencoded({extend:false}));
app.use(bodyParser.json());

// Rotas

// create
app.get('/create_post', function(req, res){
  res.render('formulario');
});

// store
app.post('/store_post', function(req, res){
  Post.create({
    titulo: req.body.titulo,
    conteudo: req.body.conteudo
  }).then(function(){
    res.send('Post criado com sucesso!');
  }).catch(function(erro){
    res.send('Erro ao criar o post: ' + erro);
  });
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

- Na pasta do projeto, entre com o comando:

```
nodemon index.js
```

```
C:\guia_programador\nodejs\postapp>nodemon index.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
body-parser deprecated undefined extended: provide extended option index.js:14:20
Servidor rodando na URL: http://localhost:8081
Executing (default): SELECT 1+1 AS result
Conectado com sucesso!
```

- Cadastre uma postagem:

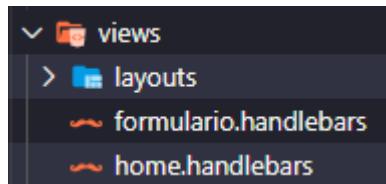
```
localhost:8081/create_post
```

The screenshot shows a web browser window titled "Postagens NodeJS". The address bar displays "localhost:8081/create_post". The main content is a form titled "Cadastro de Postagens". It has two input fields: "Título" containing "Primeira postagem" and "Conteúdo" containing "Este é o texto da minha primeira postagem!". A green "Cadastrar" button is at the bottom. Below the browser window, a smaller message box shows the confirmation: "Post criado com sucesso!".

```
mysql> select * from postagens;
+----+-----+-----+-----+
| id | titulo | conteudo | createdAt | updatedAt |
+----+-----+-----+-----+
| 1  | Primeira postagem | Esse é o texto da primeira postagem. | 2021-04-29 08:02:53 | 2021-04-29 08:02:53 |
+----+-----+-----+-----+
1 row in set (0.00 sec)
```

Redirecionamento para uma página após o cadastro

- Na pasta `views`, adicione um arquivo chamado `home.handlebars` e outro chamado `formulario.handlebars`.



`postapp\views\home.handlebars`

```
<h1>Lista de posts: </h1>
```

`postapp\views\formulario.handlebars`

```
<div class="container mt-5">
<h1 class="text-center">Cadastro de Postagens</h1>

<form action="/store_post" method="POST">
<div class="form-group">
<label for="titulo">Título: </label>
<input class="form-control" type="text" name="titulo">
</div>

<div class="form-group">
<label for="conteudo">Conteúdo: </label>
<textarea class="form-control" name="conteudo" id="conteudo" cols="30" rows="10"></textarea>
</div>

<button class="btn btn-success" type="submit">Cadastrar</button>
</form>
</div>
```

- Altere o arquivo **postapp\index.js**

postapp\index.js

```
const express = require("express");
const app = express();
const handlebars = require('express-handlebars');
const bodyParser = require('body-parser');
const Post = require('./models/Post');

// Config

// Template Engine
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Body Parser
app.use(bodyParser.urlencoded({extend:false}));
app.use(bodyParser.json());

// Rotas

app.get('/', function(req, res){
  res.render('home');
});

app.get('/create_post', function(req, res){
  res.render('formulario');
});

app.post('/store_post', function(req, res){
  Post.create({
    titulo: req.body.titulo,
    conteudo: req.body.conteudo
  }).then(function(){
    // res.send('Post criado com sucesso!');
    res.redirect('/');
  }).catch(function(erro){
    res.send('Erro ao criar o post: ' + erro);
  });
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

- No terminal, na pasta do projeto, execute o comando:

nodemon index.js

- No browser, cadastre outra postagem:

http://localhost:8081/create_post

Cadastro de Postagens

Título:

Conteúdo:

Este é o texto da segunda postagem|

Cadastrar

Postagens NodeJS

localhost:8081

Lista de posts:

- 1 | Primeira postagem | Esse é o texto da primeira postagem. | 2021-04-29 08:02:53 | 2021-04-29 08:02:53 |
- 2 | Segunda postagem | Este é o texto da segunda postagem. | 2021-04-29 08:25:26 | 2021-04-29 08:25:26 |

```
mysql> select * from postagens;
+----+-----+-----+-----+-----+
| id | titulo | conteudo          | createdAt        | updatedAt        |
+----+-----+-----+-----+-----+
| 1  | Primeira postagem | Esse é o texto da primeira postagem. | 2021-04-29 08:02:53 | 2021-04-29 08:02:53 |
| 2  | Segunda postagem | Este é o texto da segunda postagem. | 2021-04-29 08:25:26 | 2021-04-29 08:25:26 |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Aula 25 - Listando dados do banco

Nessa aula continuaremos a desenvolver a nossa aplicação com Express.js, Node.js e Sequelize, desta vez você vai aprender a como listar dados do MySQL no Node.js

postapp\index.js

```
const express = require("express");
const app = express();
const handlebars = require('express-handlebars');
const bodyParser = require('body-parser');
const Sequelize = require('sequelize');
const Post = require('./models/Post');

// Config

// Template Engine
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Body Parser
app.use(bodyParser.urlencoded({extend:false}));
app.use(bodyParser.json());

// Rotas

app.get('/', function(req, res){
  Post.findAll({order: [['id', 'desc']]})�then(function(posts){
    res.render('home', {posts: posts});
  });
});

app.get('/create_post', function(req, res){
  res.render('formulario');
});

app.post('/store_post', function(req, res){
  Post.create({
    titulo: req.body.titulo,
    conteudo: req.body.conteudo
  })�then(function(){
    res.redirect('/');
  }).catch(function(error){
    res.send('Erro ao criar o post: ' + error);
  });
});

app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

views\layouts\main.handlebars

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Postagens Node.JS</title>
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
  {{body}}
</body>
</html>
```

views\home.handlebars

```
<h1>Lista de posts: </h1>

{{#each posts}}
<small>{{dataValues.createdAt}}</small>
<h2>{{dataValues.titulo}}</h2>
<p>{{dataValues.conteudo}}</p>
<hr>
{{/each}}
```

- Na pasta do projeto, entre com o comando:

`nodemon index.js`

- Cadastre uma terceira postagem:

`localhost:8081/create_post`

Postagens Node.js

localhost:8081/create_post

Cadastro de Postagens

Título:

Conteúdo:

Este é o texto da terceira postagem.

Cadastrar

localhost:8081



Lista de posts:

Fri Apr 30 2021 11:00:36 GMT-0300 (Horário Padrão de Brasília)

Terceira postagem

Este é o texto da terceira postagem.

Thu Apr 29 2021 05:25:26 GMT-0300 (Horário Padrão de Brasília)

Segunda postagem

Este é o texto da segunda postagem.

Thu Apr 29 2021 05:02:53 GMT-0300 (Horário Padrão de Brasília)

Primeira postagem

Esse é o texto da primeira postagem.

Aula 26 - Deletando postagens

Nesta aula continuaremos desenvolvendo nosso projeto com Node.js e MySQL, e dessa vez, vamos finalizar ele com essa aula que te ensina a como deletar registros com o sequelize.

postapp\index.js

```
const express = require("express");
const app = express();
const handlebars = require('express-handlebars');
const bodyParser = require('body-parser');
const Sequelize = require('sequelize');
const Post = require('./models/Post');

// Config
// Template Engine
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Body Parser
app.use(bodyParser.urlencoded({extend:false}));
app.use(bodyParser.json());

// Rotas
app.get('/', function(req, res){
  Post.findAll({order: [['id', 'desc']]})�then(function(posts){
    res.render('home', {posts: posts});
  });
});

app.get('/create_post', function(req, res){
  res.render('formulario');
});

app.post('/store_post', function(req, res){
  Post.create({
    titulo: req.body.titulo,
    conteudo: req.body.conteudo
  })�then(function(){
    res.redirect('/');
  }).catch(function(erro){
    res.send('Erro ao criar o post: ' + erro);
  });
});

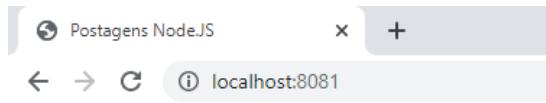
app.get('/deletar/:id', function(req, res){
  Post.destroy({where: {'id': req.params.id}})�then(function(){
    res.redirect('/');
  }).catch(function(erro){
    res.send("Esta postagem não existe!");
  });
});
```

```
app.listen(8081, function(){
  console.log("Servidor rodando na URL: http://localhost:8081");
});
```

views\home.handlebars

```
<h1>Lista de posts: </h1>

{{#each posts}}
<small>{{dataValues.createdAt}}</small>
<h2>{{dataValues.titulo}}</h2>
<p>{{dataValues.conteudo}}</p>
<a href="/deletar/{{dataValues.id}}"><button class="btn btn-danger">Deletar</button></a>
<hr>
{{/each}}
```



Lista de posts:

Fri Apr 30 2021 11:00:36 GMT-0300 (Horário Padrão de Brasília)

Terceira postagem

Este é o texto da terceira postagem.

[Deletar](#)

Thu Apr 29 2021 05:25:26 GMT-0300 (Horário Padrão de Brasília)

Segunda postagem

Este é o texto da segunda postagem.

[Deletar](#)

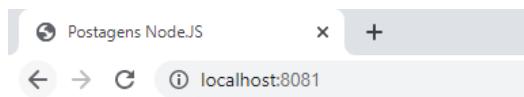
Thu Apr 29 2021 05:02:53 GMT-0300 (Horário Padrão de Brasília)

Primeira postagem

Esse é o texto da primeira postagem.

[Deletar](#)

- Clique no botão "**Deletar**" da terceira postagem:



Lista de posts:

Thu Apr 29 2021 05:25:26 GMT-0300 (Horário Padrão de Brasília)

Segunda postagem

Este é o texto da segunda postagem.

Deletar

Thu Apr 29 2021 05:02:53 GMT-0300 (Horário Padrão de Brasília)

Primeira postagem

Esse é o texto da primeira postagem.

Deletar

Aula 27 - Introdução ao MongoDB

Nesta aula vamos conhecer o sistema de banco de dados MongoDB, ou seja, vamos ver o que é mongodb e como podemos utilizar ele para desenvolver nossas aplicações web utilizando Node.js e Javascript.

- ▶ MongoDB é um banco de dados de código aberto, gratuito, de alta performance, sem esquemas e orientado à documentos

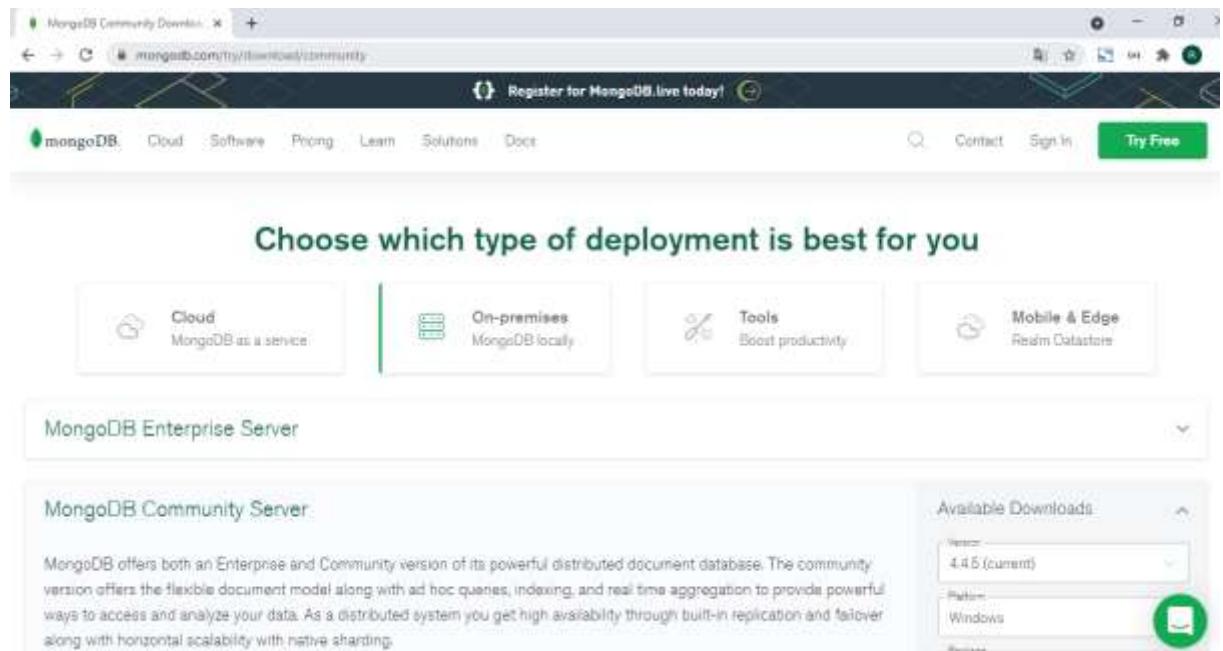
O MongoDB salva os dados em arquivos JSON e os agrupa em uma collection.

Aula 28 - Como instalar o MongoDB

Nesta aula vamos ver como instalar o MongoDB no sistema operacional windows, para que assim, podemos desenvolver nossas aplicações web back-end com node.js e javascript.

Instalação e configuração do MongoDB

<https://www.mongodb.com/download-center/community>



Choose which type of deployment is best for you

- Cloud MongoDB as a service
- On-premises MongoDB locally
- Tools Boost productivity
- Mobile & Edge Realm Datastore

MongoDB Enterprise Server

MongoDB Community Server

MongoDB offers both an Enterprise and Community version of its powerful distributed document database. The community version offers the flexible document model along with ad hoc queries, indexing, and real time aggregation to provide powerful ways to access and analyze your data. As a distributed system you get high availability through built-in replication and failover along with horizontal scalability with native sharding.

Available Downloads

- Version: 4.4.6 (current)
- Platform: Windows
- Package

- Faça o download do MongoDB Community Server e instale no computador.
- Para Windows 7 - 32 bits, acesse o link:

<https://www.mongodb.org/dl/win32/i386>

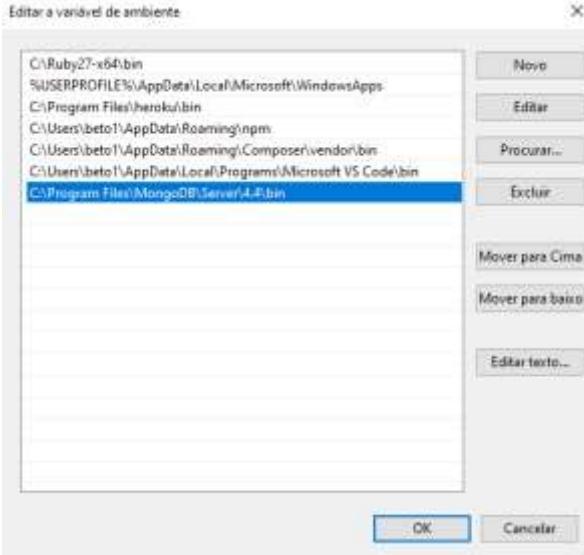
Configuração

- Crie a pasta `c:\data` e dentro dela a subpasta `db`

Importante: É dentro desta pasta `data/db` que serão salvos os arquivos criados pelo Mongo.

- Insira o caminho da pasta bin do MongoDB no `path` do Windows.

`C:\Program Files\MongoDB\Server\4.4\bin`



Abrindo o servidor do MongoDB

Sempre que for trabalhar com o MongoDB é necessário abrir o seu servidor. Para isso, entre no seu terminal com o seguinte comando:

mongod

Acessando o shell do MongoDB

- Abra um novo terminal (não feche o do servidor que está rodando) e entre com o comando:

mongo

```
Microsoft Windows [versao 10.0.19042.928]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\betol>mongo
MongoDB shell version v4.4.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b354681c-1e86-494e-92fb-33818cb9d586") }
MongoDB server version: 4.4.5
...
The server generated these startup warnings when booting:
    2021-04-30T04:07:24.205+03:00: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
...
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
>
```

Aula 29 - Iniciando com o Mongoose

Nesta aula você vai aprender a como trabalhar com o Mongoose que é uma biblioteca ODM(quase um ORM para bancos NoSQL) que serve para abstrair toda a conexão do Node.js com o MongoDB, para assim, podemos desenvolver aplicações web back-end com Node.js e Javascript

Instalação do Mongoose

- Dentro de `C:\guia_programador\nodejs` crie a pasta `testes_mongo`.
- Na pasta do projeto (`testes_mongo`), entre com o comando:

```
npm install mongoose --save
```

```
C:\guia_programador\nodejs\testes_mongo>npm install mongoose --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\testes_mongo\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\testes_mongo\package.json'
npm WARN testes_mongo No description
npm WARN testes_mongo No repository field.
npm WARN testes_mongo No README data
npm WARN testes_mongo No license field.

+ mongoose@5.12.7
added 34 packages from 92 contributors and audited 34 packages in 29.845s

2 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
```

Configurando o Mongoose

`show databases;`

```
C:\guia_programador\nodejs\testes_mongo>mongo
MongoDB shell version v4.4.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("fd4ca936-3cc9-40f6-a2bc-47aa0fe5d8b5") }
MongoDB server version: 4.4.5
```
The server generated these startup warnings when booting:
 2021-04-30T04:07:24.205-03:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
```
Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
```
> show databases;
admin 0.000GB
config 0.000GB
local 0.000GB
>
```

## **testesmongo\teste1.js**

```
const mongoose = require('mongoose')

// Configurando o mongoose

mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/testes_mongo").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});
```

Entre com o comando:

**node teste1.js**

```
C:\guia_programador\nodejs\testes_mongo>node teste1.js
(node:15336) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:15336) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoC
lient constructor.
Mongo conectado...
```

## Aula 30 - Trabalhando com o Mongoose

Nesta aula você vai continuar aprendendo a como trabalhar com o Mongoose que é uma biblioteca ODM (quase um ORM para bancos NoSQL) que serve para abstrair toda a conexão do Node.js com o MongoDB, para assim, podemos desenvolver aplicações web back-end com Node.js e Javascript

### Criando o model e cadastrando um usuário

**testesmongo\teste1.js**

```
const mongoose = require('mongoose')

// Configurando o mongoose

mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/testes_mongo").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});

// Model - User

const UserSchema = mongoose.Schema({
 nome: {
 type: String,
 require: true
 },
 sobrenome: {
 type: String,
 require: true
 },
 email: {
 type: String,
 require: true
 },
 idade: {
 type: Number,
 require: true
 },
 pais: {
 type: String
 }
})
```

```
// Collection

mongoose.model('users', UserSchema)

const user = mongoose.model('users')

new user ({
 nome: "Victor",
 sobrenome: "Lima",
 email: "victor_lima@gmail.com",
 idade: 19,
 pais: "Brasil"
}).save().then(() => {
 console.log("Usuário cadastrado com sucesso!")
}).catch((err) => {
 console.log("Houve um erro ao registrar o usuário: " + err)
})
```

node teste1.js

```
C:\guias_programador\nodejs\testes_mongo>node teste1.js
(node:15888) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:15888) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoC
lient constructor.
Mongo conectado...
Usuário cadastrado com sucesso!
```

No Mongo:

```
show.databases;
use testes_mongo;
show.collections;
db.users.find();
```

```
> show databases;
admin 0.000GB
config 0.000GB
local 0.000GB
testes_mongo 0.000GB
> use testes_mongo;
switched to db testes_mongo
> show collections;
users
> db.users.find();
{ "_id" : ObjectId("608cf415af0c313e088525c6"), "nome" : "Victor", "sobrenome" : "Lima", "email" : "victor_lima@gmail.co
m", "idade" : 19, "pais" : "Brasil", "__v" : 0 }
>
```

- Cadastre um novo usuário:
- Substitua no cadastro anterior, os dados de Victor Lima por:

```
new user ({
 nome: "Jhon",
 sobrenome: "Doe",
 email: "jhon@doe.com",
 idade: 34,
 pais: "EUA"
}).save().then(() => {
 console.log("Usuário cadastrado com sucesso!")
}).catch((err) => {
 console.log("Houve um erro ao registrar o usuário: " + err)
})
```

node teste1.js

```
C:\guia_programador\nodejs\testes_mongo>node teste1.js
(node:14728) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:14728) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoC
lient constructor.
Mongo conectado...
Usuário cadastrado com sucesso!
```

No Mongo:

```
db.users.find();
```

```
> db.users.find();
{ "_id" : ObjectId("608cf415af0c313e088525c6"), "nome" : "Victor", "sobrenome" : "Lima", "email" : "victor_lima@gmail.co
m", "idade" : 19, "pais" : "Brasil", "__v" : 0 }
{ "_id" : ObjectId("608cf524eb75263980690ae7"), "nome" : "Jhon", "sobrenome" : "Doe", "email" : "jhon@doe.com", "idade"
: 34, "pais" : "EUA", "__v" : 0 }
```

## Aula 31 - Iniciando projeto Express.js

Nesta aula vamos dar pontapé inicial no nosso principal projeto do curso que é a criação de um CRUD com MongoDB, Mongoose, Express.js, Node.js e Javascript.

### Instalando as dependências do projeto

- Em C:\guia\_programador\nodejs crie a pasta "blogapp". Essa será a pasta do novo projeto.

```
C:\guia_programador\nodejs>dir
O volume na unidade C é W10-195
O Número de Série do Volume é 6047-7D0C

Pasta de C:\guia_programador\nodejs

30/04/2021 17:55 <DIR> .
30/04/2021 17:55 <DIR> ..
30/04/2021 10:07 <DIR> app
30/04/2021 08:40 <DIR> basico
30/04/2021 17:55 <DIR> blogapp
30/04/2021 10:38 <DIR> postapp
01/05/2021 03:11 <DIR> testes_mongo
 0 arquivo(s) 0 bytes
 7 pasta(s) 114.600.468.480 bytes disponíveis
```

Dentro dela instale as seguintes dependências:

```
cd blogapp
code .
```

```
C:\guia_programador\nodejs>cd blogapp
C:\guia_programador\nodejs\blogapp>code .
```

```
npm install express --save
```

```
C:\guia_programador\nodejs\blogapp>npm install express --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm WARN blogapp No description
npm WARN blogapp No repository field.
npm WARN blogapp No README data
npm WARN blogapp No license field.

+ express@4.17.1
added 29 packages from 25 contributors and audited 155 packages in 17.274s

3 packages are looking for funding
 run `npm fund` for details

found 0 vulnerabilities
```

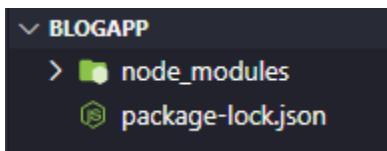
```
npm install express-handlebars --save
```

```
C:\guia_programador\nodejs\blogapp>npm install express-handlebars --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm WARN blogapp No description
npm WARN blogapp No repository field.
npm WARN blogapp No README data
npm WARN blogapp No license field.

+ express-handlebars@5.3.0
added 19 packages from 43 contributors and audited 19 packages in 13.339s

1 package is looking for funding
 run `npm fund` for details

found 0 vulnerabilities
```



```
npm install body-parser --save
```

```
C:\guia_programador\nodejs\blogapp>npm install body-parser --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm WARN blogapp No description
npm WARN blogapp No repository field.
npm WARN blogapp No README data
npm WARN blogapp No license field.

+ body-parser@1.19.0
added 22 packages from 17 contributors and audited 57 packages in 10.593s

1 package is looking for funding
 run `npm fund` for details

found 0 vulnerabilities
```

```
npm install mongoose --save
```

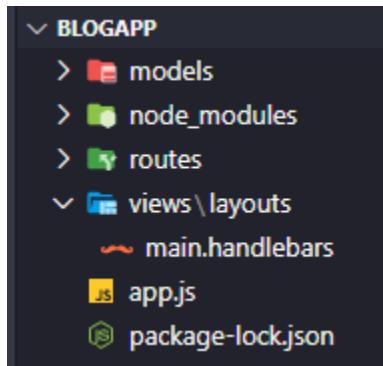
```
C:\guia_programador\nodejs\blogapp>npm install mongoose --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm WARN blogapp No description
npm WARN blogapp No repository field.
npm WARN blogapp No README data
npm WARN blogapp No license field.

+ mongoose@5.12.7
added 32 packages from 92 contributors and audited 109 packages in 13.562s

3 packages are looking for funding
 run `npm fund` for details

found 0 vulnerabilities
```

## Montando a estrutura do projeto



### views\layouts\main.handlebars

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta http-equiv="X-UA-Compatible" content="ie=edge">
 <title>Blog Node.JS</title>
</head>
<body>
 {{body}}
</body>
</html>
```

- Crie o arquivo app.js (arquivo principal)

### app.js

```
// Carregando módulos
const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
// const mongoose = require('mongoose')
const app = express()

// Configurações
// Body Parser
app.use(bodyParser.urlencoded({extend:false}));
app.use(bodyParser.json());

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Mongoose

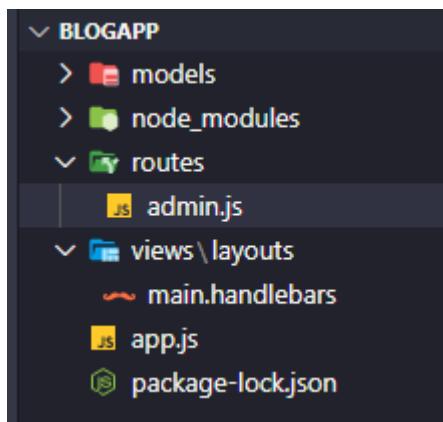
// Rotas

// Outros
const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

## Aula 32 - Grupo de rotas no Express.js

Nesta aula vamos ver como agrupar rotas no Express.js, para economizar linhas de código quando estivermos desenvolvendo aplicações web com Node.js e Javascript.



### routes\admin.js

```
const express = require('express')
const router = express.Router()

// Definindo as rotas

router.get('/', (req, res) => {
 res.send('Página principal do painel administrativo')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get('/categorias', (req, res) => {
 res.send('Página de categorias')
})

router.get('/teste', (req, res) => {
 res.send('Isso é um teste')
})

module.exports = router
```

## app.js

```
// Carregando módulos
const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
const app = express()
const admin = require('./routes/admin')
// const mongoose = require('mongoose')

// Configurações
// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose

// Rotas

app.get('/', (req, res) => {
 res.send('Rota principal')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)

// Outros
const PORT = '8081'

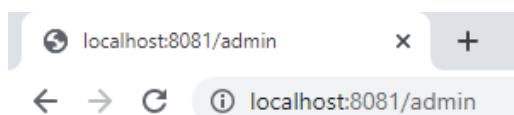
app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

nodemon app.js

```
C:\guia_programador\nodejs\blogapp>nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:11:22
Servidor rodando!
```

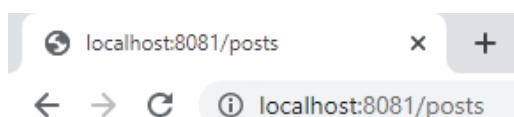
- No browser:

<http://localhost:8081/admin>



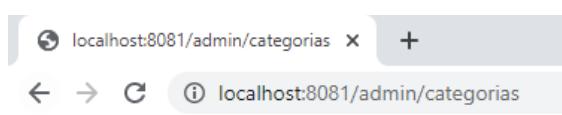
Página principal do painel administrativo

<http://localhost:8081/posts>



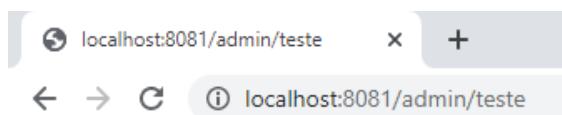
Lista de posts

<http://localhost:8081/admin/categorias>



Página de categorias

<http://localhost:8081/admin/teste>



Isso é um teste

## Aula 33 - Arquivos estáticos

Nesta aula você vai aprender a como trabalhar com arquivos estáticos no Node.js com o Express.js, esses arquivos estáticos são basicamente arquivos de imagem, html, css e js para o front-end.

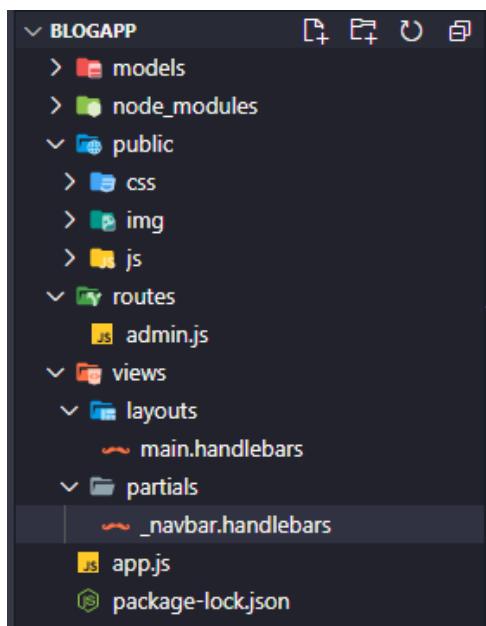
### Instalando o Bootstrap

- Acesse a URL:

<https://getbootstrap.com/docs/4.3/getting-started/download/>

E baixe os arquivos do bootstrap.

- Crie a pasta "**public**" e insira as pastas **css** e **js** do Bootstrap dentro dele.
- Crie dentro da pasta "**public**" a subpasta "**img**"
- Dentro da pasta **views**, crie a subpasta **partials** e dentro dela crie o arquivo **\_navbar.handlebars**.



