# RPG0035 - SOFTWARE SEM SEGURANÇA NÃO SERVE!

## Missão Prática | Nível 4 | Mundo 5


- **Aluno:** Fábio Henrique Morales Prado
- **Matrícula:** 202211280754
- **Campus:** Sudoeste


- **Curso:** Desenvolvimento Full-Stack
- **Disciplina:** RPG0035 - SOFTWARE SEM SEGURANÇA NÃO SERVE!
- **Turma:** 2024.3
- **Semestre Letivo:** 5º Semestre

**Relatório**

- api.js

```
const express = require('express');
const db = require('./db');
const jwt = require('jsonwebtoken');
const bodyParser = require('body-parser');
```

```
const app = express();


 app.use(bodyParser.json());


 const port = process.env.PORT || 3000;


 // Chave secreta para assinar os tokens JWT
 const secretKey = 'P@%+~~=0[2YW59l@M+5ctb-
 ;|Y4{z;1om1CuyN#n0t)pm0/yEC0"dn`wvg92D7A';


 app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
 });



 // Middleware para verificar o token JWT
 function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader CC authHeader.split(' ')[1];


  if (token == null) return res.status(401).json({ message: 'Token not provided' });


  jwt.verify(token, secretKey, (err, user) => {
   if (err) return res.status(403).json({ message: 'Invalid token' });
   req.user = user;
   next();
  });
 }


 // Middleware para verificar o perfil do usuário
 function authorizeAdmin(req, res, next) {
```

```
  getPerfil(req.user.usuario_id).then(perfil => {
    if (perfil !== 'admin') {
      return res.status(403).json({ message: 'Forbidden: Admins only' });
    }
    next();
  }).catch(err => {
    res.status(500).json({ message: 'Internal Server Error' });
  });
}

// Endpoint para login do usuário
app.post('/api/auth/login', (req, res) => {
const credentials = req.body;

  doLogin(credentials).then(userData => {
    if (userData) {
      // Cria o token que será usado como session id
      const token = jwt.sign({ usuario_id: userData.id }, secretKey, { expiresIn: '1h' });
      res.json({ sessionid: token });
    } else {
      res.status(401).json({ message: 'Invalid credentials' });
    }
  }).catch(err => {
    res.status(500).json({ message: 'Internal Server Error' });
  });
});

// Endpoint para recuperação dos dados do usuário logado
app.get('/api/me', authenticateToken, (req, res) => {
getUserById(req.user.usuario_id).then(userData => {
    res.status(200).json({ data: userData });
```

```
  }).catch(err => {
    res.status(500).json({ message: 'Internal Server Error' });
  });
});

// Endpoint para recuperação dos dados de todos os usuários cadastrados
app.get('/api/users', authenticateToken, authorizeAdmin, (req, res) => {
getAllUsers().then(users => {
    res.status(200).json({ data: users });
  }).catch(err => {
    res.status(500).json({ message: 'Internal Server Error' });
  });
});

// Endpoint para recuperação dos contratos existentes
app.get('/api/contracts/:empresa/:inicio', authenticateToken, authorizeAdmin, async (req,
res) => {
  const { empresa, inicio } = req.params;

  try {
    const result = await getContracts(empresa, inicio);
    if (result.length > 0) {
      res.status(200).json({ data: result });
    } else {
      res.status(404).json({ data: 'Dados Não encontrados' });
    }
  } catch (error) {
    res.status(500).json({ message: 'Internal Server Error' });
  }
});
```

```
// Função genérica para executar consultas SQL
function executeQuery(query, params = []) {
return new Promise((resolve, reject) => {
  db.get(query, params, (err, row) => {
    if (err) {
      reject(err);
    } else {
      resolve(row);
    }
  });
 });
}

// Função genérica para executar consultas SQL que retornam múltiplas linhas
function executeQueryAll(query, params = []) {
 return new Promise((resolve, reject) => {
  db.all(query, params, (err, rows) => {
    if (err) {
      reject(err);
    } else {
      resolve(rows);
    }
  });
 });
}

// Recupera os dados do usuário através do id
function getUserById(userId) {
 // Consulta parametrizada previne SQL Injection
 return executeQuery('SELECT id, username, email, perfil FROM users WHERE id = ?',
[userId]);
```

```
}

// Recupera todos os usuários
function getAllUsers() {
  // Consulta parametrizada previne SQL Injection
  return executeQueryAll('SELECT * FROM users');
}

// Realiza o login do usuário
function doLogin(credentials) {
  // Consulta parametrizada previne SQL Injection
  return executeQuery('SELECT * FROM users WHERE username = ? AND password = ?',
[credentials.username, credentials.password]);
}

// Recupera o perfil do usuário através do id
function getPerfil(userId) {
  // Consulta parametrizada previne SQL Injection
  return executeQuery('SELECT perfil FROM users WHERE id = ?', [userId]).then(row =>
row.perfil);
}

// Recupera, no banco de dados, os dados dos contratos
function getContracts(empresa, inicio) {
  // Consulta parametrizada previne SQL Injection
  return executeQueryAll('SELECT * FROM contracts WHERE empresa = ? AND data_inicio
= ?', [empresa, inicio]);
}
```

- db.js

```
const sqlite3 = require('sqlite3').verbose();

const db = new sqlite3.Database(':memory:');
```

```
db.serialize(() => {
  db.run(`CREATE TABLE contracts (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    empresa TEXT,
    data_inicio TEXT
  )`);

  db.run(`CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT,
    password TEXT,
    email TEXT,
    perfil TEXT
  )`);

  // Inserção de dados de exemplo
  db.run(`INSERT INTO contracts (empresa, data_inicio) VALUES ('empresa1', '2023-01-01')`);
  db.run(`INSERT INTO contracts (empresa, data_inicio) VALUES ('empresa2', '2023-02-01')`);

  db.run(`INSERT INTO users (username, password, email, perfil) VALUES ('user', '123456', 'user@dominio.com', 'user')`);
  db.run(`INSERT INTO users (username, password, email, perfil) VALUES ('admin', '123456789', 'admin@dominio.com', 'admin')`);
  db.run(`INSERT INTO users (username, password, email, perfil) VALUES ('colab', '123', 'colab@dominio.com', 'user')`);
});

module.exports = db;
```

