

# Curso de Node JS

## Celke (Cesar Szpak)

[https://www.youtube.com/watch?v=jqrKQEJ6DpY&list=PLmY5AEiqDWwBHJ3i\\_8MDSszXXRTcFdkSu](https://www.youtube.com/watch?v=jqrKQEJ6DpY&list=PLmY5AEiqDWwBHJ3i_8MDSszXXRTcFdkSu)

Resumo feito por Roberto Pinheiro

## Aula 02 - Como usar o Node na prática

**index.js**

```
console.log("Olá Mundo!");
```

node index.js

```
C:\celke\nodejs\aula02>node index.js
Olá Mundo!
```

**index.js**

```
var discountFunc = require('./modules/calDiscount');

console.log("Gerenciador Financeiro");

var client = "César Szpak";

console.log("Cliente: " + client);

valProduct = 100;
valDiscount = 37;

var finalValue = discountFunc(valProduct, valDiscount);

console.log("Valor final do produto: R$ " + finalValue);
```

```
C:\celke\nodejs\aula02>node index.js
Gerenciador Financeiro
Cliente: César Szpak
Valor final do produto: R$ 63
```

## Aula 03 - Criando o servidor http

index.js

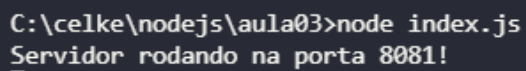
```
const http = require('http');

const PORT = 8081;

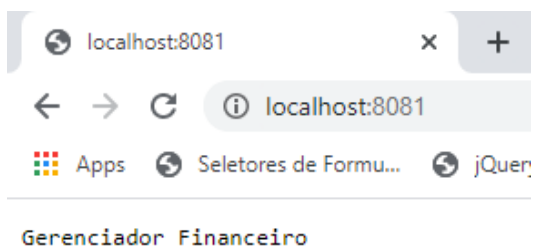
http.createServer(function(req, res){
  res.end("Gerenciador Financeiro");
}).listen(PORT);

console.log("Servidor rodando na porta " + PORT + "!");
```

node index.js



```
C:\celke\nodejs\aula03>node index.js
Servidor rodando na porta 8081!
```



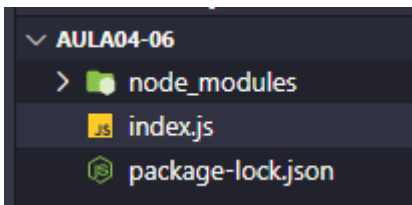
## Aula 04 - Como instalar o Express e criar rotas no Nodejs

- Vamos instalar o Express para poder trabalhar com rotas.

`npm install express --save`

```
C:\celke\nodejs\aula04-06>npm install express --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\celke\nodejs\aula04-06\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:\celke\nodejs\aula04-06\package.json'
npm WARN aula04-06 No description
npm WARN aula04-06 No repository field.
npm WARN aula04-06 No README data
npm WARN aula04-06 No license field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 41.07s
found 0 vulnerabilities
```



### index.js

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

const app = express();

app.get('/', function(req, res){
  res.send("Gerenciador Financeiro");
});

app.get('/contato', function(req, res){
  res.send("Página de contato");
});

app.get('/sobre-empresa', function(req, res){
  res.send("Página sobre empresa");
});

app.get('/blog', function(req, res){
  res.send("Página do blog");
});

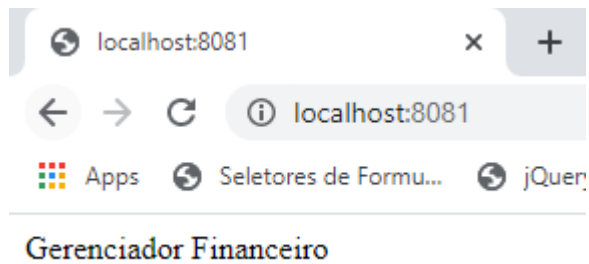
// Iniciando o servidor

const PORT = 8081;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT + "!");
```

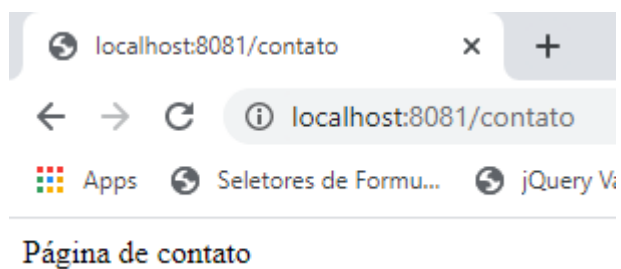
node index.js

```
C:\celke\nodejs\aula04-06>node index.js
Servidor rodando na porta 8081!
```

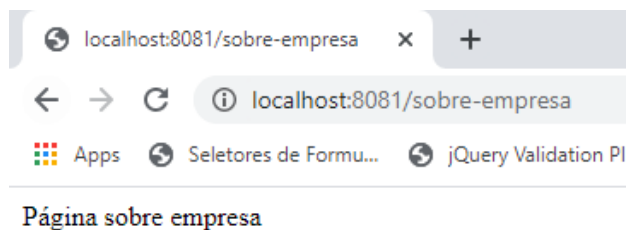
localhost:8081



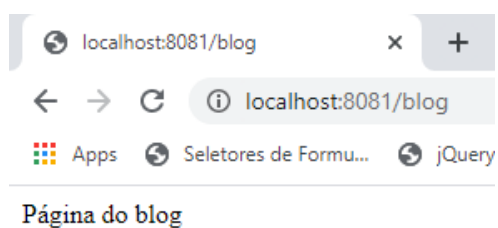
localhost:8081/contato



localhost:8081/sobre-empresa



localhost:8081/blog



## Aula 05 - Como instalar o Nodemon no Nodejs

- O nodemon faz com que o servidor seja recarregado automaticamente

`npm install -g nodemon`

```
C:\celke\nodejs\aula04-06>npm install -g nodemon
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

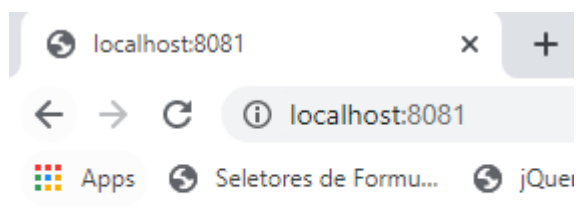
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.3.1 (node_modules\nodemon\node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ nodemon@2.0.7
updated 1 package in 30.011s
```

`nodemon index.js`

```
C:\celke\nodejs\aula04-06>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 porta 8081!
```

localhost:8081



Gerenciador Financeiro

- Para parar o servidor: `<Ctrl><C>`

## index.js

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

const app = express();

app.get('/', function(req, res){
  res.send("Gerenciador de Estoque");
});

app.get('/contato', function(req, res){
  res.send("Página de contato");
});

app.get('/sobre-empresa', function(req, res){
  res.send("Página sobre empresa");
});

app.get('/blog', function(req, res){
  res.send("Página do blog");
});

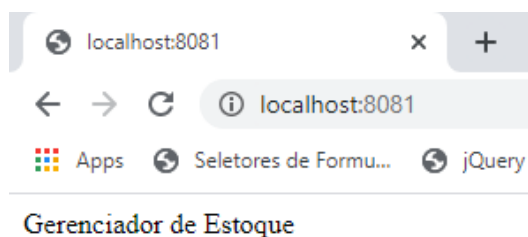
// Iniciando o servidor

const PORT = 8081;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT + "!");
```

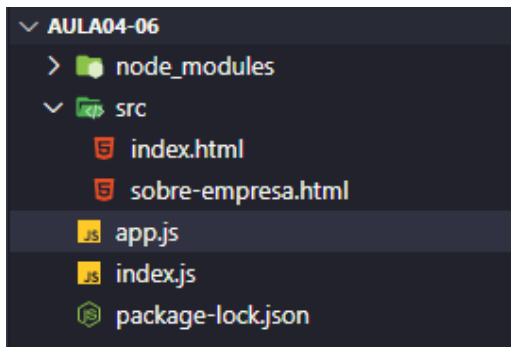
- Depois de salvar o arquivo:

```
C:\celke\nodejs\aula04-06>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 porta 8081!
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Servidor rodando na porta 8081!
█
```

localhost:8081



## Aula 06 - Como carregar arquivo HTML no Nodejs



### app.js

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

const app = express();

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

app.get('/contato', function(req, res){
  res.sendFile(__dirname + "/src/sobre-empresa.html");
});

app.get('/sobre-empresa', function(req, res){
  res.send("Página sobre empresa");
});

app.get('/blog', function(req, res){
  res.send("Página do blog");
});

// Iniciando o servidor

const PORT = 8081;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT + "!");
```

## src\index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Celke</title>
</head>
<body>
  <h1>Página inicial do site</h1>
  <p>Sed sed pharetra ipsum, ut condimentum enim. Praesent luctus velit vitae semper feugiat. Cras massa est, iaculis sed turpis ultricies, aliquam dapibus nisl. Duis tincidunt mattis leo, id hendrerit turpis auctor sit amet. Nullam eu nulla eros. Nunc luctus pretium scelerisque. Sed varius posuere aliquet. Proin at laoreet metus. Vivamus et magna elementum, commodo tellus eget, fermentum orci. Sed rutrum nisl eu est iaculis vestibulum. Duis ornare et mi ac ultricies. Fusce iaculis mi eget diam condimentum, sed placerat tortor tempus. Pellentesque id sollicitudin nisi, a tempus diam. In cursus lacus eu nisi porttitor, nec dictum massa efficitur. Nunc felis sapien, ultrices sit amet est quis, pretium fringilla nunc. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.</p>
</body>
</html>
```

## src\sobre-empresa.html

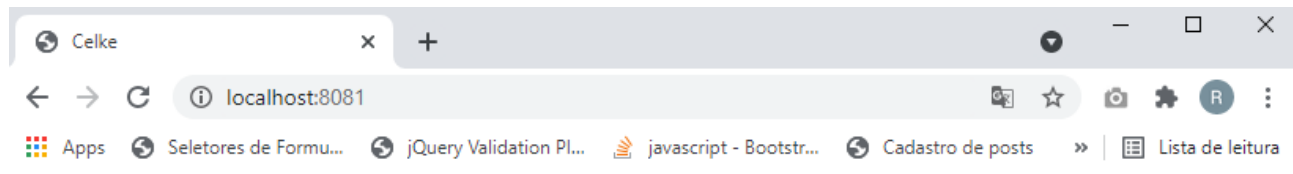
```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <title>Celke - Sobre Empresa</title>
</head>
<body>
  <h1>Página sobre empresa</h1>
  <p>Sed sed pharetra ipsum, ut condimentum enim. Praesent luctus velit vitae semper feugiat. Cras massa est, iaculis sed turpis ultricies, aliquam dapibus nisl. Duis tincidunt mattis leo, id hendrerit turpis auctor sit amet. Nullam eu nulla eros. Nunc luctus pretium scelerisque. Sed varius posuere aliquet. Proin at laoreet metus. Vivamus et magna elementum, commodo tellus eget, fermentum orci. Sed rutrum nisl eu est iaculis vestibulum. Duis ornare et mi ac ultricies. Fusce iaculis mi eget diam condimentum, sed placerat tortor tempus. Pellentesque id sollicitudin nisi, a tempus diam. In cursus lacus eu nisi porttitor, nec dictum massa efficitur. Nunc felis sapien, ultrices sit amet est quis, pretium fringilla nunc. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.</p>
</body>
</html>
```

## nodemon app.js

```
C:\celke\nodejs\aula04-06>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`
Servidor rodando na porta 8081!
█
```



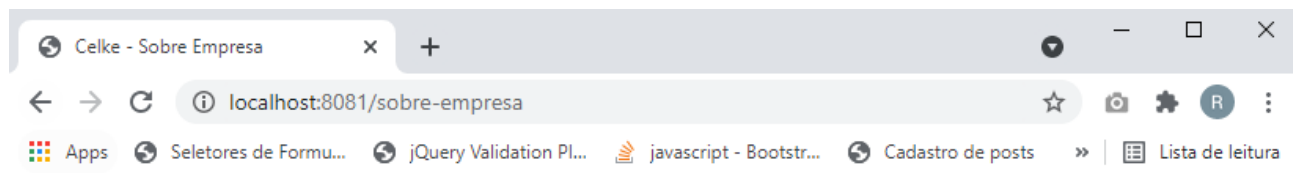
localhost:8081



## Página inicial do site

Sed sed pharetra ipsum, ut condimentum enim. Praesent luctus velit vitae semper feugiat. Cras massa est, iaculis sed turpis ultricies, aliquam dapibus nisl. Duis tincidunt mattis leo, id hendrerit turpis auctor sit amet. Nullam eu nulla eros. Nunc luctus pretium scelerisque. Sed varius posuere aliquet. Proin at laoreet metus. Vivamus et magna elementum, commodo tellus eget, fermentum orci. Sed rutrum nisl eu est iaculis vestibulum. Duis ornare et mi ac ultricies. Fusce iaculis mi eget diam condimentum, sed placerat tortor tempus. Pellentesque id sollicitudin nisi, a tempus diam. In cursus lacus eu nisi porttitor, nec dictum massa efficitur. Nunc felis sapien, ultrices sit amet est quis, pretium fringilla nunc. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

localhost:8081/sobre-empresa



## Página sobre empresa

Sed sed pharetra ipsum, ut condimentum enim. Praesent luctus velit vitae semper feugiat. Cras massa est, iaculis sed turpis ultricies, aliquam dapibus nisl. Duis tincidunt mattis leo, id hendrerit turpis auctor sit amet. Nullam eu nulla eros. Nunc luctus pretium scelerisque. Sed varius posuere aliquet. Proin at laoreet metus. Vivamus et magna elementum, commodo tellus eget, fermentum orci. Sed rutrum nisl eu est iaculis vestibulum. Duis ornare et mi ac ultricies. Fusce iaculis mi eget diam condimentum, sed placerat tortor tempus. Pellentesque id sollicitudin nisi, a tempus diam. In cursus lacus eu nisi porttitor, nec dictum massa efficitur. Nunc felis sapien, ultrices sit amet est quis, pretium fringilla nunc. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

# Aula 07 - Node + MySQL + Workbench - Como conectar o Node com banco de dados

## Instalação do MySQL

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

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

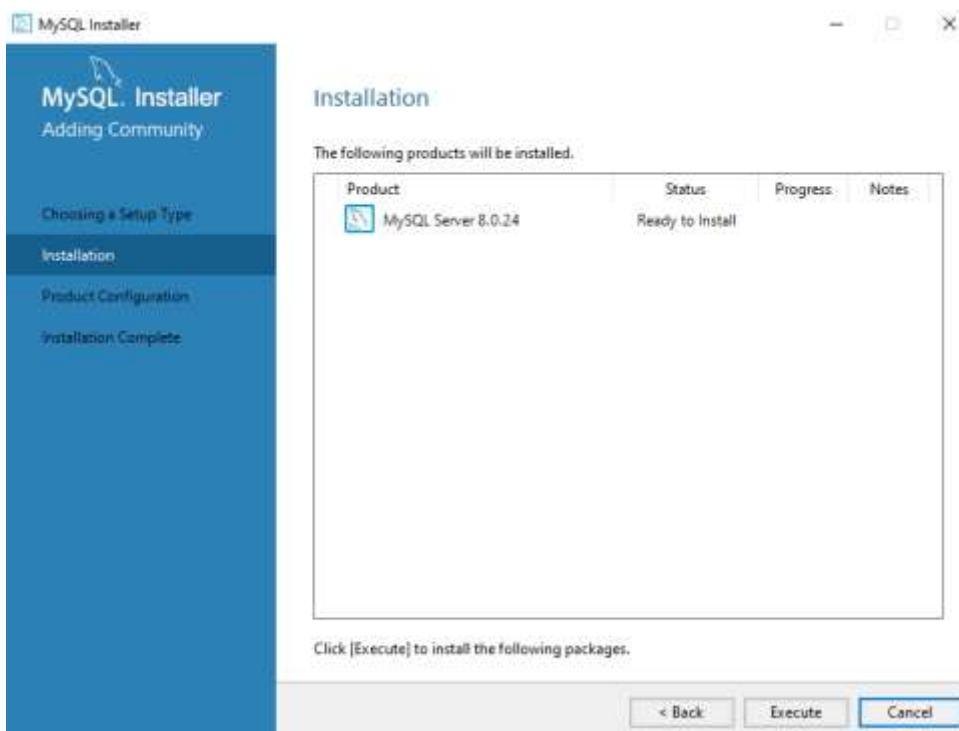
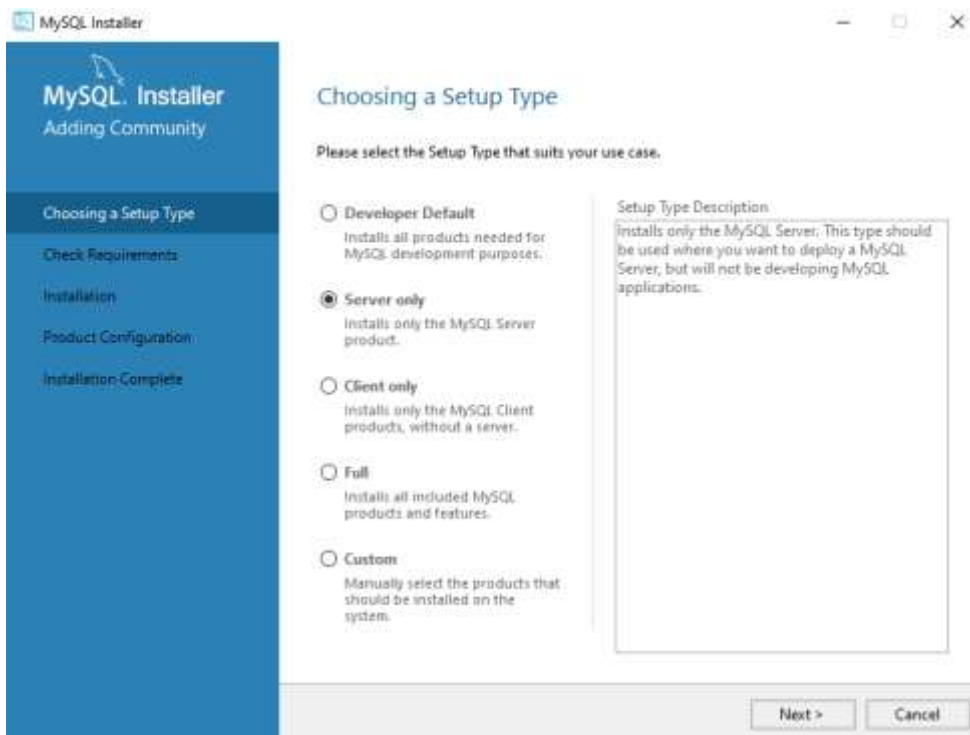


<https://dev.mysql.com/downloads/>

- Baixe e instale o MySQL Community Server

## 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



MySQL Installer

MySQL Server 8.0.24

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Apply Configuration

## Type and Networking

### Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: Development Computer

### Connectivity

Use the following controls to select how you would like to connect to this server.

☒ TCP/IP Port: 3306 X Protocol Port: 33060

☒ Open Windows Firewall ports for network access

☐ Named Pipe Pipe Name: MYSQL

☐ Shared Memory Memory Name: MYSQL

### Advanced Configuration

Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.

☐ Show Advanced and Logging Options

Next > Cancel

MySQL Installer

MySQL Server 8.0.24

Type and Networking

Authentication Method

Accounts and Roles


Windows Service

Apply Configuration

## Authentication Method

☒ **Use Strong Password Encryption for Authentication (RECOMMENDED)**

MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.



Attention: This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching\_sha2\_password authentication).

Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.

☐ **Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)**

Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:

- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.

Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security.

< Back Next > Cancel

# MySQL. Installer

MySQL Server 8.0.24

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Apply Configuration

## Accounts and Roles

### Root Account Password

Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

●●●●●●●●

Repeat Password:

●●●●●●●●

Password strength: **Weak**

### MySQL User Accounts

Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role

Add User

Edit User

Delete

&lt; Back

Next &gt;

Cancel

# MySQL. Installer

MySQL Server 8.0.24

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Apply Configuration

## Windows Service

☒ Configure MySQL Server as a Windows Service

### Windows Service Details

Please specify a Windows Service name to be used for this MySQL Server instance. A unique name is required for each instance.

Windows Service Name: MySQL80

☒ Start the MySQL Server at System Startup

### Run Windows Service as ...

The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

☒ Standard System Account

Recommended for most scenarios.

