

NodeJS – Módulo 01 – B7Web

Material de estudo desenvolvido por:
Daniel Teixeira Quadros

Sumário

Aula 01 – Instalando o NODE.....	2
Aula 02 – Iniciando o Projeto (1/2).....	2
Aula 03 – Iniciando o Projeto (2/2).....	5
Aula 04 – Instalando o Mongo DB (Windows).....	8

Aula 01 – Instalando o NODE

link:

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

comandos para verificar se o node e o npm estão instalados:

- Acesse o Prompt de Comandos e digite:

node -v → Para verificar a versão do node que está instalado.

npm -v → Para verificar a versão do npm instalado.

Estes são os aplicativos básicos iniciais para estudarmos e utilizarmos o NodeJS.

Aula 02 – Iniciando o Projeto (1/2)

Através do terminal do VSCode acessamos a pasta onde iremos iniciar o nosso projeto. E nesta pasta então, executamos o comando para iniciar o projeto.

1 - npm init → Comando que inicia um projeto.

Resultado:

```
See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (aula02) █
```

2 - Após este comando o sistema solicita que seja inserido as configurações básicas para o projeto, conforma o Print abaixo.

```

Press ^C at any time to quit.
package name: (aula02)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\Informática\Programação\2021\Daniel\B7Web\NodeJS\modulo1\aula02\package.json:

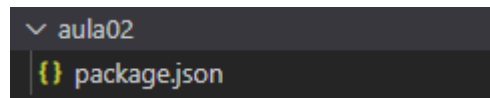
{
  "name": "aula02",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)
PS D:\Informática\Programação\2021\Daniel\B7Web\NodeJS\modulo1\aula02>

```

Obs.: Aqui na aula, apenas teclamos 'Enter' para todas as perguntas. De acordo com o Bonieky, nós preencheremos essas informações em outro momento.

3 – Com o nosso projeto criado, obtemos o arquivo package.json. Veja o resultado no VSCode:



4 – Abrindo o arquivo, temos os seguintes comandos iniciais:

```

aula02 > {} package.json > ...
1  {
2    "name": "aula02",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "",
10   "license": "ISC"
11 }
12

```

5 – Nesta etapa, precisamos instalar duas dependências iniciais, para isto, voltamos a acessar o terminal do projeto:

5.1 – **Express** → **npm install express --save** (Express é uma biblioteca que auxilia no desenvolvimento).

```
PS D:\Informática\Programação\2021\Daniel\B7Web\NodeJS\modulo1\aula02> npm install express --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN aula02@1.0.0 No description
npm WARN aula02@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 4.778s
found 0 vulnerabilities

PS D:\Informática\Programação\2021\Daniel\B7Web\NodeJS\modulo1\aula02>
```

5.2 – **Nodemon** → **npm install nodemon --save-dev** → O -dev instala o nodemon nas dependências de desenvolvimento. Isto será relevante quando estivermos fazendo BUILDING na nossa aplicação para poder botar ela online, ou enviar para um servidor...

```
PS D:\Informática\Programação\2021\Daniel\B7Web\NodeJS\modulo1\aula02> npm install nodemon --save-dev

> nodemon@2.0.7 postinstall D:\Informática\Programação\2021\Daniel\B7Web\NodeJS\modulo1\aula02\node_modules\nodemon
> node bin/postinstall || exit 0

Love nodemon? You can now support the project via the open collective:
> https://opencollective.com/nodemon/donate

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@~2.3.1 (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"})
npm WARN aula02@1.0.0 No description
npm WARN aula02@1.0.0 No repository field.

+ nodemon@2.0.7
added 117 packages from 53 contributors and audited 168 packages in 10.862s

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

found 0 vulnerabilities

PS D:\Informática\Programação\2021\Daniel\B7Web\NodeJS\modulo1\aula02>
```

Obs.: O nodemon, reinicia o nosso servidor a cada momento em que atualizamos a nossa aplicação, facilitando o trabalho, nos momentos que testarmos o que foi alterado. Utilizando o nodemon não precisamos reiniciar o servidor manualmente.

Resultado no VSCode:

```
aula02 > {} package.json > ...
1  {
2    "name": "aula02",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "",
10   "license": "ISC",
11   "dependencies": {
12     "express": "^4.17.1"
13   },
14   "devDependencies": {
15     "nodemon": "^2.0.7"
16   }
17 }
18
```

6 – Acompanhando o Bonieky, faremos aqui algumas modificações no nosso arquivo package.json apenas para ele ficar mais conciso:

Vamos **retirar** a linha '**descrição**', já que não utilizaremos.

