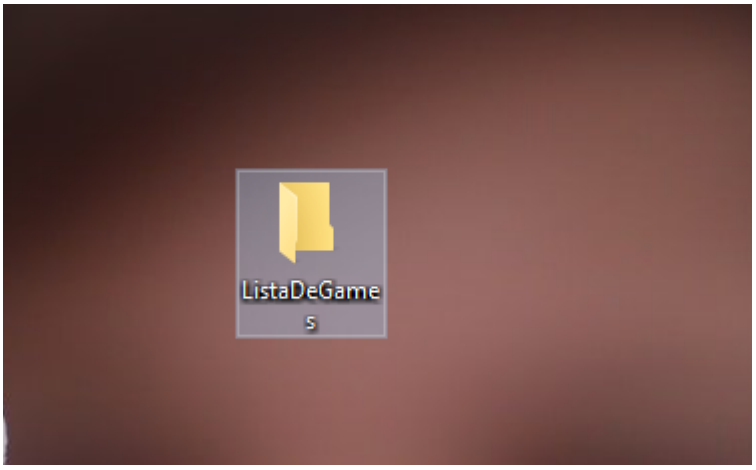
Conteúdo

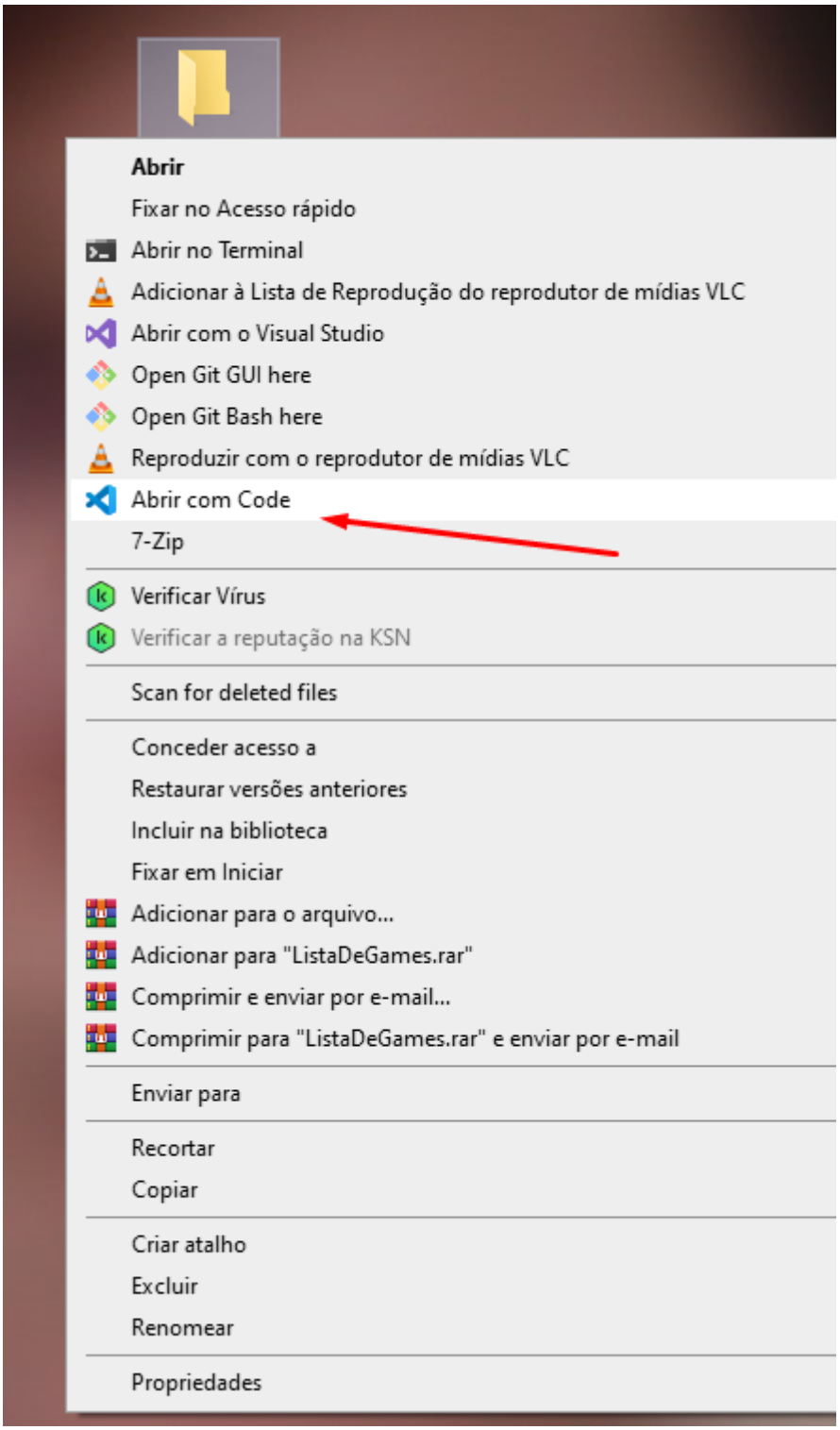
Módulo 1: Introdução ao Node.JS

- Criando pasta na área de trabalho
- Instalando Express
- Instalando Nodemon
- Criando Mini Projeto Express na prática

