BANCO DE DADOS SIMPLES CRIADO EM UM MÓDULO DO NODE...

```
//BANCO DE DADOS:

//Abaixo temos a criação de Banco de Dados fictício para armazenar alguns produtos, que irá simular o recebimento e mantenimento de dados dentro de uma Banco de Dados, bem como cada valor incerido irá receber um número sequencial.

//Abaixo temos um objeto responsável por números á sequência dos produtos adicionados ao banco de dados...
const sequence = {
    _id: 1, //Perceba que criamos a nossa propriedade (apenas um convenção) para dar início a sequência de ids...
    get id() {return this._id++} //Usando o método "get" toda vez que a chave "id" for referenciada, ela dará start a função "id()" que sempre irá retornar o id atual de acordo com o elemento que está chamando, que no caso será o objeto que guardará os produtos...
}

//Abaico temos o objeto que receberá os produtos, com suas chaves id e valores...
const produtos = {}

//Aqui temos uma função responsável por adicionar produtos ao objeto "produtos"...
function salvarProduto(produto) {
    if (!produto.id) produto.id = sequence.id //Note que a função verifica se o produto tem id ou não, se não tiver, ela adiciona um id de acordo com a sequência de ids do objeto "produtos"
    produtos[produto.id] = produto //Se o id já tiver sido passado ela simplesmente passa o valor id para a chave do produto e o recebe o produto como valor para aquele respectivo "id"
    return produto
}

//Essa função mostra um determinado produto de acordo com o seu número id correspondente, se o id não existir ele mostra uma mensagem de que o produto não existe...
function getProduto(id) {
    return produtos[id] || console.log('Produto inexistente!')
}

//Essa função retorna todos os produtos do DB...
function getProdutos() {
    return Object.values(produtos)
}
```

```
//Essa função deleta um valor específico, de um DB...
function excluirProdutos(id) {
    const produto = produtos[id] //Primeiro armazenamos o valor que será deletado numa variável e ...
    delete produtos[id] //Depois deletamos o valor do banco de fato...
    return produto //Por fim, mostramos o valor que foi deletado...
}

//Como o nosso Banco será um Módulo do Node, usamos o module.exports para tornar as funções visíveis para o nosso arquivo servidor...
//Perceba que somente as funções estão visíveis, pois será por através delas que iremos visualizar o que temos no banco, e adicionar produtos...
module.exports = {
    salvarProduto,
    getProduto,
    getProdutos,
    excluirProdutos
}
```

SERVIDOR CRIADO COM EXPRESS E BODY-PARSER...

```
//BANCO DE DADOS DE UM SERVIDOR:

//Abaixo temos a criação de um servidor para trabalhar com um DB que está em memória. O DB ele não possuí nenhuma integração com SQL ou coisa do tipo, é simplesmente um arquivo em memória que vai funcionar como uma espécie de DB fictício.
const porta = 3003


//IMPORT DE EXPRESS...
const express = require('express')
const app = express()


//IMPORT DE BODY-PARSER...
const bodyParser = require('body-parser')


//IMPORT DO BANCO DE DADOS...
