

Faculdade Estácio - Polo Curitiba - Centro

Curso: Desenvolvimento Full Stack

Disciplina: Software sem segurança não serve!

Número da Turma: RPG0035

Semestre Letivo: 5

Integrante: Mariana Lucas Fernandes Onório

Repositório: <https://github.com/MariLFO/estacio-mundo5-missao-nivel-5>

Sumário:

Faculdade Estácio - Polo Curitiba - Centro	1
Sumário:	1
1. Título da Prática:	2
2. Objetivos da Prática:	2
3. Códigos do roteiro:	2
4. Resultados da execução dos códigos:	2
Execução dos códigos originais, sem tratamento:	2
• Endpoint '/api/auth/login'	2
• Endpoint '/api/auth/decrypt/:sessionid'	3
• Endpoint '/api/users/:sessionid'	3
• Endpoint '/api/contracts/:empresa/:inicio/:sessionid'	4
Execução dos códigos originais, após tratamento:	4
• Endpoint '/api/auth/login' - Como Admin	4
• Endpoint '/api/me' - Como Admin	5
• Endpoint '/api/users/' - Como Admin	5
• Endpoint '/api/contracts/empresa1/:inicio/' - Como Admin	6
• Endpoint '/api/contracts/empresa2/:inicio/' - Como Admin	6
• Endpoint '/api/auth/login' - Como Usuário padrão	7
• Endpoint '/api/me' - Como Usuário padrão	7
• Endpoint '/api/users/' - Como Usuário padrão	8
• Endpoint '/api/contracts/empresa1/:inicio/' - Como Usuário padrão	8
• Endpoint '/api/contracts/empresa2/:inicio/' - Como Usuário padrão	9
• Endpoint '/api/me' - Como token expirado	9

1. Título da Prática:

RPG0035

Software sem segurança não serve!

2. 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;

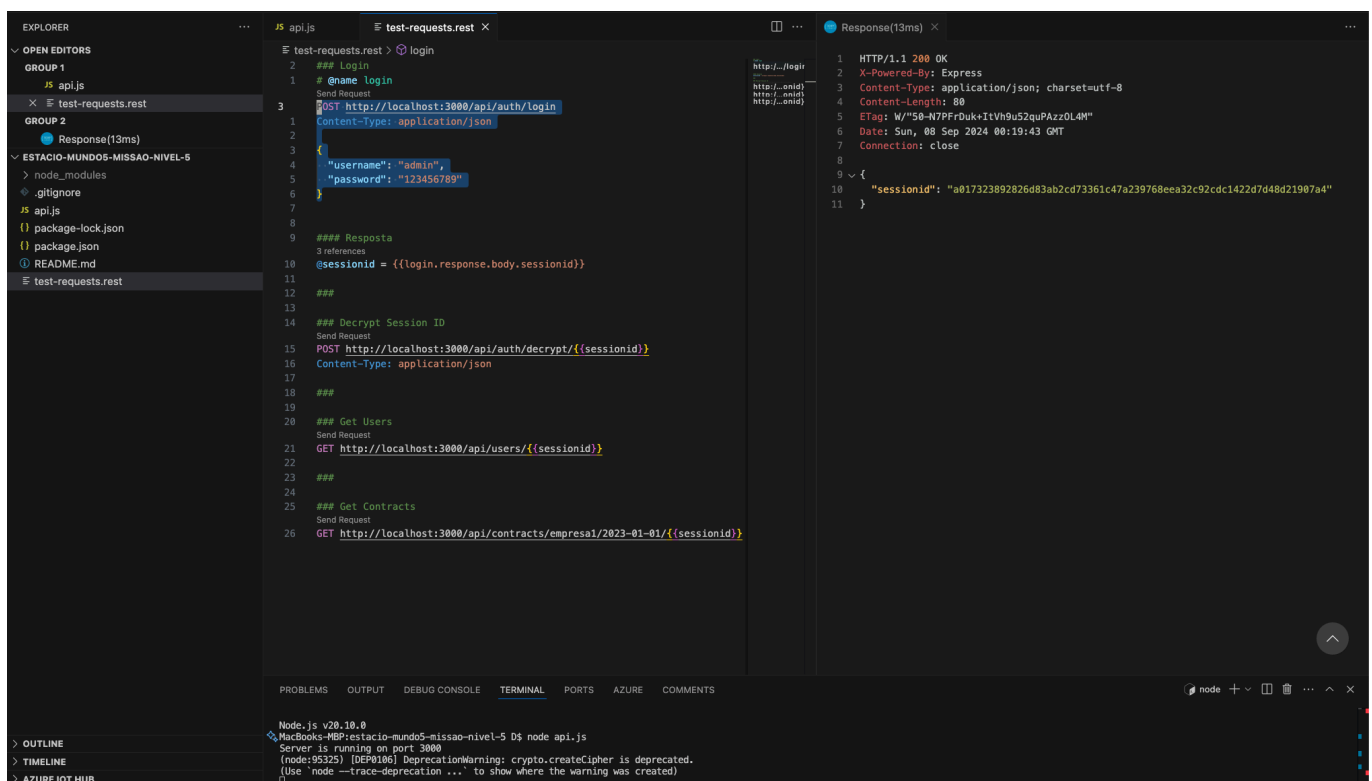
3. Códigos do roteiro:

<https://github.com/MariLFO/estacio-mundo5-missao-nivel-5>

4. Resultados da execução dos códigos:

Execução dos códigos originais, sem tratamento:

- Endpoint '/api/auth/login'



```
test-requests.rest x
1  ## Login
2  # @name login
3  Send Request
4  POST http://localhost:3000/api/auth/login
5  Content-Type: application/json
6  {
7    "username": "admin",
8    "password": "123456789"
9  }
10 ## Resposta
11 3 references
12 sessionid = {{login.response.body.sessionid}}
13
14 ## Decrypt Session ID
15 Send Request
16 POST http://localhost:3000/api/auth/decrypt/{{sessionid}}
17 Content-Type: application/json
18
19 ##
20 ## Get Users
21 Send Request
22 GET http://localhost:3000/api/users/{{sessionid}}
23
24 ##
25 ## Get Contracts
26 Send Request
27 GET http://localhost:3000/api/contracts/empresa1/2023-01-01/{{sessionid}}
```

```
Response(13ms) x
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 80
5 ETag: W/"58-N7PfrDuk+ItVh9u52quPAzz0LAM"
6 Date: Sun, 08 Sep 2024 00:19:43 GMT
7 Connection: close
8
9 {
10   "sessionId": "a017323892826d83ab2cd73361c47a239768eea32c92cdc1422d7d48d21907a4"
11 }
```

Node.js v20.10.0
MacBooks-MBP:estacio-mundo5-missao-nivel-5 D\$ node api.js
Server is running on port 3000
(node:92325) [DEP0186] DeprecationWarning: crypto.createCipher is deprecated.
(Use 'node --trace-deprecation ...' to show where the warning was created)

- Endpoint '/api/auth/decrypt/:sessionId'

The screenshot shows the VS Code interface with the REST client extension. The Explorer sidebar on the left shows the project structure with files like `api.js`, `package-lock.json`, `package.json`, and `test-requests.rest`. The main editor displays the `test-requests.rest` file, which contains several test requests. The first request is a POST to `http://localhost:3000/api/auth/login` with a JSON body containing `username: "admin"` and `password: "123456789"`. The response is a JSON object with a `sessionId` property. The second request is a POST to `http://localhost:3000/api/auth/decrypt/{{sessionId}}` with a `Content-Type: application/json` header. The response is a JSON object with a `data` property containing an array of user information. The third request is a GET to `http://localhost:3000/api/users/{{sessionId}}`. The fourth request is a GET to `http://localhost:3000/api/contracts/empresa1/2023-01-01/{{sessionId}}`. The bottom panel shows the terminal with the command `node api.js` and the output indicating the server is running on port 3000.

```
test-requests.rest
16 ## Login
17 # @name login
18 Send Request
19 POST http://localhost:3000/api/auth/login
20 Content-Type: application/json
21 {
22   "username": "admin",
23   "password": "123456789"
24 }
25 ## Resposta
26 3 references
27 @sessionId = {{login.response.body.sessionid}}
28 ##
29 ## Decrypt Session ID
30 Send Request
31 POST http://localhost:3000/api/auth/decrypt/{{sessionId}}
32 Content-Type: application/json
33 ##
34 ## Get Users
35 Send Request
36 GET http://localhost:3000/api/users/{{sessionId}}
37 ##
38 ## Get Contracts
39 Send Request
40 GET http://localhost:3000/api/contracts/empresa1/2023-01-01/{{sessionId}}
```

