



Missão Prática – Mundo 5 – Nível 5

Andrey Haertel Aires - Matrícula: 2021.07.22851-2

Polo Centro - Palhoça –SC

**RPG0035 - SOFTWARE SEM SEGURANÇA NÃO SERVE!**

**T 9001 – 5º Semestre Letivo**

Github: [https://github.com/AndreyHaires/MissaoPraticaMundo5\\_N5](https://github.com/AndreyHaires/MissaoPraticaMundo5_N5)

## Refatorar o método de criptografia substituindo “session-id” por tokens JWT.

Antes o código usava crypto para criptografar os dados do usuário, gerando o session-id:

```
JS app_original.js > ...
95 const secretKey = 'nomedaempresa';
96
97 function encrypt(text) {
98   const cipher = crypto.createCipher('aes-256-cbc', secretKey);
99   let encrypted = cipher.update(text, 'utf8', 'hex');
100   encrypted += cipher.final('hex');
101   return encrypted;
102 }
```

Agora ajustado utiliza **JWT** para gerar o token de autenticação com jsonwebtoken:

```
JS app.js X
JS app.js > app.post('/api/auth/login') callback > token
53 app.post('/api/auth/login', (req, res) => {
54
55   const userData = doLogin(credentials);
56
57   if (userData) {
58     const token = jwt.sign(
59       {
60         id: userData.id,
61         perfil: userData.perfil,
62       },
63       SECRET_KEY,
64       { expiresIn: '1h' }
65     );
66     return res.json({ token });
67   } else {
68     return res.status(401).json({ message: 'Invalid credentials' });
69   }
70 });
71
```

## Traferir o token através do header da requisição.

Antes o token (session-id) era enviado pela URI:

```
JS app_original.js > app.get('/api/users/:sessionid') callback
44
45 // Endpoint para recuperação dos dados de todos os usuários cadastrados
46 app.get('/api/users/:sessionid', (req, res) => {
47   const sessionid = req.params.sessionid;
48   const perfil = getPerfil(sessionid);
49
50   if (perfil !== 'admin') {
51     res.status(403).json({ message: 'Forbidden' });
52   } else {
53     res.status(200).json({ data: users });
54   }
55 });
56
```

Foi ajustado, o token agora é enviado no header Authorization:

```
JS app.js x
JS app.js > app.get('/api/users') callback > message
72 // Endpoint para recuperação dos dados de todos os usuários (somente admin)
73 app.get('/api/users', verifyToken, (req, res) => {
74   if (req.user.perfil !== 'admin') {
75     return res.status(403).json({ message: 'Forbidden: Admin access only' });
76   }
77   res.status(200).json({ data: users });
78 });
79
```

**Validar o token de segurança em cada requisição.**

Não havia validação de expiração ou identidade do usuário em cada requisição.

```
JS app_original.js
JS app_original.js > Repository > execute
112
113 // Recupera o perfil do usuário através da session-id
114 function getPerfil(sessionId) {
115   const user = JSON.parse(decrypt(sessionId));
116
117   const userData = users.find(item => {
118     if (parseInt(user.usuario_id) === parseInt(item.id)) return item;
119   });
120   return userData?.perfil || 'user';
121 }
122
```

Foi adicionado o middleware verifyToken para validação do token:

```
JS app.js x
JS app.js > app.listen() callback
22
23 // Middleware para validar o token JWT
24 function verifyToken(req, res, next) {
25   const authHeader = req.headers['authorization'];
26   if (!authHeader) {
27     return res.status(401).json({ message: 'Header de autorização ausente' });
28   }
29
30   const token = authHeader.split(' ')[1]; // Formato esperado: Bearer <token>
31   if (!token) {
32     return res.status(401).json({ message: 'Token não fornecido' });
33   }
34
35   try {
36     const decodedToken = jwt.verify(token, SECRET_KEY);
37
38     // Verifica expiração
39     const currentTime = Math.floor(Date.now() / 1000);
40     if (decodedToken.exp < currentTime) {
41       return res.status(403).json({ message: 'Token expirado' });
42     }
43
44     // Adiciona os dados do usuário na requisição
45     req.user = decodedToken;
46     next();
47   } catch (error) {
48     return res.status(403).json({ message: 'Token Invalido', error: error.message });
49   }
50 }
51
```

