


Atividade 08 | Implementação de Middleware, Interceptadores e Filtros


▼ Fluxograma do Fluxo HTTP com IA

flowchart TD


A[Cliente Envia Requisição] → B{Middleware de Autenticação}


B → C{Authorization
header existe?}

C --  Não → D[Retorna 401
Token não fornecido]

C --  Sim → E[Controller Trata Rota
ex: /users]

E → F{Operação
Bem-sucedida?}

F --  Não - Lança Exceção → G[Filtro de Exceção]

F --  Sim → H[Interceptor de Resposta]

G → I[Retorna JSON de Erro
Personalizado]

H → J[Envolve Resposta em
 success: true, data: ...]

I & J → K[Cliente Recebe Resposta]

▼ Reflexão com IA

Nesta etapa, a proposta é incentivar o **uso crítico da Inteligência Artificial** para aprofundar sua compreensão técnica e levantar novas dúvidas.

Abaixo estão **10 perguntas reflexivas** relacionadas aos temas abordados na atividade:

1. Você acha que apenas verificar se o cabeçalho `Authorization` existe é suficiente para proteger uma API? Em que casos isso pode falhar? Só verificar existência do header Authorization não é suficiente. Falha se o token for inválido/expirado, for reutilizado (replay), for apenas um valor estático ou se não houver verificação de assinatura, scopes e revogação. É apenas um primeiro check; é preciso validar/decodificar o token (JWT/OAuth), checar exp/iss/aud, e possivelmente consultar blacklist/refresh tokens.

2. Em uma API real, todas as rotas devem exigir autenticação? Como lidar com rotas públicas?

Nem todas as rotas devem exigir autenticação. Rotas públicas comuns: health, login/register, docs, arquivos públicos. Use guards/whitelist para proteger rotas, e aplique autenticação apenas onde há necessidade (principle of least privilege). Documente claramente quais rotas são públicas.

3. Qual é a real vantagem de usar um interceptor para padronizar as respostas? Isso pode causar algum problema?

Vantagens do interceptor para padronizar respostas: contrato uniforme para o cliente, menos repetição no controller, facilita handling no frontend e logging. Problemas potenciais: pode esconder códigos HTTP reais se mal implementado, interferir em middlewares de cache, ou duplicar formatos já esperados por consumidores.

4. Por que é importante personalizar mensagens de erro com filtros? Em que situação isso se torna essencial?

Filtrar e personalizar erros é importante para segurança (não vazar stack traces), UX consistente e observabilidade (logs/monitoring).

Essencial em APIs públicas, integrações B2B e quando políticas de compliance exigem mensagens padronizadas.

5. O que muda na ordem do fluxo se o erro for lançado diretamente do middleware em vez do controller?

Se o erro for lançado no middleware, o fluxo é interrompido antes do controller; dependendo de como o middleware responde, os Exception Filters do Nest podem não ser acionados (se o middleware já respondeu). Portanto, a ordem altera quem trata/format a exceção e qual stack/contexto está disponível.

6. Se você quisesse adicionar cache de resposta (como Redis), onde no fluxo da requisição isso entraria?

Cache (ex.: Redis) costuma entrar após autenticação e antes do processamento pesado — geralmente como um interceptor (response cache) ou via camada de serviço. Interceptor permite retornar cached response sem chamar controller/service; middleware é cedo demais (antes do roteamento).

7. Você usaria interceptadores também para medir tempo de resposta ou registrar logs? Como?

Sim — interceptors são apropriados para medir tempo de resposta ou logs: capture timestamp antes e depois de next.handle(), calcule duração e logue. Use com cuidado para não bloquear I/O e para agregar/tracer IDs para correlação.

8. O que poderia dar errado ao usar muitos middlewares em sequência?

Problemas com muitos middlewares: ordem e dependências se dificultam, impacto de performance, maior latência, risco de side-effects e dificuldade de depuração. Mantenha middlewares pequenos e bem documentados.

9. Se sua API fosse pública e usada por desenvolvedores terceiros, quais cuidados teria com os erros retornados?

Para API pública: retornar erros consistentes, códigos HTTP corretos, mensagens úteis, mas não sensíveis, documentar erros (codes/messages), versionar, e oferecer suporte para troubleshooting (correlation id). Evitar expor stack traces e dados sensíveis.