Response(10ms)

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 45
5 ETag: W/"2d-yFnaujoGnuXhU6KTb6LFxwDjw"
6 Date: Sun, 08 Sep 2024 00:20:12 GMT
7 Connection: close
8
9 {
10   "decryptedSessionId": "{\\\"usuario_id\\\":124}"
11 }
```

Node.js v20.10.0
MacBooks-MBP:estacio-mundo5-missao-nivel-5 D\$ node api.js
Server is running on port 3000
(node:95325) [DEP0106] DeprecationWarning: crypto.createCipher is deprecated.
(Use 'node --trace-deprecation ...' to show where the warning was created)

- Endpoint '/api/users/:sessionId'

The screenshot shows the VS Code interface with the REST client extension. The Explorer sidebar on the left shows the project structure with files like `api.js`, `package-lock.json`, `package.json`, and `test-requests.rest`. The main editor displays the `test-requests.rest` file, which contains several test requests. The first request is a POST to `http://localhost:3000/api/auth/login` with a JSON body containing `username: "admin"` and `password: "123456789"`. The response is a JSON object with a `sessionId` property. The second request is a POST to `http://localhost:3000/api/auth/decrypt/{{sessionId}}` with a `Content-Type: application/json` header. The response is a JSON object with a `data` property containing an array of user information. The third request is a GET to `http://localhost:3000/api/users/{{sessionId}}`. The fourth request is a GET to `http://localhost:3000/api/contracts/empresa1/2023-01-01/{{sessionId}}`. The bottom panel shows the terminal with the command `node api.js` and the output indicating the server is running on port 3000.

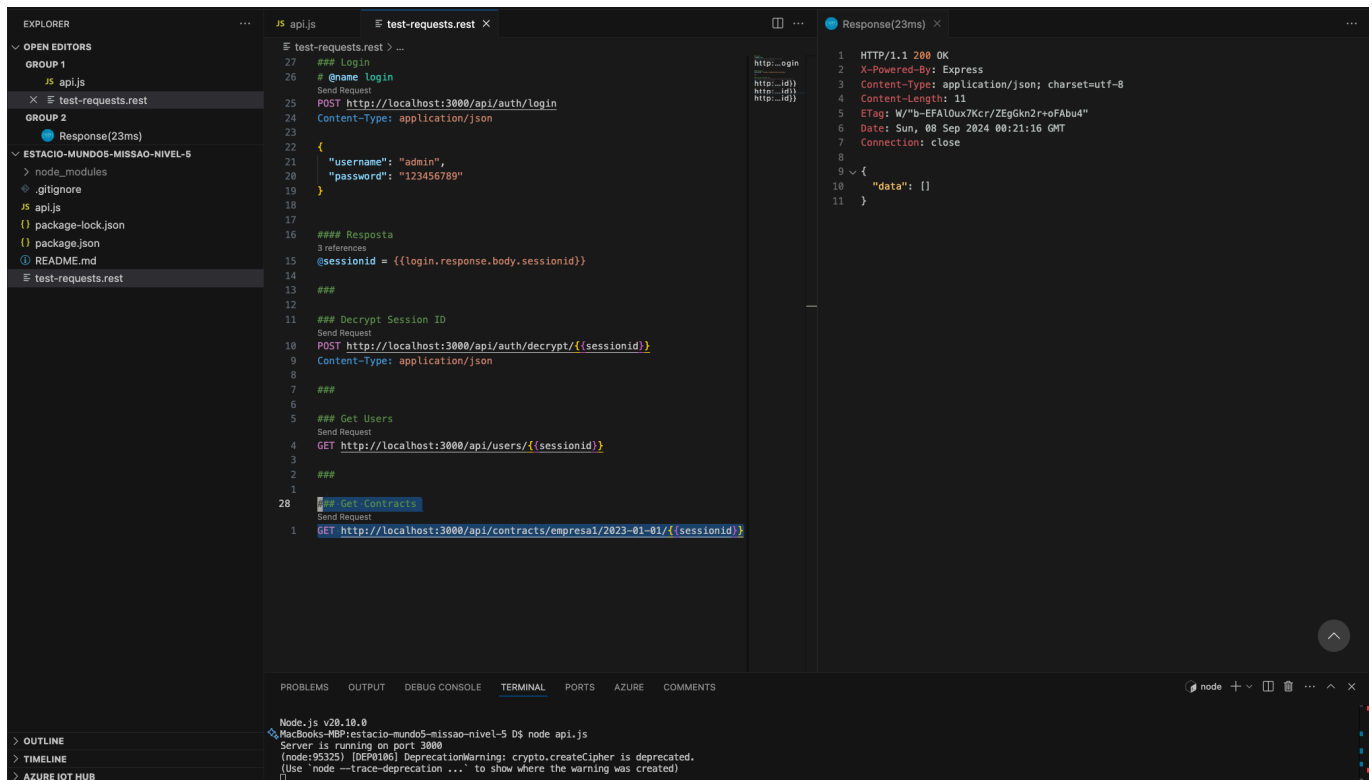
```
test-requests.rest
22 ## Login
23 # @name login
24 Send Request
25 POST http://localhost:3000/api/auth/login
26 Content-Type: application/json
27 {
28   "username": "admin",
29   "password": "123456789"
30 }
31 ## Resposta
32 3 references
33 @sessionId = {{login.response.body.sessionid}}
34 ##
35 ## Decrypt Session ID
36 Send Request
37 POST http://localhost:3000/api/auth/decrypt/{{sessionId}}
38 Content-Type: application/json
39 ##
40 ## Get Users
41 Send Request
42 GET http://localhost:3000/api/users/{{sessionId}}
43 ##
44 ## Get Contracts
45 Send Request
46 GET http://localhost:3000/api/contracts/empresa1/2023-01-01/{{sessionId}}
```

Response(37ms)

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 291
5 ETag: W/"123-SHq+WB5KATUhoY5G8H2iFgoI45Q"
6 Date: Sun, 08 Sep 2024 00:20:58 GMT
7 Connection: close
8
9 {
10   "data": [
11     {
12       "username": "user",
13       "password": "123456",
14       "id": 123,
15       "email": "user@dominio.com",
16       "perfil": "user"
17     },
18     {
19       "username": "admin",
20       "password": "123456789",
21       "id": 124,
22       "email": "admin@dominio.com",
23       "perfil": "admin"
24     },
25     {
26       "username": "colab",
27       "password": "123",
28       "id": 125,
29       "email": "colab@dominio.com",
30       "perfil": "user"
31     }
32   ]
33 }
```