## Controle de acesso com base no perfil do usuário.

Antes:

```
JS app_original.js X
JS app_original.js > app.get('/api/users/:sessionid') callback
44
45 // Endpoint para recuperação dos dados de todos os usuários cadastrados
46 app.get('/api/users/:sessionid', (req, res) => {
47   const sessionid = req.params.sessionid;
48   const perfil = getPerfil(sessionid);
49
50   if (perfil !== 'admin') {
51     res.status(403).json({ message: 'Forbidden' });
52   } else {
53     res.status(200).json({ data: users });
54   }
55 });
56
```

Agora os endpoints validam o perfil:

```
JS app.js X
JS app.js > ...
71
72 // Endpoint para recuperação dos dados de todos os usuários (somente admin)
73 app.get('/api/users', verifyToken, (req, res) => {
74   if (req.user.perfil !== 'admin') {
75     return res.status(403).json({ message: 'Forbidden: Somente acesso de admin' });
76   }
77   res.status(200).json({ data: users });
78 });
79
```

Novo endpoint para recuperação do usuário logado:

```
JS app.js X
JS app.js > app.get('/api/user/profile') callback
92
93 // Endpoint para dados do usuário logado
94 app.get('/api/user/profile', verifyToken, (req, res) => {
95   const userData = users.find(user => user.id === req.user.id);
96   res.status(200).json({ user: userData });
97 });
98
```

## Proteção contra SQL Injection.

A query era construída de forma vulnerável:

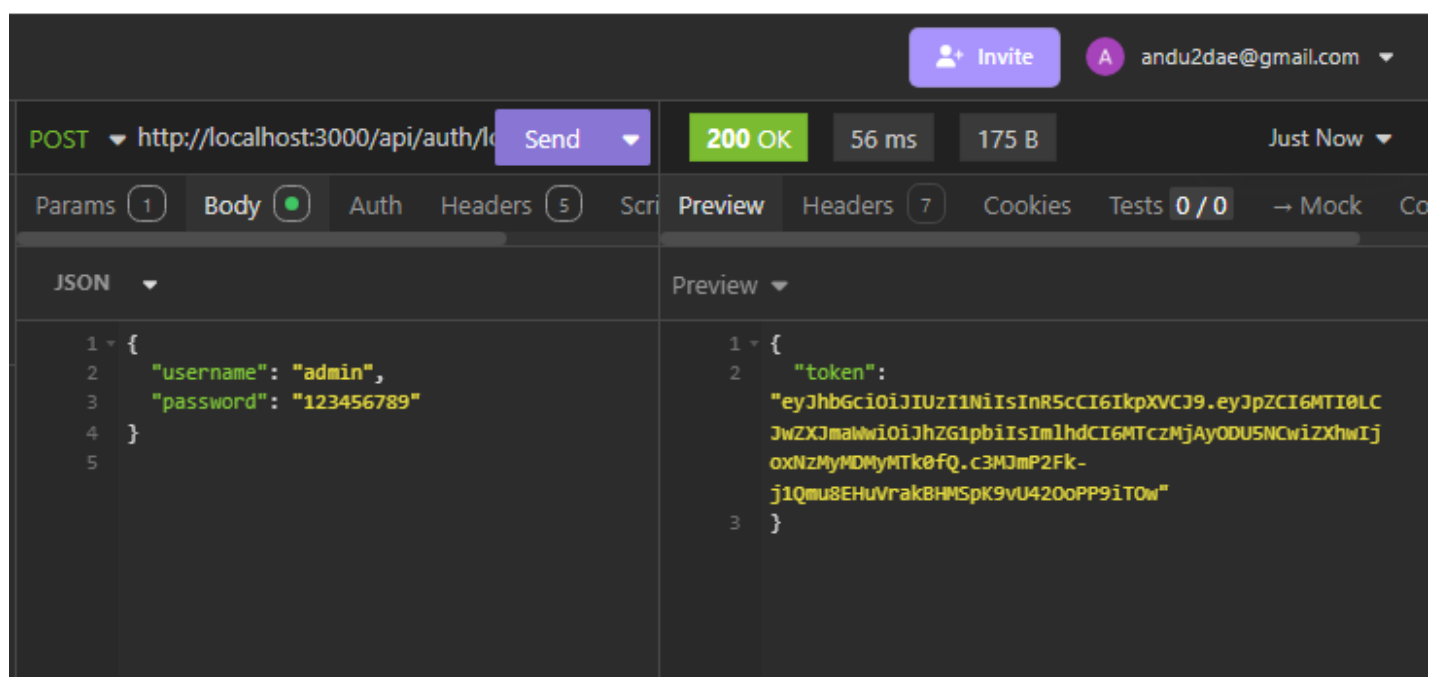
```
JS app_original.js X
JS app_original.js > ...
130 // Recupera, no banco de dados, os dados dos contratos
131 // Método não funcional, servindo apenas para fins de estudo
132 function getContracts(empresa, inicio) {
133   const repository = new Repository();
134   const query = `Select * from contracts Where empresa = '${empresa}' And data_inicio = '${inicio}'`;
135
136   const result = repository.execute(query);
137   return result;
138 }
139 |
```

