

Documentação API teste AGLETS

Descrição

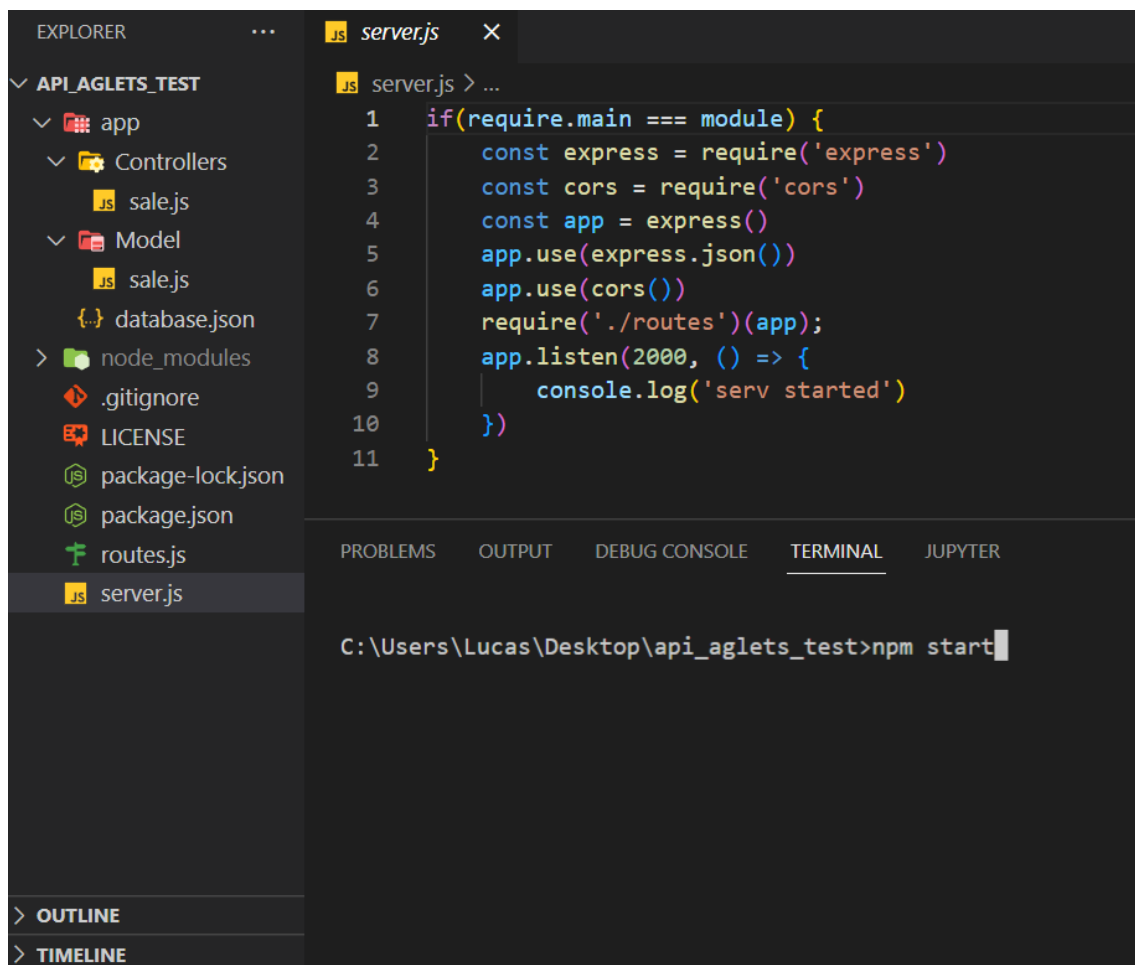
A presente API foi criada com o objetivo de demonstrar nível de conhecimento na criação e configuração de uma API utilizando como base em seu Backend o NodeJS.

A mesma foi criada em cima do modelo MVC, com a ausência da interação com o Frontend. E um banco de dados, representado no exemplo por um arquivo Json. Portanto todos os testes serão demonstrados através do Postman.

