

FIAP

FIAP

SLIDER ▢■◀





Node.js



O que é Node.js

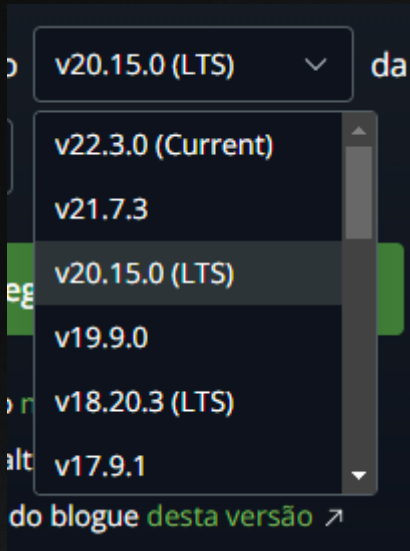
- O Node.js é um ambiente de execução Javascript fora do Browser, sendo assim possível aproveitar todos os recursos da linguagem no lado do servidor, sem depender do navegador para executar
- Uma das vantagens mais fortes do Javascript desempenhando estas tarefas é o seu trabalho assíncrono, que chamamos de “Node single thread”, onde as requisições não ficam esperando a resposta para continuar o trabalho, ou seja, não alocam uma linha de memória exclusiva para executar o serviço de cada requisição. Isso faz com que os serviços com grande quantidades de trocas de mensagem como o Web fiquem muito mais leves e performáticos, não necessitando de recursos tão robustos para estas tarefas.
- Não vamos entrar a fundo no Node, mas precisamos dele para criar aplicações utilizando o React, por isso, vamos a instalação.

Instalação

Windows

- Instalador

1. Acessar <https://nodejs.org/pt/download/prebuilt-installer>
2. Recomendado sempre baixar a versão LTS (Long Time Support) mais atual



Descarregar a Node.js®

Podemos descarregar a Node.js como quisermos.

Gestor de Pacote Instalador do Pré-compilado Binários Pré-compilados Código Fonte

Quero a versão da Node.js para executando sobre

 Descarregue o Node.js v20.15.0

A Node.js inclui o npm (10.7.0) ↗.

Ler o registo de alterações [desta versão](#) ↗

Ler a publicação do blogue [desta versão](#) ↗

Saber como verificar o SHASUMS assinado ↗

Consultar todas as opções de descarga disponíveis ↗ da Node.js

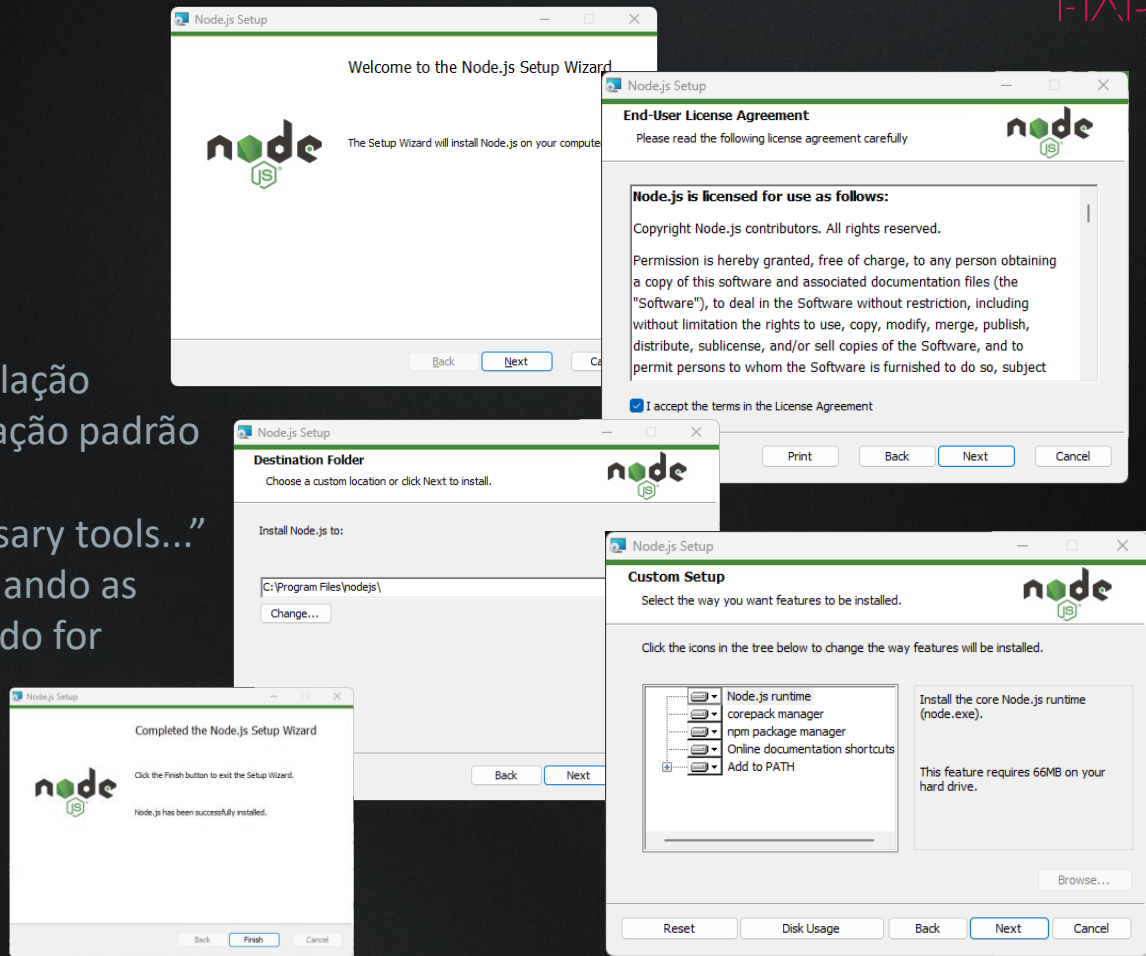
Saber mais sobre os Lançamentos da Node.js ↗