Foi adicionada validação com expressões regulares:

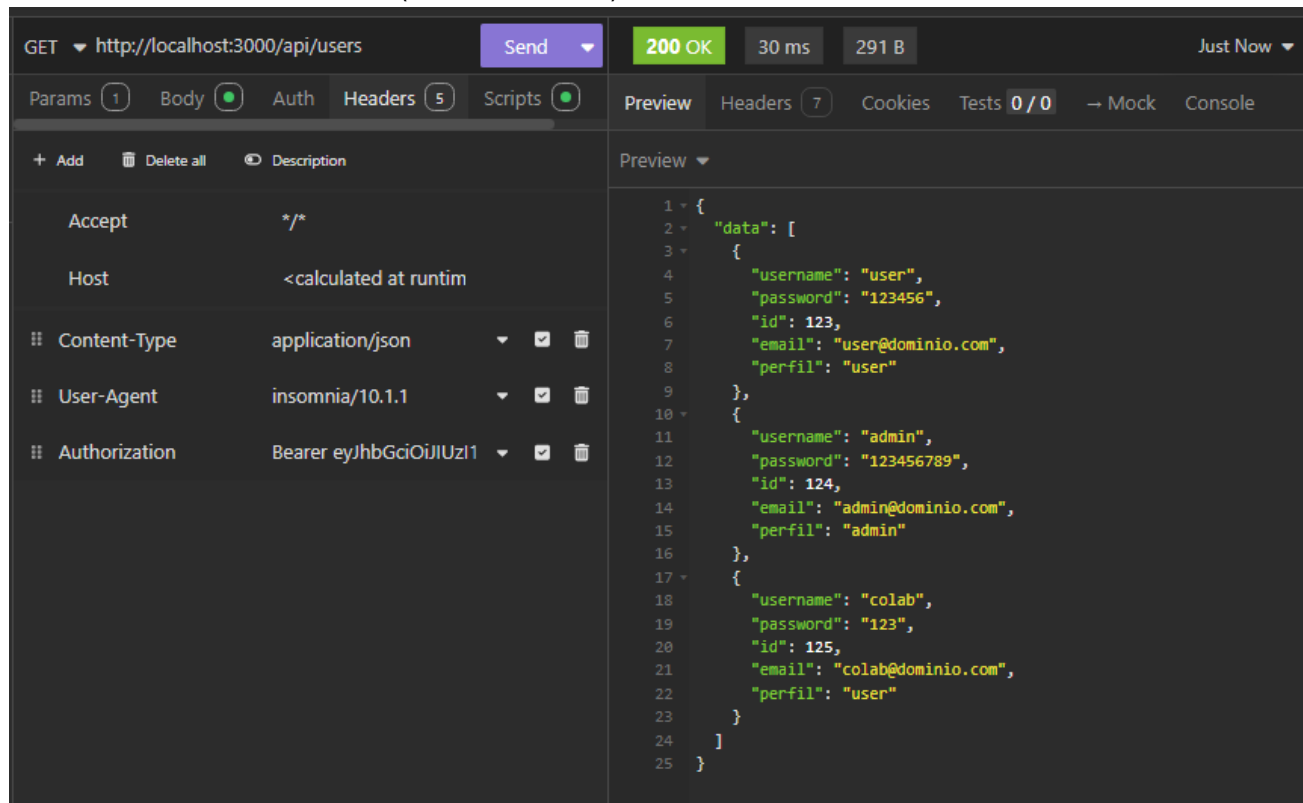
```
JS app.js X
JS app.js > ...
115 // Método ajustado para evitar SQL Injection
116 function getContracts(empresa, inicio) {
117   if (!/^([a-zA-Z0-9]+)$/.test(empresa) || !/^\d{4}-\d{2}-\d{2}$/.test(inicio)) {
118     return null; // Dados inválidos
119   }
120
121   const repository = new Repository();
122   const query = `SELECT * FROM contracts WHERE empresa = ? AND data_inicio = ?`;
123   return repository.execute(query, [empresa, inicio]); // Query preparada com parâmetros
124 }
125
126 module.exports = app;
127 |
```

## Testes dos ajustes realizados utilizando o Insomnia.

Teste de LOGIN como ADMIN, como resposta o TOKEN.



