

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE
SÃO PAULO**

Gustavo Massamichi Nakamura (SP309863X)

Programação Dinâmica para Web (PDWA5)

Projeto de criação de API

SÃO PAULO

2023

Gustavo Massamichi Nakamura (SP309863X)

Sebo Online S.A.

Primeira Entrega

Documentação do trabalho de Programação Dinâmica
para Web (5º semestre) no Instituto Federal de Educação,
Ciência e Tecnologia de São Paulo ministrado pelo
professor Luiz Fernando Postingel Quirino

SÃO PAULO

2023

Sumário

1 - Introdução	4
2 – Objetivo dessa 1ª entrega.....	4
3 - Tecnologias	5
4 – Provas de objetivos cumpridos e requisições com respostas	5

1 - Introdução

Na aula de programação dinâmica para web ministrada pelo professor Quirino no Instituto Federal de Educação, Ciência e Tecnologia de São Paulo foi requerido o projeto que aqui está sendo documentado. Esse projeto tinha como apresentação de dificuldade a seguinte descrição:

“Joana sempre foi apaixonada por livros. Desde pequena, seu lugar preferido era o SEBO da esquina, onde ela podia encontrar preciosidades literárias e histórias esquecidas pelo tempo. Com o avanço da tecnologia, muitos desses SEBOs fecharam as portas, e Joana viu sua paixão se tornar cada vez mais rara nas cidades. Porém, ao invés de lamentar, ela teve uma ideia: por que não trazer o conceito de SEBO para o mundo online?”

Assim nasceu o “Sebo Online S.A.”, uma plataforma digital onde qualquer pessoa poderia vender ou comprar livros usados, garantindo que as histórias continuassem sendo compartilhadas e lembradas.

Joana acredita que a essência de um SEBO não está apenas nos livros, mas nas pessoas que os leem e os vendem. Por isso, ela quer que o sistema permita a interação entre os usuários, e que cada livro tenha sua própria “história”, contada por quem o está vendendo.”

Dado a dificuldade anterior, foi necessário desenvolver uma API robusta e eficiente para o Sebo Online S.A. Para essa primeira entrega eu estabeleci a base do sistema, focando principalmente na gestão de usuários.

2 – Objetivo dessa 1ª entrega

-> Objetivo

- Estabelecer base do sistema, focando principalmente na gestão de usuários

-> Requisitos

- Back-End Base

- Configuração inicial do servidor.
- Definição e Conexão com banco de dados.
- Implementação da estrutura básica de roteamento.

- Usuários

- Cadastro de usuários (compradores e vendedores).
- Autenticação (login/logout).
- Edição de perfil.
- Soft delete para desativar usuários.
- Sistema de criptografia para senhas.

- Administradores

- Login de administradores.
- Visualização básica de relatórios e estatísticas (focar no retorno de um endpoint ao menos, sugestão, pelo endpoint de listagem de usuários, pode ser aprimorado nas próximas entregas).

3 - Tecnologias

- Front-End:
 - HTML;
 - CSS;
- Back-End:
 - Node.js;
- Banco de dados:
 - MySQL Workbench;
- API:
 - Postman (necessário instalação do postman agent no seu desktop);
 - Thunder Client (instalação feita no VS Code);
- IDE:
 - Visual Studio Code;
- Repositório:
 - github - > https://github.com/gmichin/Sebo_Online_S.A

4 – Provas de objetivos cumpridos e requisições com respostas

Back-End Base (configuração inicial do servidor, definição e conexão com banco de dados, implementação da estrutura básica de roteamento):

Código no MySQL Workbench, no repositório estará dentro da pasta “db”:

```
CREATE DATABASE sebo_online;

USE sebo_online;

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  senha VARCHAR(255) NOT NULL,
  status ENUM('ativo', 'inativo') NOT NULL,
  tipo ENUM('comprador', 'vendedor', 'administrador') NOT NULL,
  area_especializacao VARCHAR(255),
  token VARCHAR(10) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

CREATE TABLE admin (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  senha VARCHAR(255) NOT NULL,
  status ENUM('ativo', 'inativo') NOT NULL,
  tipo ENUM('comprador', 'vendedor', 'administrador') NOT NULL,
  area_especializacao VARCHAR(255),
  token VARCHAR(10) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  dataInicio TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

DELIMITER //
CREATE TRIGGER generate_token_before_insert
BEFORE INSERT ON users
FOR EACH ROW
BEGIN
  SET NEW.token = SUBSTRING(MD5(RAND()), 1, 10);
END;
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER generate_token_before_insert_admin
BEFORE INSERT ON admin
FOR EACH ROW
BEGIN
  SET NEW.token = SUBSTRING(MD5(RAND()), 1, 10);
END;
//
DELIMITER ;
```

DB.JS (olhar instruções comentadas na imagem abaixo):

```
const mysql = require('mysql2');

const db = mysql.createConnection({
  host: '127.0.0.1', //host da sua conexão no MySQL Workbench
  user: 'gustavo', //Seu user da conexão MySQL Workbench
  password: 'juliemei2014', //sua senha da conexão MySQL Workbench
  database: 'sebo_online', //Sua database do MySQL Workbench
});

db.connect((err) => {
  if (err) {
    console.error('Erro ao conectar ao banco de dados:', err);
    return;
  }
  console.log('Conectado ao banco de dados MySQL');
});

module.exports = db;
```

APP.JS:

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;
const path = require('path');

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use(express.static(path.join(__dirname, 'public')));

app.use('/path', require('./routes/path'));