Instalação

Windows

- Instalador

3. Abrir o instalador
4. Aceitar os termos
5. Evite mudar o local de instalação
6. Manter as opções de instalação padrão
7. Não marcar a caixa
“Automatically install the necessary tools...”
8. Prosseguir com a instalação dando as permissões administrador quando for solicitado

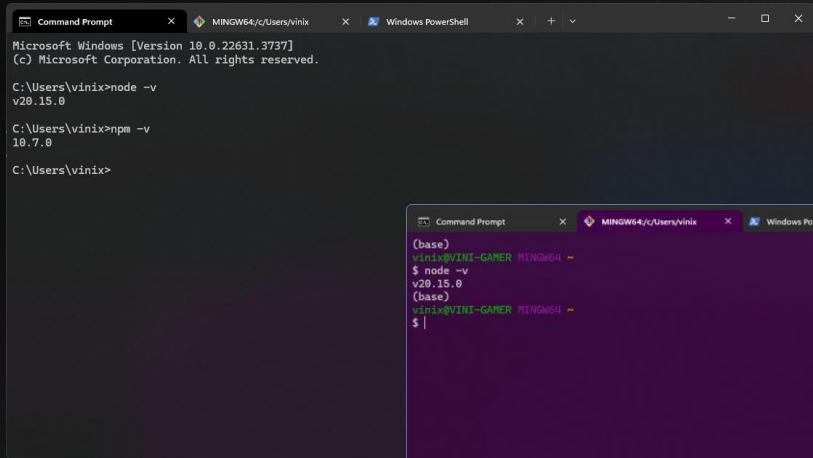


Instalação

Windows

- Instalador

Para testar se tudo ocorreu como deveria, basta abrirmos o CMD ou o PowerShell ou o GitBash ou qualquer outro terminal instalado no Windows e digitar o comando “**node -v**” ou “**npm -v**”

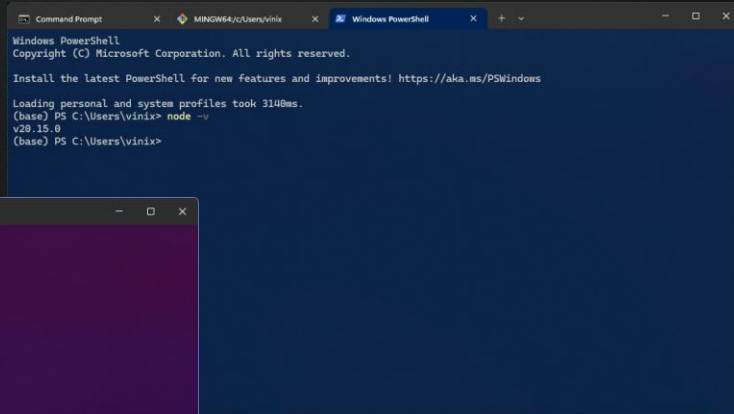


```
Command Prompt
Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vinix>node -v
v20.15.0

C:\Users\vinix>npm -v
10.7.0

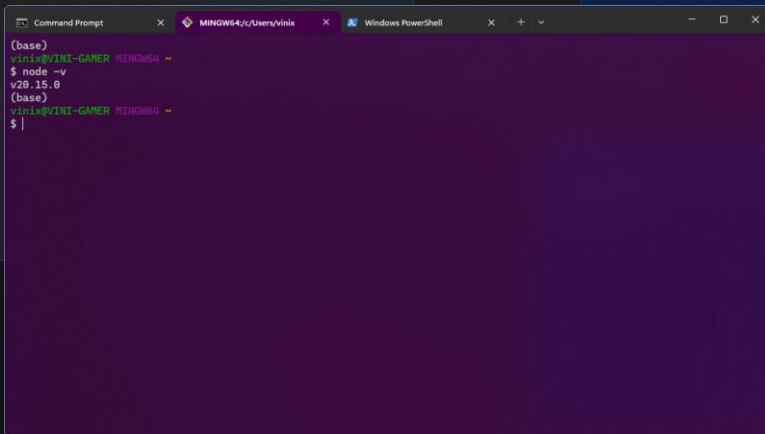
C:\Users\vinix>
```



```
Command Prompt
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Loading personal and system profiles took 3140ms.
(base) PS C:\Users\vinix> node -v
v20.15.0
(base) PS C:\Users\vinix>
```



```
Command Prompt
(base)
vinix@VINI-GAMER MINGW64 ~
$ node -v
v20.15.0
(base)
vinix@VINI-GAMER MINGW64 ~
$ |
```