const db = require('./02-DB_Express_DB_Side')
```

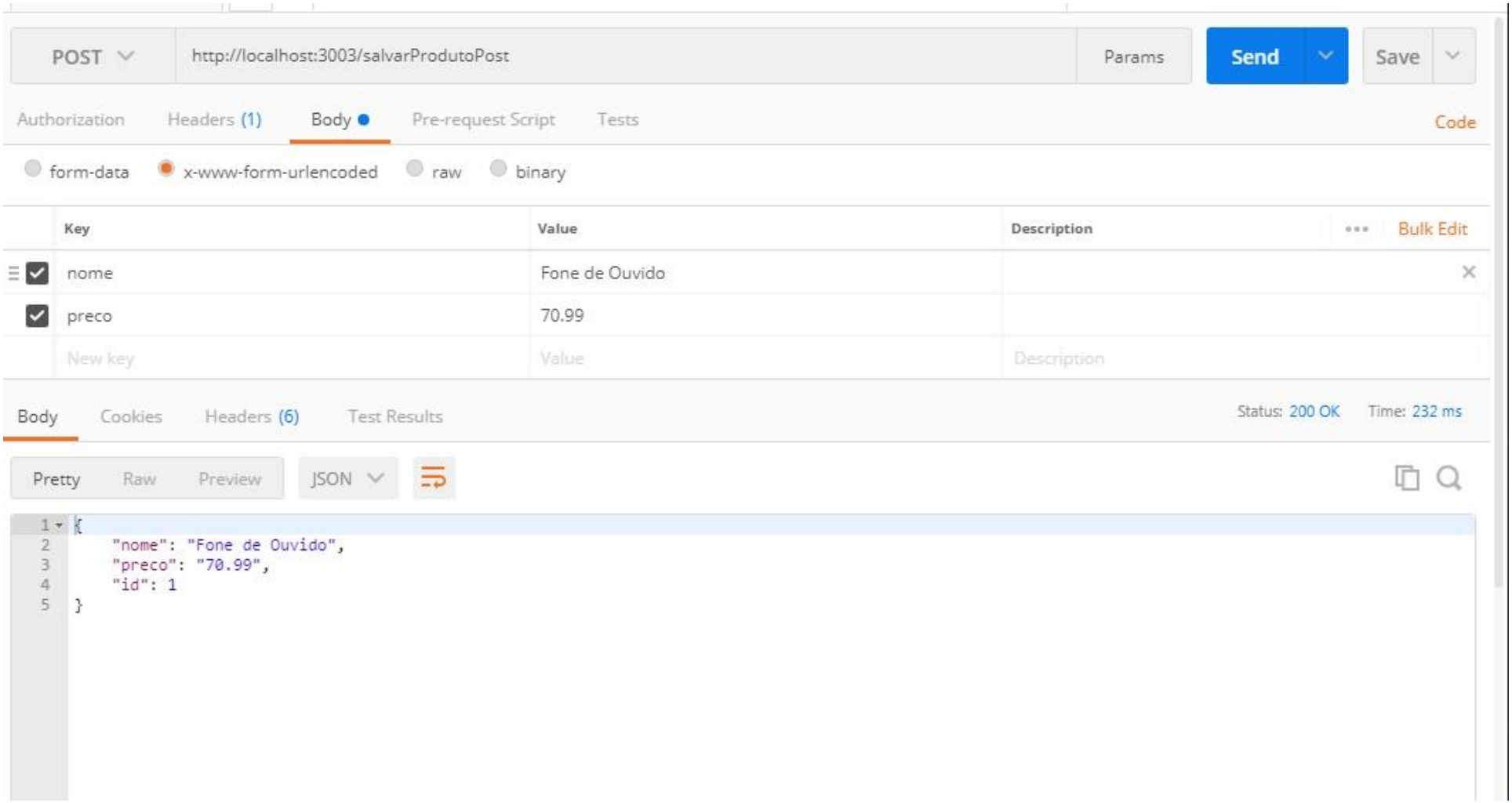
```
//Aqui, usando a biblioteca body-parser, nós criamos um parser para objetos passados por através de uma requisição post possam ser codificiados para o formato JSON no corpo da nossa função, sem esse urlencoded não poderíamos fazer a codificação para JSON, gerando erro na nossa aplicação.
app.use(bodyParser.urlencoded({extended: true})) //Perceba que usamos um middleware do tipo "use", que receberá toda e qualquer requisição, fazendo com que ela passe pelo "use" e seja decodificada para o urlencoded, para que possa ser interpretada pela nossa aplicação no Node.