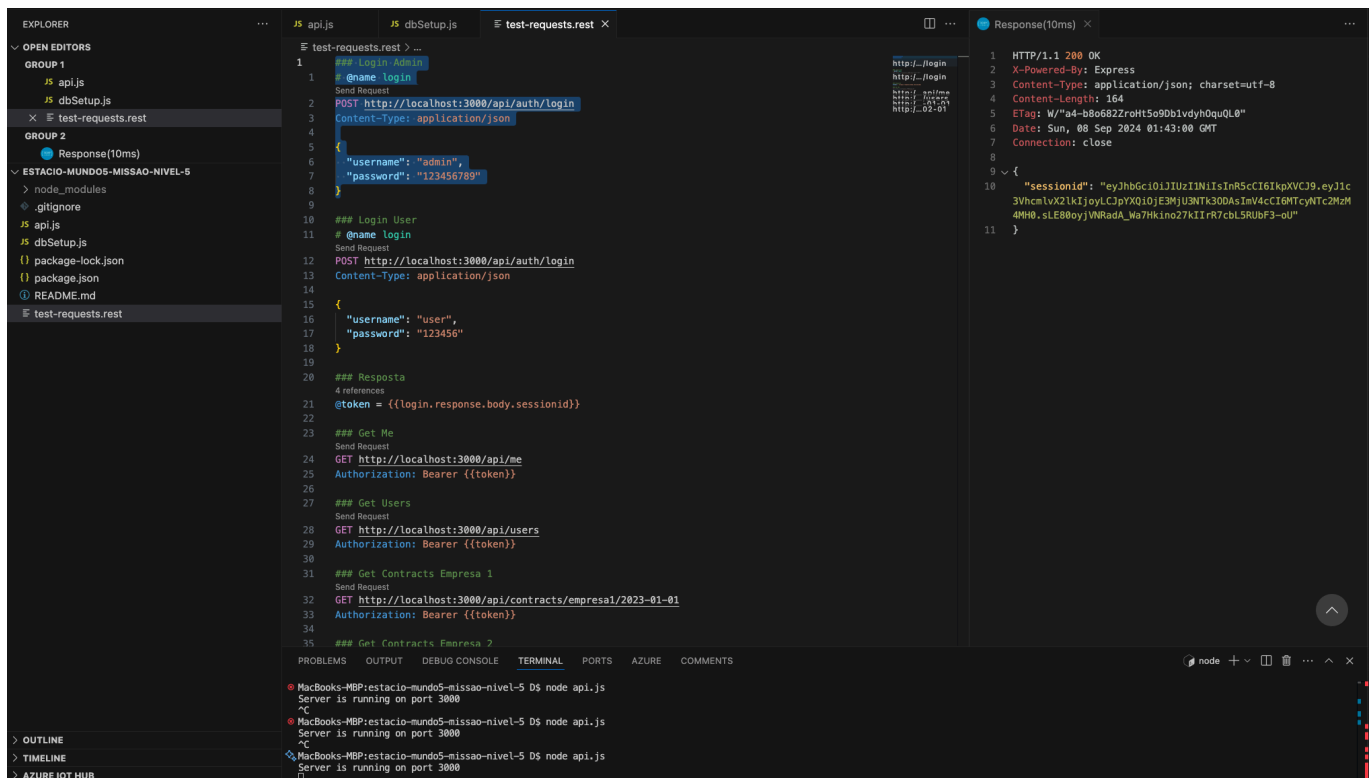
Node.js v20.10.0
MacBooks-MBP:estacio-mundo5-missao-nivel-5 D\$ node api.js
Server is running on port 3000
(node:95325) [DEP0106] DeprecationWarning: crypto.createCipher is deprecated.
(Use 'node --trace-deprecation ...' to show where the warning was created)

