# **Instalação do NodeJs**

# **Introdução**

Bem vindo ao curso de NodeJs. Durante toda a jornada de aprendizado de Javascript, aprenderemos bastante sobre a plataforma Node, entenderemos o que é esse tal de NodeJs, e como vamos utilizá-lo como plataforma para desenvolvermos nossa aplicação.

Antes de mais nada, assim como em vários ambientes de programação, a gente precisa ter uma configuração básica na nossa máquina, para que tudo funcione.

Por exemplo, se você já programou alguma vez em Java, ou se você quer programar no Java, depois que você escreveu os arquivos com a extensão .java, com código fonte java, você precisa ter alguém para rodar aquele seu código, no caso do Java você precisa ter um compilador.E em seguida você precisa ter um ambiente para executar o seu código, você instala a máquina virtual do Java, que é o seu ambiente, e ela se torna a sua plataforma onde você roda código.

No caso do Javascript, nós somos bem acostumados a rodar os nossos códigos no navegador, mas agora iremos rodar código em outro ambiente, que é justamente o NodeJs. Então encare ele como a sua plataforma, o seu ambiente onde você vai executar o seu código fonte. O Node vai te fornecer muitas bibliotecas, a nós iremos desbravá-lo bem durante o curso.

## **Download e Instalação**

Na data de publicação deste curso, a versão atual era a 4.2.1. O NodeJs se fundiu com outro projeto que se chamava io.js, que era uma versão separada, que chamamos de "fork", do NodeJs e que evolui num ritmo um pouco mais acelerado. Passado um tempo, depois entenderam que era melhor juntar forças para deixar sempre as últimas atualizações no NodeJs, já que o Node é o cara mais conhecido na comunidade Javascript, em vez de dividir a comunidade entre vários outros forks. Com o Node podemos utilizar a última versão suportada do Javascript.

Você vai baixar o NodeJS para o seu ambiente pelo link <https://nodejs.org/en/> . No ambiente do Mac, clicando em "Download for OS X (x64)" ele vai baixar o executável do Mac para poder ser instalado. No Linux você pode instalar também dessa maneira, ou pode instalar via apt-get. No Windows você vai ter o instalador do Node no seu ambiente tando em x86 ou 64bits.

## **Testando o NodeJs**

Com o NodeJs instalado, você automaticamente vai ganhar no seu terminal o executável node. Então por exemplo, se executarmos o comando node -v, ele vai mostrar a versão que está instalada do NodeJs.

Também é importante, porém não será explicado exatamente agora, mas usaremos bastante o npm , que é chamado de Node Package Manager, o cara que vai nos ajudar a baixar bibliotecas, rodar scripts e tudo mais. Nós iremos falar dele durante o curso, mas é importante que depois de instalar o NodeJS, você também seja capaz de exibir a versão do npm com o comando npm -v. Faça este teste e confira se ambas as instalações ocorreram corretamente.

## **Rodando a Primeira Aplicação**

Com o NodeJS já instalado na nossa máquina, chegou a hora de rodar o nosso primeiro código do lado do servidor. A primeira coisa que vamos fazer é simplesmente criar um diretório para abrigar esse arquivo e manter uma organização aqui dentro. Criaremos o diretório chamado teste-node, executando os seguintes comandos no terminal do Mac, ou no prompt do Windows:

Para criar o diretório: mkdir teste-node

Navegar para esse diretório: cd teste-node

Agora a gente precisa criar um arquivo que vai ser interpretado e executado dentro do NodeJs. Então abrindo o nosso editor de texto, aqui estamos usando o Sublime (fique a vontade para usar o editor que você preferir). Vamos criar um novo arquivo dentro da pasta teste-node e vamos salvá-lo chamando-o de produtos.js, justamente um arquivo para já entramos no clima do que vamos desenvolver durante o curso, que é uma parte da administração da casa do código. Nós vamos cadastrar um produto, listar um produto, remover um produto e editar... E depois a gente vai exibir os produtos cadastrados como se fosse a home da casa do código.