## app.js

```
// Carregando módulos
const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
const app = express()
const admin = require('./routes/admin')
// const path = require('path')
// const mongoose = require('mongoose')

// Configurações
// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose

// Public
app.use(express.static(path.join(__dirname, "public")))

// Rotas

app.get('/', (req, res) => {
 res.send('Rota principal')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)

// Outros
const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

## **views\layouts\main.handlebars**

```
<!DOCTYPE html>
<html lang="pt-br">

<head>
 <link rel="stylesheet" href="/css/bootstrap.css">
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta http-equiv="X-UA-Compatible" content="ie=edge">
 <title>Blog Node.JS</title>
</head>

<body>
 {{body}}

 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
 integrity="sha384-q8i/X+965DzOOrT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
 crossorigin="anonymous"></script>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
 integrity="sha384-ZMP7rVo3mlykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
 crossorigin="anonymous"></script>
 <script src="/js/bootstrap.js"></script>
</body>

</html>
```

## **views\admin\index.handlebars**

```
<button class="btn btn-success">Teste</button>
```

## routes\admin.js

```
const express = require('express')
const router = express.Router()

// Definindo as rotas

router.get('/', (req, res) => {
 // res.send('Página principal do painel administrativo')
 res.render("admin/index")
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get('/categorias', (req, res) => {
 res.send('Página de categorias')
})

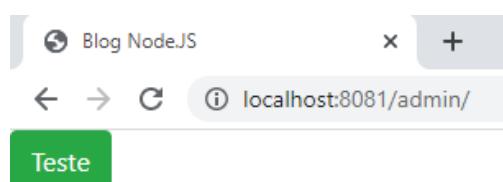
router.get('/teste', (req, res) => {
 res.send('Isso é um teste')
})

module.exports = router
```

## nodemon app.js

```
C:\guia_programador\nodejs\blogapp>nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:12:22
Servidor rodando!
```

<http://localhost:8081/admin/>



## Trabalhando com partials

- No site do **bootstrap**, em **Navbar**, copie o código do cabeçalho para o arquivo abaixo:

**views\partials\\_navbar.handlebars**

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
 Navbar
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

 </button>

 <div class="collapse navbar-collapse" id="navbarSupportedContent">
 <ul class="navbar-nav mr-auto">
 <li class="nav-item active">
 Home (current)

 <li class="nav-item">
 Link

 <li class="nav-item dropdown">

 Dropdown

 <div class="dropdown-menu" aria-labelledby="navbarDropdown">
 Action
 Another action
 <div class="dropdown-divider"></div>
 Something else here
 </div>

 <li class="nav-item">
 Disabled

 <form class="form-inline my-2 my-lg-0">
 <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
 <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
 </form>
 </div>
</nav>
```

## views\layouts\main.handlebars

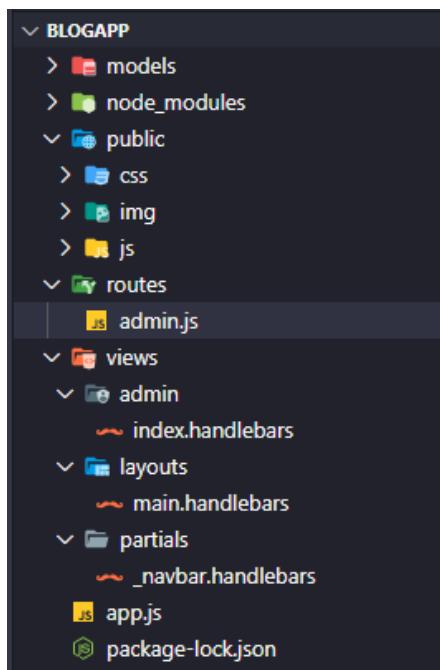
```
<!DOCTYPE html>
<html lang="pt-br">

<head>
 <link rel="stylesheet" href="/css/bootstrap.css">
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta http-equiv="X-UA-Compatible" content="ie=edge">
 <title>Blog Node.JS</title>
</head>

<body>
 {{> _navbar}}
 {{body}}

 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
 integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
 crossorigin="anonymous"></script>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
 integrity="sha384-ZMP7rVo3mlykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
 crossorigin="anonymous"></script>
 <script src="/js/bootstrap.js"></script>
</body>

</html>
```



nodemon app.js

```
C:\guia_programador\nodejs\blogapp>nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:12:22
Servidor rodando!
```



## Aula 34 - Formulário de categorias

Nessa aula vamos continuar desenvolvendo nossa sistema, agora dessa vez desenvolvendo alguns formulários com Node.js e Express.js

views\partials\\_navbar.handlebars

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
 Navbar
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
 aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

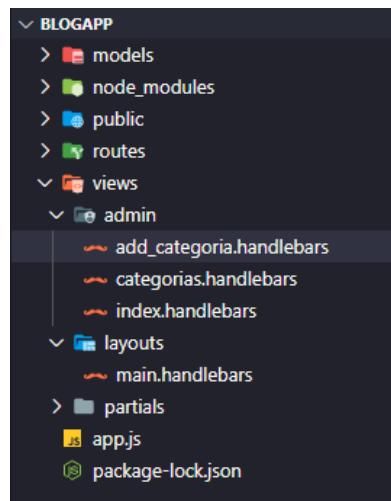
 </button>

 <div class="collapse navbar-collapse" id="navbarSupportedContent">
 <ul class="navbar-nav mr-auto">
 <li class="nav-item active">
 Home (current)

 <li class="nav-item">
 Link

 <li class="nav-item">
 Disabled

 </div>
</nav>
```



## views\layouts\main.handlebars

```
<!DOCTYPE html>
<html lang="pt-br">

 <head>
 <link rel="stylesheet" href="/css/bootstrap.css">
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta http-equiv="X-UA-Compatible" content="ie=edge">
 <title>Blog Node.JS</title>
 </head>

 <body>
 {{>_navbar}}

 <div class="container mt-4">
 {{>body}}
 </div>

 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
 integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
 crossorigin="anonymous"></script>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
 integrity="sha384-ZMP7rVo3mlykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
 crossorigin="anonymous"></script>
 <script src="/js/bootstrap.js"></script>
 </body>

</html>
```

## nodemon app.js

```
C:\guia_programador\nodejs\blogapp>nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:12:22
Servidor rodando!
```

- No browser:

<http://localhost:8081/admin/>



### routes\admin.js

```
const express = require('express')
const router = express.Router()

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get('/categorias', (req, res) => {
 res.render('admin/categorias')
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

module.exports = router
```

### views\admin\categorias.handlebars

```
<h2>Lista de categorias:</h2>
<hr>
<button class="btn btn-success">Nova categoria</button>
```

## views\admin\add\_categoria.handlebars

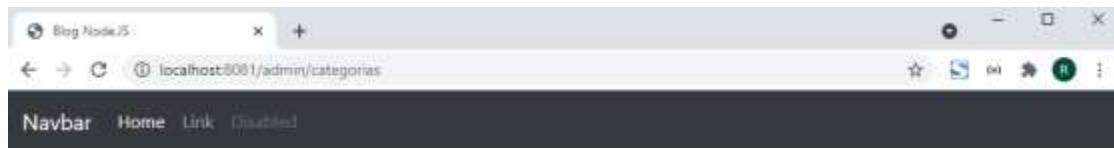
```
<h3>Nova categoria</h3>

<div class="card">
 <div class="card-body">
 <form action="/admin/categorias/nova" method="POST">
 <label for="nome">Nome: </label>
 <input type="text" id="nome" name="nome" placeholder="Nome da categoria" class="form-control">

 <label for="slug">Slug: </label>
 <input type="text" id="slug" name="slug" placeholder="Slug da categoria" class="form-control">

 <button type="submit" class="btn btn-success mt-4">Criar categoria</button>
 </form>
 </div>
</div>
```

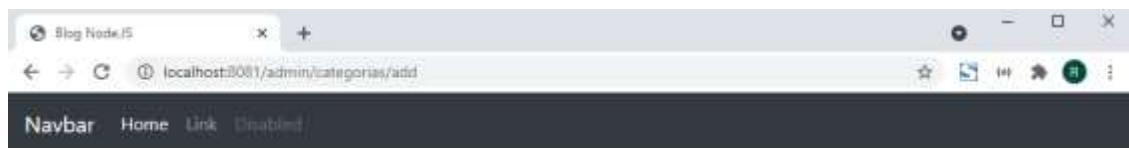
<http://localhost:8081/admin/categorias>



### Lista de categorias:

[Nova categoria](#)

<http://localhost:8081/admin/categorias/add>



### Nova categoria

Nome:

Nome da categoria

Slug:

Slug da categoria

[Criar categoria](#)

## Aula 35 - Definindo o model de categorias

Nesta aula vamos continuar desenvolvendo nossa sistema com Node.js e Express.js

- Abra o servidor do Mongo

**mongod**

### app.js

```
/* Carregando módulos */

const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
const app = express()
const admin = require('./routes/admin')
const path = require('path')
const mongoose = require('mongoose')

/* Configurações */

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/blogapp").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});

// Public
app.use(express.static(path.join(__dirname, "public")))

/* Rotas */

app.get('/', (req, res) => {
 res.send('Página principal')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)
```

```

/* Outros */

const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})

```

## nodemon app.js

```

C:\guia_programador\nodejs\blogapp>nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,jsx,json
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:14:22
(node:15664) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:15664) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
Servidor rodando!
Mongo conectado...

```

## Model\Categoria.js

```

const mongoose = require('mongoose')
const Schema = mongoose.Schema

// Model - Categoria

const Categoria = new Schema({
 nome: {
 type: String,
 required: true
 },
 slug: {
 type: String,
 required: true
 },
 date: {
 type: Date,
 default: Date.now()
 }
})

// Collection

mongoose.model('categorias', Categoria)

```

## Aula 36 - Cadastrando categorias no BD

Nesta aula vamos ver como cadastrar novas categorias no banco de dados da aplicação Node.js com MongoDB.

### routes\admin.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Categoria')
const Categoria = mongoose.model("categorias")

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get('/categorias', (req, res) => {
 res.render('admin/categorias')
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', (req, res) => {
 const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
 }

 new Categoria(novaCategoria).save().then(() => {
 console.log('Categoria salva com sucesso!')
 }).catch((err) => {
 console.log('Houve um erro ao salvar categoria: ' + err)
 })

 res.render('admin/categorias')
})

module.exports = router
```

```
C:\cader
mongod --storageEngine=mmapv1 --dbpath c:/data/db
2019-08-05T12:00:54.117-0300 I CONTROL [main]
2019-08-05T12:00:54.117-0300 I CONTROL [main] 32-bit servers don't have journaling enabled by default. Please use --journal if you want durability.
2019-08-05T12:00:54.118-0300 I CONTROL [main]
2019-08-05T12:00:54.127-0300 I CONTROL [main] Hotfix KB2731288 or later update is not installed, will zero-out data files
2019-08-05T12:00:54.133-0300 I CONTROL [initandlisten] MongoDB starting : pid=18888 port=27017 dbpath=c:/data/db 32-bit host=Beta-PC
2019-08-05T12:00:54.133-0300 I CONTROL [initandlisten] targetOS: Windows Vista/Windows Server 2008
2019-08-05T12:00:54.134-0300 I CONTROL [initandlisten] db version v3.2.22-rc0
2019-08-05T12:00:54.134-0300 I CONTROL [initandlisten] git version: 165acca8d043f0a47cia5bd608fd7133840a58dd
2019-08-05T12:00:54.135-0300 I CONTROL [initandlisten] allocator: tcmalloc
2019-08-05T12:00:54.135-0300 I CONTROL [initandlisten] modules: none
2019-08-05T12:00:54.135-0300 I CONTROL [initandlisten] build environment:
2019-08-05T12:00:54.135-0300 I CONTROL [initandlisten] distarch: i386
2019-08-05T12:00:54.136-0300 I CONTROL [initandlisten] target_arch: i386
2019-08-05T12:00:54.136-0300 I CONTROL [initandlisten] options: { storage: { dbPath: "c:/data/db", engine: "mmapv1" } }
2019-08-05T12:00:54.214-0300 I CONTROL [initandlisten]
2019-08-05T12:00:54.214-0300 I CONTROL [initandlisten] ** WARNING: This 32-bit MongoDB binary is deprecated
2019-08-05T12:00:54.215-0300 I CONTROL [initandlisten]
2019-08-05T12:00:54.215-0300 I CONTROL [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
2019-08-05T12:00:54.215-0300 I CONTROL [initandlisten] ** 32 bit builds are limited to less than 2GB of data (or less with --journal).
2019-08-05T12:00:54.215-0300 I CONTROL [initandlisten] ** Note that journaling defaults to off for 32 bit and is currently off.
2019-08-05T12:00:54.215-0300 I CONTROL [initandlisten] ** See http://dochub.mongodb.org/core/32bit
2019-08-05T12:00:54.216-0300 I CONTROL [initandlisten]
2019-08-05T12:00:54.217-0300 I INDEX [initandlisten] allocating new ns File c:/data/db/local.ns, filling with zeroes...
2019-08-05T12:00:55.140-0300 I STORAGE [FileAllocator] allocating new file c:/data/db/local_0, filling with zeroes...
2019-08-05T12:00:55.151-0300 I STORAGE [FileAllocator] creating directory c:/data/db/tmp
2019-08-05T12:00:55.631-0300 I STORAGE [FileAllocator] done allocating datafile c:/data/db/local_0 size: 64MB, took 0.483 secs
2019-08-05T12:00:56.068-0300 I NETWORK [hostnameCanonicalizationWorker] Starting hostname canonicalization worker
2019-08-05T12:00:56.057-0300 I FTDC [FTDC] Failed to initialize Performance Counters for FTDC: WindowsPdhError: PdhExpandCounterPathW failed with 0x80004005
2019-08-05T12:00:56.057-0300 I FTDC [FTDC] Failed to initialize Performance Counters for FTDC: WindowsPdhError: PdhExpandCounterPathW failed with 0x80004005
2019-08-05T12:00:56.059-0300 I NETWORK [initandlisten] Initializing full-time diagnostic data capture with directory 'c:/data/db/diagnostic.data'
2019-08-05T12:00:56.072-0300 I NETWORK [initandlisten] connection accepted from 127.0.0.1:52527 #1 (1 connection now open)
2019-08-05T12:01:09.872-0300 I NETWORK [initandlisten] connection accepted from 127.0.0.1:52527 #2 (2 connections now open)
2019-08-05T12:01:42.481-0300 I NETWORK [conn2] received client metadata from 127.0.0.1:52527: { driver: { name: "nodejs", version: "8.11.3" }, os: { type: "Windows_NT", name: "win32", architecture: "ia32", version: "6.3.7601" }, platform: "Node.js v10.15.3, i686, mongodb-core: 3.2.7" }
2019-08-05T12:01:43.485-0300 I NETWORK [conn2]
```

## nodemon app.js

```
C:\guia_programador\nodejs\blogapp>nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:14:22
(node:16120) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:16120) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
Servidor rodando!
Mongo conectado...
```

```
C:\guia_programador\nodejs\blogapp>mongo
MongoDB shell version v4.4.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("683e89e9-d4d2-47f2-bd91-9aa9e77d63b6") }
MongoDB server version: 4.4.5
...
The server generated these startup warnings when booting:
2021-04-30T04:07:24.205-03:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
...
Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
...
> show databases;
admin 0.000GB
config 0.000GB
local 0.000GB
testes_mongo 0.000GB
>
```

Nova categoria

Nome:  
Desenvolvimento Web

Slug:  
desenvolvimento-web

**Criar categoria**

- Clique no botão "Criar categoria":

Lista de categorias:

**Nova categoria**

```
Servidor rodando!
Mongo conectado...
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
body-parser deprecated undefined extended; provide extended option app.js:14:22
(node:3708) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:3708) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient
constructor.
Servidor rodando!
Mongo conectado...
Categoria salva com sucesso!
```

mongo

show databases;

```
> show databases;
admin 0.000GB
blogapp 0.000GB
config 0.000GB
local 0.000GB
testes_mongo 0.000GB
> -
```

```
use blogapp;
```

```
> use blogapp;
switched to db blogapp
```

```
show collections;
```

```
> show collections;
categorias
```

```
db.categorias.find()
```

```
> db.categorias.find()
{ "_id" : ObjectId("608d783f5240060e7cc559c0"), "date" : ISODate("2021-05-01T15:39:34.272Z"), "nome" : "Desenvolvimento
Web", "slug" : "desenvolvimento-web", "__v" : 0 }
```

## Aula 37 - O que são sessões e cookies

Nesta aula vamos entender o que são sessões e cookies, elementos essenciais no desenvolvimento de qualquer aplicação web, principalmente no Node.js com Express.js

### Cookies

- ▶ Basicamente, um Cookie é um arquivo de texto muito simples, cuja composição depende diretamente do conteúdo do endereço Web visitado. Por exemplo, a maioria dos sites armazenam informações básicas, como endereços IP e preferências sobre idiomas, cores, etc. Contudo, em portais como o Gmail e o Hotmail, nomes de usuários e senhas de email também fazem parte dos Cookies.

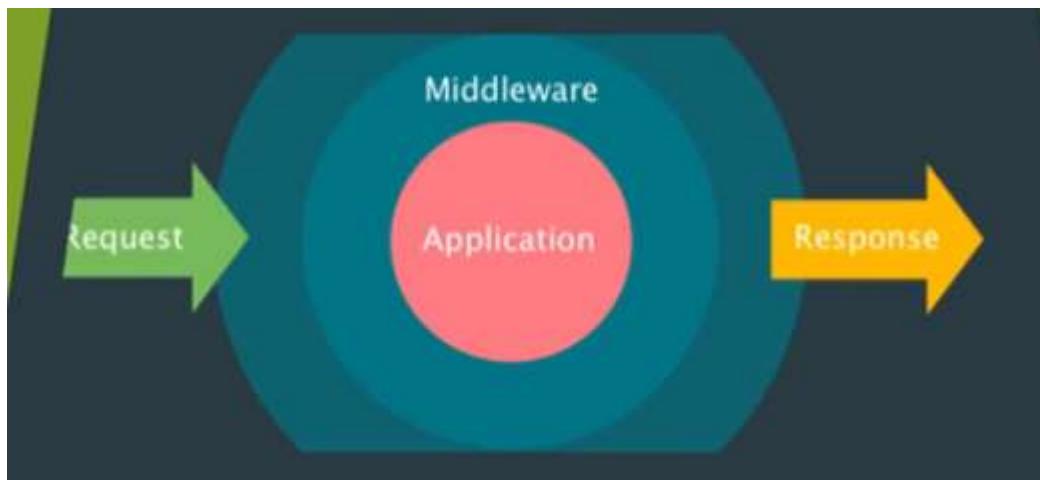
### Sessões

- ▶ Sessões geralmente dependem de cookies, mas os dados são guardados no servidor. Funciona assim:
- ▶ Uma sessão é iniciada no servidor, que envia um cookie ao browser com um ID único daquela sessão.
- ▶ Qualquer dado associado à sessão é armazenado no servidor, associado a esse ID.
- ▶ Em toda requisição, o browser envia de volta o cookie com o ID da sessão, o que permite ao servidor dar acesso aos dados associados àquele ID.
- ▶ Portanto, usar sessões é um pouco mais seguro que guardar dados diretamente em cookies, já que se alguém tiver acesso ao cookie não tem acesso direto aos dados (isso sem falar que não cabem muitos dados nos cookies).

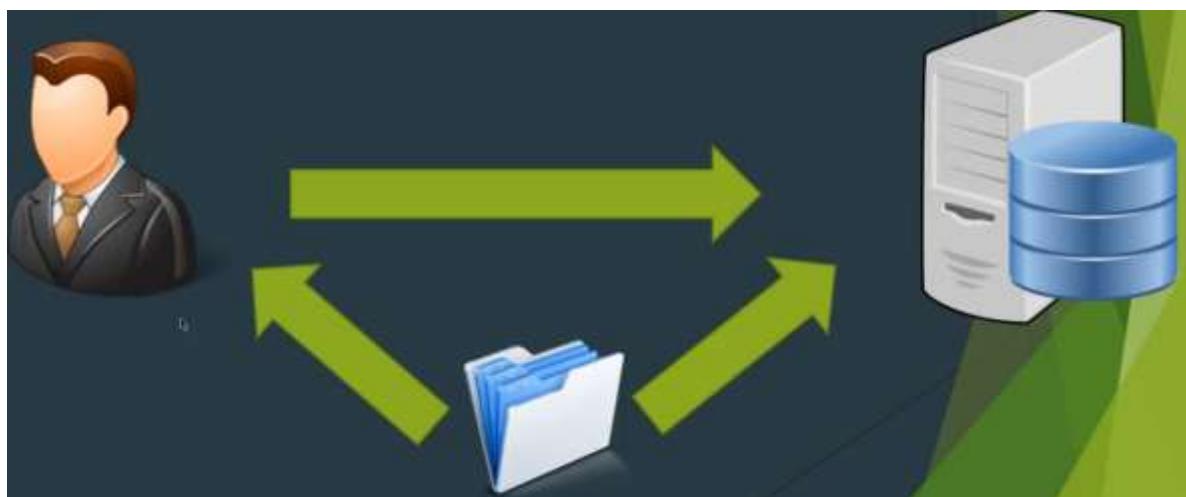
## Aula 38 - Middlewares

Nesta aula vamos entender o que são middlewares no Node.js e como podemos utilizar ele para tirar proveito na hora do desenvolvimento de aplicações web mais robustas.

### O que é um middleware



É um conceito abstrato.



O middleware é uma pequena parte de nossa aplicação que vai ficar intermediando a comunicação cliente/servidor.

Ou seja, o middleware faz a intermediação de todas as requisições feitas pelo usuário.

Com o middleware é possível manipular essas requisições e respostas antes delas chegarem ao destino.

**app.js**

```
/* Carregando módulos */

const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
const app = express()
const admin = require('./routes/admin')
const path = require('path')
const mongoose = require('mongoose')

/* Configurações */

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/blogapp").then(() => {
 console.log('Mongo conectado...')

}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});

// Public
app.use(express.static(path.join(__dirname, "public")))
app.use((req, res, next) => {
 console.log('Oi, eu sou um middleware!')
 next()
})

/* Rotas */

app.get('/', (req, res) => {
 res.send('Página principal')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)

/* Outros */

const PORT = '8081'
```

```
app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

Cada vez que for feita uma requisição (uma página for carregada), irá aparecer a mensagem "Oi, eu sou um middleware!".

```
Servidor rodando!
Mongo conectado...
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:14:22
(node:10476) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:10476) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoC
lient constructor.
Servidor rodando!
Mongo conectado...
Oi, eu sou um middleware!
[]
```

## Aula 39 - Como configurar sessões

Nesta aula vamos ver como trabalhar com sessões no Node.js junto com o Express.js, as sessões são utilitários que servem para armazenar informações temporárias de um cliente em tempo de execução.

### Instalando dependências de projeto

```
npm install express-session --save
```

```
C:\guia_programador\nodejs\blogapp>npm install express-session --save
npm [WARN] saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm [WARN] enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm [WARN] blogapp No description
npm [WARN] blogapp No repository field.
npm [WARN] blogapp No README data
npm [WARN] blogapp No license field.

+ express-session@1.17.1
added 6 packages from 5 contributors and audited 177 packages in 9.071s

3 packages are looking for funding
 run `npm fund` for details

found 0 vulnerabilities
```

```
npm install connect-flash --save
```

```
C:\guia_programador\nodejs\blogapp>npm install connect-flash --save
npm [WARN] saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm [WARN] enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm [WARN] blogapp No description
npm [WARN] blogapp No repository field.
npm [WARN] blogapp No README data
npm [WARN] blogapp No license field.

+ connect-flash@0.1.1
added 1 package from 1 contributor and audited 181 packages in 4.951s

3 packages are looking for funding
 run `npm fund` for details

found 0 vulnerabilities
```

## app.js

```
/* Carregando módulos */

const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
const app = express()
const admin = require('./routes/admin')
const path = require('path')
const mongoose = require('mongoose')
const session = require('express-session')
const flash = require('connect-flash')

/* Configurações */

// Session
app.use(session({
 secret: "cursodenode",
 resave: true,
 saveUninitialized: true
}))
app.use(flash())

// Middleware
app.use((req, res, next) => {
 res.locals.success_msg = req.flash("success_msg")
 res.locals.error_msg = req.flash("error_msg")
 next()
})

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/blogapp").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});

// Public
app.use(express.static(path.join(__dirname, "public")))
app.use((req, res, next) => {
 console.log('Oi, eu sou um middleware!')
 next()
})

/* Rotas */
```

```
app.get('/', (req, res) => {
 res.send('Página principal')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)

/* Outros */

const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

## Aula 40 - Como validar formulários no Express.js

Nesta aula vamos aprender a como validar formulários no framework do Node.js Express.js utilizando algumas técnicas básicas de validação do javascript.

### routes\admin.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Categoria')
const Categoria = mongoose.model("categorias")

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get('/categorias', (req, res) => {
 res.render('admin/categorias')
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', (req, res) => {

 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }

 if(req.body.nome.length < 2)
 {
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }

 if(erros.length > 0){
```

```

 res.render("admin/add_categoria", {erros: erros})
} else {
 const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
 }

 new Categoria(novaCategoria).save().then(() => {
 req.flash("success_msg", "Categoria criada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a categoria! Tente novamente.")
 res.redirect("/admin")
 })
}

module.exports = router

```

### views\admin\add\_categoria.handlebars

```

{{#each erros}}
 <div class="alert alert-danger">{{texto}}</div>
{{else}}

{{/each}}

<h3>Nova categoria</h3>

<div class="card">
 <div class="card-body">
 <form action="/admin/categorias/nova" method="POST">
 <label for="nome">Nome: </label>
 <input type="text" id="nome" name="nome" placeholder="Nome da categoria" class="form-control">

 <label for="slug">Slug: </label>
 <input type="text" id="slug" name="slug" placeholder="Slug da categoria" class="form-control">

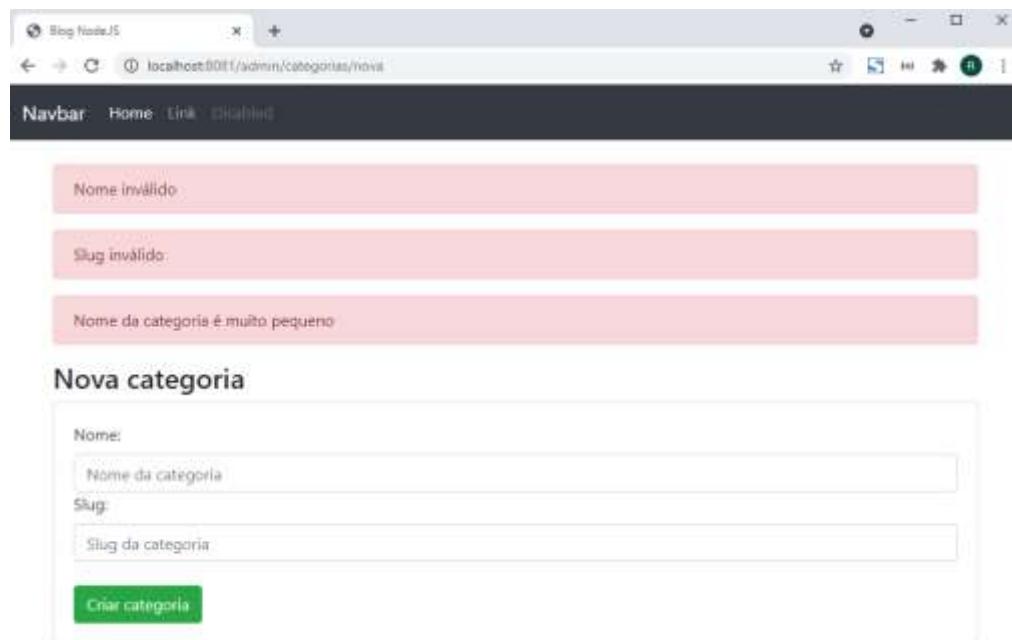
 <button type="submit" class="btn btn-success mt-4">Criar categoria</button>
 </form>
 </div>
</div>

```

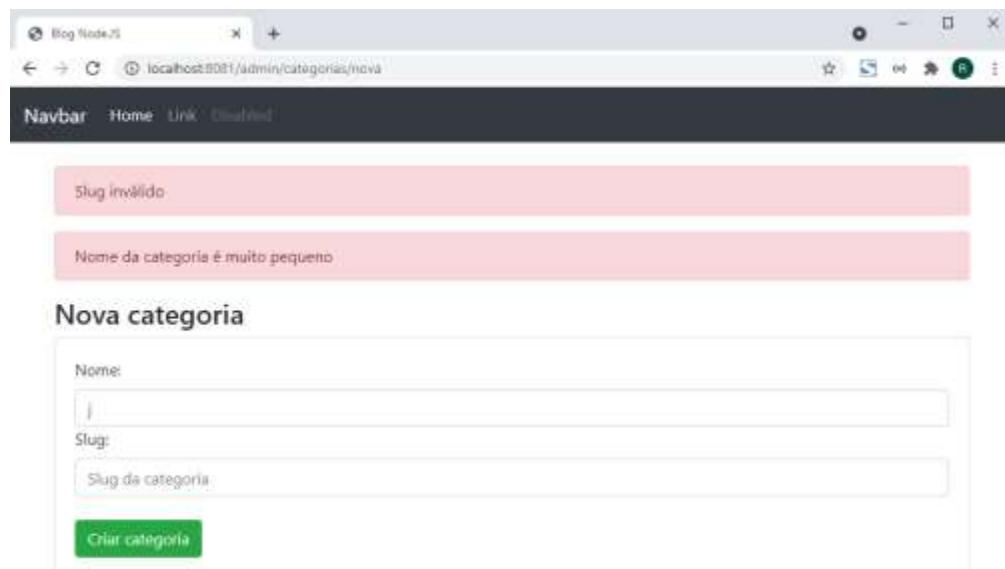
## nodemon app.js