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

app.listen(port, () => {
  console.log(`\nServidor rodando: http://localhost:\${port}`);
});
```

Nota: Para rodar o programa, é necessário escrever no terminal “node app.js”;

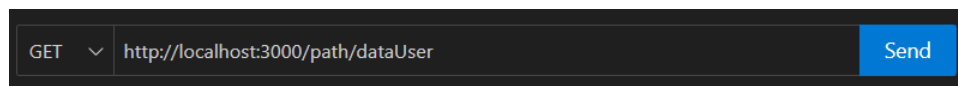
SOBRE TODAS AS REQUISIÇÕES, ELAS ESTÃO CONFIGURADAS EM “ROUTES/PATH.JS” DO ARQUIVO DO GITHUB

Visualização básica de relatórios e estatísticas (focar no retorno de um endpoint ao menos, sugestão, pelo endpoint de listagem de usuários, pode ser aprimorado nas próximas entregas):

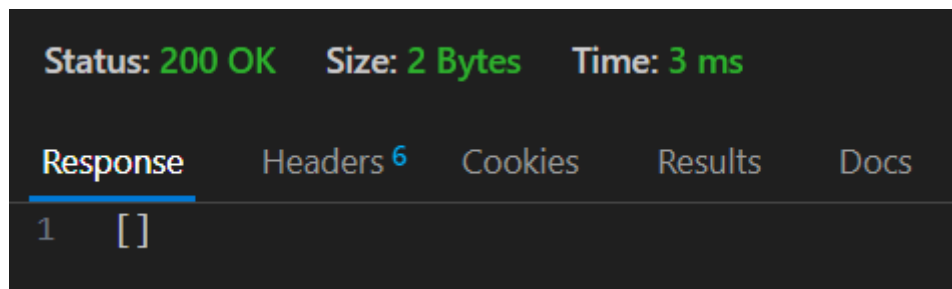
GET – retornar usuários

Link: <http://localhost:3000/path/dataUser>

THUNDER CLIENT



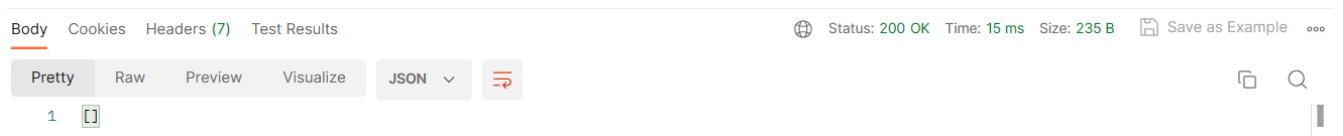
Resposta:



POSTMAN



Resposta:

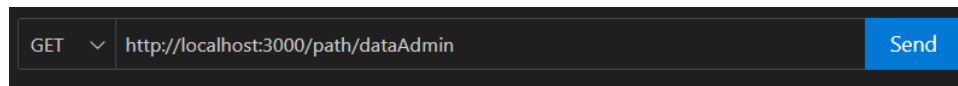


Nota: Essas respostas são de antes de ser feito qualquer cadastro;

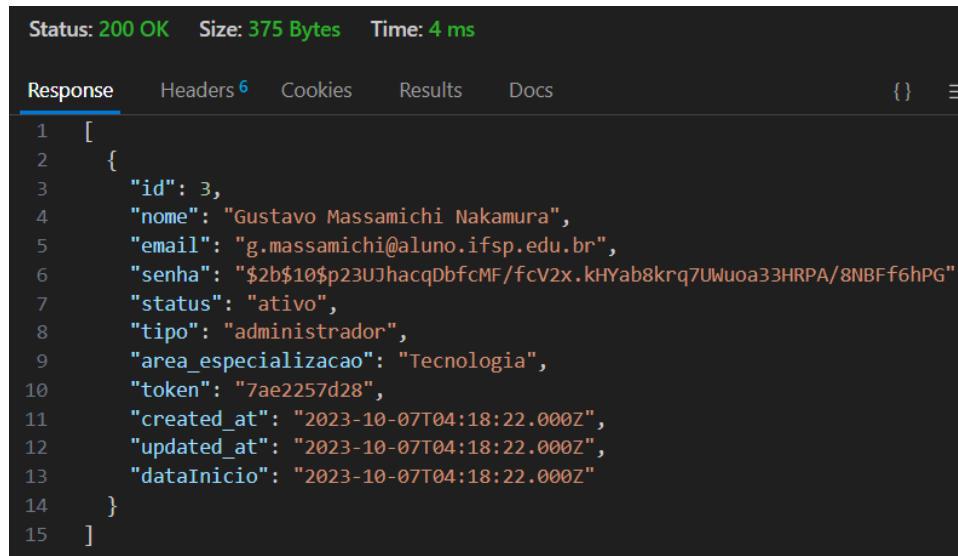
GET – retornar administradores

Link: <http://localhost:3000/path/dataAdmin>

THUNDER CLIENT



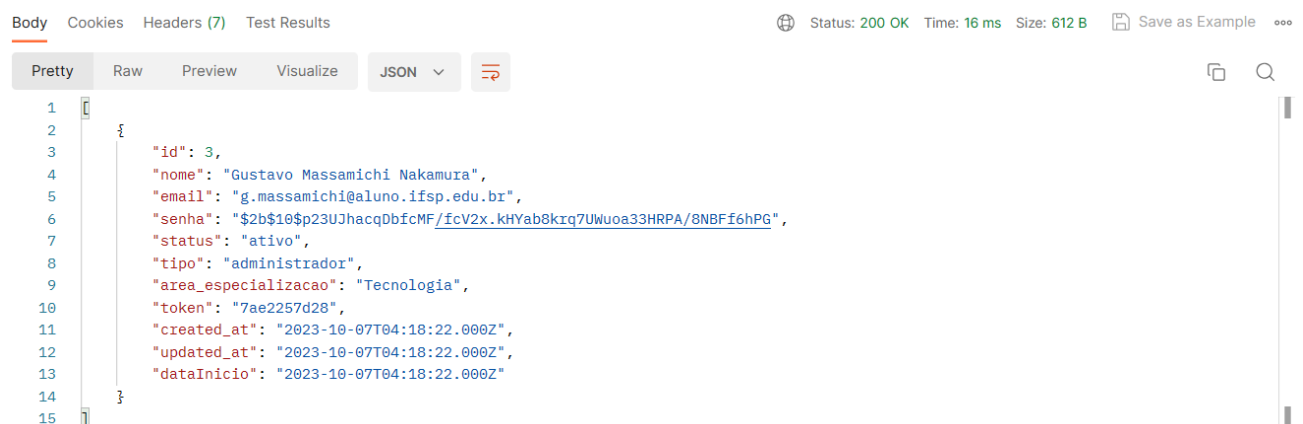
Resposta:



POSTMAN



Resposta:



Nota: Essas respostas são de antes de ser feito qualquer cadastro;

Cadastro de usuários (compradores e vendedores):

POST - cadastro de usuários e administradores:

Link: <http://localhost:3000/path/signup>

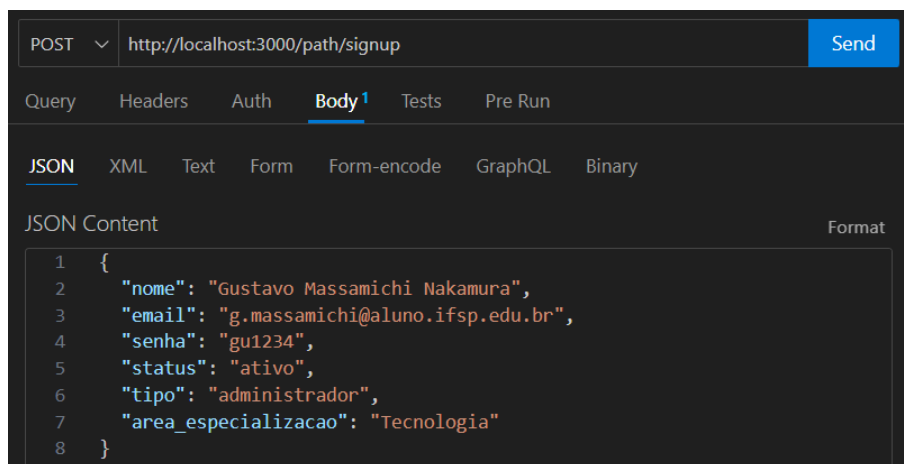
Body (json):