Aqui para começar vamos simplesmente imprimir uma mensagem, justamente simulando o que a gente quer fazer, que é a nossa listagem de produtos.

console.log("listagem de produtos");

Quem já vem de aplicações web, quem já teve de lidar com Javascript no navegador, já deve ter usado bastante o console para logar as cores, para saber se um evento está funcionando, e por aí vai.

Voltando para o Terminal, dentro do diretório teste-node, basta utilizar o executável node que fica disponível para você acessar pela linha de comando, e passar o nome do seu arquivo:

node produtos.js

Conseguimos executar justamente o que precisava passando o Javascript e usando o Node para executá-lo, a gente já tem o nosso primeiro código Javascript rodando num servidor.

“Listando os produtos" no Navegador

Legal galera, a gente acabou mostrando nossa mensagem que a gente estava listando os produtos no console, mas se formos pensar bem, a gente não quer bem isso. Nós queremos poder listar nossos produtos numa página do navegador. Então para a gente começar a pensar um pouco mais sobre isso, vamos começar a ver o que a gente vai precisar fazer.

Basicamente vamos no navegador, queremos digitar ‘localhost:3000’e queremos que apareça nossa listagem de produtos, pois nesse momento está aparecendo ‘This webpage is not available’. Então para resolvermos isso, primeiramente vamos fechar esse arquivo que criamos, a gente vai criar um novo arquivo que já vou salvar e vamos chamar de ‘produtosWeb.js’. A ideia é justamente que a gente faça o nosso primeiro código que dado o que a gente vai digitar o endereço no navegador, que ele consiga interpretar o que foi digitado ali e devolver uma resposta para ser exibida no navegador.

Para conseguir digitar alguma coisa no navegador, dar enter e devolver alguma coisa para o navegador, a gente precisa de um servidor web, um servidor que consiga escutar essas solicitações que vem de um cliente, no caso o navegador. Para fazer isso, nós vamos utilizar uma biblioteca disponível aqui no NodeJs que chama ‘http’.

‘var http = require(‘http’);’

Acostume-se a utilizar o ‘require’, pois ele é a função no Javascript dentro do Node para a gente importar tudo que a gente precisa, então por exemplo, se você quer importar um módulo que já vem pronto no NodeJs como o http, que é a biblioteca responsável por subir o servidor, fazer ele escutar e entender o pedido e tudo mais. Você vai querer fazer referência a um arquivo que você criou no seu projeto e importar ele em outro arquivo, você vai usar o ‘require’. Você vai querer importar uma biblioteca que você adicionou durante a implementação do seu projeto, com alguma funcionalidade adicional que você não queria implementar o do zero, você vai usar o ‘require’. Então vamos utilizar ele bastante durante o nosso curso.

Importando o ‘require’, ele importou um objeto que representa justamente a biblioteca do http. Dado isso, a gente vai agora invocar um método cujo nome é ‘createServer’, para nós criarmos o nosso servidor.

http.createServer();

Logo abaixo, nós vamos simplesmente mostrar uma mensagem dizendo que o servidor está rodando.

console.log(“servidor ta rodando);

A gente invocou o método que cria, agora vamos rodar o nosso arquivo no terminal com o executável NodeJs disponível depois da nossa instalação e o nome do arquivo, com o comando ‘node produtosWeb.js’, e ele exibe a mensagem ‘servidor ta rodando’.

Se formos no navegador, ele ainda exibe a mensagem ‘This webpage is not available’, então temos que pensar um pouco no porquê da gente não ter conseguido ter nossa página aí.

Se você ficou bem atento, você deve ter ficado questionando durante o momento que fomos executar, pois não criamos nada de html em lugar nenhum se magicamente iria aparecer alguma coisa. Você tem razão, não apareceu nada, e tem outro detalhe que é interessante, a gente deu um ‘servidor ta rodando’ e o terminal não ficou bloqueado para a gente, apareceu a mensagem e ele já liberou. Se ele estava rodando, por que ele não ficou esperando aqui com o terminal travado. Essa é só uma coisa que você pode parar para pensar. Você poderia deixar ele rodando por exemplo em background e o terminal não ficaria travado do mesmo jeito, é só um detalhe que você pode prestar atenção.