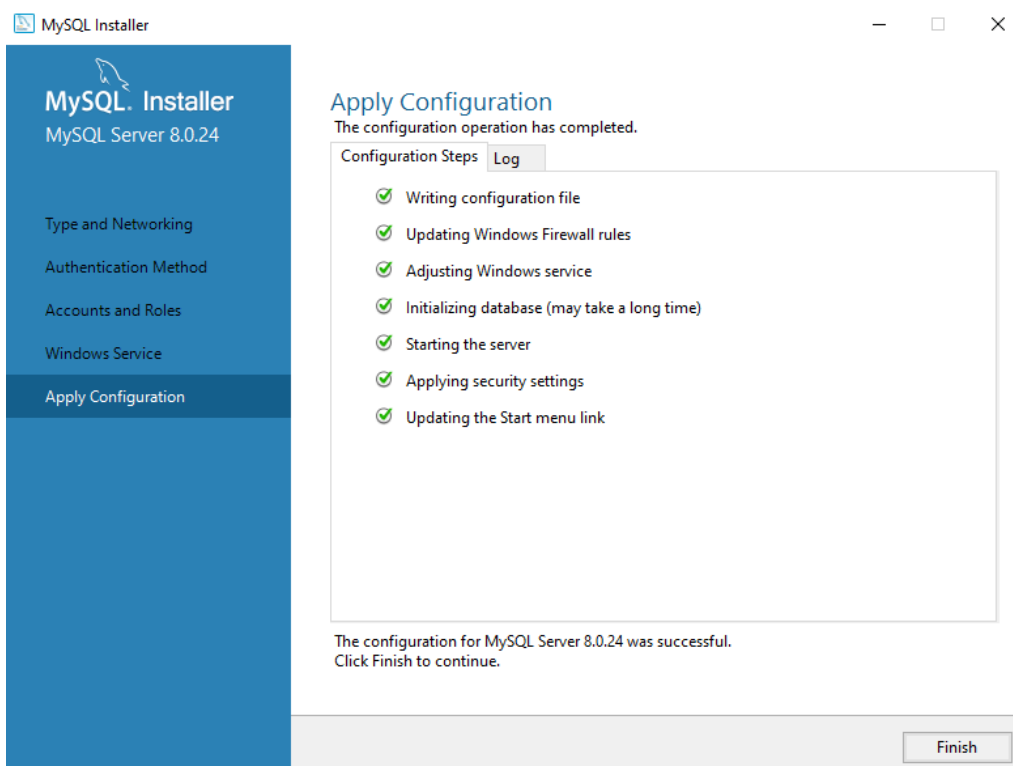
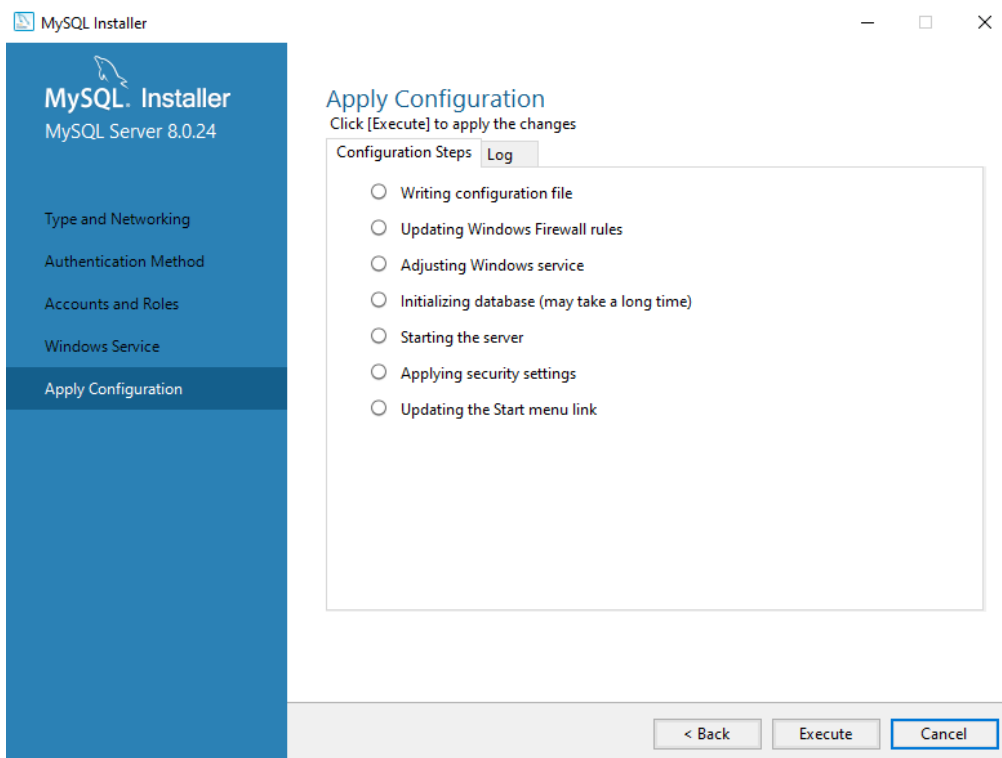
☐ Custom User

An existing user account can be selected for advanced scenarios.

&lt; Back

Next &gt;

Cancel

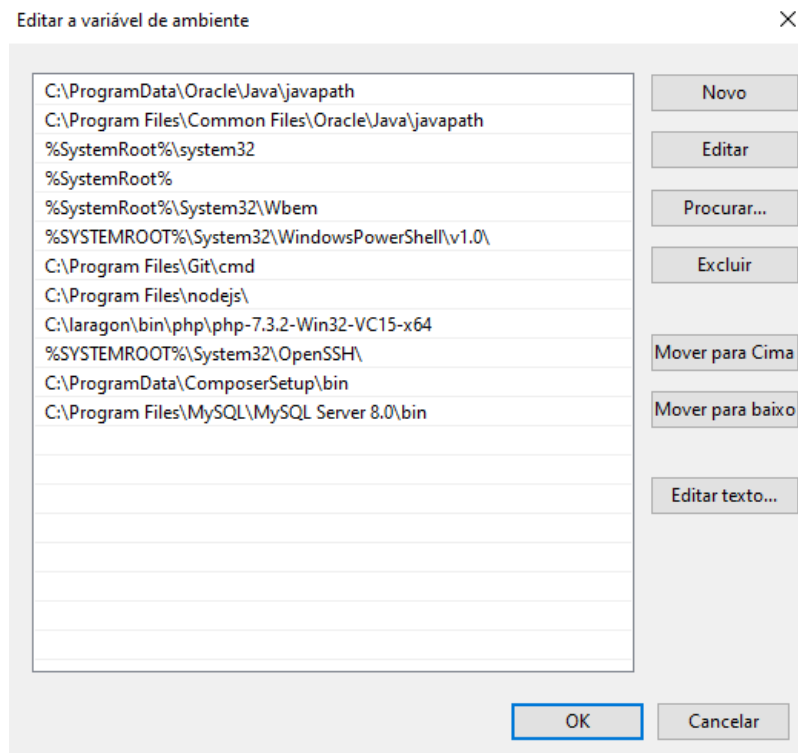


Inserir o path em variáveis de ambiente:

- 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 -h localhost -u root -p**

E insira a senha.

**show databases;**

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| postapp |
| sistema_cadastro |
| sys |
| test |
+-----+
7 rows in set (0.47 sec)
```

create database celke;

```
mysql> create database celke;  
Query OK, 1 row affected (0.60 sec)
```

use celke;

```
mysql> use celke;  
Database changed
```

create table users (nome VARCHAR(220), email VARCHAR(220));

```
mysql> use celke;  
Database changed  
mysql> create table users (nome VARCHAR(220), email VARCHAR(220));  
Query OK, 0 rows affected (2.42 sec)
```

show tables;

```
mysql> show tables;  
+-----+  
| Tables_in_celke |  
+-----+  
| users            |  
+-----+  
1 row in set (0.23 sec)
```

insert into users (nome, email) values ('Cesar', 'cesar@celke.com.br');

```
mysql> insert into users (nome, email) values ('Cesar', 'cesar@celke.com.br');  
Query OK, 1 row affected (0.18 sec)
```

select \* from users;

```
mysql> select * from users;  
+-----+-----+  
| nome | email                |  
+-----+-----+  
| Cesar | cesar@celke.com.br |  
+-----+-----+  
1 row in set (0.09 sec)
```



## MySQL Workbench

- Baixe e instale o programa MySQL Workbench

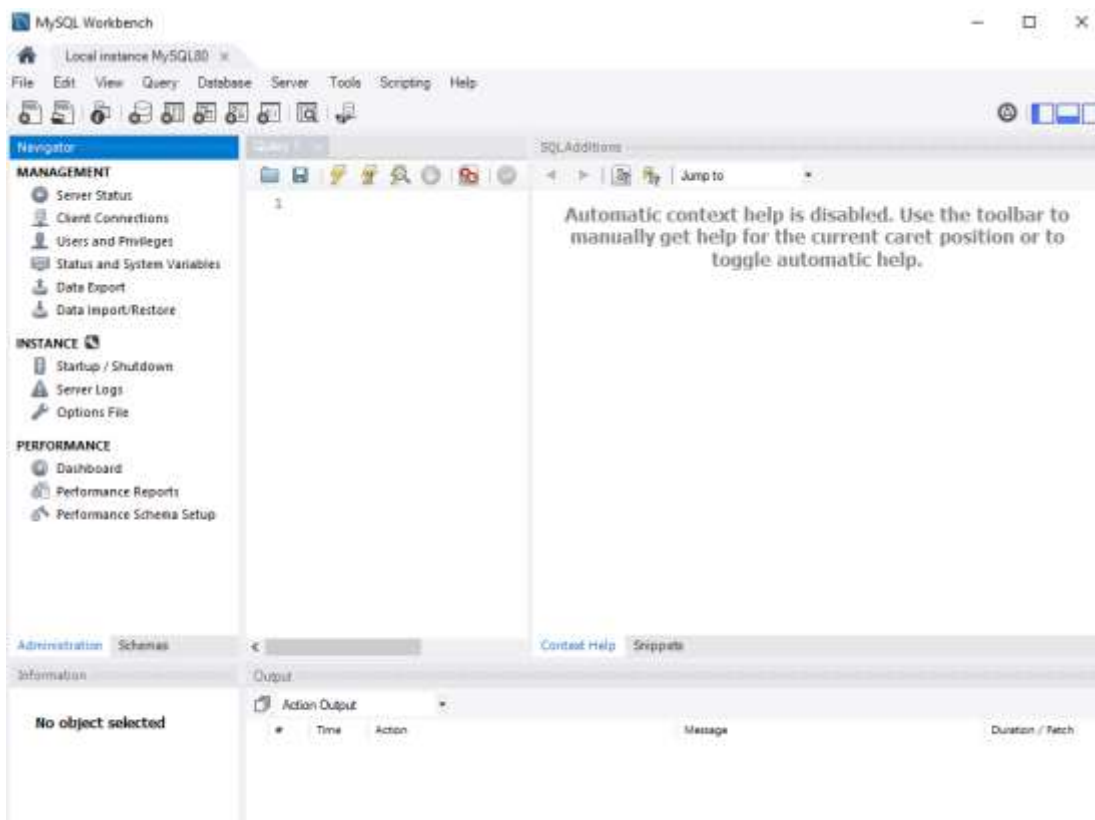
<https://dev.mysql.com/downloads/workbench/>

The screenshot shows the MySQL Community Downloads page for MySQL Workbench. The browser address bar shows the URL [dev.mysql.com/downloads/workbench/](https://dev.mysql.com/downloads/workbench/). The page title is "MySQL Community Downloads" with a sub-header "MySQL Workbench". There are tabs for "General Availability (GA) Releases", "Archives", and "Downloads". The main section is titled "MySQL Workbench 8.0.24". Below this, there is a "Select Operating System:" dropdown menu set to "Microsoft Windows". Under "Recommended Download:", there is a graphic for "MySQL Installer for Windows" with the text "All MySQL Products. For All Windows Platforms. In One Package." and a "Go to Download Page >" button. Below this, it says "Windows (x86, 32 & 64-bit), MySQL Installer MSI". Under "Other Downloads:", there is a table with the following data:

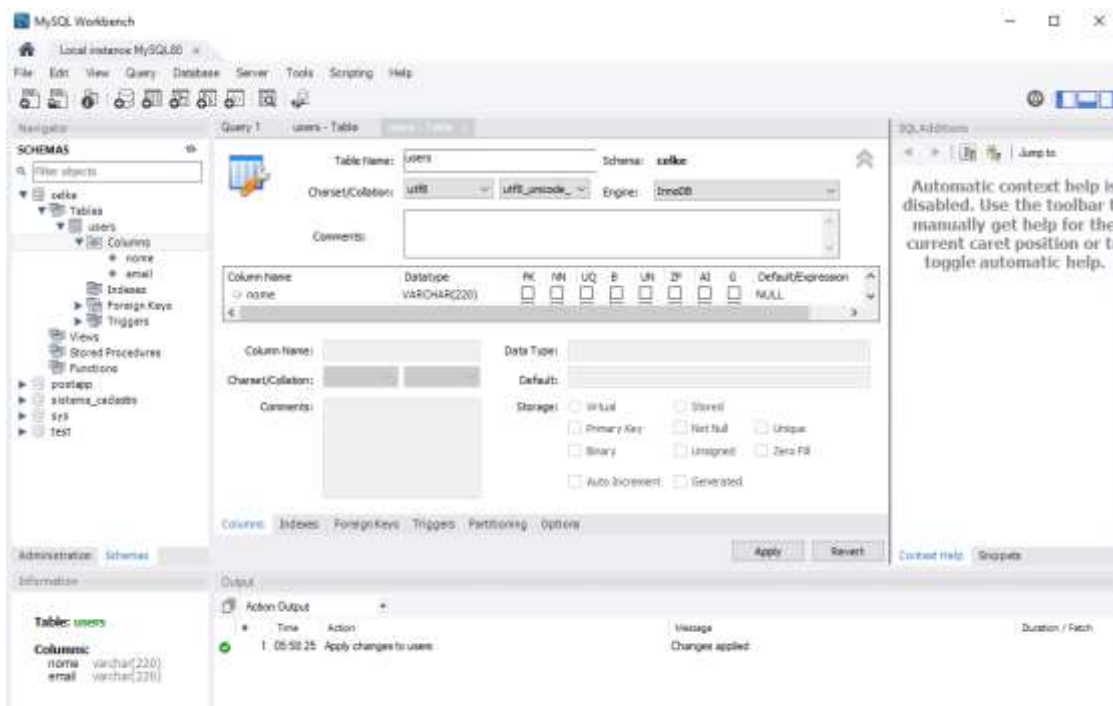
Operating System	Version	Size	Action
Windows (x86, 64-bit), MSI Installer	8.0.24	42.2M	<a href="#">Download</a>

Below the table, there is a link to the download page: [\(mysql-workbench-community-8.0.24-winx64.msi\)](#) and a hash: `MDS: 4535827195f8e950b2f45b09adbe2e39 | Signature`.

The screenshot shows the MySQL Workbench application interface. The title bar says "MySQL Workbench". The menu bar includes "File", "Edit", "View", "Database", "Tools", "Scripting", and "Help". The main window displays a "Welcome to MySQL Workbench" message. Below the message, there is a description: "MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database." There are three links: [Browse Documentation >](#), [Read the Blog >](#), and [Discuss on the Forums >](#). Below these links, there is a section titled "MySQL Connections" with a search icon and the text "Filter connections". There is a list of connections, including "Local instance MySQL80" with a user "root" and host "localhost:3306".



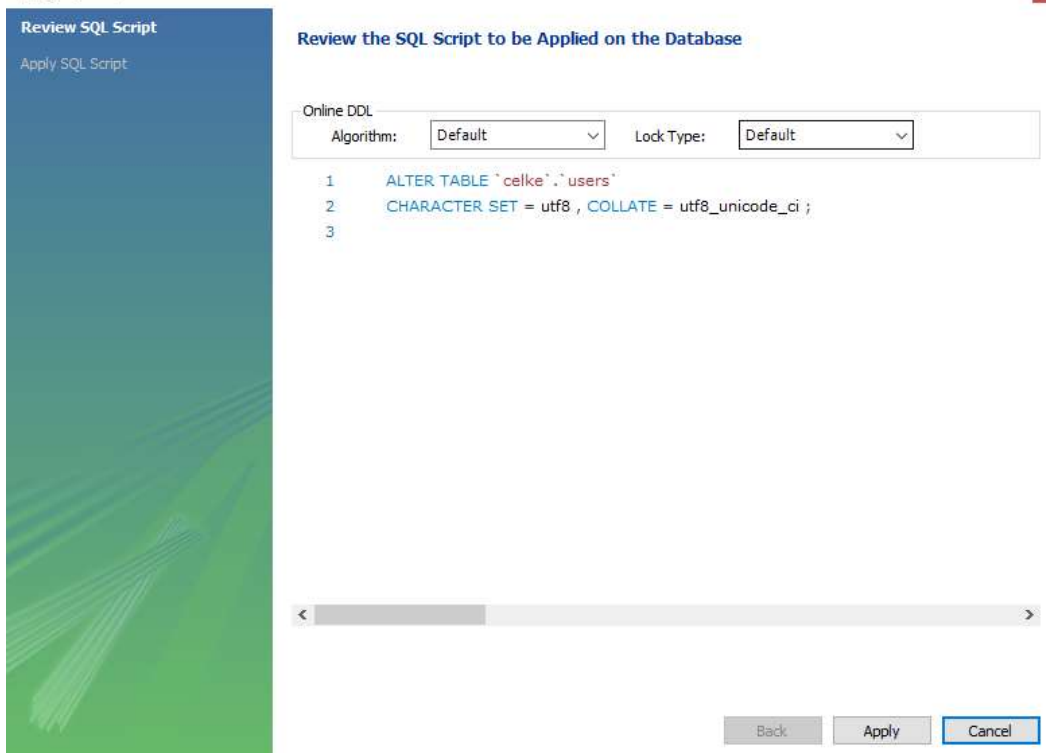
- Clique na aba "Schemas"



Charset: utf8

Collation: utf8\_unicode\_ci

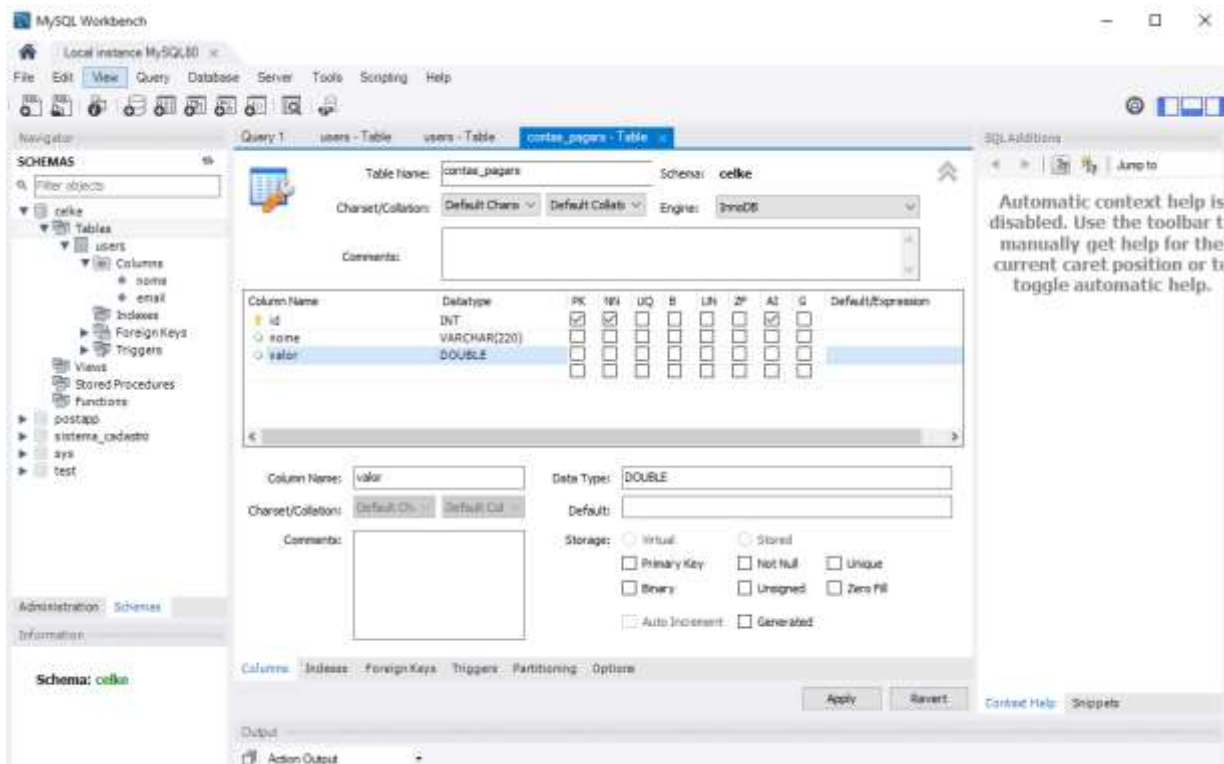
- Clique no botão "Apply"



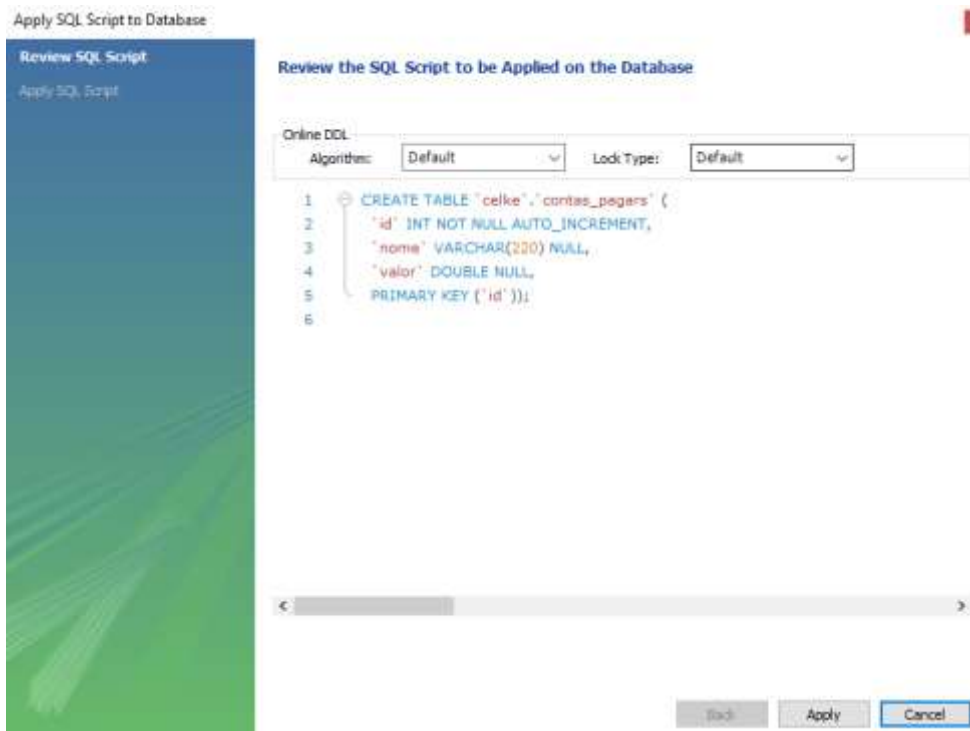
- Clique no botão "Apply"

## Criando uma nova tabela

- Crie uma tabela chamada "contas\_pagar"

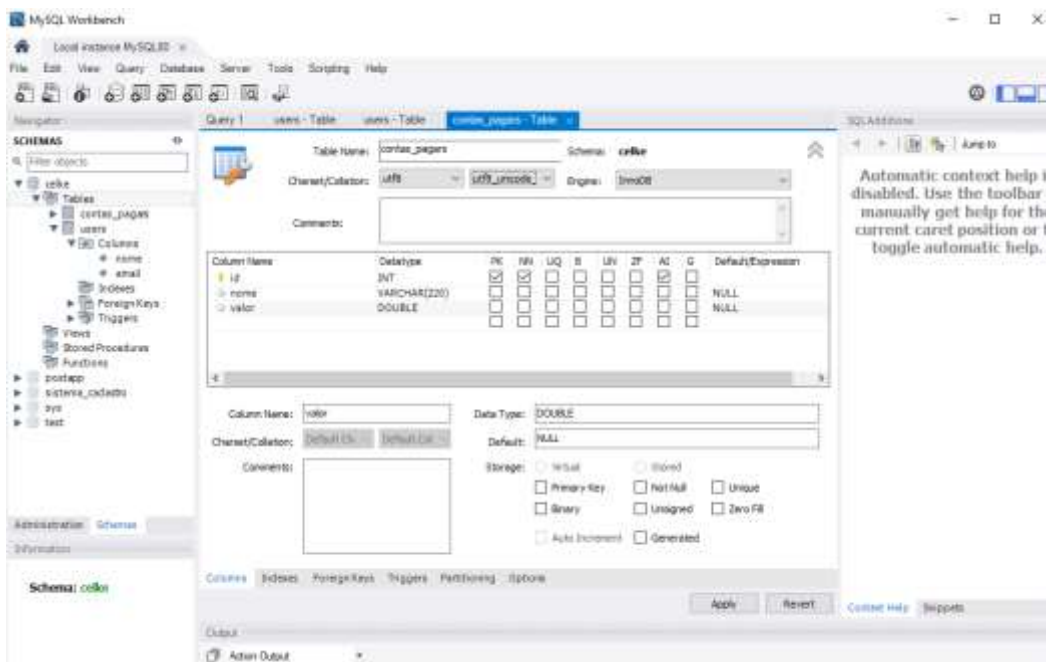


- Clique no botão "Apply"



- Clique novamente no botão "Apply"

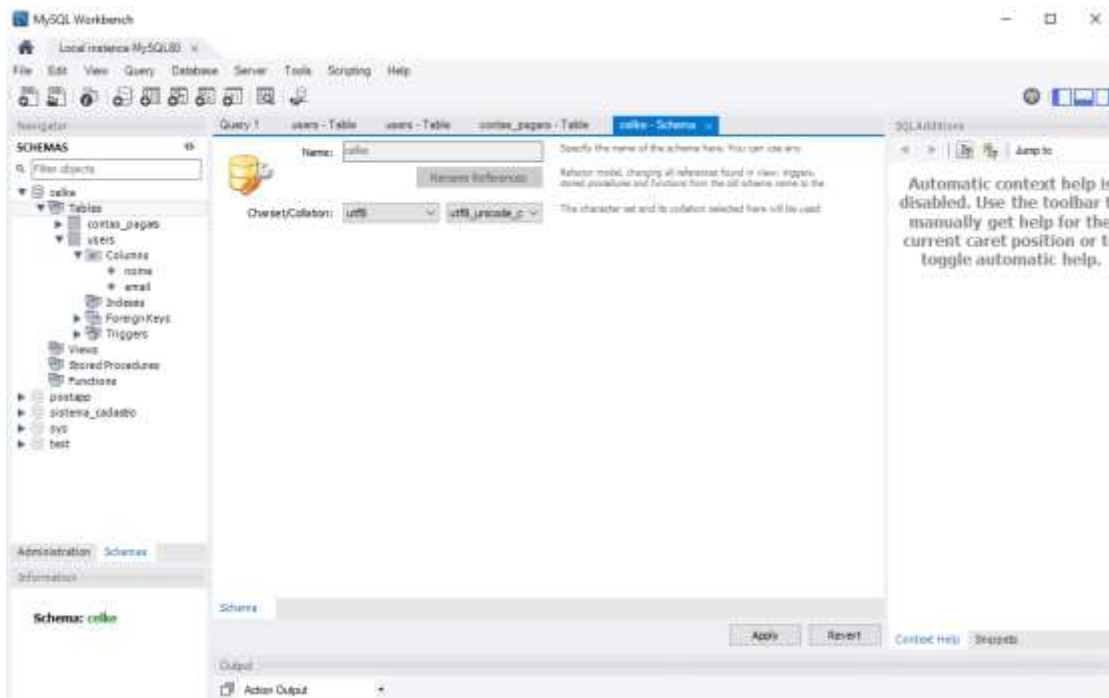
```
mysql> show tables;  
+-----+  
| Tables_in_celke |  
+-----+  
| contas_pagar    |  
| users           |  
+-----+  
2 rows in set (0.13 sec)
```



Charset: **utf8**

Collation: **utf8\_unicode\_ci**

- Clique no botão "**Apply**"



Charset: **utf8**

Collation: **utf8\_unicode\_ci**

- Clique no botão "**Apply**"

## Implementando a conexão com o banco de dados

- Em `C:\celke\nodejs` crie uma subpasta chamada `aula07`.

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

Pasta de C:\celke\nodejs

11/05/2021  16:37    <DIR>        .
11/05/2021  16:37    <DIR>        ..
10/05/2021  14:07    <DIR>        aula02
10/05/2021  14:29    <DIR>        aula03
10/05/2021  20:07    <DIR>        aula04-06
11/05/2021  16:36    <DIR>        aula07
               0 arquivo(s)                0 bytes
               6 pasta(s)  118.102.528.000 bytes disponíveis
```