Retiraremos a linha **'main'**, que não é essencial
O **'test'** que está abaixo da linha script, também removemos
Retiraremos também o **'author'** e a **'licence'**

Resultado no VSCode:

```
aula02 > {} package.json > ...
1  {
2    "name": "aula02",
3    "version": "1.0.0",
4    "scripts": {
5    },
6    "dependencies": {
7      "express": "^4.17.1"
8    },
9    "devDependencies": {
10     "nodemon": "^2.0.7"
11   }
12 }
13
```

Obs: O item scripts será utilizado para receber comandos(atalhos) específicos que podemos rodar.

Vamos criar os seguintes atalhos:

"start": "nodemon ./server.js" → Atalho que iniciará o nosso servidor.

O comando **nodemon ./server.js** irá rodar, quando acessarmos o nosso terminal e digitarmos **npm start** ele irá executar o arquivo **server.js** utilizando a dependência nodemon

Aqui, neste momento, o arquivo **server.js** ainda não foi criado, mas criaremos logo em seguida.

Resultado no VSCode:

```

MODULO1
└─ aula02
   ├── > node_modules
   ├── {} package-lock.json
   └── {} package.json

aula02 > {} package.json > {} scripts > start
1  {
2    "name": "aula02",
3    "version": "1.0.0",
4    "scripts": {
5      "start": "nodemon ./server.js"
6    },
7    "dependencies": {
8      "express": "^4.17.1"
9    },
10   "devDependencies": {
11     "nodemon": "^2.0.7"
12   }
13 }
14
```

Aula 03 – Iniciando o Projeto (2/2)

Obs.: Continuamos utilizando os mesmos arquivos criados na aula2

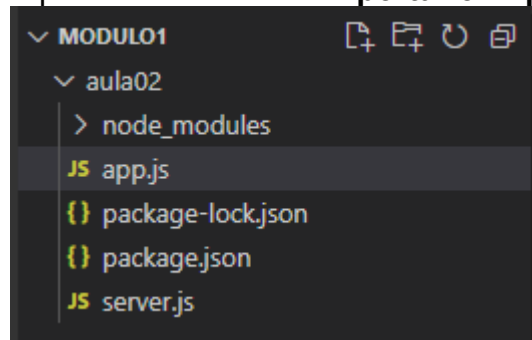
7 – Aqui, vamos criar o nosso arquivo **server.js** em paralelo com o package.json no VSCode.

8 – Neste arquivo, iremos puxar a nossa aplicação utilizando o Express (dependência que importamos anteriormente).

Obs.: Logo em seguida, faremos ela rodar em uma porta específica e iniciar o servidor.

9 – Criamos outro arquivo, também paralelo com o package.json. O **app.js**. → o nome app.js é apenas para dar a ideia de aplicação.

10 – No app.js, a primeira coisa que precisamos fazer é **importar o Express**.



11 – Para importar o Express para o app.js utilizamos os seguintes comando:

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

12 - Precisamos estabelecer as rotas:

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

Estamos utilizando o Express para criar nossa primeira rota

13 – Utilizando a variável **'router'**

```
router.get('/', (req, res) => {  
  res.send('Olá Mundo!')  
})
```

Esta é a primeira rota, a **'/'** → raiz (home ou página inicial) do site, do sistema.

Criamos, após o caminho uma função anônima. Esta função tem dois parâmetros:

1º - **req** → requisição

2º - **res** → resposta

Obs.: Por enquanto, utilizaremos apenas a resposta.

Quando o usuário acessar a página inicial, o comando **res.send** enviará uma resposta

14 – Configurações:

const app = express() → Inicia o aplicativo.

app.use('/', router) → Esta linha de comando enviará todas as rotas existentes. Até o momento, criamos apenas uma. O Bonieky não explicou claramente o **.use**, porém sobre esta linha, por enquanto isto foi o que ele explicou sobre o que ela faz.

module.exports = app → Esta linha é responsável por exportar o app.js.