```
{
  "nome": "Inserir seu nome",
  "email": "Inserir seu email",
  "senha": "Inserir sua senha",
  "status": "Escolha entre ativo e inativo",
  "tipo": "Escolha entre comprador, vendedor e administrador",
  "area_especializacao": "Insira sua área de especialização"
}
```

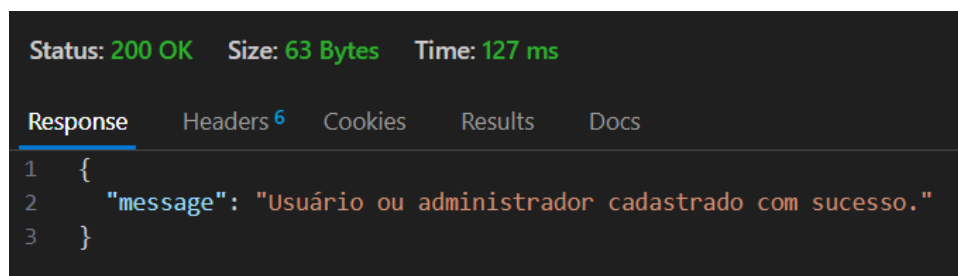
Nota 1: Ao escolher o "tipo" no json acima ele irá cadastrar automaticamente na tabela (table do SQL) de "admin" ou de "users" dependendo se o tipo for vendedor, comprador ou administrador;

Nota 2: Nessa mesma requisição podemos cadastrar tanto usuários quanto administradores;

THUNDER CLIENT



Resposta:



GET após um adm cadastrado - não é o mesmo adm da imagem de requisição
POST cadastro anterior

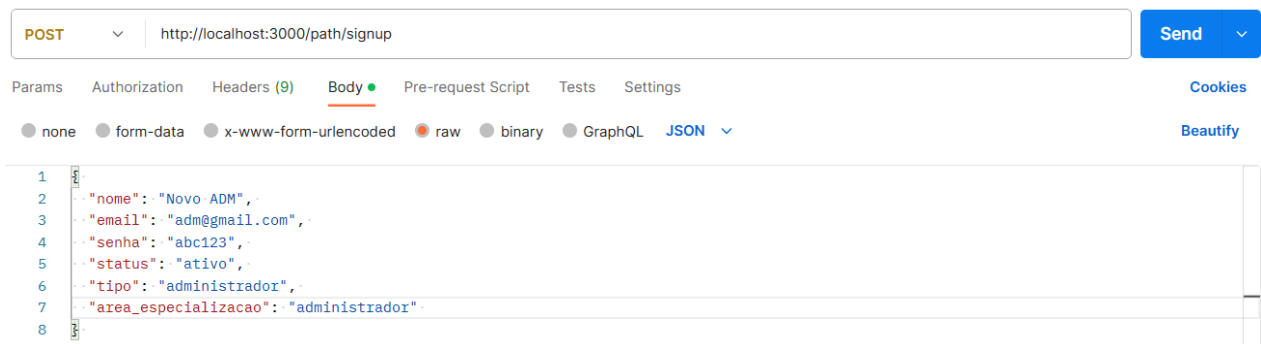
```
Status: 200 OK   Size: 725 Bytes   Time: 5 ms

Response  Headers 6  Cookies  Results  Docs  {}  ≡
12  updated_at: "2023-10-07T04:18:22.000Z",
13  "dataInicio": "2023-10-07T04:18:22.000Z"
14  },
15  {
16    "id": 4,
17    "nome": "Usuário Adm",
18    "email": "usuarioAdm@gmail.com",
19    "senha": "$2b$10$.tQ8OVmymPMx/AY/S9/gSuy5Yk7dCVwi0Df6NA7x4MnhC5/zT6yK.",
20    "status": "ativo",
21    "tipo": "administrador",
22    "area_especializacao": "Tecnologia",
23    "token": "2b99da0dd1",
24    "created_at": "2023-10-07T05:10:22.000Z",
25    "updated_at": "2023-10-07T05:10:22.000Z",
26    "dataInicio": "2023-10-07T05:10:22.000Z"
27  }
28  ]
```

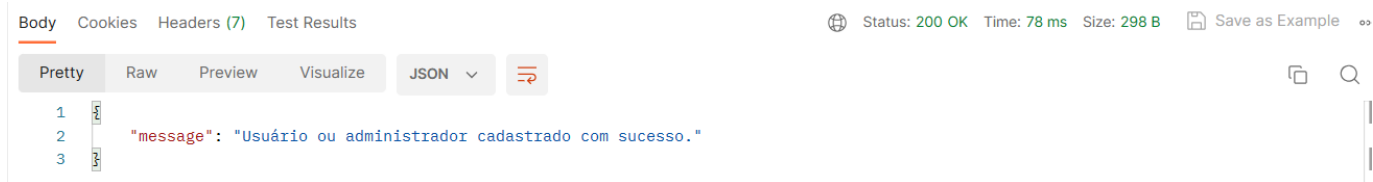
GET após um usuário cadastrado - não é o mesmo usuário da imagem de
requisição POST cadastro anterior

