Curso de NODEjs da Rocketseat modulo-1

O Nodejs surgiu para que possamos utilizar javascript no backEnd.

> **Principal Diferença**

Noblockio - input e output não bloqueável. No nodejs , eu não tenho uma sequência que o código deve correr, ele tem algo que chamamos de **eventloop**. Podemos utilizar multiplas ações ao mesmo tempo.

Seções

Instalação

> **Instalação do Node e Yarn**

Para Ubuntu e Debian.

Links

Instalação do Node

Instalação do Yarn

Para verificar se o nodejs está instalado:

```
node -v
```

Para verificar se o yarn está instalado:

```
yarn -v
```

Criar um novo projeto com nodes

Deve criar uma pasta em qualquer ludar de preferencia. Eu criei uma pasta chamada **www**. Dentro dela existem vários projetos. Esse primeiro projeto, vamos chamar de modulo1.

```
sudo mkdir modulo1
```

Vou entrar na pasta **cd /modulo1** e para iniciar o projeto vou colocar o seguinte código:

```
yarn init -y // o init é para inicializar um projeto e o -y seria para confirmar todos os pacotes que ele deseja instalar
```

Para visualizar o código vamos utilizar o visual studio code.

Ele irá criar um arquivo chamado package.json. Onde irá ficar todas as dependências do projeto.

Configurações do Vscode e Instalação de Dependências.

Editorconfig

- **Configurar o EditorConfig**

  Isso vai garantir que nosso editor esteja configurado, independente de qual editor, estamos utilizado, seja vscode, seja sublime etc. Isso é para que possamos trabalhar em equipe e todos terem o mesmo Permite que todos tenham a mesma configuração não importando em que ambiente esteja, seja linux , mac ou windows.

  Para que possamos instalar deve procurar em extensões "Extensions" no Vscode e instalar.

   Philadelphia's Magic Gardens. This place was so cool!

  Para que possamos configurar, vamos criar um arquivo na raiz do projeto chamado **.editorconfig** e vamos colocar algumas instruções no arquivo.

```
root = true

[*]
indent_style = space
indent_size = 2
charset = utf-8
trim_trailing_whitespace = true
insert_final_newline = true
```

> root = true //siginifica que é o arquivo inicial do editorconfig

> [*] // siginifica que é para todos os tipos de arquivos, poderiamos ter colocado [js] ou [njk] etc

> indent_style = space // significa que iremos identar utilizando espaços ao invés de tabs

> indent_size = 2 // a cada linha eu quero 2 espaços de identação

> charset = utf-8 // o tipo de charset que estou utilizando

> trim_trailing_whitespace = true //remove os espaços em brancos utilizados no final da linha

> insert_final_newline = true // significa que iremos colocar sempre uma linha em branco no final do código

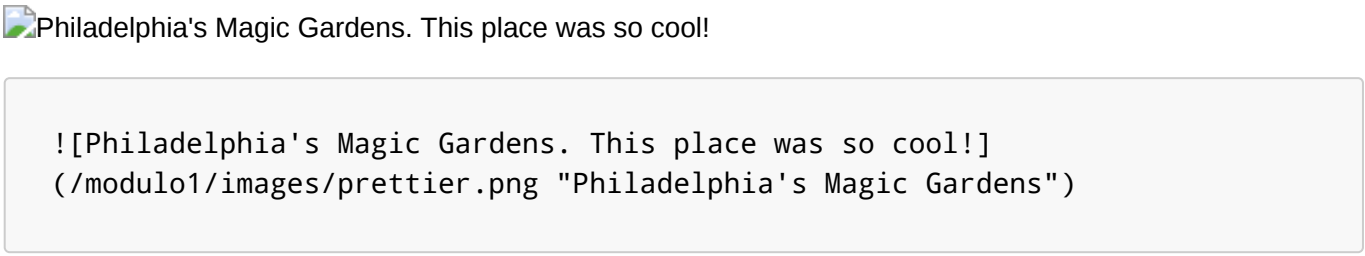
---

ESLint

- **Configurar ESLint e Prettier**

> O ESLint ele vai avisar o dev , caso algum problema de estilo no código ou seja o código pode não ter ponto e vírgula no final, string com aspas simples etc. Isso é importante para que a equipe possa estar formatando os códigos da mesma forma.
>
> O prittier ele irá fazer de forma automática a formatação.

Precisamos instalar as dependências indo no Extensions




```
![Philadelphia's Magic Gardens. This place was so cool!]
(/modulo1/images/prettier.png "Philadelphia's Magic Gardens")
```

Devemos ir no terminal e digital:

> yarn add eslint -D

Depois de instalar deve ir no terminal novamente e digitar:

> npx eslint --init

Irá aparecer algumas perguntas: Clique em

> Use a popular style guide

depois:

> Standart

depois o tipo de arquivo:

> JSON

e por ultimo ele pede se você quer instalar as dependências que estão faltando, então clique em Yes

> Y

Ele irá criar um arquivo chamado **package-lock.json**. Esse arquivo foi criado pelo npm. Para que possamos utilizar somente yarn, temos que apagar esse arquivo e digitar depois no terminal.

> yarn

Configuração:

Para configurar, deve abrir o user settings e colocar duas informações.