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

`npm init`

```
About to write to C:\celke\nodejs\aula07\package.json:

{
  "name": "curso-nodejs",
  "version": "1.0.0",
  "description": "cursonode",
  "main": "app.js",
  "dependencies": {
    "express": "^4.17.1"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "César Szpak",
  "license": "ISC"
}

Is this OK? (yes) █
```

### package.json

```
{
  "name": "curso-nodejs",
  "version": "1.0.0",
  "description": "cursonode",
  "main": "app.js",
  "dependencies": {
    "express": "^4.17.1"
  },
  "devDependencies": {},
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "César Szpak",
  "license": "ISC"
}
```

## Instalando o pacote mysql

<https://www.npmjs.com/package/mysql#install>

### Install

This is a **Node.js** module available through the **npm registry**.

Before installing, **download and install Node.js**. Node.js 0.6 or higher is required.

Installation is done using the **npm install command**:

```
$ npm install mysql
```

For information about the previous 0.9.x releases, visit the **v0.9 branch**.

Sometimes I may also ask you to install the latest version from Github to check if a bugfix is working. In this case, please do:

```
$ npm install mysqljs/mysql
```

**npm install mysql --save**

```
C:\celke\nodejs\aula07>npm install mysql --save
npm WARN curso-nodejs@1.0.0 No repository field.

+ mysql@2.18.1
added 9 packages from 14 contributors and audited 59 packages in 12.441s
found 0 vulnerabilities
```

## Introduction

This is a node.js driver for mysql. It is written in JavaScript, does not require compiling, and is 100% MIT licensed.

Here is an example on how to use it:

```
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host        : 'localhost',
  user        : 'me',
  password    : 'secret',
  database    : 'my_db'
});

connection.connect();
```

```
connection.connect(function(err) {  
  if (err) {  
    console.error('error connecting: ' + err.stack);  
    return;  
  }  
  
  console.log('connected as id ' + connection.threadId);  
});
```

## Criando um novo usuário

create user 'celkeone'@'localhost' identified with mysql\_native\_password by '123456';

```
mysql> create user 'celkeone'@'localhost' identified with mysql_native_password by '123456';  
Query OK, 0 rows affected (0.11 sec)
```

## Dando permissões de administrador para o usuário criado

grant all privileges on \*.\* to 'celkeone'@'localhost';

```
mysql> grant all privileges on *.* to 'celkeone'@'localhost';  
Query OK, 0 rows affected (0.50 sec)
```



## app.js

```
//Conexao com BD MySQL
const mysql = require('mysql');

//A partir do MySQL 8 apresenta o erro ao utilizar o usuário root para conexão, necessário criar novo
usuário (instrução no Readme)
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'celkeone',
  password: '123456',
  database: 'celke'
});

connection.connect(function (err) {
  if (err) {
    console.error('error connecting: ' + err.stack);
    return;
  }
  console.log('connected as id ' + connection.threadId);
});

connection.query('SELECT * FROM users', function(err, rows, fields){
  if(!err){
    console.log('Resultado: ', rows);
  }else{
    console.log('Erro ao realizar a consulta');
  }
});
```

## nodemon app.js

```
C:\celke\nodejs\aula07>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`
connected as id 63
Resultado: [ RowDataPacket { nome: 'Cesar', email: 'cesar@celke.com.br' } ]
█
```

## Readme.md

Acessar o banco de dados:

```
mysql -h 'servidor' -u 'usuario' -p
```

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

Em seguida digitar a senha do usuário root no banco de dados.

Listar as base de dados:

```
SHOW DATABASES;
```

Criar base de dados:

```
CREATE DATABASE 'nome_da_base_dados';
```

Exemplo: `CREATE DATABASE celke;`

Acessar a base de dados:

```
USE 'nome_da_base_dados';
```

Exemplo: `USE celke;`

Criar nova tabela:

```
CREATE TABLE 'nome_tabela' (coluna1, coluna2,...);
```

Exemplo: `CREATE TABLE users (nome VARCHAR(220), email VARCHAR(220));`

Listar as tabelas da base de dados:

```
SHOW TABLES;
```

Cadastrar registro no banco de dados:

```
INSERT INTO 'nome_tabela' (coluna1, coluna2) VALUES ('valor1_coluna1', 'valor_coluna2');
```

Exemplo: `INSERT INTO users (nome, email) VALUES ('Cesar', 'cesar@celke.com.br');`

Listar registros da tabela:

```
SELECT * FROM 'nome_tabela';
```

Exemplo: `SELECT * FROM users;`

Criar usuário:

```
CREATE USER 'novousuario'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';
```

```
CREATE USER 'foo'@'localhost' IDENTIFIED WITH mysql_native_password BY 'bar';
```

Liberar permissão para acessar a base de dados:

```
GRANT ALL PRIVILEGES ON * . * TO 'foo'@'localhost';
```

Uma vez finalizadas as permissões que você quer definir para os seus novos usuários, certifique-se sempre de recarregar todos os privilégios.

```
FLUSH PRIVILEGES;
```

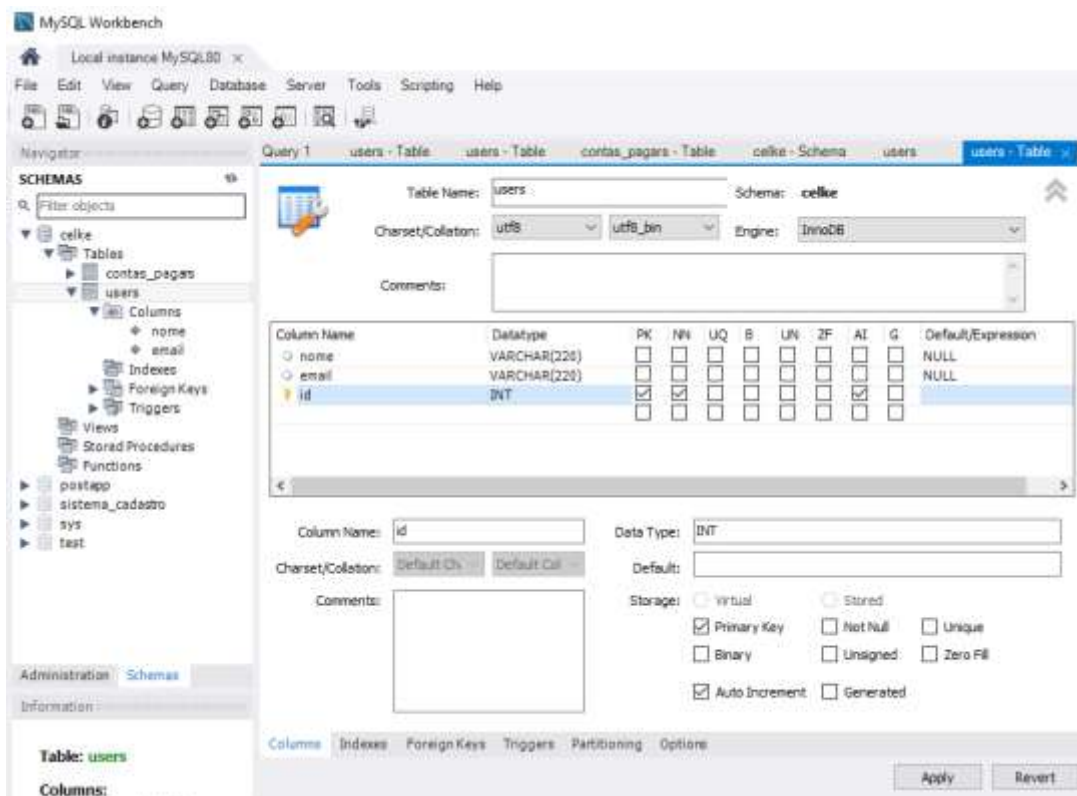
## Aula 08 - Como cadastrar com Nodejs no banco de dados

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

Pasta de C:\celke\nodejs

11/05/2021  17:14    <DIR>          .
11/05/2021  17:14    <DIR>          ..
10/05/2021  14:07    <DIR>          aula02
10/05/2021  14:29    <DIR>          aula03
10/05/2021  20:07    <DIR>          aula04-06
11/05/2021  17:13    <DIR>          aula07
11/05/2021  17:14    <DIR>          aula08
                0 arquivo(s)                0 bytes
                7 pasta(s) 118.062.518.272 bytes disponíveis
```

- Insira a coluna **id** (como chave primária e auto-increment) na tabela **users**:



- Clique no botão "Apply"



## Review SQL Script

### Review the SQL Script to be Applied on the Database

Online DDL

Algorithm:

Default

Lock Type:

Default

```
1 ALTER TABLE `celke`.`users`
2 ADD COLUMN `id` INT NOT NULL AUTO_INCREMENT AFTER `email`,
3 ADD PRIMARY KEY (`id`);
4 ;
5
```

[Back](#)

Apply

Cancel

- Clique no botão "Apply"
- Coloque o campo "id" para a primeira posição, e clique no botão "Apply":

## app.js

```
//Conexao com BD MySQL
const mysql = require('mysql');

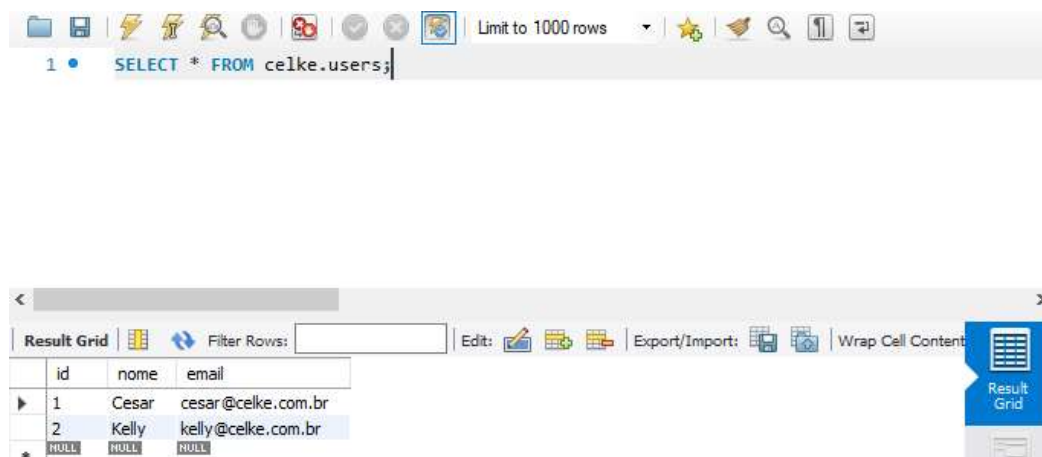
//A partir do MySQL 8 apresenta o erro ao utilizar o usuário root para conexão, necessário criar novo usuário
(instrução no Readme)
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'celkeone',
  password: '123456',
  database: 'celke'
});

connection.connect(function(err){
  if (err) console.error('Erro ao realizar a conexão com BD: ' + err.stack); return;
});

connection.query("INSERT INTO users(nome, email) VALUES ('Kelly', 'kelly@celke.com.br')",function(err, result){
  if(!err){
    console.log('Usuario cadastrado com sucesso!');
  }else{
    console.log('Erro ao cadastra o usuario!');
  }
});
```

node app.js

```
C:\celke\nodejs\aula08>node app.js
Usuario cadastrado com sucesso!
```



The screenshot shows a database client interface. At the top, the query bar contains the SQL statement: `SELECT * FROM celke.users;`. Below the query bar, a 'Result Grid' is displayed, showing the results of the query. The grid has four columns: 'id', 'nome', and 'email'. There are two rows of data: the first row has '1' for id, 'Cesar' for nome, and 'cesar@celke.com.br' for email; the second row has '2' for id, 'Kelly' for nome, and 'kelly@celke.com.br' for email. The grid also shows a row of 'NULL' values at the bottom, indicating the end of the result set.

	id	nome	email
1	1	Cesar	cesar@celke.com.br
2	2	Kelly	kelly@celke.com.br
*	NULL	NULL	NULL

## app.js