Obs.: Precisamos exportar o app.js, porque importaremos ele lá no server.js.

```
aula02 > JS app.js > ...  
1  const express = require('express')  
2  const router = express.Router()  
3  router.get('/', (req, res) => {  
4    res.send('Olá Mundo!')  
5  })  
6  const app = express()  
7  app.use('/', router)  
8  module.exports = app
```

INICIALMENTE, PARA A NOSSA APLICAÇÃO FUNCIONAR, JÁ TEMOS O MÍNIMO NECESSÁRIO NO NOSSO app.js.

App.js fica assim:

AGORA PASSAREMOS PARA AS CONFIGURAÇÕES REALIZADAS NO ARQUIVO `server.js`.

`const app = require('./app')` → Esta linha que está importando o `app.js` para cá.

`app.set('port', 7777)` → Nesta linha, estamos configurando uma porta para o nosso servidor, que aqui, estamos utilizando, juntamente com o Bonieky a porta `'7777'`.

**`const server = app.listen(app.get('port'), ()=>{
 console.log("Servidor rodando na porta"+server.address().port)
})`** → Esta linha é que realmente inicia o nosso servidor. Com o comando **`listen`**, ele passa a observar (ouvir) uma determinada porta (que no nosso caso é a `'7777'`) e utilizamos o **`app.get('port')`** para puxar da própria porta que configuramos.

Obs.: Mais para frente, faremos com que o comando `'app.set('port', 7777)'` fique dinâmico, por isso utilizamos `'app.get('port')'` aqui.

Existe um segundo parâmetro que podemos configurar no `app.get` que nos auxilia a mostrar no terminal que o servidor está rodando e em qual porta. Isto é utilizado para que o usuário possa ver estas informações e facilitar o seu trabalho.

Para isto, criamos uma função anônima no segundo parâmetro e para o usuário poder visualizar estas informações, adicionamos no `console.log` o comando `'server.address().port'` → Esta função pega a porta em que o servidor está rodando.

INICIALMENTE, PARA A NOSSA APLICAÇÃO FUNCIONAR, JÁ TEMOS O MÍNIMO NECESSÁRIO NO NOSSO `server.js`.

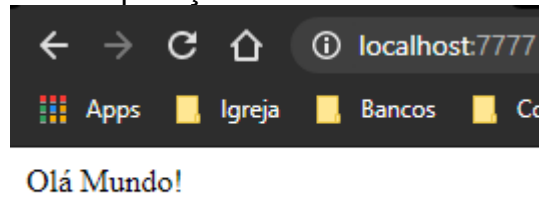
O `server.js` ficou assim:

```
aula02 > JS server.js > ...  
 1  const app = require('./app')  
 2  app.set('port', 7777)  
 3  const server = app.listen(app.get('port'), ()=>{  
 4    console.log("Servidor rodando na porta"+server.address().port)  
 5  })
```

A partir daqui, já temos o nosso servidor criado aqui no `'server.js'` e a nossa aplicação criada no `'app.js'`. Agora **para dar início** (subir) o projeto, vamos no terminal e executamos o comando **`npm start`**.

```
PS D:\Informática\Programação\2021\Daniel\B7Web\NodeJS\modulo1\aula02> npm start  
  
> aula02@1.0.0 start D:\Informática\Programação\2021\Daniel\B7Web\NodeJS\modulo1\aula02  
> nodemon ./server.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 ./server.js`  
Servidor rodando na porta7777
```

E, ao digitarmos no navegador o comando **localhost:7777** (localhost → máquina local, 7777 → porta que configuramos) conseguimos ver a nossa aplicação rodando:



Aula 04 – Instalando o Mongo DB (Windows)

link:

<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>

Procurar o link MongoDB Community Edition e Mongo DB Download Center.
MongoDB

No link que abrir, precisamos escolher o MogoDB Community Server

- Version.
- Plataforma
- Package

MongoDB Community Server

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


The MongoDB Enterprise Server gives you all of this and more. Review the Enterprise Server tab to learn what else is available.