> "prettier.eslintIntegration":true,

> "editor.formatOnSave":true

Depois dessa configuração só salvar o arquivo novamente.

Express

- **Instalar o Express**

  Vamos instalar o miniframework chamado **Express**. Ele serve para trabalharmos com rotas e views. Para instalarmos. Só entrar no terminal pelo vscode mesmo e digitar:

  > yarn add express

  Para analisarmos, podemos observar que no arquivo package.json , já foi criado a dependência do express. Ele sempre pega a última versão do pacote.

```
"dependencies": {
    "express": "^4.16.4"
  }
```

Também podemos ver que foi criada uma pasta **node_modules** , ela guarda todas as dependências da nossa aplicação.

Nodemon

- **Instalar o Nodemon**

  O Nodemon, serve para que possamos iniciar o servidor automaticamente, sempre que tiver uma mudança no código, isso facilita em não mais ter que parar o servidor e dar start para que a mudança possa aparecer. Para que possamos instalar devemos digitar o seguinte código no terminal:

  > yarn add nodemon -D

  O **-D** siginifica que iremos instalar uma dependência de desenvolvimento ou seja quando subir a aplicação para produção essa dependência não é necessária.

- **Instalar o Nunjucks** Serve para retorna uma view em html. Conteúdo estruturado e estilisado. Ou seja uma forma de renderizar o html e javascript. Para instalar devemos digitar o seguinte código no terminal:

  > yarn add nunjucks

Http

Criar um servidor HTTP simples:

O http é para criarmos servidores simples, sem muita lógica. Vamos importar o http.

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

Vamos utilizar o metodo **createServer** do http para criarmos o nosso servidor

```
http.createServer(req, res);
```

O **" req "** é uma função para manipular as requisições do servidor.

O **" res "** é uma função para manipular as respostas do servidor.

Existe outra função que é a **.listen(3000)**, onde declaramos em qual porta vai ser executado nosso servidor.

O código ficará assim:

```
http.createServer((req, res) => {
    console.log(req);
    return res.end("Hello world");
  }).listen(3000);
```

O **console.log** funciona como o println,mostra o resultado no terminal

Ao chamar o servidor,ele retorna o "Hello world"

Agora iremos utilizar todas as bibliotecas que instalamos, o express e o nunjucks.

Vamos tirar o http e colocarmos express

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

Agora para criarmos um servidor com o express é muito mais fácil , basta utilizar:

```
const app = express();

app.listen(3000);
```

Com isso vamos trabalhar com rotas, basta digitarmos

```
app.get("/", (req, res) => {});
```

Se eu não quiser nenhuma rota, basta deixar apenas **'/'**

O express tem algumas funcionalidades a mais como o res.send, res.json etc. Agora para retornarmos algo na chamada da rota , basta colocar isso no código

```
return res.send("hello world")
```

Para vermos o poder do express, vamos incluir mais uma rota

```
app.get("/login", (req, res) => {
  return res.send("Login");
});
```

O código até aqui fica dessa forma:

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

const app = express();

app.get("/", (req, res) => {
  return res.send("hello world");
});

app.get("/login", (req, res) => {
  return res.send("Login");
});

app.listen(3000);
```

Podemos também criarmos rotas com parametros Para isso basta criarmos a seguinte rota

```
app.get("/nome/:name", (req, res) => {
  return res.send("Login");
});
```

> onde **name** é o parametro Para pegarmos o parametro basta incluir no código ${req.params.name}

Que foi o nome que demos para o parametro

código:

```
app.get("/nome/:name", (req, res) => {
  return res.send(`Bem vindo - ${req.params.name}`);
});
```

Há também os **queryParms** Onde passamos os parametros por e comercial ou interrogação. Exemplo: http://localhost:3000/?estufa=07&checkdate=2018-11-21

Para pegarmos o valor do parametro utilizamos

> req.query.name

```

```

```
app.get("/", (req, res) => {
  return res.send(`Bem vindo - ${req.query.name}`);
});
```

Também existe outra forma de resposta que é a **json**

```
app.get("/", (req, res) => {
  return res.json({
    message: `Bem vindo, ${req.params.name}`
  });
});
```

Middleware

Toda vez que criamos uma função que recebe **req** e **res** chamamos isso no node como se fosse middleware. Ele é um interceptador. Ele pode interceptar uma requisição e devolver uma resposta.

Vamos testar, criando um middleware de log com os parametros que recebemos na requisição

```
const logMiddleware = (req, res, next) => {
  console.log(
    `HOST:  ${req.headers.host} | URL: ${req.url} | METHOD: ${req.method}`
  );

  req.appName = "GoNode";

  return next();
};
```

Para isso, vamos criar um arquivo chamado **index.js**

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

nunjucks.configure("views", {
  autoescape: true,
  express: app,
  watch: true
});

app.use(express.urlencoded({ extended: false }));

app.set("view engine", "njk");
```

```
// aqui crio array
const users = ["Samira Santos", "Sofia Santos", "Daniel Santana"];

app.get("/", (req, res) => {
  return res.render("list", { users });
});

app.get("/new", (req, res) => {
  return res.render("new");
});

app.post("/create", (req, res) => {
  // push para adicionar dados no array
  users.push(req.body.user);

  // redirecionar para a listagem
  return res.redirect("/");
});

app.listen(3000);
```