

Conteúdo

Módulo 2: Preparando o Ambiente

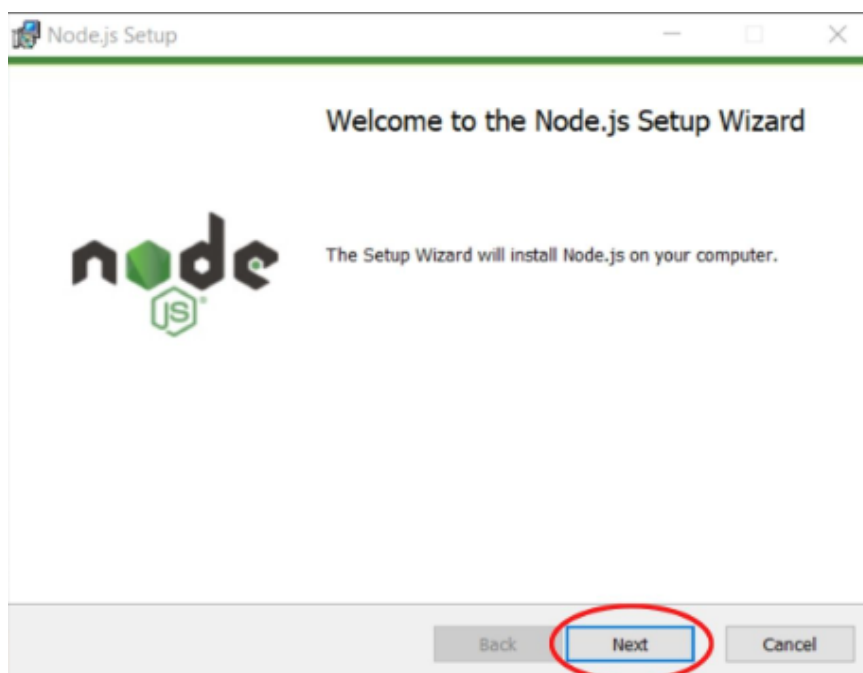
- Instalação Node.JS em Ambiente Windows
- Instalação Node.JS em Ambiente Mac OS
- Instalação Node.JS em Ambiente Linux
- Instalação Visual Studio Code (Editor de texto para desenvolvimento de aplicações web) no Windows
- O que é NPM?
- Prática sobre NPM
- Como funciona o NPM ?
- Atividades da OT

PRIMEIRO PASSO: INSTALAÇÃO DO NODEJS:

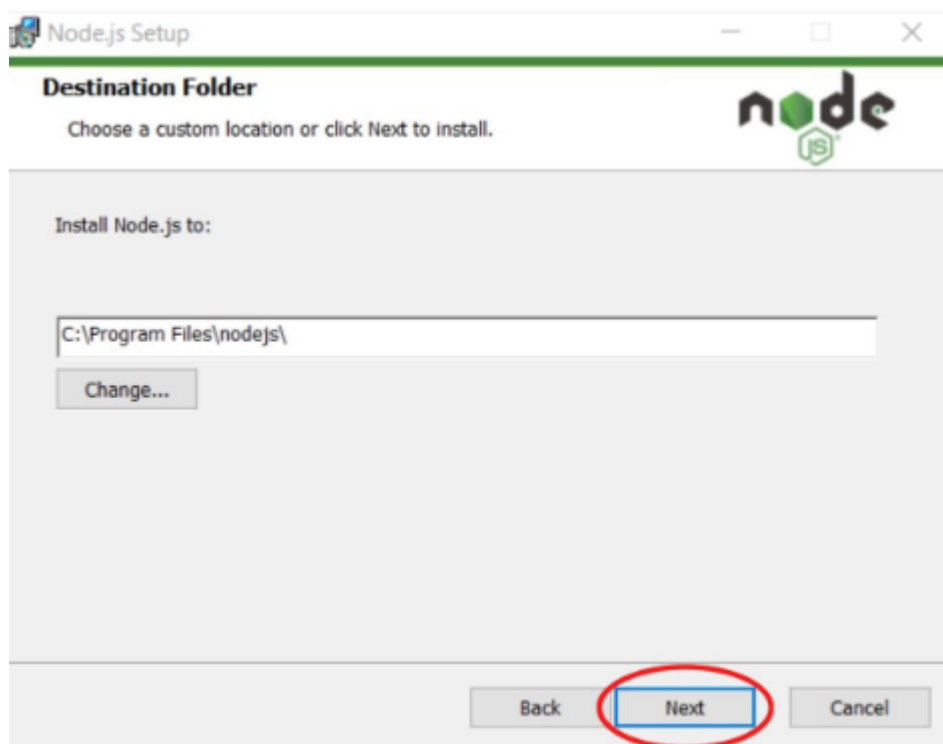
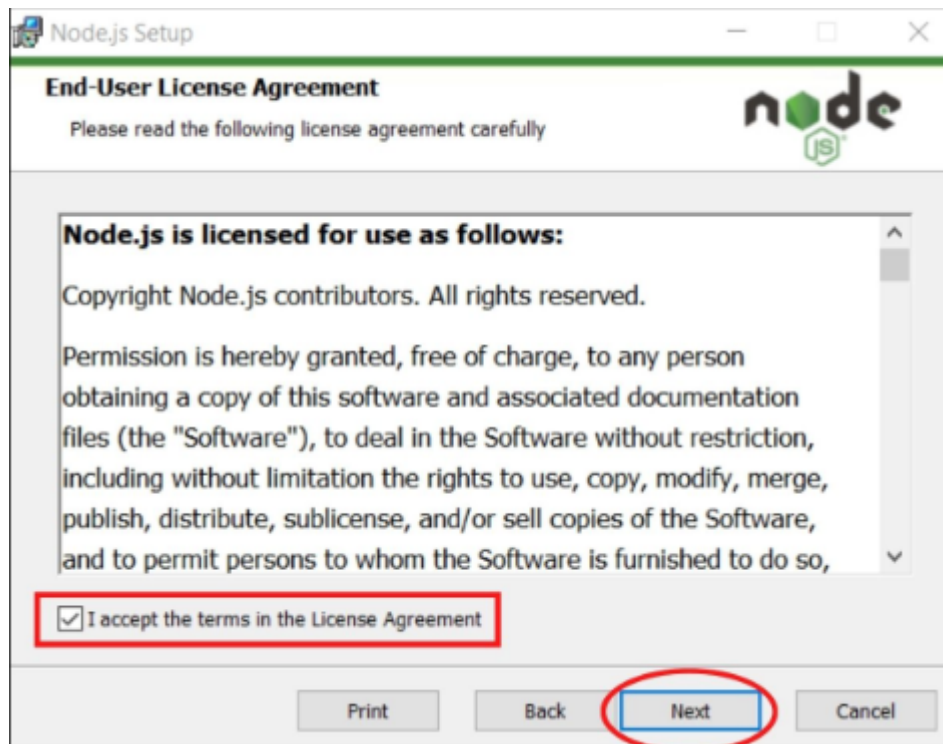
A primeira coisa que precisamos fazer é instalar o Node.js em nossa máquina. Para fazer isso, basta acessar a página oficial do Node e baixar a versão compatível com seu sistema operacional (instalaremos a versão LTS), como vemos na imagem abaixo.