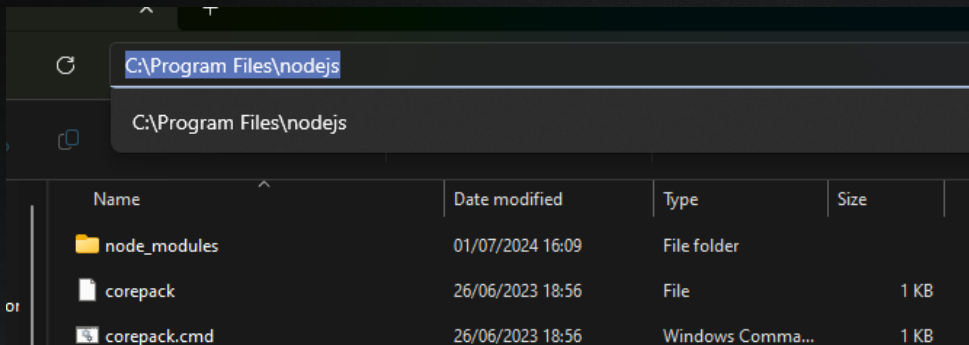
Instalação

Windows

Possíveis problemas – Criando a variável de ambiente manualmente

Caso o comando não funcione mesmo com o instalador dando sucesso, pode ser o caso de uma falha na criação da variável de ambiente do node na máquina. Para corrigir esse problema no Windows, precisamos criar essa variável manualmente.

1. Encontre a pasta em que você instalou o seu node. No meu caso ficou em “C:\Program Files\nodejs”



Instalação

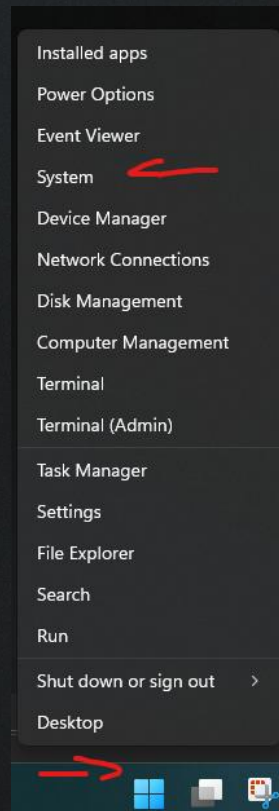
Windows

Possíveis problemas – Criando a variável de ambiente manualmente

2. Encontre a pasta em que você instalou o seu node.

No meu caso ficou em “C:\Program Files\nodejs”