```
//Conexao com BD MySQL
const mysql = require('mysql');

//A partir do MySQL 8 apresenta o erro ao utilizar o usuário root para conexão, necessário criar novo usuário
(instrução no Readme)
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'celkeone',
  password: '123456',
  database: 'celke'
});

connection.connect(function(err){
  if (err) console.error('Erro ao realizar a conexão com BD: ' + err.stack); return;
});

//connection.query("INSERT INTO users(nome, email) VALUES ('Kelly', 'kelly@celke.com.br')",function(err, result){
//  if(!err){
//    console.log('Usuario cadastrado com sucesso!');
//  }else{
//    console.log('Erro ao cadastra o usuario!');
//  }
//});

connection.query("INSERT INTO users(nome, email) VALUES ('Jessica', 'jessica@celke.com.br')",function(err, result){
  if(!err){
    console.log('Usuario cadastrado com sucesso!');
  }else{
    console.log('Erro ao cadastra o usuario!');
  }
});
```

node app.js

```
C:\celke\nodejs\aula08>node app.js
Usuario cadastrado com sucesso!
```

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- celke
  - Tables
    - contas\_pagars
    - users
  - Columns
    - id
    - nome
    - email
  - Indexes
  - Foreign Keys
  - Triggers
- Views
- Stored Procedures
- Functions
- postapp
- sistema\_cadastro
- sys
- test

Table users - Table contas\_pagars - Table celke - Schema users users - Table users

Limit to 1000 rows

1 • `SELECT * FROM celke.users;`

Result Grid

id	nome	email
1	Cesar	cesar@celke.com.br
2	Kelly	kelly@celke.com.br
3	Jessica	jessica@celke.com.br
NULL	NULL	NULL

Filter Rows: Edit: Export/Import: Wrap Cell Content: Result Grid Form Editor

## Aula 09 - Como editar registro no banco de dados MySQL com Node

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

Pasta de C:\celke\nodejs

11/05/2021  17:23    <DIR>        .
11/05/2021  17:23    <DIR>        ..
10/05/2021  14:07    <DIR>        aula02
10/05/2021  14:29    <DIR>        aula03
10/05/2021  20:07    <DIR>        aula04-06
11/05/2021  17:13    <DIR>        aula07
11/05/2021  17:16    <DIR>        aula08
11/05/2021  17:23    <DIR>        aula09
               0 arquivo(s)              0 bytes
               8 pasta(s) 118.345.207.808 bytes disponíveis
```

### app.js

```
//Conexao com BD MySQL
const mysql = require('mysql');

//A partir do MySQL 8 apresenta o erro ao utilizar o usuário root para conexão, necessário criar novo usuário
(instrução no Readme)
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'celkeone',
  password: '123456',
  database: 'celke'
});

connection.connect(function(err){
  if (err) console.error('Erro ao realizar a conexão com BD: ' + err.stack); return;
});

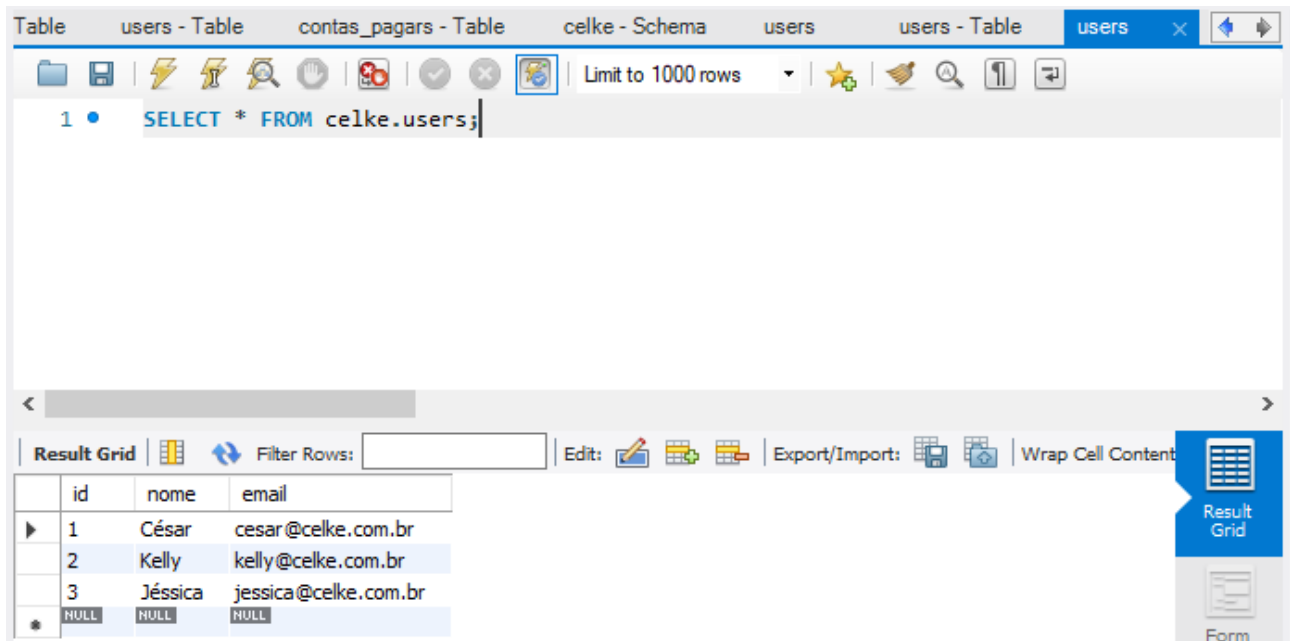
connection.query("UPDATE users SET nome = 'César' WHERE id = 1", function(err, result){
  if(!err){
    console.log('Usuario editado com sucesso!');
  }else{
    console.log('Erro: o usuario não foi editado com sucesso!');
  }
});

connection.query("UPDATE users SET nome = 'Jéssica' WHERE id = 3", function(err, result){
  if(!err){
    console.log('Usuario editado com sucesso!');
  }else{
    console.log('Erro: o usuario não foi editado com sucesso!');
  }
});
```



node app.js

```
C:\celke\nodejs\aula09>node app.js
Usuario editado com sucesso!
Usuario editado com sucesso!
```



The screenshot shows a database management interface with a query editor and a results grid. The query editor contains the SQL statement: `SELECT * FROM celke.users;`. The results grid displays the data from the `celke.users` table, which has three columns: `id`, `nome`, and `email`. The results are as follows:

	id	nome	email
1	1	César	cesar@celke.com.br
2	2	Kelly	kelly@celke.com.br
3	3	Jéssica	jessica@celke.com.br
*	NULL	NULL	NULL

The interface also includes a toolbar with various icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Filter Rows' input field. On the right side, there are buttons for 'Result Grid' and 'Form'.

## Aula 10 - Como apagar registro no banco de dados MySQL com Node

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

Pasta de C:\celke\nodejs

12/05/2021  06:40    <DIR>      .
12/05/2021  06:40    <DIR>      ..
10/05/2021  14:07    <DIR>      aula02
10/05/2021  14:29    <DIR>      aula03
10/05/2021  20:07    <DIR>      aula04-06
11/05/2021  17:13    <DIR>      aula07
11/05/2021  17:16    <DIR>      aula08
11/05/2021  17:24    <DIR>      aula09
12/05/2021  06:40    <DIR>      aula10
               0 arquivo(s)                0 bytes
               9 pasta(s) 118.332.407.808 bytes disponíveis
```

### app.js

```
//Conexao com BD MySQL
const mysql = require('mysql');
```

//A partir do MySQL 8 apresenta o erro ao utilizar o usuário root para conexão, necessário criar novo usuário (instrução no Readme)

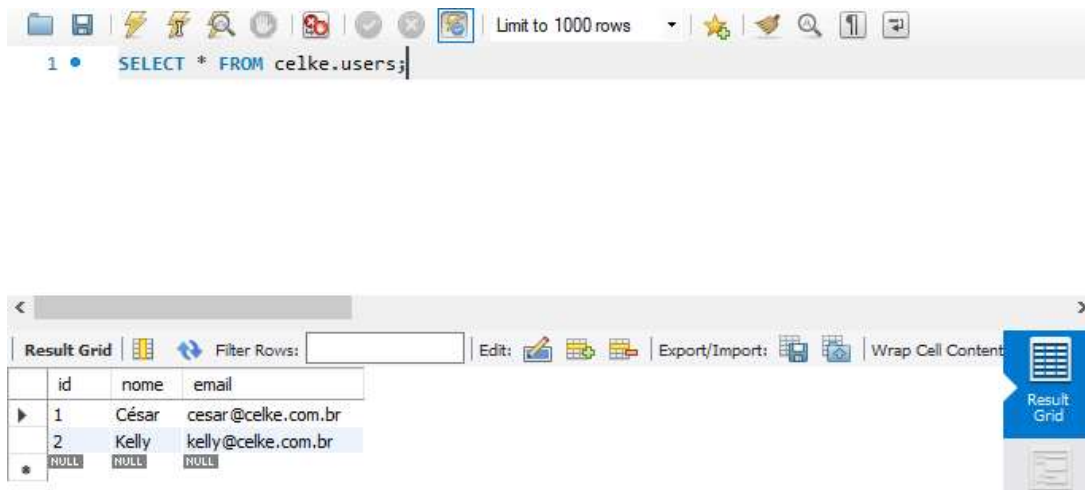
```
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'celkeone',
  password: '123456',
  database: 'celke'
});
```

```
connection.connect(function(err){
  if (err) console.error('Erro ao realizar a conexão com BD: ' + err.stack); return;
});
```

```
connection.query("DELETE FROM users WHERE id = 3", function(err, result){
  if(!err){
    console.log("Usuario apagado com sucesso!");
  }else{
    console.log("Erro: o usuario não foi apagado com sucesso!");
  }
});
```

node app.js

```
C:\celke\nodejs\aula10>node app.js
Usuario apagado com sucesso!
```



## Aula 11 - Como instalar e usar o Sequelize no Nodejs

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

Pasta de C:\celke\nodejs

12/05/2021  06:46    <DIR>        .
12/05/2021  06:46    <DIR>        ..
10/05/2021  14:07    <DIR>        aula02
10/05/2021  14:29    <DIR>        aula03
10/05/2021  20:07    <DIR>        aula04-06
11/05/2021  17:13    <DIR>        aula07
11/05/2021  17:16    <DIR>        aula08
11/05/2021  17:24    <DIR>        aula09
12/05/2021  06:40    <DIR>        aula10
12/05/2021  06:46    <DIR>        aula11
               0 arquivo(s)                0 bytes
              10 pasta(s) 118.328.057.856 bytes disponíveis
```

### Instalando o Sequelize

npm install sequelize --save

```
C:\celke\nodejs\aula11>npm install sequelize --save
npm WARN curso-nodejs@1.0.0 No repository field.

+ sequelize@6.6.2
added 19 packages from 80 contributors and audited 78 packages in 19.002s
found 0 vulnerabilities
```

npm install mysql2 --save

```
C:\celke\nodejs\aula11>npm install mysql2 --save
npm WARN curso-nodejs@1.0.0 No repository field.

+ mysql2@2.2.5
added 12 packages from 18 contributors and audited 90 packages in 7.193s
found 0 vulnerabilities
```

## Conectando a um BD

<https://sequelize.org/master/manual/getting-started>

### Connecting to a database

To connect to the database, you must create a Sequelize instance. This can be done by either passing the connection parameters separately to the Sequelize constructor or by passing a single connection URI:

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

// Option 1: Passing a connection URI
const sequelize = new Sequelize('sqlite::memory:') // Example for sqlite
const sequelize = new Sequelize('postgres://user:pass@example.com:5432/dbname') // Example for postgres

// Option 2: Passing parameters separately (sqlite)
const sequelize = new Sequelize({
  dialect: 'sqlite',
  storage: 'path/to/database.sqlite'
});

// Option 2: Passing parameters separately (other dialects)
const sequelize = new Sequelize('database', 'username', 'password', {
  host: 'localhost',
  dialect: /* one of 'mysql' | 'mariadb' | 'postgres' | 'mssql' */
});
```

- Vamos utilizar a opção 2.

## Testando a conexão

### Testing the connection

You can use the `.authenticate()` function to test if the connection is OK:

```
try {
  await sequelize.authenticate();
  console.log('Connection has been established successfully.');
```

```
} catch (error) {
  console.error('Unable to connect to the database:', error);
}
```

### app.js

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

```
// Passing parameters separately (other dialects)
const sequelize = new Sequelize('celke', 'celkeone', '123456', {
  host: 'localhost',
  dialect: 'mysql'
});
```

```
// Testando a conexão com o BD
```

```
sequelize.authenticate().then(function(){
  console.log("Conexão realizada com sucesso!");
}).catch(function(err){
  console.log("Erro ao realizar a conexão com o BD: " + err);
});
```

node app.js

```
C:\celke\nodejs\aula11>node app.js
Executing (default): SELECT 1+1 AS result
Conexão realizada com sucesso!
```

- Se for alterado o nome do BD para 'celke2'

```
C:\celke\nodejs\aula11>node app.js
Erro ao realizar a conexão com o BD: SequelizeConnectionError: Unknown database 'celke2'
```

## Criando uma tabela

### Using `sequelize.define`:

```
const { Sequelize, DataTypes } = require('sequelize');
const sequelize = new Sequelize('sqlite::memory:');

const User = sequelize.define('User', {
  // Model attributes are defined here
  firstName: {
    type: DataTypes.STRING,
    allowNull: false
  },
  lastName: {
    type: DataTypes.STRING
    // allowNull defaults to true
  }
}, {
  // Other model options go here
});

// `sequelize.define` also returns the model
console.log(User === sequelize.models.User); // true
```

- No MySQL Workbench apague todas as tabelas criadas;

### app.js

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

// Passing parameters separately (other dialects)
const sequelize = new Sequelize('celke', 'celkeone', '123456', {
  host: 'localhost',
  dialect: 'mysql'
});

// Testando a conexão com o BD

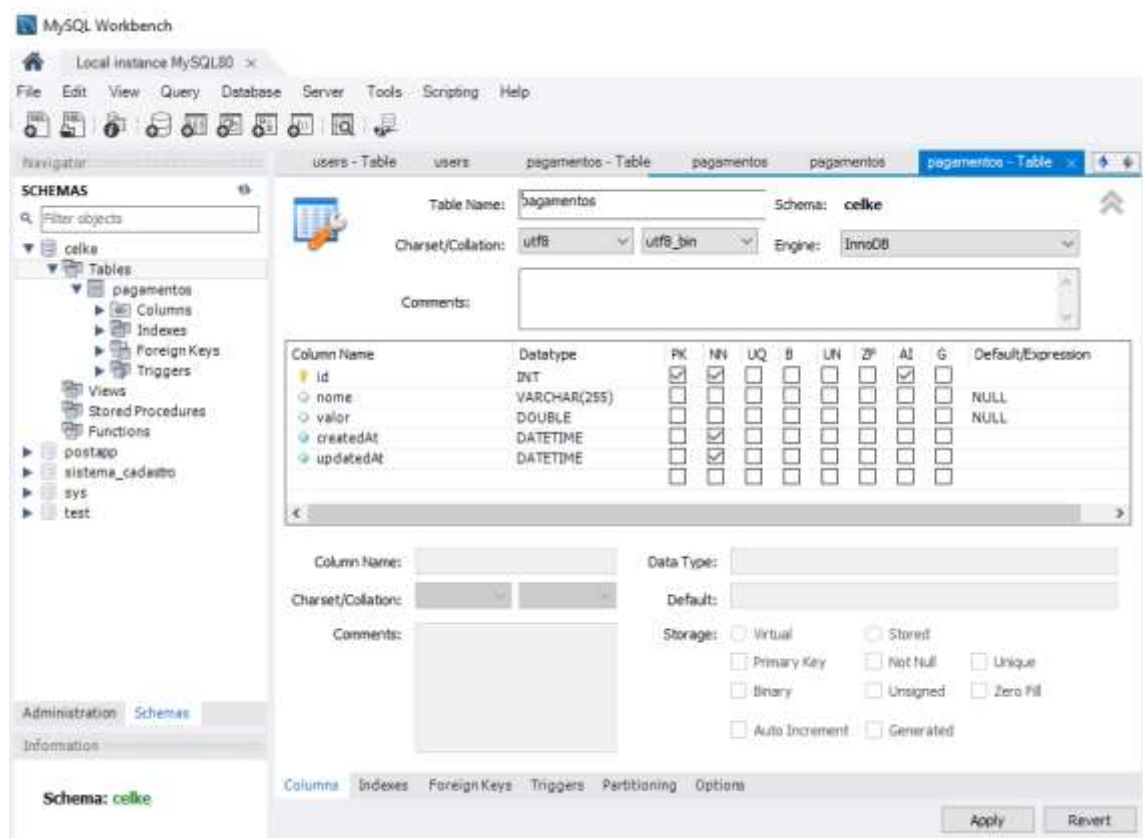
sequelize.authenticate().then(function(){
  console.log("Conexão realizada com sucesso!");
}).catch(function(err){
  console.log("Erro ao realizar a conexão com o BD: " + err);
});
```

```
const Pagamento = sequelize.define('pagamentos', {
  // Model attributes are defined here
  nome: {
    type: Sequelize.STRING
  },
  valor: {
    type: Sequelize.DOUBLE
  }
});
```

```
// Criar tabela com Sequelize
Pagamento.sync({force: true});
```

node app.js

```
C:\celke\nodejs\aula11>node app.js
Executing (default): SELECT 1+1 AS result
Executing (default): DROP TABLE IF EXISTS `pagamentos`;
Conexão realizada com sucesso!
Executing (default): CREATE TABLE IF NOT EXISTS `pagamentos` (`id` INTEGER NOT NULL auto_increment, `nome` VARCHAR(255), `valor` DOUBLE PRECISION, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB;
Executing (default): SHOW INDEX FROM `pagamentos`
```



## Inserindo registros

- Comente a última linha do código, para evitar que a tabela seja recriada toda vez que se executar o arquivo app.js

## app.js

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

// Passing parameters separately (other dialects)
const sequelize = new Sequelize('celke', 'celkeone', '123456', {
  host: 'localhost',
  dialect: 'mysql'
});

// Testando a conexão com o BD

sequelize.authenticate().then(function(){
  console.log("Conexão realizada com sucesso!");
}).catch(function(err){
  console.log("Erro ao realizar a conexão com o BD: " + err);
});

const Pagamento = sequelize.define('pagamentos', {
  // Model attributes are defined here
  nome: {
    type: Sequelize.STRING
  },
  valor: {
    type: Sequelize.DOUBLE
  }
});

// Criar tabela com Sequelize
// Pagamento.sync({force: true});

Pagamento.create({
  nome: "Energia",
  valor: 220
}).then(function(){
  console.log("Pagamento cadastrado com sucesso!");
}).catch(function(err){
  console.log("Erro ao cadastrar pagamento: " + err);
});
```

## node app.js

```
C:\celke\nodejs\aula11>node app.js
Executing (default): SELECT 1+1 AS result
Executing (default): INSERT INTO `pagamentos` (`id`,`nome`,`valor`,`createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?);
Conexão realizada com sucesso!
Pagamento cadastrado com sucesso!
```



Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content

	id	nome	valor	createdAt	updatedAt
▶	1	Energia	220	2021-05-11 19:10:51	2021-05-11 19:10:51
✱	NULL	NULL	NULL	NULL	NULL

## Aula 12 - Como instalar o Handlebars no Node e criar o layout padrão para o projeto

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

Pasta de C:\celke\nodejs

12/05/2021  07:12    <DIR>        .
12/05/2021  07:12    <DIR>        ..
10/05/2021  14:07    <DIR>        aula02
10/05/2021  14:29    <DIR>        aula03
10/05/2021  20:07    <DIR>        aula04-06
11/05/2021  17:13    <DIR>        aula07
11/05/2021  17:16    <DIR>        aula08
11/05/2021  17:24    <DIR>        aula09
12/05/2021  06:40    <DIR>        aula10
12/05/2021  06:51    <DIR>        aula11
12/05/2021  07:12    <DIR>        aula12
               0 arquivo(s)                0 bytes
              11 pasta(s) 118.315.233.280 bytes disponíveis
```

### Instalando o handlebars

<https://handlebarsjs.com/>

`npm install express-handlebars --save`

```
C:\celke\nodejs\aula12>npm install express-handlebars --save
npm WARN curso-nodejs@1.0.0 No repository field.

+ express-handlebars@5.3.2
added 18 packages from 43 contributors and audited 108 packages in 18.152s

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

found 0 vulnerabilities
```