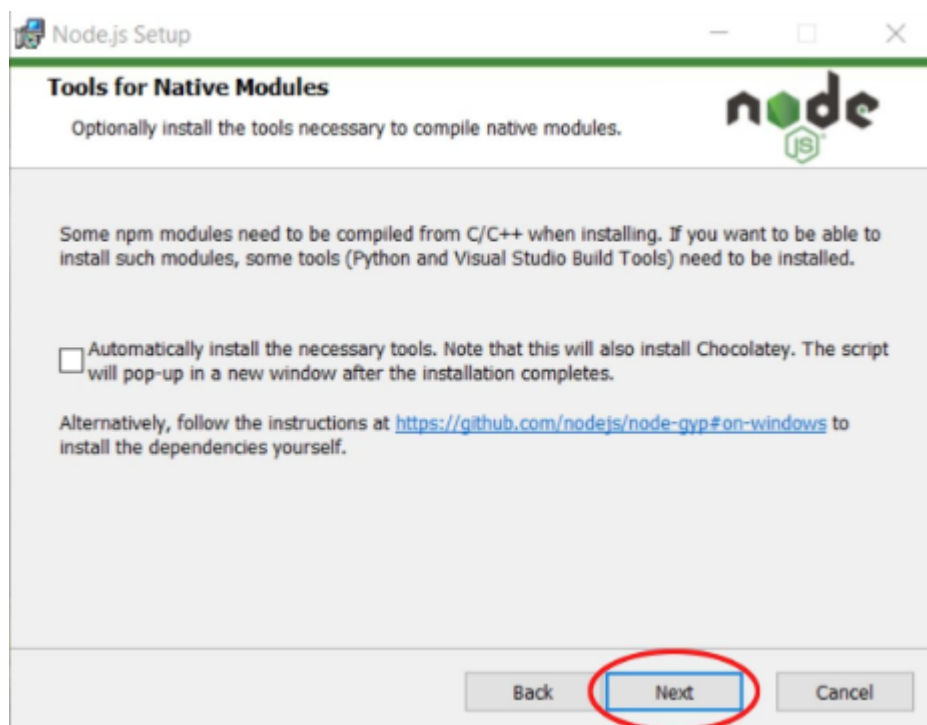
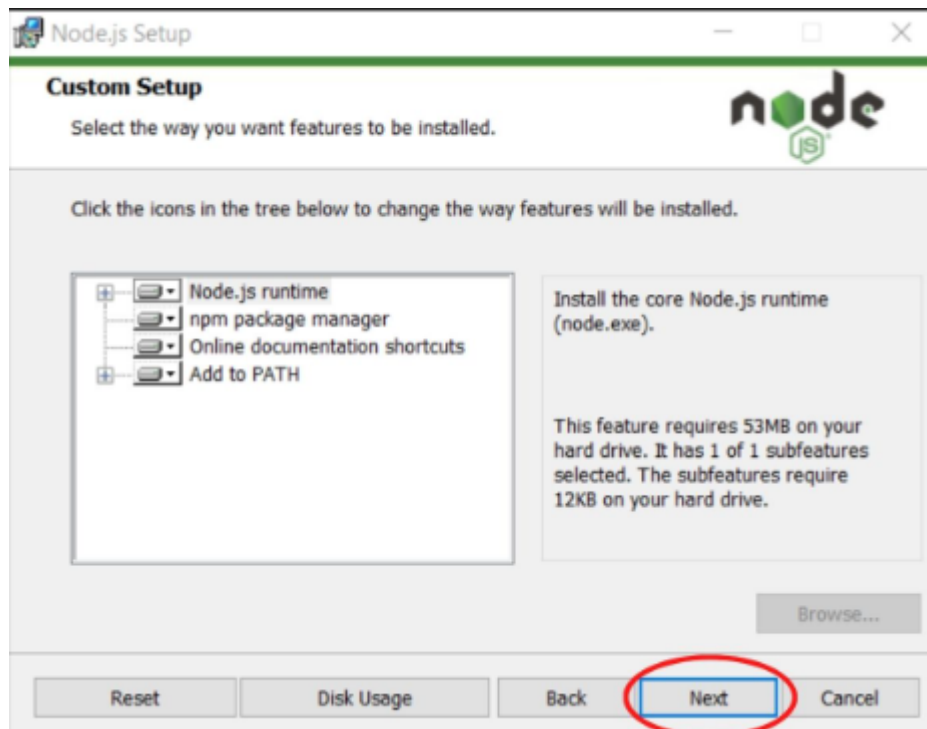
Podemos observar a instalação do Node no flow abaixo:



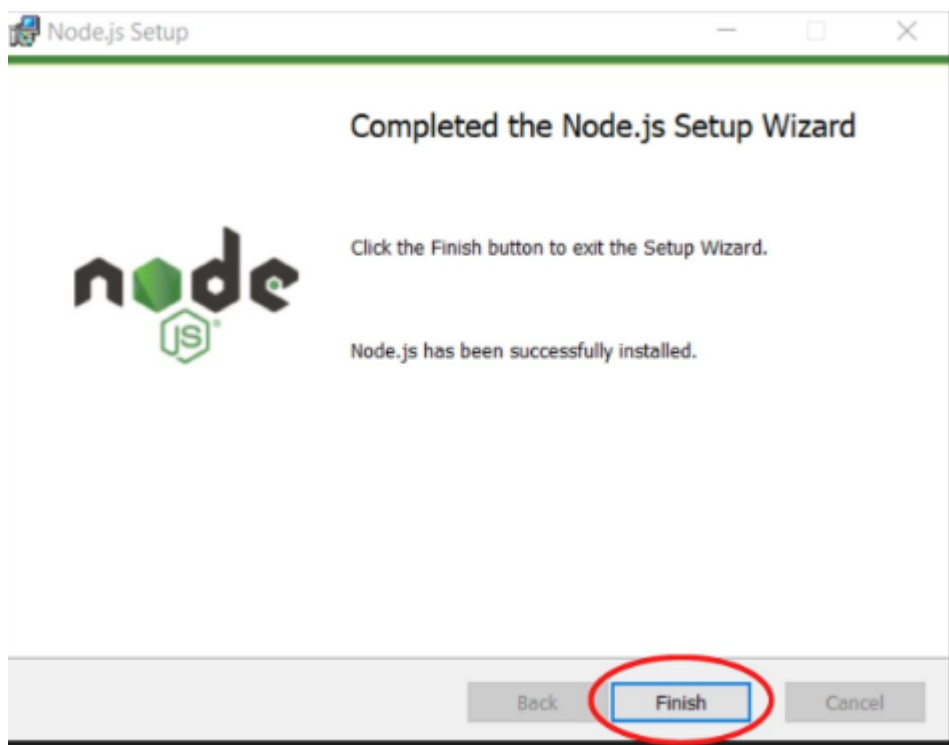
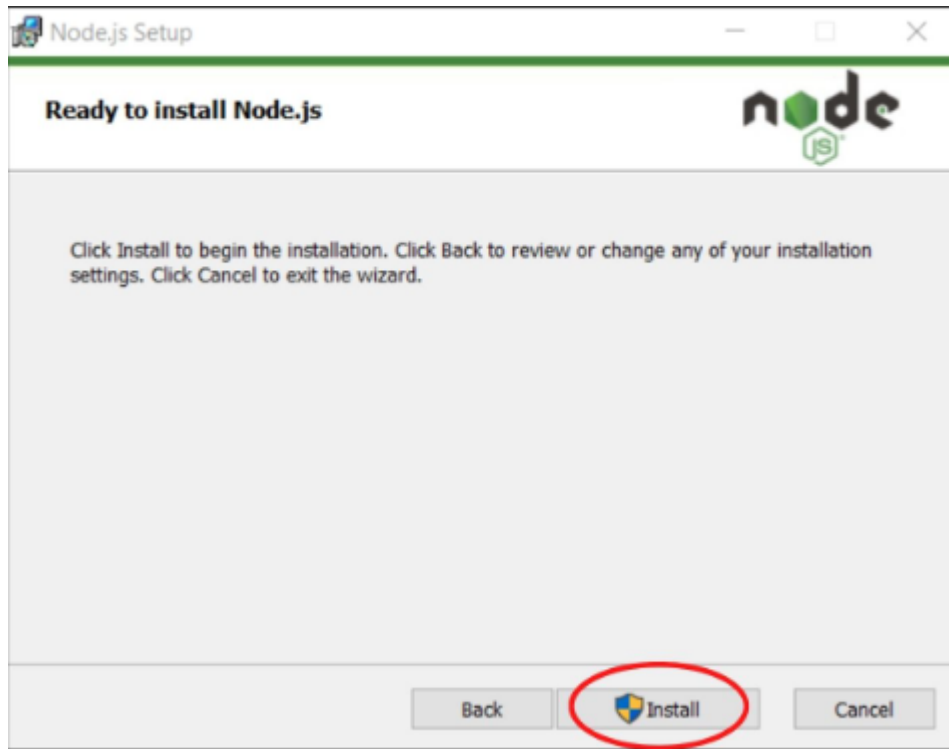
UniSENAI



UniSENAI



UniSENAI



Testando a instalação do Node

Para verificar se tudo correu bem durante a instalação, devemos verificar a versão do Node que está rodando no computador. Para isso vá ao terminal de comandos (prompt) e digite:

```
1 | node -v
```

CA Prompt de Comando

```
C:\Users>node -v
v18.12.0
C:\Users>
```

INSTALAÇÃO NO MAC OS

A instalação no macOS é bastante similar com a instalação do Windows, pois também usa um pacote de instalação.