Por se tratar de um projeto mais básico o mesmo não possui validação de acesso e nenhuma checagem de campos no Header ou Session, além de não possuir condicionais para a formatação dos campos.

Iniciar

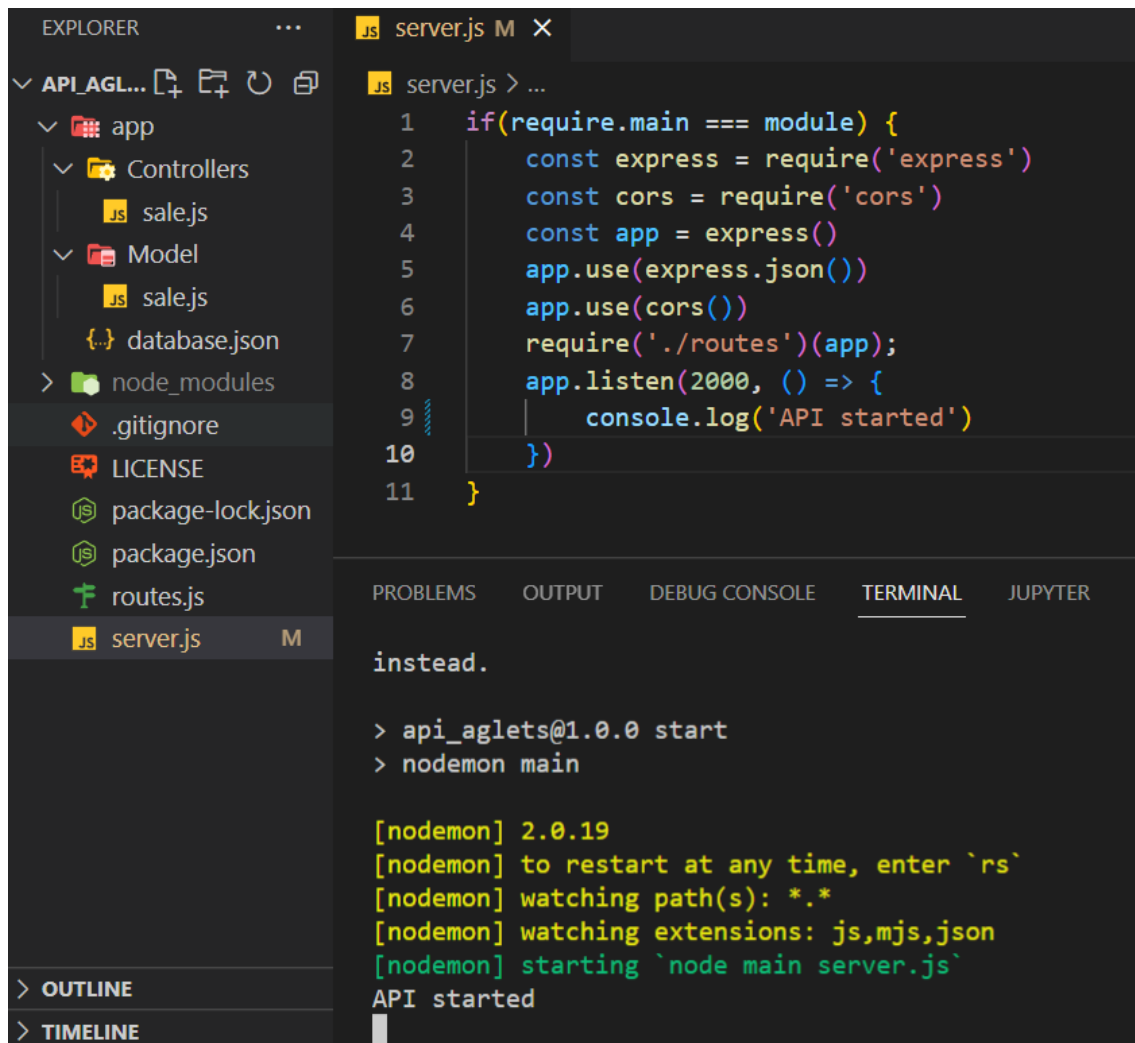
Após a abertura do projeto em sua IDE, neste exemplo usando o Visual Studio Code, rode o comando “**npm start**” em seu terminal para colocar em funcionamento a aplicação, como mostra a imagem.