## app.js

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

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

// Rotas

app.get('/pagamento', function(req, res){
  res.send("Página para listar os pagamentos");
});

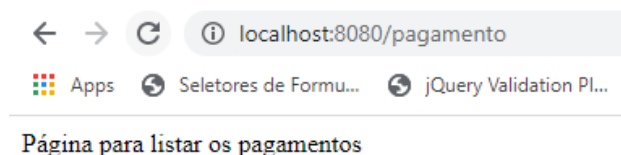
app.get('/add-pagamento', function(req, res){
  res.send("Formulário para cadastrar pagamento");
});

const PORT = 8080;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT)
```

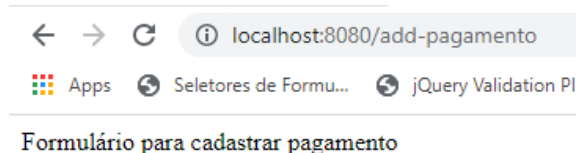
node app.js

```
C:\celke\nodejs\aula12>node app.js
Servidor rodando na porta 8080
█
```

[localhost:8080/pagamento](http://localhost:8080/pagamento)



[localhost:8080/add-pagamento](http://localhost:8080/add-pagamento)



## Usando handlebars

### views\layouts\main.handlebars

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

### views\pagamento.handlebars

```
<h1>Listar pagamentos</h1>
```

### views\add-pagamento.handlebars

```
<h1>Formulário para cadastrar pagamento</h1>
```

### app.js

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

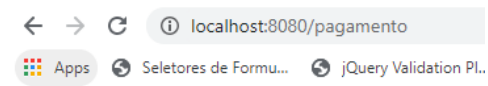
app.engine('handlebars', handlebars({defaultLayout: 'main'}));
app.set('view engine', 'handlebars');

// Rotas

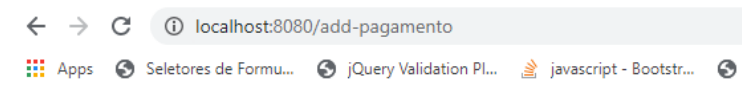
app.get('/pagamento', function(req, res){
  res.render('pagamento');
});

app.get('/add-pagamento', function(req, res){
  res.render('add-pagamento');
});

const PORT = 8080;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT)
```



## Listar pagamentos



## Formulário para cadastrar pagamento

## Aula 13 - Como criar o formulário com Node e salvar no banco de dados

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

Pasta de C:\celke\nodejs

12/05/2021  07:57    <DIR>        .
12/05/2021  07:57    <DIR>        ..
10/05/2021  14:07    <DIR>        aula02
10/05/2021  14:29    <DIR>        aula03
10/05/2021  20:07    <DIR>        aula04-06
11/05/2021  17:13    <DIR>        aula07
11/05/2021  17:16    <DIR>        aula08
11/05/2021  17:24    <DIR>        aula09
12/05/2021  06:40    <DIR>        aula10
12/05/2021  06:51    <DIR>        aula11
12/05/2021  07:24    <DIR>        aula12
12/05/2021  07:57    <DIR>        aula13
               0 arquivo(s)                0 bytes
               12 pasta(s) 118.287.560.704 bytes disponíveis
```

- Instale o pacote body-parser:

`npm install body-parser --save`

```
C:\celke\nodejs\aula13>npm install body-parser --save
npm WARN curso-nodejs@1.0.0 No repository field.

+ body-parser@1.19.0
updated 1 package and audited 109 packages in 15.403s

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

found 0 vulnerabilities
```

## app.js

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

// Configurações

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

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: false }))

// parse application/json
app.use(bodyParser.json())

// Rotas

app.get('/pagamento', function(req, res){
  // res.send("Página para listar os pagamentos");
  res.render('pagamento');
});

app.get('/cad-pagamento', function(req, res){
  res.render('cad-pagamento');
});

app.post('/add-pagamento', function(req, res){
  var nome = req.body.nome;
  var valor = req.body.valor;
  res.send("Nome: " + nome + "<br>Valor: " + valor + "<br>");
});

const PORT = 8080;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT)
```

## views\cad-pagamento.handlebars

```
<h1>Formulário para cadastrar pagamento</h1>

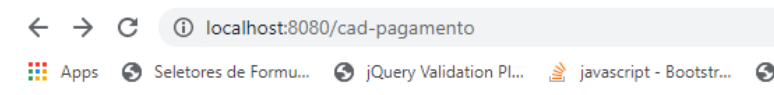
<form action="/add-pagamento" method="POST">
  <label for="nome">Nome</label>
  <input type="text" name="nome"><br><br>

  <label for="valor">Valor</label>
  <input type="text" name="valor"><br><br>

  <button type="submit">Cadastrar</button>
</form>
```

node app.js

localhost:8080/cad-pagamento

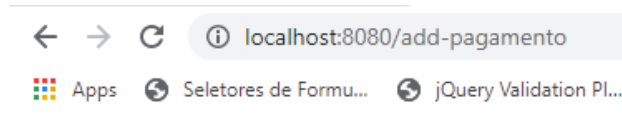


## Formulário para cadastrar pagamento

Nome

Valor

- Clique no botão "Cadastrar":



Nome: Condomínio  
Valor: 620

## Recriando a tabela "pagamentos"

### models\db.js

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

const sequelize = new Sequelize('celke', 'celkeone', '123456' {
  host: 'localhost',
  dialect: 'mysql'
});

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

### models\Pagamento.js

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

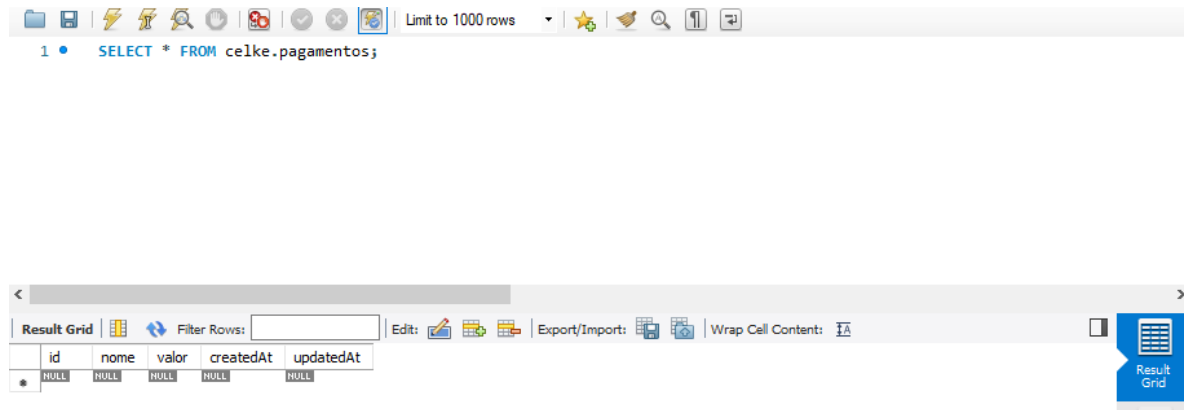
const Pagamento = db.sequelize.define('pagamentos', {
  nome: {
    type: db.Sequelize.STRING
  },
  valor: {
    type: db.Sequelize.DOUBLE
  }
});

// Criar a tabela
Pagamento.sync({force: true});
```



cd models  
node Pagamento.js

```
C:\celke\nodejs\aula13\models>node Pagamento.js
Executing (default): DROP TABLE IF EXISTS `pagamentos`;
Executing (default): CREATE TABLE IF NOT EXISTS `pagamentos` (`id` INTEGER NOT NULL auto_increment, `nome` VARCHAR(255), `valor` DOUBLE PRECISION, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB;
Executing (default): SHOW INDEX FROM `pagamentos`
```



- Comente a última linha para que a tabela não seja recriada quando o arquivo for novamente executado.

### models\Pagamento.js

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

const Pagamento = db.sequelize.define('pagamentos', {
  nome: {
    type: db.Sequelize.STRING
  },
  valor: {
    type: db.Sequelize.DOUBLE
  }
});
```

```
// Criar a tabela
// Pagamento.sync({force: true});
```

```
module.exports = Pagamento;
```

## app.js

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

// Configurações

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

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: false }))

// parse application/json
app.use(bodyParser.json())

// Rotas

app.get('/pagamento', function(req, res){
  // res.send("Página para listar os pagamentos");
  res.render('pagamento');
});

app.get('/cad-pagamento', function(req, res){
  // res.send("Formulário para cadastrar pagamento");
  res.render('cad-pagamento');
});

app.post('/add-pagamento', function(req, res){

  Pagamento.create({
    nome: req.body.nome,
    valor: req.body.valor
  }).then(function(){
    res.send('Pagamento cadastrado com sucesso!');
  }).catch(function(erro){
    res.send('Erro: Pagamento não foi cadastrado com sucesso! ' + erro);
  });

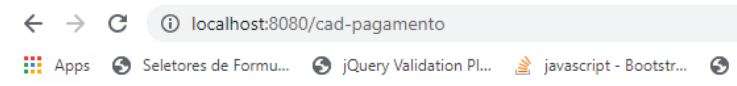
});

const PORT = 8080;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT)
```

## node app.js

```
C:\celke\nodejs\aula13>node app.js
Servidor rodando na porta 8080
█
```

localhost:8080/cad-pagamento

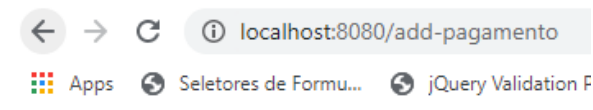


## Formulário para cadastrar pagamento

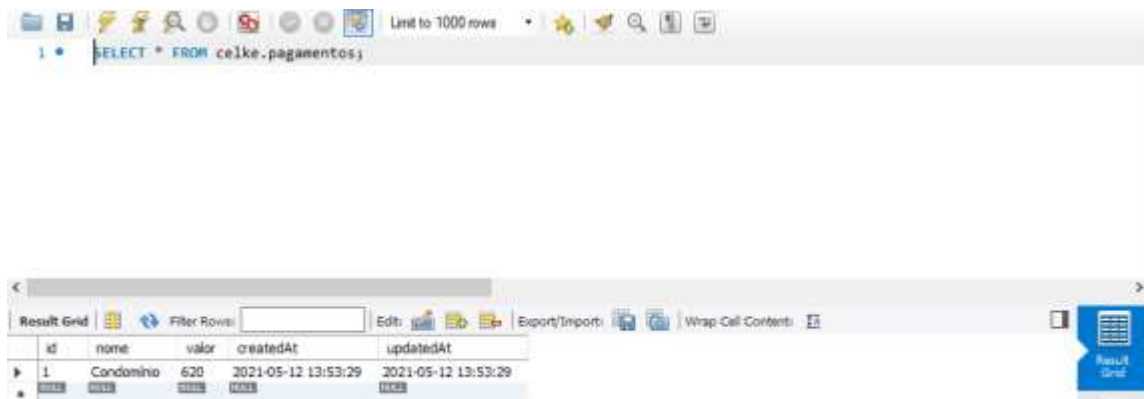
Nome

Valor

- Clique no botão "Cadastrar"



Pagamento cadastrado com sucesso!



## Redirecionando para a lista de pagamentos

### app.js

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

// Configurações

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

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: false }))

// parse application/json
app.use(bodyParser.json())

// Rotas

app.get('/pagamento', function(req, res){
  // res.send("Página para listar os pagamentos");
  res.render('pagamento');
});

app.get('/cad-pagamento', function(req, res){
  // res.send("Formulário para cadastrar pagamento");
  res.render('cad-pagamento');
});

app.post('/add-pagamento', function(req, res){

  // var nome = req.body.nome;
  // var valor = req.body.valor;

  // res.send("Nome: " + nome + "<br>Valor: " + valor + "<br>");

  Pagamento.create({
    nome: req.body.nome,
    valor: req.body.valor
  }).then(function(){
    // res.send('Pagamento cadastrado com sucesso!');
    res.redirect('/pagamento');
  }).catch(function(erro){
    res.send('Erro: Pagamento não foi cadastrado com sucesso! ' + erro);
  });

});

const PORT = 8080;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT)
```

node app.js

localhost:8080/cad-pagamento

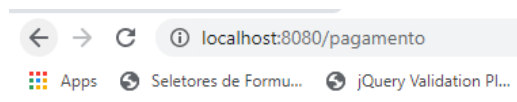


Formulário para cadastrar pagamento

Nome:

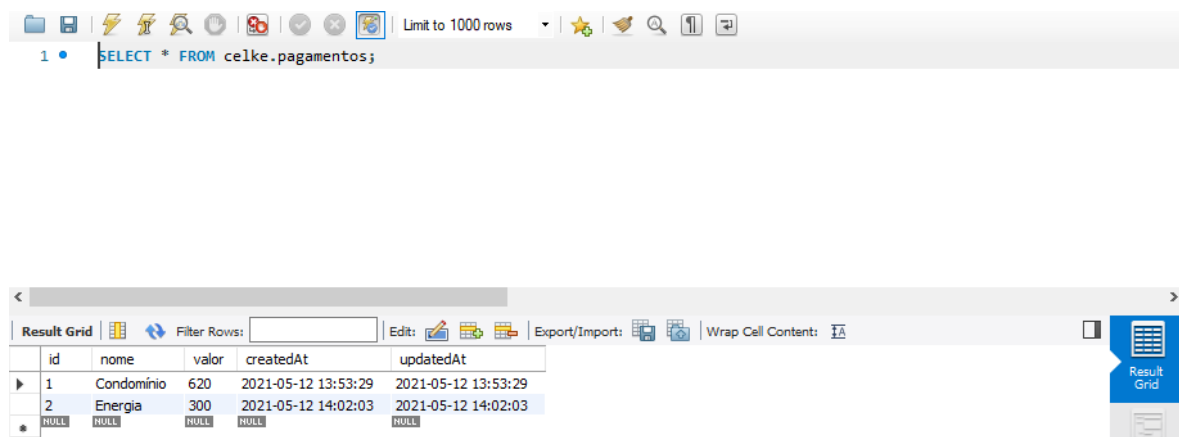
Valor:

- Clique no botão "Cadastrar"



localhost:8080/pagamento

## Listar pagamentos



1 • `SELECT * FROM celke.pagamentos;`

	id	nome	valor	createdAt	updatedAt
1	1	Condominio	620	2021-05-12 13:53:29	2021-05-12 13:53:29
2	2	Energia	300	2021-05-12 14:02:03	2021-05-12 14:02:03
*	NULL	NULL	NULL	NULL	NULL

## Aula 14 - Como listar registros do banco de dados com Nodejs

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

Pasta de C:\celke\nodejs

12/05/2021  11:09    <DIR>        .
12/05/2021  11:09    <DIR>        ..
10/05/2021  14:07    <DIR>        aula02
10/05/2021  14:29    <DIR>        aula03
10/05/2021  20:07    <DIR>        aula04-06
11/05/2021  17:13    <DIR>        aula07
11/05/2021  17:16    <DIR>        aula08
11/05/2021  17:24    <DIR>        aula09
12/05/2021  06:40    <DIR>        aula10
12/05/2021  06:51    <DIR>        aula11
12/05/2021  07:24    <DIR>        aula12
12/05/2021  09:55    <DIR>        aula13
12/05/2021  11:09    <DIR>        aula14
               0 arquivo(s)                0 bytes
               13 pasta(s)  117.451.726.848 bytes disponíveis
```

### findAll - Search for multiple elements in the database

```
// find multiple entries
Project.findAll().then(projects => {
  // projects will be an array of all Project instances
})

// search for specific attributes - hash usage
Project.findAll({ where: { name: 'A Project' } }).then(projects => {
  // projects will be an array of Project instances with the specified name
})

// search within a specific range
Project.findAll({ where: { id: [1,2,3] } }).then(projects => {
  // projects will be an array of Projects having the id 1, 2 or 3
  // this is actually doing an IN query
})
```

### app.js

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

// Configurações

app.engine('handlebars', handlebars({
  {
    defaultLayout: 'main',
    runtimeOptions: { allowProtoPropertiesByDefault: true, allowProtoMethodsByDefault: true }
  }
}))

app.set('view engine', 'handlebars');

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: false }));
```

```
// parse application/json
app.use(bodyParser.json())

// Rotas

app.get('/pagamento', function(req, res){
  Pagamento.findAll().then(function(pagamentos){
    res.render('pagamento', {pagamentos: pagamentos});
  });
});

app.get('/cad-pagamento', function(req, res){
  // res.send("Formulário para cadastrar pagamento");
  res.render('cad-pagamento');
});

app.post('/add-pagamento', function(req, res){

  // var nome = req.body.nome;
  // var valor = req.body.valor;

  // res.send("Nome: " + nome + "<br>Valor: " + valor + "<br>");

  Pagamento.create({
    nome: req.body.nome,
    valor: req.body.valor
  }).then(function(){
    // res.send('Pagamento cadastrado com sucesso!');
    res.redirect('/pagamento');
  }).catch(function(erro){
    res.send('Erro: Pagamento não foi cadastrado com sucesso! ' + erro);
  });

});

const PORT = 8080;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT)
```

[localhost:8080/pagamento](http://localhost:8080/pagamento)



← → ↻ ⓘ localhost:8080/pagamento

Apps Seletores de Formu... jQuery Validation Pl... javascript - Bootstr...

## Listar pagamentos

Nome: Condomínio  
 Valor: 620  
 Cadastrado: Wed May 12 2021 10:53:29 GMT-0300 (Horário Padrão de Brasília)

---

Nome: Energia  
 Valor: 300  
 Cadastrado: Wed May 12 2021 11:02:03 GMT-0300 (Horário Padrão de Brasília)

---

## Colocando a data em um formato mais adequado

`npm install moment --save`

```
C:\celke\nodejs\aula14>npm install moment --save
npm WARN curso-nodejs@1.0.0 No repository field.

+ moment@2.29.1
updated 1 package and audited 109 packages in 20.52s

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

found 0 vulnerabilities
```

### app.js

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

// Configurações

app.engine('handlebars', handlebars({
  {
    defaultLayout: 'main',
    runtimeOptions: {
      allowProtoPropertiesByDefault: true,
      allowProtoMethodsByDefault: true
    },
    helpers: {
      formatDate: (date) => {
        return moment(date).format('DD/MM/YYYY')
      }
    }
  }
}))

app.set('view engine', 'handlebars');

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: false }));

// parse application/json
app.use(bodyParser.json());

// Rotas

app.get('/pagamento', function(req, res){
  Pagamento.findAll().then(function(pagamentos){
    res.render('pagamento', {pagamentos: pagamentos});
  });
});
```



```

app.get('/cad-pagamento', function(req, res){
  // res.send("Formulário para cadastrar pagamento");
  res.render('cad-pagamento');
});

app.post('/add-pagamento', function(req, res){

  // var nome = req.body.nome;
  // var valor = req.body.valor;

  // res.send("Nome: " + nome + "<br>Valor: " + valor + "<br>");

  Pagamento.create({
    nome: req.body.nome,
    valor: req.body.valor
  }).then(function(){
    // res.send('Pagamento cadastrado com sucesso!');
    res.redirect('/pagamento');
  }).catch(function(erro){
    res.send('Erro: Pagamento não foi cadastrado com sucesso! ' + erro);
  });

});

const PORT = 8080;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT)

```

## views\pagamento.handlebars

```

<h1>Listar pagamentos</h1>

{{#each pagamentos}}
  Nome: {{nome}}<br>
  Valor: {{valor}}<br>
  Cadastrado: {{#formatDate createdAt}}{{/formatDate}}
  <hr>
{{/each}}

```

<http://localhost:8080/pagamento>



```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Servidor rodando na porta 8080
Executing (default): SELECT `id`, `nome`, `valor`, `createdAt`, `updatedAt` FROM `pagamentos` AS `pagamentos`;
█
```

## Ordenando por colunas

### Ordering eager loaded associations

When you want to apply `ORDER` clauses to eager loaded models, you must use the top-level `order` option with augmented arrays, starting with the specification of the nested model you want to sort.

This is better understood with examples.