Inicialmente, vamos até o site de download do Node. As versões LTS já estão disponíveis por padrão. Então, selecionamos a opção macOS Installer (.pkg), como indicado na imagem abaixo:

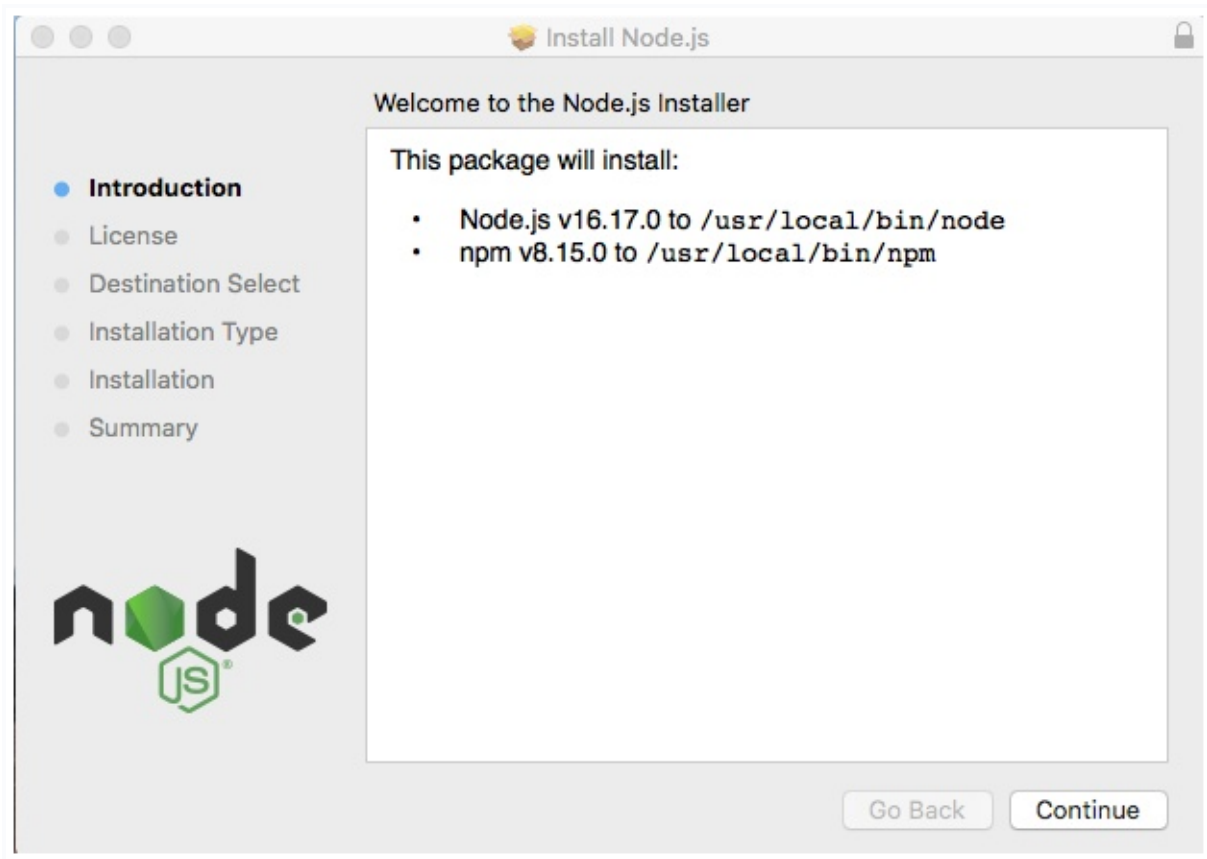
The screenshot shows the Node.js download page with two main tabs: 'LTS Recommended For Most Users' and 'Current Latest Features'. Under the 'LTS' tab, there are three options: 'Windows Installer' (node-v16.17.0-x86.msi), 'macOS Installer' (node-v16.17.0.pkg), and 'Source Code' (node-v16.17.0.tar.gz). Below these, there is a list of download options for the macOS Installer, including 'Windows Installer (.msi)', 'Windows Binary (.zip)', 'macOS Installer (.pkg)', 'macOS Binary (.tar.gz)', 'Linux Binaries (x64)', 'Linux Binaries (ARM)', and 'Source Code'. A table below the list shows the available architectures: 32-bit, 64-bit, 64-bit / ARM64 (highlighted with a red box), 64-bit, ARM64, 64-bit, ARMv7, and ARMv8. The '64-bit / ARM64' option is the one selected.

LTS Recommended For Most Users		Current Latest Features	
 Windows Installer node-v16.17.0-x86.msi		 macOS Installer node-v16.17.0.pkg	
 Source Code node-v16.17.0.tar.gz			

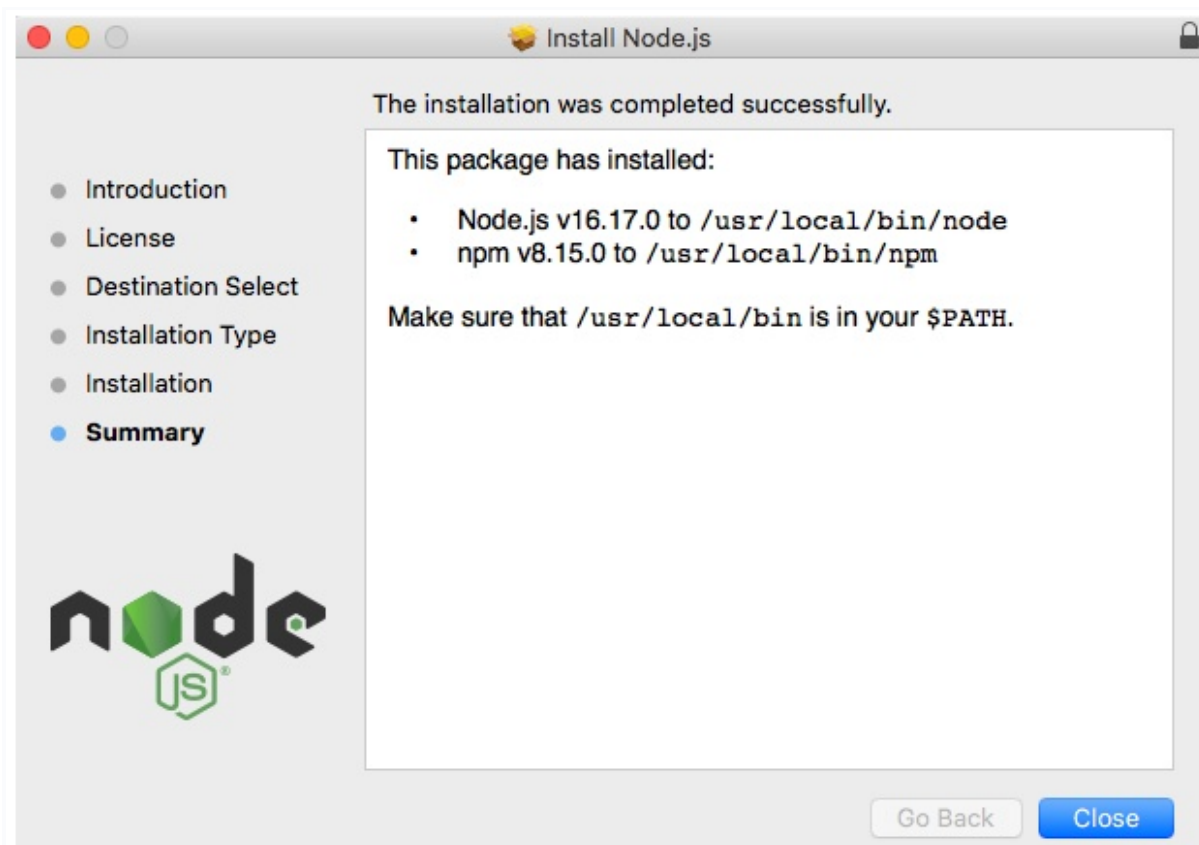
32-bit	64-bit
32-bit	64-bit
64-bit / ARM64	
64-bit	ARM64
64-bit	
ARMv7	ARMv8
node-v16.17.0.tar.gz	

Ao selecionarmos essa opção, o pacote de instalação será baixado. Assim que o download terminar, clicamos no arquivo baixado e executamos. Depois, podemos