//Geramos uma função middleware, onde pegamos todos os valores do banco de dados...
app.get('/produtosGet', (request, response, next) => {
    response.send(db.getProdutos())
})


//Aqui temos uma função que captura um produto pelo número "id", mas perceba que para chamar um produto pelo id pela URL, temos que colocar o número id dentro do URL. Para isso temos o elemento "params", simbolizado pelo sinal de ":", e na sequência temos o nome da variável que desejamos referenciar o valor, que no caso é a variável "id"...
app.get('/produtoGet/:id', (request, response, next) => {
    response.send(db.getProduto(request.params.id)) //Note que para obter o valor desejado como resposta, no método getProduto utilzamos utilizamos o "request", junto ao parâmetro e a variável, o valor que for passado logo após a barra, por exemplo: '/produtoGet/1', será procurado dentro do banco de dados...
})



//Perceba que para receber dados, o método já seria o post, um método mais seguro que precisará de ser codificado para transmitir os dados via rede. Para tanto usamos o método post() na nossa função middleware...
app.post('/salvarProdutoPost', (request, response, next) => {
    const produto = db.salvarProduto({ //Dentro da função geramos uma variável "produto" que irá receber os dados para o valor que será salvo dentro do objeto "produtos" no nosso banco de dados...
        nome: request.body.nome, //Note que cada produto terá um nome e um preco, temos que usar o request.body, que serão responsáveis por incorporar os dados recebidos no corpo da requisição, e não na URL, para tanto usaremos o postman para poder atribuir esses valores ás respectivas chaves...
        preco: request.body.preco
    })
    response.send(produto) //Ao final, nossa função retornará para nós o produto que foi adicionado ao Banco de Dados...
})
```

```
//Abaixo temos uma função que pode ser usada para modificar valores de um determinado id, para isso, usamos o método put...
app.put('/salvarProdutoPut/:id', (request, response, next) => { //Perceba que agora ela recebe um número id, para que possa modificar um produto á sua maneira
    const produto = db.salvarProduto({
        id: request.params.id, //Agora a função "salvarProduto()" também pode capturar um id específico, note que, como não desejamos criar um novo id, e sim capturar um já existente no campo de params, usamos a propriedade params para isso, invés da body, "body" coloca um valor novo.
        nome: request.body.nome,
        preco: request.body.preco
    })
    response.send(produto) //Ao final, nossa função retornará para nós o produto que foi adicionado ao Banco de Dados...
})


//Abaixo temos uma função que pode ser usada para deletar valores que correspondem a um id específico
app.delete('/excluirProdutoDelete/:id', (request, response, next) => { //Perceba que agora ela recebe um número id, para que possa deletar um determinado produto, e que, invés de post, ela utiliza o comando http "delete"
    const produto = db.excluirProdutos(request.params.id) //Para referenciar ao objeto correto que desejamos deletar usamos o params mais uma vez...
    response.send(produto) //Ao final, nossa função retornará qual foi o objeto deletado do Banco de Dados...
})


//Essa função é reponsável por fazer a nossa aplicação ser visualizada por através de uma porta...