- Endpoint `/api/contracts/:empresa/:inicio/:sessionid'`



Execução dos códigos originais, após tratamento:

- Endpoint `'/api/auth/login'` - Como Admin



- Endpoint '/api/me' - Como Admin

The screenshot shows the VS Code interface with the REST client open. The request is a GET to `http://localhost:3000/api/me` with an `Authorization: Bearer {{token}}` header. The response is a 200 OK with the following headers: `X-Powered-By: Express`, `Content-Type: application/json; charset=utf-8`, `Content-Length: 81`, `Etag: W/"51-sH0HCSTsr0uYbBdGaSVBUdWY"`, `Date: Sun, 08 Sep 2024 01:44:03 GMT`, and `Connection: close`. The JSON body is:

```
{
  "data": {
    "id": 2,
    "username": "admin",
    "email": "admin@dominio.com",
    "perfil": "admin"
  }
}
```

- Endpoint '/api/users' - Como Admin

The screenshot shows the VS Code interface with the REST client open. The request is a GET to `http://localhost:3000/api/users` with an `Authorization: Bearer {{token}}` header. The response is a 200 OK with the following headers: `X-Powered-By: Express`, `Content-Type: application/json; charset=utf-8`, `Content-Length: 285`, `Etag: W/"11d-48Q+1isa0Usxn90h15gSBVY08rc"`, `Date: Sun, 08 Sep 2024 01:44:24 GMT`, and `Connection: close`. The JSON body is:

```
{
  "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"
    }
  ]
}
```