The image is a screenshot of the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'API_AGLETS_TEST'. Inside this project, there is an 'app' folder containing 'Controllers' (with a 'sale.js' file) and a 'Model' (with a 'sale.js' file and a 'database.json' file). Other files in the root include 'node_modules', '.gitignore', 'LICENSE', 'package-lock.json', 'package.json', 'routes.js', and 'server.js'. The 'server.js' file is selected and open in the main editor. The code in 'server.js' is as follows:

```
1  if(require.main === module) {
2      const express = require('express')
3      const cors = require('cors')
4      const app = express()
5      app.use(express.json())
6      app.use(cors())
7      require('./routes')(app);
8      app.listen(2000, () => {
9          console.log('serv started')
10     })
11 }
```

At the bottom of the interface, the TERMINAL panel is active, showing the command 'C:\Users\Lucas\Desktop\api_aglets_test>npm start' being entered.

Após isso a mensagem “**API started**” deve ser mostrada para indicar o correto funcionamento da mesma.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure: API_AGL... (with icons for file operations), app (expanded), Controllers (with sale.js), Model (with sale.js and database.json), node_modules, .gitignore, LICENSE, package-lock.json, package.json, routes.js, and server.js (selected). The main editor area shows the content of server.js, which is a Node.js server using Express and CORS. The code is as follows:

```
1  if(require.main === module) {
2      const express = require('express')
3      const cors = require('cors')
4      const app = express()
5      app.use(express.json())
6      app.use(cors())
7      require('./routes')(app);
8      app.listen(2000, () => {
9          console.log('API started')
10     })
11 }
```

At the bottom, the TERMINAL panel shows the command prompt output:

```
instead.

> api_aglets@1.0.0 start
> nodemon main

[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node main server.js`
API started
```