## Testando Acesso aos Usuários (Somente Admin).



GET `http://localhost:3000/api/users` **Send** **200 OK** 30 ms 291 B Just Now

Params (1) Body Headers (5) Scripts Preview Headers (7) Cookies Tests 0/0 → Mock Console

Accept `/*/*`

Host `<calculated at runtime>`

Content-Type `application/json` ☒ ☐

User-Agent `insomnia/10.1.1` ☒ ☐

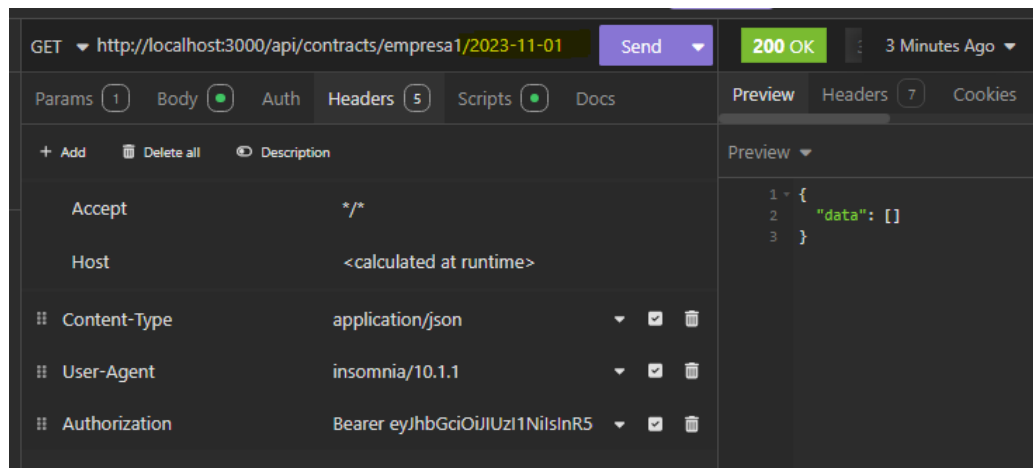
Authorization `Bearer eyJhbGciOiJIUzI1` ☒ ☐

Preview

```
1 {
2   "data": [
3     {
4       "username": "user",
5       "password": "123456",
6       "id": 123,
7       "email": "user@dominio.com",
8       "perfil": "user"
9     },
10    {
11      "username": "admin",
12      "password": "123456789",
13      "id": 124,
14      "email": "admin@dominio.com",
15      "perfil": "admin"
16    },
17    {
18      "username": "colab",
19      "password": "123",
20      "id": 125,
21      "email": "colab@dominio.com",
22      "perfil": "user"
23    }
24  ]
25 }
```

## Recuperação de Contratos (Validando Parâmetros).

Dados Válidos:



GET `http://localhost:3000/api/contracts/empresa1/2023-11-01` **Send** **200 OK** 3 Minutes Ago