- Endpoint '/api/contracts/empresa1/:inicio/' - Como Admin

```
test-requests.rest
20 POST http://localhost:3000/api/auth/login
21 {
22   "username": "admin",
23   "password": "123456789"
24 }
25
26 ## Login User
27 # @name login
28 Send Request
29 POST http://localhost:3000/api/auth/login
30 Content-Type: application/json
31
32 {
33   "username": "user",
34   "password": "123456"
35 }
36
37 ## Resposta
38 4 references
39 @token = {{login.response.body.sessionid}}
40
41 ## Get Me
42 Send Request
43 GET http://localhost:3000/api/me
44 Authorization: Bearer {{token}}
45
46 ## Get Users
47 Send Request
48 GET http://localhost:3000/api/users
49 Authorization: Bearer {{token}}
50
51 ## Get Contracts Empresa 1
52 Send Request
53 GET http://localhost:3000/api/contracts/empresa1/2023-01-01
54 Authorization: Bearer {{token}}
55
56 ## Get Contracts Empresa 2
57 Send Request
58 GET http://localhost:3000/api/contracts/empresa2/2023-02-01
59 Authorization: Bearer {{token}}
```

```
Response(11ms)
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 67
5 ETag: W/"43-EJmDcvA18ewguTtPLXuzamVxUE"
6 Date: Sun, 08 Sep 2024 01:44:49 GMT
7 Connection: close
8
9 {
10   "data": [
11     {
12       "id": 1,
13       "empresa": "empresa1",
14       "data_inicio": "2023-01-01"
15     }
16   ]
17 }
```

- Endpoint '/api/contracts/empresa2/:inicio/' - Como Admin

```
test-requests.rest
32 POST http://localhost:3000/api/auth/login
33 {
34   "username": "admin",
35   "password": "123456789"
36 }
37
38 ## Login User
39 # @name login
40 Send Request
41 POST http://localhost:3000/api/auth/login
42 Content-Type: application/json
43
44 {
45   "username": "user",
46   "password": "123456"
47 }
48
49 ## Resposta
50 4 references
51 @token = {{login.response.body.sessionid}}
52
53 ## Get Me
54 Send Request
55 GET http://localhost:3000/api/me
56 Authorization: Bearer {{token}}
57
58 ## Get Users
59 Send Request
60 GET http://localhost:3000/api/users
61 Authorization: Bearer {{token}}
62
63 ## Get Contracts Empresa 1
64 Send Request
65 GET http://localhost:3000/api/contracts/empresa1/2023-01-01
66 Authorization: Bearer {{token}}
67
68 ## Get Contracts Empresa 2
69 Send Request
70 GET http://localhost:3000/api/contracts/empresa2/2023-02-01
71 Authorization: Bearer {{token}}
```

```
Response(8ms)
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 67
5 ETag: W/"43-rxwM7RBz3aMj+5DmVjUP6v69qQ"
6 Date: Sun, 08 Sep 2024 01:45:10 GMT
7 Connection: close
8
9 {
10   "data": [
11     {
12       "id": 2,
13       "empresa": "empresa2",
14       "data_inicio": "2023-02-01"
15     }
16   ]
17 }
```

- Endpoint '/api/auth/login' - Como Usuário padrão

The screenshot shows the VS Code interface with the REST client editor open. The request is a POST to `http://localhost:3000/api/auth/login` with a JSON body containing `username: "admin"` and `password: "123456789"`. The response is a 200 OK status with a JSON body containing a `sessionId`. The terminal shows the server running on port 3000.

```
10 ## Login Admin
11 # @name login
12 Send Request
13 POST http://localhost:3000/api/auth/login
14 Content-Type: application/json
15 {
16   "username": "admin",
17   "password": "123456789"
18 }
19 ## Resposta
20 4 references
21 @token = {{login.response.body.sessionId}}
22 ## Get Me
23 Send Request
24 GET http://localhost:3000/api/me
25 Authorization: Bearer {{token}}
26 ## Get Users
27 Send Request
28 GET http://localhost:3000/api/users
29 Authorization: Bearer {{token}}
30 ## Get Contracts Empresa 1
31 Send Request
32 GET http://localhost:3000/api/contracts/empresa1/2023-01-01
33 Authorization: Bearer {{token}}
34 ## Get Contracts Empresa 2
35 Send Request
36 GET http://localhost:3000/api/contracts/empresa2/2023-02-01
37 Authorization: Bearer {{token}}
```

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 164
5 ETag: W/"ad-0n8RrdI1c3MSPH2Vn2CEn3TGP1"
6 Date: Sun, 08 Sep 2024 01:45:37 GMT
7 Connection: close
8
9 {
10   "sessionId": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1IjoiMjVhcmVvX2k1Ij01E3MjU3NTk5MzcsImV4cCI6MTcyNTc2MzUzN3B0LjQ1YmYyYQhT46TCEQbP4Vg5Pr11-NK_hAGXcW4o"
11 }
```

- Endpoint '/api/me' - Como Usuário padrão

The screenshot shows the VS Code interface with the REST client editor open. The request is a GET to `http://localhost:3000/api/me` with an `Authorization: Bearer {{token}}` header. The response is a 200 OK status with a JSON body containing user information. The terminal shows the server running on port 3000.

