

Conteúdo

Módulo 1: Introdução ao Node.JS

- Criando o método de inserir dados via API

Primeiro passo é preparar a aplicação para que ela possa enviar e receber os dados, para isso precisamos por o seguinte comando :

```
app.use(express.json());
```

Este comando iremos colocar logo abaixo a instância do express , nas linhas iniciais.

Após isso iremos criar uma rota que vai ser responsável apenas por receber dados que o usuário enviar para ela, e a função dessa nova rota é cadastrar um novo jogo.

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

Essa nova rota ficará abaixo da rota do tipo get que criamos anteriormente. Notem que o tipo dela é “POST” sempre que precisar enviar e receber informações utilizaremos esta rota.

E o que precisamos enviar para esta rota? O título do jogo, o studio do jogo e o preço do jogo, para que com essas informações eu possa realizar o salvamento ok? Como pegamos essa informações?

Segue o passo a passo para capturamos as informações:

```
app.post("/novogame", (req, res) =>{  
  let title = req.body.title;  
  let studio = req.body.studio;  
  let price = req.body.price;  
});
```

Eu estou esperando que o usuário me envie estas 3 informações, para isso peço sempre no body da mensagem.

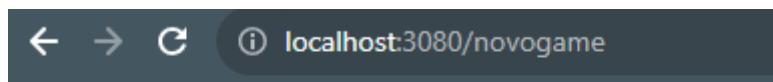
Agora iremos dar um console.log e ver o que está acontecendo ok?

```
app.post("/novogame", (req, res) =>{  
  let title = req.body.title;  
  let studio = req.body.studio;  
  let price = req.body.price;  
  
  console.log(title);  
  console.log(studio);  
  console.log(price);  
});
```

Após fazermos estes comandos, precisamos sempre enviar uma resposta, caso não façamos isso, a aplicação irá travar e o tempo de resposta irá expirar e ocorrerá um erro de timeout por tempo de espera ,notem que pegamos a requisição e falta enviarmos uma resposta e para isso basta passarmos o seguinte comando :

```
app.post("/novogame", (req, res) =>{  
  let title = req.body.title;  
  let studio = req.body.studio;  
  let price = req.body.price;  
  
  console.log(title);  
  console.log(studio);  
  console.log(price);  
  
  res.send("OK");  
});
```

Agora como iremos ver se a rota está funcionando? se tentarmos acessar pela página olha o que acontece:



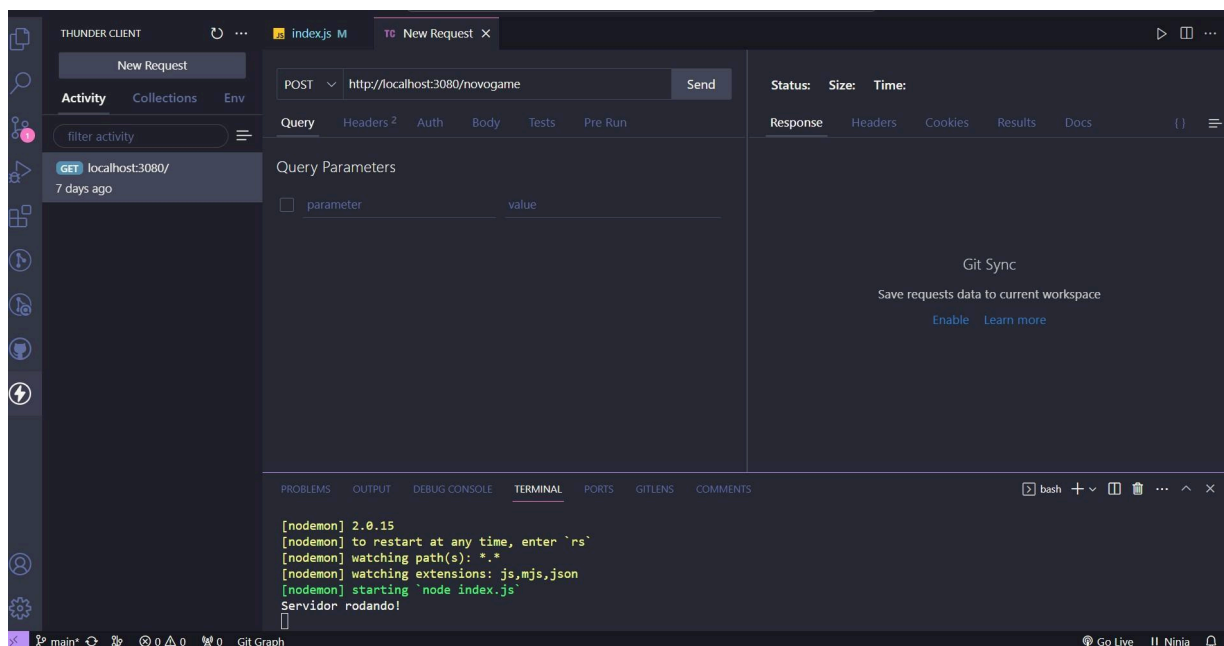
Cannot GET /novogame

Lembram que a rota que criamos é do tipo “POST” ? Então o erro já diz que não consegue encontrar uma rota do tipo GET chamada novogame, pois não existe mesmo compreendem?

para acessarmos precisamos acessar usando o Thunder Client, Postman, Insomnia.

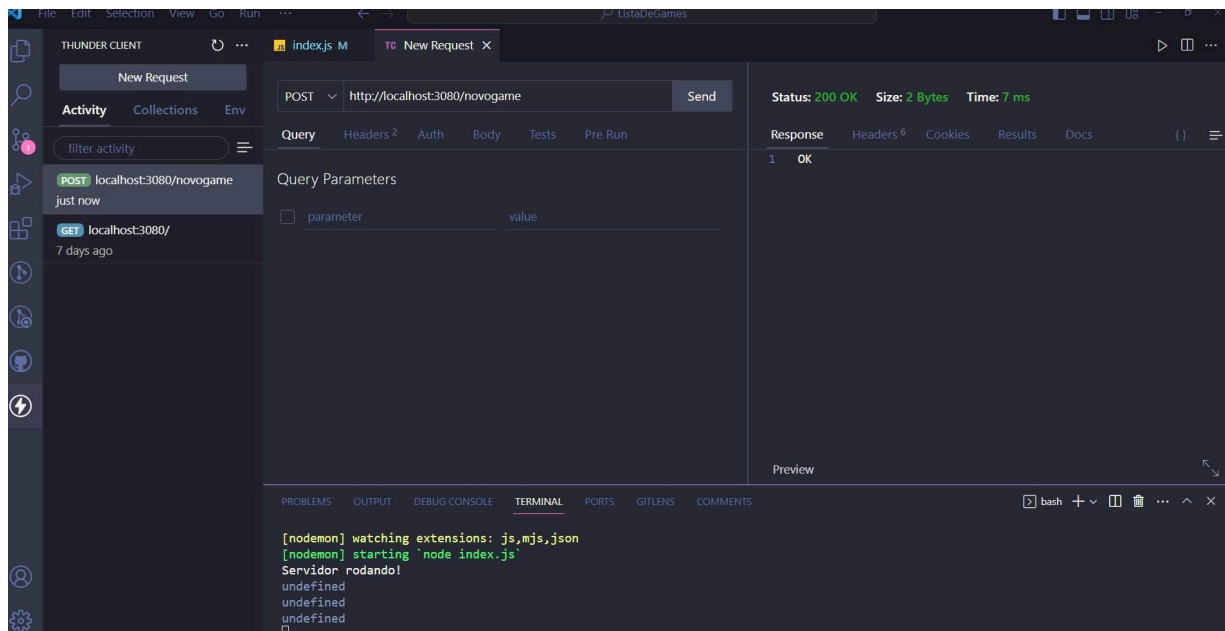
No Front quem irá tratar isso será o VUEJS, REACTJS, etc eles conseguem tratar rotas do tipo POST. Na ausência, utilizaremos o Thunder Client.