Primeiro passo vamos criar uma pasta na área de trabalho ou em meus documentos no computador com o nome de **ListaDeGames:**



Na sequência abra com o Vs Code :

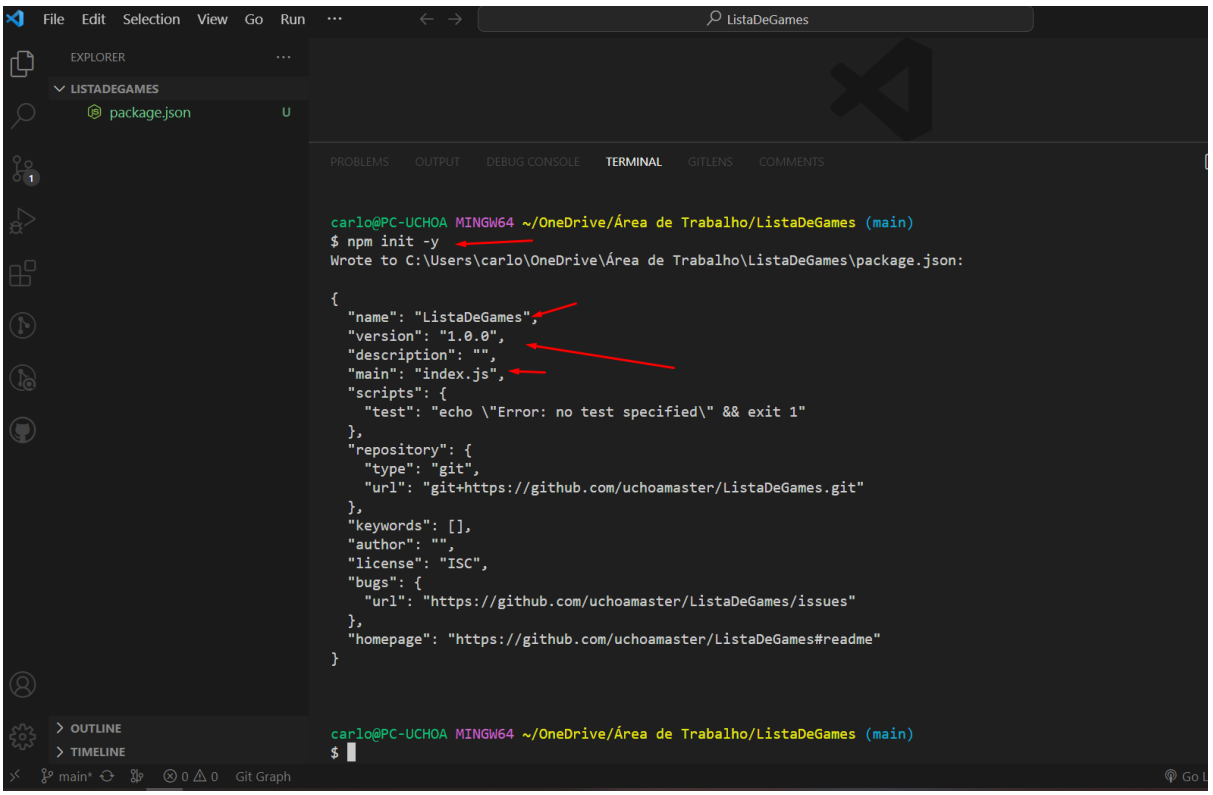


Agora abriremos o terminal integrado no vscode e rodaremos o comando :