```
21 POST http://localhost:3000/api/auth/login
18 {
19   "username": "admin",
20   "password": "123456789"
21 }
22 ## Resposta
23 4 references
24 @token = {{login.response.body.sessionId}}
25 ## Get Me
26 Send Request
27 GET http://localhost:3000/api/me
28 Authorization: Bearer {{token}}
29 ## Get Users
30 Send Request
31 GET http://localhost:3000/api/users
32 Authorization: Bearer {{token}}
33 ## Get Contracts Empresa 1
34 Send Request
35 GET http://localhost:3000/api/contracts/empresa1/2023-01-01
36 Authorization: Bearer {{token}}
37 ## Get Contracts Empresa 2
38 Send Request
39 GET http://localhost:3000/api/contracts/empresa2/2023-02-01
40 Authorization: Bearer {{token}}
```

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 78
5 ETag: W/"4e-nv0yK8zbl5iAGb2Y/jkyDopWek"
6 Date: Sun, 08 Sep 2024 01:45:59 GMT
7 Connection: close
8
9 {
10   "data": {
11     "id": 1,
12     "username": "user",
13     "email": "user@dominio.com",
14     "perfil": "user"
15   }
16 }
```

- Endpoint '/api/users/' - Como Usuário padrão

The screenshot shows the VS Code interface with the REST client extension. The Explorer panel on the left shows the project structure with files like api.js, dbSetup.js, and test-requests.rest. The main editor displays the test-requests.rest file with the following content:

```
27 POST http://localhost:3000/api/auth/login
28 {
29   "username": "admin",
30   "password": "123456789"
31 }
32
33 ## Login User
34 # @name login
35 Send Request
36 POST http://localhost:3000/api/auth/login
37 Content-Type: application/json
38
39 {
40   "username": "user",
41   "password": "123456"
42 }
43
44 ## Resposta
45 4 references
46 @token = {{login.response.body.sessionid}}
47
48 ## Get Me
49 Send Request
50 GET http://localhost:3000/api/me
51 Authorization: Bearer {{token}}
52
53 ## Get Users
54 Send Request
55 GET http://localhost:3000/api/users
56 Authorization: Bearer {{token}}
57
58 ## Get Contracts Empresa 1
59 Send Request
60 GET http://localhost:3000/api/contracts/empresa1/2023-01-01
61 Authorization: Bearer {{token}}
62
63 ## Get Contracts Empresa 2
64 Send Request
65 GET http://localhost:3000/api/contracts/empresa2/2023-02-01
66 Authorization: Bearer {{token}}
```

The right panel shows the response for the GET /api/users/ request:

```
1 HTTP/1.1 403 Forbidden
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 36
5 ETag: W/"24-2NwC28ycIzJBXKV/sY/hp855DAI"
6 Date: Sun, 08 Sep 2024 01:46:09 GMT
7 Connection: close
8
9 {
10   "message": "Forbidden: Admins only"
11 }
```

The bottom panel shows the terminal with the command 'node api.js' and the output 'Server is running on port 3000'.

- Endpoint '/api/contracts/empresa1/:inicio/' - Como Usuário padrão

The screenshot shows the VS Code interface with the REST client extension. The Explorer panel on the left shows the project structure with files like api.js, dbSetup.js, and test-requests.rest. The main editor displays the test-requests.rest file with the following content:

```
29 POST http://localhost:3000/api/auth/login
30 {
31   "username": "admin",
32   "password": "123456789"
33 }
34
35 ## Login User
36 # @name login
37 Send Request
38 POST http://localhost:3000/api/auth/login
39 Content-Type: application/json
40
41 {
42   "username": "user",
43   "password": "123456"
44 }
45
46 ## Resposta
47 4 references
48 @token = {{login.response.body.sessionid}}
49
50 ## Get Me
51 Send Request
52 GET http://localhost:3000/api/me
53 Authorization: Bearer {{token}}
54
55 ## Get Users
56 Send Request
57 GET http://localhost:3000/api/users
58 Authorization: Bearer {{token}}
59
60 ## Get Contracts Empresa 1
61 Send Request
62 GET http://localhost:3000/api/contracts/empresa1/2023-01-01
63 Authorization: Bearer {{token}}
64
65 ## Get Contracts Empresa 2
66 Send Request
67 GET http://localhost:3000/api/contracts/empresa2/2023-02-01
68 Authorization: Bearer {{token}}
```