Params (1) Body Headers (5) Scripts Docs Preview Headers (7) Cookies

Accept `/*/*`

Host `<calculated at runtime>`

Content-Type `application/json` ☒ ☐

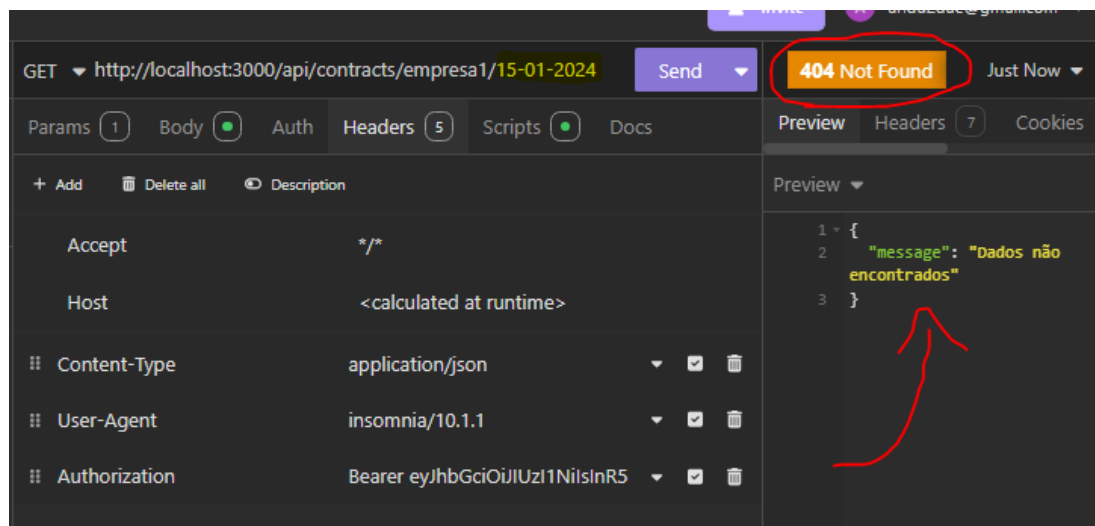
User-Agent `insomnia/10.1.1` ☒ ☐

Authorization `Bearer eyJhbGciOiJIUzI1NiIsInR5` ☒ ☐

Preview

```
1 {
2   "data": []
3 }
```

Dados inválidos – Erro 404 Not Found: (Data em formato incorreto)



GET `http://localhost:3000/api/contracts/empresa1/15-01-2024` **Send** **404 Not Found** Just Now

Params (1) Body Headers (5) Scripts Docs Preview Headers (7) Cookies

Accept `/*/*`

Host `<calculated at runtime>`

Content-Type `application/json` ☒ ☐

User-Agent `insomnia/10.1.1` ☒ ☐

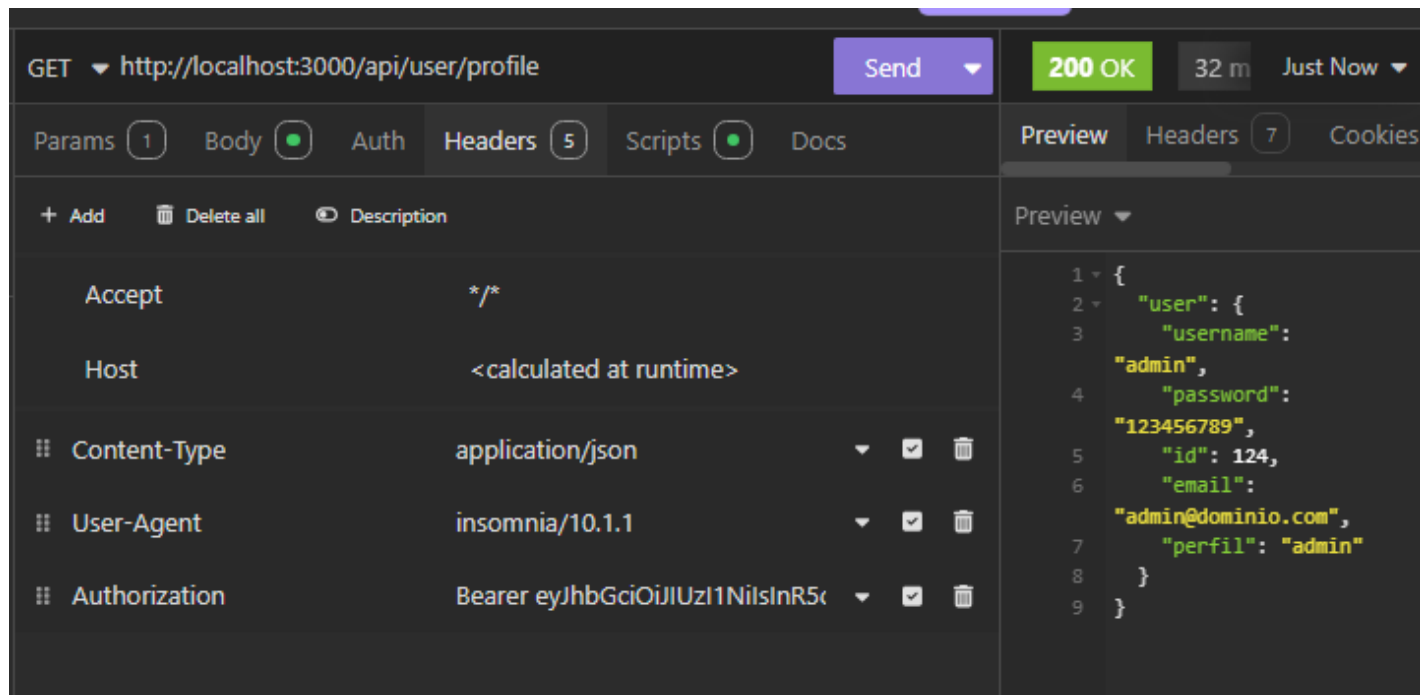
Authorization `Bearer eyJhbGciOiJIUzI1NiIsInR5` ☒ ☐