10. Se você precisasse registrar tudo que acontece em uma requisição — do início ao fim — onde colocaria esses logs? Usaria middleware, interceptor, ambos ou outro recurso? Por quê?

Para registrar todo o ciclo da requisição: combine middleware + interceptor + filtros:

- Middleware: iniciar traceld, coletar headers e início do request.
- Interceptor: medir tempo, registrar resposta, enriquecer logs com traceld.
- Exception Filter: logar erros formatados.

Use OpenTelemetry / centralized logging (winston/elastic/tempo) para correlação. Middleware inicia o contexto; interceptor e filter completam.

▼ Prints

The screenshot shows a VS Code editor with three panels. The top panel is a REST client with a test named 'Testes para Atividade 08 - Middleware, Interceptor'. It contains three test cases: 1. A GET request to 'http://localhost:3000/users' with no headers, expecting a 401 status and a message 'Token não fornecido'. 2. A GET request with a 'Bearer fake-token' header, expecting a 200 status and a user object. 3. A GET request with a 'Basic dGVzZG90dG8vdG9uYm91' header, expecting a 200 status. The middle panel shows the response for the first test, which is a 401 Unauthorized status with a JSON body: {'message': 'Token não fornecido'}. The bottom panel is a terminal showing the output of 'npm run start', displaying logs from NestJS, including the initialization of the UsersModule and the mapping of routes for the UsersController.

```
test-apl.http x  users.controllers  users.modules  ...  Response(24ms) x
atividade-08 x  test-apl.http x  GET /users
You, há 6 horas (1 autor (você))
# Testes para Atividade 08 - Middleware, Interceptor
# Filtro de Exceção
# Observação: rode o servidor antes: npm run start:dev
Send Request
1 http://localhost:3000/users
# Testes para Atividade 08 - Middleware, Interceptor
# Filtro de Exceção
# Observação: rode o servidor antes: npm run start:dev
### TESTE 1 - MIDDLEWARE: Requisição SEM cabeçalho Authorization (esperado 401)
Send Request
2 GET http://localhost:3000/users
# Esperado (HTTP 401):
3 {
4   "message": "Token não fornecido"
5 }
### TESTE 2 - MIDDLEWARE: Requisição COM Bearer Token (esperado 200)
Send Request
6 GET http://localhost:3000/users
7 Authorization: Bearer fake-token
# Esperado (HTTP 200) - Interceptor deve envolver a resposta:
8 {
9   "success": true,
10  "data": { "id": 1, "name": "John Doe" } }
### TESTE 3 - INTERCEPTOR: Requisição COM Basic Auth (esperado 200)
Send Request
11 GET http://localhost:3000/users
12 Authorization: Basic dGVzZG90dG8vdG9uYm91
PROBLEMAS SAÍDA SONARQUBE TERMINAL SPELL CHECKER
+ 2: npm (atividade-08)
Klsio27@pop-os: /media/Klsio27/outher-files/documentos/utfpr/topicos-especiais/atividade-08$ npm run start
[Nest] 23935 - 05/10/2025, 12:49:17 LOG [InstanceLoader] UsersModule dependencies initialized +0ms
[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RoutesResolver] ApplicationController {/}: +5ms
[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RouterExplorer] Mapped {/, GET} route +3ms
[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RoutesResolver] UsersController {/users}: +0ms
[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RouterExplorer] Mapped {/users, GET} route +1ms
[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RouterExplorer] Mapped {/users/:id, GET} route +1ms
[Nest] 23935 - 05/10/2025, 12:49:17 LOG [NestApplication] Nest application successfully started +2ms
```