```
C:\guias_programador\nodejs\blogapp>nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:31:22
(node:19428) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:19428) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoC
lient constructor.
Servidor rodando!
Mongo conectado...
```

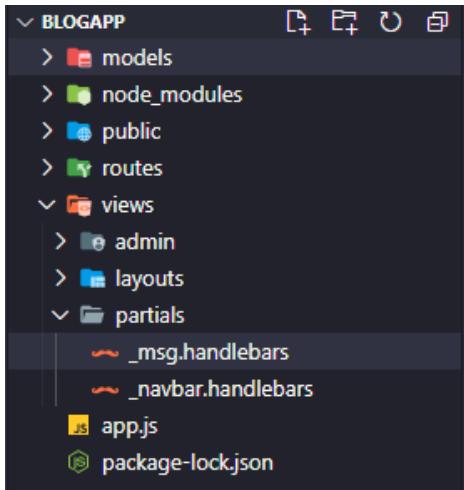
- Com os campos em branco, clique no botão "Criar categoria"



The screenshot shows a browser window with the title "Blog NodeJS". The address bar indicates the URL is "localhost:8081/admin/categorias/nova". The page has a dark header with "Navbar" and "Home" links. Below the header, there are three red validation error boxes: "Nome inválido", "Slug inválido", and "Nome da categoria é muito pequeno". The main form area is titled "Nova categoria" and contains two input fields: "Nome" (with placeholder "Nome da categoria") and "Slug" (with placeholder "Slug da categoria"). A green "Criar categoria" button is at the bottom.



The screenshot shows a browser window with the title "Blog NodeJS". The address bar indicates the URL is "localhost:8081/admin/categorias/nova". The page has a dark header with "Navbar" and "Home" links. Below the header, there are two red validation error boxes: "Slug inválido" and "Nome da categoria é muito pequeno". The main form area is titled "Nova categoria" and contains two input fields: "Nome" (empty) and "Slug" (empty). A green "Criar categoria" button is at the bottom.



### views\partials\\_msg.handlebars

```
{#{if success_msg}
 <div class="alert alert-success">{{success_msg}}</div>
{#/if}

{#{if error_msg}
 <div class="alert alert-danger">{{error_msg}}</div>
{#/if}}
```

## views\layouts\main.handlebars

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
 <link rel="stylesheet" href="/css/bootstrap.css">
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta http-equiv="X-UA-Compatible" content="ie=edge">
 <title>Blog Node.JS</title>
</head>
<body>
 {{>_navbar}}

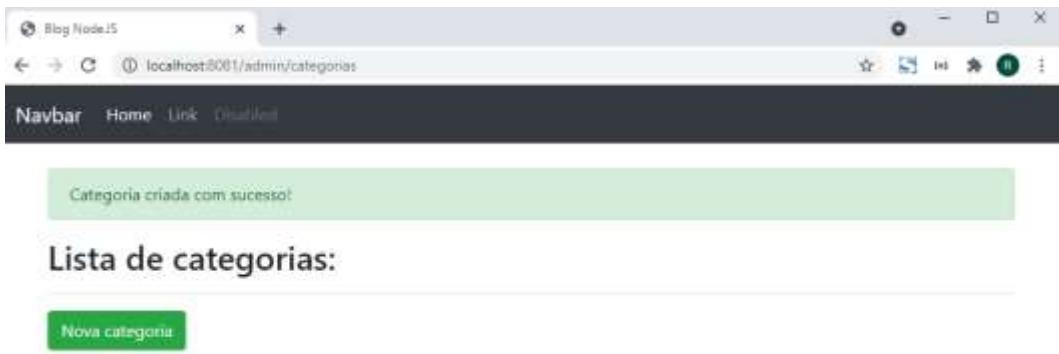
 <div class="container mt-4">
 {{>_msg}}
 {{{body}}}
 </div>

 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzOOrT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abTE1Pi6jizo" crossorigin="anonymous"></script>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
 <script src="/js/bootstrap.js"></script>
</body>
</html>
```

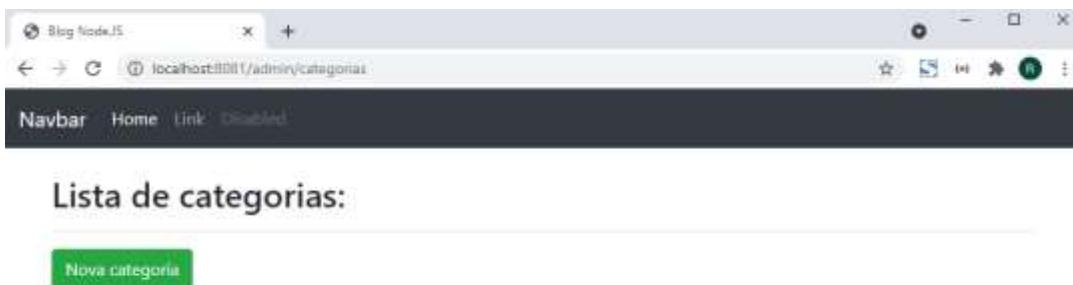


The screenshot shows a web browser window with the title "Blog Node.JS". The address bar displays "localhost:8081/admin/categorias/add". The page has a dark header bar with the text "Navbar" and "Home". Below the header is a form titled "Nova categoria". The form contains two input fields: "Nome" with the value "Marketing" and "Slug" with the value "marketing". At the bottom of the form is a green button labeled "Criar categoria".

- Clique no botão "**Criar categoria**"



- Recarregando a página, a mensagem desaparece:



- Crie uma nova categoria chamada **PHP**:

The screenshot shows a web browser window titled "Blog NodeJS". The URL is "localhost:8081/admin/categorias/add". The page has a dark header with "Navbar", "Home", "Link", and "Disabled" items. Below the header is a form titled "Nova categoria". The form has two input fields: "Nome" containing "PHP" and "Slug" containing "php". A green button labeled "Criar categoria" is at the bottom of the form.

The screenshot shows a web browser window titled "Blog NodeJS". The URL is "localhost:8081/admin/categorias". The page has a dark header with "Navbar", "Home", "Link", and "Disabled" items. Below the header is a green success message box containing the text "Categoria criada com sucesso!".

- Crie uma nova categoria chamada **Javascript**:

The screenshot shows a web browser window titled "Blog NodeJS". The URL is "localhost:8081/admin/categorias/add". The page has a dark header with "Navbar", "Home", "Link", and "Disabled" items. Below the header is a form titled "Nova categoria". The form has two input fields: "Nome" containing "Javascript" and "Slug" containing "javascript". A green button labeled "Criar categoria" is at the bottom of the form.

The screenshot shows a browser window titled "Blog NodeJS" at the URL "localhost:8081/admin/categorias". The page has a dark header with "Navbar", "Home", "Link", and "Disabled" buttons. A green success message box contains the text "Categoria criada com sucesso!". Below it, a section titled "Lista de categorias:" lists five categories: "Desenvolvimento Web", "Marketing", "PHP", "Javascript", and "Jquery". A green button labeled "Nova categoria" is visible.

- No Mongo:

db.categorias.find()

```
> db.categorias.find()
[{"_id": ObjectId("608d783f5240060e7cc559c0"), "date": ISODate("2021-05-01T15:39:34.272Z"), "nome": "Desenvolvimento Web", "slug": "desenvolvimento-web", "__v": 0},
 {"_id": ObjectId("608d8ee80c36f44bdc6c3d40"), "date": ISODate("2021-05-01T16:58:18.549Z"), "nome": "Marketing", "slug": "marketing", "__v": 0},
 {"_id": ObjectId("608dfdf40c36f44bdc6c3d41"), "date": ISODate("2021-05-01T16:58:18.549Z"), "nome": "PHP", "slug": "php", "__v": 0},
 {"_id": ObjectId("608d9d235a530140ec81579b"), "date": ISODate("2021-05-01T18:16:42.282Z"), "nome": "Javascript", "slug": "javascript", "__v": 0}]
```

## Aula 41 - Listando categorias

routes\admin.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Categoria')
const Categoria = mongoose.model("categorias")

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get("/categorias", (req, res) => {
 Categoria.find().lean().sort({date: 'desc'}).then((categorias) => {
 res.render("admin/categorias", {categorias: categorias})
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao listar as categorias")
 res.redirect("/admin")
 })
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', (req, res) => {
 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2)
 {
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
})
```

```

if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
} else {
 const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
 }

 new Categoria(novaCategoria).save().then(() => {
 req.flash("success_msg", "Categoria criada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a categoria! Tente novamente.")
 res.redirect("/admin")
 })
}

module.exports = router

```

### views\admin\categorias.handlebars

```

<h2>Lista de categorias:</h2>
<hr>
<button class="btn btn-success">Nova categoria</button>

{{#each categorias}}
 <div class="card mt-4">
 <div class="card-body">
 <h4>{{nome}}</h4>
 <small>Slug: {{slug}}</small>

 <small>Data de criação: {{date}}</small>
 </div>
 </div>
{{else}}
{{/each}}

```

- Adicione uma categoria chamada "HTML":

The screenshot shows a web application interface for managing categories. At the top, there's a header with a logo, a search bar, and navigation links for 'Home' and 'Logout'. Below the header, a modal window titled 'Nova categoria' (New category) is open, containing fields for 'Nome' (Name) and 'Slug' (Slug), both set to 'HTML'. A green button labeled 'Cadastrar' (Register) is at the bottom of the modal. The main content area shows a success message: 'Categoria criada com sucesso!' (Category created successfully!). Below this, a section titled 'Lista de categorias:' (List of categories) displays five items in cards:

- HTML**  
Slug: html  
Data de criação: Sat May 01 2021 13:51:51 GMT-0300 (Horário Padrão de Brasília)
- Javascript**  
Slug: javascript  
Data de criação: Sat May 01 2021 13:58:42 GMT-0300 (Horário Padrão de Brasília)
- Marketing**  
Slug: marketing  
Data de criação: Sat May 01 2021 13:58:18 GMT-0300 (Horário Padrão de Brasília)
- PHP**  
Slug: php  
Data de criação: Sat May 01 2021 13:58:18 GMT-0300 (Horário Padrão de Brasília)
- Desenvolvimento Web**  
Slug: desenvolvimento-web  
Data de criação: Sat May 01 2021 12:39:34 GMT-0300 (Horário Padrão de Brasília)

- No Mongo:

```
> db.categorias.find()
{ "_id" : ObjectId("608d783f5240060e7cc559c0"), "date" : ISODate("2021-05-01T15:39:34.272Z"), "name" : "Desenvolvimento Web", "slug" : "desenvolvimento-web", "__v" : 0 }
{ "_id" : ObjectId("608d8ee80c36f44bdc6c3d40"), "date" : ISODate("2021-05-01T16:58:18.549Z"), "name" : "Marketing", "slug" : "marketing", "__v" : 0 }
{ "_id" : ObjectId("608d8fd40c36f44bdc6c3d41"), "date" : ISODate("2021-05-01T16:58:18.549Z"), "name" : "PHP", "slug" : "php", "__v" : 0 }
{ "_id" : ObjectId("608d9d235a530148ec81579b"), "date" : ISODate("2021-05-01T18:16:42.282Z"), "name" : "Javascript", "slug" : "javascript", "__v" : 0 }
{ "_id" : ObjectId("608da431b47b124b9cc8a9d0"), "date" : ISODate("2021-05-01T18:51:51.816Z"), "name" : "HTML", "slug" : "html", "__v" : 0 }
```

## Aula 42 - Editar categoria

routes\admin.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Categoria')
const Categoria = mongoose.model("categorias")

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get("/categorias", (req, res) => {
 Categoria.find().lean().sort({date: 'desc'}).then((categorias) => {
 res.render("admin/categorias", {categorias: categorias})
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as categorias")
 res.redirect("/admin")
 })
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', (req, res) => {
 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2){
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
 if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
 }
})
```

```

} else {
 const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
 }

 new Categoria(novaCategoria).save().then(() => {
 req.flash("success_msg", "Categoria criada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a categoria! Tente novamente.")
 res.redirect("/admin")
 })
}

router.get("/categorias/edit/:id", (req, res) => {
 Categoria.findOne({_id: req.params.id}).then((categoria) => {
 res.render("admin/edit_categoria", {categoria: categoria})
 }).catch((err) => {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/admin/categorias")
 })
})

router.post("/categorias/edit", (req, res) => {

 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2){
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
 if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
 } else {

 Categoria.findOne({_id: req.body.id}).lean().then((categoria) => {

 categoria.nome = req.body.nome
 categoria.slug = req.body.slug

 categoria.save().then(() => {
 req.flash("success_msg", "Categoria editada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {

```

```
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da categoria")
 res.redirect("/admin/categorias")
 })

}).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a categoria!")
 res.redirect("/admin/categorias")
})

}

module.exports = router
```

### views\admin\categoria.handlebars

```
<h2>Lista de categorias:</h2>
<hr>
<button class="btn btn-success">Nova categoria</button>

{{#each categorias}}
 <div class="card mt-4">
 <div class="card-body">
 <h4>{{nome}}</h4>
 <small>Slug: {{slug}}</small>

 <small>Data de criação: {{date}}</small>

 <button class="btn btn-success mt-4">Editar</button>
 </div>
 </div>
{{else}}
{{/each}}
```

## views\admin\edit\_categoria.handlebars

```
{#{each erros}}
 <div class="alert alert-danger">{{texto}}</div>
{#{else}}

{#/each}

<h3>Edição de categoria</h3>

<div class="card">
 <div class="card-body">
 <form action="/admin/categorias/edit" method="POST">
 <input type="hidden" name="id" value="{{categoria._id}}>
 <label for="nome">Nome: </label>
 <input type="text" id="nome" name="nome" placeholder="Nome da categoria" class="form-control" value="{{categoria.nome}}>

 <label for="slug">Slug: </label>
 <input type="text" id="slug" name="slug" placeholder="Slug da categoria" class="form-control" value="{{categoria.slug}}>

 <button type="submit" class="btn btn-success mt-4">Editar</button>
 </form>
 </div>
</div>
```

The screenshot shows a web application interface for managing categories. At the top, there's a header bar with a logo, a search icon, and other navigation elements. Below it, a dark navigation bar contains links for 'Home', 'Link', and 'Logout'. The main content area is titled 'Lista de categorias:'.

Three categories are listed:

- HTML**  
Slug: html  
Data de criação: Saturday, 01 May 2021 11:01:10 GMT-0300 (Horário Padrão do Brasil)  
[Editar](#)
- Javascript**  
Slug: javascript  
Data de criação: Saturday, 01 May 2021 11:01:42 GMT-0300 (Horário Padrão do Brasil)  
[Editar](#)
- Marketing**  
Slug: marketing  
Data de criação: Saturday, 01 May 2021 11:01:10 GMT-0300 (Horário Padrão do Brasil)  
[Editar](#)

- Clique no botão "Editar" da categoria "Marketing":



Blog Node.js

localhost:8081/admin/categorias/edit/608d8ee80c36f44bdc6c3d40

Navbar Home Link Disabled

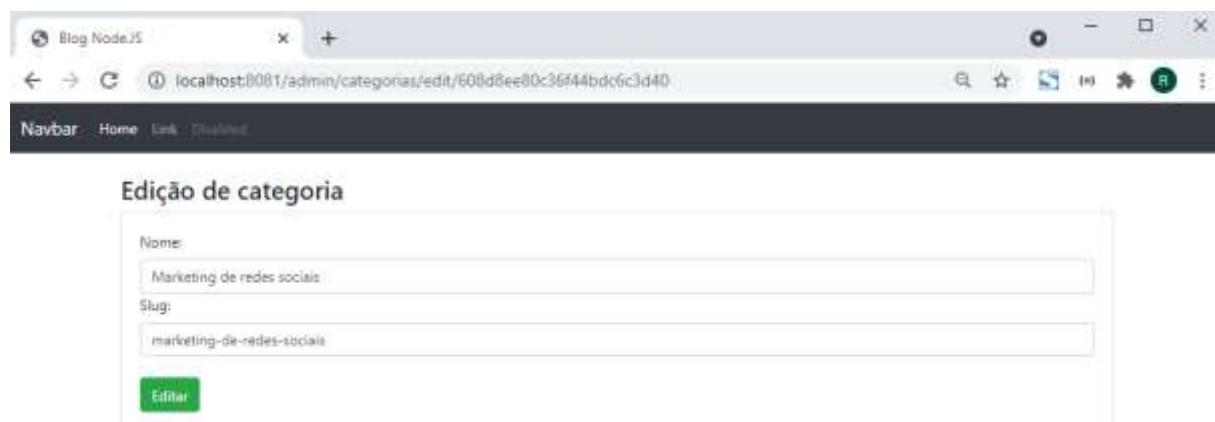
## Edição de categoria

Nome:

Slug:

**Editar**

- Altere para Nome: "Marketing de redes sociais", Slug: "marketing-de-redes-sociais"



Blog Node.js

localhost:8081/admin/categorias/edit/608d8ee80c36f44bdc6c3d40

Navbar Home Link Disabled

## Edição de categoria

Nome:

Slug:

**Editar**

- Clique no botão "Editar"

Navbar Home Link

Categoria editada com sucesso!

### Lista de categorias:

[Nova categoria](#)

**HTML**

Slug: html  
Data de criação: Sat May 01 2021 13:51:51 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

**Javascript**

Slug: javascript  
Data de criação: Sat May 01 2021 13:16:42 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

**Marketing de redes sociais**

Slug: marketing-de-redes-sociais  
Data de criação: Sat May 01 2021 13:58:18 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

**PHP**

Slug: php  
Data de criação: Sat May 01 2021 13:58:18 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

**Desenvolvimento Web**

Slug: desenvolvimento-web  
Data de criação: Sat May 01 2021 12:39:34 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

## Aula 43 - Deletando categorias

routes\admin.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Categoria')
const Categoria = mongoose.model("categorias")

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get("/categorias", (req, res) => {
 Categoria.find().sort({date:'desc'}).then((categorias) => {
 res.render("admin/categorias", {categorias: categorias})
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as categorias")
 res.redirect("/admin")
 })
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', (req, res) => {
 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2)
 {
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
})
```

```

if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
} else {
 const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
 }

 new Categoria(novaCategoria).save().then(() => {
 req.flash("success_msg", "Categoria criada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a categoria! Tente novamente.")
 res.redirect("/admin")
 })
}

router.get("/categorias/edit/:id", (req, res) => {
 Categoria.findOne({_id: req.params.id}).then((categoria) => {
 res.render("admin/edit_categoria", {categoria: categoria})
 }).catch((err) => {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/admin/categorias")
 })
})

router.post("/categorias/edit", (req, res) => {

 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }

 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }

 if(req.body.nome.length < 2)
 {
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }

 if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
 } else {

 Categoria.findOne({_id: req.body.id}).then((categoria) => {

```

```

 categoria.nome = req.body.nome
 categoria.slug = req.body.slug

 categoria.save().then(() => {
 req.flash("success_msg", "Categoria editada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da categoria")
 res.redirect("/admin/categorias")
 })

 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a categoria!")
 res.redirect("/admin/categorias")
 })
}

router.post("/categorias/deletar", (req, res) => {

 Categoria.remove({_id: req.body.id}).then(() => {
 req.flash("success_msg", "Categoria deletada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao deletar a categoria!")
 res.redirect("/admin/categorias")
 })
}

module.exports = router

```

## views\admin\categoria.handlebars

```

<h2>Lista de categorias:</h2>
<hr>
<button class="btn btn-success">Nova categoria</button>

{{#each categorias}}
 <div class="card mt-4">
 <div class="card-body">
 <h4>{{nome}}</h4>
 <small>Slug: {{slug}}</small>

 <small>Data de criação: {{date}}</small>

 <button class="btn btn-success mt-4">Editar</button>

 <form action="/admin/categorias/deletar" method="POST">
 <input type="hidden" name="id" value="{{_id}}>
 <button type="submit" class="btn btn-danger mt-2">Deletar</button>
 </form>
 </div>
 </div>

```

```
</div>
{{else}}
<h4 class="mt-3">Nenhuma categoria registrada</h4>
{{/each}}
```

Navbar: Home | Link | Desenvolvedor

## Lista de categorias:

[Nova categoria](#)

### HTML

Slug: html

Data de criação: Sat May 01 2021 15:51:51 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

[Deletar](#)

### Javascript

Slug: javascript

Data de criação: Sat May 01 2021 15:16:42 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

[Deletar](#)

### Marketing de redes sociais

Slug: marketing-de-redes-sociais

Data de criação: Sat May 01 2021 13:58:18 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

[Deletar](#)

### PHP

Slug: php

Data de criação: Sat May 01 2021 13:58:18 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

[Deletar](#)

### Desenvolvimento Web

Slug: desenvolvimento-web

Data de criação: Sat May 01 2021 12:39:34 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

[Deletar](#)

- Delete todas as categorias

A screenshot of a web browser window titled "Blog NodeJS". The address bar shows "localhost:8081/admin/categorias". The page has a dark header with "Navbar", "Home", and "Link Disabled". A green success message box at the top says "Categoria deletada com sucesso!". Below it, the heading "Lista de categorias:" is displayed, followed by a green "Nova categoria" button.

Categoria deletada com sucesso!

## Lista de categorias:

[Nova categoria](#)

Nenhuma categoria registrada

- Adicione a categoria "**Javascript**"

A screenshot of a web browser window titled "Blog NodeJS". The address bar shows "localhost:8081/admin/categorias/add". The page has a dark header with "Navbar", "Home", and "Link Disabled". The main content area is titled "Nova categoria" and contains fields for "Nome" (with "Javascript" typed) and "Slug" (with "javascript"). A green "Criar categoria" button is at the bottom.

### Nova categoria

Nome:

Slug:

[Criar categoria](#)

A screenshot of a web browser window titled "Blog NodeJS". The address bar shows "localhost:8081/admin/categorias". The page has a dark header with "Navbar", "Home", and "Link Disabled". A green success message box at the top says "Categoria criada com sucesso!". Below it, the heading "Lista de categorias:" is displayed, followed by a green "Nova categoria" button.

Categoria criada com sucesso!

## Lista de categorias:

[Nova categoria](#)

### Javascript

Slug: javascript

Data de criação: Sat May 01 2021 18:46:10 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

[Deletar](#)

## Aula 44 - Definindo o model de postagens

models\Postagem.js

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

// Model - Postagem

const Postagem = new Schema({
 titulo: {
 type: String,
 required: true
 },
 slug: {
 type: String,
 required: true
 },
 descricao: {
 type: String,
 required: true
 },
 conteudo: {
 type: String,
 required: true
 },
 categoria: {
 type: Schema.Types.ObjectId,
 ref: "categorias",
 required: true
 },
 data: {
 type: Date,
 default: Date.now
 }
})

// Collection

mongoose.model('postagens', Postagem)
```

## Aula 45 - Formulário de postagens

routes\admin.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Categoria')
const Categoria = mongoose.model("categorias")
require('../models/Postagem')
const Postagem = mongoose.model("postagens")

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get("/categorias", (req, res) => {
 Categoria.find().sort({date:'desc'}).then((categorias) => {
 res.render("admin/categorias", {categorias: categorias})
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as categorias")
 res.redirect("/admin")
 })
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', (req, res) => {
 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2){
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
})
```

```

if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
} else {
 const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
 }
 new Categoria(novaCategoria).save().then(() => {
 req.flash("success_msg", "Categoria criada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a categoria! Tente novamente.")
 res.redirect("/admin")
 })
}
}

router.get("/categorias/edit/:id", (req, res) => {
 Categoria.findOne({_id: req.params.id}).then((categoria) => {
 res.render("admin/edit_categoria", {categoria: categoria})
 }).catch((err) => {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/admin/categorias")
 })
})

router.post("/categorias/edit", (req, res) => {
 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2)
 {
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
 if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
 } else {
 Categoria.findOne({_id: req.body.id}).then((categoria) => {

 categoria.nome = req.body.nome
 categoria.slug = req.body.slug

 categoria.save().then(() => {
 req.flash("success_msg", "Categoria editada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da categoria")
 })
 })
 }
})
}

```

```

 res.redirect("/admin/categorias")
 })

}).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a categoria!")
 res.redirect("/admin/categorias")
})
}

router.post("/categorias/deletar", (req, res) => {

 Categoria.remove({_id: req.body.id}).then(() => {
 req.flash("success_msg", "Categoria deletada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao deletar a categoria!")
 res.redirect("/admin/categorias")
 })
})

router.get("/postagens", (req, res) => {
 Postagem.find().lean().sort({date:'desc'}).then((postagens) => {
 res.render("admin/postagens", {postagens: postagens})
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as postagens")
 res.redirect("/admin")
 })
})

router.get('/postagens/add', (req, res) => {
 Categoria.find().lean().then((categorias) => {
 res.render('admin/add_postagem', {categorias: categorias})
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao carregar o formulário")
 res.redirect("/admin")
 })
})

module.exports = router

```

## views\admin\postagens.handlebars

```

<h2>Lista de postagens:</h2>
<hr>
<button class="btn btn-success">Nova postagem</button>

```

## views\admin\add\_postagem.handlebars

```
{{#each erros}}
 <div class="alert alert-danger">{{texto}}</div>
{{else}}
{{/each}}

<h3>Nova postagem</h3>

<div class="card">
 <div class="card-body">
 <form action="/admin/postagens/nova" method="POST">
 <label for="titulo">Título: </label>
 <input type="text" id="titulo" name="titulo" placeholder="Título da postagem" class="form-control mb-1">

 <label for="slug">Slug: </label>
 <input type="text" id="slug" name="slug" placeholder="Slug da postagem" class="form-control mb-1">

 <label for="descricao">Descrição: </label>
 <input type="text" id="descricao" name="descricao" placeholder="Descrição da postagem" class="form-control mb-1">

 <label for="conteudo">Conteúdo: </label>
 <textarea id="conteudo" name="conteudo" rows="8" placeholder="Conteúdo da postagem" class="form-control mb-1"></textarea>

 <label for="categoria">Categoria: </label>
 <select id="categoria" name="categoria" class="custom-select mt-2">
 {{#each categorias}}
 <option value="{{_id}}>{{nome}}</option>
 {{else}}
 <option value="0">Nenhuma categoria registrada</option>
 {{/each}}
 </select>

 <button type="submit" class="btn btn-success mt-4">Criar postagem</button>
 </form>
 </div>
</div>
```

**Lista de categorias:**

**Javascript**  
Slug: javascript  
Data de criação: Sat May 01 2021 18:46:10 GMT-0300 (Horário Padrão de Brasília)

**Editar** **Deletar**

- Delete a categoria "Javascript"

**Categoria deletada com sucesso!**

**Lista de categorias:**

Nenhuma categoria registrada

<http://localhost:8081/admin/postagens/add>

**Nova postagem**

**Titulo:**  
Título da postagem

**Slug:**  
Slug da postagem

**Descrição:**  
Descrição da postagem

**Conteúdo:**  
Conteúdo da postagem

**Categorias:**  
Nenhuma categoria registrada

**Criar postagem**

- Adicione as categorias "PHP" e "Javascript"

Nova categoria

Nome:  
PHP

Slug:  
php

Criar categoria

Categoria criada com sucesso!

Lista de categorias:

Nova categoria

**PHP**  
Slug: php  
Data de criação: Sat May 01 2021 19:20:03 GMT-0300 (Horário Padrão de Brasília)

Editar

Deletar

Nova categoria

Nome:  
Javascript

Slug:  
javascript

Criar categoria

A screenshot of a web browser window titled "Blog NodeJS". The address bar shows "localhost:8081/admin/categorias". The page has a dark header with "Navbar", "Home", and "Link Disabled". A green success message box says "Categoria criada com sucesso!". Below it, the heading "Lista de categorias:" is followed by two items in boxes:

- PHP**  
Slug: php  
Data de criação: Sat May 01 2021 19:20:03 GMT-0300 (Horário Padrão do Brasil)  
[Editar](#) [Deletar](#)
- Javascript**  
Slug: javascript  
Data de criação: Sat May 01 2021 19:20:03 GMT-0300 (Horário Padrão do Brasil)  
[Editar](#) [Deletar](#)

<http://localhost:8081/admin/postagens/add>

A screenshot of a web browser window titled "Blog NodeJS". The address bar shows "localhost:8081/admin/postagens/add". The page has a dark header with "Navbar", "Home", and "Link Disabled". The main content is a form titled "Nova postagem" with fields for:

- Título:
- Slug:
- Descrição:
- Conteúdo:
- Postagem:

At the bottom is a green "Criar postagem" button.

## Aula 46 - Salvando postagens no banco de dados

routes\admin.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Categoria')
const Categoria = mongoose.model("categorias")
require('../models/Postagem')
const Postagem = mongoose.model("postagens")

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get("/categorias", (req, res) => {
 Categoria.find().sort({date:'desc'}).then((categorias) => {
 res.render("admin/categorias", {categorias: categorias})
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as categorias")
 res.redirect("/admin")
 })
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', (req, res) => {
 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2){
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
})
```

```

if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
} else {
 const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
 }

 new Categoria(novaCategoria).save().then(() => {
 req.flash("success_msg", "Categoria criada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a categoria! Tente novamente.")
 res.redirect("/admin")
 })
}
}

router.get("/categorias/edit/:id", (req, res) => {
 Categoria.findOne({_id: req.params.id}).then((categoria) => {
 res.render("admin/edit_categoria", {categoria: categoria})
 }).catch((err) => {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/admin/categorias")
 })
})

router.post("/categorias/edit", (req, res) => {

 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2){
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
 if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
 } else {

 Categoria.findOne({_id: req.body.id}).then((categoria) => {

 categoria.nome = req.body.nome
 categoria.slug = req.body.slug

 categoria.save().then(() => {
 req.flash("success_msg", "Categoria editada com sucesso!")
 })
 })
 }
})
}

```

```

 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da categoria")
 res.redirect("/admin/categorias")
 })

}).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a categoria!")
 res.redirect("/admin/categorias")
})
}
})

router.post("/categorias/deletar", (req, res) => {

Categoria.remove({_id: req.body.id}).then(() => {
 req.flash("success_msg", "Categoria deletada com sucesso!")
 res.redirect("/admin/categorias")
}).catch((err) => {
 req.flash("error_msg", "Houve um erro ao deletar a categoria!")
 res.redirect("/admin/categorias")
})
}

})

router.get("/postagens", (req, res) => {
 res.render("admin/postagens")
}).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as postagens")
 res.redirect("/admin")
})
}

router.get('/postagens/add', (req, res) => {
 Categoria.find().lean().then((categorias) => {
 res.render('admin/add_postagem', {categorias: categorias})
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao carregar o formulário")
 res.redirect("/admin")
 })
}

)

router.post('/postagens/nova', (req, res) => {
 var erros = []

if(!req.body.titulo || typeof req.body.titulo == undefined || req.body.titulo == null){
 erros.push({
 texto: 'Título inválido'
 })
}

if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
}
})

```

```

if(!req.body.descricao || typeof req.body.descricao == undefined || req.body.descricao == null){
 erros.push({
 texto: 'Descrição inválida'
 })
}

if(!req.body.conteudo || typeof req.body.conteudo == undefined || req.body.conteudo == null){
 erros.push({
 texto: 'Conteúdo inválido'
 })
}

if(req.body.categoria == 0){
 erros.push({
 texto: 'Categoria inválida! Registre uma categoria.'
 })
}

if(req.body.descricao.length < 3)
{
 erros.push({
 texto: "A descrição da postagem é muito pequena!"
 })
}

if(erros.length > 0){
 res.render("admin/add_postagem", {erros: erros})
} else {
 const novaPostagem = {
 titulo: req.body.titulo,
 slug: req.body.slug,
 descricao: req.body.descricao,
 conteudo: req.body.conteudo,
 categoria: req.body.categoria
 }

 new Postagem(novaPostagem).save().then(() => {
 req.flash("success_msg", "Postagem criada com sucesso!")
 res.redirect("/admin/postagens")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a postagem! Tente novamente.")
 res.redirect("/admin/postagens")
 })
}
}

module.exports = router

```

- Cadastre uma postagem:

Título: Javascript do futuro

Slug: javascript-do-futuro

Descrição: JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

Conteúdo: Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. E a mesma ainda tem grande espaço para evoluir e já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades.