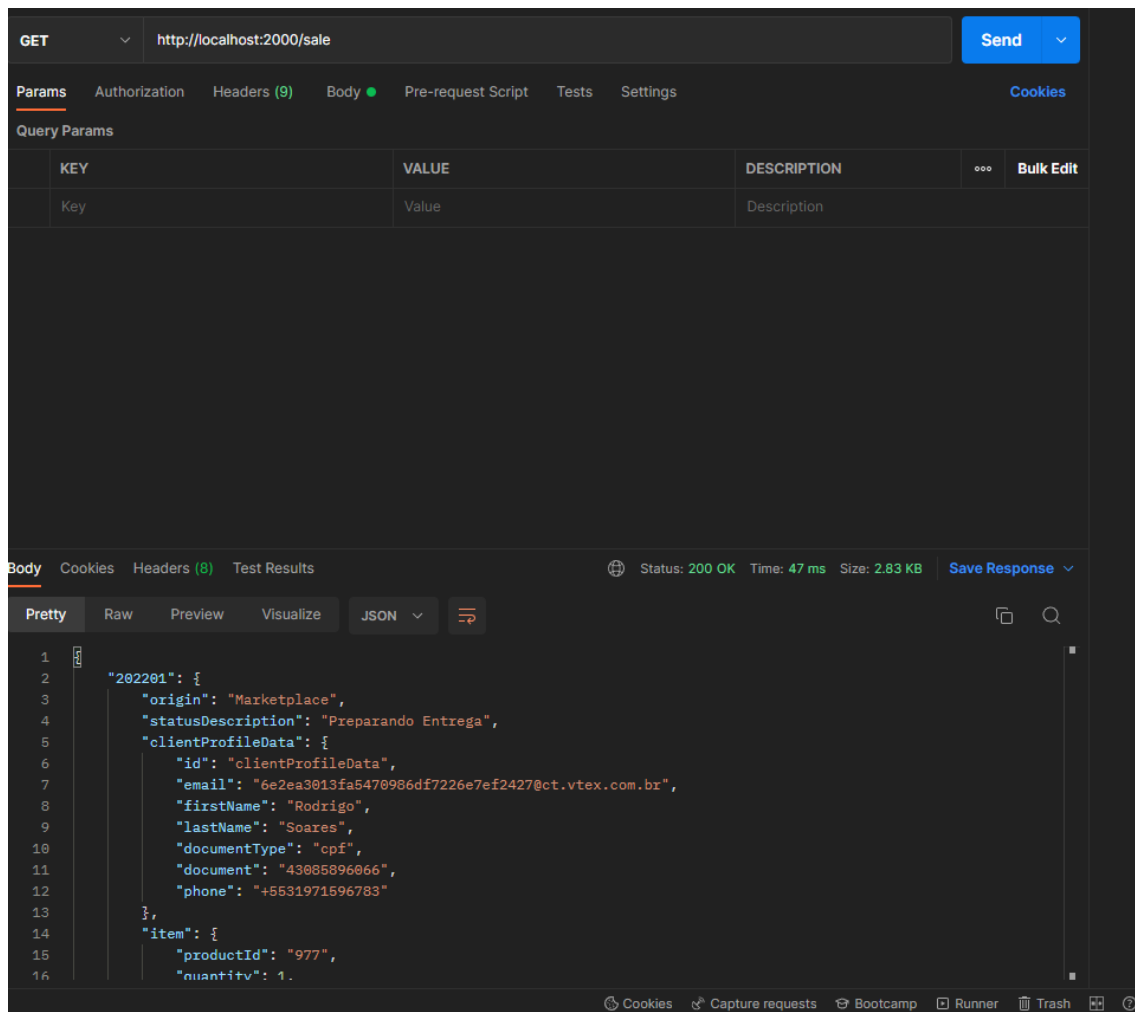
Testes

Após esta mensagem a aplicação estará disponível para acesso na rede local.

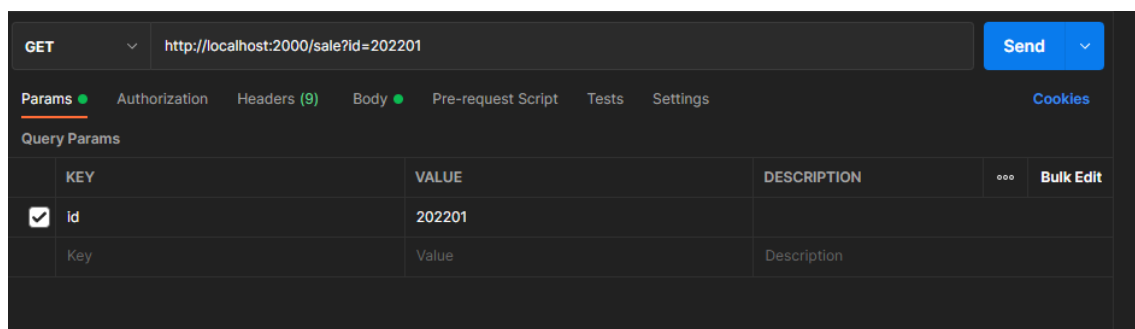
Para garantir a integridade da aplicação para futuros teste todas alterações na base de dados são realizadas somente no ambiente de funcionamento da aplicação, o arquivo inicial não é reescrito. Portanto ao reiniciar a aplicação a mesma voltará ao seu banco de dados original.

GET

Para realizar uma requisição de todos os dados de vendas presentes no banco de dados basta realizar uma solicitação de GET ao link “**http://localhost:2000/sale**”.



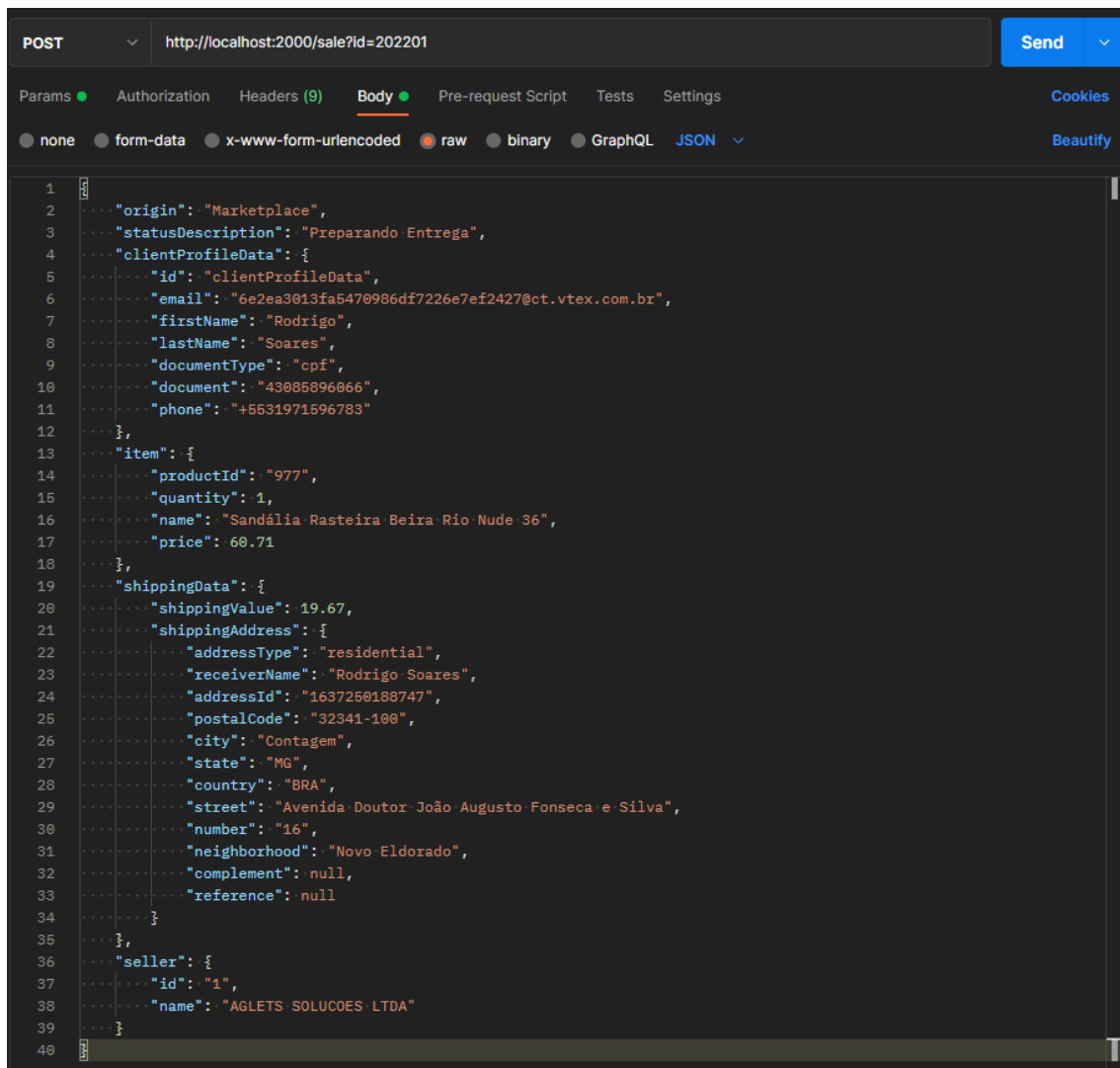
É possível realizar a consulta de um ID específico enviando o mesmo como parâmetro da requisição com a Key sendo “id” e o “Value” sendo o valor do mesmo.