test-api.http x users.controllers users.modules ... Response(12ms) x

atividade-08 > test-api.http > GET /users

```
1 # Testes para Atividade 08 - Middleware, Interceptor
2 # Filtro de Exceção
3 # Observação: rode o servidor antes: npm run start:dev
4 Send Request
5 http
6 # Testes para Atividade 08 - Middleware, Interceptor
7 # Filtro de Exceção
8 # Observação: rode o servidor antes: npm run start:dev
9
10 ### TESTE 1 - MIDDLEWARE: Requisição SEM cabeçalho
11 # Autorização (esperado 401)
12 Send Request
13 GET http://localhost:3000/users
14
15 # Esperado (HTTP 401):
16 # {
17 #   "message": "Token não fornecido"
18 # }
19
20 ### TESTE 2 - MIDDLEWARE: Requisição COM Bearer
21 # Token (esperado 200)
22 Send Request
23 GET http://localhost:3000/users
24 Authorization: Bearer fake-token
25
26 # Esperado (HTTP 200) - Interceptor deve envolver a
27 # resposta:
28 # {
29 #   "success": true,
30 #   "data": [ { "id": 1, "name": "John Doe" } ]
31 # }
32
33 ### TESTE 3 - INTERCEPTOR: Requisição COM Basic Auth
34 # (esperado 200)
35 Send Request
36 GET http://localhost:3000/users
37 Authorization: Basic dGVzZGU6dGVzZGUxMjM=
```

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 52
5 ETag: W/"34-x0rbtdvWsfjRd+stuwFeyl4raA"
6 Date: Sun, 05 Oct 2025 22:09:00 GMT
7 Connection: close
8
9 {
10   "success": true,
11   "data": [
12     {
13       "id": 1,
14       "name": "John Doe"
15     }
16   ]
17 }
```

PROBLEMAS SAÍDA SONARQUE TERMINAL SPELL CHECKER

+ 2 npm (atividade-08) x

ksio27@pop-os: /media/ksio27/outher-files/documentos/utfpr/topicos-especiais/atividade-08\$ npm run start

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [InstanceLoader] UsersModule dependencies initialized +0ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RoutesResolver] AppController {/}: +5ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RouterExplorer] Mapped {/, GET} route +3ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RoutesResolver] UsersController {/users}: +0ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RouterExplorer] Mapped {/users, GET} route +1ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RoutesResolver] Mapped {/users/:id, GET} route +1ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [NestApplication] Nest application successfully started +2ms

CHAT GEMINI CODE ASSIST EXTENSÕES

Tarefas pendentes (2/3) - ✓ Escrever instruções passo a passo para Insomnia/Postman

Responda das esses quetões de forma organizada>>> ### 8. Reflexão com IA

Nesta etapa, a proposta é incentivar o uso crítico da Inteligência Artificial para aprofundar sua compreensão técnica e levantar novas dúvidas.

Abaixo estão 10 perguntas reflexivas relacionadas aos temas abordados na atividade:

1. Você acha que apenas verificar se o cabeçalho Authorization existe é suficiente para proteger uma API? Em que casos isso pode falhar?

2. Em uma API real, todas as rotas devem exigir autenticação? Como lidar com rotas públicas?

3. Qual é a real vantagem de usar um interceptor para padronizar as respostas? Isso pode causar algum problema?

4. Por que é importante personalizar mensagens de erro com filtros? Em que situação isso se torna essencial?

5. O que muda na ordem do fluxo se o erro for lançado diretamente do middleware em vez do controller?

6. Se você quisesse adicionar cache de resposta (como Redis), onde no fluxo da requisição isso entraria?

7. Você usaria interceptadores também para medir tempo de resposta ou registrar logs? Como?

8. O que poderia dar errado ao usar muitos middlewares em sequência?

9. Se sua API fosse pública e usada por desenvolvedores terceiros, quais cuidados teria com os erros retornados?

10. Se você precisasse registrar tudo que acontece em uma requisição - do início ao fim - onde colocaria esses logs? Usaria middleware, interceptor, ambos ou outro recurso? Por quê?

Utilize uma ferramenta de IA (como ChatGPT ou similar) para explorar pelo menos uma destas perguntas:

Escolha uma pergunta da lista de 10 perguntas logo acima e pergunte à IA.

Dê um print da resposta que recebeu.

Se a resposta da IA despertar uma nova dúvida ou outra pergunta, ótimo!

→ Faça a nova pergunta à IA e também dê um print da nova resposta.

Adicionar contexto... test-api.http x

Adicionar contexto (a), extensões (@), comandos (!)

Ask GPT-5 mini

test-api.http x users.controllers users.modules ... Response(8ms) x

atividade-08 > test-api.http > GET /users

