



Estácio

Universidade Estácio de Sá

- DESENVOLVIMENTO FULL STACK-
 - Disciplina: RPG0035 - SOFTWARE SEM SEGURANÇA NÃO SERVE!
 - Semestre Letivo: 2024.2
 - Repositorio Git: <https://github.com/Gregdev22/Missao-5-Mundo-5>
-

- [EMERSON GREGORIO ALVES](#) - MATRICULA: 2022.0908.4986
-

Missão Prática | Nível 5 | Mundo 5

SOFTWARE SEM SEGURANÇA NÃO SERVE!

Objetivos da Prática

- Descrever o controle básico de acesso a uma API Rest;
- Descrever o tratamento de dados sensíveis e log de erros com foco em segurança;
- Descrever a prevenção de ataques de acesso não autorizado com base em tokens desprotegidos/desatualizados;
- Descrever o tratamento de SQL Injection em códigos-fonte; Descrever o tratamento de CRLF Injection em códigos-fonte;
- Descrever a prevenção a ataques do tipo CSRF em sistemas web;

Códigos

- 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 && 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 getUserId(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  \"name\": \"Add your name in the body\"\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 Pro  
tips\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💡 Related  
templates\n[API testing  
basics](https://go.postman.co/redirect/workspace?type=personal&collectionTemplateId=e9a37a28-055b-49cd-8c7e-97494a21eb54&sourceTemplateId=ddb19591-3097-41cf-82af-c84273e56719)\n[API  
documentation](https://go.postman.co/redirect/workspace?type=personal&collectionTe  
mplateId=e9c28f47-1253-44af-a2f3-20dce4da1f18&sourceTemplateId=ddb19591-3097-  
41cf-82af-c84273e56719)\n[Authorization  
methods](https://go.postman.co/redirect/workspace?type=personal&collectionTemplate  
Id=31a9a6ed-4cdf-4ced-984c-d12c9aec1c27&sourceTemplateId=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":



"eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJ1c3VhcmlvX2lkljoyLCJpYXQiOjE3MjkzNTQ4OTYsImV4cCl6MTcyOTM1ODQ5Nn0.KJ7_-9IlrlAlhhIjkdi4fuFXWLRKjmUgCs7Ur-QhSMI8",

                    "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": "token",  
                        "type": "string"  
                    }  
                ]  
            }  
        }  
    }  
},  
{  
    "name": "usuarios",  
    "event": [  
        {  
            "listen": "test",  
            "script": {  
                "exec": [  
                    "pm.test(\"Successful POST request\",  
function () {",  
                    "pm.sendRequest({",  
                    "url": "http://www.example.com/api/users",  
                    "method": "POST",  
                    "headers": {  
                        "Content-Type": "application/json",  
                        "Accept": "application/json"  
                    },  
                    "body": {  
                        "id": 1,  
                        "name": "John Doe",  
                        "email": "john.doe@example.com",  
                        "password": "password123",  
                        "role": "user"  
                    }  
                }  
            }  
        }  
    ]  
}
```

```
        "value":  
        "eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJ1c3VhcmlvX2lkljoyLCJpYXQiOjE3MjkzNTQ4OTYsImV4cCl6MTcyOTM1ODQ5Nn0.KJ7_-9IlrAlhhIjkdi4fuFXWLRKjmUgCs7Ur-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": "token"
                    }
                ]
            }
        }
    ]
}
```