Available Downloads

Version
4.4.6 (current) ✓

Platform
Windows ✓

Package
msi ✓

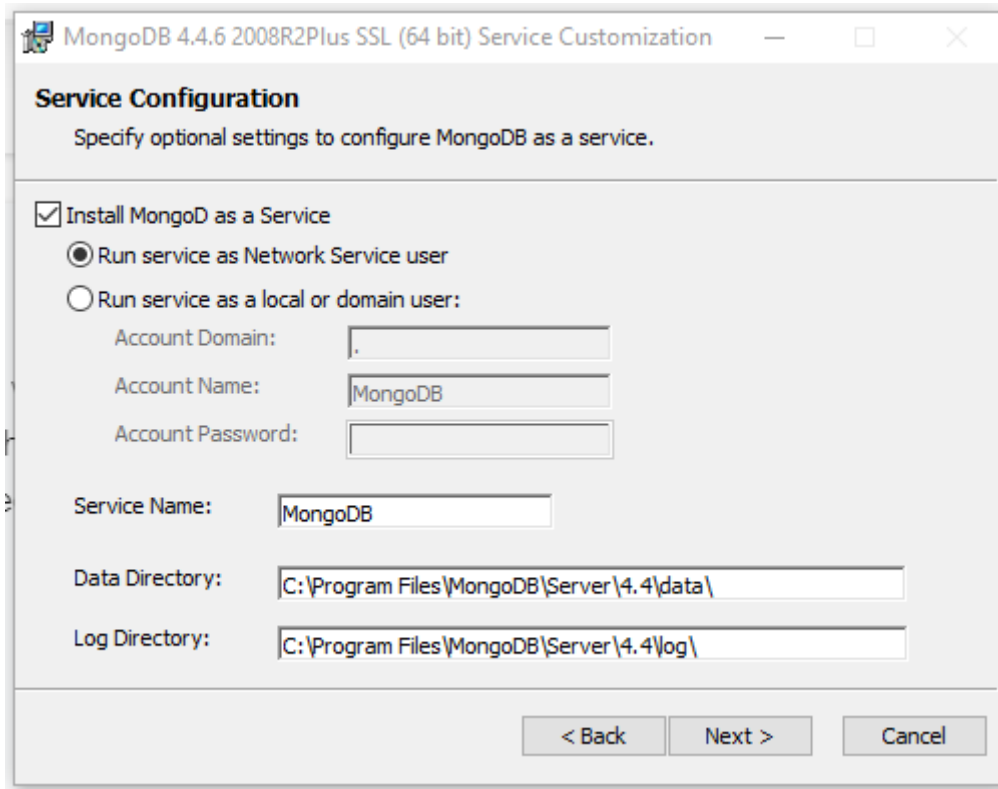
 **Download** [Copy Link](#)

Durante a instalação do MongoDB, escolhemos a versão completa.

Nesta tela, existe a opção de não deixarmos o MongoDB rodando sempre, desde que o computador iniciar, ou a opção de iniciarmos o MongoDB manualmente:

1 – Se quisermos iniciar o MongoDB manualmente, precisamos desmarcar a opção ‘Install MongoDB as a Service’.

Obs.: Aqui, a recomendação do Bonieky é que deixemos marcado esta opção, para não termos que ficar preocupados se o servidor está rodando ou não.



O Restante das configurações deixamos como está.

Obs.: É muito importante que anotemos em algum lugar de fácil acesso e visualização os dois diretórios que aparecem aqui:

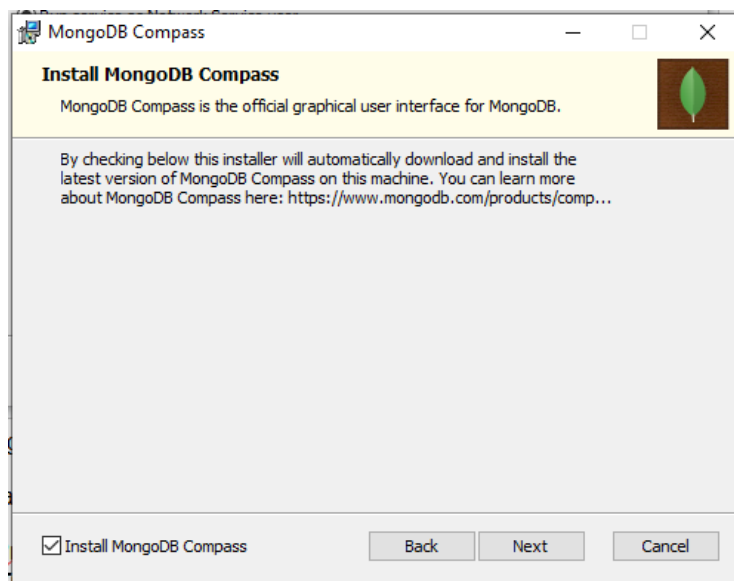
Data Directory: C:\Program Files\MongoDB\Server\4.4\data\

Log Directory: C:\Program Files\MongoDB\Server\4.4\log\

O Log Directory é onde serão armazenados todos os logs do sistema, é onde poderemos encontrar algum erro que venha acontecer, lista de acesso, bem como também, todos os outros eventos.