- MP5M5.postman_collection.json

```
{
```

```
"info": {
		"_postman_id": "fbb27f52-0b20-4161-91e0-cdeeac8c4c00",
		"name": "MP5M5",
		"description": "# 🚀 Get started here\n\nThis template guides you through CRUD operations (GET, POST, PUT, DELETE), variables, and tests.\n\n## ⚡️ **How to use this template**\n\n#### **Step 1: Send requests**\n\nRESTful APIs allow you to perform CRUD operations using the POST, GET, PUT, and DELETE HTTP methods.\n\nThis collection contains each of these [request](https://learning.postman.com/docs/sending-requests/requests/) types. Open each request and click \"Send\" to see what happens.\n\n#### **Step 2: View responses**\n\nObserve the response tab for status code (200 OK), response time, and size.\n\n#### **Step 3: Send new Body data**\n\nUpdate or add new data in \"Body\" in the POST request. Typically, Body data is also used in PUT request.\n\n```\n{\n  \"name\": \"Add your name in the body\"\n}\n```\n\n#### **Step 4: Update the variable**\n\nVariables enable you to store and reuse values in Postman. We have created a [variable](https://learning.postman.com/docs/sending-requests/variables/) called `base_url` with the sample request [https://postman-api-learner.glitch.me](https://postman-api-learner.glitch.me). Replace it with your API endpoint to customize this collection.\n\n#### **Step 5: Add tests in the \"Scripts\" tab**\n\nAdding tests to your requests can help you confirm that your API is working as expected. You can write test scripts in JavaScript and view the output in the \"Test Results\" tab.\n\n<img src=\"https://content.pstmn.io/fa30ea0a-373d-4545-a668-e7b283cca343/aW1hZ2UucG5n\" alt=\"\" height=\"1530\" width=\"2162\">\n\n## 💡 Pro tips\n\n- Use folders to group related requests and organize the collection.\n- Add more [scripts](https://learning.postman.com/docs/writing-scripts/intro-to-scripts/) to verify if the API works as expected and execute workflows.\n\n## 🤝 Related templates\n\n[API testing basics](https://go.postman.co/redirect/workspace?type=personalCcollectionTemplateId=e9a37a28-055b-49cd-8c7e-97494a21eb54CsourceTemplateId=ddb19591-3097-41cf-82af-c84273e56719)\n[API documentation](https://go.postman.co/redirect/workspace?type=personalCcollectionTemplateId=e9c28f47-1253-44af-a2f3-20dce4da1f18CsourceTemplateId=ddb19591-3097-41cf-82af-c84273e56719)\n[Authorization methods](https://go.postman.co/redirect/workspace?type=personalCcollectionTemplateId=31a9a6ed-4cdf-4ced-984c-d12c9aec1c27CsourceTemplateId=ddb19591-3097-41cf-82af-c84273e56719)",
		"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
		"_exporter_id": "24144890"
	},
	"item": [
		{
```

```
                "name": "login adm",
                "event": [
                        {
                                "listen": "test",
                                "script": {
                                        "exec": [
                                                "pm.test(\"Successful POST request\", function () {",
                                                "    pm.expect(pm.response.code).to.be.oneOf([200,  201]);",
                                                "});",
                                                ""
                                        ],
                                        "type": "text/javascript",
                                        "packages": {}
                                }
                        }
                ],
                "request": {
                        "method": "POST",
                        "header": [],
                        "body": {
                                "mode": "raw",
                                "raw": "{\n\"username\": \"admin\", \"password\": \"123456789\"\n}",
                                "options": {
                                        "raw": {
                                                "language": "json"
                                        }
                                }
                        },
                        "url": {
```

```
                "raw": "http://localhost:3000/api/auth/login",
                "protocol": "http",
                "host": [
                    "localhost"
                ],
                "port": "3000",
                "path": [
                    "api",
                    "auth",
                    "login"
                ]
            },
            "description": "This is a POST request, submitting data to an API via the request body. This request submits JSON data, and the data is reflected in the response.\n\nA successful POST request typically returns a `200 OK` or `201 Created` response code."
        },
        "response": []
    },
    {
        "name": "adm logado",
        "event": [
            {
                "listen": "test",
                "script": {
                    "exec": [
                        "pm.test(\"Status code is 200\", function () {",
                        "    pm.response.to.have.status(200);",
                        "});"
                    ],
                    "type": "text/javascript",
```

```
                "packages": {}
            }
        }
    ],
    "request": {
        "auth": {
            "type": "bearer",
            "bearer": [
                {
                    "key": "token",
                    "value": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c3VhcmlvX2lkIjoyLCJpYXQiOjE3MjkzNTQ4OTYsImV4cCI6MTcyOTM1ODQ5Nn0.KJ7_-9IIrAIhhIjkdi4fuFXWLRKjmUgCs7Ur-QhSMl8",
                    "type": "string"
                }
            ]
        },
        "method": "GET",
        "header": [],
        "url": {
            "raw": "http://localhost:3000/api/me",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "3000",
            "path": [
                "api",
                "me"
            ]
        },
```

```
                "description": "This is a GET request and it is used to \"get\" data from an endpoint. There is no request body for a GET request, but you can use query parameters to help specify the resource you want data on (e.g., in this request, we have `id=1`).\n\nA successful GET response will have a `200 OK` status, and should include some kind of response body - for example, HTML web content or JSON data."
            },
            "response": []
        },
        {
            "name": "usuarios",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Successful PUT request\", function () {",
                            "    pm.expect(pm.response.code).to.be.oneOf([200, 201, 204]);",
                            "});",
                            ""
                        ],
                        "type": "text/javascript",
                        "packages": {}
                    }
                }
            ],
            "request": {
                "auth": {
                    "type": "bearer",
                    "bearer": [
                        {
                            "key": "token",
```

```
                                        "value":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c3VhcmlvX2lkIjoyLCJpYXQiOjE3MjkzNTQ4O
TYsImV4cCI6MTcyOTM1ODQ5Nn0.KJ7_-9IIrAIhhIjkdi4fuFXWLRKjmUgCs7Ur-QhSMl8",
                                        "type": "string"
                                    }
                                ]
                            },
                            "method": "GET",
                            "header": [],
                            "body": {
                                "mode": "raw",
                                "raw": "{\n\t\"name\": \"Add your name in the
body\"\n}",
                                "options": {
                                    "raw": {
                                        "language": "json"
                                    }
                                }
                            },
                            "url": {
                                "raw": "http://localhost:3000/api/users",
                                "protocol": "http",
                                "host": [
                                    "localhost"
                                ],
                                "port": "3000",
                                "path": [
                                    "api",
                                    "users"
                                ]
                            },
```

```
                "description": "This is a PUT request and it is used to overwrite an existing piece of data. For instance, after you create an entity with a POST request, you may want to modify that later. You can do that using a PUT request. You typically identify the entity being updated by including an identifier in the URL (eg. `id=1`).\n\nA successful PUT request typically returns a `200 OK`, `201 Created`, or `204 No Content` response code."
            },
            "response": []
        },
        {
            "name": "obter contrato",
            "event": [
                {
                    "listen": "test",
                    "script": {
                        "exec": [
                            "pm.test(\"Successful DELETE request\", function () {",
                            "    pm.expect(pm.response.code).to.be.oneOf([200, 202, 204]);",
                            "});",
                            ""
                        ],
                        "type": "text/javascript",
                        "packages": {}
                    }
                }
            ],
            "request": {
                "auth": {
                    "type": "bearer",
                    "bearer": [
                        {
                            "key": "token",
```

```
                                        "value":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c3VhcmlvX2lkIjoyLCJpYXQiOjE3MjkzNTQ4O
TYsImV4cCI6MTcyOTM1ODQ5Nn0.KJ7_-9IIrAIhhIjkdi4fuFXWLRKjmUgCs7Ur-QhSMl8",
                                        "type": "string"
                                }
                        ]
                },
                "method": "GET",
                "header": [],
                "body": {
                        "mode": "raw",
                        "raw": "",
                        "options": {
                                "raw": {
                                        "language": "json"
                                }
                        }
                },
                "url": {
                        "raw":
"http://localhost:3000/api/contracts/empresa2/2023-02-01",
                        "protocol": "http",
                        "host": [
                                "localhost"
                        ],
                        "port": "3000",
                        "path": [
                                "api",
                                "contracts",
                                "empresa2",
                                "2023-02-01"
                        ]
```

```
                },
                "description": "This is a DELETE request, and it is used to delete data that was previously created via a POST request. You typically identify the entity being updated by including an identifier in the URL (eg. `id=1`).\n\nA successful DELETE request typically returns a `200 OK`, `202 Accepted`, or `204 No Content` response code."
            },
            "response": []
        },
        {
            "name": "login usuario",
            "request": {
                "auth": {
                    "type": "bearer",
                    "bearer": [
                        {
                            "key": "token",
                            "value": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c3VhcmlvX2lkIjoxLCJpYXQiOjE3MjkzNTU2OTMsImV4cCI6MTcyOTM1OTI5M30.SWM8i_kYpkrFtAeu80rMtnxN8vYkXDZAnMJjey-dXZc",
                            "type": "string"
                        }
                    ]
                },
                "method": "POST",
                "header": [],
                "body": {
                    "mode": "raw",
                    "raw": "{\r\n\"username\": \"user\", \"password\": \"123456\"\r\n}",
                    "options": {
                        "raw": {
                            "language": "json"
                        }
```

```
                    }
                },
                "url": {
                    "raw": "http://localhost:3000/api/auth/login",
                    "protocol": "http",
                    "host": [
                        "localhost"
                    ],
                    "port": "3000",
                    "path": [
                        "api",
                        "auth",
                        "login"
                    ]
                }
            },
            "response": []
        },
        {
            "name": "usuario logado",
            "request": {
                "auth": {
                    "type": "bearer",
                    "bearer": [
                        {
                            "key": "token",
                            "value":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c3VhcmlvX2lkIjoxLCJpYXQiOjE3MjkzNTU2OTMsImV4cCI6MTcyOTM1OTI5M30.SWM8i_kYpkrFtAeu80rMtnxN8vYkXDZAnMJjey-dXZc",
                            "type": "string"
                        }
                    ]
```

```
                },
                "method": "GET",
                "header": [],
                "url": {
                        "raw": "http://localhost:3000/api/me",
                        "protocol": "http",
                        "host": [
                                "localhost"
                        ],
                        "port": "3000",
                        "path": [
                                "api",
                                "me"
                        ]
                }
        },
        "response": []
},
{
        "name": "consulta usuarios",
        "request": {
                "auth": {
                        "type": "bearer",
                        "bearer": [
                                {
                                        "key": "token",
                                        "value":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c3VhcmlvX2lkIjoxLCJpYXQiOjE3MjkzNTU2O
TMsImV4cCI6MTcyOTM1OTI5M30.SWM8i_kYpkrFtAeu80rMtnxN8vYkXDZAnMJjey-dXZc",
                                        "type": "string"
                                }
                        ]
```

```
                },
                "method": "GET",
                "header": [],
                "url": {
                        "raw": "http://localhost:3000/api/users",
                        "protocol": "http",
                        "host": [
                                "localhost"
                        ],
                        "port": "3000",
                        "path": [
                                "api",
                                "users"
                        ]
                }
        },
        "response": []
},
{
        "name": "consulta contrato",
        "request": {
                "auth": {
                        "type": "bearer",
                        "bearer": [
                                {
                                        "key": "token",
                                        "value":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c3VhcmlvX2lkIjoxLCJpYXQiOjE3MjkzNTU2O
TMsImV4cCI6MTcyOTM1OTI5M30.SWM8i_kYpkrFtAeu80rMtnxN8vYkXDZAnMJjey-dXZc",
                                        "type": "string"
                                }
                        ]
```

```
                },
                "method": "GET",
                "header": [],
                "url": {
                        "raw":
"http://localhost:3000/api/contracts/empresa2/2023-02-01",
                        "protocol": "http",
                        "host": [
                                "localhost"
                        ],
                        "port": "3000",
                        "path": [
                                "api",
                                "contracts",
                                "empresa2",
                                "2023-02-01"
                        ]
                }
        },
        "response": []
    }
],
"event": [
    {
        "listen": "prerequest",
        "script": {
                "type": "text/javascript",
                "exec": [
                        ""
                ]
        }
```