Nova postagem

Título:  
Javascript do futuro

Slug:  
javascript-do-futuro

Descrição:  
JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares.

Conteúdo:  
Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. E a mesma ainda tem grande espaço para evoluir e já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades.

Javascript

Criar postagem

- Clique no botão "Criar postagem"

Postagem criada com sucesso!

Lista de postagens:

Nova postagem

## Aula 47 - Listando postagens

routes\admin.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Categoria')
const Categoria = mongoose.model("categorias")
require('../models/Postagem')
const Postagem = mongoose.model("postagens")

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get("/categorias", (req, res) => {
 Categoria.find().lean().sort({date:'desc'}).then((categorias) => {
 res.render("admin/categorias", {categorias: categorias})
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as categorias")
 res.redirect("/admin")
 })
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', (req, res) => {
 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2){
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
})
```

```

if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
} else {
 const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
 }

 new Categoria(novaCategoria).save().then(() => {
 req.flash("success_msg", "Categoria criada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a categoria! Tente novamente.")
 res.redirect("/admin")
 })
}

router.get("/categorias/edit/:id", (req, res) => {
 Categoria.findOne({_id: req.params.id}).lean().then((categoria) => {
 res.render("admin/edit_categoria", {categoria: categoria})
 }).catch((err) => {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/admin/categorias")
 })
})

router.post("/categorias/edit", (req, res) => {
 var erros = []

 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }

 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }

 if(req.body.nome.length < 2)
 {
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }

 if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
 } else {

 Categoria.findOne({_id: req.body.id}).then((categoria) => {

 categoria.nome = req.body.nome
 categoria.slug = req.body.slug
 })
 }
})

```

```

 categoria.save().then(() => {
 req.flash("success_msg", "Categoria editada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da categoria")
 res.redirect("/admin/categorias")
 })

 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a categoria!")
 res.redirect("/admin/categorias")
 })
 }
 })

router.post("/categorias/deletar", (req, res) => {
 Categoria.remove({_id: req.body.id}).then(() => {
 req.flash("success_msg", "Categoria deletada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao deletar a categoria!")
 res.redirect("/admin/categorias")
 })
})

router.get("/postagens", (req, res) => {
 Postagem.find().lean().populate("categoria").sort({data: "desc"}).then((postagens) => {
 res.render("admin/postagens", {postagens: postagens})
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao listar as postagens")
 res.redirect("/admin")
 })
})

router.get('/postagens/add', (req, res) => {
 Categoria.find().lean().then((categorias) => {
 res.render('admin/add_postagem', {categorias: categorias})
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao carregar o formulário")
 res.redirect("/admin")
 })
})

router.post('/postagens/nova', (req, res) => {
 var erros = []

 if(!req.body.titulo || typeof req.body.titulo == undefined || req.body.titulo == null){
 erros.push({
 texto: 'Título inválido'
 })
 }

 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
})

```

```

}

if(!req.body.descricao || typeof req.body.descricao == undefined || req.body.descricao == null){
 erros.push({
 texto: 'Descrição inválida'
 })
}

if(!req.body.conteudo || typeof req.body.conteudo == undefined || req.body.conteudo == null){
 erros.push({
 texto: 'Conteúdo inválido'
 })
}

if(req.body.categoria == 0){
 erros.push({
 texto: 'Categoria inválida! Registre uma categoria.'
 })
}

if(req.body.descricao.length < 3)
{
 erros.push({
 texto: "A descrição da postagem é muito pequena!"
 })
}

if(erros.length > 0){
 res.render("admin/add_postagem", {erros: erros})
} else {
 const novaPostagem = {
 titulo: req.body.titulo,
 slug: req.body.slug,
 descricao: req.body.descricao,
 conteudo: req.body.conteudo,
 categoria: req.body.categoria
 }

 new Postagem(novaPostagem).save().then(() => {
 req.flash("success_msg", "Postagem criada com sucesso!")
 res.redirect("/admin/postagens")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a postagem! Tente novamente.")
 res.redirect("/admin/postagens")
 })
}

module.exports = router

```

## views\admin\postagens.handlebars

```
<h2>Lista de postagens:</h2>
<hr>
<button class="btn btn-success">Nova postagem</button>

{{#each postagens}}
 <div class="card mt-4">
 <div class="card-body">
 <h4>{{titulo}}</h4>
 <p><small>Slug: {{slug}}</small>

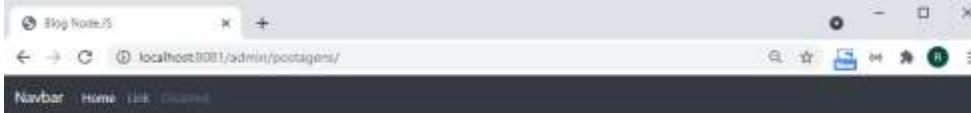
 <small>Categoria: {{categoria.nome}}</small>

 <small>Data de criação: {{data}}</small></p>
 <p>Descrição: {{descricao}}</p>
 <p>Conteúdo: {{conteudo}}</p>
 </div>
 </div>
{{else}}
 <h4 class="mt-3">Nenhuma postagem registrada!</h4>
{{/each}}
```

## nodemon app.js

```
C:\guia_programador\nodejs\blogapp>nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
body-parser deprecated undefined extended: provide extended option app.js:11:22
(node:17460) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(use --trace-deprecation ... to show where the warning was created)
(node:17460) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoC
lient constructor.
Serviço rodando...
Mongo conectado...
```

<http://localhost:8081/admin/postagens>



The screenshot shows a browser window titled 'Blog Notes'. The address bar displays the URL 'localhost:8081/admin/postagens/'. The page content is a list of posts under the heading 'Lista de postagens:'. There is a green button labeled 'Nova postagem'. One post is visible, titled 'Javascript do futuro'. Below the title, it says 'Slug: javascript-future', 'Categoria: Javascript', and 'Data de criação: Sat May 01 2021 00:44:39 GMT-0200 (Horário de Brasília)'. A detailed description follows: 'Descrição: JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares.' and 'Conteúdo: Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. Ela mesma ainda tem grande espaço para evoluir e já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades.'

No Mongo:

**db.postagens.find();**

```
> db.postagens.find();
{
 "_id" : ObjectId("608de33491c8b94a8c3d56b0"),
 "titulo" : "Javascript do futuro",
 "slug" : "javascript-do-futuro",
 "descricao" : "JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares.",
 "conteudo" : "Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. E a mesma ainda tem grande espaço para evoluir e já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades.",
 "categoria" : ObjectId("608ddb01883f192880f859ba"),
 "data" : ISODate("2021-05-01T23:24:36.247Z"),
 "__v" : 0
}
```

## Aula 48 - Editando postagens

routes\admin.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Categoria')
const Categoria = mongoose.model("categorias")
require('../models/Postagem')
const Postagem = mongoose.model("postagens")

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get("/categorias", (req, res) => {
 Categoria.find().sort({date:'desc'}).then((categorias) => {
 res.render("admin/categorias", {categorias: categorias})
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as categorias")
 res.redirect("/admin")
 })
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', (req, res) => {
 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2){
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
})
```

```

if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
} else {
 const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
 }

 new Categoria(novaCategoria).save().then(() => {
 req.flash("success_msg", "Categoria criada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a categoria! Tente novamente.")
 res.redirect("/admin")
 })
}

router.get("/categorias/edit/:id", (req, res) => {
 Categoria.findOne({_id: req.params.id}).then((categoria) => {
 res.render("admin/edit_categoria", {categoria: categoria})
 }).catch((err) => {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/admin/categorias")
 })
})

router.post("/categorias/edit", (req, res) => {

 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2)
 {
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
 if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
 } else {

 Categoria.findOne({_id: req.body.id}).then((categoria) => {

 categoria.nome = req.body.nome
 categoria.slug = req.body.slug

 categoria.save().then(() => {
 req.flash("success_msg", "Categoria editada com sucesso!")
 })
 })
 }
})
}

```

```

 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da categoria")
 res.redirect("/admin/categorias")
 })

}).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a categoria!")
 res.redirect("/admin/categorias")
})

}

})

router.post("/categorias/deletar", (req, res) => {
 Categoria.remove({ _id: req.body.id }).then(() => {
 req.flash("success_msg", "Categoria deletada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao deletar a categoria!")
 res.redirect("/admin/categorias")
 })
})

router.get("/postagens", (req, res) => {
 Postagem.find().lean().populate("categoria").sort({ data: "desc" }).then((postagens) => {
 res.render("admin/postagens", { postagens: postagens })
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao listar as postagens")
 res.redirect("/admin")
 })
})

router.get('/postagens/add', (req, res) => {
 Categoria.find().then((categorias) => {
 res.render('admin/add_postagem', { categorias: categorias })
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao carregar o formulário")
 res.redirect("/admin")
 })
})

router.post('/postagens/nova', (req, res) => {

 var erros = []

 if(!req.body.titulo || typeof req.body.titulo == undefined || req.body.titulo == null){
 erros.push({
 texto: 'Título inválido'
 })
 }

 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
})

```

```

 })
 }

 if(!req.body.descricao || typeof req.body.descricao == undefined || req.body.descricao == null){
 erros.push({
 texto: 'Descrição inválida'
 })
 }

 if(!req.body.conteudo || typeof req.body.conteudo == undefined || req.body.conteudo == null){
 erros.push({
 texto: 'Conteúdo inválido'
 })
 }

 if(req.body.categoria == 0){
 erros.push({
 texto: 'Categoria inválida! Registre uma categoria.'
 })
 }

 if(req.body.descricao.length < 3)
 {
 erros.push({
 texto: "A descrição da postagem é muito pequena!"
 })
 }

 if(erros.length > 0){
 res.render("admin/add_postagem", {erros: erros})
 } else {
 const novaPostagem = {
 titulo: req.body.titulo,
 slug: req.body.slug,
 descricao: req.body.descricao,
 conteudo: req.body.conteudo,
 categoria: req.body.categoria
 }

 new Postagem(novaPostagem).save().then(() => {
 req.flash("success_msg", "Postagem criada com sucesso!")
 res.redirect("/admin/postagens")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a postagem! Tente novamente.")
 res.redirect("/admin/postagens")
 })
 }
}

router.get("/postagens/edit/:id", (req, res) => {
 Postagem.findOne({_id: req.params.id}).lean().then((postagem) => {
 Categoria.find().then((categorias) => {
 res.render("admin/edit_postagem", {categorias: categorias, postagem: postagem})
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao listar as categorias!")
 res.redirect("/admin/postagens")
 })
 })
}

```

```
}).catch((err) => {
 req.flash("error_msg", "Houve um erro ao carregar o formulário de edição!")
 res.redirect("/admin/postagens")
})
})

router.post("/postagens/edit", (req, res) => {

 var erros = []

 Postagem.findOne({_id: req.body.id}).then((postagem) => {

 postagem.titulo = req.body.titulo
 postagem.slug = req.body.slug
 postagem.descricao = req.body.descricao
 postagem.conteudo = req.body.conteudo
 postagem.categoria = req.body.categoria

 postagem.save().then(() => {
 req.flash("success_msg", "Postagem editada com sucesso!")
 res.redirect("/admin/postagens")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da postagem")
 res.redirect("/admin/postagens")
 })
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a postagem!")
 res.redirect("/admin/postagens")
 })
}

module.exports = router
```

## views\admin\postagens.handlebars

```
<h2>Lista de postagens:</h2>
<hr>
<button class="btn btn-success">Nova postagem</button>

{{#each postagens}}
 <div class="card mt-4 mb-2">
 <div class="card-body">
 <h2>{{titulo}}</h2>
 <p class="mt-3">Slug: {{slug}} | Categoria: {{categoria.nome}} | Data de criação: {{data}}</p>
 <p>Descrição: {{descricao}}</p>
 <p>Conteúdo: {{conteudo}}</p>
 <button class="btn btn-success mt-2">Editar</button>
 <form action="/admin/postagens/deletar" method="POST">
 <input type="hidden" name="id" value="{{_id}}>
 <button type="submit" class="btn btn-danger mt-2">Deletar</button>
 </form>
 </div>
 </div>
{{else}}
 <h4 class="mt-3">Nenhuma postagem registrada!</h4>
{{/each}}
```

**views\admin\edit\_postagem.handlebars**

```
{{"#each erros}}
<div class="alert alert-danger">{{texto}}</div>
{{else}}

{{/each}}

<h3>Editar postagem</h3>

<div class="card">
<div class="card-body">
<form action="/admin/postagens/edit" method="POST">
 <input type="hidden" name="id" value="{{postagem._id}}>

 <label for="titulo">Título: </label>
 <input type="text" id="titulo" name="titulo" placeholder="Título da postagem" class="form-control" value="{{postagem.titulo}}>

 <label for="slug">Slug: </label>
 <input type="text" id="slug" name="slug" placeholder="Slug da postagem" class="form-control" value="{{postagem.slug}}>

 <label for="descricao">Descrição: </label>
 <input type="text" id="descricao" name="descricao" placeholder="Título da postagem" class="form-control" value="{{postagem.descricao}}>

 <label for="conteudo">Conteúdo: </label>
 <textarea id="conteudo" name="conteudo" rows="6" class="form-control">{{postagem.conteudo}}</textarea>

 <label for="categoria">Categoria: </label>
 <select id="categoria" name="categoria" class="custom-select mt-2">
 {{#each categorias}}
 <option value="{{_id}}>{{nome}}</option>
 {{else}}
 <option value="0">Nenhuma categoria registrada</option>
 {{/each}}
 </select>

 <button type="submit" class="btn btn-success mt-4">Editar</button>
</form>
</div>
</div>
```

nodemon app.js

```
C:\guia_programador\nodejs\blogapp>nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *
[nodemon] watching extensions: js,mjs,json
[nodemon] starting 'node app.js'
body-parser deprecated undefined extended: provide extended option app.js:31:22
(node:14816) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:14816) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
Servidor rodando!
Mongo conectado...
[]
```

<http://localhost:8081/admin/postagens>

The screenshot shows a web browser window titled "Blog NodeJS". The address bar displays the URL "localhost:8081/admin/postagens". The page content is a list titled "Lista de postagens:" with one item visible:

**Javascript - Uma trilha para o futuro**

**Slug:** javascript-do-futuro | **Categoria:** Javascript | **Data de criação:** Sat May 01 2021 20:24:36 GMT-0300 (Horário Padrão de Brasília)

**Descrição:** JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares.

**Conteúdo:** Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. Ela mesma ainda tem grande espaço para evoluir e já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades.

[Editar](#)

- Clique no botão "editar" e altere o título para "Javascript - Uma trilha para o futuro"

The screenshot shows a web application interface for editing a blog post. The title bar says "Blog NodeJS". The URL is "localhost:8081/admin/postagens/edit/608de33491c8b94a8c3d56b0". The page has a dark header with "Navbar", "Home", "Link", and "Dashboard". Below is a form titled "Editar postagem".  
Fields:

- Título: Javascript - Uma trilha para o futuro
- Slug: javascript-do-futuro
- Descrição: JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares.
- Conteúdo: Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. É a mesma ainda tem grande espaço para evoluir e já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades.
- Categoria: Javascript

An "Editar" button is at the bottom left of the form.

- Clique no botão "Editar"

The screenshot shows a list of posts. A green banner at the top says "Postagem editada com sucesso!". Below is a heading "Lista de postagens:" and a "Nova postagem" button.  
A post is listed:  
**Javascript - Uma trilha para o futuro**  
Slug: javascript-do-futuro | Categoria: PHP | Data de criação: Sat May 01 2021 20:24:36 GMT-0300 (Horário Padrão de Brasília)  
Descrição: JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.  
Conteúdo: Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. É a mesma ainda tem grande espaço para evoluir e já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades.  
Buttons: "Editar" (green) and "Deletar" (red).

## Aula 49 - Deletando postagens

routes\admin.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Categoria')
const Categoria = mongoose.model("categorias")
require('../models/Postagem')
const Postagem = mongoose.model("postagens")

// Definindo as rotas

router.get('/', (req, res) => {
 res.render('admin/index')
})

router.get('/posts', (req, res) => {
 res.send('Página de posts')
})

router.get("/categorias", (req, res) => {
 Categoria.find().sort({date:'desc'}).then((categorias) => {
 res.render("admin/categorias", {categorias: categorias})
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as categorias")
 res.redirect("/admin")
 })
})

router.get('/categorias/add', (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', (req, res) => {
 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2){
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
})
```

```

if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
} else {
 const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
 }

 new Categoria(novaCategoria).save().then(() => {
 req.flash("success_msg", "Categoria criada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a categoria! Tente novamente.")
 res.redirect("/admin")
 })
}
}

router.get("/categorias/edit/:id", (req, res) => {
 Categoria.findOne({_id: req.params.id}).then((categoria) => {
 res.render("admin/edit_categoria", {categoria: categoria})
 }).catch((err) => {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/admin/categorias")
 })
})

router.post("/categorias/edit", (req, res) => {

 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2){
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
 if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
 } else {

 Categoria.findOne({_id: req.body.id}).then((categoria) => {

 categoria.nome = req.body.nome
 categoria.slug = req.body.slug

 categoria.save().then(() => {
 req.flash("success_msg", "Categoria editada com sucesso!")
 })
 })
 }
})
}

```

```

 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da categoria")
 res.redirect("/admin/categorias")
 })

}).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a categoria!")
 res.redirect("/admin/categorias")
})

}

})

router.post("/categorias/deletar", (req, res) => {
 Categoria.remove({ _id: req.body.id }).then(() => {
 req.flash("success_msg", "Categoria deletada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao deletar a categoria!")
 res.redirect("/admin/categorias")
 })
})

router.get("/postagens", (req, res) => {
 Postagem.find().populate("categoria").sort({ date: 'desc' }).then((postagens) => {
 res.render("admin/postagens", { postagens: postagens })
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as postagens")
 res.redirect("/admin")
 })
})

router.get('/postagens/add', (req, res) => {
 Categoria.find().then((categorias) => {
 res.render('admin/add_postagem', { categorias: categorias })
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao carregar o formulário")
 res.redirect("/admin")
 })
})

router.post('/postagens/nova', (req, res) => {

 var erros = []

 if(!req.body.titulo || typeof req.body.titulo == undefined || req.body.titulo == null){
 erros.push({
 texto: 'Título inválido'
 })
 }

 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
})

```

```

 })
 }

 if(!req.body.descricao || typeof req.body.descricao == undefined || req.body.descricao == null){
 erros.push({
 texto: 'Descrição inválida'
 })
 }

 if(!req.body.conteudo || typeof req.body.conteudo == undefined || req.body.conteudo == null){
 erros.push({
 texto: 'Conteúdo inválido'
 })
 }

 if(req.body.categoria == 0){
 erros.push({
 texto: 'Categoria inválida! Registre uma categoria.'
 })
 }

 if(req.body.descricao.length < 3)
 {
 erros.push({
 texto: "A descrição da postagem é muito pequena!"
 })
 }

 if(erros.length > 0){
 res.render("admin/add_postagem", {erros: erros})
 } else {
 const novaPostagem = {
 titulo: req.body.titulo,
 slug: req.body.slug,
 descricao: req.body.descricao,
 conteudo: req.body.conteudo,
 categoria: req.body.categoria
 }

 new Postagem(novaPostagem).save().then(() => {
 req.flash("success_msg", "Postagem criada com sucesso!")
 res.redirect("/admin/postagens")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a postagem! Tente novamente.")
 res.redirect("/admin/postagens")
 })
 }
}
}

router.get("/postagens/edit/:id", (req, res) => {
 Postagem.findOne({_id: req.params.id}).then((postagem) => {
 Categoria.find().then((categorias) => {
 res.render("admin/edit_postagem", {categorias: categorias, postagem: postagem})
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao listar as categorias!")
 })
 })
})
}

```

```

 res.redirect("/admin/postagens")
 })
}).catch((err) => {
 req.flash("error_msg", "Houve um erro ao carregar o formulário de edição!")
 res.redirect("/admin/postagens")
})
}

router.post("/postagens/edit", (req, res) => {
 var erros = []

 Postagem.findOne({_id: req.body.id}).then((postagem) => {
 postagem.titulo = req.body.titulo
 postagem.slug = req.body.slug
 postagem.descricao = req.body.descricao
 postagem.conteudo = req.body.conteudo
 postagem.categoria = req.body.categoria

 postagem.save().then(() => {
 req.flash("success_msg", "Postagem editada com sucesso!")
 res.redirect("/admin/postagens")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da postagem")
 res.redirect("/admin/postagens")
 })
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a postagem!")
 res.redirect("/admin/postagens")
 })
})

router.get("/postagens/deletar/:id", (req, res) => {
 Postagem.remove({_id: req.params.id}).then(() => {
 req.flash("success_msg", "Postagem deletada com sucesso!")
 res.redirect("/admin/postagens")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao deletar a postagem!")
 res.redirect("/admin/postagens")
 })
})

module.exports = router

```

**views\admin\edit\_postagem.handlebars**

```
<h2>Lista de postagens:</h2>
<hr>
<button class="btn btn-success">Nova postagem</button>

{{#each postagens}}
 <div class="card mt-4 mb-2">
 <div class="card-body">
 <h2>{{titulo}}</h2>
 <p>Slug: {{slug}} | Categoria: {{categoria.nome}} | Data de criação: {{data}}</p>
 <p>Descrição: {{descricao}}</p>
 <p>Conteúdo: {{conteudo}}</p>
 <button class="btn btn-success mt-2">Editar</button>
 </div>
 </div>
{{else}}
 <h4 class="mt-3">Nenhuma postagem registrada!</h4>
{{/each}}
```

## views\admin\postagens.handlebars

```
<h2>Lista de postagens:</h2>
<hr>
<button class="btn btn-success">Nova postagem</button>

{{#each postagens}}
<div class="card mt-4">
 <div class="card-body">
 <h4>{{titulo}}</h4>
 <p class="mt-3">Slug: {{slug}} | Categoria: {{categoria.nome}} | Data de criação: {{data}}</p>
 <p>Descrição: {{descricao}}</p>
 <p>Conteúdo: {{conteudo}}</p>
 <button class="btn btn-success mt-2">Editar</button>
 <button class="btn btn-danger mt-2">Deletar</button>
 </div>
</div>
{{else}}
<h4 class="mt-3">Nenhuma postagem registrada!</h4>
{{/each}}
```

- Cadastre uma nova postagem:

Título: O que é o PHP

Slug: o-que-e-o-php

Descrição: PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

Conteúdo: A diferença de PHP com relação a linguagens semelhantes a JavaScript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro. Desta maneira é possível interagir com banco de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.

The screenshot shows a web browser window with the title 'Blog NodeJS'. The address bar displays 'localhost:8081/admin/postagens/add'. The page has a dark header with 'Navbar', 'Home', 'Link', and 'Discussions' buttons. Below the header, the main content area has a title 'Nova postagem'. It contains several input fields and a rich text editor. The 'Título:' field contains 'O que é o PHP'. The 'Slug:' field contains 'o-que-e-o-php'. The 'Descrição:' field contains a short description of PHP. The 'Conteúdo:' field contains a larger block of text about PHP's execution environment and security benefits. At the bottom left is a dropdown menu set to 'PHP', and at the bottom right is a green 'Criar postagem' button.

The screenshot shows a web application titled "Blog NodeJS" at the URL "localhost:8081/admin/postagens". The page has a dark header with "Navbar", "Home", and "Link Disabled" buttons. A green success message box says "Postagem criada com sucesso!". Below it, the heading "Lista de postagens:" is displayed. There are two post entries in a card format:

- O que é o PHP**  
Slug: o-que-e-o-php | Categoria: PHP | Data de criação: Sun May 02 2021 06:08:28 GMT-0300 (Horário Padrão de Brasília)  
Descrição: PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.  
Conteúdo: A diferença de PHP com relação a linguagens semelhantes a JavaScript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro. Desta maneira é possível interagir com banco de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.  
Buttons: Editar (green), Deletar (red)
- Javascript - Uma trilha para o futuro**  
Slug: javascript-do-futuro | Categoria: PHP | Data de criação: Sat May 01 2021 20:24:36 GMT-0300 (Horário Padrão de Brasília)  
Descrição: JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.  
Conteúdo: Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. E a mesma ainda tem grande espaço para evoluir e já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades.  
Buttons: Editar (green), Deletar (red)

- Delete a postagem sobre PHP:

The screenshot shows the same web application at "localhost:8081/admin/postagens". The page displays a green success message box saying "Postagem deletada com sucesso!". Below it, the heading "Lista de postagens:" is shown. The "Javascript - Uma trilha para o futuro" post entry from the previous screenshot is present, but its "Deletar" button has been clicked, resulting in its removal from the list.

## Aula 50 - Criando Home Page

views\partials\\_navbar.handlebars

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
 Blog do Node
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

 </button>

 <div class="collapse navbar-collapse" id="navbarSupportedContent">
 <ul class="navbar-nav mr-auto">
 <li class="nav-item active">
 Home (current)

 <li class="nav-item">
 Categorias

 <li class="nav-item">
 Login

 <li class="nav-item">
 Registro

 </div>
</nav>
```

## app.js

```
/* Carregando módulos */

const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
const app = express()
const admin = require('./routes/admin')
const path = require('path')
const mongoose = require('mongoose')
const session = require('express-session')
const flash = require('connect-flash')
require('./models/Postagem')
const Postagem = mongoose.model("postagens")

/* Configurações */

// Session
app.use(session({
 secret: "cursodenode",
 resave: true,
 saveUninitialized: true
}))
app.use(flash())

// Middleware
app.use((req, res, next) => {
 res.locals.success_msg = req.flash("success_msg")
 res.locals.error_msg = req.flash("error_msg")
 next()
})

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/blogapp").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});

// Public
app.use(express.static(path.join(__dirname, "public")))
app.use((req, res, next) => {
 console.log('Oi, eu sou um middleware!')
 next()
})
```

```
/* Rotas */

app.get('/', (req, res) => {
 Postagem.find().populate("Categoria").lean().sort({data: "desc"}).then((postagens) => {
 res.render("index", {postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro interno!")
 res.redirect('/404')
 })
})

app.get('/404', (req, res) => {
 res.send('Erro 404!')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)

/* Outros */

const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

## views\index.handlebars