3. Clique com o direito no ícone do Windows do menu iniciar e clique em Sistema

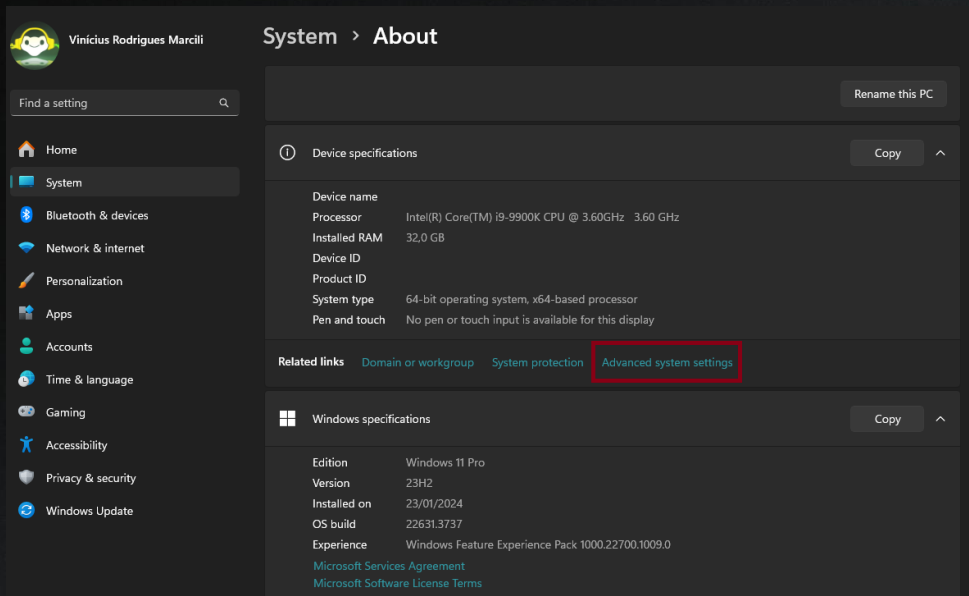


Instalação

Windows

Possíveis problemas – Criando a variável de ambiente manualmente

4. Clique em “Configurações avançadas do sistema”

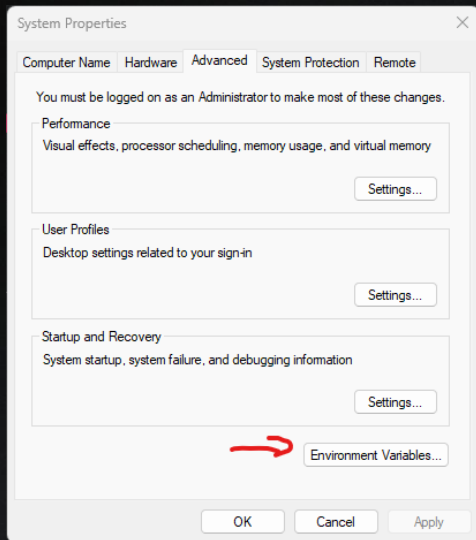


Instalação

Windows

Possíveis problemas – Criando a variável de ambiente manualmente

5. Clique em “Variáveis de Ambiente”

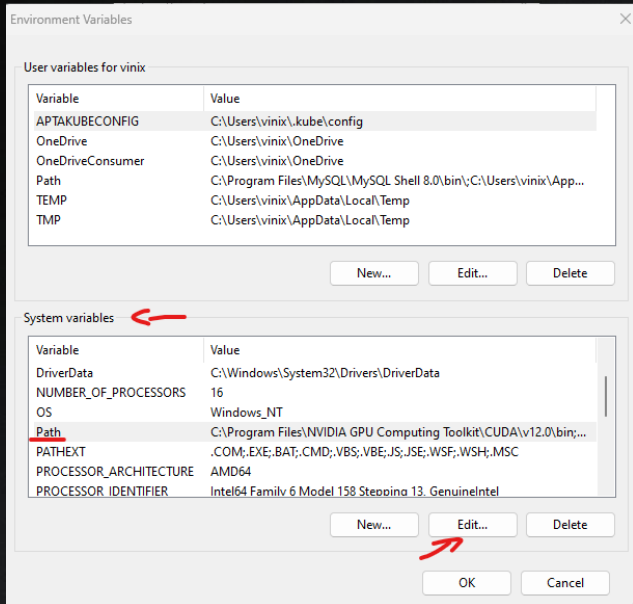


Instalação

Windows

Possíveis problemas – Criando a variável de ambiente manualmente

6. Na parte inferior da janela (Variáveis do Sistema), procure por “PATH” e clique em “Editar”

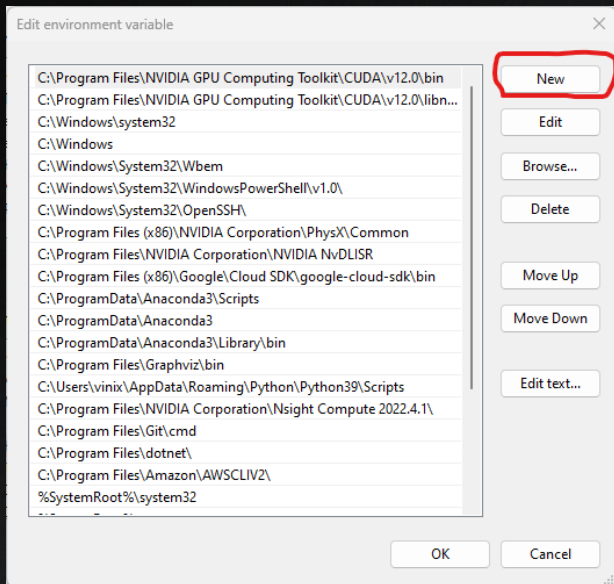


Instalação

Windows

Possíveis problemas – Criando a variável de ambiente manualmente

7. Na nova janela, clique em “Novo”

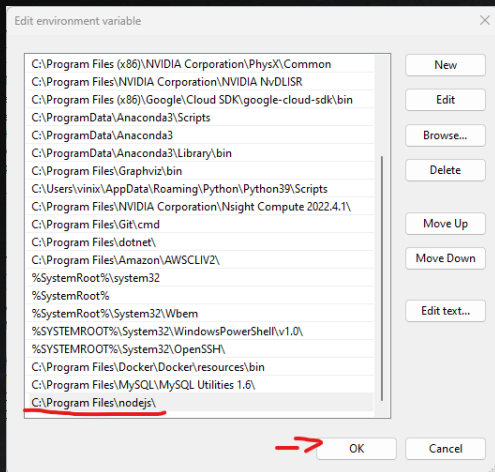


Instalação

Windows

Possíveis problemas – Criando a variável de ambiente manualmente

8. Agora vamos colar o caminho que encontramos do node no começo. No meu caso é o “C:\Program Files\nodejs”. Clique em OK em todas as janelas que abrimos, reinicie o computador e teste os comandos novamente



Instalação

Linux/Macbook

- Gerenciador de pacotes
 1. Acessar <https://nodejs.org/pt/download/package-manager>
 2. Executar o comando de instalação do NVM (Node version manager) – Linha 1
 3. Executar o comando de instalação do node pelo nvm – Linha 5