app.listen(porta, () => {
    console.log(`O servidor está ativo na porta ${porta}...`)
})
```

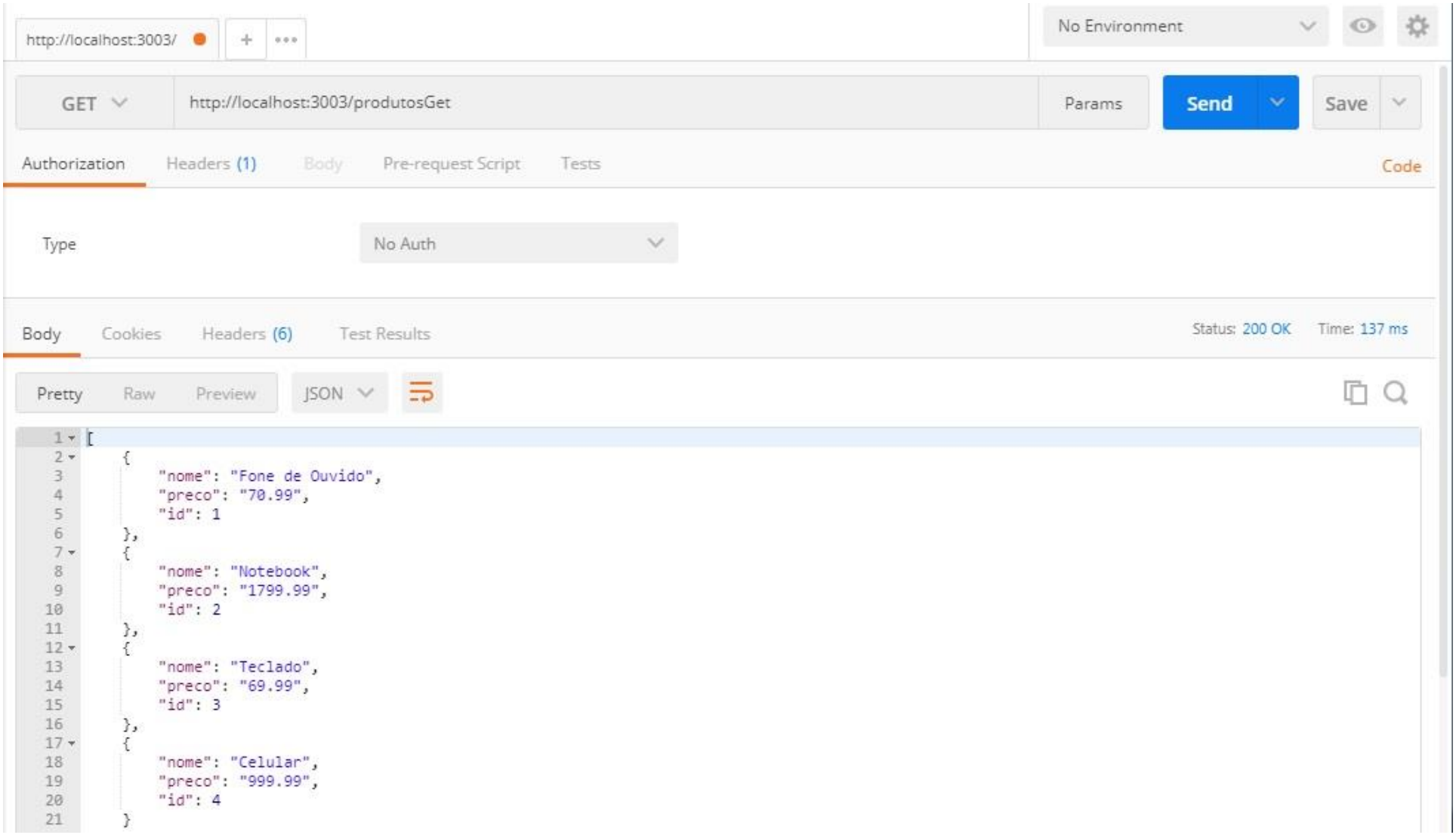
RESULTADO NO CONSOLE QUANDO O SERVIDOR É ABERTO...

```
[Running] node "c:\Users\GaGra\Documents\javascript\arquivos_das_aulas\Projetos_Back_End\Projeto_API_Express\src\02-DB_Express_Server_Side.js"
O servidor está ativo na porta 3003...