```
<div class="jumbotron">
 <h1 class="display-4">Bem-vindo ao Blog do Node!</h1>
 <p class="lead">This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.</p>
 <hr class="my-4">
 <p>It uses utility classes for typography and spacing to space content out within the larger container.</p>
 Crie uma conta
</div>

<hr>

<h2>Postagens recentes</h2>

{{#each postagens}}
 <div class="card">
 <div class="card-body">
 <h3>{{titulo}}</h3>
 <p>{{descricao}}</p>
 </div>
 </div>
{{else}}
 <h5>Nenhuma postagem</h5>
{{/each}}
```

- Crie uma categoria chamada "**Desenvolvimento Web**"



The screenshot shows a web browser window with the URL `localhost:8081/admin/categories/add`. The page title is "Blog do Node". The main content area has a heading "Nova categoria". There are two input fields: "Nome" containing "Desenvolvimento Web" and "Slug" containing "desenvolvimento-web". A green button labeled "Criar categoria" is at the bottom.

Categoria criada com sucesso!

## Lista de categorias:

[Nova categoria](#)

### Desenvolvimento Web

Slug: desenvolvimento-web

Data de criação: Sun May 02 2021 07:41:03 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

[Deletar](#)

### PHP

Slug: php

Data de criação: Sat May 01 2021 19:20:09 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

[Deletar](#)

### Javascript

Slug: javascript

Data de criação: Sat May 01 2021 19:20:02 GMT-0300 (Horário Padrão de Brasília)

[Editar](#)

[Deletar](#)

- Crie mais duas postagens:

Título: O que é o PHP

Slug: o-que-e-o-php

Descrição: PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

Conteúdo: A diferença de PHP com relação a linguagens semelhantes a JavaScript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro. Desta maneira é possível interagir com banco de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.

The screenshot shows a web browser window with the URL `localhost:8081/admin/postagens/add` in the address bar. The page title is "Blog do Node". The main content area is titled "Nova postagem" (New Post). The form fields are as follows:

- Título:** O que é o PHP
- Slug:** o-que-e-o-php
- Descrição:** PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL.
- Conteúdo:** A diferença de PHP com relação a linguagens semelhantes a JavaScript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro. Desta maneira é possível interagir com banco de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.

At the bottom of the form is a button labeled "Criar postagem" (Create post).

Postagem criada com sucesso!

## Lista de postagens:

[Nova postagem](#)

### O que é o PHP

**Slug:** o-que-e-o-php | **Categoria:** PHP | **Data de criação:** Sun May 02 2021 09:08:05 GMT-0300 (Horário Padrão de Brasília)

Descrição: PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

Conteúdo: A diferença de PHP com relação a linguagens semelhantes a JavaScript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro. Desta maneira é possível interagir com banco de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.

[Editar](#)

[Deletar](#)

### Javascript - Uma trilha para o futuro

**Slug:** javascript-do-futuro | **Categoria:** Javascript | **Data de criação:** Sat May 01 2021 20:24:36 GMT-0300 (Horário Padrão de Brasília)

Descrição: JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

Conteúdo: Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. E a mesma ainda tem grande espaço para evoluir já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades.

[Editar](#)

[Deletar](#)

Título: Personagens principais no desenvolvimento Web

Slug: personagens-principais-no-desenvolvimento-web

Descrição: O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

Conteúdo: O W3C cuida dos padrões. Ele tem ideias, ele prevê problemas e tenta solucioná-los. Ele incorpora boas ideias dos desenvolvedores e principalmente dos browsers. O W3C não aplica, ele apenas planeja. É um trabalho difícil porque é necessário uma visão muito apurada da web. Uma decisão errada pode acarretar problemas que levarão anos para serem solucionados. Por isso esse trabalho de planejamento deve ser meticuloso. Devo confessar que em muitos casos o W3C não supera as expectativas e faz com que iniciativas paralelas surjam e direcionem a Web para um caminho mais correto. Foi o que aconteceu com o HTML 5. Os browsers, por sua vez, precisam entender e adotar as ideias do W3C absorvendo as soluções e criando suporte nos seus softwares. Esse trabalho também tem seus perigos. Os browsers precisam pesquisar quais das necessidades dos desenvolvedores é mais importante e assim implementá-las. Não pode ser algo feito ao acaso. Os browsers precisam ouvir os desenvolvedores para implementar soluções que realmente ajudem na produção diária. Obviamente alguns decidem suportar

The screenshot shows a web browser window with the URL `localhost:8081/admin/postagens/adit`. The page title is "Nova postagem". The form fields are as follows:

- Título:** Personagens principais no desenvolvimento Web
- Slug:** personagens-principais-no-desenvolvimento-web
- Descrição:** O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.
- Conteúdo:** O W3C cuida dos padrões. Ele tem ideias, ele prevê problemas e tenta solucioná-los. Ele incorpora boas ideias dos desenvolvedores e principalmente dos browsers. O W3C não aplica, ele apenas planeja. É um trabalho difícil porque é necessário uma visão muito apurada da web. Uma decisão errada pode acarretar problemas que levarão anos para serem solucionados. Por isso esse trabalho de planejamento deve ser meticuloso. Devo confessar que em muitos casos o W3C não supera as expectativas e faz com que iniciativas paralelas surjam e direcionem a Web para um caminho mais correto. Foi o que aconteceu com o HTML 5. Os browsers, por sua vez, precisam entender e adotar as ideias do W3C absorvendo as soluções e criando suporte nos seus softwares. Esse trabalho também tem seus perigos. Os browsers precisam pesquisar quais das necessidades dos desenvolvedores é mais importante e assim implementá-las. Não pode ser algo feito ao acaso. Os browsers precisam ouvir os desenvolvedores para implementar soluções que realmente ajudem na produção diária. Obviamente alguns decidem suportar
- Categorias:** Desenvolvimento Web
- Botão:** Criar postagem

Postagem criada com sucesso!

## Lista de postagens:

[Nova postagem](#)

### Personagens principais no desenvolvimento Web

**Slug:** personagens-principais-no-desenvolvimento-web | **Categoria:** Desenvolvimento Web | **Data de criação:** Sun May 02 2021 09:15:53 GMT-0300 (Horário Padrão de Brasília)

**Descrição:** O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

**Conteúdo:** O W3C cuida dos padrões. Ele tem ideias, ele prevê problemas e tenta solucioná-los. Ele incorpora boas ideias dos desenvolvedores e principalmente dos browsers. O W3C não aplica, ele apenas planeja. É um trabalho difícil porque é necessário uma visão muito apurada da web. Uma decisão errada pode acarretar problemas que levarão anos para serem solucionados. Por isso esse trabalho de planejamento deve ser meticoloso. Devo confessar que em muitos casos o W3C não supera as expectativas e faz com que iniciativas paralelas surjam e direcionem a Web para um caminho mais correto. Foi o que aconteceu com o HTML 5. Os browsers, por sua vez, precisam entender e adotar as ideias do W3C absorvendo as soluções e criando suporte nos seus softwares. Esse trabalho também tem seus perigos. Os browsers precisam pesquisar quais das necessidades dos desenvolvedores é mais importante e assim implementá-las. Não pode ser algo feito ao acaso. Os browsers precisam ouvir os desenvolvedores para implementar soluções que realmente ajudem na produção diária. Obviamente alguns decidem suportar aquelas soluções que darão mais pontos estratégicos contra o concorrente. Por que você acha que o Safari tem um bom suporte aos padrões desde o primeiro iPhone? Finalizando o ciclo, os desenvolvedores aplicam tudo o que o W3C define, mas apenas aquilo que os browsers 'querem' ou podem suportar. E isso, claro, faz com que o desenvolvedor tenha problemas na produção. Principalmente de compatibilidade. Nesse ponto, hoje está tudo muito mais tranquilo.

[Editar](#) [Deletar](#)

### O que é o PHP

**Slug:** o-que-e-o-php | **Categoria:** PHP | **Data de criação:** Sun May 02 2021 09:08:05 GMT-0300 (Horário Padrão de Brasília)

**Descrição:** PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

**Conteúdo:** A diferença de PHP com relação a linguagens semelhantes a JavaScript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro. Desta maneira é possível interagir com banco de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial.

[Editar](#) [Deletar](#)

### Javascript - Uma trilha para o futuro

**Slug:** javascript-do-futuro | **Categoria:** Javascript | **Data de criação:** Sat May 01 2021 20:24:38 GMT-0300 (Horário Padrão de Brasília)

**Descrição:** JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

**Conteúdo:** Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. E a mesma ainda tem grande espaço para evoluir e já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades.

[Editar](#) [Deletar](#)

<http://localhost:8081/>

The screenshot shows the homepage of a blog titled "Blog do Node". The header includes navigation links for "Blog do Node", "Home", "Categorias", "Login", and "Registro". The main content features a hero unit with the heading "Bem-vindo ao Blog do Node!" and a subtext explaining it's a jumbotron-style component. A blue button labeled "Crie uma conta" is visible. Below the hero unit, there's a section titled "Postagens recentes" containing three cards: "Personagens principais no desenvolvimento Web", "O que é o PHP", and "Javascript - Uma trilha para o futuro". Each card has a brief description and a link.

Blog do Node Home Categorias Login Registro

# Bem-vindo ao Blog do Node!

This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.

It uses utility classes for typography and spacing to space content out within the larger container.

Crie uma conta

## Postagens recentes

**Personagens principais no desenvolvimento Web**

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

**O que é o PHP**

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

**Javascript - Uma trilha para o futuro**

Javascript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

## Aula 51 - Página de postagens

views\index.handlebars

```
<div class="jumbotron">
 <h1 class="display-4">Bem-vindo ao Blog do Node!</h1>
 <p class="lead">This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.</p>
 <hr class="my-4">
 <p>It uses utility classes for typography and spacing to space content out within the larger container.</p>
 Crie uma conta
</div>

<hr>

<h2>Postagens recentes</h2>

{{#each postagens}}
 <div class="card mt-3 mb-3">
 <div class="card-body">
 <h3>{{titulo}}</h3>
 <p>{{descricao}}</p>
 <button class="btn btn-success">Leia mais</button>
 <hr>
 <small>Categoria: {{categoria.nome}}</small>
 <small>Data de publicação: {{data}}</small>
 </div>
 </div>
{{else}}
 <h5>Nenhuma postagem</h5>
{{/each}}
```

## app.js

```
/* Carregando módulos */

const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
const app = express()
const admin = require('./routes/admin')
const path = require('path')
const mongoose = require('mongoose')
const session = require('express-session')
const flash = require('connect-flash')
require('./models/Postagem')
const Postagem = mongoose.model("postagens")

/* Configurações */

// Session
app.use(session({
 secret: "cursodenode",
 resave: true,
 saveUninitialized: true
}))
app.use(flash())

// Middleware
app.use((req, res, next) => {
 res.locals.success_msg = req.flash("success_msg")
 res.locals.error_msg = req.flash("error_msg")
 next()
})

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/blogapp").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});

// Public
app.use(express.static(path.join(__dirname, "public")))
app.use((req, res, next) => {
 console.log('Oi, eu sou um middleware!')
 next()
})
```

```
/* Rotas */

app.get('/', (req, res) => {
 Postagem.find().populate("categoria").sort({data: "desc"}).then((postagens) => {
 res.render("index", {postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro interno!")
 res.redirect('/404')
 })
})

app.get('/postagem/:slug', (req, res) => {
 Postagem.findOne({slug: req.params.slug}).lean().then((postagem) =>{
 if(postagem){
 res.render("postagem/index", {postagem: postagem})
 } else {
 req.flash("error_msg", "Essa postagem não existe!")
 res.redirect("/")
 }
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
 })
})

app.get('/404', (req, res) => {
 res.send('Erro 404!')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)

/* Outros */

const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

## views\postagem\index.handlebars

```
<div class="card">
 <div class="card-body">
 <h1>{{postagem.titulo}}</h1>
 <hr>
 <small>Data da publicação: {{postagem.data}}</small>
 <hr>
 <p>Descrição: {{postagem.descricao}}</p>
 <hr>
 <p>{{postagem.conteudo}}</p>
 </div>
</div>
```

## views\partials\\_navbar.handlebars

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
 Blog do Node
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

 </button>

 <div class="collapse navbar-collapse" id="navbarSupportedContent">
 <ul class="navbar-nav mr-auto">
 <li class="nav-item active">
 Home (current)

 <li class="nav-item">
 Categorias

 <li class="nav-item">
 Login

 <li class="nav-item">
 Registro

 </div>
</nav>
```

# Bem-vindo ao Blog do Node!

This is a simple hero unit, a simplejumbotron-style component for calling extra attention to featured content or information.

It uses utility classes for typography and spacing to space content out within the larger container.

Crie uma conta

## Postagens recentes

## Personagens principais no desenvolvimento Web

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões da forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

Lec man

Categoria: Desenvolvimento Web

Data de publicação: 1uv May-02 2021 09:15:53 (GMT-03:00) (Brasília)

## O que é o PHP

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links.

Lipid profile

Categoria: 2013

Data de publicação: 1st May 2021 09:08:06 (GMT+0300 (Horário Padrão de Brasília))

Javascript - Uma trilha para o futuro

JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

Lektionen

## Categorie: Java script

Data de publicação: Sat May 09 2021 20:24:46 GMT-0300 (Horário Padrão de Brasília)

## Personagens principais no desenvolvimento Web

Data da publicação: Sun, May 01 2021 08:12:53 GMT-0300 (Horário Padrão de Brasília)

**Descrição:** O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões da forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

O W3C cuida dos padrões. Ele tem ideias, ele prevê problemas e tenta solucioná-los. Ele incorpora boas ideias dos desenvolvedores e principalmente dos browsers. O W3C não aplica, ele apenas planeja. É um trabalho difícil porque é necessário uma visão muito avançada da web. Uma decisão errada pode acarretar problemas que levarão anos para serem solucionados. Por isso esse trabalho de planejamento deve ser meticuloso. Deve confessar que em muitos casos o W3C não supera as expectativas e faz com que iniciativas paralelas surjam e direcionem a Web para um caminho mais correto. Foi o que aconteceu com o HTML5. Os browsers, por sua vez, precisam entender e adotar as ideias do W3C absorvendo as soluções e criando suporte nos seus softwares. Esse trabalho também tem seu pênis. Os browsers precisam pesquisar quais das necessidades dos desenvolvedores e mais importante e assim implementá-las. Não pode ser algo feito ao acaso. Os browsers precisam unir os desenvolvedores para implementar soluções que realmente ajudem na produção clara. Claramente alguns decidem suportar aquelas soluções que dão mais pontos estratégicos contra o concorrente. Por que você acha que o Safari tem um bom suporte aos padrões desde o primeiro iPhone? Finalizando o ciclo, os desenvolvedores aplicam tudo o W3C define, mas apenas aquilo que os browsers ‘querem’ ou podem suportar. E isso, claro, faz com que o desenvolvedor tenha problemas na produção. Principalmente de compatibilidade. Nesse ponto, hoje está tudo muito mais tranquilo.

A screenshot of a web browser window. The address bar shows the URL: `localhost:8081/postagem/o-que-e-o-php`. The page title is "Blog do Node". The main content is an article titled "O que é o PHP". Below the title is a timestamp: "Data da publicação: Sat May 01 2021 09:08:08 GMT-0300 (Horário Padrão de Brasília)". A description follows: "Descrição: PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links." At the bottom of the article is a note: "A diferença de PHP com relação a linguagens semelhantes a JavaScript é que o código PHP é executado no servidor, sendo enviado para o cliente apenas html puro. Desta maneira é possível interagir com banco de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. Isso pode ser útil quando o programa está lidando com senhas ou qualquer tipo de informação confidencial." The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with icons for refresh, save, and others.

A screenshot of a web browser window. The address bar shows the URL: `localhost:8081/postagem/javascript-do-futuro`. The page title is "Blog do Node". The main content is an article titled "Javascript - Uma trilha para o futuro". Below the title is a timestamp: "Data da publicação: Sat May 01 2021 10:04:36 GMT-0300 (Horário Padrão de Brasília)". A description follows: "Descrição: JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo." At the bottom of the article is a note: "Pode-se projetar um futuro onde haja muitos recursos e ferramentas desenvolvidas por esta linguagem. E a mesma ainda tem grande espaço para evoluir e já está presente em todos os tipos de dispositivos eletrônicos. Por este crescimento exponencial, se espera que esteja cada vez mais incluída não apenas nas comunidades tecnológicas, mas nas mais diversas comunidades." The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with icons for refresh, save, and others.

## Aula 52 - Listagem de categorias

### app.js

```
/* Carregando módulos */

const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
const app = express()
const admin = require('./routes/admin')
const path = require('path')
const mongoose = require('mongoose')
const session = require('express-session')
const flash = require('connect-flash')
require('./models/Postagem')
const Postagem = mongoose.model("postagens")
require('./models/Categoria')
const Categoria = mongoose.model("categorias")

/* Configurações */

// Session
app.use(session({
 secret: "cursodenode",
 resave: true,
 saveUninitialized: true
}))
app.use(flash())

// Middleware
app.use((req, res, next) => {
 res.locals.success_msg = req.flash("success_msg")
 res.locals.error_msg = req.flash("error_msg")
 next()
})

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/blogapp").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});
```

```

// Public
app.use(express.static(path.join(__dirname, "public")))
app.use((req, res, next) => {
 console.log('Oi, eu sou um middleware!')
 next()
})

/* Rotas */

app.get('/', (req, res) => {
 Postagem.find().populate("categoria").sort({data: "desc"}).then((postagens) => {
 res.render("index", {postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro interno!")
 res.redirect('/404')
 })
})

app.get('/postagem/:slug', (req, res) => {
 Postagem.findOne({slug: req.params.slug}).then((postagem) =>{
 if(postagem){
 res.render("postagem/index", {postagem: postagem})
 } else {
 req.flash("error_msg", "Essa postagem não existe!")
 res.redirect("/")
 }
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
 })
})

app.get('/categorias', (req, res) => {
 Categoria.find().lean().then((categorias) => {
 res.render("categorias/index", {categorias: categorias})
 }).catch(() => {
 req.flash("error_msg", "Erro interno ao listar categorias!")
 res.redirect("/")
 })
})

app.get('/categorias/:slug', (req, res) => {
 Categoria.findOne({slug: req.params.slug}).then((categoria) => {
 if(categoria){
 Postagem.find({categoria: categoria._id}).lean().then((postagens) => {
 res.render("categorias/postagens", {categoria: categoria, postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro ao listar os posts")
 res.redirect("/")
 })
 } else {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/")
 }
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao carregar a página dessa categoria!")
 })
})

```

```
 res.redirect("/")
 })
}

app.get('/404', (req, res) => {
 res.send('Erro 404!')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)

/* Outros */

const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

## views\categorias\index.handlebars

```
<div class="card">
 <div class="card-body">
 <h2>Categorias:</h2>
 <hr>

 {{#each categorias}}
 <h5>{{nome}}</h5>
 {{else}}
 {{/each}}

 </div>
</div>
```

## views\categorias\postagens.handlebars

```
<h1>{{categoria.nome}}</h1>
{{#each postagens}}
 <div class="card mt-3 mb-3">
 <div class="card-body">
 <h3>{{titulo}}</h3>
 <p>{{descricao}}</p>
 <button class="btn btn-success">Leia mais</button>
 <hr>
 <small>Data de publicação: {{data}}</small>
 </div>
 </div>
{{else}}
 <h5>Nenhuma postagem</h5>
{{/each}}
```

## views\partials\\_navbar.handlebars

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
 Blog do Node
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent">
 aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

 </button>

 <div class="collapse navbar-collapse" id="navbarSupportedContent">
 <ul class="navbar-nav mr-auto">
 <li class="nav-item active">
 Home (current)

 <li class="nav-item">
 Categorias

 <li class="nav-item">
 Login

 <li class="nav-item">
 Registro

 </div>
</nav>
```

- Clique no link "Categorias" (no menu superior):

Categorias:

- PHP
- Javascript
- Desenvolvimento Web

- Clicando na categoria "PHP"

O que é o PHP

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links.

[Leia mais](#)

Data de publicação: Sun May 02 2021 09:08:03 GMT-0300 (Horário Padrão de Brasília)

- Clicando na categoria "Javascript"

Javascript - Uma trilha para o futuro

JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

[Leia mais](#)

Data de publicação: Sat May 01 2021 20:14:36 GMT-0300 (Horário Padrão de Brasília)

- Clicando na categoria "Desenvolvimento Web"

Personagens principais no desenvolvimento Web

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

[Leia mais](#)

Data de publicação: Sun May 02 2021 09:10:53 GMT-0300 (Horário Padrão de Brasília)

## Aula 53 - Definindo model de usuário

### models/Usuario.js

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

// Model - Usuario

const Usuario = new Schema({
 nome: {
 type: String,
 required: true
 },
 email: {
 type: String,
 required: true
 },
 senha: {
 type: String,
 required: true
 }
})

// Collection

mongoose.model('usuarios', Usuario)
```

## Aula 54 - Registro de usuários

routes\usuario.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Usuario')
const Usuario = mongoose.model("usuarios")

router.get('/registro', (req, res) => {
 res.render("usuarios/registro")
})

router.post('/registro', (req, res) => {
 var erros = [];

 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({texto: "Nome inválido!"})
 }

 if(!req.body.email || typeof req.body.email == undefined || req.body.email == null){
 erros.push({texto: "Email inválido!"})
 }

 if(!req.body.senha || typeof req.body.senha == undefined || req.body.senha == null){
 erros.push({texto: "Senha inválida!"})
 }

 if(req.body.senha.length < 4){
 erros.push({texto: "A senha deve ter no mínimo 4 caracteres!"})
 }

 if(req.body.senha2 != req.body.senha){
 erros.push({texto: "As senhas digitadas são diferentes!"})
 }

 if(erros.length > 0){
 res.render("usuarios/registro", {erros: erros})
 } else {

 }
})

module.exports = router
```

## app.js

```
/* Carregando módulos */

const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
const app = express()
const admin = require('./routes/admin')
const usuarios = require('./routes/usuario')
const path = require('path')
const mongoose = require('mongoose')
const session = require('express-session')
const flash = require('connect-flash')
require('./models/Postagem')
const Postagem = mongoose.model("postagens")
require('./models/Categoria')
const Categoria = mongoose.model("categorias")

/* Configurações */

// Session
app.use(session({
 secret: "cursodenode",
 resave: true,
 saveUninitialized: true
}))
app.use(flash())

// Middleware
app.use((req, res, next) => {
 res.locals.success_msg = req.flash("success_msg")
 res.locals.error_msg = req.flash("error_msg")
 next()
})

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/blogapp").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});

// Public
app.use(express.static(path.join(__dirname, "public")))
app.use((req, res, next) => {
```

```

console.log('Oi, eu sou um middleware!')
next()
})

/* Rotas */

app.get('/', (req, res) => {
 Postagem.find().populate("categoria").sort({data: "desc"}).then((postagens) => {
 res.render("index", {postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro interno!")
 res.redirect('/404')
 })
})

app.get('/postagem/:slug', (req, res) => {
 Postagem.findOne({slug: req.params.slug}).then((postagem) =>{
 if(postagem){
 res.render("postagem/index", {postagem: postagem})
 } else {
 req.flash("error_msg", "Essa postagem não existe!")
 res.redirect("/")
 }
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
 })
})

app.get('/categorias', (req, res) => {
 Categoria.find().then((categorias) => {
 res.render("categorias/index", {categorias: categorias})
 }).catch(() => {
 req.flash("error_msg", "Erro interno ao listar categorias!")
 res.redirect("/")
 })
})

app.get('/categorias/:slug', (req, res) => {
 Categoria.findOne({slug: req.params.slug}).then((categoria) => {
 if(categoria){
 Postagem.find({categoria: categoria._id}).then((postagens) => {
 res.render("categorias/postagens", {categoria: categoria, postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro ao listar os posts")
 res.redirect("/")
 })
 } else {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/")
 }
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao carregar a página dessa categoria!")
 res.redirect("/")
 })
})
}
)

```

```
app.get('/404', (req, res) => {
 res.send('Erro 404!')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)
app.use('/usuarios', usuarios)

/* Outros */

const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

## views\usuarios\registro.handlebars

```
{{#each erros}}
 <div class="alert alert-danger">{{texto}}</div>
{{else}}

{{/each}}

<h3>Criar conta</h3>
<div class="card">
 <div class="card-body">

 <form action="" method="POST">
 <label for="nome">Nome: </label>
 <input type="text" id="nome" name="nome" placeholder="Digite o seu nome" class="form-control mb-1">

 <label for="email">Email: </label>
 <input type="email" id="email" name="email" placeholder="Digite o seu email" class="form-control mb-1">

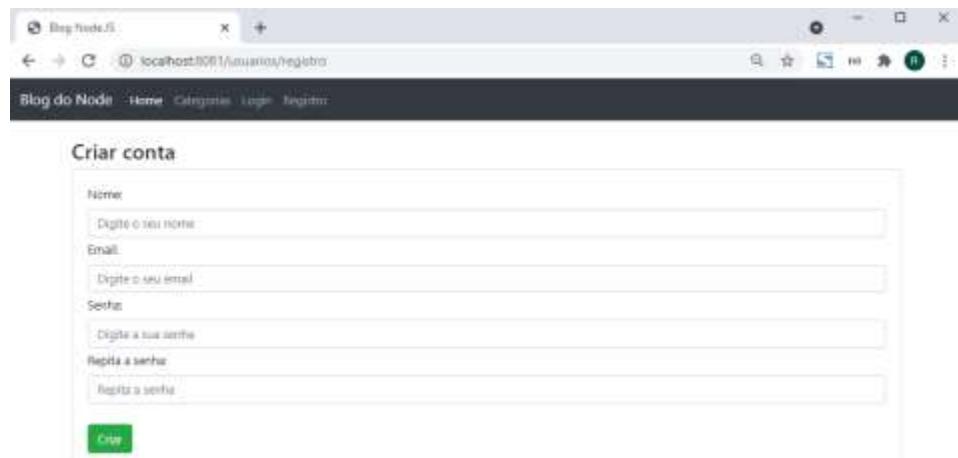
 <label for="senha">Senha: </label>
 <input type="password" id="senha" name="senha" placeholder="Digite a sua senha" class="form-control mb-1">

 <label for="senha2">Repita a senha: </label>
 <input type="password" id="senha2" name="senha2" placeholder="Repita a senha" class="form-control mb-1">

 <button type="submit" class="btn btn-success mt-4">Criar</button>
 </form>

 </div>
</div>
```

<http://localhost:8081/usuarios/registro>



## Testando validações

The screenshot shows a web browser window with the URL `localhost:8081/usuarios/registro`. The page title is "Blog do Node". The main content is a "Criar conta" (Create account) form. It includes fields for Name, Email, Password, and Password Confirmation. Below the form, there is a green "Criar" button.

Nome:  
Digite o seu nome

Email:  
Digite o seu email

Senha:  
\*\*\*\*\*

Repita a senha:  
\*\*\*\*\*

**Criar**

The screenshot shows a web browser window with the URL `localhost:8081/usuarios/registro`. The page title is "Blog do Node". The main content is a "Criar conta" (Create account) form. Above the form, three error messages are displayed in pink boxes: "Nome inválido!", "Email inválido!", and "As senhas digitadas são diferentes!". Below the error messages, the registration form fields are shown again.

Nome inválido!

Email inválido!

As senhas digitadas são diferentes!

Nome:  
Digite o seu nome

Email:  
Digite o seu email

Senha:  
Digite a sua senha

Repita a senha:  
Repita a senha

**Criar**

## Aula 55 - Fazendo correção

### models\Usuario.js

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

// Model - Usuario

const Usuario = new Schema({
 nome: {
 type: String,
 required: true
 },
 email: {
 type: String,
 required: true
 },
 eAdmin: {
 type: Number,
 default: 0
 },
 senha: {
 type: String,
 required: true
 }
})

// Collection

mongoose.model('usuarios', Usuario)
```

## Aula 56 - Cadastrando usuários no BD

### Instalando o bcrypt

```
npm install bcryptjs --save
```

```
C:\guia_programador\nodejs\blogapp>npm install bcryptjs --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm WARN blogapp No description
npm WARN blogapp No repository field.
npm WARN blogapp No README data
npm WARN blogapp No license field.

