



Guia Postman - EmployeeVirtual JWT API



TL;DR: Sim, Postman funciona perfeitamente!

Seu sistema suporta **ambas as formas**:

- ☒ **HttpOnly Cookies** (automático no Postman)
- ☒ **Bearer Token** (tradicional)



Método 1: HttpOnly Cookies (Recomendado)

Passo 1: Login

```
http
POST http://localhost:8000/api/auth/login
Content-Type: application/json

{
  "email": "test@example.com",
  "password": "senha123"
}
```

Passo 2: Postman salva cookies automaticamente! ✨

Resposta:

```
json
{
  "message": "Login realizado com sucesso",
  "user": {...},
  "token_type": "httponly_cookie"
}
```

Postman detecta `Set-Cookie: access_token=...` e salva automaticamente!

Passo 3: Requests seguintes funcionam sem configuração! 🎉

```
http
```

```
GET http://localhost:8000/api/agents/  
# ← Cookie enviado automaticamente!
```

```
GET http://localhost:8000/api/auth/me  
# ← Funciona sem fazer nada!
```

📌 Como ver os cookies no Postman:

1. Abra a aba "**Cookies**" embaixo da response
2. Ou vá em **Manage Cookies** no canto superior direito
3. Você verá: `access_token` e `refresh_token` salvos! 🍪

🔑 Método 2: Bearer Token (Tradicional)

Passo 1: Login sem cookies

```
http  
POST http://localhost:8000/api/auth/login?use_cookies=false  
Content-Type: application/json  
  
{  
  "email": "test@example.com",  
  "password": "senha123"  
}
```

Passo 2: Copiar token da resposta

Resposta:

```
json  
  
{  
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  
  "token_type": "bearer",  
  "expires_in": 900  
}
```

Passo 3: Configurar Authorization no Postman

1. Na aba "**Authorization**"
2. Type: "**Bearer Token**"

3. Token: Cole o `access_token` copiado

Passo 4: Requests protegidos

http

GET http://localhost:8000/api/agents/

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

Método 3: Collection Automatizada (Profissional)






Setup uma única vez:

1. **Importe a Collection** (JSON fornecido acima)
2. **Importe o Environment** (JSON fornecido acima)
3. **Configure variáveis:**
 - `baseUrl`: `http://localhost:8000`
 - `testUserEmail`: `test@example.com`
 - `testUserPassword`: `senha123`

Como usar:

1. **Execute "Register User"** (se necessário)
2. **Execute "Login (Bearer Token)"**
 - Token é **salvo automaticamente** na variável `accessToken`
3. **Todas as outras requests funcionam automaticamente!** ✨

Vantagens da Collection:

-  **Auto-salva tokens**
-  **Testes automáticos**
-  **Environment separado** (dev/prod)
-  **Scripts de pré/pós processamento**
-  **Reutilizável** pela equipe

Exemplos específicos:

Testar endpoint protegido:

http

1. Faça login (qualquer método acima)

POST /api/auth/login

2. Teste endpoint protegido (automático!)

GET /api/agents/

← Funciona automaticamente com cookies OU bearer token

Testar diferentes níveis de acesso:

http

Endpoint básico (qualquer usuário logado)

GET /api/agents/

Endpoint premium (só usuários premium)

GET /api/agents/premium-feature

← Retorna 402 se não for premium

Endpoint admin (só administradores)

DELETE /api/agents/admin/delete-agent/1

← Retorna 403 se não for admin

Endpoint opcional (funciona com ou sem login)

GET /api/agents/public-with-personalization

← Personaliza se logado, genérico se não logado

Testar refresh token:

http

Esperar 15+ minutos para access token expirar, então:

POST /api/auth/refresh

← Gera novo access token automaticamente

Testar logout:

http

POST /api/auth/logout

← Revoga tokens e limpa cookies

Tentar acessar endpoint protegido após logout:

GET /api/agents/

← Retorna 401 Unauthorized ☒

Debug e troubleshooting:

Ver informações do token:

http

GET /api/auth/token-info

← Mostra detalhes do token atual (para debug)

Verificar configurações de segurança:

http

GET /api/auth/security-info

← Mostra que proteções estão ativas

Health check:

http

GET /health

← Verifica se API está funcionando

Configurações no Postman:

Para Cookies (recomendado):

1. **Settings** → **General** → "Automatically follow redirects": ☒ ON
2. **Settings** → **General** → "Send cookies with requests": ☒ ON
3. **Settings** → **Cookies** → "Capture cookies": ☒ ON

Para Bearer Token:

1. Collection → **Authorization** → Type: "**Bearer Token**"

2. Token: `{{accessToken}}` (usando variável)

3. **Inherit auth from parent:** ☒ ON nos requests

Comparação dos métodos:

Aspecto	HttpOnly Cookies	Bearer Token	Collection
Segurança	✅ Máxima (XSS proof)	⚠️ Boa	✅ Máxima
Facilidade	✅ Automático	❌ Manual	✅ Automático
Flexibilidade	❌ Menos	✅ Mais	✅ Máxima
Produção	✅ Recomendado	⚠️ OK	✅ Ideal
Debug	❌ Mais difícil	✅ Fácil	✅ Muito fácil

Recomendação por caso:

Para desenvolvimento rápido:

```
http
# Use cookies - zero configuração!
POST /api/auth/login
GET /api/agents/ ← Funciona automaticamente
```

Para debug/troubleshooting:

```
http
# Use bearer token - mais visibilidade
POST /api/auth/login?use_cookies=false
# Copie token manualmente para ver o que está acontecendo
```

Para equipe/CI/CD:

```
# Use Collection - profissional e reutilizável
# Import, configure environment, done!
```

Pro Tips:

1. **Use o método que preferir** - ambos funcionam!
2. **Cookies são mais seguros** para produção

3. **Bearer token é melhor** para debug
4. **Collection é ideal** para equipes
5. **Teste sem login** endpoints opcionais
6. **Monitore expiração** do token (15 min)

Seu JWT está funcionando perfeitamente! 🎉