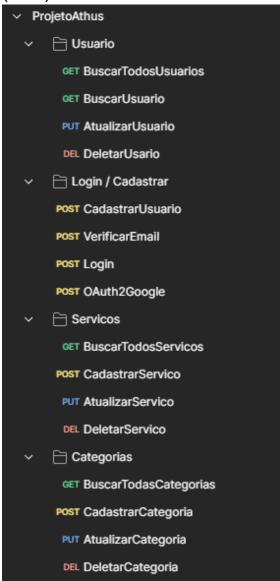
Documentação da Endpoints API

Está documentação tem como objetivo deixar mais claros todos os endpoints da API até o momento.

Todos estão separados em suas devidas pastas, e são marcados pelos métodos que exercem como – GET, POST, PUT e DELETE (DEL)



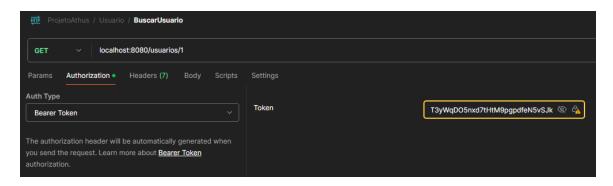
Na pasta é referenciado qual é o endpoint (Segue exemplo - Categorias = localhost:8080/categorias)

Endpoints que necessitam de autenticação, devem ser utilizados o Bearer Token gerado. Exemplo:

Ao logar é gerado o seu Bearer Token

```
*headers*: {},
*body': {
*accessToken*: "eyJhbGci013IUzIINiIsInPBcCI6IkpXVCJ9.
*accessToken*: "eyJhbGci013IUzIINiIsInPBcCI6IkpXVCJ9.
*eyJpc3Mi013b2dpbi1hdXRoLNFwaSIsInNiYiI6ImpvcZYzbZFyZXNnYMxZYMByMUBnbNFpbC6jb28iLCJ@exBlJjoiYNNJZXNzLXRvaZVuIiwiZXhwIjoxNzQz00MxMTE3fQ.cy9ZfyQklg7GFSTj--Yeo4863itmBvcCtEjeHUfuu3E*,
```

É necessário utilizar ele na aba "Authorization".



Métodos autenticação

POST /auth/cadastrar

Descrição: Cria o cadastro inicial do usuário.

Autenticação: Não requer autenticação.

```
Exemplo de Body:
```

```
{
"nome": "José",
"email": "josesoaresgalvao21@gmail.com",
"senha": "123"
}
```

Respostas:

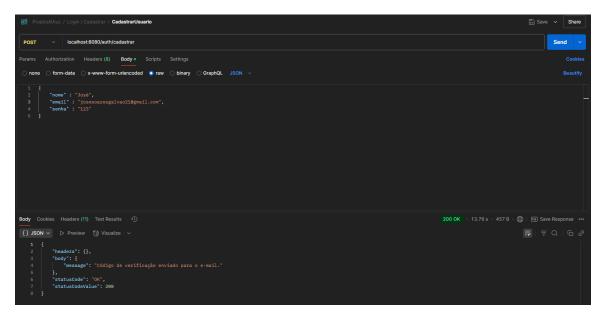
• 200 OK:

```
{
    "body": {
```

```
"message": "Código de verificação enviado para o e-mail."
}

• 409 Conflict:

{
   "message": "Não é possível cadastrar o usuário, pois o e-mail já está em uso."
}
```



POST /auth/validarcodigo

Descrição: Valida o código enviado por e-mail.

Autenticação: Não requer autenticação.

```
Exemplo de Body:
{
    "email": "josesoaresgalvao21@gmail.com",
    "codigo": "113398"
}
```

Respostas:

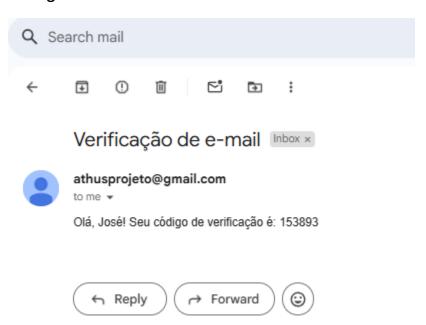
200 OK:"body": {
 "refreshToken": "...",
 "accessToken": "..."
}

• 422 Unprocessable Entity:

Nenhum código foi gerado para esse e-mail.