+ bcryptjs@2.4.3
added 1 package from 6 contributors and audited 182 packages in 6.967s

3 packages are looking for funding
 run `npm fund` for details

found 0 vulnerabilities
```

### routes\usuario.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Usuario')
const Usuario = mongoose.model("usuarios")
const bcrypt = require('bcryptjs')

router.get('/registro', (req, res) => {
 res.render("usuarios/registro")
})

router.post('/registro', (req, res) => {
 var erros = [];

 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({texto: "Nome inválido!"})
 }

 if(!req.body.email || typeof req.body.email == undefined || req.body.email == null){
 erros.push({texto: "Email inválido!"})
 }

 if(!req.body.senha || typeof req.body.senha == undefined || req.body.senha == null){
 erros.push({texto: "Senha inválida!"})
 }

 if(req.body.senha.length < 4){
 erros.push({texto: "A senha deve ter no mínimo 4 caracteres!"})
 }

 if(req.body.senha2 != req.body.senha){
 erros.push({texto: "As senhas digitadas são diferentes!"})
 }
})
```

```

}

if(erros.length > 0){
 res.render("usuarios/registro", {erros: erros})
} else {
 Usuario.findOne({email: req.body.email}).then((usuario) => {
 if(usuario){
 req.flash("error_msg", "Já existe uma conta com este email no nosso sistema!")
 res.redirect("/usuarios/registro")
 } else {
 const novoUsuario = new Usuario({
 nome: req.body.nome,
 email: req.body.email,
 senha: req.body.senha
 })

 bcrypt.genSalt(10, (erro, salt) => {
 bcrypt.hash(novoUsuario.senha, salt, (erro, hash) => {
 if(erro){
 req.flash("error_msg", "Houve um erro durante o salvamento do usuário!")
 res.redirect("/")
 }

 novoUsuario.senha = hash

 novoUsuario.save().then(() => {
 req.flash("success_msg", "Usuário criado com sucesso!")
 res.redirect("/")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao criar o usuário, tente novamente!")
 res.redirect("/usuarios/registro")
 })
 })
 })
 }
 })
}

}).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
})
}

module.exports = router

```

nodemon app.js

```
C:\guia_programador\nodejs\blogapp>nodemon app.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting node app.js
body-parser deprecated undefined extended: provide extended option app.js:36:22
(node:18228) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:18228) [MONGOO8 DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
Servidor rodando!
Mongo conectado...
```

<http://localhost:8081/usuarios/registro>

senha: 123456

The screenshot shows a web browser window with the title 'Blog NodeJS'. The address bar displays the URL 'localhost:8081/usuarios/registro'. The page content is a registration form titled 'Criar conta' (Create account). The form fields are as follows:

- Nome: Victor Lima
- Email: victor\_lima@gmail.com
- Senha: (redacted)
- Repita a senha: (redacted)

A green 'Criar' (Create) button is located at the bottom left of the form.



## Postagens recentes

### Personagens principais no desenvolvimento Web

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e mantém padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

[Leia mais](#)

Categoria: Desenvolvimento Web

Data da publicação: Sun May 02 2021 09:55:52 GMT-0200 (Horário Padrão de Brasília)

### O que é o PHP

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

[Leia mais](#)

Categoria: PHP

Data da publicação: Sun May 02 2021 09:00:03 GMT-0200 (Horário Padrão de Brasília)

### Javascript - Uma trilha para o futuro

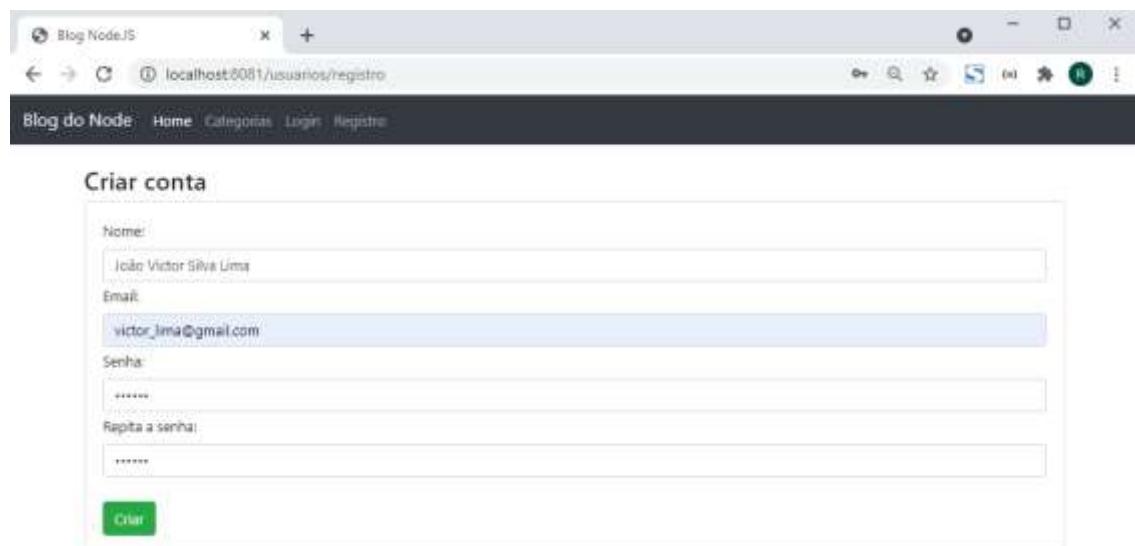
JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

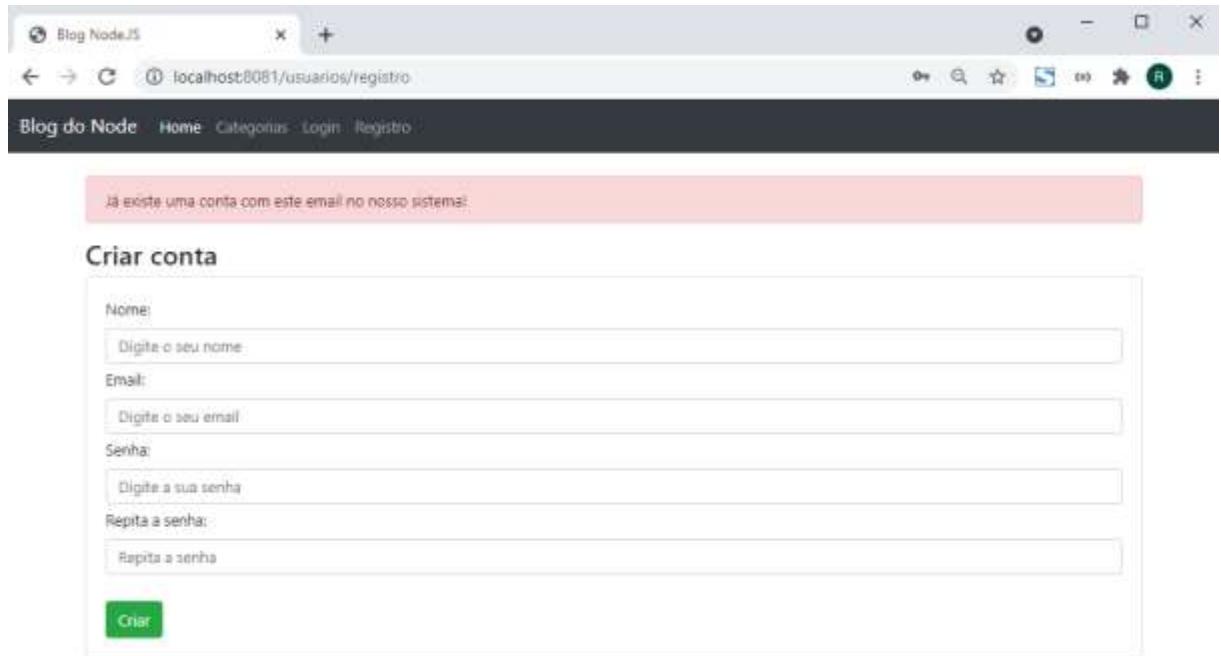
[Leia mais](#)

Categoria: Javascript

Data da publicação: Sat May 01 2021 20:24:58 GMT-0200 (Horário Padrão de Brasília)

## Tentando criar um usuário com um email já cadastrado





- No Mongo:

```
> show databases;
admin 0.000GB
blogapp 0.000GB
config 0.000GB
local 0.000GB
testes_mongo 0.000GB
> use blogapp;
switched to db blogapp
> show collections;
categorias
postagens
usuarios
> db.usuarios.find();
{ "_id" : ObjectId("608f3f00b6d0e84734619ef2"), "eAdmin" : 0, "nome" : "Victor Lima", "email" : "victor_lima@gmail.com",
"senha" : "$2a$10$TlpPJD880H7RvYwdo1Xp00pIf0f.bK6wa5C1qzjLGUVYdxjNs9wy", "__v" : 0 }
```

## Tentando cadastrar outro usuário com o mesmo email

Criar conta

Nome:  
Luis Victor Lima

Email:  
victor\_lima@gmail.com

Senha:  
\*\*\*\*\*

Repita a senha:  
\*\*\*\*\*

**Criar**

Já existe uma conta com este email no nosso sistema.

Criar conta

Nome:  
Digite o seu nome

Email:  
Digite o seu email

Senha:  
Digite a sua senha

Repita a senha:  
Repita a senha

**Criar**

## Aula 57 - Formulário de Login

routes\usuario.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Usuario')
const Usuario = mongoose.model("usuarios")
const bcrypt = require('bcryptjs')

router.get('/registro', (req, res) => {
 res.render("usuarios/registro")
})

router.post('/registro', (req, res) => {
 var erros = [];

 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({texto: "Nome inválido!"})
 }

 if(!req.body.email || typeof req.body.email == undefined || req.body.email == null){
 erros.push({texto: "Email inválido!"})
 }

 if(!req.body.senha || typeof req.body.senha == undefined || req.body.senha == null){
 erros.push({texto: "Senha inválida!"})
 }

 if(req.body.senha.length < 4){
 erros.push({texto: "A senha deve ter no mínimo 4 caracteres!"})
 }

 if(req.body.senha2 != req.body.senha){
 erros.push({texto: "As senhas digitadas são diferentes!"})
 }

 if(erros.length > 0){
 res.render("usuarios/registro", {erros: erros})
 } else {
 Usuario.findOne({email: req.body.email}).then((usuario) => {
 if(usuario){
 req.flash("error_msg", "Já existe uma conta com este email no nosso sistema!")
 res.redirect("/usuarios/registro")
 } else {
 const novoUsuario = new Usuario({
 nome: req.body.nome,
 email: req.body.email,
 senha: req.body.senha
 })

 bcrypt.genSalt(10, (erro, salt) => {
```

```

bcrypt.hash(novoUsuario.senha, salt, (erro, hash) => {
 if(erro){
 req.flash("error_msg", "Houve um erro durante o salvamento do usuário!")
 res.redirect("/")
 }

 novoUsuario.senha = hash

 novoUsuario.save().then(() => {
 req.flash("success_msg", "Usuário criado com sucesso!")
 res.redirect("/")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao criar o usuário, tente novamente!")
 res.redirect("/usuarios/registro")
 })
}

})
})

}).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
})
})

router.get('/login', (req, res) => {
 res.render("usuarios/login")
})

module.exports = router

```

## views\usuarios\login.handlebars

```

<h4>Login</h4>
<div class="card">
 <div class="card-body">
 <form action="" method="POST">
 <label for="email">Email</label>
 <input type="email" class="form-control" name="email" required>
 <label for="senha">Senha</label>
 <input type="password" class="form-control" name="senha" required>

 <button type="submit" class="btn btn-success">Entrar</button>
 </form>
 </div>
</div>

```

<http://localhost:8081/usuarios/login>

The screenshot shows a web browser window titled "Blog Node.js". The address bar displays the URL "localhost:8081/usuarios/login". The page content is a login form for a blog application. The form has a title "Login" and two input fields: "Email" and "Senha". Below the inputs is a green "Entrar" button.

Blog do Node Home Categorias Login Registro

Login

Email

Senha

Entrar

## Aula 58 - Configurando o Passport

<http://www.passportjs.org/>

The screenshot shows the official Passport.js website. At the top, there's a navigation bar with icons for back, forward, search, and refresh, followed by a warning 'Não seguro' and the URL 'www.passportjs.org'. Below the header is a large green 'Passport' logo. To the left is a sidebar with links: 'Home', 'Documentation', 'Features', and 'Strategies'. The main content area features a search bar with placeholder 'Search for Strategies'. Below it is a section titled 'Simple, unobtrusive authentication for Node.js'. A text block explains that Passport is authentication middleware for Node.js, supporting various strategies like GitHub, Facebook, Twitter, and local password authentication. To the right of this text is a small screenshot of a terminal window showing the command 'passport.authenticate("github")'. At the bottom, a green button says 'VIEW ALL STRATEGIES'.

The screenshot shows the 'SEARCH FOR STRATEGIES' page. The search bar contains the placeholder 'Start typing' and displays '519 STRATEGIES'. Below the search bar is a grid of six strategy cards:

- passport-facebook**: Facebook authentication strategy for Passport. Rating: 99,757 stars, 1253 reviews. Status: FEATURED.
- passport-http-bearer**: HTTP Bearer authentication strategy for Passport. Rating: 122,785 stars, 907 reviews. Status: FEATURED.
- passport-google-oauth**: Google (OAuth) authentication strategies for Passport. Rating: 66,397 stars, 712 reviews. Status: FEATURED.
- passport-twitter**: Twitter authentication strategy for Passport. Rating: 48,479 stars, 442 reviews. Status: FEATURED.
- passport-auth0**: Auth0 platform authentication strategy for Passport.js. Rating: 37,446 stars, 296 reviews. Status: FEATURED.
- passport-local**: Local username and password authentication strategy for Passport. Rating: 455,571 stars, 2462 reviews. Status: FEATURED.

Usaremos a estratégia de autenticação passport local

## Instalação do passport

```
npm install passport --save
```

```
C:\guia_programador\nodejs\blogapp>npm install passport --save
npm [WARN] saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm [WARN] enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm [WARN] blogapp No description
npm [WARN] blogapp No repository field.
npm [WARN] blogapp No README data
npm [WARN] blogapp No license field.

+ passport@0.4.1
added 3 packages from 2 contributors and audited 185 packages in 7.769s

3 packages are looking for funding
 run `npm fund` for details

found 0 vulnerabilities
```

## Instalando a estratégia

```
npm install passport-local --save
```

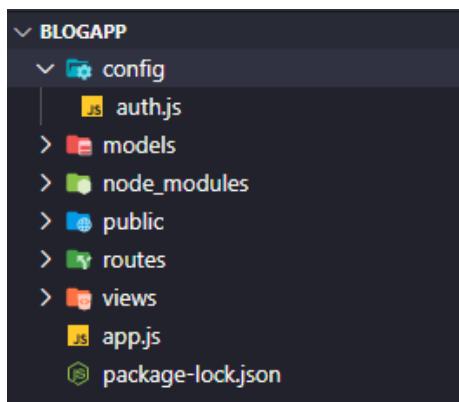
```
C:\guia_programador\nodejs\blogapp>npm install passport-local --save
npm [WARN] saveError ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm [WARN] enoent ENOENT: no such file or directory, open 'C:\guia_programador\nodejs\blogapp\package.json'
npm [WARN] blogapp No description
npm [WARN] blogapp No repository field.
npm [WARN] blogapp No README data
npm [WARN] blogapp No license field.

+ passport-local@1.0.0
added 1 package from 1 contributor and audited 188 packages in 4.843s

3 packages are looking for funding
 run `npm fund` for details

found 0 vulnerabilities
```

- Crie uma pasta chamada **config** e dentro dela um arquivo chamado **auth.js**



## config\auth.js

```
const localStrategy = require("passport-local").Strategy
const mongoose = require("mongoose")
const bcrypt = require("bcryptjs")

// Model de usuário

require("../models/Usuario")
const Usuario = mongoose.model("usuarios")

module.exports = function(passport){
 passport.use(new localStrategy({usernameField: 'email', passwordField: 'senha'}, (email, senha, done) => {
 Usuario.findOne({email: email}).then((usuario) => {
 if(!usuario){
 return done(null, false, {message: "Esta conta não existe!"})
 }

 bcrypt.compare(senha, usuario.senha, (erro, batem) => {
 if(batem){
 return done(null, usuario)
 } else {
 return done(null, false, {message: "Senha incorreta!"})
 }
 })
 })
)))

 passport.serializeUser((usuario, done) => {
 done(null, usuario.id)
 })

 passport.deserializeUser((id, done) => {
 Usuario.findById(id, (err, usuario) => {
 done(err, usuario)
 })
 })
}
```

## app.js

```
/* Carregando módulos */

const express = require("express")
const handlebars = require("express-handlebars")
const bodyParser = require("body-parser")
const app = express()
const admin = require("./routes/admin")
const usuarios = require("./routes/usuario")
const path = require("path")
const mongoose = require("mongoose")
const session = require("express-session")
const flash = require("connect-flash")
require("./models/Postagem")
const Postagem = mongoose.model("postagens")
require("./models/Categoria")
const Categoria = mongoose.model("categorias")
const passport = require("passport")
require("./config/auth")(passport)

/* Configurações */

// Session
app.use(session({
 secret: "cursodenode",
 resave: true,
 saveUninitialized: true
}))
app.use(passport.initialize())
app.use(passport.session())
app.use(flash())

// Middleware
app.use((req, res, next) => {
 res.locals.success_msg = req.flash("success_msg")
 res.locals.error_msg = req.flash("error_msg")
 next()
})

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/blogapp").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});
```

```

// Public
app.use(express.static(path.join(__dirname, "public")))
app.use((req, res, next) => {
 console.log('Oi, eu sou um middleware!')
 next()
})

/* Rotas */

app.get('/', (req, res) => {
 Postagem.find().populate("categoria").sort({data: "desc"}).then((postagens) => {
 res.render("index", {postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro interno!")
 res.redirect('/404')
 })
})

app.get('/postagem/:slug', (req, res) => {
 Postagem.findOne({slug: req.params.slug}).then((postagem) =>{
 if(postagem){
 res.render("postagem/index", {postagem: postagem})
 } else {
 req.flash("error_msg", "Essa postagem não existe!")
 res.redirect("/")
 }
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
 })
})

app.get('/categorias', (req, res) => {
 Categoria.find().then((categorias) => {
 res.render("categorias/index", {categorias: categorias})
 }).catch(() => {
 req.flash("error_msg", "Erro interno ao listar categorias!")
 res.redirect("/")
 })
})

app.get('/categorias/:slug', (req, res) => {
 Categoria.findOne({slug: req.params.slug}).then((categoria) => {
 if(categoria){
 Postagem.find({categoria: categoria._id}).then((postagens) => {
 res.render("categorias/postagens", {categoria: categoria, postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro ao listar os posts")
 res.redirect("/")
 })
 } else {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/")
 }
 }).catch((err) => {

```

```
req.flash("error_msg", "Houve um erro interno ao carregar a página dessa categoria!")
res.redirect("/")
})

app.get('/404', (req, res) => {
 res.send('Erro 404!')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)
app.use('/usuarios', usuarios)

/* Outros */

const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

## Aula 59 - Finalizando autenticação

### routes/usuario.js

```
const express = require("express")
const router = express.Router()
const mongoose = require("mongoose")
require("../models/Usuario")
const Usuario = mongoose.model("usuarios")
const bcrypt = require("bcryptjs")
const passport = require("passport")

router.get('/registro', (req, res) => {
 res.render("usuarios/registro")
})

router.post('/registro', (req, res) => {
 var erros = [];

 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({texto: "Nome inválido!"})
 }

 if(!req.body.email || typeof req.body.email == undefined || req.body.email == null){
 erros.push({texto: "Email inválido!"})
 }

 if(!req.body.senha || typeof req.body.senha == undefined || req.body.senha == null){
 erros.push({texto: "Senha inválida!"})
 }

 if(req.body.senha.length < 4){
 erros.push({texto: "A senha deve ter no mínimo 4 caracteres!"})
 }

 if(req.body.senha2 != req.body.senha){
 erros.push({texto: "As senhas digitadas são diferentes!"})
 }

 if(erros.length > 0){
 res.render("usuarios/registro", {erros: erros})
 } else {
 Usuario.findOne({email: req.body.email}).then((usuario) => {
 if(usuario){
 req.flash("error_msg", "Já existe uma conta com este email no nosso sistema!")
 res.redirect("/usuarios/registro")
 } else {
 const novoUsuario = new Usuario({
 nome: req.body.nome,
 email: req.body.email,
 senha: req.body.senha
 })
 }
 })
 }
})
```

```

bcrypt.genSalt(10, (erro, salt) => {
 bcrypt.hash(novoUsuario.senha, salt, (erro, hash) => {
 if(erro){
 req.flash("error_msg", "Houve um erro durante o salvamento do usuário!")
 res.redirect("/")
 }
 novoUsuario.senha = hash

 novoUsuario.save().then(() => {
 req.flash("success_msg", "Usuário criado com sucesso!")
 res.redirect("/")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao criar o usuário, tente novamente!")
 res.redirect("/usuarios/registro")
 })
}

})
})

})
})

})

router.get('/login', (req, res) => {
 res.render("usuarios/login")
})

router.post('/login', (req, res, next) => {
 passport.authenticate("local", {
 successRedirect: "/",
 failureRedirect: "/usuarios/login",
 failureFlash: true
 })(req, res, next)
})

module.exports = router

```

## views\usuarios\login.handlebars

```
<div class="card">
 <div class="card-body">
 <h3>Login</h3>
 <div class="card">
 <div class="card-body">

 <form action="/usuarios/login" method="POST">

 <label for="email">Email: </label>
 <input type="email" id="email" name="email" placeholder="Digite o seu email" class="form-control mb-1" required>

 <label for="senha">Senha: </label>
 <input type="password" id="senha" name="senha" placeholder="Digite a sua senha" class="form-control mb-1" required>

 <button type="submit" class="btn btn-success mt-4">Entrar</button>
 </form>

 </div>
 </div>
```

## app.js

```
/* Carregando módulos */

const express = require("express")
const handlebars = require("express-handlebars")
const bodyParser = require("body-parser")
const app = express()
const admin = require("./routes/admin")
const usuarios = require("./routes/usuario")
const path = require("path")
const mongoose = require("mongoose")
const session = require("express-session")
const flash = require("connect-flash")
require("./models/Postagem")
const Postagem = mongoose.model("postagens")
require("./models/Categoria")
const Categoria = mongoose.model("categorias")
const passport = require("passport")
require("./config/auth")(passport)

/* Configurações */

// Session
app.use(session({
 secret: "cursodenode",
 resave: true,
 saveUninitialized: true
}))
app.use(passport.initialize())
app.use(passport.session())
app.use(flash())

// Middleware
app.use((req, res, next) => {
 res.locals.success_msg = req.flash("success_msg")
 res.locals.error_msg = req.flash("error_msg")
 res.locals.error = req.flash("error")
 next()
})

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/blogapp").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
```

```

});

// Public
app.use(express.static(path.join(__dirname, "public")))
app.use((req, res, next) => {
 console.log('Oi, eu sou um middleware!')
 next()
})

/* Rotas */

app.get('/', (req, res) => {
 Postagem.find().populate("categoria").sort({data: "desc"}).then((postagens) => {
 res.render("index", {postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro interno!")
 res.redirect('/404')
 })
})

app.get('/postagem/:slug', (req, res) => {
 Postagem.findOne({slug: req.params.slug}).then((postagem) =>{
 if(postagem){
 res.render("postagem/index", {postagem: postagem})
 } else {
 req.flash("error_msg", "Essa postagem não existe!")
 res.redirect("/")
 }
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
 })
})

app.get('/categorias', (req, res) => {
 Categoria.find().then((categorias) => {
 res.render("categorias/index", {categorias: categorias})
 }).catch(() => {
 req.flash("error_msg", "Erro interno ao listar categorias!")
 res.redirect("/")
 })
})

app.get('/categorias/:slug', (req, res) => {
 Categoria.findOne({slug: req.params.slug}).then((categoria) => {
 if(categoria){
 Postagem.find({categoria: categoria._id}).then((postagens) => {
 res.render("categorias/postagens", {categoria: categoria, postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro ao listar os posts")
 res.redirect("/")
 })
 } else {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/")
 }
 }).catch((err) => {

```

```
req.flash("error_msg", "Houve um erro interno ao carregar a página dessa categoria!")
res.redirect("/")
})

app.get('/404', (req, res) => {
 res.send('Erro 404!')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)
app.use('/usuarios', usuarios)

/* Outros */

const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

## views\layouts\main.handlebars

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
 <link rel="stylesheet" href="/css/bootstrap.css">
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <meta http-equiv="X-UA-Compatible" content="ie=edge">
 <title>Blog Node.JS</title>
</head>
<body>
 {{>_navbar}}
```

```
<div class="container mt-4">
 {{#if error}}
 <div class="alert alert-danger">{{error}}</div>
 {{else}}
 {{/if}}
```

```
 {{>_msg}}
 {{>body}}
</div>
```

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
<script src="/js/bootstrap.js"></script>
</body>
</html>
```

Senha: 123456

<http://localhost:8081/usuarios/login>

## Login

Email

Senha

[Blog do Node](#) | [Home](#) | [Categorias](#) | [Login](#) | [Registrar](#)

# Bem-vindo ao Blog do Node!

This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.

It uses utility classes for typography and spacing to space content out within the larger container.

[Crie uma conta](#)

## Postagens recentes

### Personagens principais no desenvolvimento Web

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

[Leia mais](#)

Categoria: Desenvolvimento Web

Data de publicação: Sun May 02 2021 09:19:53 GMT-0300 (Horário Padrão de Brasília)

### O que é o PHP

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

[Leia mais](#)

Categoria: PHP

Data de publicação: Sun May 02 2021 09:08:06 GMT-0300 (Horário Padrão de Brasília)

### Javascript - Uma trilha para o futuro

JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

[Leia mais](#)

Categoria: Javascript

Data de publicação: Sat May 01 2021 20:24:38 GMT-0300 (Horário Padrão de Brasília)

Tentando entrar com senha incorreta:

Login

Email

Senha

Senha incorreta!

Login

Email

Senha

Tentando entrar com email incorreto:

Login

Email

Senha

Esta conta não existe!

Login

Email

Senha

## Aula 60 - Controlando acesso

Nesta aula vamos desenvolver o sistema de permissões de rotas protegidas onde só os administradores vão poder acessar o painel administrativo do site.

### app.js

```
/* Carregando módulos */

const express = require("express")
const handlebars = require("express-handlebars")
const bodyParser = require("body-parser")
const app = express()
const admin = require("./routes/admin")
const usuarios = require("./routes/usuario")
const path = require("path")
const mongoose = require("mongoose")
const session = require("express-session")
const flash = require("connect-flash")
require("./models/Postagem")
const Postagem = mongoose.model("postagens")
require("./models/Categoria")
const Categoria = mongoose.model("categorias")
const passport = require("passport")
require("./config/auth")(passport)

/* Configurações */

// Session
app.use(session({
 secret: "cursodenode",
 resave: true,
 saveUninitialized: true
}))
app.use(passport.initialize())
app.use(passport.session())
app.use(flash())

// Middleware
app.use((req, res, next) => {
 res.locals.success_msg = req.flash("success_msg")
 res.locals.error_msg = req.flash("error_msg")
 res.locals.error = req.flash("error")
 res.locals.user = req.user || null
 next()
})

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())
```

```

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect("mongodb://localhost/blogapp").then(() => {
 console.log('Mongo conectado...')
}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});

// Public
app.use(express.static(path.join(__dirname, "public")))
app.use((req, res, next) => {
 console.log('Oi, eu sou um middleware!')
 next()
})

/* Rotas */

app.get('/', (req, res) => {
 Postagem.find().populate("categoria").sort({data: "desc"}).then((postagens) => {
 res.render("index", {postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro interno!")
 res.redirect('/404')
 })
})

app.get('/postagem/:slug', (req, res) => {
 Postagem.findOne({slug: req.params.slug}).then((postagem) =>{
 if(postagem){
 res.render("postagem/index", {postagem: postagem})
 } else {
 req.flash("error_msg", "Essa postagem não existe!")
 res.redirect("/")
 }
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
 })
})

app.get('/categorias', (req, res) => {
 Categoria.find().then((categorias) => {
 res.render("categorias/index", {categorias: categorias})
 }).catch(() => {
 req.flash("error_msg", "Erro interno ao listar categorias!")
 res.redirect("/")
 })
})

```

```

app.get('/categorias/:slug', (req, res) => {
 Categoria.findOne({slug: req.params.slug}).then((categoria) => {
 if(categoria){
 Postagem.find({categoria: categoria._id}).then((postagens) => {
 res.render("categorias/postagens", {categoria: categoria, postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro ao listar os posts")
 res.redirect("/")
 })
 } else {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/")
 }
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao carregar a página dessa categoria!")
 res.redirect("/")
 })
})

app.get('/404', (req, res) => {
 res.send('Erro 404!')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)
app.use('/usuarios', usuarios)

/* Outros */

const PORT = '8081'

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})

```

- Crie uma pasta chamada **helpers** e dentro dela o arquivo **eAdmin.js** que vai verificar se o usuário está autenticado e se é admin.

Helper são pequenas funções que servem para ajudar em alguma coisa.

Vamos criar um helper para que o usuário não entre em uma rota específica.

## helpers\ eAdmin.js

```
module.exports = {

 eAdmin: function(req, res, next){
 if(req.isAuthenticated()){
 if(req.user.eAdmin == 1) {
 return next()
 } else {
 req.flash("error_msg", "Você não tem permissão para realizar essa operação!")
 res.redirect("/")
 }
 } else {
 req.flash("error_msg", "Você não tem permissão para realizar essa operação!")
 res.redirect("/usuarios/login")
 }
 }
}
```

## views\partials\navbar.handlebars.js

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
 Blog do Node
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">

 </button>

 <div class="collapse navbar-collapse" id="navbarSupportedContent">
 <ul class="navbar-nav mr-auto">
 <li class="nav-item active">
 Home (current)

 <li class="nav-item">
 Categories

 {{#if user}}
 {{else}}
 <li class="nav-item">
 Login

 <li class="nav-item">
 Registro

 {{/if}}
 {{/else}}

 </div>
</nav>
```

<http://localhost:8081/usuarios/login>

### Login



A screenshot of a login form. It has two input fields: 'Email' containing 'victor\_lima@gmail.com' and 'Senha' containing '\*\*\*\*\*'. Below the inputs is a green 'Entrar' button.

- Quando o usuário ou o administrador realizam o login e acessam o sistema, no menu superior não são mais exibidos os links de login e de registro.