```
15 ### TESTE 2 - MIDDLEWARE: Requisição COM Bearer
16 # Token (esperado 200)
17 Send Request
18 GET http://localhost:3000/users
19 Authorization: Bearer fake-token
20
21 # Esperado (HTTP 200) - Interceptor deve envolver a
22 # resposta:
23 # {
24 #   "success": true,
25 #   "data": [ { "id": 1, "name": "John Doe" } ]
26 # }
27
28 ### TESTE 3 - INTERCEPTOR: Requisição COM Basic Auth
29 # (esperado 200)
30 Send Request
31 GET http://localhost:3000/users
32 Authorization: Basic dGVzZGU6dGVzZGUxMjM=
33 # Nota: dGVzZGU6dGVzZGUxMjM= é "teste:teste123" em
34 # Base64
35
36 # Esperado (HTTP 200):
37 # {
38 #   "success": true,
39 #   "data": [ { "id": 1, "name": "John Doe" } ]
40 # }
41
42 ### TESTE 4 - FILTRO: Usuário inexistente (esperado
43 # 404)
44 Send Request
45 GET http://localhost:3000/users/999
46 Authorization: Bearer fake-token
47
48 # Esperado (HTTP 404):
49 # {
50 #   "statusCode": 404,
51 #   "message": "Usuário não encontrado",
52 #   "timestamp": "2024-11-08T12:00:00.000Z"
53 # }
```