```
Company.findAll({
  include: Division,
  order: [
    // We start the order array with the model we want to sort
    [Division, 'name', 'ASC']
  ]
});
Company.findAll({
  include: Division,
  order: [
    [Division, 'name', 'DESC']
  ]
});
```

### app.js

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

```
// Configurações
```

```
app.engine('handlebars', handlebars({
  {
    defaultLayout: 'main',
    runtimeOptions: {
      allowProtoPropertiesByDefault: true,
      allowProtoMethodsByDefault: true
    },
    helpers: {
      formatDate: (date) => {
        return moment(date).format('DD/MM/YYYY')
      }
    }
  })
})
```

```
app.set('view engine', 'handlebars');
```

```
// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: false }));
```

```
// parse application/json
app.use(bodyParser.json())

// Rotas

app.get('/pagamento', function(req, res){
  Pagamento.findAll({order: [['id', 'DESC']}).then(function(pagamentos){
    res.render('pagamento', {pagamentos: pagamentos});
  });
});

app.get('/cad-pagamento', function(req, res){
  // res.send("Formulário para cadastrar pagamento");
  res.render('cad-pagamento');
});

app.post('/add-pagamento', function(req, res){

  // var nome = req.body.nome;
  // var valor = req.body.valor;

  // res.send("Nome: " + nome + "<br>Valor: " + valor + "<br>");

  Pagamento.create({
    nome: req.body.nome,
    valor: req.body.valor
  }).then(function(){
    // res.send("Pagamento cadastrado com sucesso!");
    res.redirect('/pagamento');
  }).catch(function(erro){
    res.send('Erro: Pagamento não foi cadastrado com sucesso! ' + erro);
  });

});

const PORT = 8080;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT)
```

## views\pagamento.handlebars

```
<h1>Listar pagamentos</h1>
```

```
{{#each pagamentos}}
  Id: {{id}}<br>
  Nome: {{nome}}<br>
  Valor: {{valor}}<br>
  Cadastrado: {{#formatDate createdAt}}{{/formatDate}}
  <hr>
{{/each}}
```

```
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
Servidor rodando na porta 8080
Executing (default): SELECT `id`, `nome`, `valor`, `createdAt`, `updatedAt` FROM `pagamentos` AS `pagamentos` ORDER BY `pagamentos`.`id` DESC;
[]
```

## Listar pagamentos

Id: 2  
Nome: Energia  
Valor: 300  
Cadastrado: 12/05/2021

---

Id: 1  
Nome: Condomínio  
Valor: 620  
Cadastrado: 12/05/2021

---

## Aula 15 - Como apagar registro do banco de dados com Nodejs

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

Pasta de C:\celke\nodejs

12/05/2021  14:42    <DIR>        .
12/05/2021  14:42    <DIR>        ..
10/05/2021  14:07    <DIR>        aula02
10/05/2021  14:29    <DIR>        aula03
10/05/2021  20:07    <DIR>        aula04-06
11/05/2021  17:13    <DIR>        aula07
11/05/2021  17:16    <DIR>        aula08
11/05/2021  17:24    <DIR>        aula09
12/05/2021  06:40    <DIR>        aula10
12/05/2021  06:51    <DIR>        aula11
12/05/2021  07:24    <DIR>        aula12
12/05/2021  09:55    <DIR>        aula13
12/05/2021  14:04    <DIR>        aula14
12/05/2021  14:42    <DIR>        aula15
               0 arquivo(s)                0 bytes
              14 pasta(s)  117.039.988.736 bytes disponíveis
```

### views\pagamento.handlebars

<h1>Listar pagamentos</h1>

```
{{#each pagamentos}}
  Id: {{id}}<br>
  Nome: {{nome}}<br>
  Valor: {{valor}}<br>
  Cadastrado: {{#formatDate createdAt}}{{/formatDate}}<br>
  <a href="/del-pagamento/{{id}}"><button>Apagar</button></a>
<hr>
{{/each}}
```

### app.js

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

// Configurações

```
app.engine('handlebars', handlebars({
  {
    defaultLayout: 'main',
    runtimeOptions: {
      allowProtoPropertiesByDefault: true,
      allowProtoMethodsByDefault: true
    },
    helpers: {
      formatDate: (date) => {
        return moment(date).format('DD/MM/YYYY')
      }
    }
  }
}));
```

```

    }
  })
)

app.set('view engine', 'handlebars');

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: false }))

// parse application/json
app.use(bodyParser.json())

// Rotas

app.get('/pagamento', function(req, res){
  Pagamento.findAll({order: [['id', 'DESC']]}).then(function(pagamentos){
    res.render('pagamento', {pagamentos: pagamentos});
  });
});

app.get('/cad-pagamento', function(req, res){
  // res.send("Formulário para cadastrar pagamento");
  res.render('cad-pagamento');
});

app.post('/add-pagamento', function(req, res){

  Pagamento.create({
    nome: req.body.nome,
    valor: req.body.valor
  }).then(function(){
    // res.send("Pagamento cadastrado com sucesso!");
    res.redirect('/pagamento');
  }).catch(function(erro){
    res.send('Erro: Pagamento não foi cadastrado com sucesso! ' + erro);
  });
});

app.get('/del-pagamento/:id', function(req, res){
  Pagamento.destroy({
    where: {'id': req.params.id}
  }).then(function(){
    res.send("Pagamento apagado com sucesso!")
  }).catch(function(erro){
    res.send("Pagamento não foi apagado com sucesso! " + erro)
  });
});

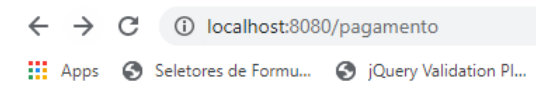
const PORT = 8080;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT)

```

Rodando o servidor:

`nodemon app.js`

<http://localhost:8080/pagamento>

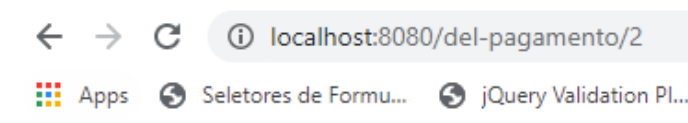


## Listar pagamentos

Id: 2  
Nome: Energia  
Valor: 300  
Cadastrado: 12/05/2021  
Apagar

Id: 1  
Nome: Condomínio  
Valor: 620  
Cadastrado: 12/05/2021  
Apagar

- Clique no botão "Apagar" no pagamento de Id: 2



Pagamento apagado com sucesso!

## Redirecionando após a deleção

- Adicione um novo pagamento:

[localhost:8080/cad-pagamento](http://localhost:8080/cad-pagamento)



## Formulário para cadastrar pagamento

Nome   
Valor   
Cadastrar

## Listar pagamentos

Id: 3  
Nome: Energia  
Valor: 220  
Cadastrado: 12/05/2021  
Apagar

Id: 1  
Nome: Condomínio  
Valor: 620  
Cadastrado: 12/05/2021  
Apagar

### app.js

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

// Configurações

app.engine('handlebars', handlebars({
  {
    defaultLayout: 'main',
    runtimeOptions: {
      allowProtoPropertiesByDefault: true,
      allowProtoMethodsByDefault: true
    },
    helpers: {
      formatDate: (date) => {
        return moment(date).format('DD/MM/YYYY')
      }
    }
  }
}))

app.set('view engine', 'handlebars');

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: false }))

// parse application/json
app.use(bodyParser.json())

// Rotas

app.get('/pagamento', function(req, res){
  Pagamento.findAll({order: [['id', 'DESC']]}).then(function(pagamentos){
    res.render('pagamento', {pagamentos: pagamentos});
  });
});
```



```

app.get('/cad-pagamento', function(req, res){
  // res.send("Formulário para cadastrar pagamento");
  res.render('cad-pagamento');
});

app.post('/add-pagamento', function(req, res){

  // var nome = req.body.nome;
  // var valor = req.body.valor;

  // res.send("Nome: " + nome + "<br>Valor: " + valor + "<br>");

  Pagamento.create({
    nome: req.body.nome,
    valor: req.body.valor
  }).then(function(){
    // res.send("Pagamento cadastrado com sucesso!");
    res.redirect('/pagamento');
  }).catch(function(erro){
    res.send('Erro: Pagamento não foi cadastrado com sucesso! ' + erro);
  });

});

app.get('/del-pagamento/:id', function(req, res){
  Pagamento.destroy({
    where: {id: req.params.id}
  }).then(function(){
    // res.send("Pagamento apagado com sucesso!")
    res.redirect('/pagamento');
  }).catch(function(erro){
    res.send("Pagamento não foi apagado com sucesso! " + erro)
  });
});

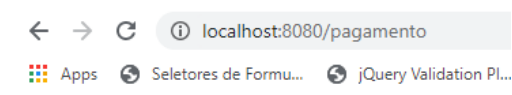
const PORT = 8080;
app.listen(PORT);
console.log("Servidor rodando na porta " + PORT)

```

Rodando o servidor:

`nodemon app.js`

- Clique no botão "Apagar" do pagamento de **Id:3**



## Listar pagamentos

Id: 1  
 Nome: Condomínio  
 Valor: 620  
 Cadastrado: 12/05/2021

## Aula 16 - Como criar API com Nodejs - Parte 1

- Dentro de `C:\celke\nodejs` crie uma pasta chamada `tutorial_api`:

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

Pasta de C:\celke\nodejs

12/05/2021  15:22    <DIR>        .
12/05/2021  15:22    <DIR>        ..
10/05/2021  14:07    <DIR>        aula02
10/05/2021  14:29    <DIR>        aula03
10/05/2021  20:07    <DIR>        aula04-06
11/05/2021  17:13    <DIR>        aula07
11/05/2021  17:16    <DIR>        aula08
11/05/2021  17:24    <DIR>        aula09
12/05/2021  06:40    <DIR>        aula10
12/05/2021  06:51    <DIR>        aula11
12/05/2021  07:24    <DIR>        aula12
12/05/2021  09:55    <DIR>        aula13
12/05/2021  14:04    <DIR>        aula14
12/05/2021  14:44    <DIR>        aula15
12/05/2021  15:22    <DIR>        tutorial_api
                0 arquivo(s)                0 bytes
                15 pasta(s)  117.011.857.408 bytes disponíveis
```



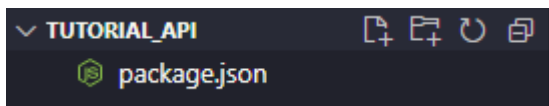
### Instalando as dependências

### Criando o arquivo package.json

`npm init`

About to write to C:\celke\nodejs\tutorial\_api\package.json:

```
{
  "name": "tutorial_api",
  "version": "1.0.0",
  "description": "Tutorial API",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "César Szpak",
  "license": "ISC"
}
```



## package.json

```
{
  "name": "tutorial_api",
  "version": "1.0.0",
  "description": "Tutorial API",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "César Szpak",
  "license": "ISC"
}
```

## README.md

### ----- SEQUENCIA PARA CRIAR O PROJETO -----

Criar o arquivo package  
npm init

//Gerencia as requisições, rotas e URLs, entre outra funcionalidades  
npm install express

//Instalar o módulo para reiniciar o servidor sempre que houver alteração no código fonte  
npm install -D nodemon

//Rodar o projeto usando o nodemon  
nodemon app.js

### ----- COMO RODAR O PROJETO BAIXADO -----

//Instalar todas as dependencias indicada pelo package.json  
npm install

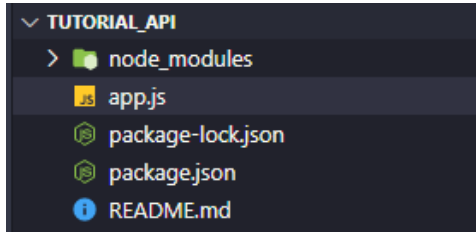
//Rodar o projeto usando o nodemon  
nodemon app.js

## Instalando o express

npm install express --save

```
C:\celke\nodejs\tutorial_api>npm install express --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN tutorial_api@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 20.24s
found 0 vulnerabilities
```



## app.js

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

```
app.use(express.json());
```

```
// Criando rotas
```

```
app.get('/', (req, res) => {
  // res.send("introdução a API");
  return res.json({titulo: "Como criar API"});
});
```

```
const PORT = 8080;
```

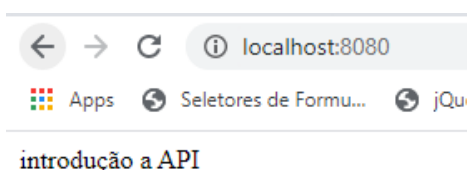
```
app.listen(PORT, ()=>{
  console.log("Servidor iniciado na porta " + PORT);
})
```

Rodando o servidor:

```
node app.js
```

```
C:\celke\nodejs\tutorial_api>node app.js
Servidor iniciado na porta 8080
█
```

localhost:8080

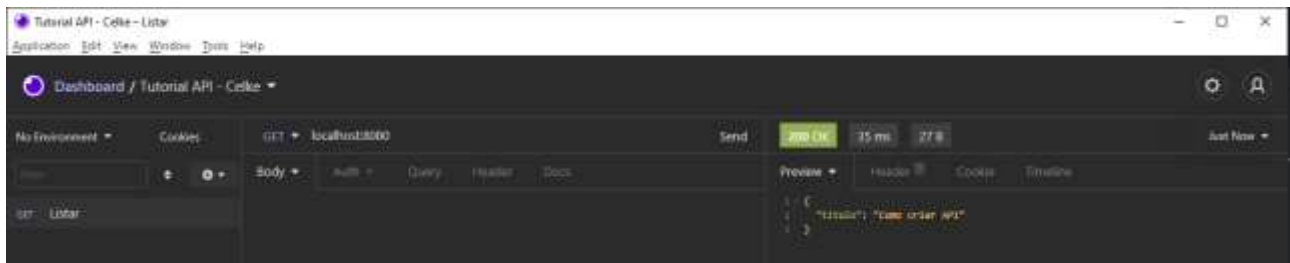


- Baixe e instale o **Insomnia** (para simulação de requisições)
- Crie uma nova coleção de requisições:



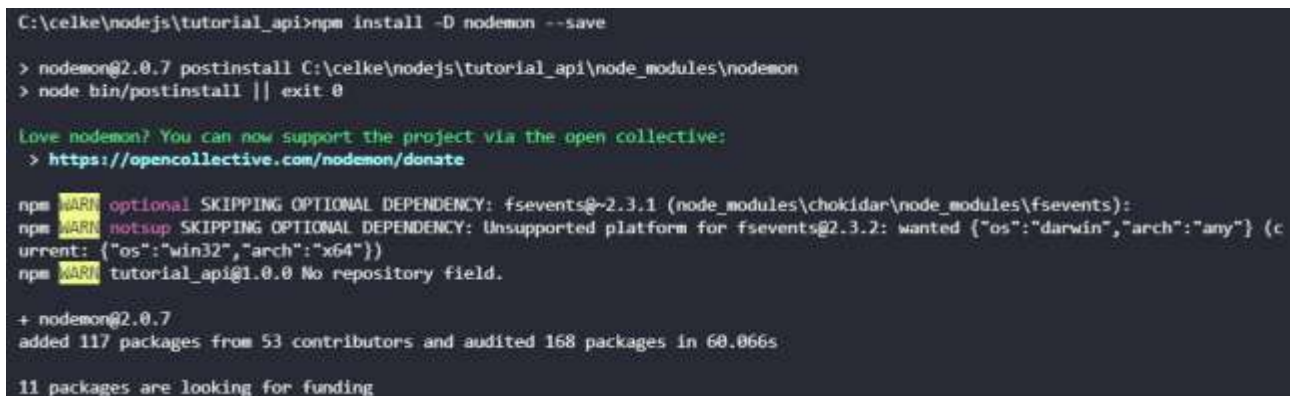
- Crie uma requisição chamada "**Listar**"

GET - localhost:8080



## Instalando o nodemon

`npm install -D nodemon --save`



## package.json

```
{
  "name": "tutorial_api",
  "version": "1.0.0",
  "description": "Tutorial API",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
}
```

```
"author": "César Szpak",  
"license": "ISC",  
"dependencies": {  
  "express": "^4.17.1"  
},  
"devDependencies": {  
  "nodemon": "^2.0.7"  
}  
}
```

nodemon app.js

```
C:\celke\nodejs\tutorial_api>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`  
Servidor iniciado na porta 8080  
█
```

## Aula 17 - Como criar o cadastrar na API com Nodejs - Parte 2

### README.md

---

#### SEQUENCIA PARA CRIAR O PROJETO

---

Criar o arquivo package  
npm init

//Gerencia as requisições, rotas e URLs, entre outra funcionalidades  
npm install express

//Instalar o módulo para reiniciar o servidor sempre que houver alteração no código fonte  
npm install -D nodemon

//Rodar o projeto usando o nodemon  
nodemon app.js

//Instalar o MongoDB  
npm install --save mongodb

//Instalar o Mongoose - Mongoose traduz os dados do banco de dados para objetos JavaScript para que possam ser utilizados por sua aplicação.  
npm install --save mongoose

---

#### COMO RODAR O PROJETO BAIXADO

---

//Instalar todas as dependencias indicada pelo package.json  
npm install

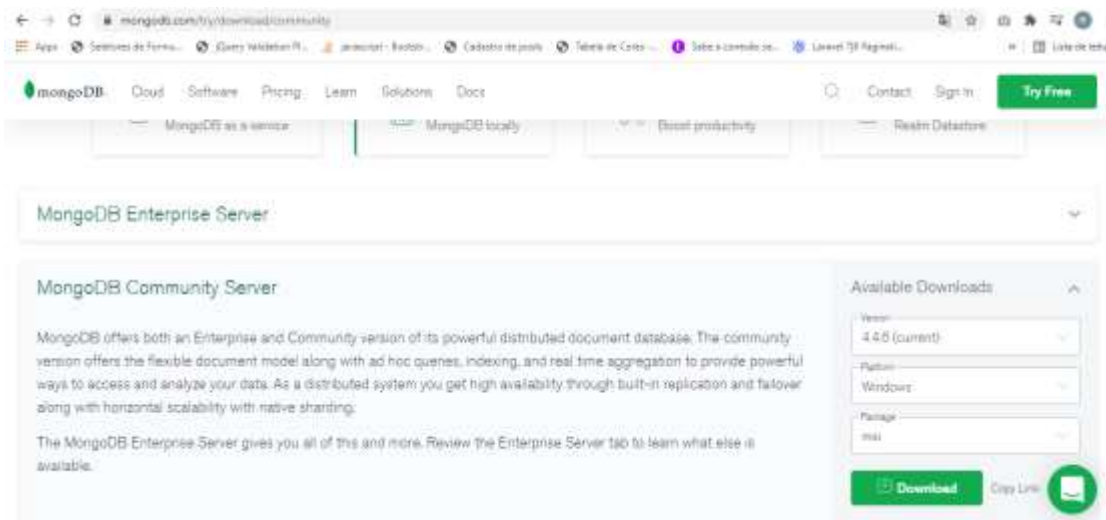
//Rodar o projeto usando o nodemon  
nodemon app.js

## Instalando o banco de dados não relacional MongoDB

### Instalando o MongoDB no computador

- Baixe e instale o MongoDB

<https://www.mongodb.com/try/download/community>



- Ao instalar, deixe a caixa marcada para instalar também o **MongoDB Compass**

### Instalando o MongoDB no projeto

- Instale a dependência no projeto:

`npm install --save mongodb`

```
C:\celke\nodejs\tutorial_api>npm install --save mongodb
npm WARN tutorial_api@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (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"})

+ mongodb@3.6.6
added 14 packages from 9 contributors and audited 182 packages in 16.796s

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

found 0 vulnerabilities
```



## Instalando o Mongoose

Mongoose traduz os dados do banco de dados para objetos JavaScript para que possam ser utilizados por sua aplicação.