Data Directory é onde efetivamente ficarão os bancos de dados dos sistemas que iremos criar.

Na próxima tela deixamos marcado o Compass, que é a aplicação visual do banco de dados



Após a instalação, o Compass

irá iniciar pela primeira vez. Então aparecerá a tela abaixo:

Aqui, podemos deixar todas as opções marcadas.

Privacy Settings

To enhance the user experience, Compass can integrate with 3rd party services, which requires external network requests. Please choose from the settings below:

- ☒ **Enable Product Feedback Tool**
Enables a tool for sending feedback or talking to our Product and Development teams directly from Compass.
- ☒ **Enable Geographic Visualizations**
Allow Compass to make requests to a 3rd party mapping service.
- ☒ **Enable Crash Reports**
Allow Compass to send crash reports containing stack traces and unhandled exceptions.
- ☒ **Enable Usage Statistics**
Allow Compass to send anonymous usage statistics.
- ☒ **Enable Automatic Updates**
Allow Compass to periodically check for new updates.

With any of these options, none of your personal information or stored data will be submitted.
Learn more: [MongoDB Privacy Policy](#)

[Start Using Compass](#)

Uma vez instalado, ele já está rodando o Node no Computador. Podemos realizar o primeiro teste:

Obs.: É importante saber, que apesar de termos instalado esta interface gráfica do Mongo, também, se quisermos, podemos utilizá-lo no terminal.

Ao iniciarmos o MongoDB Compass, há uma pequena diferença entre a versão que ele ensinou no curso. Ao invés de iniciar com a tela de configurações iniciais, a primeira tela que aparece tem apenas o campo para inserirmos a nossa **connection String**. Para irmos para as configurações, precisamos clicar em **Fill in connection fields individually**

New Connection

☆ FAVORITE

Fill in connection fields individually

Paste your connection string (SRV or Standard ⓘ)

e.g. mongodb+srv://username:password@cluster0-jtpxd.mongodb.net/admin

[Connect](#)

Ao clicarmos em 'Fill in connection fields individually', Temos duas guias com todas as configurações:

New Connection

☆ FAVORITE

Paste connection string

Hostname

More Options

Hostname

localhost

Port

27017

SRV Record

☐

Authentication

None

Connect

e

New Connection

☆ FAVORITE

Paste connection string

Hostname

More Options

Replica Set Name

Read Preference

Primary

SSL

None

SSH Tunnel

None

Connect

- Agora vamos conectar em um **host específico**

Hostname: **localhost** → nosso computador

Port: **27017**

Authentication: Não precisa de autenticação, pois não estamos rodando com nenhum serviço aqui. Apenas isto. Se quisermos salvar, podemos clicar no link 'Favorite' e darmos um nome a ele.




Por último, clicamos em 'Connect'.

Automaticamente ele já cria os três servidores abaixo:

Databases

Performance

CREATE DATABASE

Database Name ^	Storage Size	Collections	Indexes	
admin	20.0KB	0	1	
config	4.0KB	0	2	
local	36.0KB	1	1	

Obs.: Se precisarmos desconectar o servidor para conectarmos outro, basta clicarmos em 'Connect' no canto esquerdo superior da tela e Desconectarmos o servidor ativo.

Veremos agora, como realizamos a conexão do servidor, via Prompt de comando, se necessário:

No Prompt de comando, executamos o comando 'mongo' dentro do seguinte diretório:

```
c:\Program Files\MongoDB\Server\4.4\bin>mongo
```

Obs. Aqui, a pasta **4.4** pode mudar de nome, de acordo com a versão instalada.

Ao executarmos, temos o seguinte resultado:

```
c:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.6
connecting to: mongod://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongod
Implicit session: session { "id" : UUID("edd9d71d-8dcd-4ad2-91b5-1c276254c963") }
MongoDB server version: 4.4.6
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
  https://community.mongodb.com
---
The server generated these startup warnings when booting:
  2021-05-18T09:09:11.704-03:00: 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()
---
```

Para fechar o mongo aqui, precisamos as teclas de atalho 'CTRL+C'

```
> ^C
bye

c:\Program Files\MongoDB\Server\4.4\bin>
```