npm init -y

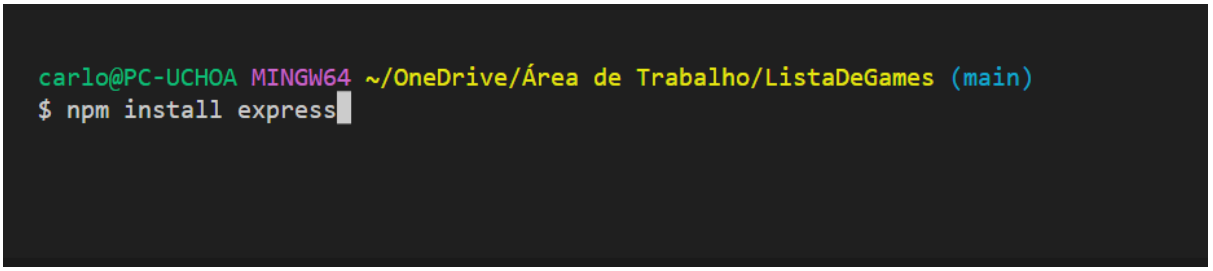
Este comando criará o arquivo package.json para nós com a flag -y, daremos sim para todas as opções que ele nos pergunta ao criar o arquivo.

Notem que por padrão com o node o arquivo **index.js** é nosso padrão temos de cria-lo a seguir:



fechou mas antes iremos instalar o express com o seguinte comando :

npm install express



Após instalarmos o **express**, iremos instalar o **nodemon.**

Nodemon é uma ferramenta que ajuda a desenvolver aplicativos baseados em Node.js, reiniciando automaticamente o aplicativo do nó quando são detectadas alterações no arquivo no diretório.

Nodemon não requer *nenhuma* alteração adicional em seu código ou método de desenvolvimento. Nodemon é um wrapper substituto para `node`. Para usar `nodemon`, substitua a palavra `node` na linha de comando ao executar seu script.

Instalação

Através da clonagem com git ou usando npm (a forma recomendada):

```
npm install -g nodemon # ou usando yarn: yarn global add nodemon
```

E o nodemon será instalado globalmente no caminho do seu sistema.

Você também pode instalar o nodemon como uma dependência de desenvolvimento:

```
npm install --save-dev nodemon # ou usando fio: yarn add nodemon -D
```

Com uma instalação local, o nodemon não estará disponível no caminho do sistema ou você não poderá usá-lo diretamente na linha de comando. Em vez disso, a instalação local do nodemon pode ser executada

chamando-o de dentro de um script npm (como `npm start`) ou usando `npx nodemon`.

```
carlo@PC-UCHOA MINGW64 ~/OneDrive/Área de Trabalho/ListaDeGames (main)
$ npm install -g nodemon
```

Agora garantimos que qualquer mudança em nosso projeto o nodemon se encarregará de **restartar o servidor web**.

Agora iremos criar nosso arquivo principal da aplicação chamado **index.js**,

e dentro dele iremos começar a criar nosso script, iremos carregar o express e criar nossa primeira rota.

index.js U

```
1  //carregando o express
2  const express = require("express");
3  //instancio o express e carregando a biblioteca do express dentro dessa const app
4  const app = express();
5
6  app.listen(3080,() => {
7      console.log("Servidor rodando!");
8  })
```

Feito isso podemos rodar o projeto e ver o mesmo rodando como não chamei nenhuma pagina vai informar que não terá acesso a nada.

```
carlo@PC-UCHOA MINGW64 ~/OneDrive/Área de Trabalho/ListaDeGames (main)
$ node index.js
Servidor rodando!
```

Se acessarmos:

Vamos criar nossa primeira rota:

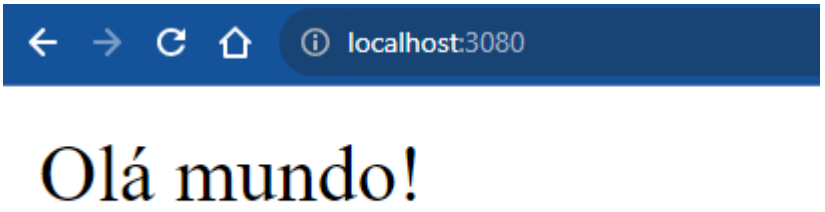
```
app.get("/", (req, res) => {
    res.send("Olá mundo!");
})
```

E como instalamos o nodemon, para não precisarmos ficar reiniciando nossa aplicação, vamos parar agora com o comando **ctrl + c** e rodar novamente agora com o nodemon realizando o monitoramento com o comando:

npx nodemon index.js

```
carlo@PC-UCHOA MINGW64 ~/OneDrive/Área de Trabalho/ListaDeGames (main)
$ npx nodemon index.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Servidor rodando!
```

Com isso o Nodemon ficará monitorando qualquer modificação nos arquivos do projeto e ficará reiniciando o servidor em tempo real.



Ao recarregarmos nossa rota, já mostrará a mensagem na rota principal que criamos.

Como nossa aplicação é uma listagem de Games então de onde iremos puxar essa lista? De um Array Javascript que iremos criar na aplicação com alguns nomes de games, segue imagem do código que iremos adicionar.