Gestor de Pacote Instalador do Pré-compilado Binários Pré-compilados Código Fonte

Instalar a Node.js v20.15.0 (LTS) no Linux usando o nvm

```
1 # installs nvm (Node Version Manager)
2 curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
3
4 # download and install Node.js (you may need to restart the terminal)
5 nvm install 20
6
7 # verifies the right Node.js version is in the environment
8 node -v # should print 'v20.15.0'
9
10 # verifies the right NPM version is in the environment
11 npm -v # should print '10.7.0'
```

Bash Copiar para a área de transferência

NPM (Node Package Manager)

NPM é o gerenciador de pacotes do Node.js. Ele permite que desenvolvedores baixem, instalem e gerenciem bibliotecas e ferramentas que facilitam o desenvolvimento de aplicações JavaScript, incluindo React.

Criando um projeto com NPM

Podemos criar um projeto vazio com o NPM, rodando o comando **npm init**

```
MINGW64/c/Users/vinix/Desktop/Projeto1
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: (projeto1) projeto-1 <- Nome do projeto, sem espaços
version: (1.0.0) <- Versão do projeto
description: Eu sou um projeto em NPM iniciado do zero <- Descrição por extenso do projeto
entry point: (index.js) <- Arquivo ponto de entrada do projeto. Não é obrigatório sempre, podemos manter a opção padrão como index.js mesmo
test command: <- Comando para rodar testes automatizados, podemos deixar em branco
git repository: <- Caso você já tenha criado um repositório no git, pode colar a URL aqui. Podemos deixar em branco e editar depois
keywords: <- Palavras chaves do projeto, podemos deixar em branco
author: Vinicius Marcili <- Nome do autor
license: (ISC) <- Tipo de licença de software, podemos deixar o padrão
About to write to C:\Users\vinix\Desktop\Projeto1\package.json:

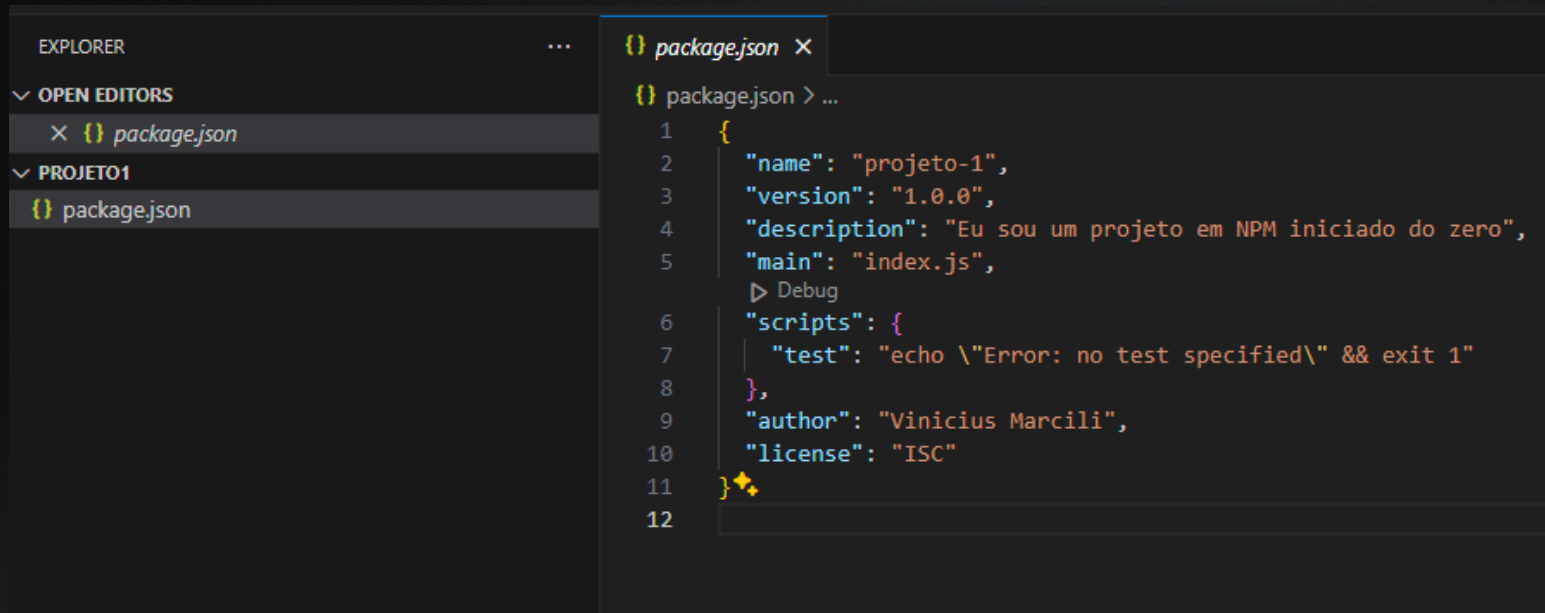
{
  "name": "projeto-1",
  "version": "1.0.0",
  "description": "Eu sou um projeto em NPM iniciado do zero",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Vinicius Marcili",
  "license": "ISC"
}

Is this OK? (yes)

npm notice
npm notice New minor version of npm available! 10.7.0 -> 10.8.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.1
npm notice To update run: npm install -g npm@10.8.1
npm notice
(base)
vinix@VINI-GAMER MINGW64 ~/Desktop/Projeto1
$ |
```