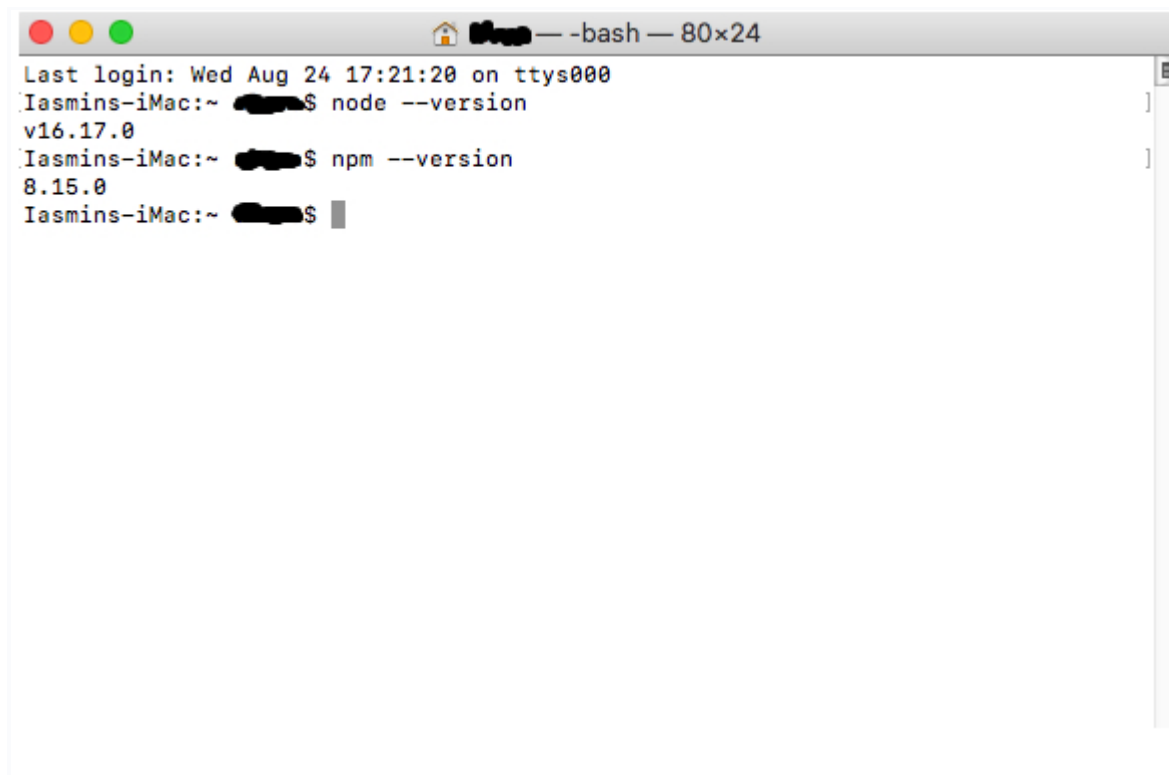
prosseguir ao clicar em *Continue*, sem nos esquecermos de aceitar os termos de uso.



Após clicar em Continue, teremos a seguinte tela:



Clique em *Close* e agora vamos conferir se o Node e o npm foram realmente instalados. Isso é feito de forma parecida com a verificação no Linux. Comece digitando `node --version` e `npm --version`. Vai aparecer o seguinte resultado:

A screenshot of a macOS terminal window. The title bar shows standard macOS window controls (red, yellow, green buttons) and a home icon followed by the text "— -bash — 80x24". The terminal content shows the login history "Last login: Wed Aug 24 17:21:20 on ttys000" and the user "Iasmins-iMac". The user runs the command "node --version" and receives the output "v16.17.0". Then, the user runs "npm --version" and receives the output "8.15.0". The prompt "Iasmins-iMac:~" is followed by a cursor.

```
Last login: Wed Aug 24 17:21:20 on ttys000
Iasmins-iMac:~ $ node --version
v16.17.0
Iasmins-iMac:~ $ npm --version
8.15.0
Iasmins-iMac:~ $
```

Esse resultado mostra que conseguimos concluir a instalação do Node e do npm.

INSTALAÇÃO NO LINUX (UBUNTU)

Para instalar a versão LTS no Linux Ubuntu, devemos digitar no terminal os seguintes comandos:

```
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E
bash -
```

```
sudo apt-get install -y nodejs
```

Depois rodarmos os comandos acima, a tela do terminal ficará assim:

```
Lendo informação de estado... Pronto
Os seguintes pacotes foram instalados automaticamente e já não são necessários:
bridge-utils docker-ce-rootless-extras docker-scan-plugin gyp
javascript-common libc-ares2 libfwupdplugin1 libjs-inherits
libjs-is-typedarray libjs-psl libjs-typedarray-to-buffer libpython2-stdlib
libpython2.7-minimal libpython2.7-stdlib libuv1-dev pigz
python-pkg-resources python2 python2-minimal python2.7 python2.7-minimal
slirp4netns ubuntu-fan
Utilize 'sudo apt autoremove' para os remover.
Os pacotes a seguir serão REMOVIDOS:
libnode-dev libnode64 nodejs-doc
Os NOVOS pacotes a seguir serão instalados:
nodejs
0 pacotes atualizados, 1 pacotes novos instalados, 3 a serem removidos e 92 não atualizados.
É preciso baixar 27,1 MB de arquivos.
Depois desta operação, 95,6 MB adicionais de espaço em disco serão usados.
Obter:1 https://deb.nodesource.com/node_16.x focal/main amd64 nodejs amd64 16.17.0-deb-1node
source1 [27,1 MB]
Baixados 27,1 MB em 4s (6.788 kB/s)
(Lendo banco de dados ... 238235 ficheiros e directórios actualmente instalados.)
A remover libnode-dev:amd64 (10.19.0-dfsg-3ubuntu1) ...
A remover libnode64:amd64 (10.19.0-dfsg-3ubuntu1) ...
A remover nodejs-doc (10.19.0-dfsg-3ubuntu1) ...
A seleccionar pacote anteriormente não seleccionado nodejs.
(Lendo banco de dados ... 238001 ficheiros e directórios actualmente instalados.)
A preparar para desempacotar .../nodejs_16.17.0-deb-1nodesource1_amd64.deb ...
A descompactar nodejs (16.17.0-deb-1nodesource1) ...
Configurando nodejs (16.17.0-deb-1nodesource1) ...
A processar 'triggers' para libc-bin (2.31-0ubuntu9.9) ...
A processar 'triggers' para man-db (2.9.1-1) ...
```

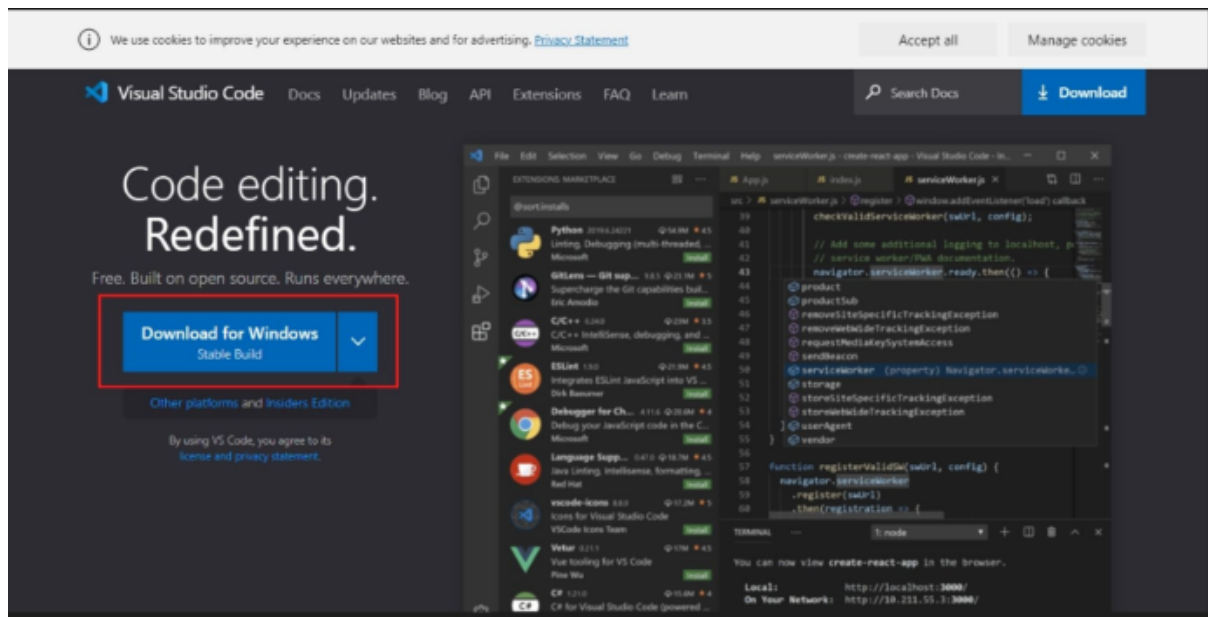
Feito isso, o Node.js já está instalado! Para confirmar que deu tudo certo com a instalação, digite o comando `node --version`. Já para verificar a instalação do npm, que é o gerenciador de pacotes do Node, que é baixado no Linux junto com ele, digite o comando: `npm --version`. Feito isso, vai aparecer algo parecido com isso:

```
iasmin@iasmin-All-Series:~$ node --version
v16.17.0
iasmin@iasmin-All-Series:~$ npm --version
8.15.0
```