The right panel shows the response for the GET /api/contracts/empresa1/:inicio/ request:

```
1 HTTP/1.1 403 Forbidden
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 36
5 ETag: W/"24-2NwC28ycIzJBXKV/sY/hp855DAI"
6 Date: Sun, 08 Sep 2024 01:47:52 GMT
7 Connection: close
8
9 {
10   "message": "Forbidden: Admins only"
11 }
```

The bottom panel shows the terminal with the command 'node api.js' and the output 'Server is running on port 3000'.

- Endpoint '/api/contracts/empresa2/:inicio/' - Como Usuário padrão

The screenshot shows the VS Code interface with the REST client open. The request is a GET to `http://localhost:3000/api/contracts/empresa2/2023-02-01` with an authorization header `Authorization: Bearer {{token}}`. The response is a 403 Forbidden error with the message "Forbidden: Admins only".

```
test-requests.rest
33 POST http://localhost:3000/api/auth/login
38 {
39   "username": "admin",
40   "password": "123456789"
41 }
42
43 ## Login User
44 # @name login
45 Send Request
46 POST http://localhost:3000/api/auth/login
47 Content-Type: application/json
48 {
49   "username": "user",
50   "password": "123456"
51 }
52
53 ## Resposta
54 4 references
55 @token = {{login.response.body.sessionid}}
56
57 ## Get Me
58 Send Request
59 GET http://localhost:3000/api/me
60 Authorization: Bearer {{token}}
61
62 ## Get Users
63 Send Request
64 GET http://localhost:3000/api/users
65 Authorization: Bearer {{token}}
66
67 ## Get Contracts Empresa 1
68 Send Request
69 GET http://localhost:3000/api/contracts/empresa1/2023-01-01
70 Authorization: Bearer {{token}}
71
72
73 ## Get Contracts Empresa 2
74 Send Request
75 GET http://localhost:3000/api/contracts/empresa2/2023-02-01
76 Authorization: Bearer {{token}}
```

Response(10ms)

```
1 HTTP/1.1 403 Forbidden
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 36
5 ETag: W/"24-ZNwC28ycIzJBXKV/sY/hp8550AI"
6 Date: Sun, 08 Sep 2024 01:48:11 GMT
7 Connection: close
8
9 {
10   "message": "Forbidden: Admins only"
11 }
```

- Endpoint '/api/me' - Como token expirado

The screenshot shows the VS Code interface with the REST client open. The request is a GET to `http://localhost:3000/api/me` with an authorization header `Authorization: Bearer {{token}}`. The response is a 403 Forbidden error with the message "Invalid token".

```
test-requests.rest
23 ## Login Admin
22 # @name login
21 Send Request
20 POST http://localhost:3000/api/auth/login
19 Content-Type: application/json
18 {
17   "username": "admin",
16   "password": "123456789"
15 }
14
13 ## Login User
12 # @name login
11 Send Request
10 POST http://localhost:3000/api/auth/login
9 Content-Type: application/json
8 {
7   "username": "user",
6   "password": "123456"
5 }
4
3 ## Resposta
2 4 references
1 @token = {{login.response.body.sessionid}}
24
25 ## Get Me
26 Send Request
27 GET http://localhost:3000/api/me
28 Authorization: Bearer {{token}}
29
30 ## Get Users
31 Send Request
32 GET http://localhost:3000/api/users
33 Authorization: Bearer {{token}}
34
35 ## Get Contracts Empresa 1
36 Send Request
37 GET http://localhost:3000/api/contracts/empresa1/2023-01-01
38 Authorization: Bearer {{token}}
39
40 ## Get Contracts Empresa 2
```

Response(10ms)

```
1 HTTP/1.1 403 Forbidden
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 27
5 ETag: W/"1b-w408U+0Ll+SArwAX18RvHmp1Q"
6 Date: Sun, 08 Sep 2024 01:48:56 GMT
7 Connection: close
8
9 {
10   "message": "Invalid token"
11 }
```