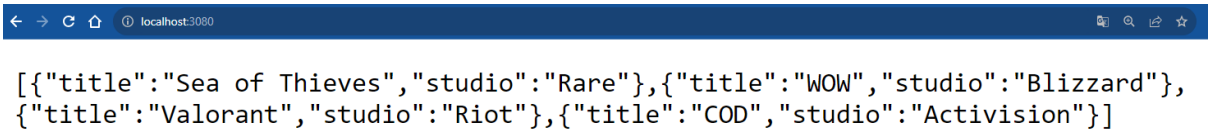
```
//carregando o express
const express = require("express");
//instancio o express e carregando a biblioteca do express dentro dessa const app
const app = express();

//Lista de Games


let games = [
    {title: "Sea of Thieves", studio: "Rare"},
    {title: "WOW", studio: "Blizzard"},
    {title: "Valorant", studio: "Riot"},
    {title: "COD", studio: "Activision"}
]
```

E como que iremos passar essa lista para o navegador?

Bem simples em vez de passarmos a msg Olá mundo iremos passar essa lista em um formato Json, da seguinte forma:

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

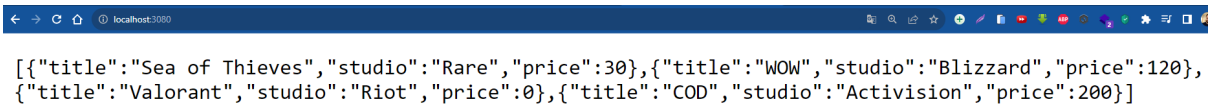
Ao verificarmos fica assim no navegador:



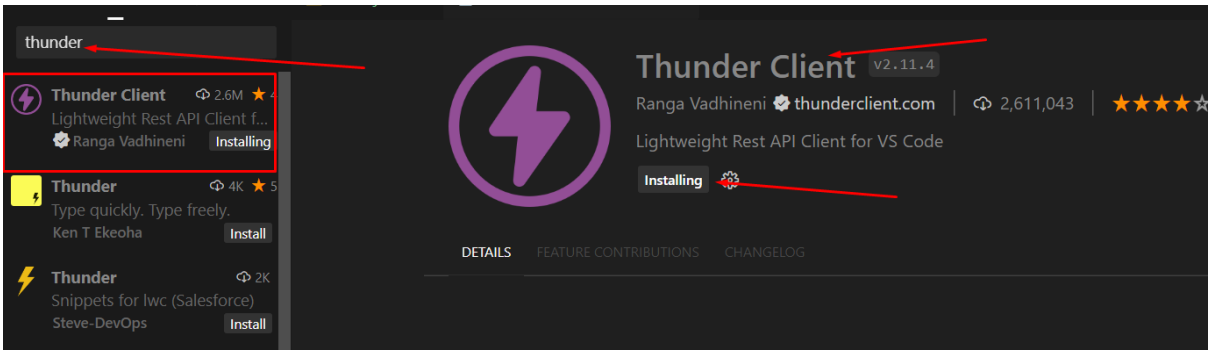
Vamos por mais um campo no caso o preço dos jogos, só para tornar nosso projeto mais completo:

```
let games = [
    {title: "Sea of Thieves", studio: "Rare", price: 30},
    {title: "WOW", studio: "Blizzard", price: 120},
    {title: "Valorant", studio: "Riot", price: 0},
    {title: "COD", studio: "Activision", price: 200}
]
```