```
1 HTTP/1.1 404 Not Found
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 96
5 ETag: W/"5e-hMBtV5W01a3/Xnohbs5Hhv2eUY"
6 Date: Sun, 05 Oct 2025 22:09:22 GMT
7 Connection: close
8
9 {
10   "statusCode": 404,
11   "message": "Usuário não encontrado",
12   "timestamp": "2025-10-05T22:09:22.773Z"
13 }
```

PROBLEMAS SAÍDA SONARQUE TERMINAL SPELL CHECKER

+ 2 npm (atividade-08) x

ksio27@pop-os: /media/ksio27/outher-files/documentos/utfpr/topicos-especiais/atividade-08\$ npm run start

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [InstanceLoader] UsersModule dependencies initialized +0ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RoutesResolver] AppController {/}: +5ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RouterExplorer] Mapped {/, GET} route +3ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RoutesResolver] UsersController {/users}: +0ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RouterExplorer] Mapped {/users, GET} route +1ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [RoutesResolver] Mapped {/users/:id, GET} route +1ms

[Nest] 23935 - 05/10/2025, 12:49:17 LOG [NestApplication] Nest application successfully started +2ms

CHAT GEMINI CODE ASSIST EXTENSÕES

Tarefas pendentes (2/3) - ✓ Escrever instruções passo a passo para Insomnia/Postman

Responda das esses quetões de forma organizada>>> ### 8. Reflexão com IA

Nesta etapa, a proposta é incentivar o uso crítico da Inteligência Artificial para aprofundar sua compreensão técnica e levantar novas dúvidas.

Abaixo estão 10 perguntas reflexivas relacionadas aos temas abordados na atividade:

1. Você acha que apenas verificar se o cabeçalho Authorization existe é suficiente para proteger uma API? Em que casos isso pode falhar?

2. Em uma API real, todas as rotas devem exigir autenticação? Como lidar com rotas públicas?

3. Qual é a real vantagem de usar um interceptor para padronizar as respostas? Isso pode causar algum problema?

4. Por que é importante personalizar mensagens de erro com filtros? Em que situação isso se torna essencial?

5. O que muda na ordem do fluxo se o erro for lançado diretamente do middleware em vez do controller?

6. Se você quisesse adicionar cache de resposta (como Redis), onde no fluxo da requisição isso entraria?

7. Você usaria interceptadores também para medir tempo de resposta ou registrar logs? Como?

8. O que poderia dar errado ao usar muitos middlewares em sequência?

9. Se sua API fosse pública e usada por desenvolvedores terceiros, quais cuidados teria com os erros retornados?

10. Se você precisasse registrar tudo que acontece em uma requisição - do início ao fim - onde colocaria esses logs? Usaria middleware, interceptor, ambos ou outro recurso? Por quê?

Utilize uma ferramenta de IA (como ChatGPT ou similar) para explorar pelo menos uma destas perguntas:

Escolha uma pergunta da lista de 10 perguntas logo acima e pergunte à IA.

Dê um print da resposta que recebeu.

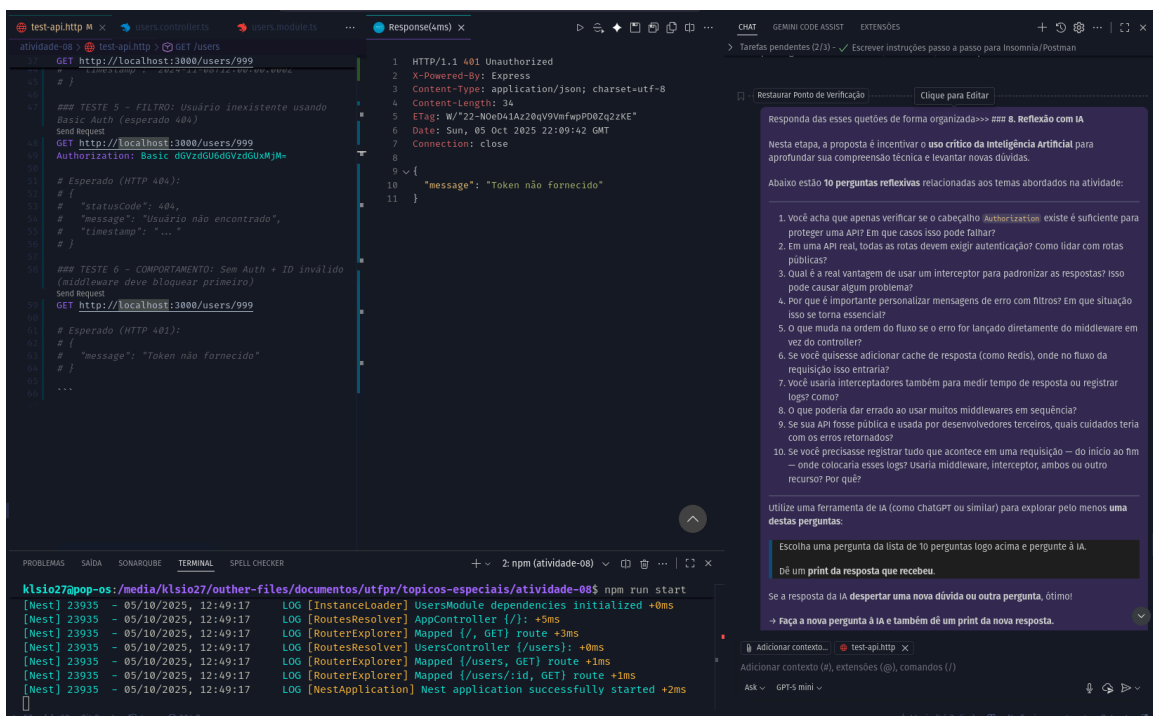
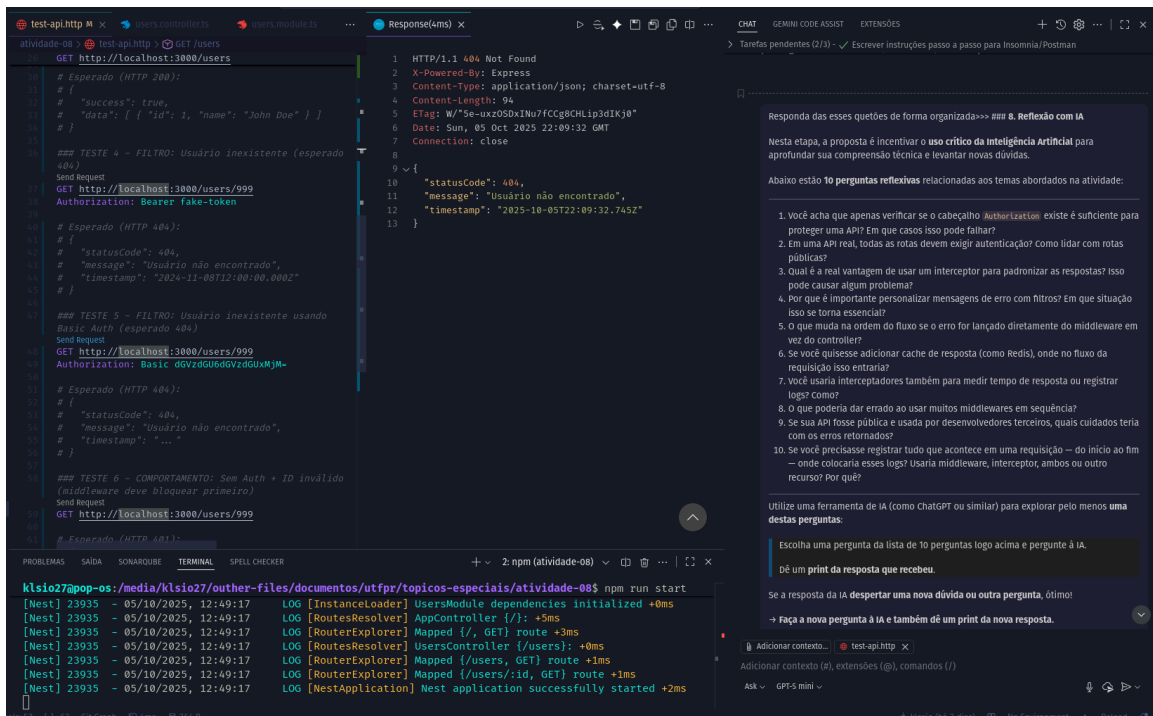
Se a resposta da IA despertar uma nova dúvida ou outra pergunta, ótimo!

→ Faça a nova pergunta à IA e também dê um print da nova resposta.

Adicionar contexto... test-api.http x

Adicionar contexto (a), extensões (@), comandos (!)

Ask GPT-5 mini



Testes para Atividade 08 - Middleware, Interceptor e Filtro de Exceção

O

Observação: rode o servidor antes: `npm run start:dev`

``http

Testes para Atividade 08 - Middleware, Interceptor e Filtro de Exceção

O