Ao enviar um ID não existente no banco a mensagem “ID not found” é retornada.

POST

Para realizar uma inserção no banco de dados deve-se realizar uma requisição com o método **POST** enviando no **raw** do body o Json a ser inserido conforme o modelo apresentado.



Note que não é necessário enviar o ID bem como o valor total ou as datas de criação e modificação. As mesmas devem ser geradas pela aplicação e inseridas de forma automática.

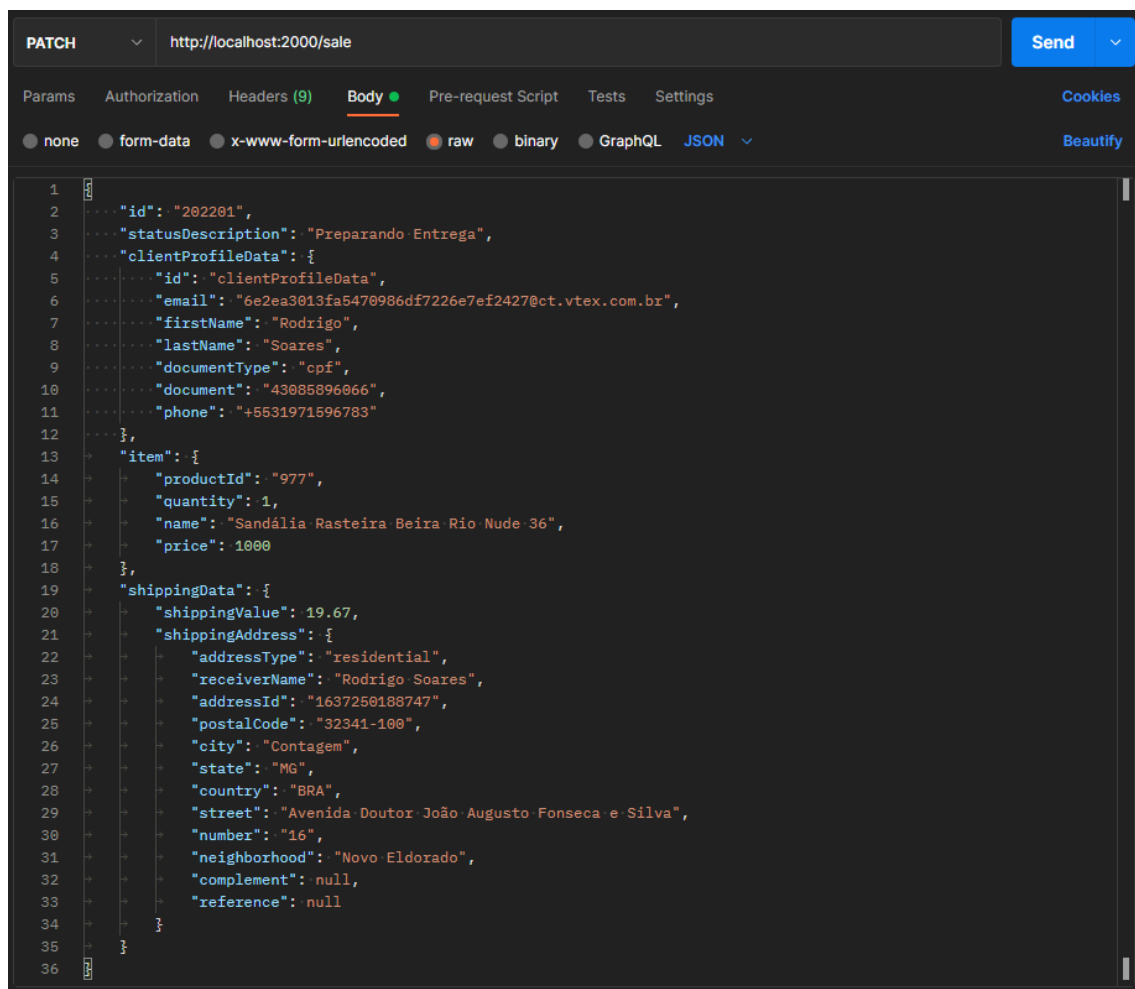
MODELO

```
{
  "origin": "Marketplace",
  "statusDescription": "Preparando Entrega",
  "clientProfileData": {
    "id": "clientProfileData",
    "email": "6e2ea3013fa5470986df7226e7ef2427@ct.vtex.com.br",
    "firstName": "Rodrigo",
    "lastName": "Soares",
    "documentType": "cpf",
    "document": "43085896066",
```

```
    "phone": "+5531971596783"
  },
  "item": {
    "productId": "977",
    "quantity": 1,
    "name": "Sandália Rasteira Beira Rio Nude 36",
    "price": 60.71
  },
  "shippingData": {
    "shippingValue": 19.67,
    "shippingAddress": {
      "addressType": "residential",
      "receiverName": "Rodrigo Soares",
      "addressId": "1637250188747",
      "postalCode": "32341-100",
      "city": "Contagem",
      "state": "MG",
      "country": "BRA",
      "street": "Avenida Doutor João Augusto Fonseca e Silva",
      "number": "16",
      "neighborhood": "Novo Eldorado",
      "complement": null,
      "reference": null
    }
  },
  "seller": {
    "id": "1",
    "name": "AGLETS SOLUCOES LTDA"
  }
}
```