```

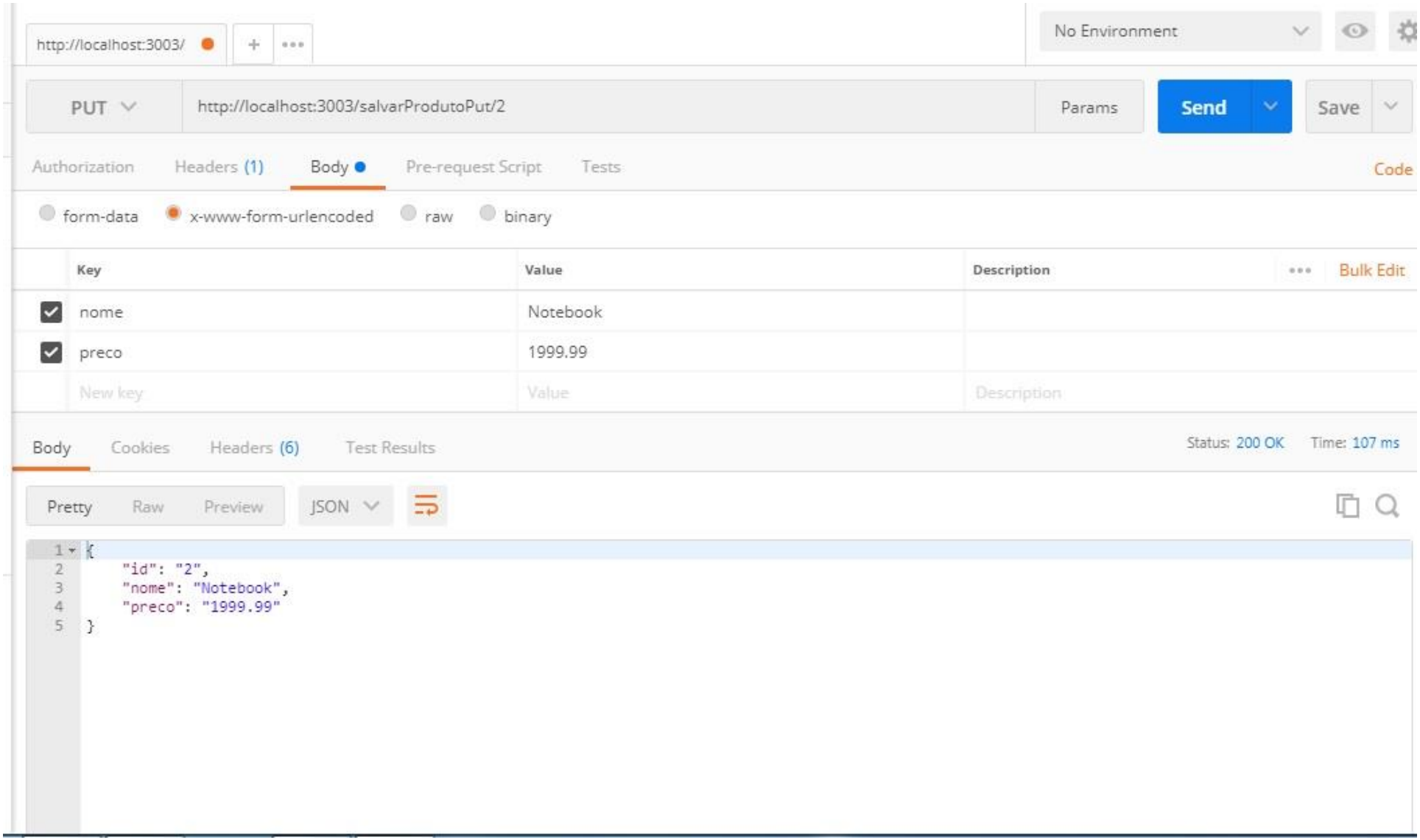
RESULTADO NO POSTMAN QUANDO INCLUÍMOS OBJETOS NO BANCO...