Então o nosso problema foi que quando você invoca o ‘createServer’, ele até cria um objeto que representa o seu servidor, você pode guardar aqui na variável.

‘var server = http.createServer();’

Só que ele não levanta esse servidor pra ficar escutando os pedidos que a gente vai fazer a partir do navegador.

Para fazer isso, no editor nós invocamos um método chamando ‘listen’ e dizemos exatamente a porta que queremos escutar, que no nosso caso, é a porta 3000.

‘server.listen(3000);’

Se rodarmos o arquivo novamente, o terminal agora irá ficar bloqueado e a mensagem ‘servidor ta rodando’ aparecerá. Indo no navegador e digitando localhost:3000, até irá aparecer o gif animado circular informando que a requisição está sendo feita, só que nada é respondido para a gente.

Se a gente for olhar bem, só precisamos entender um pouquinho mais de como funciona uma solicitação de uma página no navegador. Por exemplo, indo no navegador e digitando alura.com.br, a página aparece no navegador. Basicamente quando você digita endereço na barra do navegador você está fazendo o que a gente chama de requisição para o servidor, você está fazendo um novo pedido por uma página geralmente. Tanto que se você clicar com o botão direito no seu Chrome em todas as páginas que você acessa e der um “View page source” (em português, “Exibir código fonte da página”), você vai ver que todo o HTML foi devolvido aqui. Nem sempre vai ser devolvido um HTML, pode ser que seja devolvido um Javascript, pode ser que seja devolvido um arquivo de estilo, e por aí vai, mas em geral ele vai devolver páginas HTML.

Então basicamente é isso que a gente precisa fazer agora, a gente precisa ensinar para o servidor que a gente acabou de criar que ele é capaz de atender uma requisição e devolver uma resposta. Por isso que a gente passa para ele uma função que recebe justamente uma requisição e uma resposta. A ideia é que essa função seja chamada sempre que chegue uma nova requisição no servidor que você acabou de criar. Você pode pegar aqui agora e fazer o código ‘res.end();’. Repare que aqui a gente pode inclusive escrever o HTML que a gente quer devolver.

‘var server = http.createServer(function(req, res){ res.end(“<html><body><h1>Listando os produtos</h1></body></html>“); });’

Se você for no terminal agora e subir de novo o servidor com o comando ‘node produtosWeb.js’, e acessar o nosso endereço ‘localhost:3000’ no navegador, “Listando os produtos” já aparece para a gente no navegador. Se a gente olhar o código fonte, a gente vai ver que é justamente o código HTML que a gente tinha escrito no nosso arquivo no servidor.

Detalhes que são interessantes aqui da gente prestar atenção. A gente acabou de fazer um código tipicamente de servidor, a gente usou uma biblioteca que levanta um servidor HTTP, que é uma situação que só vai acontecer se você estiver fazendo uma aplicação no lado do servidor. A gente acabou de devolver um HTML, imagina que mais para frente a gente vai querer devolver um HTML dinâmico, típico código que é desenvolvido no lado do servidor.

Um outro detalhe para você começar a ficar atento, percebe que a gente teve que passar essa função como argumento para ela ser chamada pelo NodeJs quando a nova requisição chegar. O tempo inteiro a gente vai estar passando funções como argumento programando dentro do NodeJs. A plataforma que ele disponibiliza para a gente, as bibliotecas que ele disponibiliza para a gente funcionam muito com essas funções sendo passadas como argumento, e a gente vai discutir mais elas durante o treinamento.

Um ponto para a gente já ficar alerta também, talvez de má pratica aqui. Percebe que escrevemos o HTML inteiro dentro desse arquivo. E se fosse realmente uma página completa, com CSS, Javascript, imagens, fazendo código dinâmico... imagina o tamanho disso. É justamente esse ponto que a gente vai começar a atacar em seguida.