```
# Observação: rode o servidor antes: npm run start:dev

### TESTE 1 - MIDDLEWARE: Requisição SEM cabeçalho Authorization
(esperado 401)
GET http://localhost:3000/users

# Esperado (HTTP 401):
# {
#   "message": "Token não fornecido"
# }

### TESTE 2 - MIDDLEWARE: Requisição COM Bearer Token (esperado 200)
GET http://localhost:3000/users
Authorization: Bearer fake-token

# Esperado (HTTP 200) — Interceptor deve envolver a resposta:
# {
#   "success": true,
#   "data": [ { "id": 1, "name": "John Doe" } ]
# }

### TESTE 3 - INTERCEPTOR: Requisição COM Basic Auth (esperado 200)
GET http://localhost:3000/users
Authorization: Basic dGVzdGU6dGVzdGUxMjM=
# Nota: dGVzdGU6dGVzdGUxMjM= é "teste:teste123" em Base64

# Esperado (HTTP 200):
# {
#   "success": true,
#   "data": [ { "id": 1, "name": "John Doe" } ]
# }

### TESTE 4 - FILTRO: Usuário inexistente (esperado 404)
GET http://localhost:3000/users/999
Authorization: Bearer fake-token
```



```
# Esperado (HTTP 404):
# {
#   "statusCode": 404,
#   "message": "Usuário não encontrado",
#   "timestamp": "2024-11-08T12:00:00.000Z"
# }

### TESTE 5 - FILTRO: Usuário inexistente usando Basic Auth (esperado 404)
GET http://localhost:3000/users/999
Authorization: Basic dGVzdGU6dGVzdGUxMjM=

# Esperado (HTTP 404):
# {
#   "statusCode": 404,
#   "message": "Usuário não encontrado",
#   "timestamp": "..."
# }

### TESTE 6 - COMPORTAMENTO: Sem Auth + ID inválido (middleware deve bloquear primeiro)
GET http://localhost:3000/users/999

# Esperado (HTTP 401):
# {
#   "message": "Token não fornecido"
# }

...
```

▼ Repositório

<https://github.com/klsio22/topicos-especiais/tree/main/atividade-08>