Preview

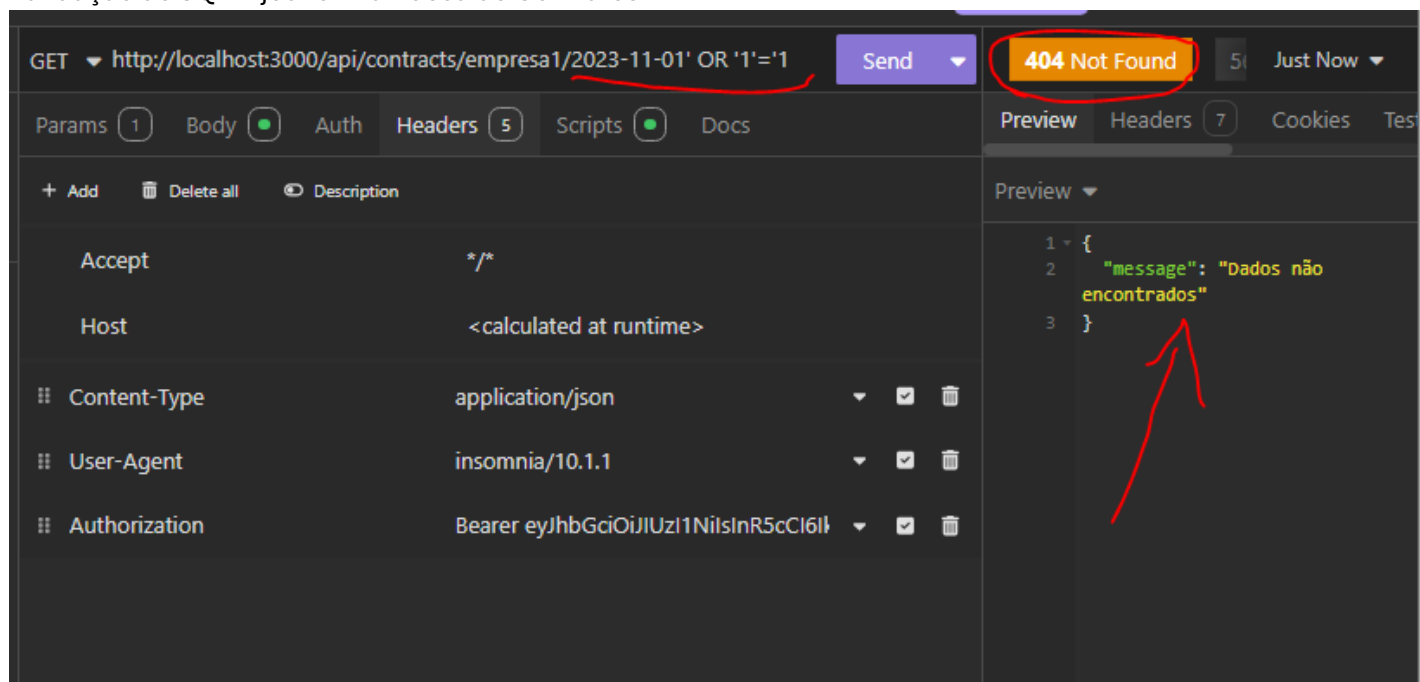
```
1 {
2   "message": "Dados não encontrados"
3 }
```

A red arrow points to the error message in the preview pane.