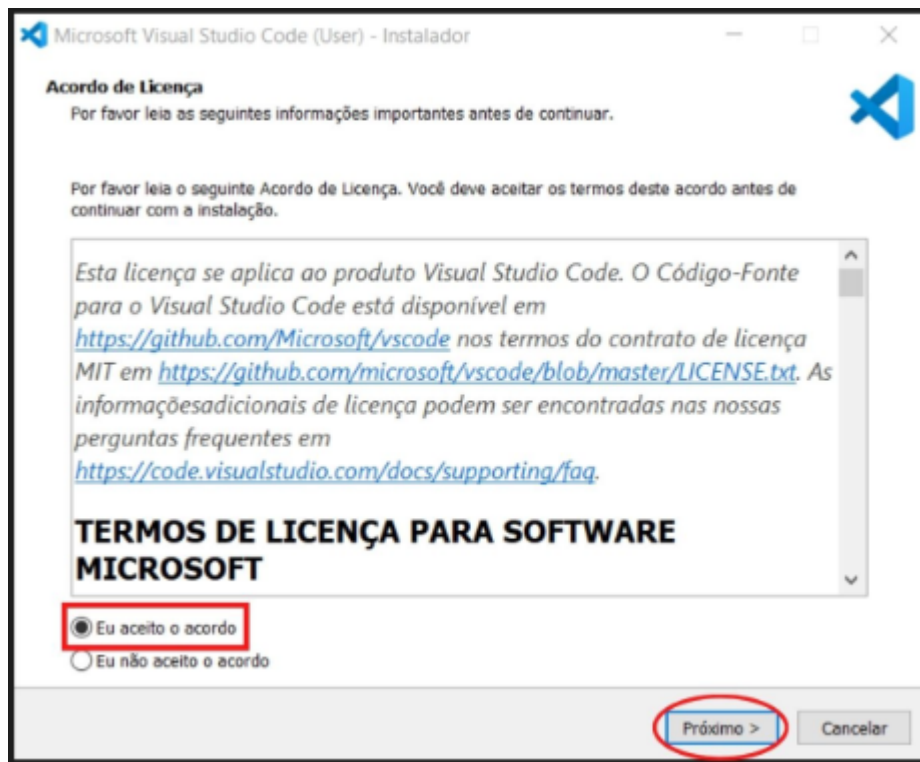
Segundo passo: Instalação do Visual Studio Code

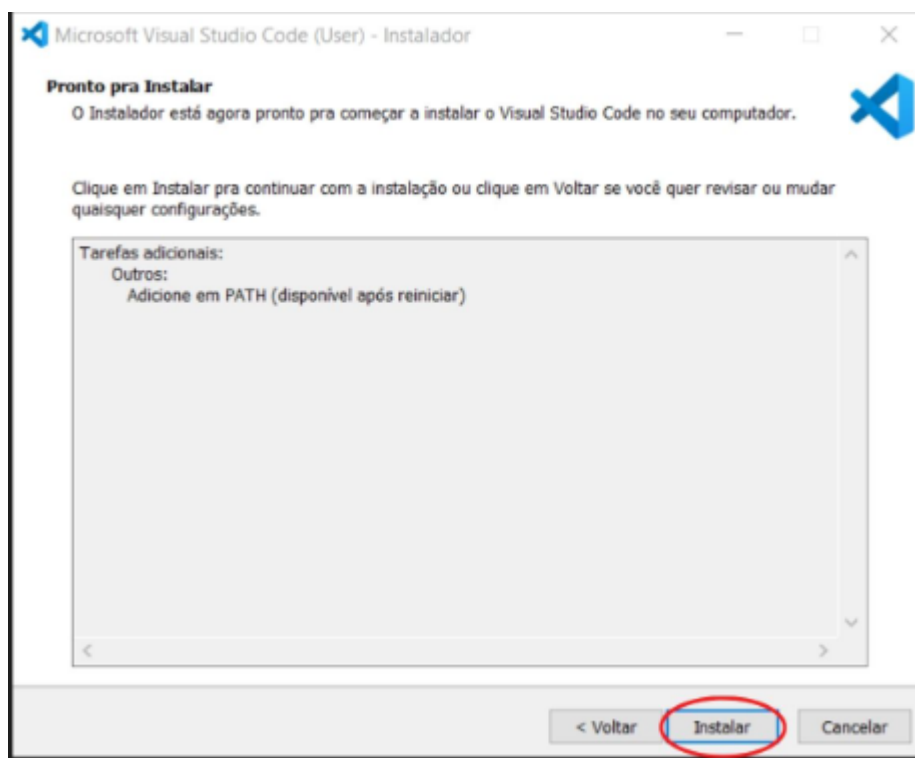
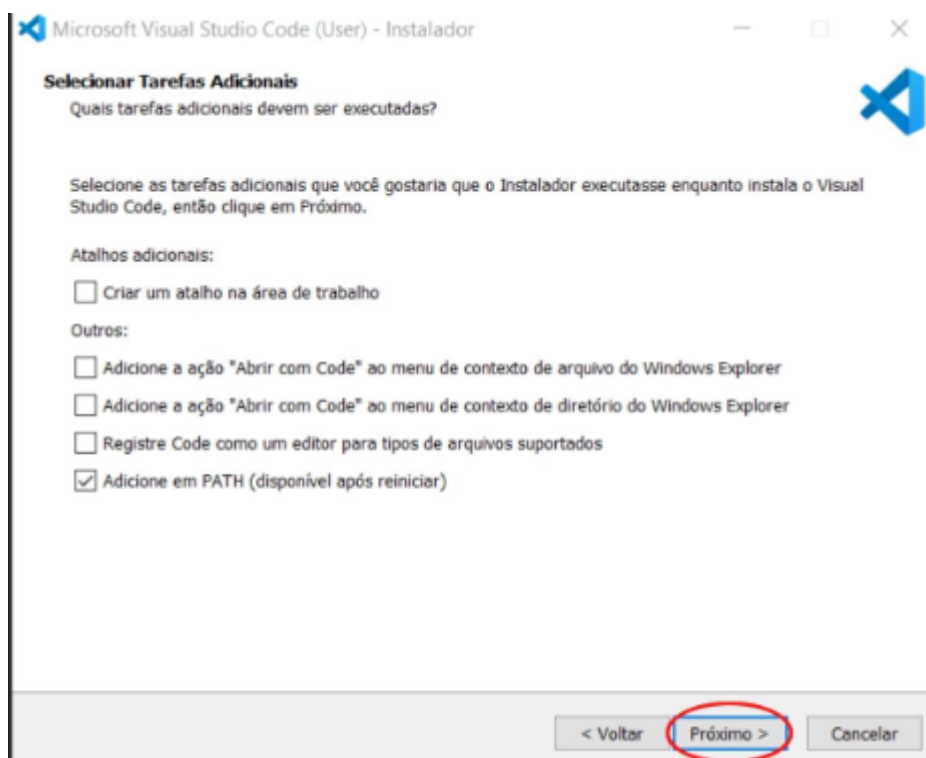
Para codificarmos vamos precisar de um editor de texto. Você pode ficar livre para utilizar o editor de sua preferência, mas neste curso utilizaremos o Visual Studio Code, que pode ser baixado na sua página oficial, como vemos na imagem abaixo.