```
[
  {
    "id": 6,
    "nome": "Usuário Comum",
    "email": "usuario@gmail.com",
    "senha": "$2b$10$bDsS.Y9rCcWrgby7e61YB.MuT1E20rVJzSvY3n051k0BADQLsSXSm",
    "status": "ativo",
    "tipo": "comprador",
    "area_especializacao": "Tecnologia",
    "token": "7813a45506",
    "created_at": "2023-10-07T05:02:10.000Z",
    "updated_at": "2023-10-07T05:02:10.000Z"
  }
]
```

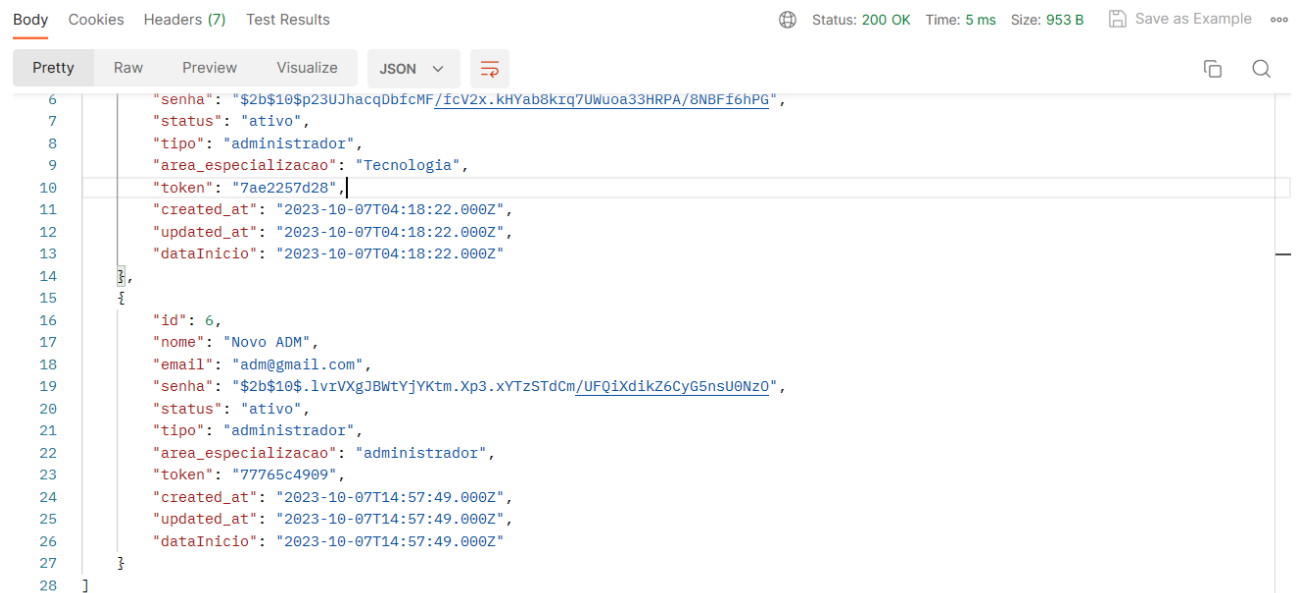
POSTMAN



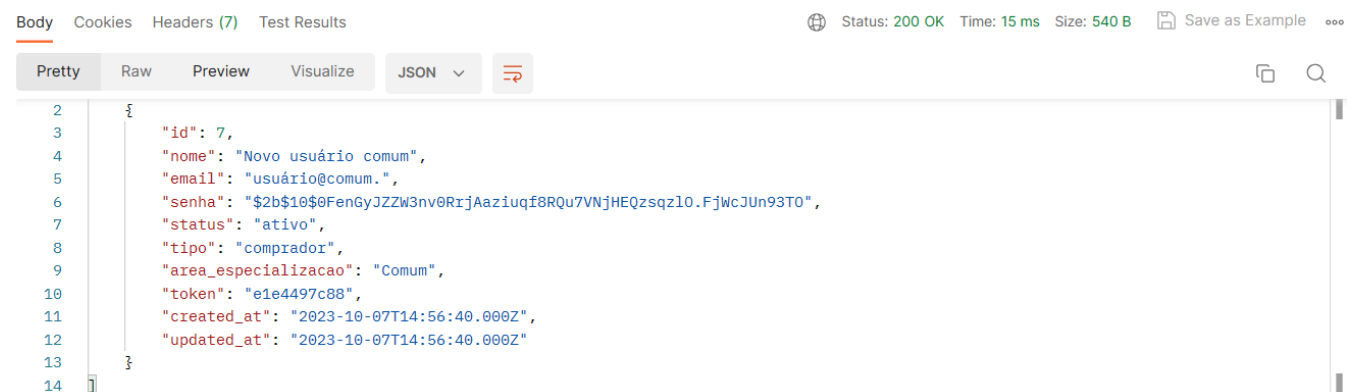
Resposta:



GET após um adm cadastrado



GET após um usuário cadastrado - não é o mesmo usuário da imagem de requisição POST cadastro anterior



Autenticação (login/logout) de usuários e de administradores:

POST - login de usuários e administradores

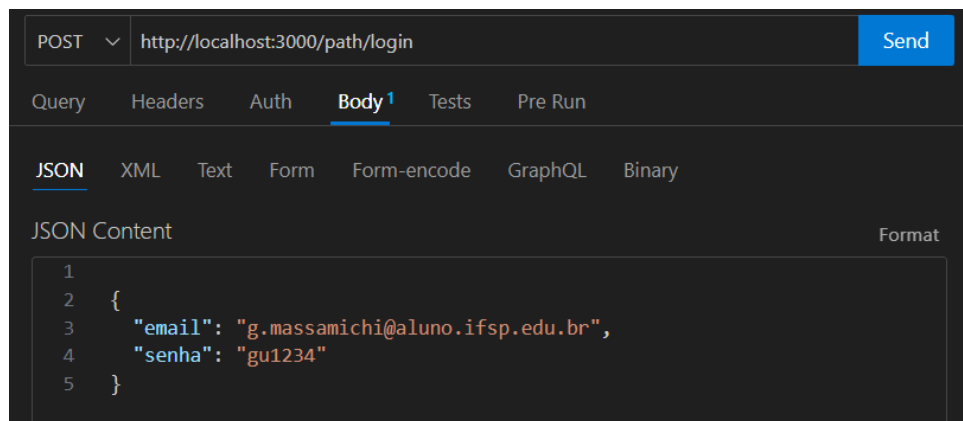
Link: <http://localhost:3000/path/login>

Body (json):

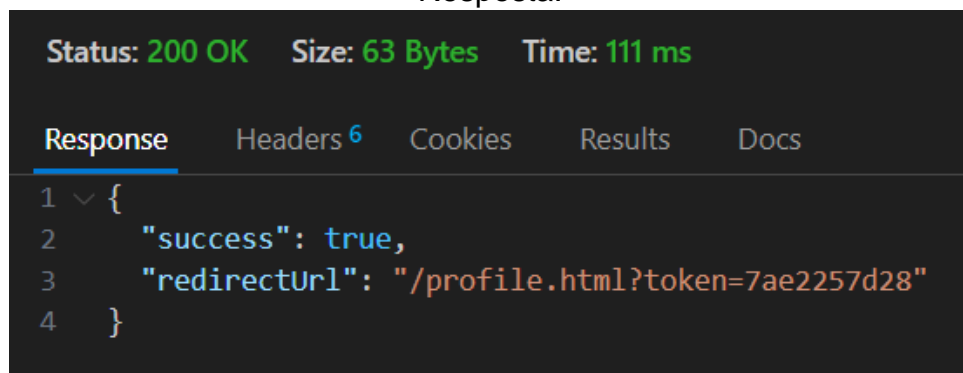
```
{
  "email": "Inserir seu email",
  "senha": "Inserir sua senha"
}
```

Nota: O login de usuários e de adms são feitos nessa mesma requisição;

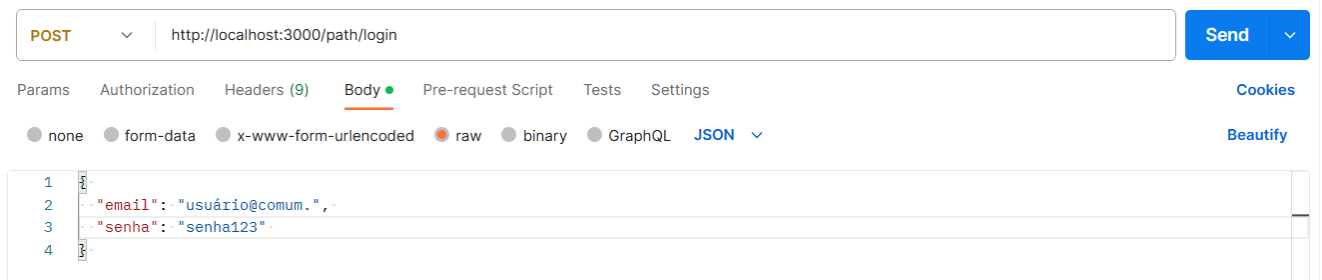
THUNDER CLIENT



Resposta:



POSTMAN



Resposta:



Nota: Como visto nos exemplos de GET anteriores, as senhas estão criptografadas, mas eu posso inserir a senha que escolhi (não criptografada);

Edição de perfil:

PUT - Edição de usuário comum

Link: [http://localhost:3000/path/users/\(token_usuario\)](http://localhost:3000/path/users/(token_usuario))

Body (json):

```
{
  "nome": "Exemplo de edição de perfil de Usuário",
  "email": "com metodo put",
  "status": "ativo",
  "tipo": "vendedor",
  "area_especializacao": "teste",
  "senha": "senha"
}
```

Nota: O token pode ser adquirido com a requisição GET mostrada anteriormente;

THUNDER CLIENT



Resposta:

```
Status: 200 OK   Size: 53 Bytes   Time: 102 ms

Response  Headers 6  Cookies  Results  Docs
1  {
2    "message": "Perfil de vendedor editado com sucesso."
3  }
```

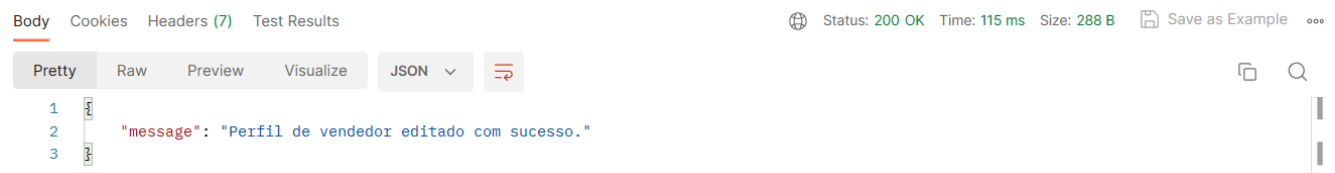
GET do usuário após edição:

```
[
  {
    "id": 6,
    "nome": "Exemplo de edição de perfil de Usuário",
    "email": "com metodo put",
    "senha": "$2b$10$N8bJs2lT6VCGl4b1RaYtO.wfbFXxA6NaFrSdahgnH.1Q0TUlvsxqu",
    "status": "ativo",
    "tipo": "vendedor",
    "area_especializacao": "teste",
    "token": "7813a45506",
    "created_at": "2023-10-07T05:02:10.000Z",
    "updated_at": "2023-10-07T05:07:48.000Z"
  }
]
```

POSTMAN



Resposta:



GET do usuário após edição:

The screenshot shows a web client interface with tabs for Body, Cookies, Headers (7), and Test Results. The 'Body' tab is active, displaying a JSON response in 'Pretty' format. The response is a successful PUT request with a status of 200 OK, a time of 5 ms, and a size of 560 B. The JSON object contains the following fields: id (7), nome (Exemplo de edição de perfil de Usuário), email (com metodo put), senha (\$2b\$10\$6aB0KRHs8SBm1F3jBV0FT07Y3cIsGvTiwR6Srdgp1m7aAVK0IwoJ2), status (ativo), tipo (vendedor), area_especializacao (teste), token (e1e4497c88), created_at (2023-10-07T14:56:40.000Z), and updated_at (2023-10-07T15:24:15.000Z).

```
1 {
2   "id": 7,
3   "nome": "Exemplo de edição de perfil de Usuário",
4   "email": "com metodo put",
5   "senha": "$2b$10$6aB0KRHs8SBm1F3jBV0FT07Y3cIsGvTiwR6Srdgp1m7aAVK0IwoJ2",
6   "status": "ativo",
7   "tipo": "vendedor",
8   "area_especializacao": "teste",
9   "token": "e1e4497c88",
10  "created_at": "2023-10-07T14:56:40.000Z",
11  "updated_at": "2023-10-07T15:24:15.000Z"
12 }
13
```

Nota: O token pode ser adquirido com a requisição GET mostrada anteriormente;

PUT - Edição de administrador para comprador

Link: [http://localhost:3000/path/admin/\(token_adm\)](http://localhost:3000/path/admin/(token_adm))

Body (json):

```
{
  "nome": "Exemplo de edição de perfil de Administrador",
  "email": "com metodo put",
  "status": "ativo",
  "tipo": "comprador",
  "area_especializacao": "teste",
  "senha": "senha"
}
```

Nota: O token pode ser adquirido com a requisição GET mostrada anteriormente;

THUNDER CLIENT

The screenshot shows the Thunder Client interface. The top bar displays the method 'PUT' and the URL 'http://localhost:3000/path/admin/2b99da0dd1'. The 'Send' button is visible. Below the top bar, there are tabs for Query, Headers, Auth, Body 1, Tests, and Pre Run. The 'Body 1' tab is active, showing the JSON content. The JSON is formatted and includes the following fields: nome (Exemplo de edição de perfil de Administrador), email (com metodo put), status (ativo), tipo (comprador), area_especializacao (teste), and senha (senha).

```
1 {
2   "nome": "Exemplo de edição de perfil de Administrador",
3   "email": "com metodo put",
4   "status": "ativo",
5   "tipo": "comprador",
6   "area_especializacao": "teste",
7   "senha": "senha"
8 }
```

Resposta:


```
Status: 200 OK   Size: 54 Bytes   Time: 309 ms

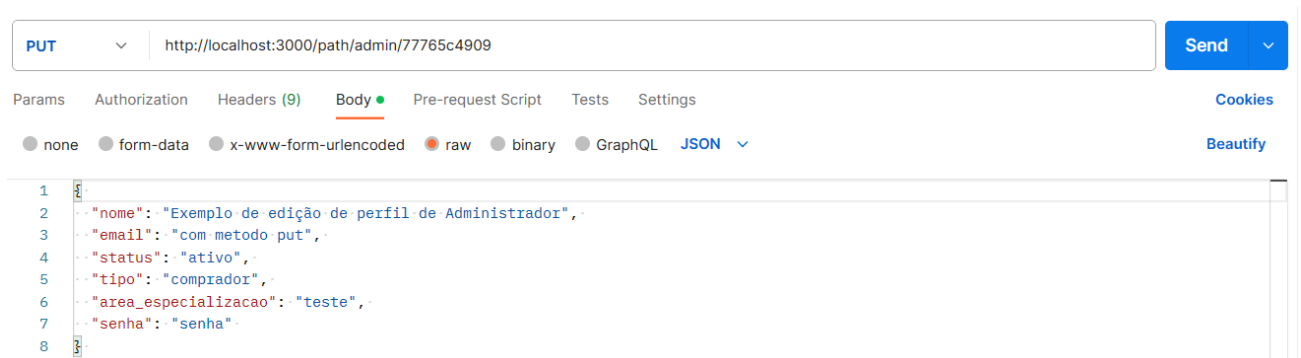
Response  Headers 6  Cookies  Results  Docs
1  {
2    "message": "Perfil de comprador editado com sucesso."
3  }
```

GET do admin após edição:

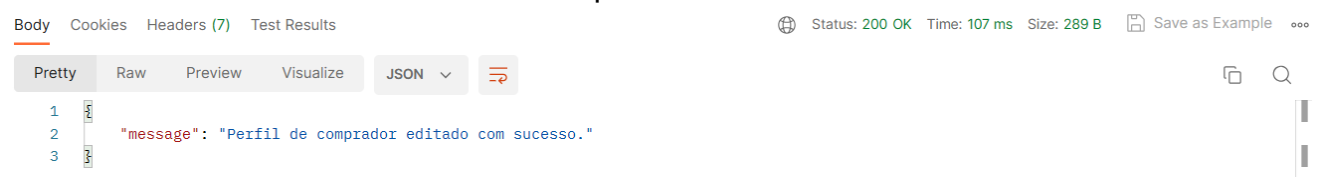
```
Status: 200 OK   Size: 743 Bytes   Time: 4 ms

Response  Headers 6  Cookies  Results  Docs  {}  ≡
12  "updated_at": "2023-10-07T04:10:22.000Z",
13  "dataInicio": "2023-10-07T04:18:22.000Z"
14  },
15  {
16    "id": 4,
17    "nome": "Exemplo de edição de perfil de Administrador",
18    "email": "com metodo put",
19    "senha": "$2b$10$WNvbeuh6H8s9Ijqjg25HZ.C5zl0QbUli5lQh.G1rxP3.RsMXTYPWa",
20    "status": "ativo",
21    "tipo": "comprador",
22    "area_especializacao": "teste",
23    "token": "2b99da0dd1",
24    "created_at": "2023-10-07T05:10:22.000Z",
25    "updated_at": "2023-10-07T05:15:51.000Z",
26    "dataInicio": "2023-10-07T05:10:22.000Z"
27  }
28  ]
```

POSTMAN



Resposta:



GET do admin após edição:

```
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 7 ms Size: 980 B Save as Example

Pretty Raw Preview Visualize JSON
{
  "senha": "$2b$10$p23UJhacq0bfcMF/fcV2x.kHYab8Kiq7UWuoa33HRPA/8NBFf6hPG",
  "status": "ativo",
  "tipo": "administrador",
  "area_especializacao": "Tecnologia",
  "token": "7ae2257d28",
  "created_at": "2023-10-07T04:18:22.000Z",
  "updated_at": "2023-10-07T04:18:22.000Z",
  "dataInicio": "2023-10-07T04:18:22.000Z"
},
{
  "id": 6,
  "nome": "Exemplo de edição de perfil de Administrador",
  "email": "com metodo put",
  "senha": "$2b$10$qJix05cdKgcwSsA/KkuM7ehAtbGYG3c7KQnbybS0hx8ZX5WdCXQn6",
  "status": "ativo",
  "tipo": "comprador",
  "area_especializacao": "teste",
  "token": "77765c4909",
  "created_at": "2023-10-07T14:57:49.000Z",
  "updated_at": "2023-10-07T15:29:57.000Z",
  "dataInicio": "2023-10-07T14:57:49.000Z"
}
```

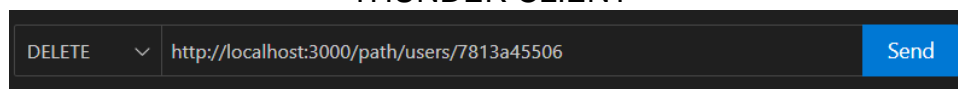
Soft delete para desativar usuários:

DELETE - Deletar usuários

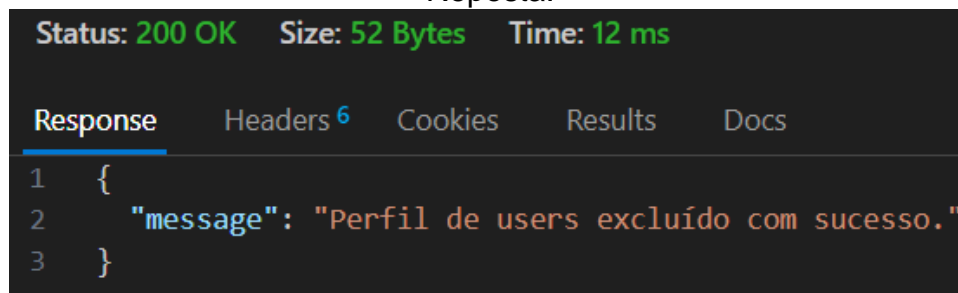
Link: [http://localhost:3000/path/users/\(token_usuario\)](http://localhost:3000/path/users/(token_usuario))

Nota: O token pode ser adquirido com a requisição GET mostrada anteriormente;

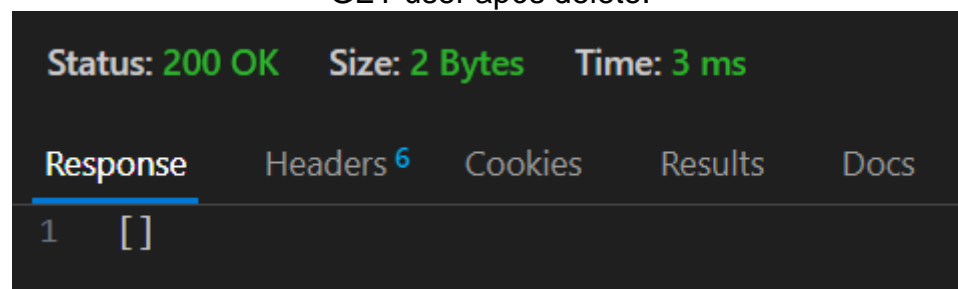
THUNDER CLIENT



Reposta:



GET user após delete:



POSTMAN

DELETE

http://localhost:3000/path/users/e1e4497c88

Send

Resposta:

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 9 ms Size: 287 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Perfil de users excluído com sucesso."
3 }
```

GET user após delete:

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 17 ms Size: 235 B Save as Example

Pretty Raw Preview Visualize JSON

```
1 {}
```

DELETE – Deletar administradores

Link: [http://localhost:3000/path/admin/\(token_admin\)](http://localhost:3000/path/admin/(token_admin))

Nota: O token pode ser adquirido com a requisição GET mostrada anteriormente;

THUNDER CLIENT (válido no postman)

DELETE

http://localhost:3000/path/admin/2b99da0dd1

Send

Resposta:

Status: 200 OK Size: 52 Bytes Time: 11 ms

Response Headers 6 Cookies Results Docs

```
1 {
2   "message": "Perfil de admin excluído com sucesso."
3 }
```

GET adm após delete:

Status: 200 OK Size: 375 Bytes Time: 4 ms

Response Headers 6 Cookies Results Docs

```
1 [
2   {
3     "id": 3,
4     "nome": "Gustavo Massamichi Nakamura",
5     "email": "g.massamichi@aluno.ifsp.edu.br",
6     "senha": "$2b$10$p23UJhacqDbfcMF/fcV2x.kHYab8krq7UWuo33HRPA/8NBFf6hPG",
7     "status": "ativo",
8     "tipo": "administrador",
9     "area_especializacao": "Tecnologia",
10    "token": "7ae2257d28",
11    "created_at": "2023-10-07T04:18:22.000Z",
12    "updated_at": "2023-10-07T04:18:22.000Z",
13    "dataInicio": "2023-10-07T04:18:22.000Z"
14  }
15 ]
```