`npm install --save mongoose`

```
C:\celke\nodejs\tutorial_api>npm install --save mongoose
npm WARN tutorial_api@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (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"})

+ mongoose@5.12.8
added 15 packages from 84 contributors and audited 197 packages in 51.676s

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

found 0 vulnerabilities
```

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

```
await mongoose.connect('mongodb://localhost/my_database', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
  useFindAndModify: false,
  useCreateIndex: true
});
```

### app.js

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

```
// Configuração
```

```
const app = express();
app.use(express.json());
```

```
mongoose.connect('mongodb://localhost/celke', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log("Conexão com mongoDB realizada com sucesso!");
}).catch((erro) => {
  console.log("Erro: Conexão com mongoDB não foi realizada com sucesso! " + erro);
});
```

```
// Criando rotas
```

```
app.get('/', (req, res) => {
  // res.send("introdução a API");
  return res.json({titulo: "Como criar API"});
});
```

```
const PORT = 8080;
```

```
app.listen(PORT, ()=>{
  console.log("Servidor iniciado na porta " + PORT);
})
```

nodemon app.js

```
C:\celke\nodejs\tutorial_api>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`
Servidor iniciado na porta 8080
Conexão com mongoDB realizada com sucesso!
```

- Para testar erro, altere "localhost" para "localhost2" e salve:

```
C:\celke\nodejs\tutorial_api>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`
Servidor iniciado na porta 8080
Erro: Conexão com mongoDB não foi realizada com sucesso! MongooseServerSelectionError: getaddrinfo ENOTFOUND localhost2
```

models\Artigo.js

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

```
const Artigo = new mongoose.Schema({
  titulo: {
    type: String,
    required: true
  },
  conteudo: {
    type: String,
    required: true
  }
},
{
  timestamps: true
});
```

```
mongoose.model('artigo', Artigo);
```

## app.js

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

require('./models/Artigo');
const Artigo = mongoose.model('artigo');

// Configuração

const app = express();
app.use(express.json());

mongoose.connect('mongodb://localhost/celke', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log("Conexão com mongoDB realizada com sucesso!");
}).catch((erro) => {
  console.log("Erro: Conexão com mongoDB não foi realizada com sucesso! " + erro);
});

// Criando rotas

app.get('/', (req, res) => {
  // res.send("introdução a API");
  return res.json({titulo: "Como criar API"});
});

app.post('/artigo', (req, res) => {
  const artigo = Artigo.create(req.body, (err) => {
    if(err) return res.status(400).json({
      error: true,
      message: "Error: Artigo não foi cadastrado com sucesso!"
    })

    return res.status(200).json({
      error: false,
      message: "Artigo cadastrado com sucesso!"
    })
  })
});

const PORT = 8080;

app.listen(PORT, ()=>{
  console.log("Servidor iniciado na porta " + PORT);
})
```

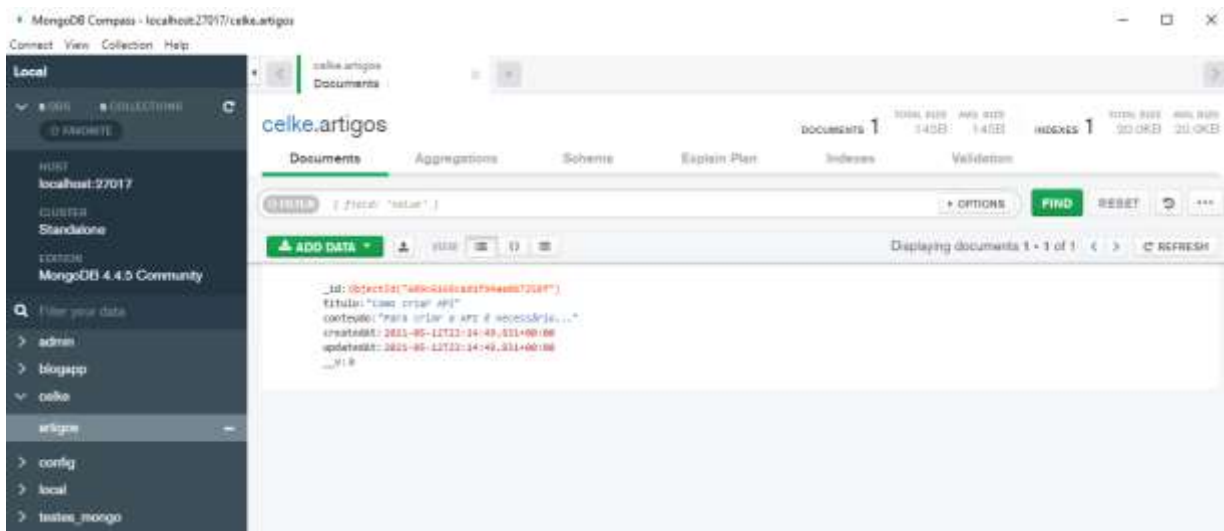
## nodemon app.js

```
C:\celke\nodejs\tutorial_api>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`
Servidor iniciado na porta 8080
Conexão com mongoDB realizada com sucesso!
```

- No insomnia:



- No MongoDB Compass:



- No terminal:

mongod

```
C:\celke\nodejs\tutorial_api>mongod
{"t":{"$date":"2021-05-12T20:18:10.320-03:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"main","msg":"Automatically
disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2021-05-12T20:18:13.221-03:00"},"s":"W", "c":"ASIO", "id":22601, "ctx":"main","msg":"No Transport
layer configured during NetworkInterface startup"}
{"t":{"$date":"2021-05-12T20:18:13.222-03:00"},"s":"I", "c":"NETWORK", "id":4648602, "ctx":"main","msg":"Implicit TCP
fastopen in use."}
{"t":{"$date":"2021-05-12T20:18:13.226-03:00"},"s":"I", "c":"STORAGE", "id":4615611, "ctx":"initandlisten","msg":"Mon
godb starting","attr":{"pid":14292,"port":27017,"dbPath":"C:/data/db/","architecture":"64-bit","host":"DESKTOP-B5HCHH0"}
{"t":{"$date":"2021-05-12T20:18:13.226-03:00"},"s":"I", "c":"CONTROL", "id":23398, "ctx":"initandlisten","msg":"Targ
et operating system minimum version","attr":{"targetMinOS":"Windows 7/Windows Server 2008 R2"}}
{"t":{"$date":"2021-05-12T20:18:13.226-03:00"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten","msg":"Bui
ld Info","attr":{"buildInfo":{"version":"4.4.5","gitVersion":"ff5cb77101b052fa02da43b8538093488cf0b3f7","modules":[],"al
locator":"tcmalloc","environment":{"distmod":"windows","distarch":"x86_64","target_arch":"x86_64"}}}}
{"t":{"$date":"2021-05-12T20:18:13.226-03:00"},"s":"I", "c":"CONTROL", "id":51765, "ctx":"initandlisten","msg":"Ope
rating System","attr":{"os":{"name":"Microsoft Windows 10","version":"10.0 (build 19042)}}}
{"t":{"$date":"2021-05-12T20:18:13.226-03:00"},"s":"I", "c":"CONTROL", "id":21951, "ctx":"initandlisten","msg":"Opt
ions set by command line","attr":{"options":{"}}}
{"t":{"$date":"2021-05-12T20:18:13.335-03:00"},"s":"W", "c":"STORAGE", "id":22271, "ctx":"initandlisten","msg":"Det
ected unclean shutdown - lock file is not empty","attr":{"lockFile":"C:\\data\\ldb\\mongod.lock"}}
{"t":{"$date":"2021-05-12T20:18:13.374-03:00"},"s":"I", "c":"STORAGE", "id":22270, "ctx":"initandlisten","msg":"Sto
rage engine to use detected by data files","attr":{"dbpath":"C:/data/db/","storageEngine":"wiredtiger"}}
{"t":{"$date":"2021-05-12T20:18:13.375-03:00"},"s":"W", "c":"STORAGE", "id":22302, "ctx":"initandlisten","msg":"Rec
eiving data from the last clean checkpoint."}
{"t":{"$date":"2021-05-12T20:18:13.377-03:00"},"s":"I", "c":"STORAGE", "id":22315, "ctx":"initandlisten","msg":"Ope
ning WiredTiger","attr":{"config":{"create,cache_size=1444M,session_max=33000,eviction=(threads_min=4,threads_max=4),conf
g_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle
time=180000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=(recovery_progress,checkpoi
```

## mongo

```
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("706fd835-073c-46af-ad40-3f06ed303e10") }
MongoDB server version: 4.4.5
---
The server generated these startup warnings when booting:
  2021-05-10T21:01:53.713-03:00: 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
blogapp              0.000GB
celke                 0.000GB
config               0.000GB
local                0.000GB
testes_mongo         0.000GB
```

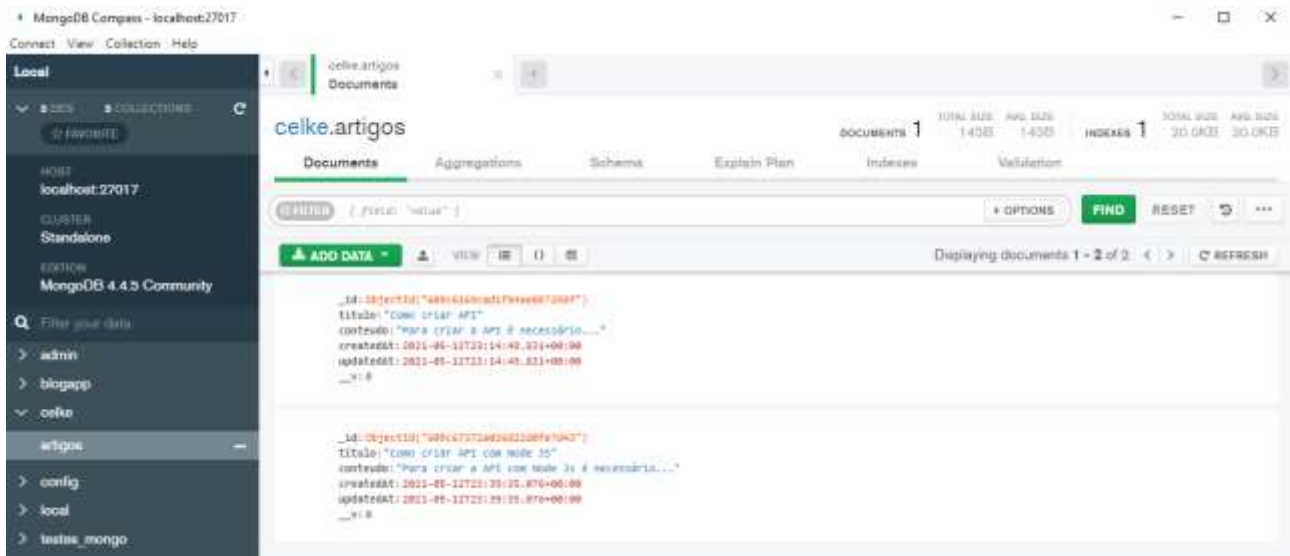
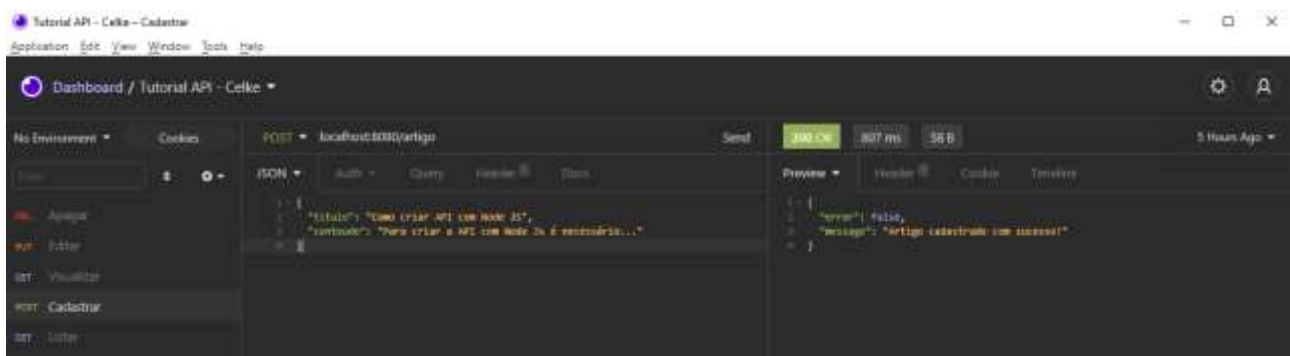
```
> use celke
switched to db celke
> show collections;
artigos
> db.artigos.find();
{ "_id" : ObjectId("609c6169cad1f94ae087258f"), "titulo" : "Como criar API", "conteudo" : "Para criar a API é necessário
...", "createdAt" : ISODate("2021-05-12T23:14:49.831Z"), "updatedAt" : ISODate("2021-05-12T23:14:49.831Z"), "__v" : 0 }
>
```

## Aula 18 - Como criar o listar na API com Nodejs - Parte 3

### Cadastrando um novo artigo

nodemon app.js

```
C:\celke\nodejs\tutorial_api>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`
Servidor iniciado na porta 8080
Conexão com mongoDB realizada com sucesso!
```



```
> db.artigos.find();
{ "_id" : ObjectId("609c6169cad1f94ae087258f"), "titulo" : "Como criar API", "conteudo" : "Para criar a API é necessário...", "createdAt" : ISODate("2021-05-12T23:14:49.831Z"), "updatedAt" : ISODate("2021-05-12T23:14:49.831Z"), "__v" : 0 }
{ "_id" : ObjectId("609c67372a026d22d0fa7d43"), "titulo" : "Como criar API com Node JS", "conteudo" : "Para criar a API com Node Js é necessário...", "createdAt" : ISODate("2021-05-12T23:39:35.076Z"), "updatedAt" : ISODate("2021-05-12T23:39:35.076Z"), "__v" : 0 }
>
```

## Listando os artigos

### app.js

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

require('./models/Artigo');
const Artigo = mongoose.model('artigo');

// Configuração

const app = express();
app.use(express.json());

mongoose.connect('mongodb://localhost/celke', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log("Conexão com mongoDB realizada com sucesso!");
}).catch((erro) => {
  console.log("Erro: Conexão com mongoDB não foi realizada com sucesso! " + erro);
});

// Criando rotas

app.get('/', (req, res) => {
  Artigo.find({}).then((artigo) =>{
    return res.json(artigo);
  }).catch((erro) => {
    return res.status(400).json({
      error: true,
      message: "Nenhum artigo encontrado!"
    });
  });
});

app.post('/artigo', (req, res) => {
  const artigo = Artigo.create(req.body, (err) => {
    if(err) return res.status(400).json({
      error: true,
      message: "Error: Artigo não foi cadastrado com sucesso!"
    })

    return res.status(200).json({
      error: false,
      message: "Artigo cadastrado com sucesso!"
    })
  })
});

const PORT = 8080;

app.listen(PORT, ()=>{
  console.log("Servidor iniciado na porta " + PORT);
})
```

Tutorial API - Celke - Listar

ApplicationFileViewWindowToolsHelp

Dashboard / Tutorial API - Celke

No Environment Cookies


GETlocalhost:8080

Send200 OK297 ms415 BJust Now

BodyAuthQueryHeaderDocs

PreviewHeaderCookiesTimeline

```
1 {
2   {
3     "_id": "666666666666666666666666",
4     "titulo": "Como criar API",
5     "descricao": "Para criar a api é necessário...",
6     "createdAt": "2023-05-12T12:14:48.831Z",
7     "updatedAt": "2023-05-12T12:14:48.831Z",
8     "__v": 0
9   },
10  {
11    "_id": "666666666666666666666666",
12    "titulo": "Como criar API com Node JS",
13    "descricao": "Para criar a API com Node JS é necessário...",
14    "createdAt": "2023-05-12T12:14:48.831Z",
15    "updatedAt": "2023-05-12T12:14:48.831Z",
16    "__v": 0
17  }
18 }
```



Select a body type from above



## Aula 19 - Como visualizar dados de um id com Nodejs na API - Parte 4

### app.js

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

require('./models/Artigo');
const Artigo = mongoose.model('artigo');

// Configuração

const app = express();
app.use(express.json());

mongoose.connect('mongodb://localhost/celke', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log("Conexão com mongoDB realizada com sucesso!");
}).catch((erro) => {
  console.log("Erro: Conexão com mongoDB não foi realizada com sucesso! " + erro);
});

// Criando rotas

app.get('/', (req, res) => {
  // return res.json({titulo: "Como criar API"});

  Artigo.find({}).then((artigo) => {
    return res.json(artigo);
  }).catch((erro) => {
    return res.status(400).json({
      error: true,
      message: "Nenhum artigo encontrado!"
    });
  });
});

app.get("/artigo/:id", (req, res) => {
  Artigo.findOne({
    _id: req.params.id
  }).then((artigo) => {
    return res.json(artigo);
  }).catch((erro) => {
    return res.status(400).json({
      error: true,
      message: "Nenhum artigo encontrado!"
    });
  });
});
});
```

```

app.post('/artigo', (req, res) => {
  const artigo = Artigo.create(req.body, (err) => {
    if(err) return res.status(400).json({
      error: true,
      message: "Error: Artigo não foi cadastrado com sucesso!"
    })

    return res.status(200).json({
      error: false,
      message: "Artigo cadastrado com sucesso!"
    })
  })
});

const PORT = 8080;

app.listen(PORT, ()=>{
  console.log("Servidor iniciado na porta " + PORT);
})

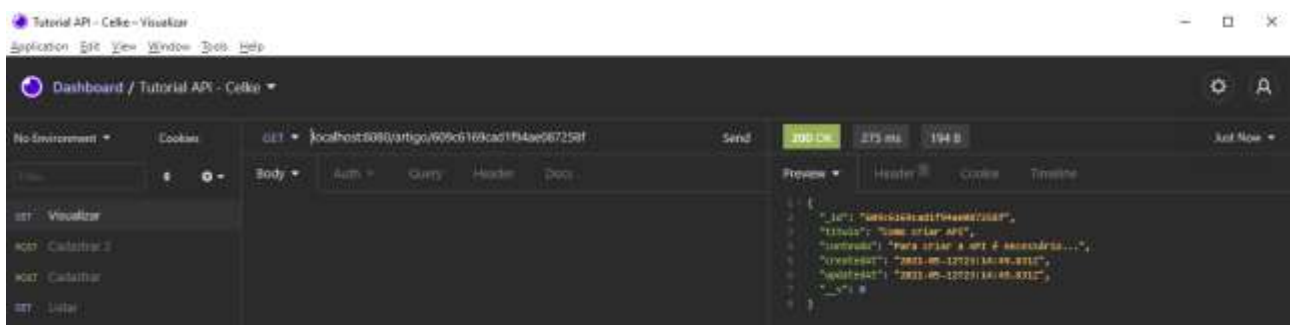
```

nodemon app.js

```

C:\celke\nodejs\tutorial_api>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`
Servidor iniciado na porta 8080
Conexão com mongoDB realizada com sucesso!

```



## Aula 20 - Como criar editar com Nodejs na API - Parte 5

### app.js

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

require('./models/Artigo');
const Artigo = mongoose.model('artigo');

// Configuração

const app = express();
app.use(express.json());

mongoose.connect('mongodb://localhost/celke', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log("Conexão com mongoDB realizada com sucesso!");
}).catch((erro) => {
  console.log("Erro: Conexão com mongoDB não foi realizada com sucesso! " + erro);
});

// Criando rotas

app.get('/', (req, res) => {
  // return res.json({titulo: "Como criar API"});

  Artigo.find({}).then((artigo) => {
    return res.json(artigo);
  }).catch((erro) => {
    return res.status(400).json({
      error: true,
      message: "Nenhum artigo encontrado!"
    });
  });
});

app.get("/artigo/:id", (req, res) => {
  Artigo.findOne({
    _id: req.params.id
  }).then((artigo) => {
    return res.json(artigo);
  }).catch((erro) => {
    return res.status(400).json({
      error: true,
      message: "Nenhum artigo encontrado!"
    });
  });
});
});
```

```

app.post('/artigo', (req, res) => {
  const artigo = Artigo.create(req.body, (err) => {
    if(err) return res.status(400).json({
      error: true,
      message: "Error: Artigo não foi cadastrado com sucesso!"
    })

    return res.status(200).json({
      error: false,
      message: "Artigo cadastrado com sucesso!"
    })
  })
});

app.put("/artigo/:id", (req, res) => {
  Artigo.updateOne({ _id: req.params.id }, req.body, (err) => {
    if(err) return res.status(400).json({
      error: true,
      message: "Error: Artigo não foi editado com sucesso!"
    });

    return res.json({
      error: false,
      message: "Artigo editado com sucesso!"
    });
  });
});

const PORT = 8080;

app.listen(PORT, ()=>{
  console.log("Servidor iniciado na porta " + PORT);
})

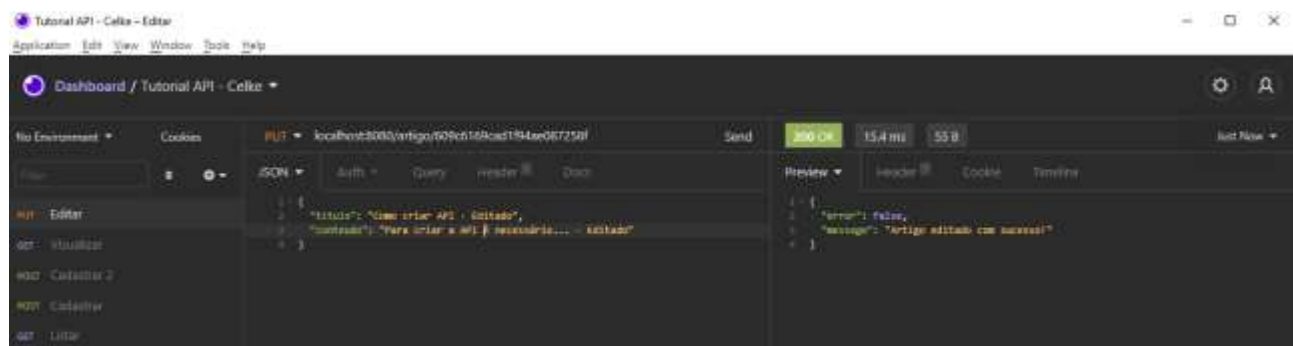
```

node app.js

```

C:\celke\nodejs\tutorial_api>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`
Servidor iniciado na porta 8080
Conexão com mongoDB realizada com sucesso!