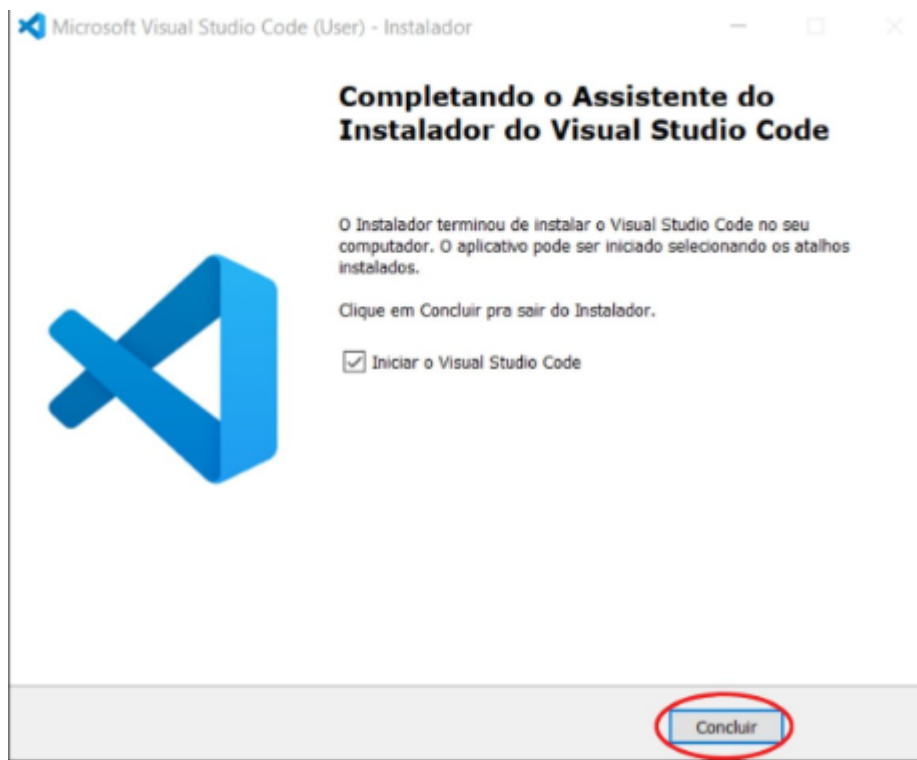
Unisenai



Podemos observar a instalação do Visual Studio Code no flow a seguir:







Node e o NPM

O NPM (Node Package Manager) é um gerenciador de dependências. O NPM é instalado automaticamente com a instalação do NODEJS.

Para entender melhor o papel de um gerenciador de dependências (NPM), vamos compreender o que são **dependências**.

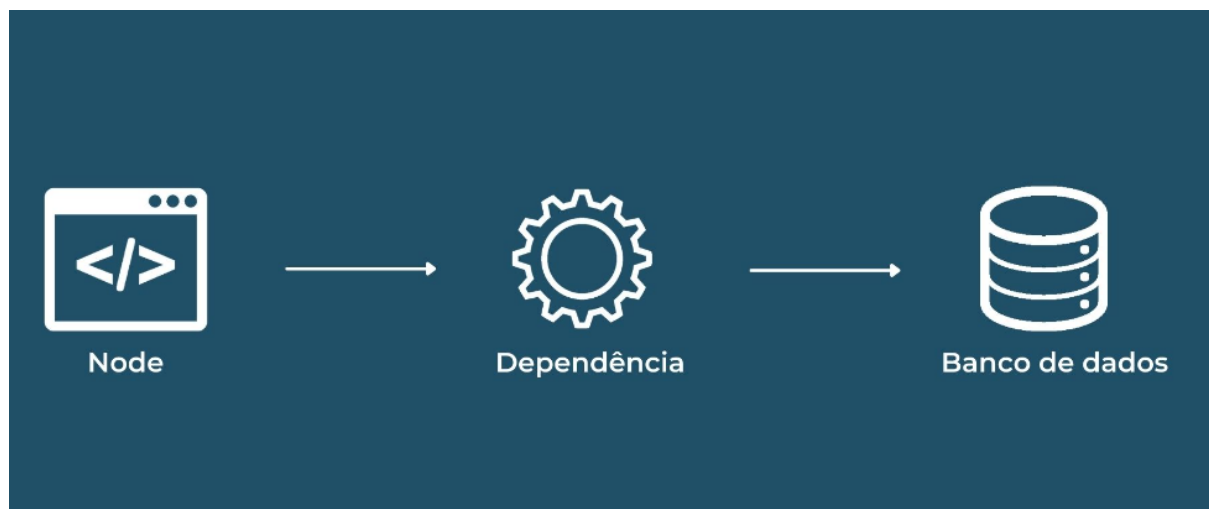
Dependências, o que são?

O **NodeJS** possui vários recursos que nos auxiliam no desenvolvimento, mas ao concluirmos sua instalação ele vem com seus recursos básicos.

Com o tempo nossa aplicação vai crescendo e os recursos básicos passam a não ser suficientes e com isso vamos precisar instalar novos recursos. Estes novos recursos que serão instalados no futuro são **dependências**.

Por exemplo: a **instalação básica** do **Node** não inclui ferramentas de conexão com banco de dados. Quando nossa aplicação precisar acessar algum banco de dados, vamos precisar instalar um módulo para isso, ou seja uma **dependência**.

Dependências são bibliotecas externas necessárias para o funcionamento da aplicação.



Gerenciador de Dependência, o que é?

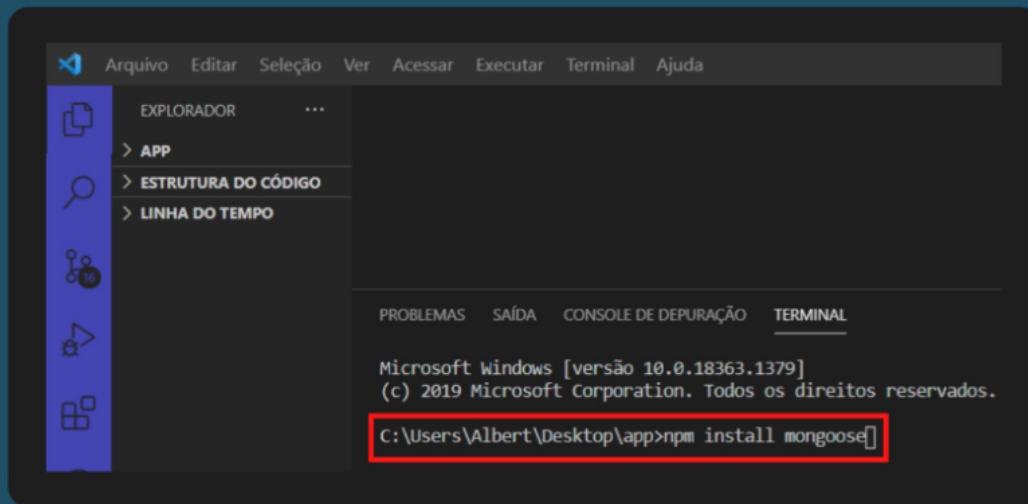
Um **gerenciador de dependências**, no nosso caso o **(NPM)** automatiza a instalação de novas dependências na aplicação.

O processo manual de instalação de dependências daria bastante trabalho, principalmente quando o número de dependências da aplicação vai crescendo.

O **NPM** nos poupa este trabalho, pois com apenas um comando ele instala a dependência solicitada no nosso projeto.

A seguir veremos um exemplo onde está sendo feita a instalação de uma dependência mongoose, ela permite que a nossa aplicação acesse um banco de dados.

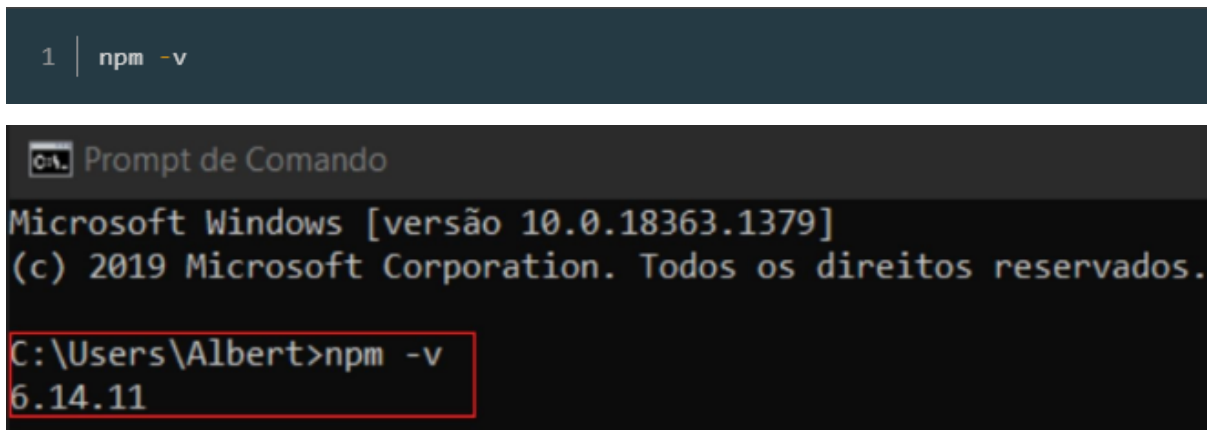
Exemplo de uso do NPM



O Node e o NPM são como unha e carne, você vai precisar do NPM o tempo todo durante sua carreira como programador Node.