```
        "value":  
        "eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJ1c3VhcmlvX2lkljoyLCJpYXQiOjE3MjkzNTQ4OTySImV4cCl6MTcyOTM1ODQ5Nn0.KJ7_-9IlrAlhhIkdi4fuFXWLRKjmUgCs7Ur-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":  
"eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJ1c3VhcmlvX2lkljoxLCJpYXQiOjE3MjkzNTU2OTMsImV4cCl6MTcyOTM1OTI5M30.SWM8i_kYpkrFtAeu80rMtnxN8vYkXDZAnMJjey-dXZc",  
                    "type": "string"  
                }  
            ]  
        },  
        "method": "POST",  
        "header": [],  
        "body": {  
            "mode": "raw",  
            "raw": "{\"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": "eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJ1c3VhcmlvX2lkjoxLCJpYXQiOjE3MjkzNTU2OTMsImV4cCl6MTcyOTM1OTI5M30.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": "eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9.eyJ1c3VhcmlvX2lkjoxLCJpYXQiOjE3MjkzNTU2OTMsImV4cCl6MTcyOTM1OTI5M30.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.eyJ1c3VhcmlvX2lkjoxLCJpYXQiOjE3MjkzNTU2OTMsImV4cCI6MTcyOTM1OTI5M30.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

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:3000/api/auth/login`. The response status is `200 OK` with a duration of `57 ms` and a size of `400 B`. The response body contains a JSON object with a session ID:

```
1 {  
2   "sessionid": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJpc3Vhcm1vX21kIjoyLCJpYXQiOjE3MjkzNTE3NTQ1MmV4cC16Mcy0TM1NTM1NH0.  
KXLgwET4zaU-yuMkA1T6BVRpmm0K70Lm01uh03kAijtw"  
3 }
```

Login como Adm - Senha Invalida

The screenshot shows the Postman interface with an unauthorized response. The URL is `http://localhost:3000/api/auth/login`. The response status is `401 Unauthorized` with a duration of `8 ms` and a size of `278 B`. The response body contains a JSON object with a message:

```
1 {  
2   "message": "Invalid credentials"  
3 }
```

Dados do Adm logado

The screenshot shows the Postman interface. On the left, the 'My Workspace' sidebar lists collections, environments, APIs, and history. In the center, a request for 'GET adm' is displayed under the 'MPSM5' collection. The request URL is 'http://localhost:3000/api/me'. The 'Authorization' tab is selected, showing a Bearer token. The response status is '200 OK' with a response time of 17 ms and a body size of 316 B. The response body is a JSON object:

```
1 {  
2   "data": {  
3     "id": 2,  
4     "username": "admin",  
5     "email": "admin@dominio.com",  
6     "perfil": "admin"  
7   }  
8 }
```

Token Invalido

The screenshot shows the Postman interface. The setup is identical to the previous one, but the 'Authorization' tab now shows 'Bearer Token' selected. A warning message appears: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables.' The response status is '403 Forbidden' with a response time of 5 ms and a body size of 269 B. The response body is a JSON object:

```
1 {  
2   "message": "Invalid token"  
3 }
```

Token não informado

The screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with collections like 'Api teste Processo' and 'MP5M5'. Under 'MP5M5', there are several endpoints: 'POST login adm', 'GET Get adm' (which is selected), 'PUT Update data', and 'DEL Delete data'. The main workspace shows a 'GET' request to 'http://localhost:3000/api/me'. The 'Authorization' tab is selected, showing 'Bearer Token' as the type. A note says: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)'. Below this, it says 'The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.' The response status is '401 Unauthorized' with a duration of '4 ms' and a size of '277 B'. The response body is a JSON object: { "message": "Token not provided" }. The bottom status bar shows the date and time as '19/10/2024'.

Contrato empresas

This screenshot shows the same Postman interface as the previous one, but with a different endpoint. The 'GET obtener contrato' request is selected. The URL is 'http://localhost:3000/api/contracts/empresa2/2023-02-01'. The response status is '200 OK' with a duration of '6 ms' and a size of '302 B'. The response body is a JSON array: [{ "data": [{ "id": 2, "empresa": "empresa2", "data_inicio": "2023-02-01" }] }]. The bottom status bar shows the date and time as '19/10/2024'.