### Recuperação do Perfil do Usuário Logado:

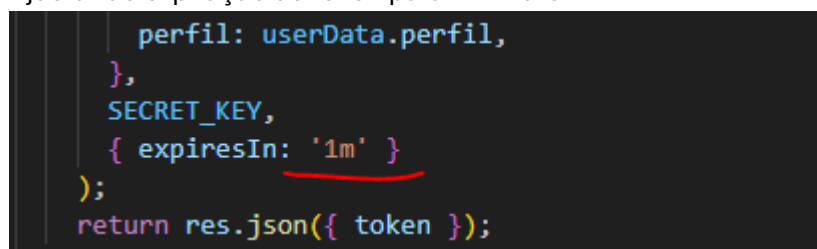


### Validação de SQL Injection na Busca de Contratos:

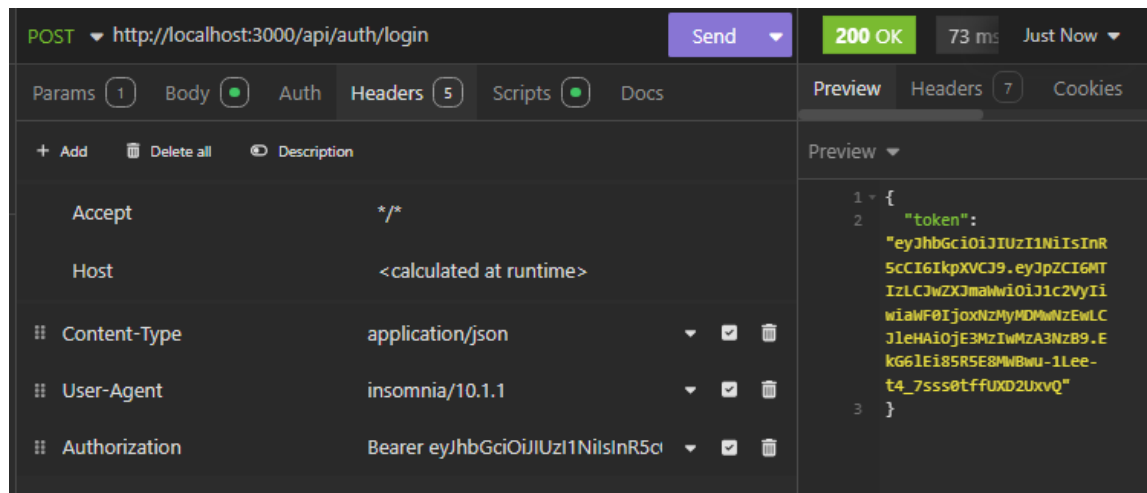


### Testando Expiração do Token.

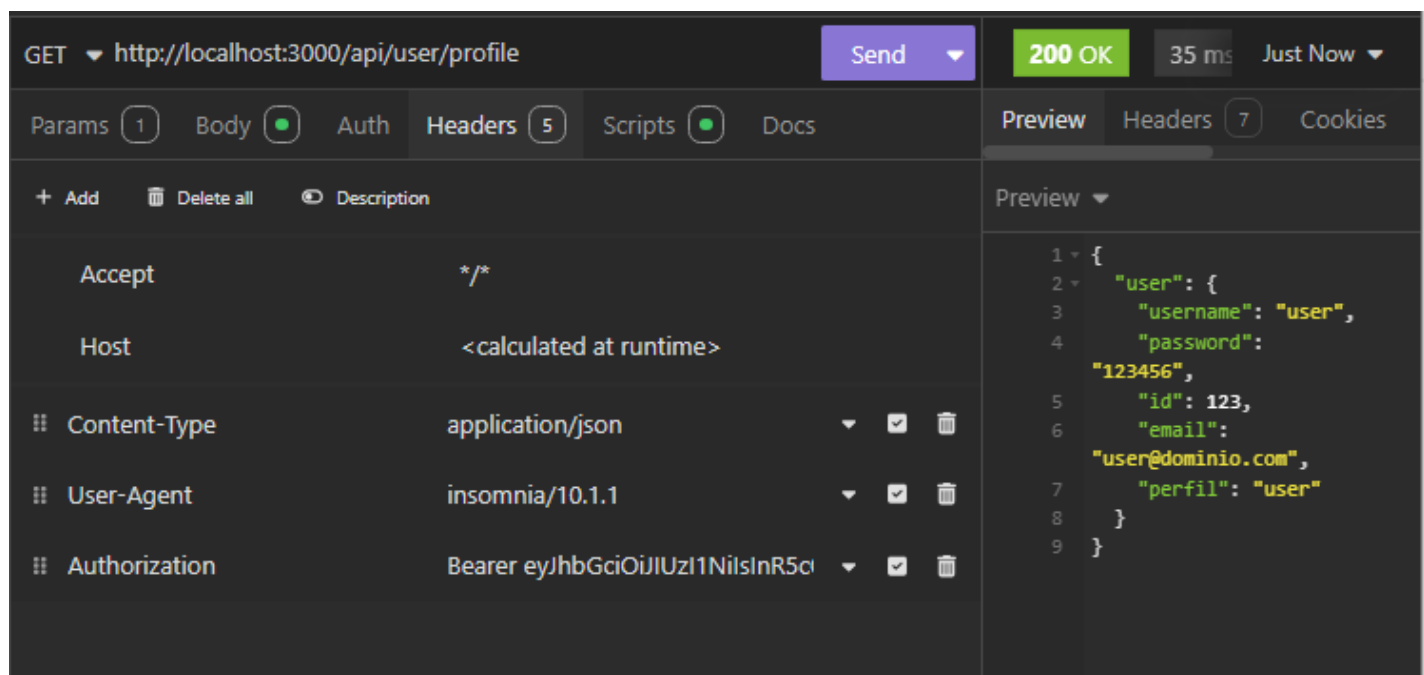
Ajustando expiração do token para 1 minuto:



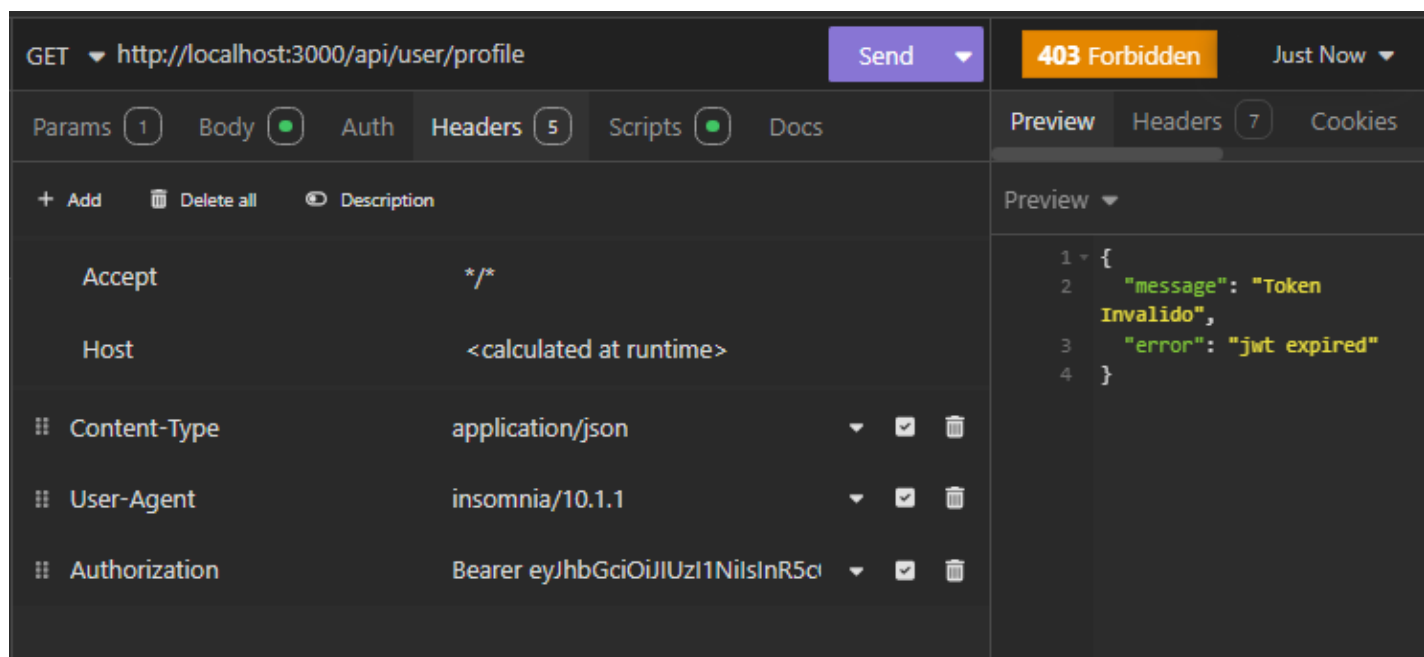
Logando como USER:



Testando requisição antes de 1 minuto:





Após 1 minuto, token expirado:








Testando restrição para usuário não administrador:

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




Send 










403 Forbidden


Just Now 

Params 1 Body  Auth Headers 5 Scripts  Docs

Preview Headers 7 Cookies

 Add  Delete all  Description

Accept	*/*		
Host	<calculated at runtime>		
Content-Type	application/json		 
User-Agent	insomnia/10.1.1		 
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6Ikpz...		 

Preview 

```
1 {
2   "message": "Forbidden:
3   Somente acesso de admin"
}
```