Verificando a instalação do NPM

Assim como verificamos a instalação do Node.js, podemos fazer o mesmo para o NPM, basta digitarmos o comando abaixo no prompt:





Nesta aula conhecemos e instalamos as ferramentas para o nosso ambiente de desenvolvimento, aprendemos o que é um gerenciador de dependências e vimos como verificar a instalação do Node e do NPM

Agora iremos iniciar uma atividade passo a passo sobre instalação de pacotes Node.JS, utilizando o npm, seu principal gerenciador de pacotes como vimos anteriormente.

COMO IMPORTAR OS MÓDULOS ESSENCIAIS DO NODE?

Então, vamos começar com um pacote bem básico, que é o fs (sistema de arquivos). Você o usa para **criar**, **ler** e **modificar arquivos**.

Para importar o módulo fs, insira este comando no arquivo:

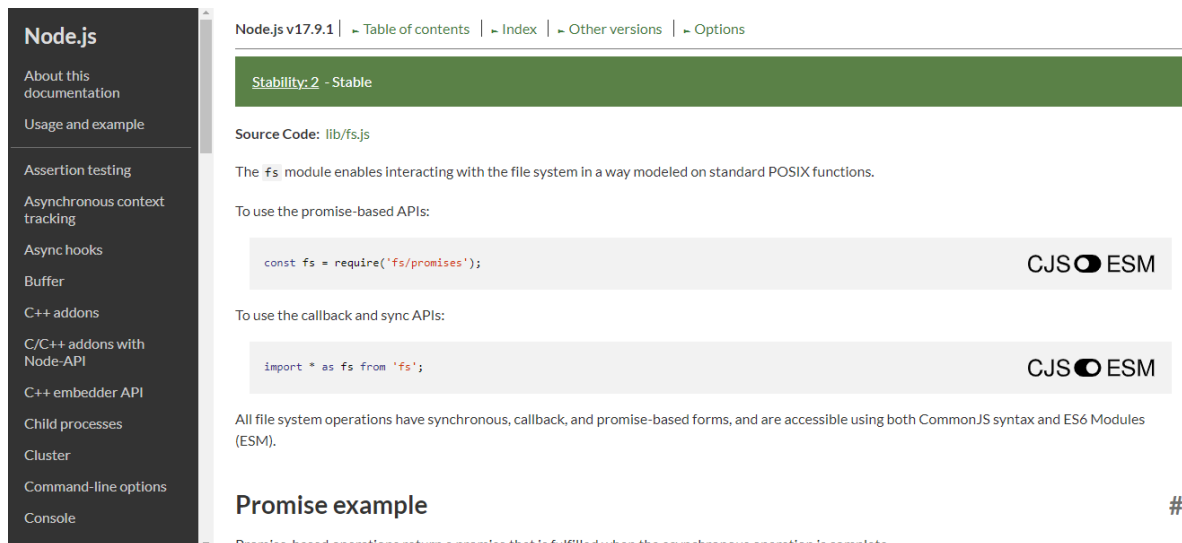
```
const fs = require("fs");
```

Agora, você pode usar qualquer função desse módulo. Para saber quais são, basta consultar a documentação (em inglês).

Link da url da documentação referente ao fs:

<https://nodejs.org/docs/latest-v17.x/api/fs.html#file-system>

Unisenai



The screenshot shows the Node.js documentation for the `fs` module. The left sidebar lists various Node.js topics, with `fs` selected. The main content area shows the `fs` module's stability (Stable), source code location (`lib/fs.js`), and a description of its purpose. It provides examples for using the module with both CommonJS (`require`) and ES Modules (`import`). A 'Promise example' section is also visible at the bottom.

Não esqueçam de criar a pasta do projeto:

Carlos						Pesquisar em Carlos
	Nome	Status	Data de modificação	Tipo	Tamanho	
	Projeto_File	✓	02/08/2023 20:27	Pasta de arquivos		

E logo após:

Por exemplo, para criar um arquivo, podemos usar a função:

```
fs.writeFileSync(filename, content);
```

O código ficará assim:

```
var fs = require('fs');
fs.appendFile(novo.txt', 'Olá NodeJS! SENAI', function (err) {
  if (err) throw err;
  console.log(Arquivo Salvo!);
});
```

Unisenai

```
JS lerArquivo.js X
JS lerArquivo.js > ...
1  var fs = require('fs');
2
3  fs.appendFile('novo.txt', 'Olá NodeJS! SENAI', function (err) {
4    if (err) throw err;
5
6    console.log('Arquivo Salvo!');
7  })
```

Ao realizar a criação do arquivo também dentro de sua pasta com seu Nome, execute o comando node + o nome do arquivo no caso = lerArquivo.js

node lerArquivo.js

Ao executar note que ele exibe a mensagem que colocamos no console.log: e cria o arquivo novo.txt dentro do diretório atual.

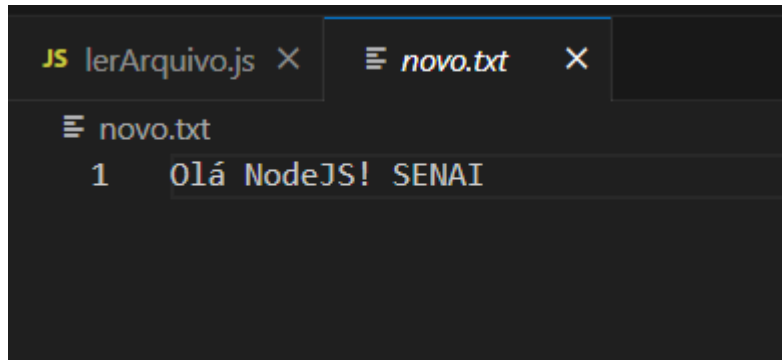
```
JS lerArquivo.js X
JS lerArquivo.js > ...
1  var fs = require('fs');
2
3  fs.appendFile('novo.txt', 'Olá NodeJS! SENAI', function (err) {
4    if (err) throw err;
5
6    console.log('Arquivo Salvo!');
7  })
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL powershell

PS C:\Users\lenni\OneDrive\Área de Trabalho\Carlos\Projeto_File> node lerArquivo.js
Arquivo Salvo!

```
JS lerArquivo.js
≡ novo.txt
```

Ao acessarmos o arquivo novo.txt vem o que solicitamos que fosse escrito:



```
JS lerArquivo.js X novo.txt X
novo.txt
1 Olá NodeJS! SENAI
```

Muito bom , Parabéns você conseguiu atingir a meta, agora vamos dificultar?

Procure o código na documentação do Node.JS para atualizar o arquivo que vc criou e atualize ele com o seguinte texto “ UNISENAI 2023”, o texto deve ficar assim :

“Olá Node.JS! UNISENAI 2023”

e após concluir mais esta etapa renomeie o nome do arquivo criado, procure como realizar a mesma situação, o novo nome do arquivo deve ficar:

ArquivoNovoRenomeado.txt

após concluir isso ,procure como deletar o arquivo e crie o comando , rode e depois crie outro arquivo novamente, para deixar na pasta e eu possa corrigir enviando para o github, para o mesmo repositório da aula passada sobre Node.JS.

Agora vamos para um próximo projeto:

Aprendendo a instalar Pacotes NPM

COMO INSTALAR PACOTES NPM

Como exemplo, vamos usar um pacote do npm muito simples, chamado **superheroes** (que é uma lista de super-heróis aleatórios) para ajudar você a entender como o npm funciona.

Para instalar qualquer pacote do npm, podemos usar este comando no terminal:

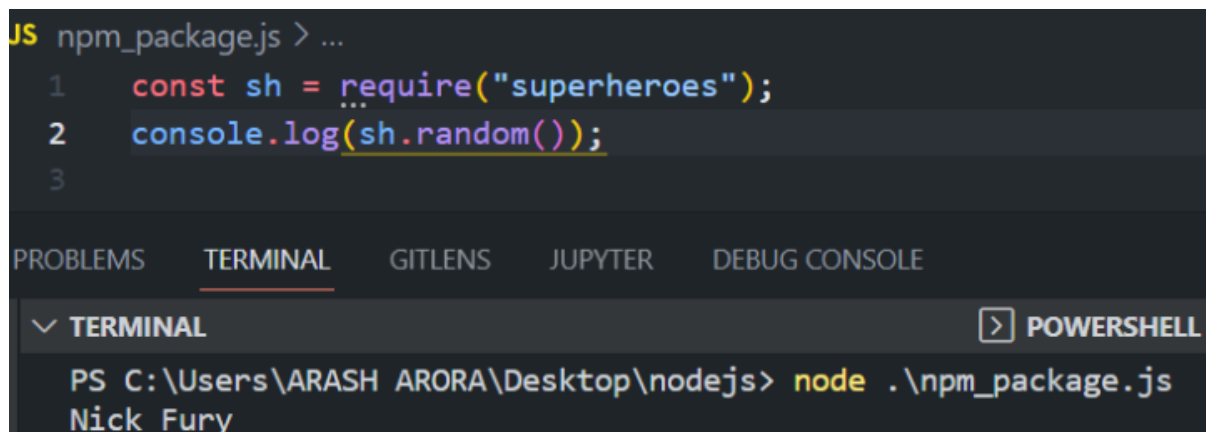
```
npm install superheroes
```

Para importar o pacote que acabamos de instalar, digitamos isto no arquivo:

```
const sh = require("superheroes");
```

Usamos o comando abaixo para exibir o nome de um super-herói aleatório:

```
console.log(sh.random());
```



The screenshot shows a code editor with a dark theme. The top part shows a JavaScript file named 'npm_package.js' with the following code:

```
JS npm_package.js > ...
1  const sh = require("superheroes");
2  console.log(sh.random());
3
```

Below the code editor, there is a panel with tabs for 'PROBLEMS', 'TERMINAL', 'GITLENS', 'JUPYTER', and 'DEBUG CONSOLE'. The 'TERMINAL' tab is selected, showing a PowerShell window. The command prompt shows the command 'node .\npm_package.js' being executed, and the output is 'Nick Fury'.

Vamos tentar outro pacote. Instalaremos um dos pacotes do npm mais usados, chamado "chalk" — ele estiliza strings de texto no terminal.

Para instalar o pacote chalk (utilizaremos a versão 2.4.2, pois ela permite importar o pacote usando o método require), digite este comando:

```
npm install chalk@2.4.2
```

Agora, para estilizar uma string de texto, use este comando para escolher a cor da string (o nome da cor precisa estar em inglês):

```
chalk.cor(texto)
```

```
// Exemplo: chalk.blue("Este é o texto que ficará em azul.");
```



The screenshot shows the Visual Studio Code interface. The editor window displays a file named `npm_package.js` with the following code:

```
JS npm_package.js > ...  
1  const sh = require("superheroes");  
2  const chalk = require("chalk");  
3  console.log(chalk.blue(sh.random()));  
4
```

Below the editor, the **TERMINAL** tab is active, showing a PowerShell session. The command `node .\npm_package.js` has been executed, resulting in the output `Clover` displayed in blue text.

Para saber mais sobre a Biblioteca Chalk -> <https://www.npmjs.com/package/chalk>

COMO INICIAR O NPM EM NOSSO PROGRAMA

Iniciamos o NPM em nosso programa com:

```
npm init
```

Em seguida, você pode responder às perguntas sobre informações do projeto, ou pressionar Enter em todas elas.

UniSENAI

```
PS C:\Users\ARASH ARORA\Desktop\nodejs\initiatenpm> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

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

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

Press ^C at any time to quit.
package name: (initiatenpm)
version: (1.0.0)
description:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\ARASH ARORA\Desktop\nodejs\initiatenpm\package.json:

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

Is this OK? (yes)
```

Se preferir usar diretamente o comando `npm init -y` (que é o mesmo que pressionar Enter para todas as perguntas).

```
PS C:\Users\ARASH ARORA\Desktop\nodejs\initiatenpm> npm init -y
Wrote to C:\Users\ARASH ARORA\Desktop\nodejs\initiatenpm\package.json:

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

Isso resultará na criação do arquivo **package.json**:

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

O que faz, afinal, o package.json?

O `package.json` é o coração de qualquer projeto do Node.js. É nele que temos o registro de todas as dependências (os pacotes do NPM) e os metadados de cada projeto.

Se outra pessoa baixar o projeto, este arquivo vai ajudá-lo a instalar todas as dependências necessárias para executar o programa.

Como usar o Moment.js — um pacote do NPM

A biblioteca Moment.js é um dos pacotes do npm mais usados do Node. Com ele, você pode analisar e validar datas.

Para instalar o pacote, execute este comando no terminal:

```
npm i moment
```

Percebeu o comando `i`? Ele é a abreviação do `install` que usamos no início. Você pode usar qualquer um dos dois.

Importamos o pacote no nosso código:

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

Criamos um objeto `Date`, buscando a data e hora atual (método nativo do JavaScript):

```
const time = new Date();
```

Unisenai

Agora, para analisar ou formatar essa data, usaremos o pacote **moment**:

```
const parsedTime = moment(time).format("h:mm:ss");
```

Exibimos no console, então, o tempo analisado:

```
console.log(parsedTime);
```

```
1  const moment = require("moment");
2  const time = new Date();
3  const parsedTime = moment(time).format("h:mm:ss");
4  console.log(parsedTime);
```

PROBLEMS TERMINAL GITLENS JUPYTER DEBUG CONSOLE

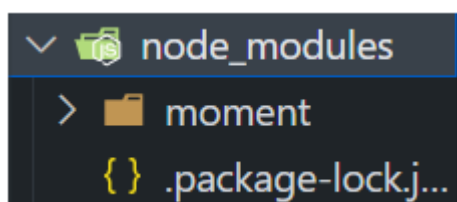
✓ TERMINAL

```
PS C:\Users\ARASH ARORA\Desktop\nodejs\initiatenpm> node .\app.js
10:55:16
```

Abaixo vemos o package.json do projeto, com todos os pacotes de dependências – neste caso, apenas o **moment** .

```
package.json X
initatenpm > package.json > ...
1  {
2    "name": "initatenpm",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "moment": "^2.29.3"
14   }
15 }
```

Também temos a pasta **node_modules** na pasta do projeto. Essa pasta contém todas as dependências das quais nosso projeto depende, incluindo o moment e outros pacotes dos quais ele depende.



O **package-lock.json** é outro arquivo em nossa pasta do projeto que contém todas as informações sobre o nome, dependências, versão das dependências e se existe alguma versão que esteja bloqueada no projeto.

Ele descreve a árvore de dependências exata que foi gerada, permitindo que qualquer nova instalação tenha uma árvore idêntica.

```
package-lock.json X
initiatenpm > package-lockjson > ...
1 {
2   "name": "initiatenpm",
3   "version": "1.0.0",
4   "lockfileVersion": 2,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "initiatenpm",
9       "version": "1.0.0",
10      "license": "ISC",
11      "dependencies": {
12        "moment": "^2.29.3"
13      }
14    },
15    "node_modules/moment": {
16      "version": "2.29.3",
17      "resolved": "https://registry.npmjs.org/moment/-/moment-2.29.3.tgz",
18      "integrity": "sha512-c6YRvhEo//6T2Jz/vVtYzqBzwvPT95JBQ+smCytzf7c50oMZRsr/a4w88aD34I+/QVSfnoAnS8FPJHIt10MJVw==",
19      "engines": {
20        "node": "*"
21      }
22    }
23  },
24  "dependencies": {
25    "moment": {
26      "version": "2.29.3",
27      "resolved": "https://registry.npmjs.org/moment/-/moment-2.29.3.tgz",
28      "integrity": "sha512-c6YRvhEo//6T2Jz/vVtYzqBzwvPT95JBQ+smCytzf7c50oMZRsr/a4w88aD34I+/QVSfnoAnS8FPJHIt10MJVw=="
29    }
30  }
}
```

Ao término das atividades enviar para a sua pasta no repositório do github sobre Node.Js que foi criado.

<https://github.com/uchoamaster/SENAI-NODE-JS>

UniSENAI

94%