# Bem-vindo ao Blog do Node!

This is a simple hero unit, a simplejumbotron-style component for calling extra attention to featured content or information.

It uses utility classes for typography and spacing to space content out within the larger container.

[Crie uma conta](#)

## Postagens recentes

### Personagens principais no desenvolvimento Web

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

[Leia mais](#)

Categoria: Desenvolvimento Web

Data de publicação: Sun May 02 2021 09:13:53 GMT-0300 (Horário Padrão de Brasília)

### O que é o PHP

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

[Leia mais](#)

Categoria: PHP

Data de publicação: Sun May 02 2021 09:08:05 GMT-0300 (Horário Padrão de Brasília)

### Javascript - Uma trilha para o futuro

JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

[Leia mais](#)

Categoria: Javascript

Data de publicação: Sat May 01 2021 20:24:36 GMT-0300 (Horário Padrão de Brasília)

## routers\admin.js

```
const express = require("express")
const router = express.Router()
const mongoose = require("mongoose")
require("../models/Categoria")
const Categoria = mongoose.model("categorias")
require("../models/Postagem")
const Postagem = mongoose.model("postagens")
const {eAdmin} = require("../helpers/eAdmin")

// Definindo as rotas

router.get('/', eAdmin, (req, res) => {
 res.render('admin/index')
})

router.get('/posts', eAdmin, (req, res) => {
 res.send('Página de posts')
})

router.get("/categorias", eAdmin, (req, res) => {
 Categoria.find().sort({date:'desc'}).then((categorias) => {
 res.render("admin/categorias", {categorias: categorias})
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as categorias")
 res.redirect("/admin")
 })
})

router.get('/categorias/add', eAdmin, (req, res) => {
 res.render('admin/add_categoria')
})

router.post('/categorias/nova', eAdmin, (req, res) => {
 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2){
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
 if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
 } else {

```

```

const novaCategoria = {
 nome: req.body.nome,
 slug: req.body.slug
}

new Categoria(novaCategoria).save().then(() => {
 req.flash("success_msg", "Categoria criada com sucesso!")
 res.redirect("/admin/categorias")
}).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a categoria! Tente novamente.")
 res.redirect("/admin")
})
}

router.get("/categorias/edit/:id", eAdmin, (req, res) => {
 Categoria.findOne({_id: req.params.id}).then((categoria) => {
 res.render("admin/edit_categoria", {categoria: categoria})
 }).catch((err) => {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/admin/categorias")
 })
})

router.post("/categorias/edit", eAdmin, (req, res) => {

 var erros = []
 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({
 texto: 'Nome inválido'
 })
 }
 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
 if(req.body.nome.length < 2)
 {
 erros.push({
 texto: "Nome da categoria é muito pequeno"
 })
 }
 if(erros.length > 0){
 res.render("admin/add_categoria", {erros: erros})
 } else {

 Categoria.findOne({_id: req.body.id}).then((categoria) => {

 categoria.nome = req.body.nome
 categoria.slug = req.body.slug

 categoria.save().then(() => {
 req.flash("success_msg", "Categoria editada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da categoria")
 })
 })
 }
})
}

```

```

 res.redirect("/admin/categorias")
 })

}).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a categoria!")
 res.redirect("/admin/categorias")
})

}

})

router.post("/categorias/deletar", eAdmin, (req, res) => {

 Categoria.remove({_id: req.body.id}).then(() => {
 req.flash("success_msg", "Categoria deletada com sucesso!")
 res.redirect("/admin/categorias")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao deletar a categoria!")
 res.redirect("/admin/categorias")
 })
})

router.get("/postagens", eAdmin, (req, res) => {
 Postagem.find().populate("categoria").sort({date:'desc'}).then((postagens) => {
 res.render("admin/postagens", {postagens: postagens})
 }).catch((err) => {
 res.flash("error_msg", "Houve um erro ao listar as postagens")
 res.redirect("/admin")
 })
})

router.get('/postagens/add', eAdmin, (req, res) => {
 Categoria.find().then((categorias) => {
 res.render('admin/add_postagem', {categorias: categorias})
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao carregar o formulário")
 res.redirect("/admin")
 })
})

router.post('/postagens/nova', eAdmin, (req, res) => {
 var erros = []

 if(!req.body.titulo || typeof req.body.titulo == undefined || req.body.titulo == null){
 erros.push({
 texto: 'Título inválido'
 })
 }

 if(!req.body.slug || typeof req.body.slug == undefined || req.body.slug == null){
 erros.push({
 texto: 'Slug inválido'
 })
 }
})

```

```

if(!req.body.descricao || typeof req.body.descricao == undefined || req.body.descricao == null){
 erros.push({
 texto: 'Descrição inválida'
 })
}

if(!req.body.conteudo || typeof req.body.conteudo == undefined || req.body.conteudo == null){
 erros.push({
 texto: 'Conteúdo inválido'
 })
}

if(req.body.categoria == 0){
 erros.push({
 texto: 'Categoria inválida! Registre uma categoria.'
 })
}

if(req.body.descricao.length < 3)
{
 erros.push({
 texto: "A descrição da postagem é muito pequena!"
 })
}

if(erros.length > 0){
 res.render("admin/add_postagem", {erros: erros})
} else {
 const novaPostagem = {
 titulo: req.body.titulo,
 slug: req.body.slug,
 descricao: req.body.descricao,
 conteudo: req.body.conteudo,
 categoria: req.body.categoria
 }

 new Postagem(novaPostagem).save().then(() => {
 req.flash("success_msg", "Postagem criada com sucesso!")
 res.redirect("/admin/postagens")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao salvar a postagem! Tente novamente.")
 res.redirect("/admin/postagens")
 })
}
}

router.get("/postagens/edit/:id", eAdmin, (req, res) => {
 Postagem.findOne({_id: req.params.id}).then((postagem) => {
 Categoria.find().then((categorias) => {
 res.render("admin/edit_postagem", {categorias: categorias, postagem: postagem})
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao listar as categorias!")
 res.redirect("/admin/postagens")
 })
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao carregar o formulário de edição!")
 res.redirect("/admin/postagens")
 })
}
)

```

```

 })
 })

router.post("/postagens/edit", eAdmin, (req, res) => {

 var erros = []

 Postagem.findOne({_id: req.body.id}).then((postagem) => {

 postagem.titulo = req.body.titulo
 postagem.slug = req.body.slug
 postagem.descricao = req.body.descricao
 postagem.conteudo = req.body.conteudo
 postagem.categoria = req.body.categoria

 postagem.save().then(() => {
 req.flash("success_msg", "Postagem editada com sucesso!")
 res.redirect("/admin/postagens")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao salvar a edição da postagem")
 res.redirect("/admin/postagens")
 })
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao editar a postagem!")
 res.redirect("/admin/postagens")
 })
}

router.get("/postagens/deletar/:id", eAdmin, (req, res) => {

 Postagem.remove({_id: req.params.id}).then(() => {
 req.flash("success_msg", "Postagem deletada com sucesso!")
 res.redirect("/admin/postagens")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao deletar a postagem!")
 res.redirect("/admin/postagens")
 })
}

module.exports = router

```

## Usuário tentando acessar uma rota administrativa

localhost:8081/admin/

The screenshot shows a web page with a dark header bar containing the text "Blog do Node", "Home", and "Categorias". Below the header is a red horizontal bar with the message "Você não tem permissão para realizar essa operação!". The main content area features a large hero unit with the heading "Bem-vindo ao Blog do Node!" and a subtext explaining it's a simple hero unit component. A blue button labeled "Crie uma conta" is visible. The overall layout is clean and modern.

### Postagens recentes

#### Personagens principais no desenvolvimento Web

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

[Leia mais](#)

Categoria: Desenvolvimento Web

Data de publicação: Sun May 02 2021 09:19:53 GMT-0300 (Horário Padrão de Brasília)

#### O que é o PHP

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

[Leia mais](#)

Categoria: PHP

Data de publicação: Sun May 02 2021 09:08:05 GMT-0300 (Horário Padrão de Brasília)

#### Javascript - Uma trilha para o futuro

JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

[Leia mais](#)

Categoria: Javascript

Data de publicação: Sat May 01 2021 20:24:36 GMT-0300 (Horário Padrão de Brasília)

localhost:8081/admin/categorias

The screenshot shows a web application interface. At the top, there's a navigation bar with links for 'Blog do Node', 'Home', and 'Categorias'. Below the navigation, a red error message box displays the text 'Você não tem permissão para realizar essa operação!'. The main content area features a large hero unit with the heading 'Bem-vindo ao Blog do Node!' and a subtext explaining it's a simple jumbotron-style component for calling extra attention to featured content or information. It includes a blue button labeled 'Crie uma conta'. Below this, there's a section titled 'Postagens recentes' containing three items:

- Personagens principais no desenvolvimento Web**  
O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.  
[Leia mais](#)  
Categoria: Desenvolvimento Web  
Data de publicação: Sun May 02 2021 08:15:53 GMT-0300 (Horário Padrão de Brasília)
- O que é o PHP**  
PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.  
[Leia mais](#)  
Categoria: PHP  
Data de publicação: Sun May 02 2021 08:08:05 GMT-0300 (Horário Padrão de Brasília)
- Javascript - Uma trilha para o futuro**  
JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.  
[Leia mais](#)  
Categoria: Javascript  
Data de publicação: Sat May 01 2021 20:24:36 GMT-0200 (Horário Padrão de Brasília)

localhost:8081/admin/postagens

The screenshot shows a web application interface for managing blog posts. At the top, there is a navigation bar with links for 'Blog do Node', 'Home', and 'Categorias'. Below the navigation, a red banner displays the message 'Você não tem permissão para realizar essa operação!'. The main content area features a large hero unit with the heading 'Bem-vindo ao Blog do Node!' and a subtext explaining it's a simple jumbotron-style component for calling extra attention to featured content or information. It also notes that it uses utility classes for typography and spacing to space content out within the larger container. A blue button labeled 'Crie uma conta' is visible. Below this, there is a section titled 'Postagens recentes' containing three blog post cards.

**Personagens principais no desenvolvimento Web**

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

Ler mais

Categoria: Desenvolvimento Web  
Data de publicação: Sun May 02 2021 08:15:53 GMT-0300 (Horário Padrão de Brasília)

**O que é o PHP**

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

Ler mais

Categoria: PHP  
Data de publicação: Sun May 02 2021 08:08:05 GMT-0300 (Horário Padrão de Brasília)

**Javascript - Uma trilha para o futuro**

JavaScript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

Ler mais

Categoria: Javascript  
Data de publicação: Sat May 01 2021 20:24:36 GMT-0200 (Horário Padrão de Brasília)

- Para criar uma conta para o administrador, no arquivo a seguir, inclua a linha em vermelho:

### routes\usuario.js

```
const express = require('express')
const router = express.Router()
const mongoose = require('mongoose')
require('../models/Usuario')
const Usuario = mongoose.model("usuarios")
const bcrypt = require('bcryptjs')
const passport = require("passport")

router.get('/registro', (req, res) => {
 res.render("usuarios/registro")
})

router.post('/registro', (req, res) => {
 var erros = [];

 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({texto: "Nome inválido!"})
 }

 if(!req.body.email || typeof req.body.email == undefined || req.body.email == null){
 erros.push({texto: "Email inválido!"})
 }

 if(!req.body.senha || typeof req.body.senha == undefined || req.body.senha == null){
 erros.push({texto: "Senha inválida!"})
 }

 if(req.body.senha.length < 4){
 erros.push({texto: "A senha deve ter no mínimo 4 caracteres!"})
 }

 if(req.body.senha2 != req.body.senha){
 erros.push({texto: "As senhas digitadas são diferentes!"})
 }

 if(erros.length > 0){
 res.render("usuarios/registro", {erros: erros})
 } else {
 Usuario.findOne({email: req.body.email}).then((usuario) => {
 if(usuario){
 req.flash("error_msg", "Já existe uma conta com este email no nosso sistema!")
 res.redirect("/usuarios/registro")
 } else {
 const novoUsuario = new Usuario({
 nome: req.body.nome,
 email: req.body.email,
 senha: req.body.senha,
eAdmin: 1
 })
 }
 })
 }
})
```

```
bcrypt.genSalt(10, (erro, salt) => {
 bcrypt.hash(novoUsuario.senha, salt, (erro, hash) => {
 if(erro){
 req.flash("error_msg", "Houve um erro durante o salvamento do usuário!")
 res.redirect("/")
 }
 novoUsuario.senha = hash

 novoUsuario.save().then(() => {
 req.flash("success_msg", "Usuário criado com sucesso!")
 res.redirect("/")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao criar o usuário, tente novamente!")
 res.redirect("/usuarios/registro")
 })
 })
}).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
})
})

router.get('/login', (req, res) => {
 res.render("usuarios/login")
})

router.post('/login', (req, res, next) => {
 passport.authenticate("local", {
 successRedirect: "/",
 failureRedirect: "/usuarios/login",
 failureFlash: true
 })(req, res, next)
})

module.exports = router
```

- Agora crie a conta:

Senha: 12345678

<http://localhost:8081/usuarios/registro>

Nome: Victor Lima Admin

Email: victor\_lima\_admin@gmail.com

Senha: \*\*\*\*\*

Repetir a senha: \*\*\*\*\*

Criar

Bem-vindo ao Blog do Node!

Crie uma conta

Postagens recentes

**Personagens principais no desenvolvimento Web**

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e consolida padrões para a web. Os browsers, que importam essas regras e padrões da forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

[Leia mais](#)

Categoria: Desenvolvimento Web  
Data de publicação: Sun May 02 2021 08:55 GMT-0300 (Horário Padrão do Brasil)

**O que é o PHP**

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

[Leia mais](#)

Categoria: PHP  
Data de publicação: Sun May 02 2021 08:55:03 GMT-0300 (Horário Padrão do Brasil)

**Javascript - Uma trilha para o futuro**

Javascript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

[Leia mais](#)

Categoria: Javascript  
Data de publicação: Sun May 02 2021 08:54:58 GMT-0300 (Horário Padrão do Brasil)

- No Mongo:

```
> db.usuarios.find();
{ "_id": ObjectId("600f8810736fec20ccca7e19"), "eAdmin": 0, "nome": "Victor Lima", "email": "victor_lima@gmail.com", "senha": "$2a$10$5AcPSLG92qQPKv8ty7TuiYzMG2BGHB/aEB5F/Ootbw5us3zVuk", "__v": 0 }
{ "_id": ObjectId("600f8a2c5ceaec0c10bf33a5"), "eAdmin": 1, "nome": "Victor Lima Admin", "email": "victor_lima_admin@gmail.com", "senha": "$2a$10$RHa5WCH5MCZyACY4RXiHNO8AXg4UyCV5kgXqZU.KYr9KgFbNykhkK", "__v": 0 }
```

<http://localhost:8081/admin/categorias/add>



- Agora que já foi criada a conta do administrador, retorne ao arquivo `routes\usuario.js`, apague a linha `eadmin: 1` e salve o arquivo.

`views\partials\_navbar.handlebars`

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
 Blog do Node
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent">
 aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

 </button>

 <div class="collapse navbar-collapse" id="navbarSupportedContent">
 <ul class="navbar-nav mr-auto">
 <li class="nav-item active">
 Home (current)

 <li class="nav-item">
 Categorias

 {{#if user}}
 {{else}}
 <li class="nav-item">
 Login

 {{/else}}

 </div>
</nav>
```

```
<li class="nav-item">
 Registro

{{/if}}

</div>
</nav>
```

## Aula 61 - Logout

### routes\usuario.js

```
const express = require("express")
const router = express.Router()
const mongoose = require("mongoose")
require("../models/Usuario")
const Usuario = mongoose.model("usuarios")
const bcrypt = require("bcryptjs")
const passport = require("passport")

router.get('/registro', (req, res) => {
 res.render("usuarios/registro")
})

router.post('/registro', (req, res) => {
 var erros = [];

 if(!req.body.nome || typeof req.body.nome == undefined || req.body.nome == null){
 erros.push({texto: "Nome inválido!"})
 }

 if(!req.body.email || typeof req.body.email == undefined || req.body.email == null){
 erros.push({texto: "Email inválido!"})
 }

 if(!req.body.senha || typeof req.body.senha == undefined || req.body.senha == null){
 erros.push({texto: "Senha inválida!"})
 }

 if(req.body.senha.length < 4){
 erros.push({texto: "A senha deve ter no mínimo 4 caracteres!"})
 }

 if(req.body.senha2 != req.body.senha){
 erros.push({texto: "As senhas digitadas são diferentes!"})
 }

 if(erros.length > 0){
 res.render("usuarios/registro", {erros: erros})
 } else {
 Usuario.findOne({email: req.body.email}).then((usuario) => {
 if(usuario){
 req.flash("error_msg", "Já existe uma conta com este email no nosso sistema!")
 res.redirect("/usuarios/registro")
 } else {
 const novoUsuario = new Usuario({
 nome: req.body.nome,
 email: req.body.email,
 senha: req.body.senha
 })
 }
 })
 }
})
```

```

bcrypt.genSalt(10, (erro, salt) => {
 bcrypt.hash(novoUsuario.senha, salt, (erro, hash) => {
 if(erro){
 req.flash("error_msg", "Houve um erro durante o salvamento do usuário!")
 res.redirect("/")
 }
 novoUsuario.senha = hash

 novoUsuario.save().then(() => {
 req.flash("success_msg", "Usuário criado com sucesso!")
 res.redirect("/")
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro ao criar o usuário, tente novamente!")
 res.redirect("/usuarios/registro")
 })
 })
}

})
}).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
})
}

})

router.get('/login', (req, res) => {
 res.render("usuarios/login")
})

router.post('/login', (req, res, next) => {

 passport.authenticate("local", {
 successRedirect: "/",
 failureRedirect: "/usuarios/login",
 failureFlash: true
 })(req, res, next)

})

router.get('/logout', (req, res) => {
 req.logout()
 req.flash("success_msg", "Deslogado com sucesso!")
 res.redirect("/usuarios/login")
})

module.exports = router

```

## views\partials\navbar.handlebars.js

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
 Blog do Node
 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

 </button>

 <div class="collapse navbar-collapse" id="navbarSupportedContent">
 <ul class="navbar-nav mr-auto">
 {{#if user}}
 <li class="nav-item active">
 Home (current)

 <li class="nav-item">
 Categorias

 <li class="nav-item">
 Sair

 {{else}}
 <li class="nav-item">
 Login

 <li class="nav-item">
 Registro

 {{/if}}

 </div>
</nav>
```

## Administrador acessando o sistema

<http://localhost:8081/usuarios/login>

Senha: 12345678

Blog do Node Login Registro

Login

Email

victor\_lima\_admin@gmail.com

Senha

\*\*\*\*\*

Entrar

Bem-vindo ao Blog do Node!

Crie uma conta

Postagens recentes

**Personagens principais no desenvolvimento Web**

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web. Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões. E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

[Leia mais](#)

Categoria: Desenvolvimento Web  
Data de publicação: Sun May 02 2021 08:08:05 GMT-0300 (Horário Fazenda de Brasília)

**O que é o PHP**

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação como o usuário através de formulários, parâmetros da URL e links.

[Leia mais](#)

Categoria: PHP  
Data de publicação: Sun May 02 2021 08:08:05 GMT-0300 (Horário Fazenda de Brasília)

**Javascript - Uma trilha para o futuro**

Javascript é uma linguagem de programação extremamente versátil e modular que está se tornando rapidamente uma das linguagens mais populares das principais empresas de tecnologia do mundo.

[Leia mais](#)

Categoria: Javascript  
Data de publicação: Sun May 01 2021 20:04:16 GMT-0300 (Horário Fazenda de Brasília)

- Clique no link "Categorias":

<http://localhost:8081/admin/postagens/add>

The screenshot shows a web browser window with the URL `localhost:8081/admin/postagens/add` in the address bar. The page title is "Blog do Node". Below the title, there are four input fields: "Título" (Title), "Slug" (Slug), "Descrição" (Description), and "Conteúdo" (Content). Each field has a placeholder text: "Título da postagem", "Slug da postagem", "Descrição da postagem", and "Conteúdo da postagem" respectively. Below the content area is a dropdown menu set to "PHP" and a green "Criar postagem" (Create Post) button.

- Clique no botão "Sair"

The screenshot shows a web browser window with the URL `localhost:8081/usuarios/login` in the address bar. The page title is "Blog do Node". A green success message at the top says "Deslogado com sucesso!". Below it is a "Login" form with two input fields: "Email" and "Senha" (Password), and a green "Entrar" (Enter) button.

## Usuário acessando o sistema

<http://localhost:8081/usuarios/login>

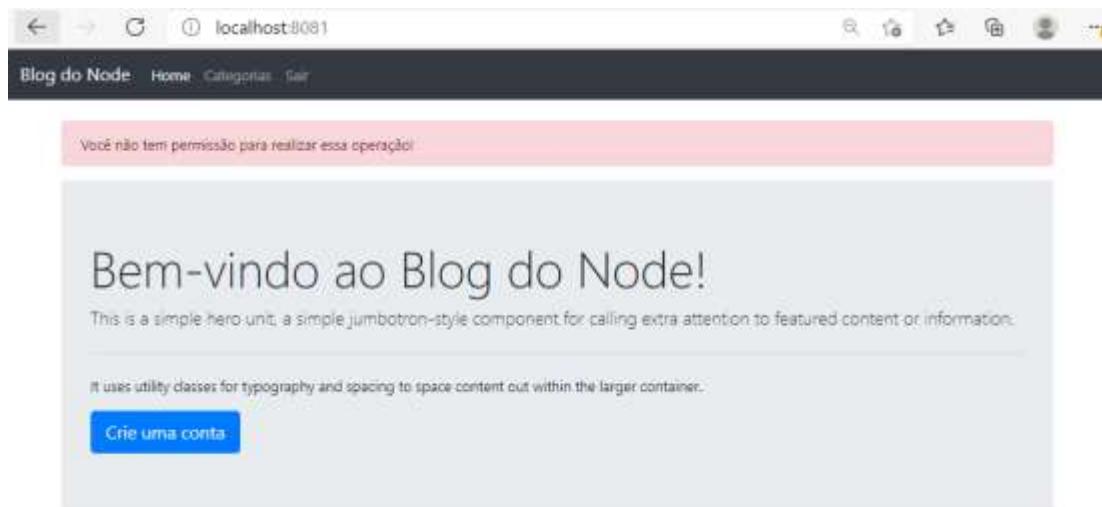
Senha: 123456



The screenshot shows a browser window with the URL 'localhost:8081/usuarios/login' in the address bar. The title bar says 'Blog do Node'. The main content is a 'Login' form with two input fields: 'Email' (containing 'victor\_lima@gmail.com') and 'Senha' (containing '\*\*\*\*\*'). Below the inputs is a green 'Entrar' button.

- Ao tentar acessar uma rota administrativa:

<http://localhost:8081/admin/postagens/add>



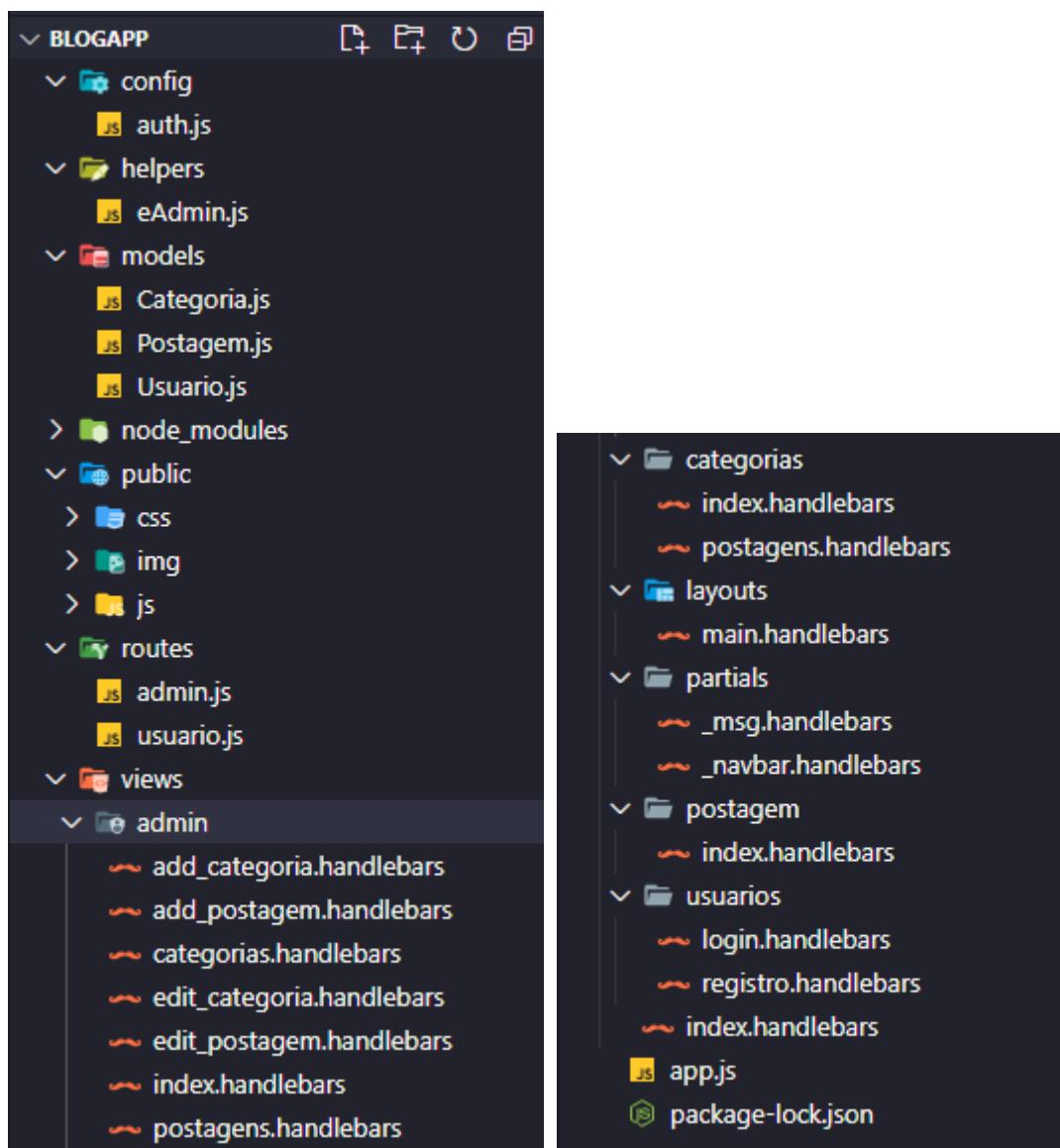
### Postagens recentes

#### Personagens principais no desenvolvimento Web

O desenvolvedor web depende de três personagens principais: W3C, que regulamenta, cria e sanciona padrões para a web; Os browsers, que importam essas regras e padrões de forma que a web seja mais homogênea e também podem criar padrões; E por fim os desenvolvedores, que aplicam os padrões em projetos, produzindo e criando na web.