Criando nossa rota POST no Thunder Client:

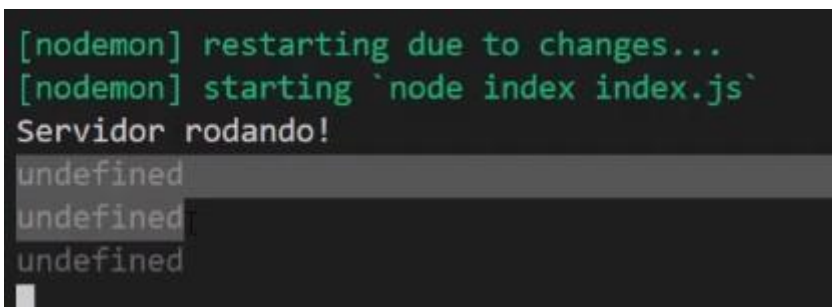


Ao clicar em send para enviar e testarmos a rota pode verificar que apareceu a mensagem “OK”:

UnisenAI

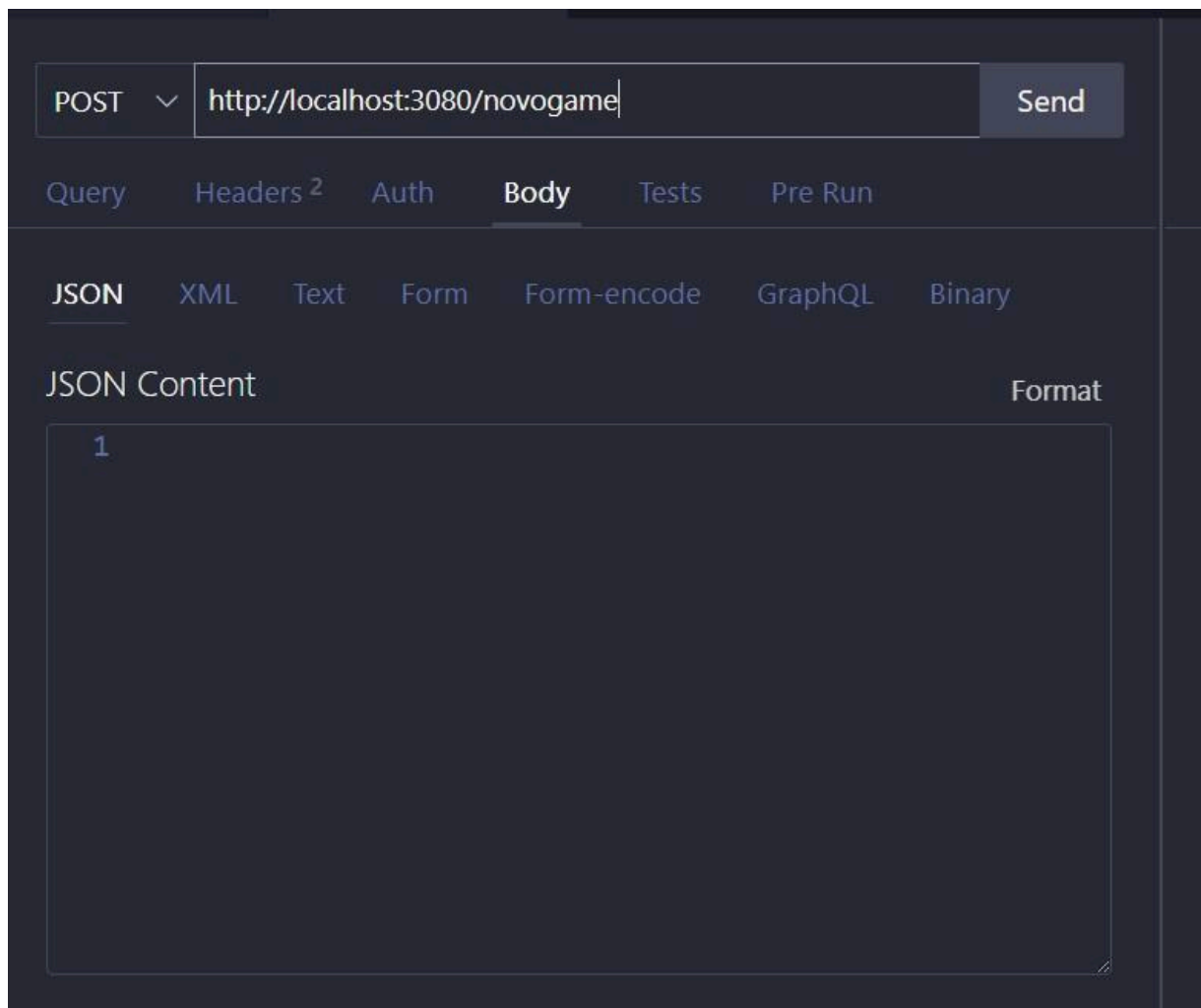


Mas observem que apareceu a mensagem “undefined” 3 vezes também no console de nossa aplicação.