Criando um projeto com NPM

Então o NPM ira gerar para nós um arquivo chamado de `package.json` com as informações que fornecemos ao comando `npm init` anteriormente



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'PROJETO1' with a file named 'package.json'. The main editor area displays the content of 'package.json' with the following JSON structure:

```
1  {
2    "name": "projeto-1",
3    "version": "1.0.0",
4    "description": "Eu sou um projeto em NPM iniciado do zero",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "Vinicius Marcili",
10   "license": "ISC"
11 }
12
```

Entendendo o package.json

Explicando um pouco dos campos que acabamos de gerar no JSON:

- **name** O nome do projeto.
- **version** A versão do projeto.
- **description** Uma breve descrição do projeto.
- **main** O arquivo de entrada do projeto.
- **scripts** Scripts de comando que você pode executar com **npm run <script-name>**.
- **keywords** Palavras-chave que ajudam a identificar o projeto.
- **author** O autor do projeto.
- **license** A licença do projeto.
- **dependencies** Dependências que o projeto necessita para funcionar.

Dependências

- O site npmjs.com (<https://www.npmjs.com/>) é um agregador de pacotes de NodeJS, onde você pode pesquisar, ler suas documentações e ver estatísticas de uso.
- Além do npmjs.com podemos instalar dependências direto de repositórios remotos como o github, ou criar **Registry** privados e instalar diretamente dele.
- Um **Registry** é o banco de dados onde todos os pacotes e suas versões estão armazenados. No caso do npmjs.com, quando você instala um pacote, ele é baixado do registro do NPM.

Instalando uma nova dependência

Para instalar uma nova dependência em um projeto que utiliza o NodeJS basta rodar o comando `npm install <nome-da-dependência>`. Vamos instalar o Tailwind CSS (<https://www.npmjs.com/package/tailwindcss>) que é um framework CSS apenas para teste.

Vamos rodar o comando `npm install tailwindcss`

```
(base) PS C:\Users\vinix\Desktop\Projeto1> npm install tailwindcss

added 113 packages, and audited 114 packages in 6s

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

found 0 vulnerabilities
(base) PS C:\Users\vinix\Desktop\Projeto1> 
```

Instalando uma nova dependência

Agora podemos notar 3 coisas após o sucesso do comando:

1. O nosso **dependencies** no package.json foi atualizado e o Tailwind foi adicionado como uma dependência.

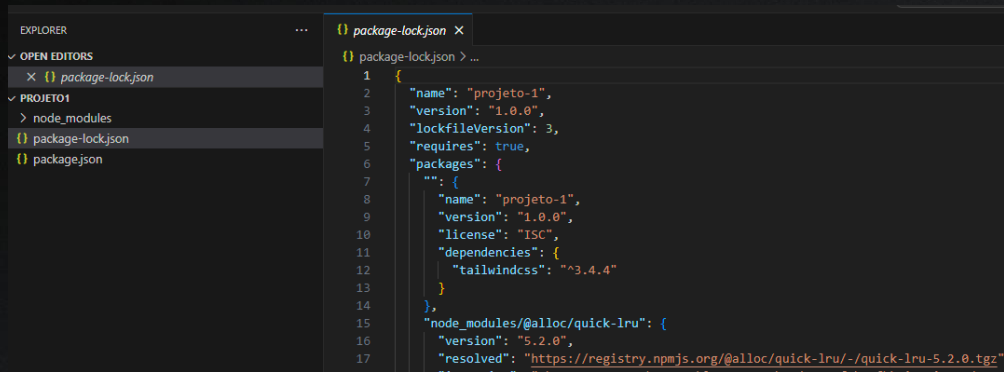
```
1 {  
2   "name": "projeto-1",  
3   "version": "1.0.0",  
4   "description": "Eu sou um projeto em NPM iniciado do zero",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "author": "Vinicius Marcili",  
10  "license": "ISC",  
11  "dependencies": {}  
12 }  
13  
14
```

```
1 {  
2   "name": "projeto-1",  
3   "version": "1.0.0",  
4   "description": "Eu sou um projeto em NPM iniciado do zero",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "author": "Vinicius Marcili",  
10  "license": "ISC",  
11  "dependencies": {  
12    "tailwindcss": "^3.4.4"  
13  }  
14 }
```

Instalando uma nova dependência

Agora podemos notar 3 coisas após o sucesso do comando:

2. Foi criado um arquivo no diretório do projeto chamado de **package-lock.json**. O **package-lock.json** é um arquivo que é automaticamente gerado pelo NPM quando você instala dependências. Ele garante que as instalações futuras sejam consistentes com as versões específicas de cada dependência que foram instaladas inicialmente.



```
1 {
2   "name": "projeto-1",
3   "version": "1.0.0",
4   "lockfileVersion": 3,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "projeto-1",
9       "version": "1.0.0",
10      "license": "ISC",
11      "dependencies": {
12        "tailwindcss": "^3.4.4"
13      }
14    },
15    "node_modules/@alloc/quick-lru": {
16      "version": "5.2.0",
17      "resolved": "https://registry.npmjs.org/@alloc/quick-lru/-/quick-lru-5.2.0.tgz"
```

Instalando uma nova dependência

Agora podemos notar 3 coisas após o sucesso do comando:

3. Foi criado um novo diretório chamado de `node_modules` que é o diretório onde as bibliotecas que instalamos são adicionadas. Uma boa prática é colocar a pasta `node_modules` dentro do nosso arquivo `.gitignore` para que ela não seja enviada para o nosso repositório.

▼ node_modules

- > .bin
- > @alloc
- > @isaacs
- > @jridgewell
- > @nodelib
- > @pkgjs
- > ansi-regex
- > ansi-styles

Instalando uma nova dependência - Global

Também podemos instalar uma dependência de forma global no nosso ambiente, assim ele fica disponível em qualquer projeto ou local da nossa máquina.

Para instalarmos globalmente uma biblioteca usamos a flag `-g` ou `--global` no final do comando da instalação, vamos executar o comando abaixo em qualquer terminal, não necessariamente em um que utilizamos o npm init anteriormente, desde que possua node instalado na máquina.

```
npm install http-server -g
```

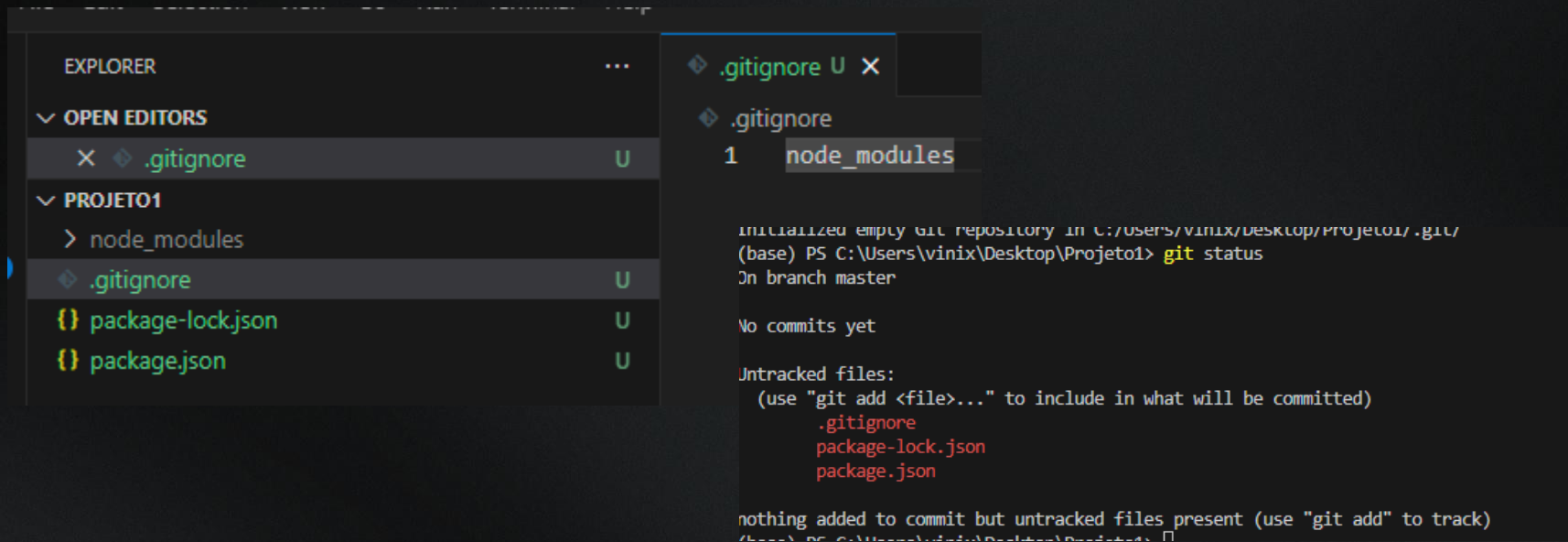
No exemplo estamos instalando uma biblioteca (<https://www.npmjs.com/package/http-server>) que serve arquivos da web de forma estática para criamos um pequeno servidor de testes localmente. Semelhante ao preview do VSCode para arquivos HTML.