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, APIs, History.
- Request URL:** http://localhost:3000/api/contracts/empresa1/2023-01-01
- Method:** GET
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```
1 {
2   "data": [
3     {
4       "id": 1,
5       "empresa": "empresa1",
6       "data_inicio": "2023-01-01"
7     }
8   ]
9 }
```

Dados dos usuarios

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, APIs, History.
- Request URL:** http://localhost:3000/api/users
- Method:** GET
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```
[{"data": [{"id": 1, "username": "user", "password": "123456", "email": "user@dominio.com", "perfil": "user"}, {"id": 2, "username": "admin", "password": "123456789", "email": "admin@dominio.com", "perfil": "admin"}, {"id": 3, "username": "colab", "password": "123", "email": "colab@dominio.com", "perfil": "user"}]}
```

My Workspace

API Network

Overview

POST login adm

GET adm logado

GET usuarios

DEL Delete data

HTTP MP5MS / usuarios

GET http://localhost:3000/api/users

Params Authorization Headers (10) Body Scripts Settings Cookies

```

Pretty Raw Preview Visualize JSON
{
  "id": 1,
  "username": "user",
  "password": "123456",
  "email": "user@dominio.com",
  "perfil": "user"
},
{
  "id": 2,
  "username": "admin",
  "password": "123456789",
  "email": "admin@dominio.com",
  "perfil": "admin"
},
{
  "id": 3,
  "username": "colab",
  "password": "123",
  "email": "colab@dominio.com",
}

```

200 OK 20 ms 522 B

Postbot Runner Start Proxy Cookies Vault Trash

Temperatura atual 29°C Proxima do re... 13:27 19/10/2024

Login como usuario

My Workspace

API Network

Overview

POST login adm

GET adm logado

GET usuarios

GET obtener contrato

POST login usuario

HTTP MP5MS / login usuario

POST http://localhost:3000/api/auth/login

Params Authorization Headers (9) Body Scripts Settings Cookies

Key	Value	Description	Bulk Edit
Key	Value	Description	

Query Params

Body Cookies Headers (7) Test Results

```

Pretty Raw Preview Visualize JSON
{
  "sessionId": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
  eyJlZmFsdWUtb3V0LWxpbmUyIjoxLCpXYQ10JE3MjkzNTU1NTUsImV4cCI6MTcyOTM1OTE1NX0.
  4g32NZ-gs4xhAlY8R1q7ABuRVnJU9uHA_mubGwRKgxE"
}

```

200 OK 7 ms 400 B

Postbot Runner Start Proxy Cookies Vault Trash

29°C Parc ensolarado 13:32 19/10/2024

Dados do usuario logado

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, APIs, History.
- Request URL:** http://localhost:3000/api/me
- Method:** GET
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```
1 {
2   "data": {
3     "id": 1,
4     "username": "user",
5     "email": "user@dominio.com",
6     "perfil": "user"
7   }
8 }
```

Consultas proibidas para usuarios

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, APIs, History.
- Request URL:** http://localhost:3000/api/contracts/empresa2/2023-02-01
- Method:** GET
- Response Status:** 403 Forbidden
- Response Body (Pretty JSON):**

```
1 {
2   "message": "Forbidden: Admins only"
3 }
```

Screenshot of the Postman application interface showing a collection named "MP5MS" with a "consultar usuarios" endpoint.

Left Sidebar:

- My Workspace
- New Import
- Overview
- GET consultar usuarios
- No environment
- Invite Upgrade

Environments:

- Collections
- + New
- Import
- MP5MS

APIs:

- POST login adm
- GET adm logado
- GET usuarios
- GET obtener contrato
- POST login usuario
- GET usuario logado
- GET consultar usuarios

HTTP Request:

Method: GET
URL: http://localhost:3000/api/users

Params: Authorization (green checkmark)
Headers: (8)
Body: (empty)
Scripts: (empty)
Settings: (empty)

Query Params:

Key	Value	Description
Key	Value	Description

Body: Cookies Headers (7) Test Results

403 Forbidden 6 ms 278 B

Code Preview:

```
1 {  
2   "message": "Forbidden: Admins only"  
3 }
```

Bottom Bar:

- Postbot
- Ctrl Alt P
- Online
- Find and replace
- Console
- Pesquisar
- Postbot
- Runner
- Start Proxy
- Cookies
- Vault
- Trash
- 13:37
- 19/10/2024