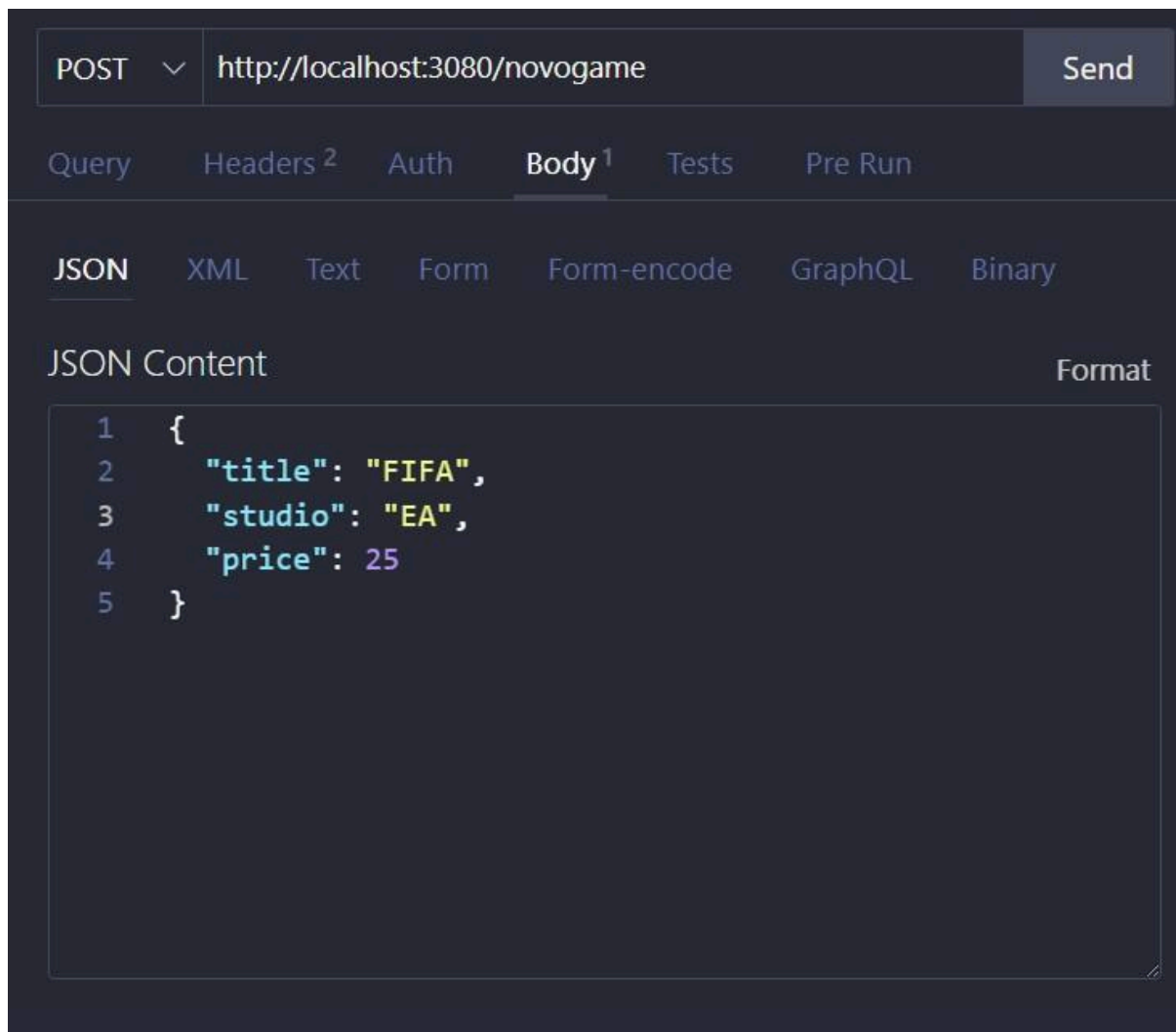
Isso significa que o usuário não está enviando os dados:

Para enviarmos os dados precisamos preencher da seguinte forma:

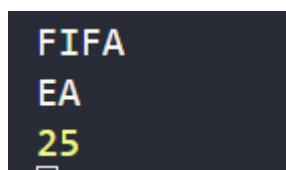


Acessamos Body, lembram da camada da aplicação onde capturamos as informações?