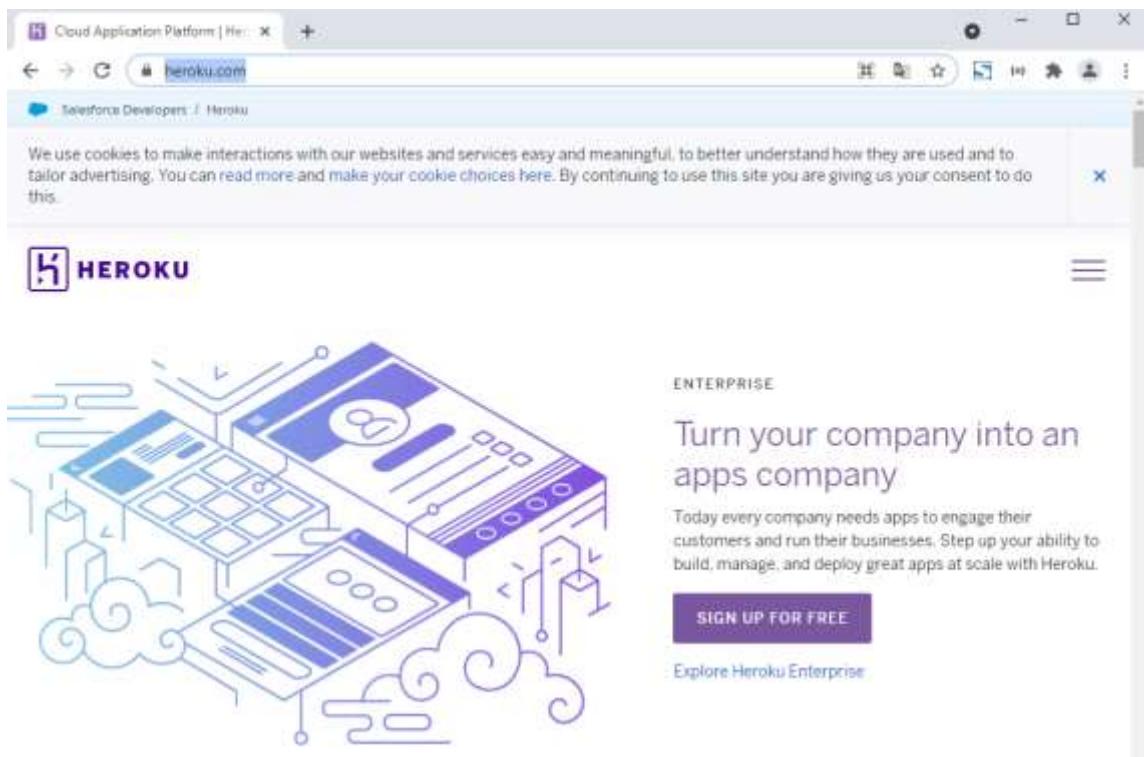
[Leia mais](#)

- Ao clicar no link de menu "Sair":



## Aula 62 - Como fazer Deploy na Heroku - Parte 1

<https://www.heroku.com/>



- Na pasta do projeto, entre com o comando:

`npm init`

```
C:\guia_programador\nodejs\blogapp>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (blogapp)
version: (1.0.0)
description: Blog App - NojeJS
entry point: (app.js)
test command:
git repository:
keywords:
author: Roberto Pinheiro
license: (ISC)
About to write to C:\guia_programador\nodejs\blogapp\package.json:
```

```
{
 "name": "blogapp",
 "version": "1.0.0",
 "description": "Blog App - NojeJS",
 "main": "app.js",
 "dependencies": {
 "bcryptjs": "^2.4.3",
 "body-parser": "^1.19.0",
 "express": "^4.17.1",
 "express-handlebars": "^5.3.0",
 "connect-flash": "^0.1.1",
 "express-session": "^1.17.1",
 "mongoose": "^5.12.7",
 "passport": "^0.4.1",
 "passport-local": "^1.0.0"
 },
 "devDependencies": {},
 "scripts": {
 "test": "echo \"Error: no test specified\" && exit 1"
 },
 "author": "Roberto Pinheiro",
 "license": "ISC"
}

Is this OK? (yes) y
C:\guia_programador\nodejs\blogapp>
```

- É criado o arquivo package.json.
- Insira neste arquivo a linha em vermelho:

### **package.json**

```
{
 "name": "blogapp",
 "version": "1.0.0",
 "description": "Blog App - NojeJS",
 "main": "app.js",
 "dependencies": {
 "bcryptjs": "^2.4.3",
 "body-parser": "^1.19.0",
 "express": "^4.17.1",
 "express-handlebars": "^5.3.0",
 "connect-flash": "^0.1.1",
 "express-session": "^1.17.1",
 "mongoose": "^5.12.7",
 "passport": "^0.4.1",
 "passport-local": "^1.0.0"
 },
 "devDependencies": {},
 "scripts": {
 "test": "echo \"Error: no test specified\" && exit 1",
 "start": "node app.js"
 },
 "author": "Roberto Pinheiro",
 "license": "ISC"
}
```

## app.js

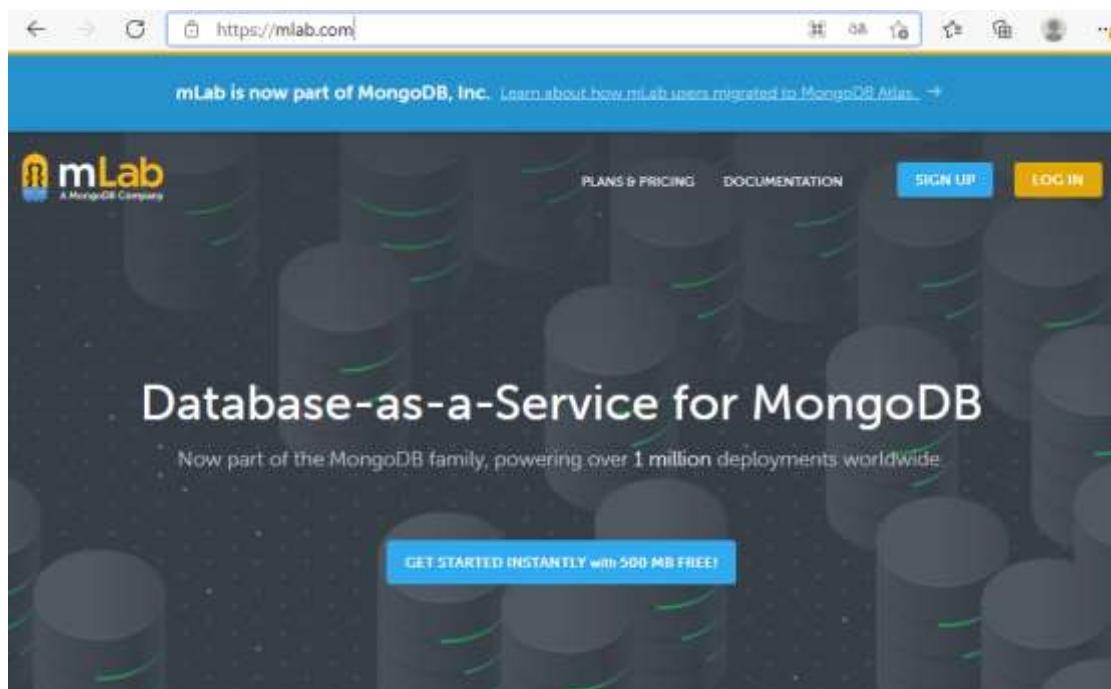
```
/* Outros */

const PORT = process.env.PORT || 8081

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

- Acesse o site do **mLab**:

<https://mlab.com/>



- Este site fornece gratuitamente 500 MBytes para hospedar um banco de dados Mongo.
- Para criar uma conta, clique no botão "**Sign Up**":

The screenshot shows a web browser window with the URL [https://www.mongodb.com/atlas-signup-from-mlab?utm\\_source=mlab...](https://www.mongodb.com/atlas-signup-from-mlab?utm_source=mlab...). The page is split into two main sections. On the left, a green sidebar features the mLab logo and the text "mLab is now part of the MongoDB family". It also contains a list of steps for creating an account, including picking a cloud provider, choosing regions, selecting cluster tier, and enabling options like multi-region and workload isolation. On the right, the MongoDB Atlas logo is displayed, followed by the heading "Try MongoDB Atlas" and the subtext "Used by millions of developers around the world." A large form is present for sign-up, requiring fields for company name, MongoDB usage, work email, first name, last name, and password. There is also a checkbox for agreeing to terms and privacy policy, and a green button at the bottom labeled "Get Started with 512 MB Free".

- Acesse a sua conta:

<https://account.mongodb.com/account/login?signedOut=true>

The screenshot shows the MongoDB account login page at <https://account.mongodb.com/account/login?signedOut=true>. The page includes a "Log in to your account" section with a "Log in with Google" button and a field for "Email Address". Below this, there are links for "New" and "Don't have an account? Sign Up". To the right, there is an illustration of a magnifying glass over a stack of cylinders, with the text "Seamless data tiering with Atlas Online Archive" and a subtext about offloading aged data. A "Learn More" link is also present.

- Crie um projeto chamado "blogapp"

- Create a Shared Cluster (Free):

- Cloud Provider: AWS
- Region: North America (N. Virginia)
- Cluster Name: Cluster0

The screenshot shows the MongoDB Atlas interface. The URL in the address bar is <https://cloud.mongodb.com/v2/0f8faaa9004c15b1f2180cd8/clusters?listPst=true>. The left sidebar has sections for DATA STORAGE (Clusters selected), Triggers, Data Lake, SECURITY, Database Access, Network Access, and Advanced. The main area is titled 'Clusters' and shows a table with one row: 'Cluster0' (Version 4.4). A modal window is open over the table, showing the configuration for the cluster: 'CLUSTER TIER' is set to 'N/A (breakout (General))', 'REGION' is 'AWS US East (N. Virginia) (us-east-1)', 'TYPE' is 'Replica Set - 3 nodes', and 'ALLOWED EXTERNAL APP' is 'MongoDB Cloud'. A status message at the bottom of the modal says 'Your cluster is being created... After creation, it will become visible in 1-2 minutes (10 minutes)'.

- Crie um Database User:



## Create a Database User

Set up database users, permissions, and authentication credentials in order to connect to your clusters.

[Add New Database User](#)

[Learn more](#)

## Add New Database User

Create a database user to grant an application or user, access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding [Access Manager](#)

### Authentication Method

Password	Certificate	AWS IAM (MongoDB 4.4 and up)
----------	-------------	---------------------------------

MongoDB uses **SCRAM** as its default authentication method.

#### Password Authentication

betopinheiro1005	
*****	SHOW
<input type="button" value="Autogenerate Secure Password"/> <input type="button" value="Copy"/>	

#### Database User Privileges

Select a [built-in role](#) or [privileges](#) for this user.

Read and write to any database	▼
--------------------------------	---

#### Restrict Access to Specific Clusters/Data Lakes

Enable to specify the resources this user can access. By default, all resources in this project are accessible.  OFF

#### Temporary User

This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week.  OFF

The screenshot shows the MongoDB Cloud interface for managing database users. The URL is <https://cloud.mongodb.com/v2/60f1aa95084115b1b2188d84/security/database/users>. The left sidebar has sections for DATA STORAGE, SECURITY, and NETWORK. The SECURITY section is expanded, showing Database Access, Network Access, and Advanced options. Under Database Access, the 'Database Users' tab is selected. A table lists a single user entry:

User Name	Authentication Method	MongoDB Roles	Resources	Actions
betopinheiro1005	SCRAM	readWriteAnyDatabase (Default)	All Resources	<input type="button" value="EDIT"/> <input type="button" value="DELETE"/>

A blue banner at the top of the page reads: "We are displaying your changes (entry added, modifying MongoDB)".

## Connect to Cluster0

Setup connection security > Choose a connection method > Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

You're ready to connect. Choose how you want to connect in the next step.

### 1 Add a connection IP address

✓ An IP address has been added to the IP Access List. [Add another address in the IP Access List tab.](#)

### 2 Create a Database User

✓ A MongoDB user has been added to this project. [Not yours? Create one in the MongoDB Users tab.](#)

**You'll need your MongoDB user's credentials in the next step.**

[Close](#)

[Choose a connection method](#)

## Connect to Cluster0

**x** Setup connection security > Choose a connection method > Connect

You can't connect yet. Set up your firewall access in the first step.

**Choose a connection method** [View documentation](#)

Get your pre-formatted connection string by selecting your tool below.



**Connect with the mongo shell**

Interact with your cluster using MongoDB's interactive Javascript interface



**Connect your application**

Connect your application to your cluster using MongoDB's native drivers



**Connect using MongoDB Compass**

Explore, modify, and visualize your data with MongoDB's GUI



[Go Back](#)

[Close](#)

- Selecione a opção "Connect using MongoDB Compass"

## Connect to Cluster0

✓ Setup connection security > ✓ Choose a connection method > Connect

I do not have MongoDB Compass

I have MongoDB Compass

- 1 Select your operating system and download MongoDB Compass

Windows 64-bit (7+)

[Download Compass \(1.26.1\)](#) or [Copy download URL](#)

- 2 Copy the connection string, then open MongoDB Compass.

```
mongodb+srv://betopinheiro1005:<password>@cluster0.gnwt9.mongodb.net/test
```



You will be prompted for the password for the **betopinheiro1005** user's (Database User) username. When entering your password, make sure that any special characters are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

- Copie a string de conexão substituindo <password> pela senha de conexão do BD.

`mongodb+srv://betopinheiro1005:angstron1005@cluster0.gnwt9.mongodb.net/test`

### config\db.js

```
if(process.env.NODE_ENV == "production"){
 module.exports = {mongoURI: " mongodb+srv://betopinheiro1005:angstron1005@cluster0.gnwt9.mongodb.net/test "}
} else {
 module.exports = {mongoURI: "mongodb://localhost/blogapp"}
}
```

## app.js

```
/* Carregando módulos */

const express = require('express')
const handlebars = require('express-handlebars')
const bodyParser = require('body-parser')
const app = express()
const admin = require('./routes/admin')
const usuarios = require('./routes/usuario')
const path = require('path')
const mongoose = require('mongoose')
const session = require('express-session')
const flash = require('connect-flash')
require('./models/Postagem')
const Postagem = mongoose.model("postagens")
require('./models/Categoria')
const Categoria = mongoose.model("categorias")
const passport = require("passport")
require("./config/auth")(passport)
const db = require("./config/db")

/* Configurações */

// Session
app.use(session({
 secret: "cursodenode",
 resave: true,
 saveUninitialized: true
}))
app.use(passport.initialize())
app.use(passport.session())
app.use(flash())

// Middleware
app.use((req, res, next) => {
 res.locals.success_msg = req.flash("success_msg")
 res.locals.error_msg = req.flash("error_msg")
 res.locals.error = req.flash("error")
 res.locals.user = req.user || null
 next()
})

// Body Parser
app.use(bodyParser.urlencoded({extend:false}))
app.use(bodyParser.json())

// Handlebars
app.engine('handlebars', handlebars({defaultLayout: 'main'}))
app.set('view engine', 'handlebars')

// Mongoose
mongoose.Promise = global.Promise
mongoose.connect(db.mongoURI).then(() => {
 console.log('Mongo conectado...')
})
```

```

}).catch((err) => {
 console.log("Houve um erro ao se conectar ao MongoDB: " + err)
});

// Public
app.use(express.static(path.join(__dirname, "public")))

// app.use((req, res, next) => {
// console.log('Oi, eu sou um middleware!')
// next()
//})

/* Rotas */

app.get('/', (req, res) => {
 Postagem.find().populate("categoria").lean().sort({data: "desc"}).then((postagens) => {
 res.render("index", {postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro interno!")
 res.redirect('/404')
 })
})

app.get('/404', (req, res) => {
 res.send('Erro 404!')
})

app.get('/postagem/:slug', (req, res) => {
 Postagem.findOne({slug: req.params.slug}).lean().then((postagem) =>{
 if(postagem){
 res.render("postagem/index", {postagem: postagem})
 } else {
 req.flash("error_msg", "Essa postagem não existe!")
 res.redirect("/")
 }
 }).catch((err) => {
 req.flash("error_msg", "Houve um erro interno")
 res.redirect("/")
 })
})

app.get('/categorias', (req, res) => {
 Categoria.find().lean().then((categorias) => {
 res.render("categorias/index", {categorias: categorias})
 }).catch(() => {
 req.flash("error_msg", "Erro interno ao listar categorias!")
 res.redirect("/")
 })
})

app.get('/categorias/:slug', (req, res) => {
 Categoria.findOne({slug: req.params.slug}).then((categoria) => {
 if(categoria){
 Postagem.find({categoria: categoria._id}).lean().then((postagens) => {
 res.render("categorias/postagens", {categoria: categoria, postagens: postagens})
 }).catch(() => {
 req.flash("error_msg", "Houve um erro ao listar os posts")
 })
 }
 })
})

```

```
 res.redirect("/")
 })
} else {
 req.flash("error_msg", "Esta categoria não existe!")
 res.redirect("/")
}
}).catch((err) => {
 req.flash("error_msg", "Houve um erro interno ao carregar a página dessa categoria!")
 res.redirect("/")
})
}

app.get('/404', (req, res) => {
 res.send('Erro 404!')
})

app.get('/posts', (req, res) => {
 res.send('Lista de posts')
})

app.use('/admin', admin)
app.use('/usuarios', usuarios)

/* Outros */

const PORT = process.env.PORT || 8081

app.listen(PORT, () => {
 console.log("Servidor rodando!")
})
```

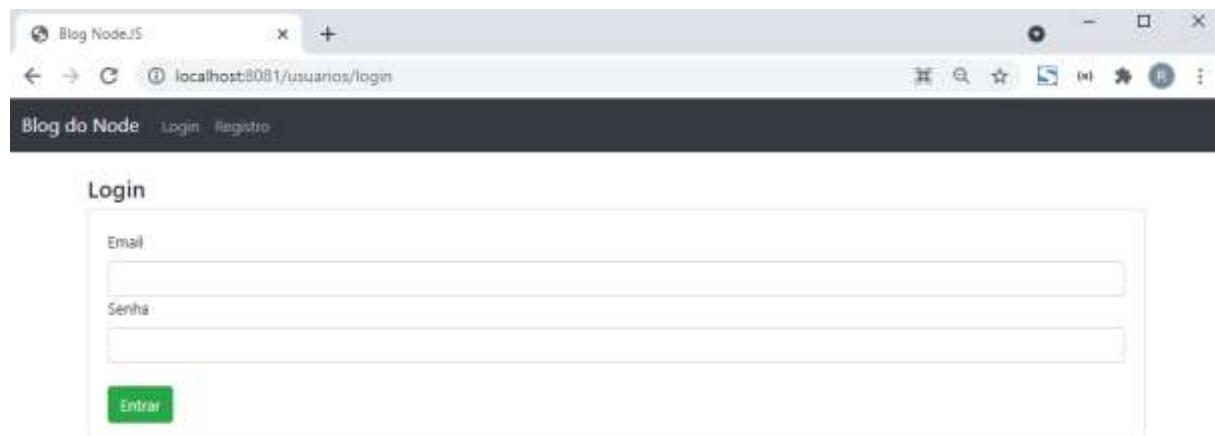
- No terminal, na pasta do projeto, entre com o comando:

**npm start**

```
C:\guia_programador\nodejs\blogapp>npm start
> blogapp@1.0.0 start C:\guia_programador\nodejs\blogapp
> node app.js

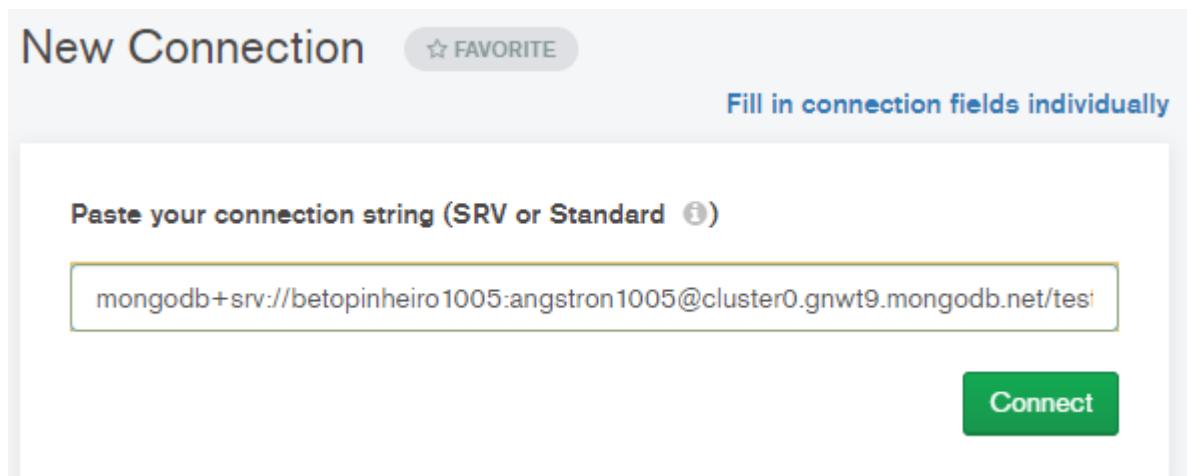
body-parser deprecated undefined extended: provide extended option app.js:43:22
(node:2796) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:2796) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient
constructor.
Servidor rodando!
Mongo conectado...
[]
```

<http://localhost:8081/usuarios/login>

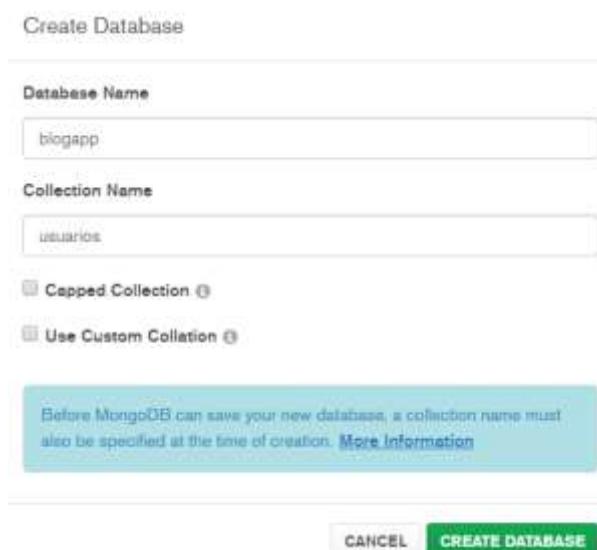


## MongoDB Compass

- Abra o MongoDB Compass e cole a string de conexão.



- Depois clique no botão "Connect"
- Clique no botão "CREATE DATABASE"



\* MongoDB Compass - cluster0.gnat0.mongodb.net

Connect View Help

Local

▼ 3 DBS 19 COLLECTIONS 0 VARIANTS

HOSTS  
cluster0-shard-00-02.gnat...  
cluster0-shard-00-01.gnat...  
cluster0-shard-00-00.gnat...

CLUSTER  
Replica Set (alias-kmig-e...  
3 Nodes)

EDITION  
MongoDB 4.4.0 Enterprise

CREATE DATABASE

Databases Performance

Database Name	Storage Size	Collections	Indexes
admin	0.0B	0	0
blogapp	12.0KB	3	3
local	0.0B	7	0

Filter your data

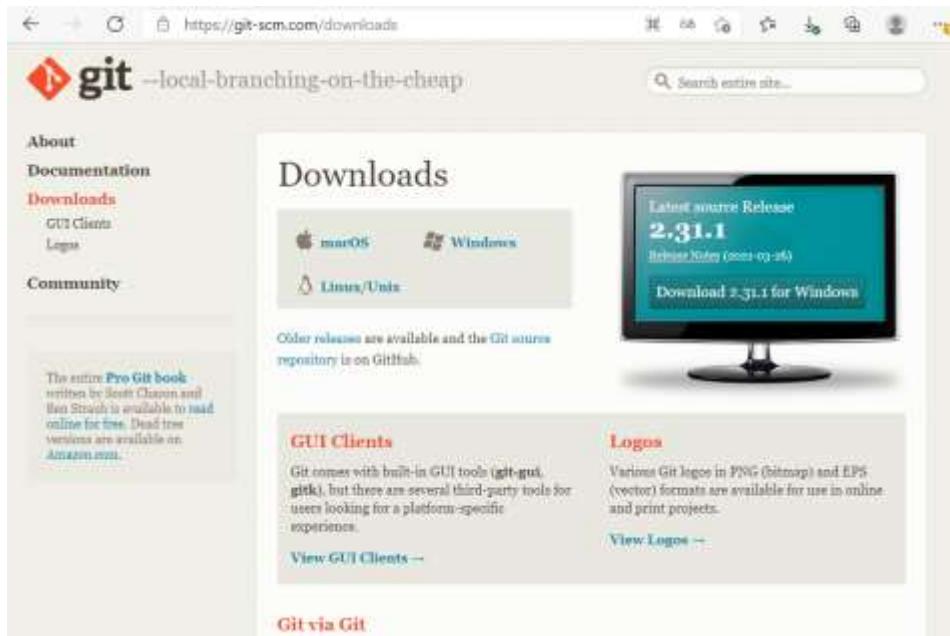
- > admin
- > blogapp
- > local

The screenshot shows the MongoDB Compass interface connected to a cluster named 'cluster0.gnat0.mongodb.net'. On the left, a sidebar displays the cluster configuration with three hosts: 'cluster0-shard-00-02.gnat...', 'cluster0-shard-00-01.gnat...', and 'cluster0-shard-00-00.gnat...'. Below this, it shows a 'CLUSTER' section for a 'Replica Set' with three nodes. The 'EDITION' is listed as 'MongoDB 4.4.0 Enterprise'. The main area is titled 'Databases' and shows a table with three rows: 'admin' (0.0B storage, 0 collections, 0 indexes), 'blogapp' (12.0KB storage, 3 collections, 3 indexes), and 'local' (0.0B storage, 7 collections, 0 indexes). A 'CREATE DATABASE' button is visible at the top right of the table. A search bar labeled 'Filter your data' is present, and a sidebar on the left lists the databases: 'admin', 'blogapp', and 'local'.

## Aula 63 - Como fazer Deploy na Heroku - Parte 2

- Para fazer deploy na Heroku é necessário o GIT.

<https://git-scm.com/downloads>



- Baixe-o e instale no seu computador.

git -

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/guia_programador/nodejs/blogapp
$ git -
unknown option: -
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
 [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
 [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
 [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
 <command> [<args>]
```

git --version

```
beto1@DESKTOP-85HCHH0 MINGW64 /c/guia_programador/nodejs/blogapp
$ git --version
git version 2.30.0.windows.1
```

.gitignore

node\_modules

- No terminal, na pasta do projeto, entre com os comandos:

```
git init
```

```
C:\guia_programador\nodejs\blogapp>git init
Initialized empty Git repository in C:/guia_programador/nodejs/blogapp/.git/
```

```
git add .
```

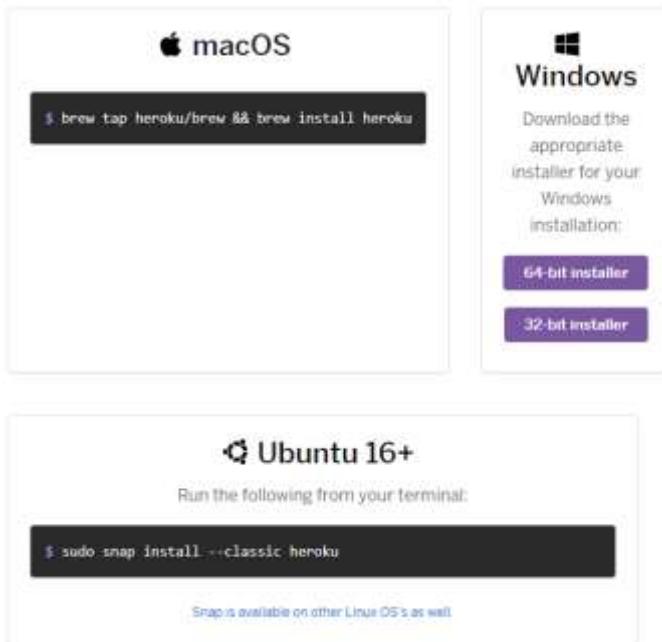
```
C:\guia_programador\nodejs\blogapp>git add .
warning: LF will be replaced by CRLF in package-lock.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in package.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/css/bootstrap-grid.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/css/bootstrap-grid.min.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/css/bootstrap-reboot.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/css/bootstrap-reboot.min.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/css/bootstrap.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/css/bootstrap.min.css.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/js/bootstrap.bundle.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/js/bootstrap.bundle.min.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/js/bootstrap.js.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in public/js/bootstrap.min.js.
The file will have its original line endings in your working directory
```

```
git commit -m "initial commit"
```

```
C:\guia_programador\nodejs\blogapp>git commit -am "initial commit"
[master (root-commit) 6451b68] initial commit
49 files changed, 23568 insertions(+)
create mode 100644 .gitignore
create mode 100644 app.js
create mode 100644 config/auth.js
create mode 100644 config/db.js
create mode 100644 helpers/eAdmin.js
create mode 100644 models/Categoria.js
create mode 100644 models/Postagem.js
create mode 100644 models/Usuario.js
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 public/css/.DS_Store
create mode 100644 public/css/bootstrap-grid.css
create mode 100644 public/css/bootstrap-grid.css.map
create mode 100644 public/css/bootstrap-grid.min.css
create mode 100644 public/css/bootstrap-grid.min.css.map
create mode 100644 public/css/bootstrap-reboot.css
create mode 100644 public/css/bootstrap-reboot.css.map
create mode 100644 public/css/bootstrap-reboot.min.css
create mode 100644 public/css/bootstrap-reboot.min.css.map
create mode 100644 public/css/bootstrap.css
create mode 100644 public/css/bootstrap.css.map
create mode 100644 public/css/bootstrap.min.css
create mode 100644 public/css/bootstrap.min.css.map
create mode 100644 public/js/bootstrap.bundle.js
create mode 100644 public/js/bootstrap.bundle.js.map
create mode 100644 public/js/bootstrap.bundle.min.js
create mode 100644 public/js/bootstrap.bundle.min.js.map
```

- Baixe e instale "**Heroku CLI**". É uma ferramenta de linha de comando.

<https://devcenter.heroku.com/articles/heroku-cli>



- Crie uma conta na Heroku.
- Faça o login. No terminal, na pasta do projeto, entre com o comando:

**heroku login**

```
beto1@DESKTOP-B5HICHE MINGW64 /c/guia_programador/nodejs/blogapp
$ heroku login
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/55407b5c-2195-4031-a5d3-150c9da4cd22?requestor=SFMyNTY.g2gDbQ
AAAAA4x0DYuMjA0LjEx0C41MwMGAAfVx0DF5AWIAAVGA.pvCtCtAr765uvA2tsEni294nidSda0Azb0ydJwfQDyLrc
Logging in... done
logged in as betopinheiro1005@yahoo.com.br
```

- Crie uma nova aplicação:

**heroku create app-blog-nodejs**

```
C:\guia_programador\nodejs\blogapp>heroku create app-blog-nodejs
Creating ⚡ app-blog-nodejs... done
https://app-blog-nodejs.herokuapp.com/ | https://git.heroku.com/app-blog-nodejs.git
```

Install the Heroku CLI

Download and install the [Heroku CLI](#).

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

Create a new Git repository

Initialize a git repository in a new or existing directory

```
$ cd my-project/
$ git init
$ heroku git:remote -a app-blog-nodejs
```

Deploy your application

Commit your code to the repository and deploy it to Heroku using Git.

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```



You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).

**heroku git:remote -a app-blog-nodejs**

```
C:\guia_programador\nodejs\blogapp>heroku git:remote -a app-blog-nodejs
set git remote heroku to https://git.heroku.com/app-blog-nodejs.git
```

## Fazendo o deploy

**git push heroku master**

**heroku open**