```



Local

DBS COLLECTIONS

FAVORITE

HOST  
localhost:27017

CLUSTER  
Standalone

EDITION  
MongoDB 4.4.5 Community

Filter your data

- admin
- blogapp
- celke
- artigos
- config
- local
- testes\_mongo

celke.artigos Documents

Documents Aggregations Schema Explain Plan Indexes Validation

ENTER  + OPTIONS FIND RESET ↺ ⋮

ADD DATA + VIEW

Displaying documents 1 - 2 of 2 < > REFRESH

```
{ "_id": "objectID('a89c618ba2f94a4087328f')",  
  "titulo": "Como criar API - Exibido",  
  "conteudo": "Para criar a API é necessário... - Exibido",  
  "createdAt": "2023-05-12T23:14:40.821+00:00",  
  "updatedAt": "2023-05-12T00:49:24.403+00:00",  
  "...": 0  
}
```

```
{ "_id": "objectID('a89c6772a93402209fa7043')",  
  "titulo": "Como criar API com Node JS",  
  "conteudo": "Para criar a API com Node JS é necessário...",  
  "createdAt": "2023-05-12T23:39:35.879+00:00",  
  "updatedAt": "2023-05-12T23:39:35.879+00:00",  
  "...": 0  
}
```

## Aula 21 - Como criar delete com Nodejs na API - Parte 6

### app.js

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

require('./models/Artigo');
const Artigo = mongoose.model('artigo');

// Configuração

const app = express();
app.use(express.json());

mongoose.connect('mongodb://localhost/celke', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log("Conexão com mongoDB realizada com sucesso!");
}).catch((erro) => {
  console.log("Erro: Conexão com mongoDB não foi realizada com sucesso! " + erro);
});

// Criando rotas

app.get('/', (req, res) => {
  // return res.json({titulo: "Como criar API"});

  Artigo.find({}).then((artigo) => {
    return res.json(artigo);
  }).catch((erro) => {
    return res.status(400).json({
      error: true,
      message: "Nenhum artigo encontrado!"
    });
  });
});

app.get("/artigo/:id", (req, res) => {
  Artigo.findOne({
    _id: req.params.id
  }).then((artigo) => {
    return res.json(artigo);
  }).catch((erro) => {
    return res.status(400).json({
      error: true,
      message: "Nenhum artigo encontrado!"
    })
  });
});
});
```

```
app.post('/artigo', (req, res) => {
  const artigo = Artigo.create(req.body, (err) => {
    if(err) return res.status(400).json({
      error: true,
      message: "Error: Artigo não foi cadastrado com sucesso!"
    })

    return res.status(200).json({
      error: false,
      message: "Artigo cadastrado com sucesso!"
    })
  })
});

app.put("/artigo/:id", (req, res) => {
  Artigo.updateOne({ _id: req.params.id}, req.body, (err) => {
    if(err) return res.status(400).json({
      error: true,
      message: "Error: Artigo não foi editado com sucesso!"
    });

    return res.json({
      error: false,
      message: "Artigo editado com sucesso!"
    });
  });
});

app.delete("/artigo/:id", (req, res) => {
  const artigo = Artigo.deleteOne({ _id: req.params.id}, req.body, (err) => {
    if(err) return res.status(400).json({
      error: true,
      message: "Error: Artigo não foi apagado com sucesso!"
    });

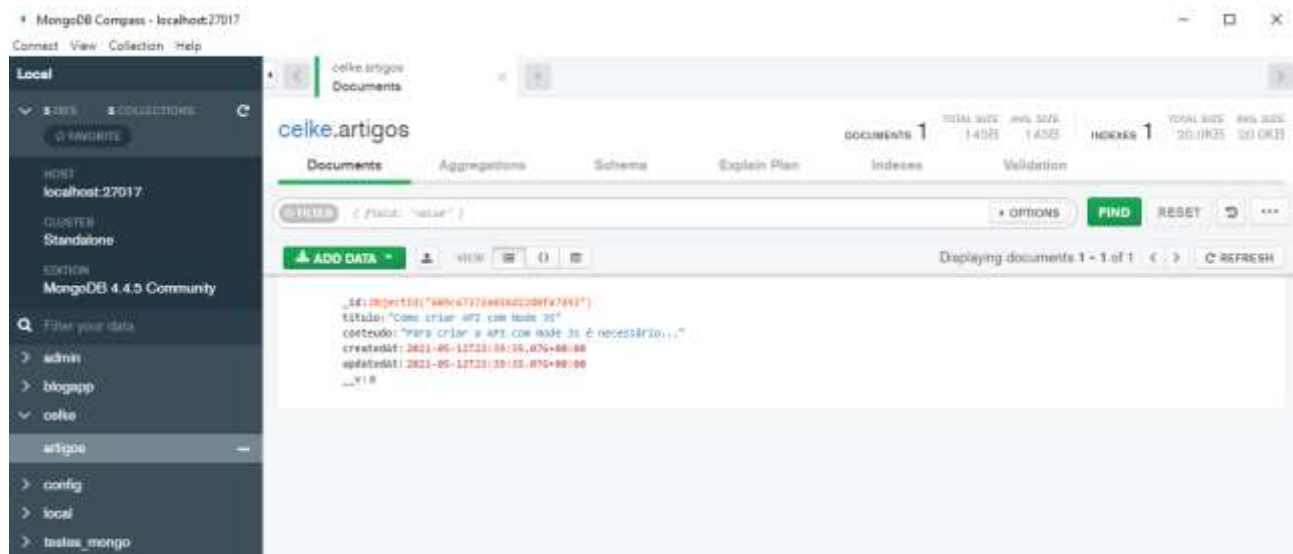
    return res.json({
      error: false,
      message: "Artigo apagado com sucesso!"
    });
  });
});

const PORT = 8080;

app.listen(PORT, ()=>{
  console.log("Servidor iniciado na porta " + PORT);
})
```

## nodemon app.js

```
C:\celke\nodejs\tutorial_api>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`
Servidor iniciado na porta 8080
Conexão com mongoDB realizada com sucesso!
```





- No insomnia, cadastre mais três artigos:



localhost:8080/artigo 90%

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
{
  "0": {
    "_id": "089c0737c405502defa9d3",
    "título": "Como criar API com Node JS",
    "conteudo": "Para criar a API com Node JS é necessário...",
    "createdAt": "2021-05-12T21:59:35.870Z",
    "updatedAt": "2021-05-12T21:59:35.870Z",
    "__v": 0
  },
  "1": {
    "_id": "089c074965939318c7510d5",
    "título": "React: o que é e como funciona essa ferramenta?",
    "conteudo": "Como definido por seus criadores, React é 'uma biblioteca JavaScript declarativa, eficiente e flexível para a criação de interfaces de usuário (UI)'."
    "createdAt": "2021-05-12T04:47:05.480Z",
    "updatedAt": "2021-05-12T04:47:05.480Z",
    "__v": 0
  },
  "2": {
    "_id": "089c08408653c518c7510d7",
    "título": "O que é e para que serve uma API?",
    "conteudo": "API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na web. Uma API é criada quando uma empresa de software tem a intenção de que outros criadores de software desenvolvam produtos associados ao seu serviço."
    "createdAt": "2021-05-12T04:51:29.343Z",
    "updatedAt": "2021-05-12T04:51:29.343Z",
    "__v": 0
  },
  "3": {
    "_id": "089c08408653c518c7510d8",
    "título": "O que é e para que serve o NodeJS?",
    "conteudo": "Node.js é uma tecnologia assíncrona que trabalha em uma única thread de execução. Por assíncrona entendo que cada requisição ao Node.js não bloqueia o processo do mesmo, atendendo a um volume simultaneamente grande de requisições ao mesmo tempo mesmo sendo single thread."
    "createdAt": "2021-05-12T04:53:57.840Z",
    "updatedAt": "2021-05-12T04:53:57.840Z",
    "__v": 0
  }
}
```

# Aula 22 - Como permitir acesso a API com CORS - Parte 7

## Como permitir e bloquear permissões de URL com CORS

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

### Installation

This is a **Node.js** module available through the **npm registry**. Installation is done using the **npm install** command:

```
$ npm install cors
```

`npm install --save cors`

```
C:\celke\nodejs\tutorial_api>npm install --save cors
npm WARN tutorial_api@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (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"})

+ cors@2.8.5
added 2 packages from 2 contributors and audited 199 packages in 40.566s

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

found 0 vulnerabilities
```

### Usage

#### Simple Usage (Enable All CORS Requests)

```
var express = require('express')
var cors = require('cors')
var app = express()

app.use(cors())

app.get('/products/:id', function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for all origins!'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

## Configuration Options

---

- `origin` : Configures the **Access-Control-Allow-Origin** CORS header. Possible values:
  - `Boolean` - set `origin` to `true` to reflect the `request origin`, as defined by `req.header('Origin')`, or set it to `false` to disable CORS.
  - `String` - set `origin` to a specific origin. For example if you set it to `"http://example.com"` only requests from `"http://example.com"` will be allowed.
  - `RegExp` - set `origin` to a regular expression pattern which will be used to test the request origin. If it's a match, the request origin will be reflected. For example the pattern `/example\.com$/` will reflect any request that is coming from an origin ending with `"example.com"`.
  - `Array` - set `origin` to an array of valid origins. Each origin can be a `String` or a `RegExp`. For example `["http://example1.com", /\.example2\.com$/]` will accept any request from `"http://example1.com"` or from a subdomain of `"example2.com"`.
  - `Function` - set `origin` to a function implementing some custom logic. The function takes the request origin as the first parameter and a callback (which expects the signature `err [object], allow [bool]`) as the second.
- `methods` : Configures the **Access-Control-Allow-Methods** CORS header. Expects a comma-delimited string (ex: `'GET,PUT,POST'`) or an array (ex: `['GET', 'PUT', 'POST']`).
- `allowedHeaders` : Configures the **Access-Control-Allow-Headers** CORS header. Expects a comma-delimited string (ex: `'Content-Type,Authorization'`) or an array (ex: `['Content-Type', 'Authorization']`). If not specified, defaults to reflecting the headers specified in the request's **Access-Control-Request-Headers** header.

// <https://celke.com.br/artigo/consumir-dados-da-api-propria-com-react>

## app.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

require('./models/Artigo');
const Artigo = mongoose.model('artigo');

// Configuração

const app = express();

// https://celke.com.br/artigo/consumir-dados-da-api-propria-com-react

app.use((req, res, next) => {
  // console.log("Acessou o Middleware!");
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Methods", 'GET,PUT,POST,DELETE');
  app.use(cors());
  next();
});

app.use(express.json());

mongoose.connect('mongodb://localhost/celke', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log("Conexão com mongoDB realizada com sucesso!");
}).catch((erro) => {
  console.log("Erro: Conexão com mongoDB não foi realizada com sucesso! " + erro);
});

// Criando rotas

app.get('/artigo', (req, res) => {
  // return res.json({titulo: "Como criar API"});

  Artigo.find({}).then((artigo) => {
    return res.json(artigo);
  }).catch((erro) => {
    return res.status(400).json({
      error: true,
      message: "Nenhum artigo encontrado!"
    });
  });
});

app.get("/artigo/:id", (req, res) => {
  Artigo.findOne({
    _id: req.params.id
  }).then((artigo) => {
    return res.json(artigo);
  }).catch((erro) => {
    return res.status(400).json({
      error: true,
      message: "Nenhum artigo encontrado!"
    })
  })
});
```

```

    });
  });

  app.post('/artigo', (req, res) => {
    const artigo = Artigo.create(req.body, (err) => {
      if(err) return res.status(400).json({
        error: true,
        message: "Error: Artigo não foi cadastrado com sucesso!"
      })

      return res.status(200).json({
        error: false,
        message: "Artigo cadastrado com sucesso!"
      })
    })
  });

  app.put("/artigo/:id", (req, res) => {
    Artigo.updateOne({ _id: req.params.id}, req.body, (err) => {
      if(err) return res.status(400).json({
        error: true,
        message: "Error: Artigo não foi editado com sucesso!"
      });

      return res.json({
        error: false,
        message: "Artigo editado com sucesso!"
      });
    });
  });

  app.delete("/artigo/:id", (req, res) => {
    const artigo = Artigo.deleteOne({ _id: req.params.id}, req.body, (err) => {
      if(err) return res.status(400).json({
        error: true,
        message: "Error: Artigo não foi apagado com sucesso!"
      });

      return res.json({
        error: false,
        message: "Artigo apagado com sucesso!"
      });
    });
  });

  const PORT = 8080;

  app.listen(PORT, ()=>{
    console.log("Servidor iniciado na porta " + PORT);
  })

```

# Como consumir dados da API com React

## Criando um projeto REACT

`npx create-react-app tutorialconsapi`

```
C:\celke\nodejs>npx create-react-app tutorialconsapi
npx: installed 67 in 42.946s

Creating a new React app in C:\celke\nodejs\tutorialconsapi.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

> core-js@2.6.12 postinstall C:\celke\nodejs\tutorialconsapi\node_modules\babel-runtime\node_modules\core-js
> node -e "try(require('./postinstall'))catch(e){}"

> core-js@3.12.1 postinstall C:\celke\nodejs\tutorialconsapi\node_modules\core-js
> node -e "try(require('./postinstall'))catch(e){}"

> core-js-pure@3.12.1 postinstall C:\celke\nodejs\tutorialconsapi\node_modules\core-js-pure
> node -e "try(require('./postinstall'))catch(e){}"

> ejs@2.7.4 postinstall C:\celke\nodejs\tutorialconsapi\node_modules\ejs
> node ./postinstall.js

+ react-dom@17.0.2
+ react@17.0.2
+ react-scripts@4.0.3
+ cra-template@1.1.2
added 1924 packages from 731 contributors and audited 1927 packages in 694.079s

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

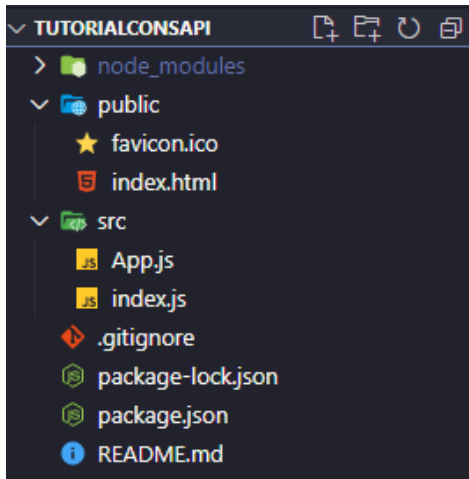
found 79 moderate severity vulnerabilities
  run `npm audit fix` to fix them, or `npm audit` for details
```

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

Pasta de C:\celke\nodejs

12/05/2021  23:13    <DIR>          .
12/05/2021  23:13    <DIR>          ..
10/05/2021  14:07    <DIR>          aula02
10/05/2021  14:29    <DIR>          aula03
10/05/2021  20:07    <DIR>          aula04-06
11/05/2021  17:13    <DIR>          aula07
11/05/2021  17:16    <DIR>          aula08
11/05/2021  17:24    <DIR>          aula09
12/05/2021  06:40    <DIR>          aula10
12/05/2021  06:51    <DIR>          aula11
12/05/2021  07:24    <DIR>          aula12
12/05/2021  09:55    <DIR>          aula13
12/05/2021  14:04    <DIR>          aula14
12/05/2021  14:44    <DIR>          aula15
12/05/2021  23:27    <DIR>          tutorialconsapi
12/05/2021  22:19    <DIR>          tutorial_api
                0 arquivo(s)                0 bytes
                16 pasta(s)          114.685.980.672 bytes disponíveis
```

- Deixar apenas os arquivos:



- Alterar os arquivos:

### **src\index.js**

```
import React from 'react';  
import ReactDOM from 'react-dom';  
import App from './App';
```

```
ReactDOM.render(<App />, document.getElementById('root'));
```

### **src\App.js**

```
import React from 'react';
```

```
function App() {  
  return (  
    <div>  
      <h1>Listar artigos</h1>  
    </div>  
  );  
}
```

```
export default App;
```



## public\index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <meta
    name="description"
    content="Web site created using create-react-app"
  />
  <title>React App</title>
</head>
<body>
  <div id="root"></div>
</body>
</html>
```

- Para rodar o projeto:

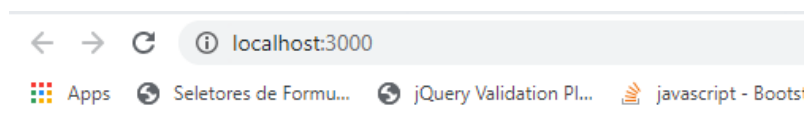
**npm start**

```
Compiled successfully!

You can now view tutorialconsapi in the browser.

Local:      http://localhost:3000
On Your Network:  http://10.0.0.2:3000

Note that the development build is not optimized.
To create a production build, use npm run build.
```



## Listar artigos

## Instalando o axios

npm install --save axios

```
C:\celke\nodejs\tutorialconsapi>npm install --save axios
npm WARN @babel/plugin-bugfix-v8-spread-parameters-in-optional-chaining@7.13.12 requires a peer of @babel/core@^7.13.0 but none is installed. You must install peer dependencies yourself.
npm WARN tsutils@3.21.0 requires a peer of typescript@>=2.8.0 || >= 3.2.0-dev || >= 3.3.0-dev || >= 3.4.0-dev || >= 3.5.0-dev || >= 3.6.0-dev || >= 3.6.0-beta || >= 3.7.0-dev || >= 3.7.0-beta but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (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"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\watchpack-chokidar2\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\webpack-dev-server\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ axios@0.21.1
added 1 package from 1 contributor and audited 1956 packages in 145.796s

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

found 79 moderate severity vulnerabilities
  run `npm audit fix` to fix them, or `npm audit` for details
```

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

### src\api.js

```
import axios from 'axios';
```

```
const api = axios.create({
  baseURL: 'http://localhost:8080'
});
```

```
export default api;
```

### src\App.js

```
import React, { Component } from 'react';
```

```
import api from './api';
```

```
class App extends Component{
```

```
  state = {
    artigos: [],
  }
```

```
  async componentDidMount(){
    const response = await api.get('/artigo');
```

```
    // console.log(response.data);
```

```
    this.setState({artigos: response.data})
  }
```

```
  render(){
```

```

const {artigos} = this.state;

console.log(artigos);

return(
  <div>
    <h1>Listar Artigos</h1>

    <ul style={{listStyleType: "none", margin: 0, padding: 0}}>
      {artigos.map(artigo => (
        <li key={artigo._id}>
          <h2>Título: {artigo.titulo}</h2>
          <p>Conteúdo: {artigo.conteudo}</p>
        </li>
      ))}
    </ul>

  </div>
);
}
}

```

export default App;

```

cd..
cd tutorial_api
code .

```

```

C:\celke\nodejs>cd tutorial_api
C:\celke\nodejs\tutorial_api>code .

```

nodemon app.js

```

C:\celke\nodejs\tutorial_api>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`
Servidor iniciado na porta 8080
Conexão com mongoDB realizada com sucesso!

```

# Listar Artigos

## **Título: Como criar API com Node JS**

Conteúdo: Para criar a API com Node Js é necessário...

## **Título: React: o que é e como funciona essa ferramenta?**

Conteúdo: Como definido por seus criadores, React é “uma biblioteca JavaScript declarativa, eficiente e flexível para a criação de interfaces de usuário (UI)”.

## **Título: O que é e para que serve uma API?**

Conteúdo: API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. Uma API é criada quando uma empresa de software tem a intenção de que outros criadores de software desenvolvam produtos associados ao seu serviço.

## **Título: O que é e para que serve o Node.JS**

Conteúdo: Node.js é uma tecnologia assíncrona que trabalha em uma única thread de execução. Por assíncrona entenda que cada requisição ao Node.js não bloqueia o processo do mesmo, atendendo a um volume absurdamente grande de requisições ao mesmo tempo mesmo sendo single thread.