PATCH

Para realizar a atualização de algum registro basta realizar a solicitação pelo método PATCH enviando no campo **raw** do body o Json como mostra o modelo.



Note que é necessário como primeiro campo do Json o ID do registro a ser editado.

Qualquer chave primaria (clientProfileData, shippingData, etc) pode ser adicionada ou removida sem afetar o funcionamento da aplicação. Ao não ser enviada uma chave mantém o seu valor original.

MODELO

```
{
  "id": "202201",
  "statusDescription": "Preparando Entrega",
  "clientProfileData": {
    "id": "clientProfileData",
    "email": "6e2ea3013fa5470986df7226e7ef2427@ct.vtex.com.br",
```

```
"firstName": "Rodrigo",
"lastName": "Soares",
"documentType": "cpf",
"document": "43085896066",
"phone": "+5531971596783"
},
"item": {
  "productId": "977",
  "quantity": 1,
  "name": "Sandália Rasteira Beira Rio Nude 36",
  "price": 1000
},
"shippingData": {
  "shippingValue": 19.67,
  "shippingAddress": {
    "addressType": "residential",
    "receiverName": "Rodrigo Soares",
    "addressId": "1637250188747",
    "postalCode": "32341-100",
    "city": "Contagem",
    "state": "MG",
    "country": "BRA",
    "street": "Avenida Doutor João Augusto Fonseca e Silva",
    "number": "16",
    "neighborhood": "Novo Eldorado",
    "complement": null,
    "reference": null
  }
}
}
```

DELETE

Para deletar algum registro basta enviar o ID do mesmo como parâmetro da requisição com o método **DELETE**.

The screenshot displays a REST client interface with a DELETE request configured. The URL is `http://localhost:2000/sale?id=202201`. The 'Params' tab is active, showing a query parameter 'id' with the value '202201'. The response status is '200 OK' with a time of '7 ms' and a size of '315 B'. The response body is shown in 'Pretty' JSON format.

DELETE `http://localhost:2000/sale?id=202201` **Send**

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	id	202201			
	Key	Value	Description		

Body Cookies Headers (8) Test Results **Status: 200 OK** Time: 7 ms Size: 315 B **Save Response**

Pretty Raw Preview Visualize **JSON**

```
1  {
2    "message": "Deleted successfully",
3    "id": "202201"
4  }
```