• 400 Bad Request:

Código inválido. Tentativas restantes: 4



```
| Body | Cookies | Headers (II) | Test Results | O | December (II) | December (II)
```

POST /auth/login

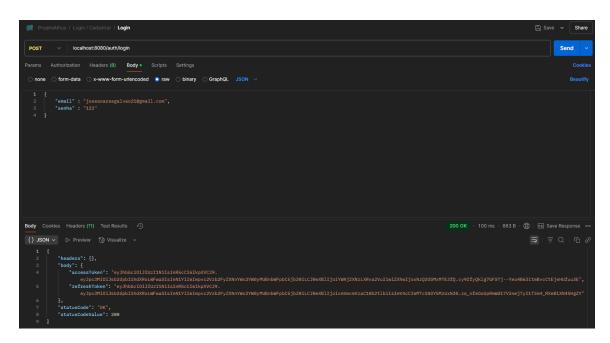
Descrição: Realiza o login no sistema.

Autenticação: Não requer autenticação.

```
Exemplo de Body:
{
    "email": "josesoaresgalvao21@gmail.com",
    "senha": "123"
}
Respostas:
• 200 OK:
{
    "body": {
        "refreshToken": "...",
        "accessToken": "..."
}
}
```

• 400 Bad Request:

Senha incorreta!



Métodos do usuário

GET /usuarios

Descrição: Lista todos os usuários.

Autenticação: Requer autenticação Bearer Token.

Respostas:

```
• 200 OK:
```

```
[{
    "id": 1,
    "nome": "José",
    "email": "josesoaresgalvao21@gmail.com",
    ...
}]
```

• 403 Forbidden:

GET /usuarios/{id}

Descrição: Retorna dados de um usuário por ID.

Autenticação: Requer autenticação Bearer Token.

Parâmetros:

• - id - ID do usuário

Respostas:

• 200 OK:

Usuário encontrado

• 403 Forbidden:

```
Body Cookles Headers (1) Test Results | ①

Body Cookles Headers (1) Test Results | ②

Body Cookles Headers (1) Test Results | ③

Body Cookles Headers (1)
```

PUT /usuarios/{id}

Descrição: Atualiza dados do usuário.

Autenticação: Requer autenticação Bearer Token.

Parâmetros:

• - id - ID do usuário

Respostas:

• 200 OK:

Usuário atualizado com sucesso

• 403 Forbidden:

```
Frums Authorization * Headers (I) Body * Scripts Settings Coakies

| Promise | Promise
```

DELETE /usuarios/{id}

Descrição: Deleta um usuário.

Autenticação: Requer autenticação Bearer Token.

Parâmetros:

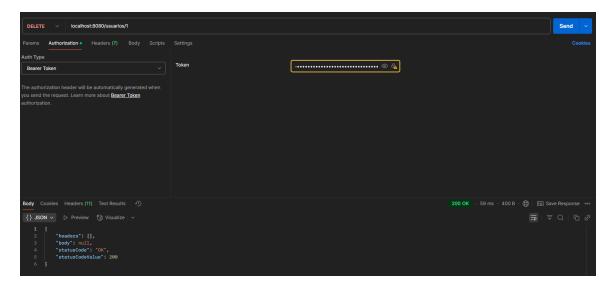
- id - ID do usuário

Respostas:

• 200 OK:

Usuário deletado com sucesso

• 403 Forbidden:



Métodos de categoria

POST /categorias

Descrição: Cria nova categoria.

Autenticação: Requer autenticação Bearer Token.

```
Exemplo de Body:

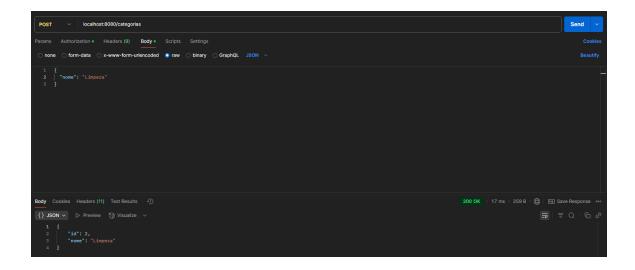
{
    "nome": "Limpeza"
}

Respostas:

• 200 OK:

{
    "id": 1,
    "nome": "Limpeza"
}
```

• 403 Forbidden:



GET /categorias

Descrição: Lista todas as categorias.

Autenticação: Requer autenticação Bearer Token.