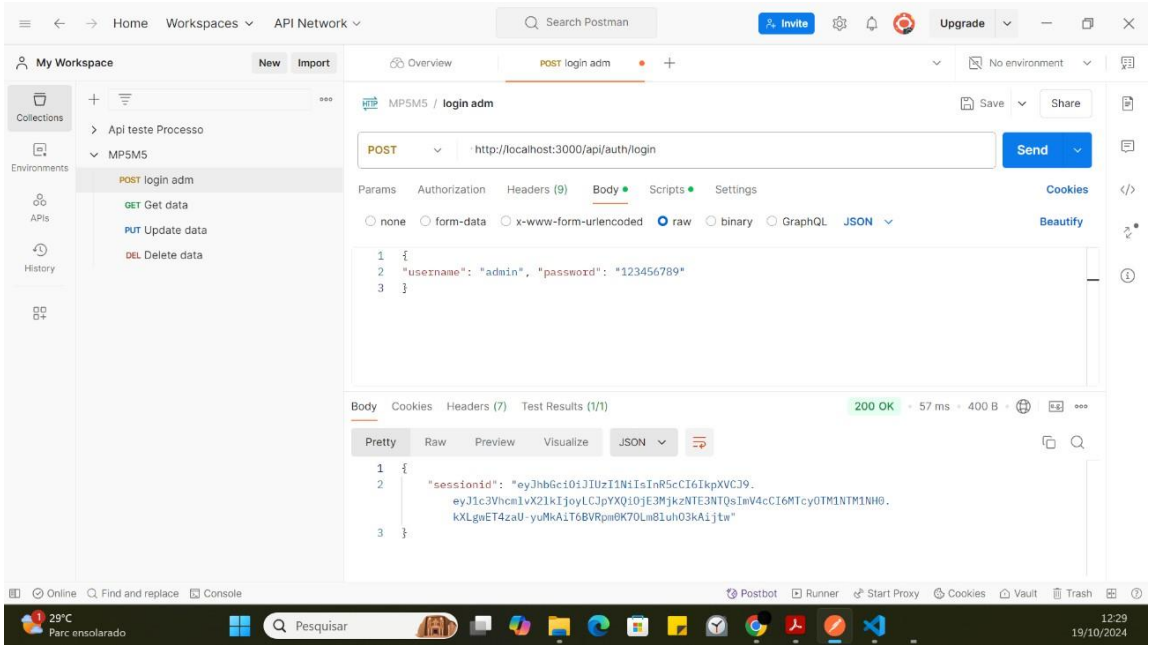
```
                },
                {
                        "listen": "test",
                        "script": {
                                "type": "text/javascript",
                                "exec": [
                                        ""
                                ]
                        }
                }
        ],
        "variable": [
                {
                        "key": "id",
                        "value": "1"
                },
                {
                        "key": "base_url",
                        "value": "https://postman-rest-api-learner.glitch.me/"
                }
        ]
}
```


---

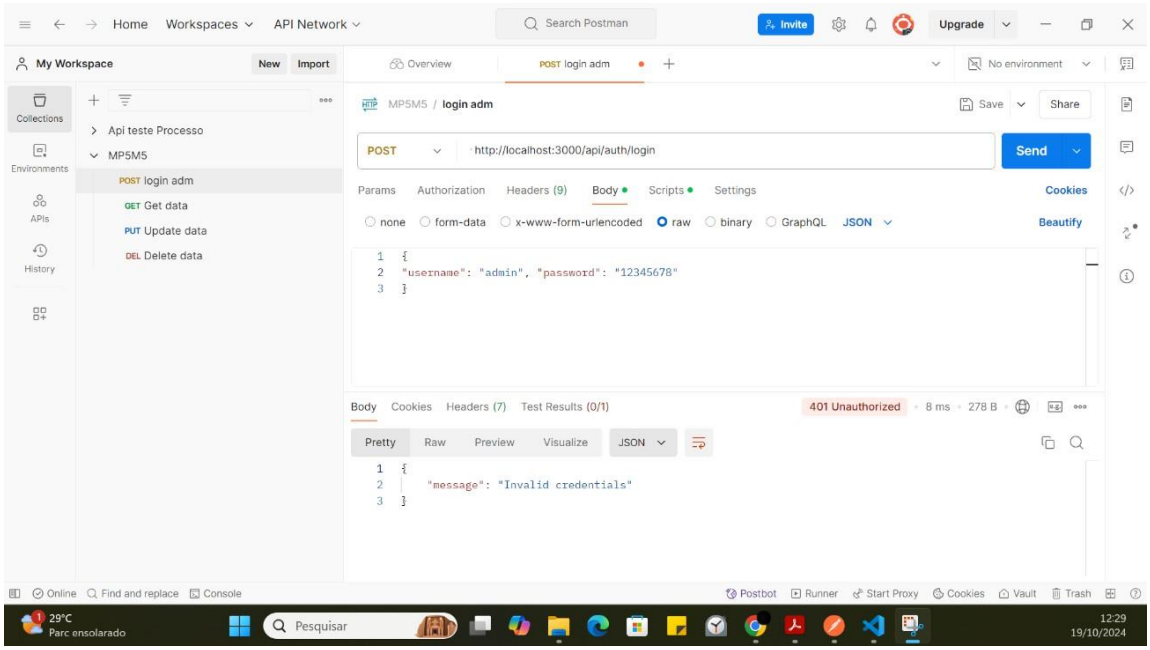
**Resultados:**

📌 Postman:

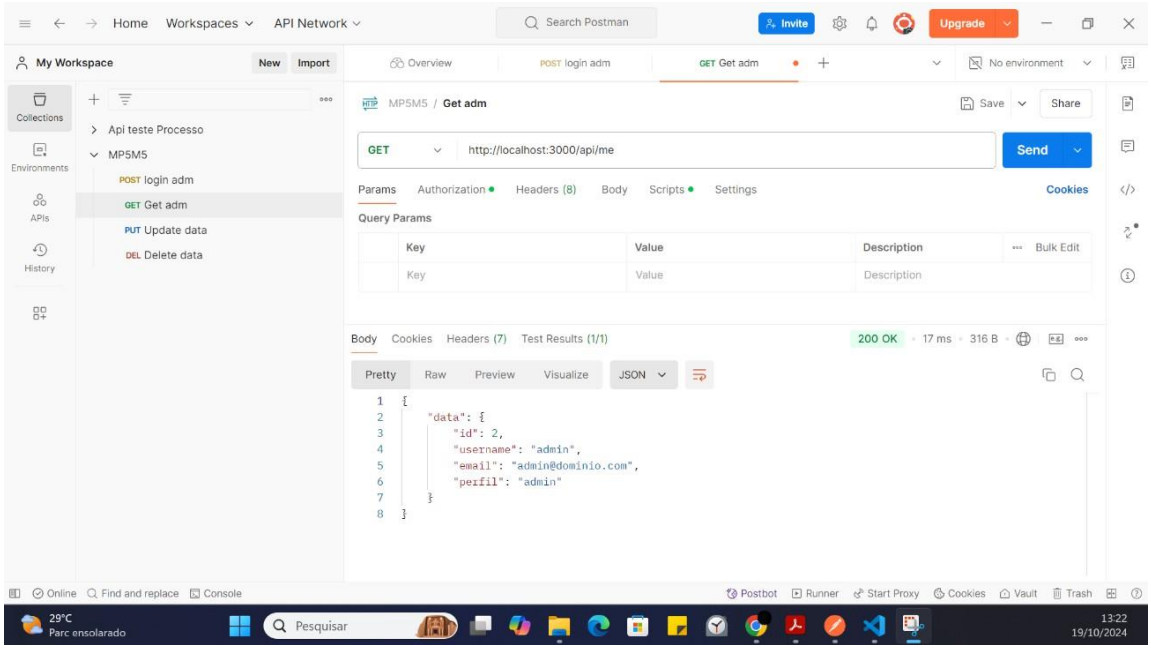
**Login como Adm**



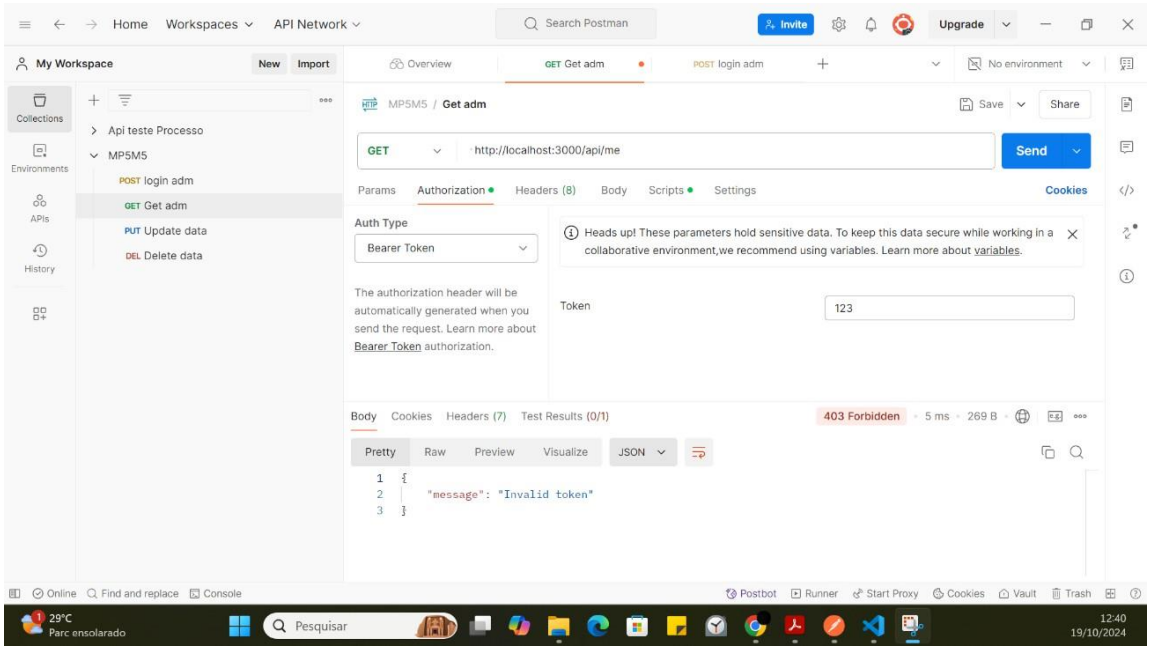

## Login como Adm - Senha Invalida



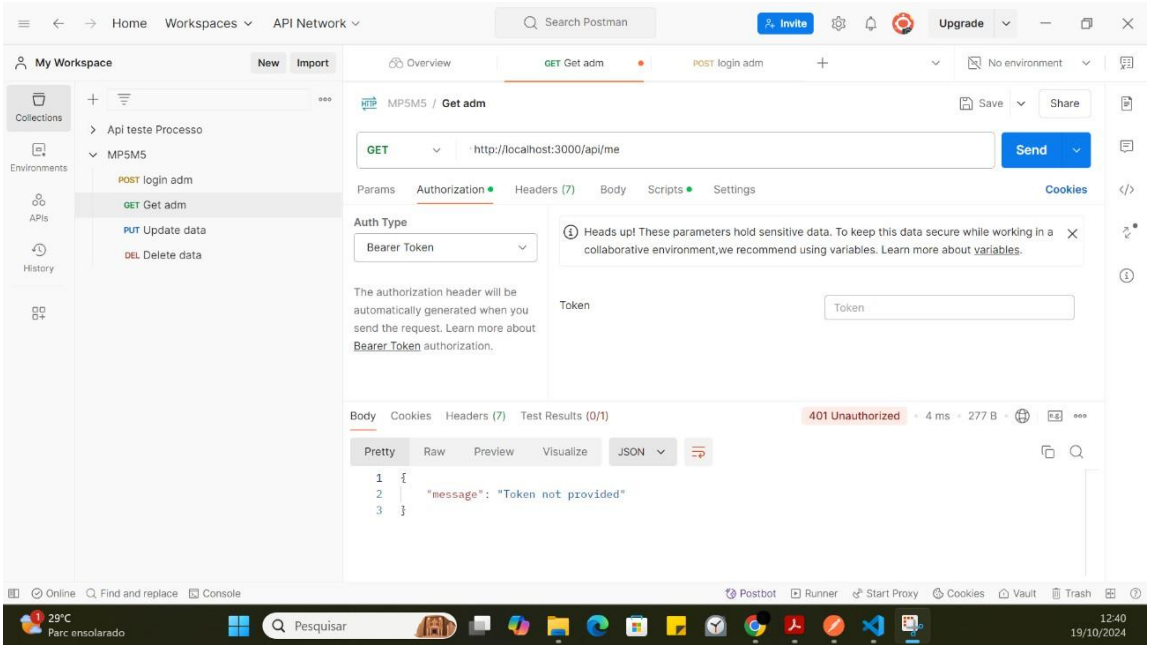

## Dados do Adm logado



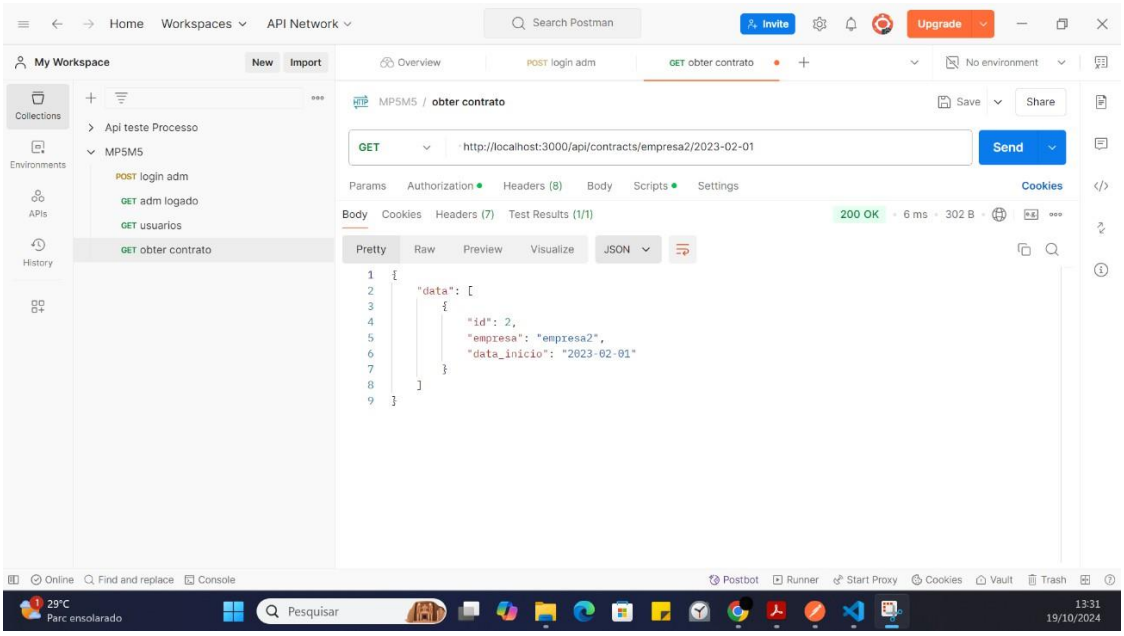

## Token Invalido

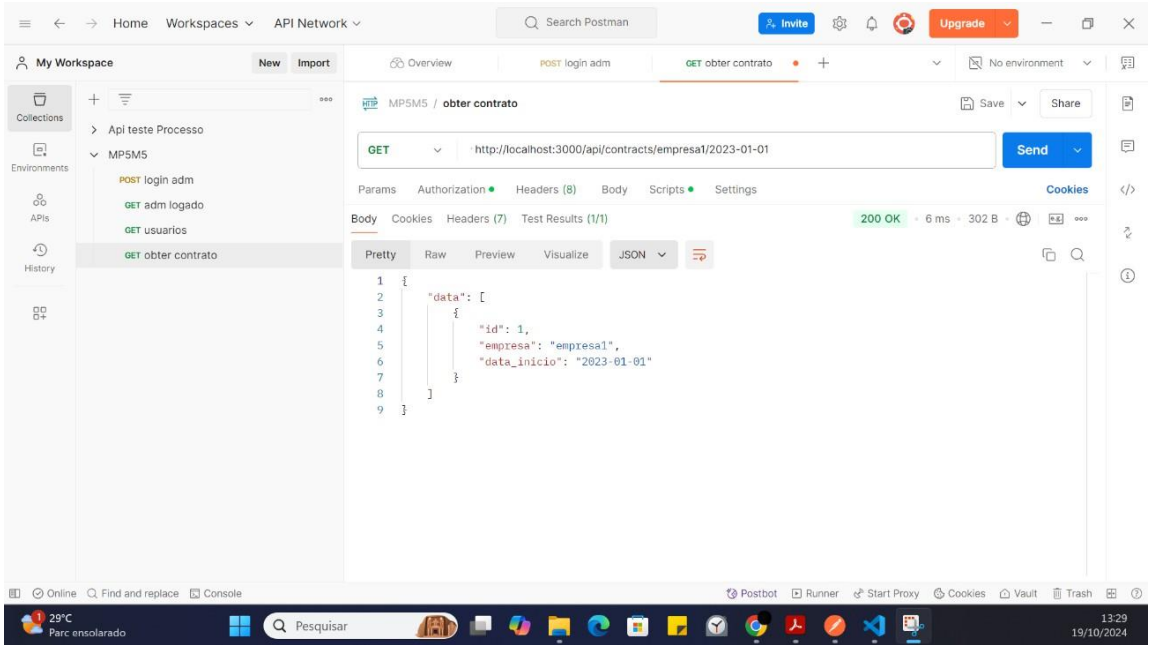