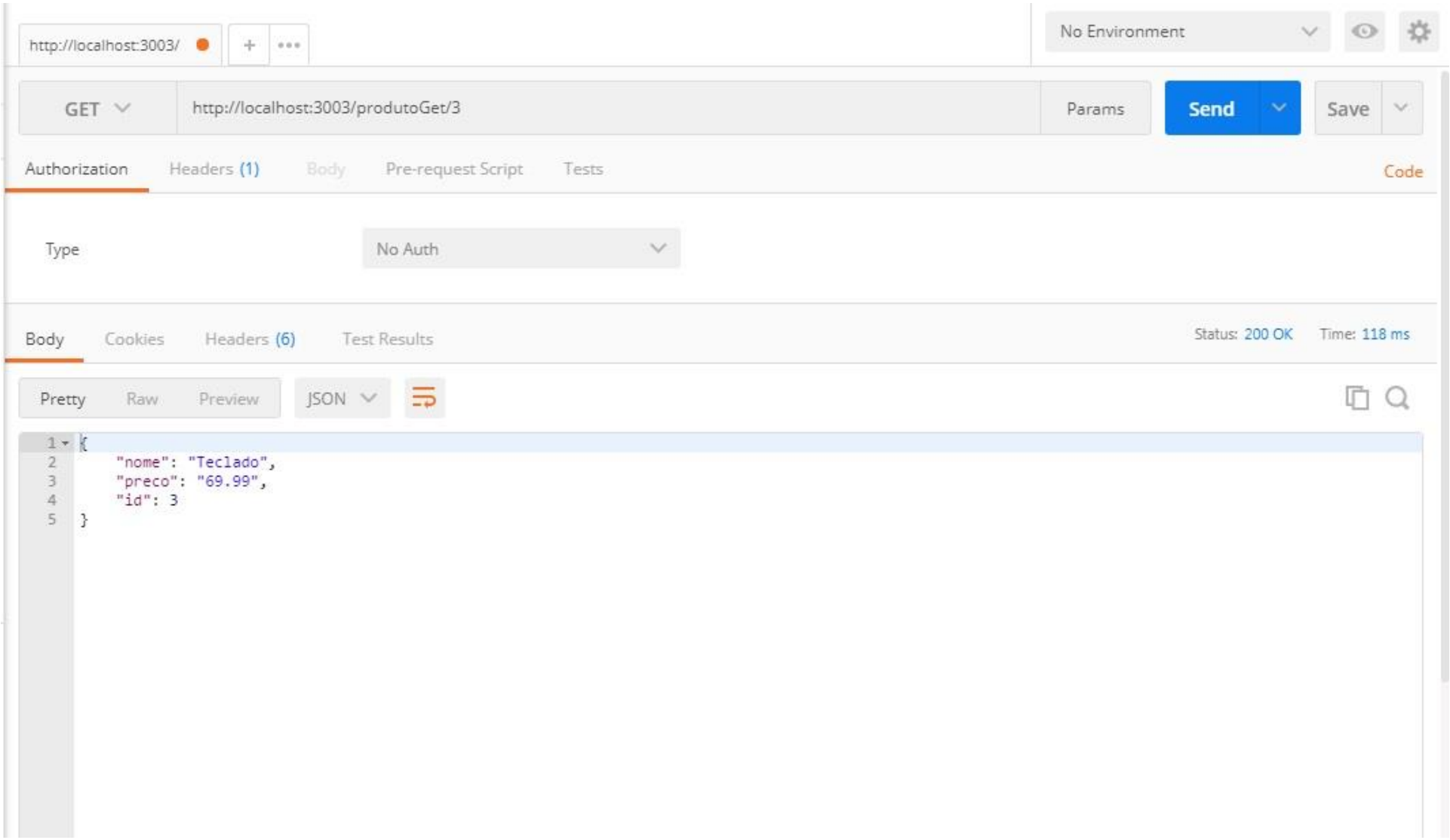
VISUALIZANDO A LISTA DE PRODUTOS SALVAS NO BANCO...