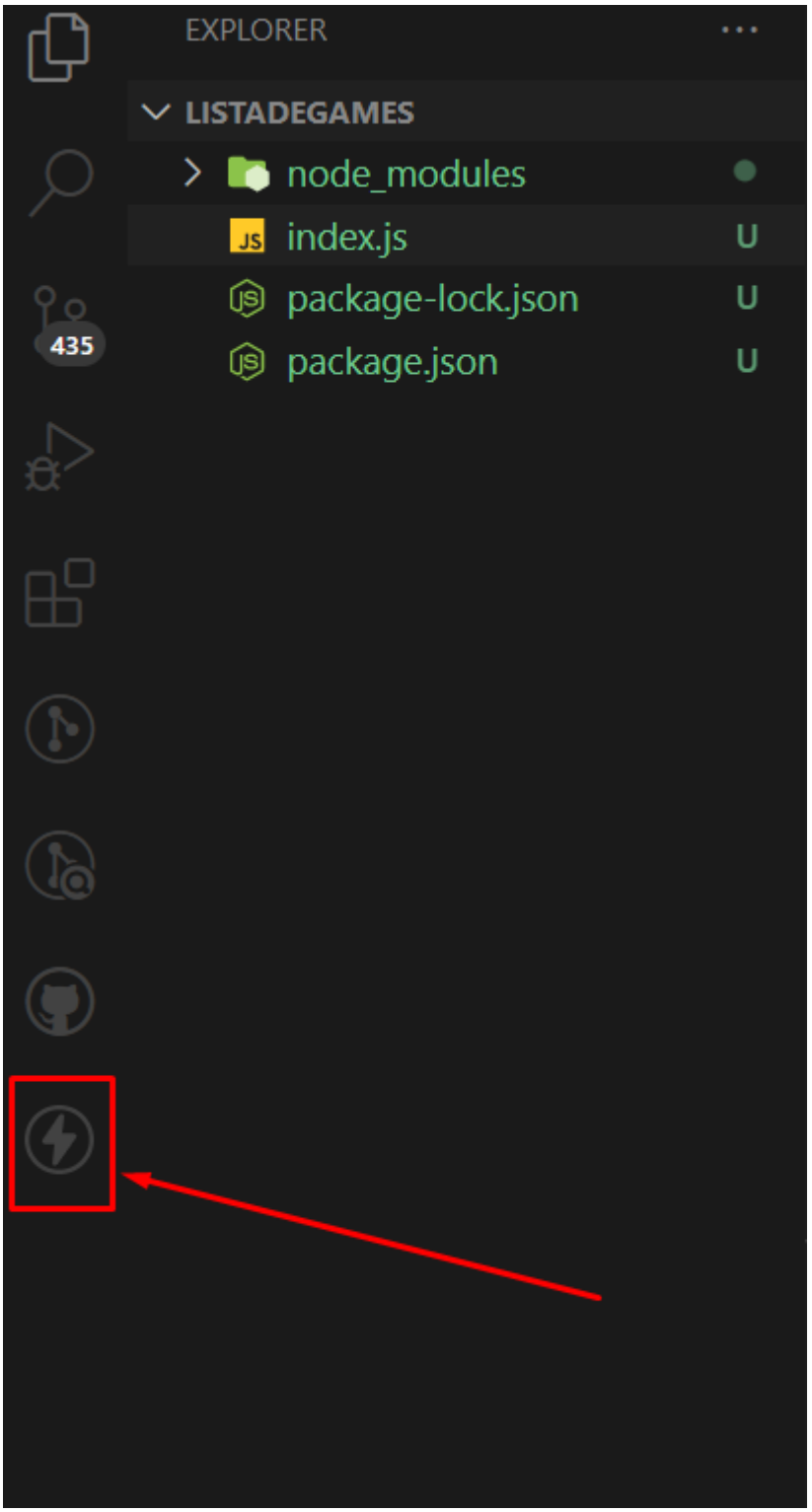
Ficará dessa forma:



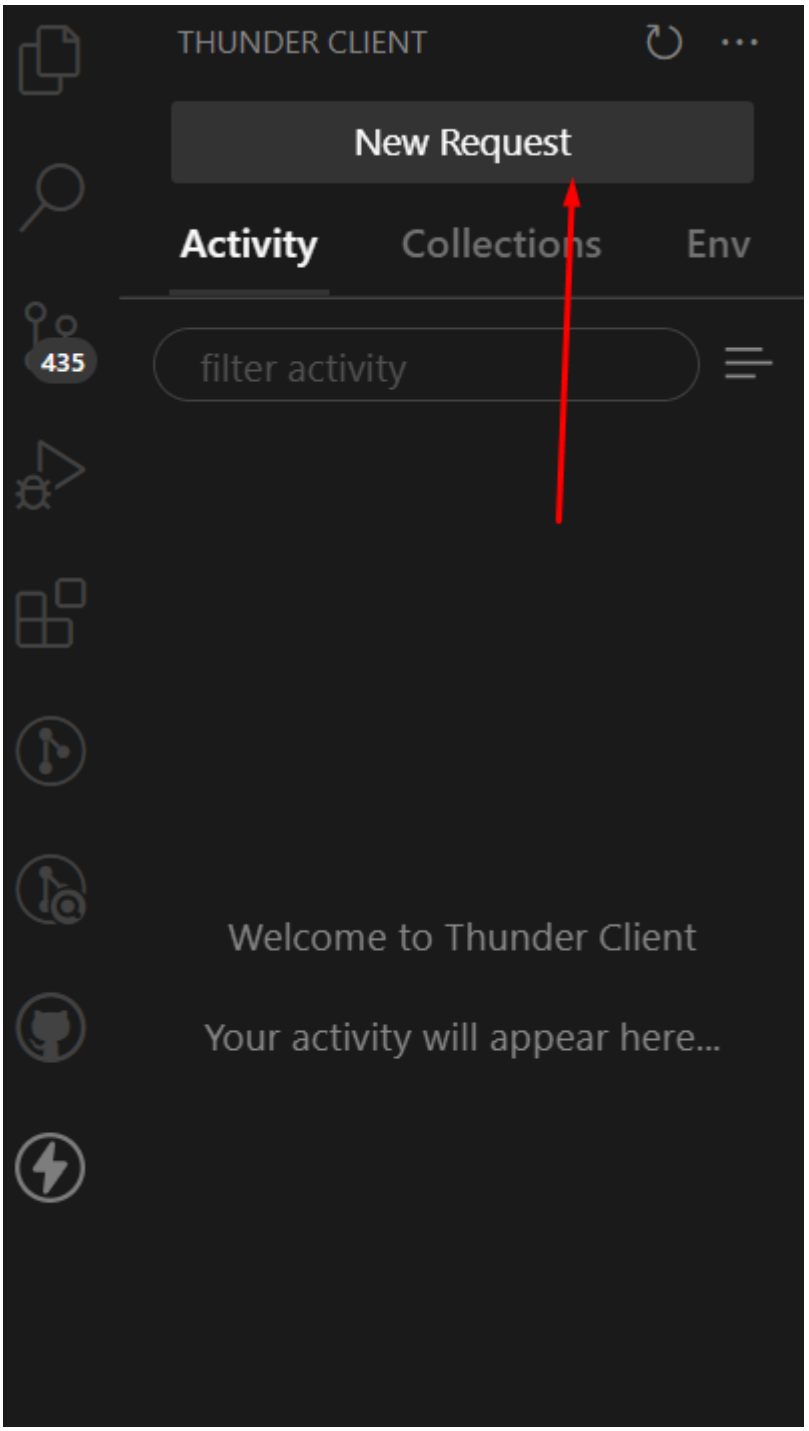
Vamos instalar agora a extensão Thunder Client para podermos manipular os dados da nossa aplicação como API de forma mais dinâmica do que pelo navegador, essa extensão é focada em testar API's web.

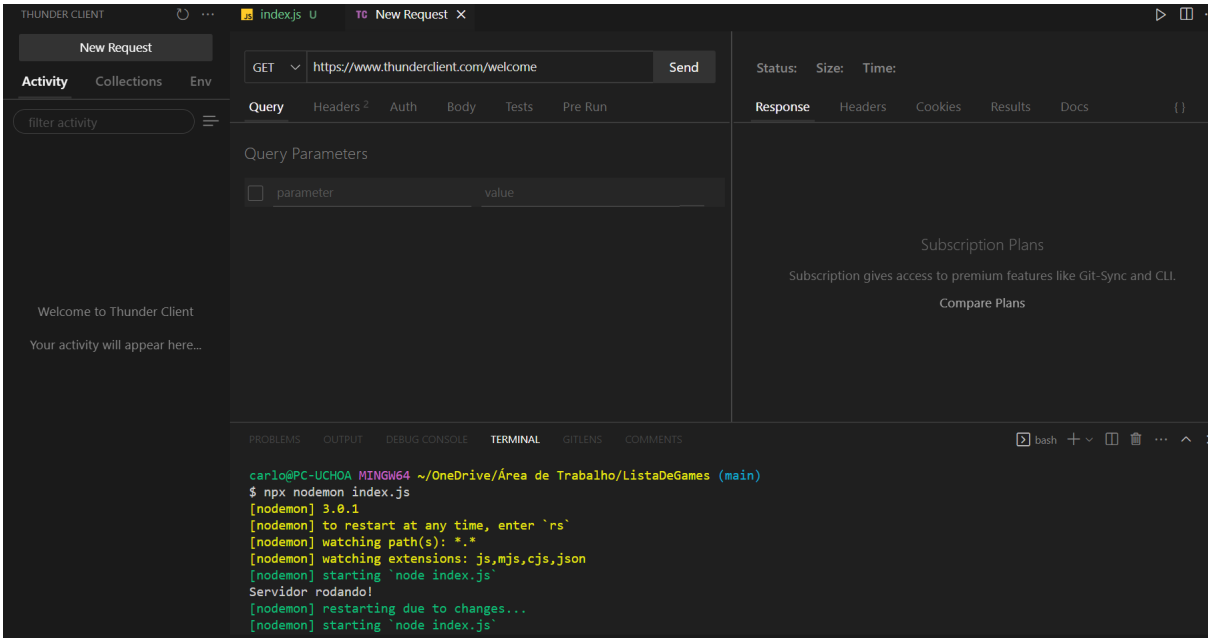


Após instalarmos vamos clicar no menu com o ícone do mesmo para acessarmos e configurarmos nossas rotas da API:



Vamos clicar em Nova Requisição (New Request) :

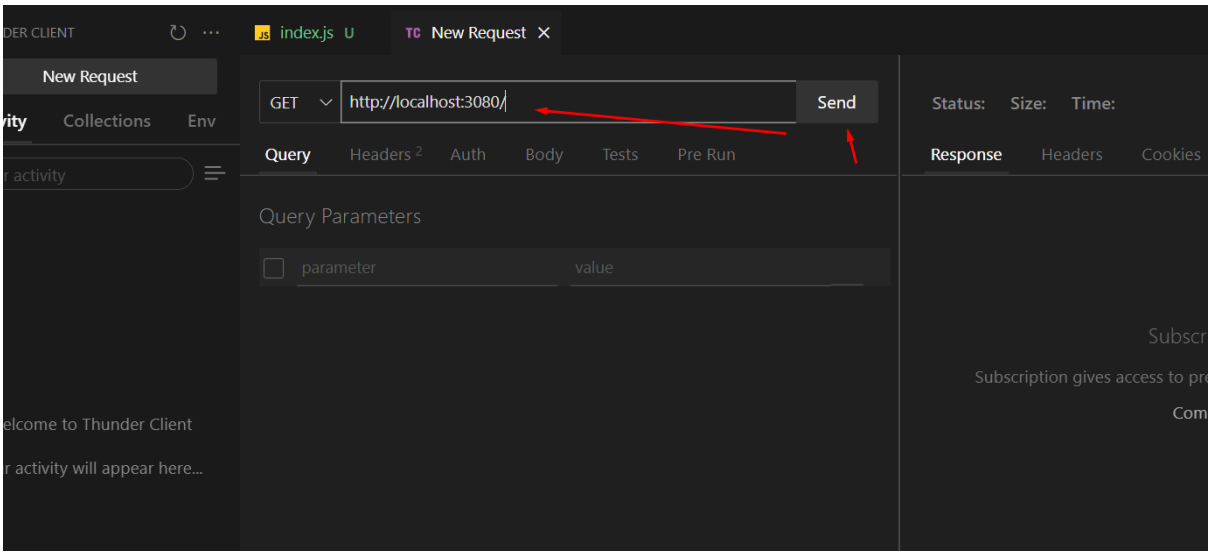

THUNDER CLIENT
New Request
Activity
Collections
Env
435
filter activity
Welcome to Thunder Client
Your activity will appear here...



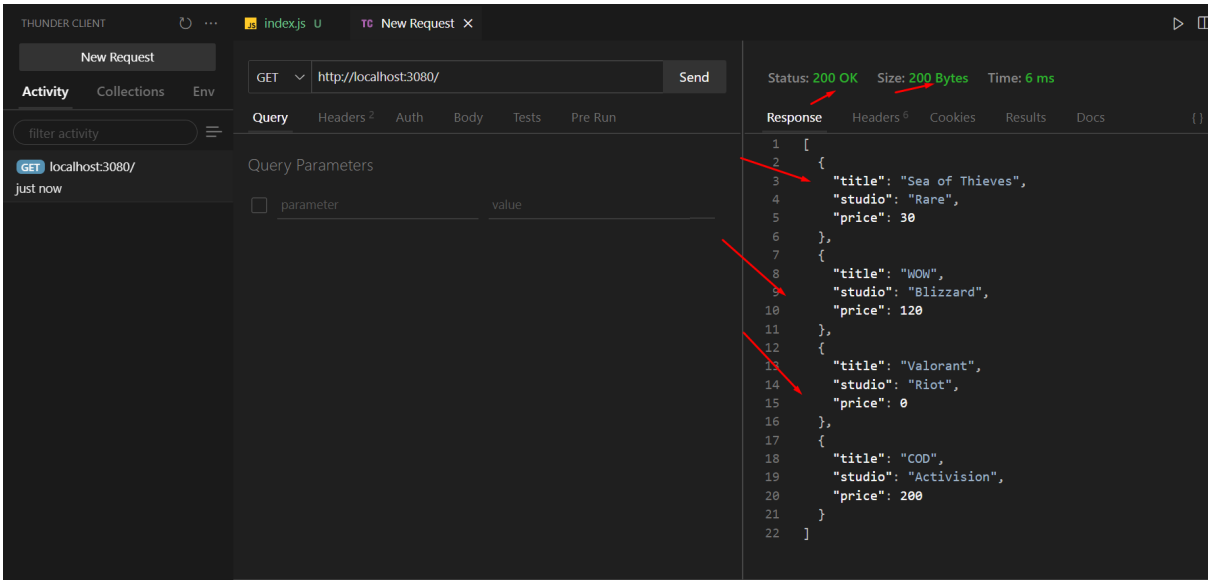
Vamos mudar nosso endereço para o da aplicação e deixarmos do tipo GET , pois nossa rota é desse tipo:

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

Mudamos e clicamos no botão para enviar:



Ao recebermos a msg com a seguinte resposta:

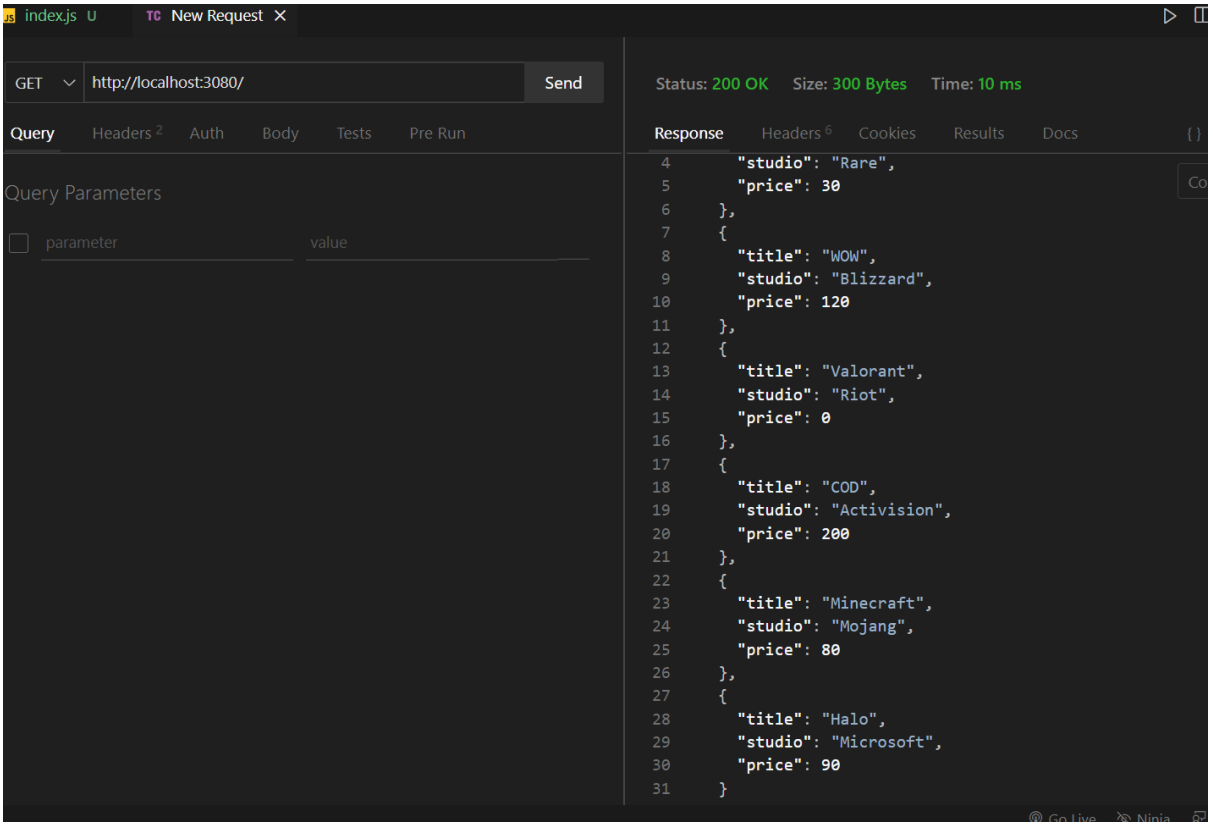


Os dados estão sendo recebidos corretamente em formato de lista Json.

Vamos cadastrar mais dois games e verificarmos:

```
let games = [
    {title: "Sea of Thieves", studio: "Rare", price: 30},
    {title: "WOW", studio: "Blizzard", price: 120},
    {title: "Valorant", studio: "Riot", price: 0},
    {title: "COD", studio: "Activision", price: 200},
    {title: "Minecraft", studio: "Mojang", price: 80},
    {title: "Halo", studio: "Microsoft", price: 90}
]
```

e ao efetuarmos o teste e buscarmos olhem só:



Muito bom para início de brincadeira com Node e Express é isso, depois iremos melhorando cada vez mais.

Atividade

Peço que enviem para o nosso repositório para validação a criação dos nomes de jogos ficam a critério de vocês , quero no mínimo 10 nomes de jogos com o estúdio que criou e o seu preço.