Para ver se instalação funcionou vamos digitar o comando `http-server -v` para verificar a versão instalada

```
vinix@VINI-GAMER MINGW64 ~/Desktop/Projeto1 (master)
$ http-server -v
v14.1.1
```

Revisão .gitignore

O arquivo **.gitignore** é um arquivo de texto simples usado para especificar quais arquivos e diretórios devem ser ignorados pelo Git. Isso significa que o Git não rastreará essas mudanças, e eles não serão incluídos nos commits. Este arquivo é extremamente útil para manter o repositório limpo e evitar a inclusão de arquivos desnecessários ou sensíveis.



```
EXPLORER
  OPEN EDITORS
    .gitignore
  PROJETO1
    node_modules
    .gitignore
    package-lock.json
    package.json

.gitignore
1 node_modules

initialized empty git repository in C:/Users/vinix/Desktop/Projeto1/.git/
(base) PS C:\Users\vinix\Desktop\Projeto1> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    package-lock.json
    package.json

nothing added to commit but untracked files present (use "git add" to track)
(base) PS C:\Users\vinix\Desktop\Projeto1>
```

Controle de Versão de Dependências no NPM

No package.json, cada dependência tem uma versão especificada que indica qual versão do pacote o projeto deve usar. Este número de versão segue o padrão SemVer (Semantic Versioning) (<https://semver.org/>), que é composto de três partes: **MAJOR.MINOR.PATCH**. Onde:

- **MAJOR** Mudanças incompatíveis na API / grandes break changes.
- **MINOR** Funcionalidades adicionadas de maneira compatível.
- **PATCH** Correções de bugs de maneira compatível.

Exemplo:

1.2.44 -> 1 é o Major, 2 é o Minor e 44 é o Patch

Símbolos para Controle de Versão

Caret (^)

Permite atualizações que não alteram a versão MAJOR. Por exemplo, ^1.2.3 permite atualizações para qualquer versão 1.x.x onde $x \geq 2$.

^1.2.3 corresponde a $\geq 1.2.3$ e $< 2.0.0$.

Tilde (~)

Permite atualizações que não alteram a versão MINOR. Por exemplo, ~1.2.3 permite atualizações para qualquer versão 1.2.x onde $x \geq 3$.

~1.2.3 corresponde a $\geq 1.2.3$ e $< 1.3.0$.

Sem Símbolo

Quando uma versão é especificada sem nenhum símbolo, é fixada a uma versão específica. Por exemplo, 1.2.3 permite apenas a versão 1.2.3.

Isso garante que nenhuma atualização, nem mesmo patches, sejam automaticamente incluídas.

Exemplos práticos

Observando a imagem ao lado

express: ^4.17.1

Permite qualquer versão 4.x.x onde $x \geq 17$.

Exemplos de versões válidas: 4.17.1, 4.18.0, 4.19.5.

lodash: ~4.17.20

Permite qualquer versão 4.17.x onde $x \geq 20$.

Exemplos de versões válidas: 4.17.20, 4.17.21, 4.17.25.

axios: 0.21.1

Permite apenas a versão 0.21.1.

Não permitirá atualizações automáticas.

```
"dependencies": {  
  "express": "^4.17.1",  
  "lodash": "~4.17.20",  
  "axios": "0.21.1"  
}
```


ANTONIO, C. Pro React: Build Complex Front-End Applications in a Composable Way With React. Apress, 2015.

BOSWELL, D; FOUCHER, T. The Art of Readable Code: Simple and Practical Techniques for Writing Better Code. Estados Unidos: O'Reilly Media, 2012.

BRITO, Robin Cris. Android Com Android Studio - Passo A Passo. Editora Ciência Moderna.

BUNA, S. React Succinctly. Estados Unidos: [s.n], 2016. Disponível em: <www.syncfusion.com/ebooks/reactjs_succinctly>. Acesso em: 12 de janeiro de 2023.

FACEBOOK (2019a). React: Getting Started. React Docs, 2019. Disponível em: <reactjs.org/docs/react-api.html>. Acesso em: 13 de janeiro de 2023.

FACEBOOK (2019b). React Without ES6. React Docs, 2019. Disponível em: <reactjs.org/docs/react-without-es6.html>. Acesso em: 10 de janeiro de 2023.

FACEBOOK (2019c). React Without JSX. React Docs, 2019. Disponível em: <reactjs.org/docs/react-without-jsx.html>. Acesso em: 10 de janeiro de 2023.

FREEMAN, Eric ROBSON, Elisabeth. Use a Cabeça! Programação em HTML5. Rio de Janeiro: Editora Alta Books, 2014

GACKENHEIMER, C. Introduction to React: Using React to Build scalable and efficient user interfaces.[s.i.]: Apress, 2015.

HUDSON, P. Hacking with React. 2016. Disponível em: <www.hackingwithreact.com/read/1/3/introduction-to-jsx>. Acesso em: 13 janeiro de 2023.

KOSTRZEWA, D. Is React.js the Best JavaScript Framework in 2018? 2018. Disponível em: <hackernoon.com/is-react-js-the-best-javascript-framework-in-2018-264a0eb373c8>. Acesso em: janeiro de 2023.

MARTIN, R. Clean Code: A Handbook of Agile Software Craftsmanship. Estados Unidos: Prentice Hall, 2009.

MDN WEB DOCS. Guia JavaScript. Disponível em <<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide>>. Acessado em 29 de janeiro de 2023.

Dúvidas, críticas ou sugestões?

FIAP