E como estamos trabalhando com Json , já escolhemos o tipo JSON para enviarmos o conteúdo e basta enviarmos com os campos que ele espera que enviemos:



Ao Preenchermos e clicarmos em send, olhem só o que acontece:



Agora ele entende , envia e capturamos o envio das informações.

Agora é bem simples, basta pegamos essas informações e salvamos dentro do array de games que criamos anteriormente, como fazemos isso?

Basta criarmos uma variavel que vai receber os valores contidos na informação enviada:

UniSENAI

```
let newGame = { title: title, studio: studio, price: price }  
  
res.send("OK");
```

Notem que os nomes das variáveis tem o mesmo nome dos campos?

o Javascript nesse caso permite aplicar um recurso para simplificar isso.

basta passarmos assim que ele descobrirá o nome do campo:

```
let newGame = { title, studio, price }
```

E após isso , chamamos o método push do Javascript que enviará as informações para dentro do array de games com o comando :

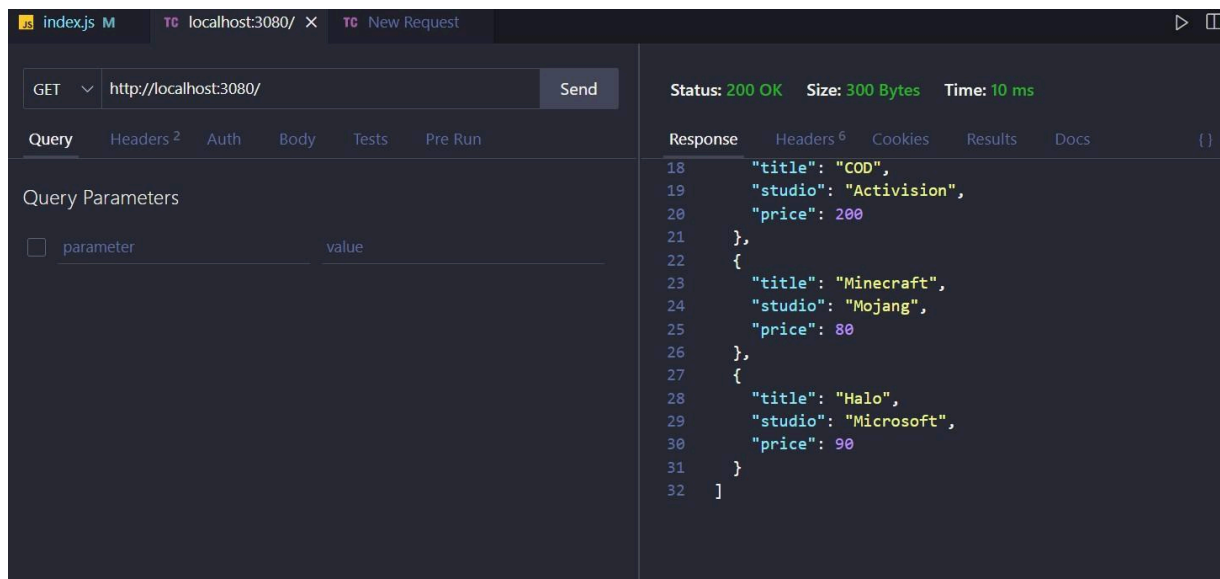
```
let newGame = { title, studio, price };  
//para enviar estes dados para o array agora utilizamos o método push do js  
games.push(newGame);
```

Vamos ver funcionando?

Vamos voltar na rota de listagem do tipo GET:

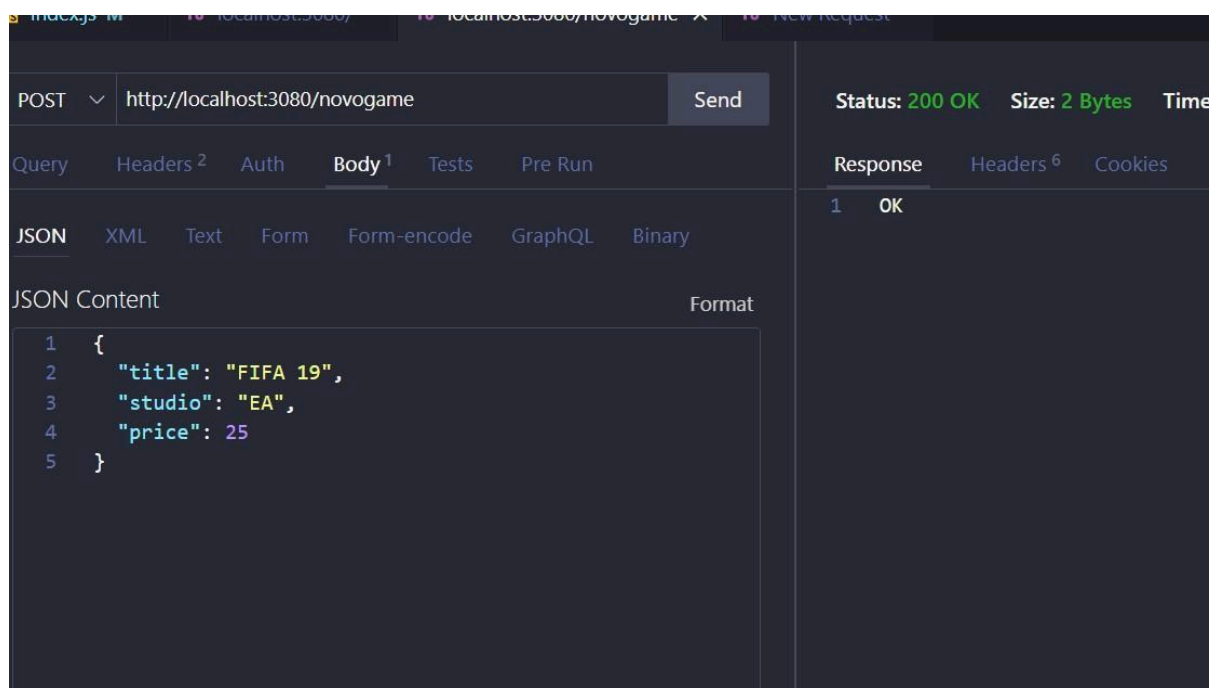
Ao listarmos o nosso array, trás os jogos que temos nele:

UnisenAI



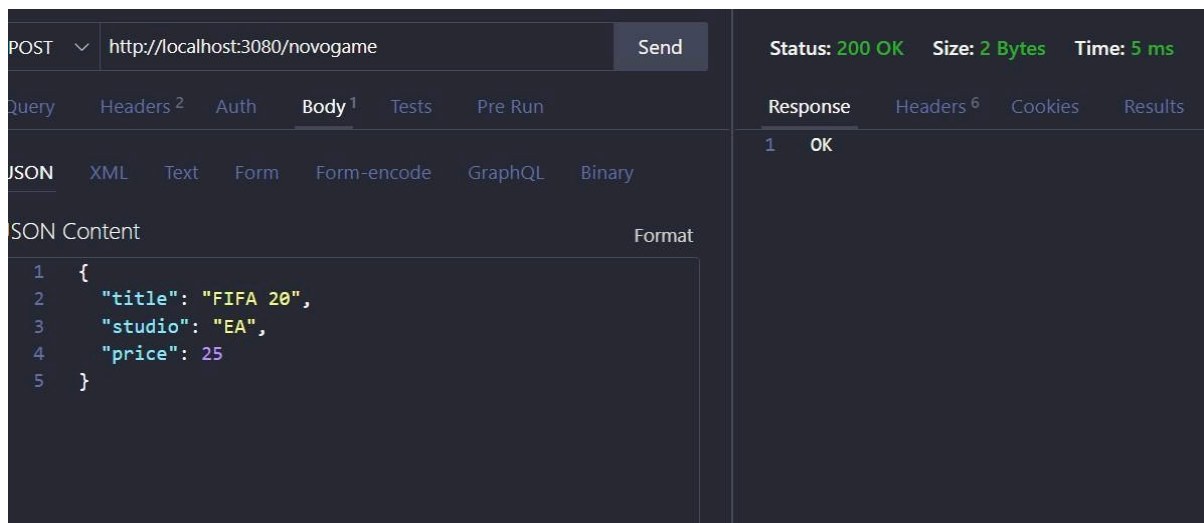
Como podem ver o último é o Halo da Microsoft, Agora vamos na rota de cadastro de games e vamos cadastrar alguns jogos:

Cadastrarmos o jogo FIFA 19:

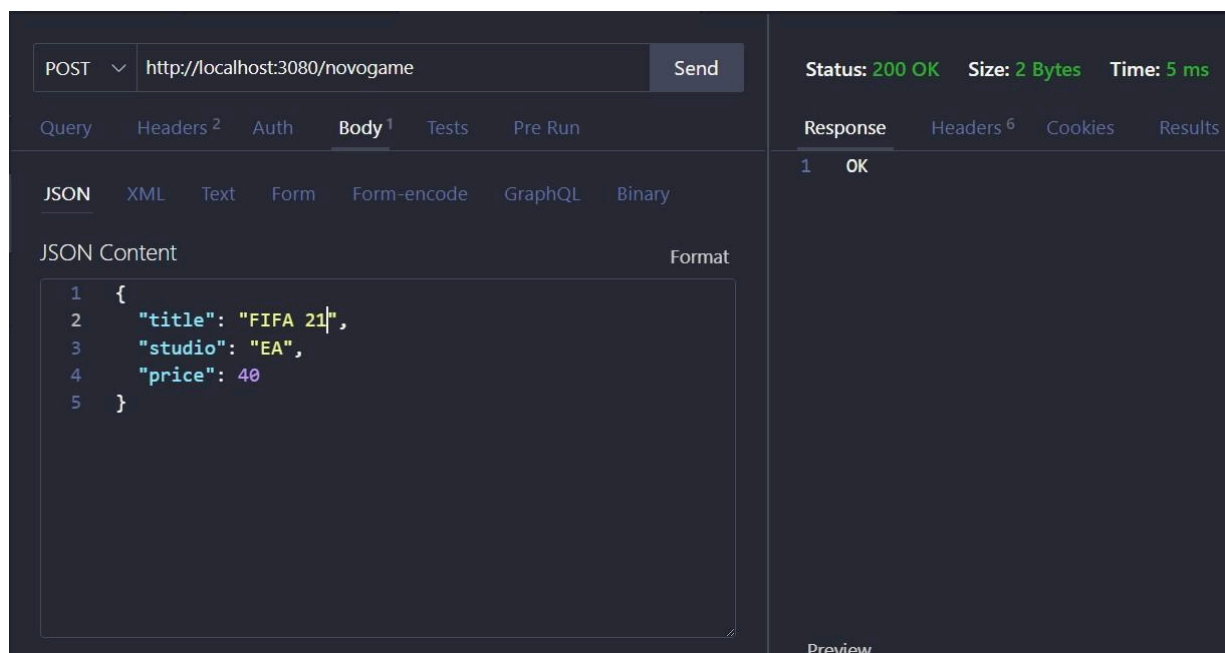


O jogo FIFA 20:

Unisenai

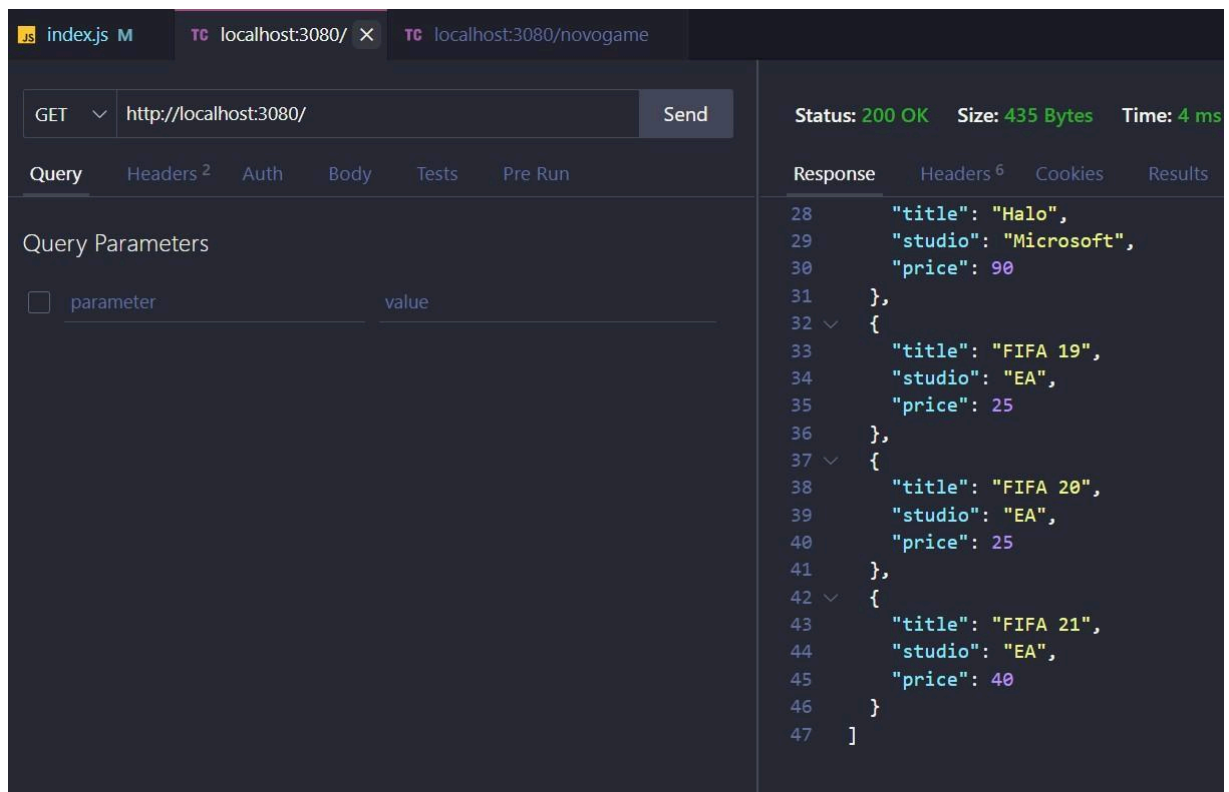


O jogo FIFA 21:



Agora quando voltarmos na rota de listagem de games novamente e clicarmos em Send:

UnisenAI



Fechou pessoal? bem interessante hein?

Espero que tenham gostado, é uma aplicação simples, mas trás muitos conceitos utilizados no dia a dia de grandes corporações, iremos com calma montando toda a estrutura e rota disponíveis ok ?

Atividade para envio no Repositório Github

1. O que é Node.js e por que é popular no desenvolvimento web?
R: Node.js é um software de código aberto, utilizando javascript no back-end usando motores de busca V8 do Google.
2. Qual é a diferença entre o Node.js e outras tecnologias de servidor, como o Apache?

R: Node utiliza um modelo de I/O não bloqueante e orientado a eventos.
3. Como você inicia um projeto Node.js usando o npm?
R: npm init -y
4. O que é o Express.js e qual é o seu papel no desenvolvimento web com Node.js?
R: s. Ele possui características que facilitam o desenvolvimento de aplicações web e APIs, como um sistema de rotas completo, tratamento de exceções, gerenciamento de requisições HTTP
5. Explique o conceito de middleware no contexto do Express.js.
R: são funções que têm acesso ao objeto de solicitação (request), ao objeto de resposta (response) e à próxima função de middleware no ciclo de requisição.
6. Como você roteia solicitações HTTP em um aplicativo Express?
R: Importando o express no projeto, instanciando no projeto `ex const app = express();` e podemos utilizar metodos get ou post desse jeito: `app.post` ou `app.get`
7. O que é o middleware de análise de corpo (body-parser) e por que é útil em um aplicativo Express?
R: Faz a analise e parse dos dados de entrada contidos no corpo da requisição, disponibilizando as propriedades em `req. body`
8. Quais são os principais métodos HTTP e como eles são usados em rotas Express?
R: GET, POST, PUT, HEAD, DELETE
`app.post()`
`app.get()`
`app.delete()`
`app.put()`
9. Como você lida com erros em um aplicativo Express?
R: exemplo de uma tratativa de erro:

```
function errorHandler(err, req, res, next) {  
  if (res.headersSent) {  
    return next(err);  
  }  
  res.status(500);  
  res.render('error', { error: err });  
}
```

10. O que é uma API RESTful e como o Express pode ser usado para criar uma?
R: API RESTful é uma arquitetura para desenvolvimento de API's, serviços para web, Express.js é um framework para Node.js que simplifica a criação de aplicativos e APIs.