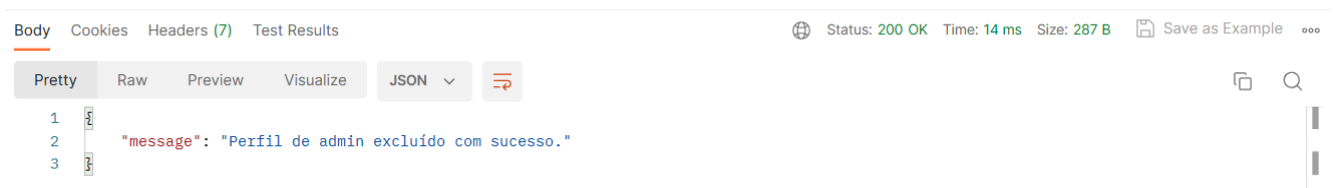
POSTMAN

DELETE

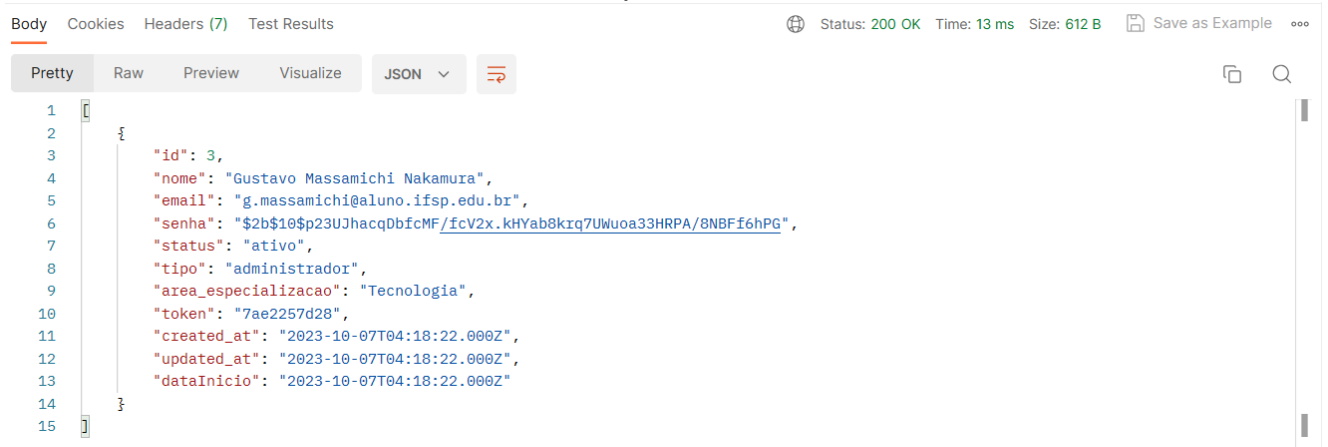
http://localhost:3000/path/admin/77765c4909

Send

Resposta:



GET admin após delete:

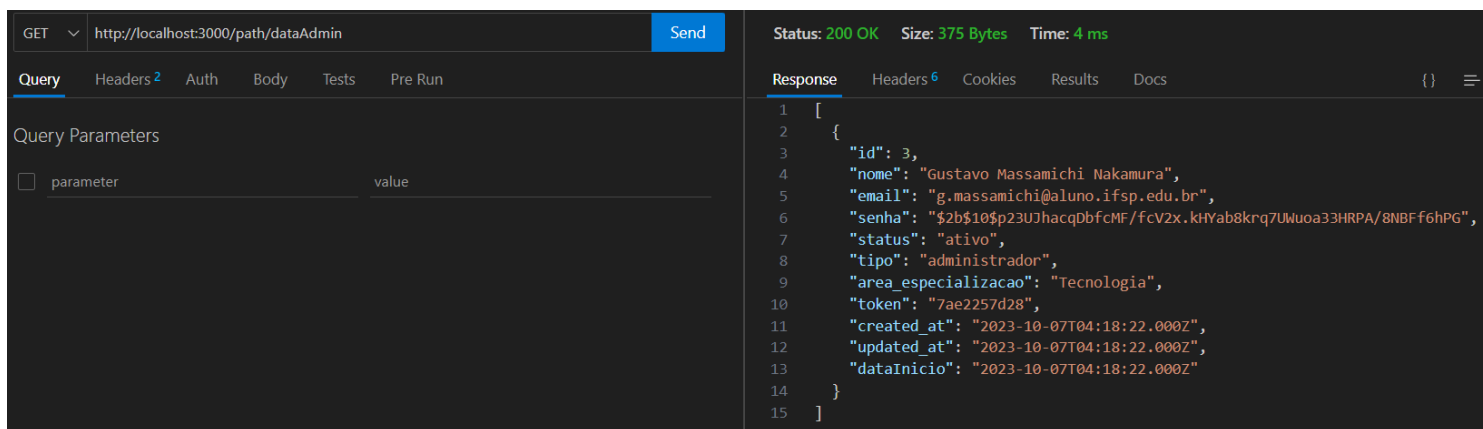


Sistema de criptografia para senhas:

No meu repositório do github, no caminho routes/path.js você poderá ver a seguinte comprovação de criptografia

```
const bcrypt = require('bcrypt');
```

Observe que a senha está criptografada:



Mesmo assim, posso colocar a senha normalmente como escolhi inicialmente:

POST

http://localhost:3000/path/login

Send

Query

Headers

Auth

Body1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

1

2

3

4

5

{

"email": "g.massamichi@aluno.ifsp.edu.br",

"senha": "gu1234"

}

Status: 200 OK

Size: 63 Bytes

Time: 111 ms

Response

Headers6

Cookies

Results

Docs

1

2

3

4

{

"success": true,

"redirectUrl": "/profile.html?token=7ae2257d28"

}

BREVE RESUMO DAS RESQUISIÇÕES

POSTMAN

Thunder Client

Sebo Online SA

users

PUT http://localhost:3000/path/u...

GET http://localhost:3000/path/d...

DEL http://localhost:3000/path/u...

admin

PUT http://localhost:3000/path/a...

DEL http://localhost:3000/path/a...

GET http://localhost:3000/path/d...

POST Login usuários e adms

POST http://localhost:3000/path/sig...

API Sebo_Online

Users

GET Get Users

6 hours ago

DEL Deletar Users

33 mins ago

PUT Edit Users

49 mins ago

Admin

GET Get Admins

6 hours ago

DEL Deletar Adm...

8 mins ago

PUT Edit Admin

41 mins ago

POST Cadastro

9 mins ago

POST Login

60 mins ago