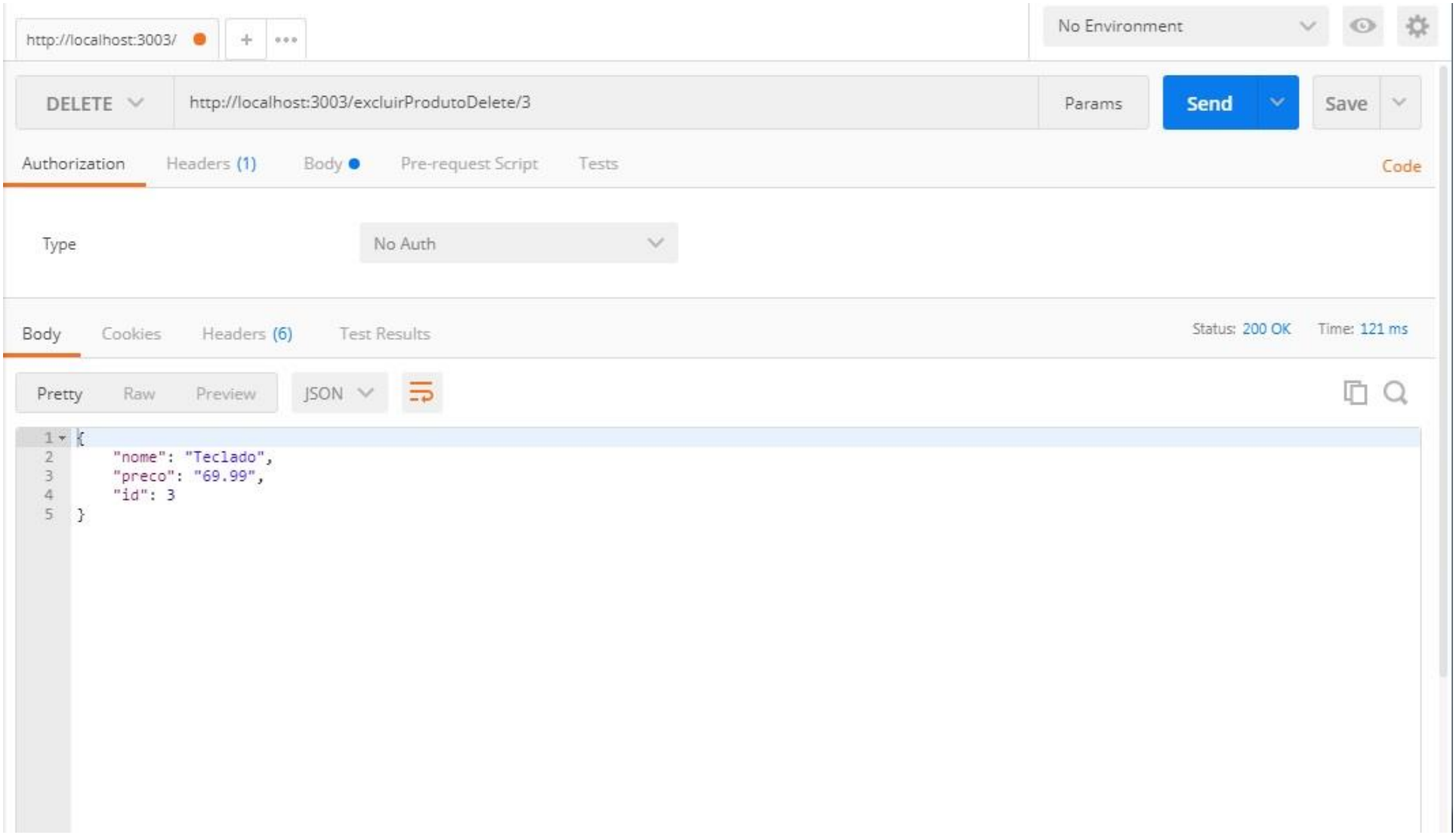
MODIFICANDO O PREÇO DO PRODUTO 2...



PESQUISANDO O PRODUTO 3...



DELETANDO SOMENTE O PRODUTO 3...



VISUALIZANDO LISTA DEPOIS QUE O PRODUTO 3 FOI EXCLUÍDO...