## Token não informado

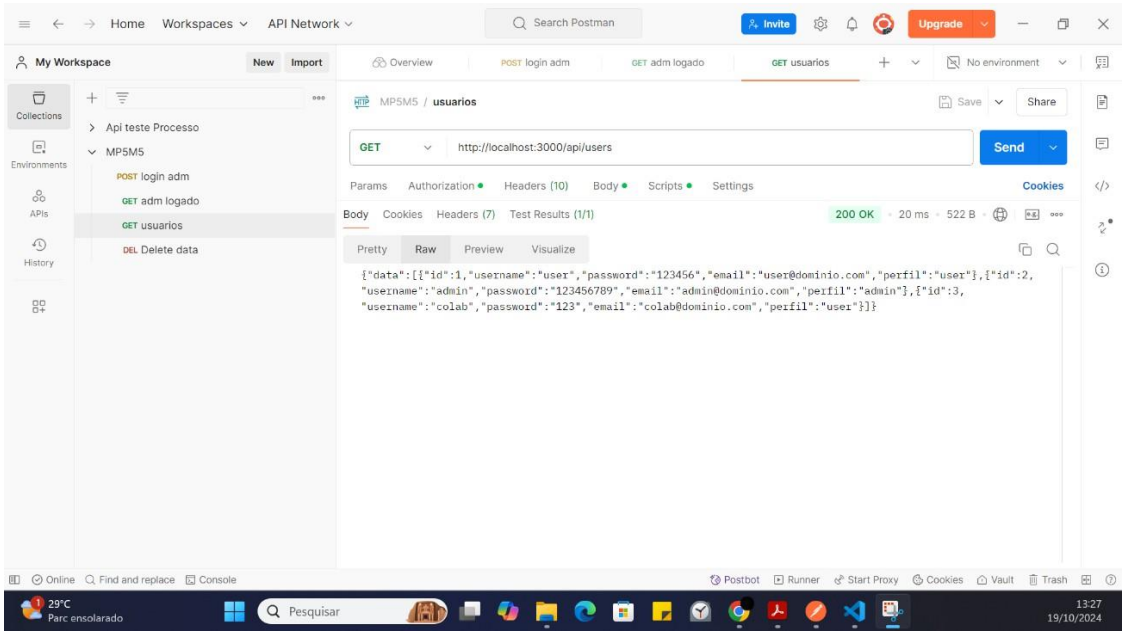


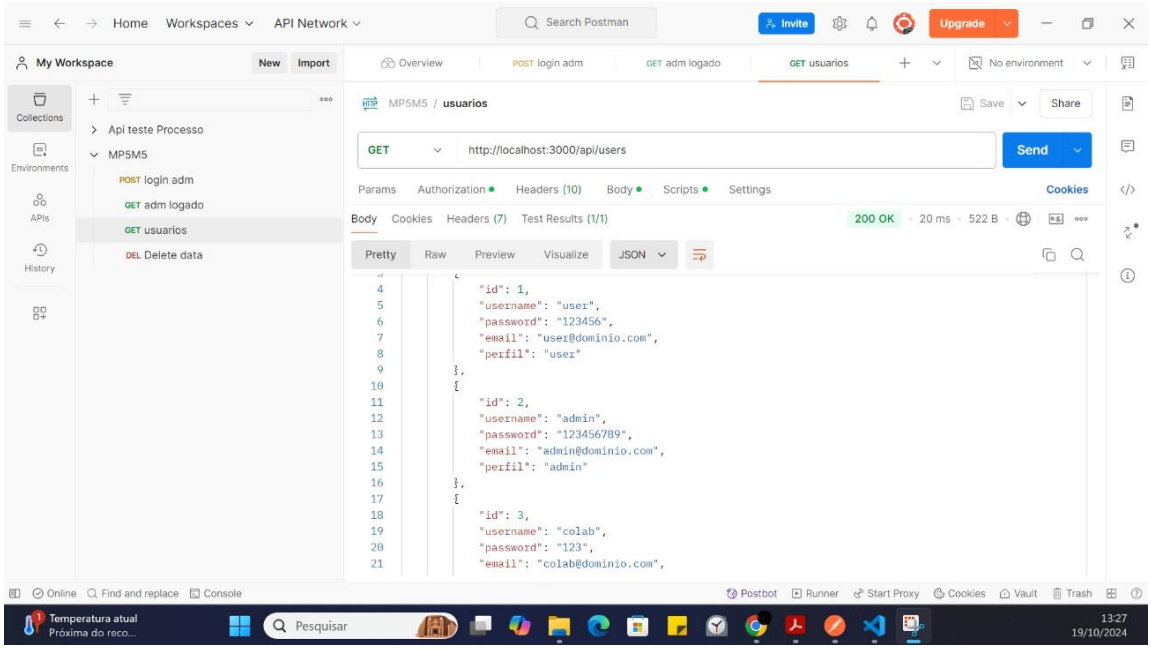

## Contrato empresas



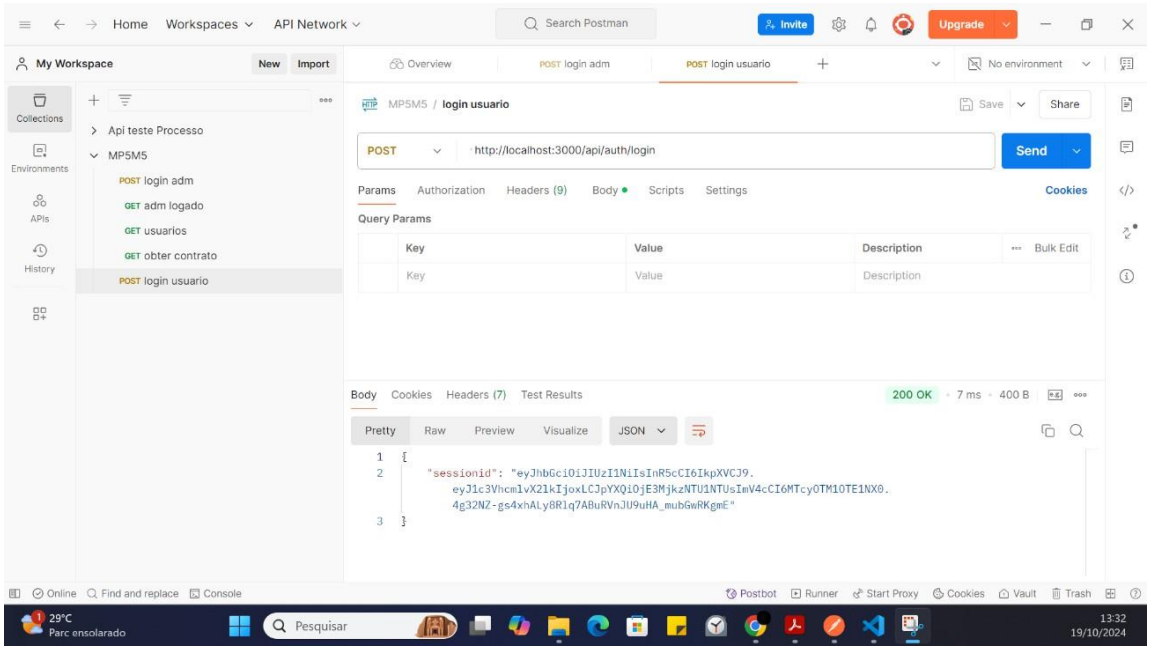



## Dados dos usuarios

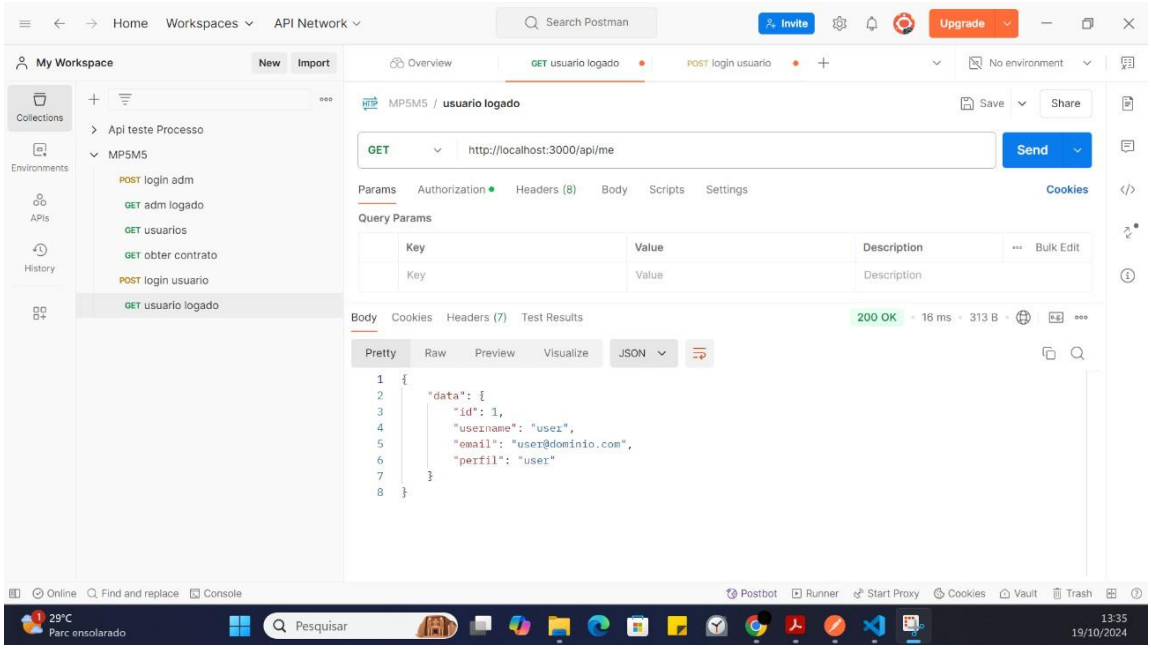




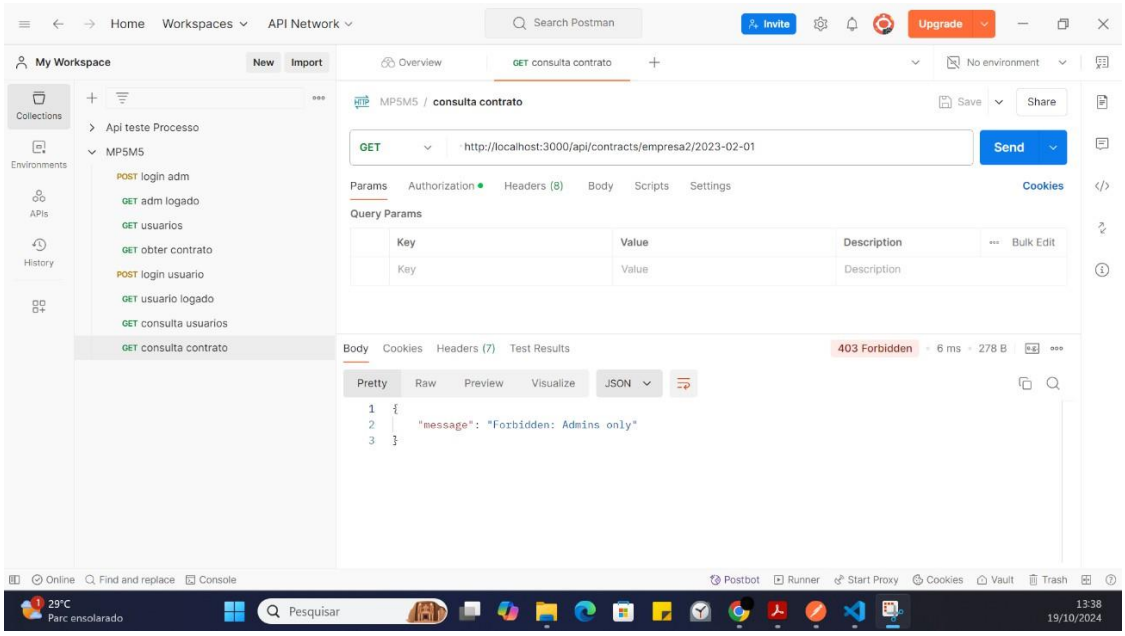

## Login como usuario

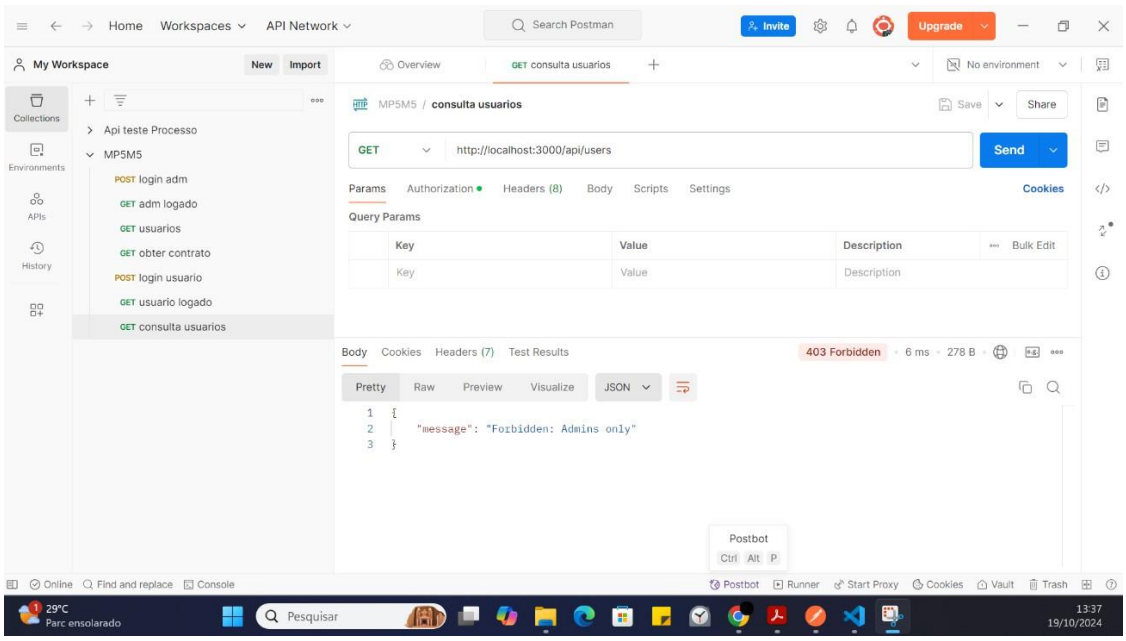



## Dados do usuario logado




## Consultas proibidas para usuarios




Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
GET consulta usuarios
No environment
Collections
Environments
APIs
History
Api teste Processo
MP5M5
POST login adm
GET adm logado
GET usuarios
GET obter contrato
POST login usuario
GET usuario logado
GET consulta usuarios
MP5M5 / consulta usuarios
Save
Share
GET
http://localhost:3000/api/users
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (7)
Test Results
403 Forbidden
6 ms
278 B
Pretty
Raw
Preview
Visualize
JSON
{
"message": "Forbidden: Admins only"
}
Postbot
Ctrl Alt P
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
29°C
Parc ensolarado
Pesquisar
13:37
19/10/2024