Respostas:

• 200 OK:

[{ 'id': 1, 'nome': 'Limpeza' }]

• 403 Forbidden:

```
Send ▼
Parama Authorization * Headers (7) Body Scripts Settings Cookies

* none Offern-data Oxwww-form-urlencoded Oraw Obinary OfrephOL

**Body Cookies Headers (11) Test Results ①

**Defend Oxwww-form-urlencoded Oraw Obinary OfrephOL

**This request does not have a body

**Defend Oxwww-form-urlencoded Oraw Oxwww-form-urlencoded Oraw Oxwww-form-urlencoded Oxwwo-form-urlencoded Oxwwo-form-urlencoded Oxww-form-urlencoded Oxww-f
```

PUT /categorias/{id}

Descrição: Atualiza uma categoria.

Autenticação: Requer autenticação Bearer Token.

Parâmetros:

• - id - ID da categoria

```
Exemplo de Body:
```

```
{
"nome": "Limpeza atualizada"
}
```

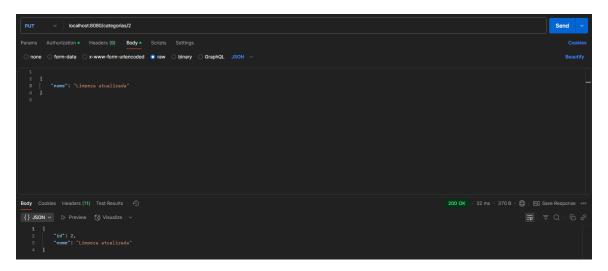
Respostas:

• 200 OK:

{ 'id': 1, 'nome': 'Limpeza atualizada' }

• 403 Forbidden:

Acesso negado



DELETE /categorias/{id}

Descrição: Deleta uma categoria.

Autenticação: Requer autenticação Bearer Token.

Parâmetros:

- id - ID da categoria

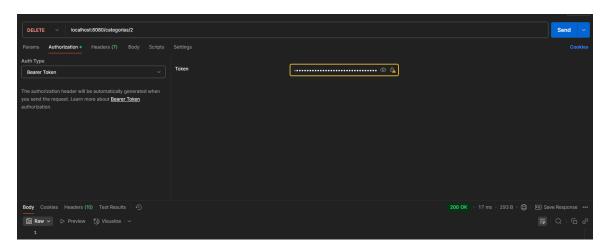
Respostas:

200 OK:

Categoria deletada com sucesso

403 Forbidden:

Acesso negado



Métodos de serviço

POST /servicos

Descrição: Cria um novo serviço.

Autenticação: Requer autenticação Bearer Token.

Exemplo de Body:

{
 "name": "Limpeza de telha",
 "categoria": {
 "id": 2

```
}
}
Respostas:
• 200 OK:
{
    "id": 1,
    "nome": null,
    "categoria": {
        "id": 2,
        "nome": null
}
}
```

• 403 Forbidden:

GET /servicos

Descrição: Lista todos os serviços.

Autenticação: Requer autenticação Bearer Token.

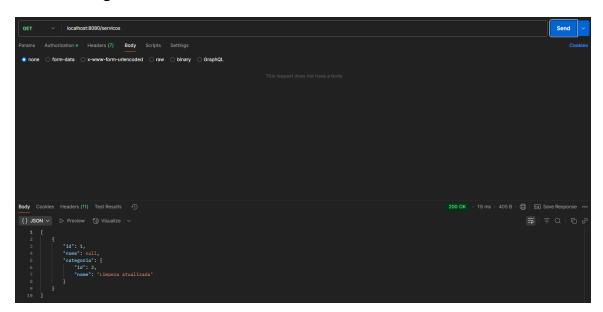
Respostas:

• 200 OK:

[...]

• 403 Forbidden:

Acesso negado



PUT /servicos/{id}

Descrição: Atualiza um serviço.

Autenticação: Requer autenticação Bearer Token.

Parâmetros:

- id - ID do serviço

Exemplo de Body:

{

"nome": "Limpeza de chão",

```
"categoria": {
    "id": 2
}

Respostas:

200 OK:
{
    "id": 1,
    "nome": "Limpeza de chão",
    "categoria": {
        "id": 2,
        "nome": "Limpeza atualizada"
    }
}
```

• 403 Forbidden:

DELETE /servicos/{id}

Descrição: Deleta um serviço.

Autenticação: Requer autenticação Bearer Token.

Parâmetros:

• - id - ID do serviço

Respostas:

• 200 OK:

Serviço deletado com sucesso

• 403 Forbidden:

