

Autenticação e autorização

← Índice da documentação da API

Tudo nesta página foi conferido em `config/SecurityConfig.java`, `config/JwtAuthenticationFilter.java`, `config/JwtSecurityConfig.java`, `config/LoginRateLimitingFilter.java`, `realtime/WebSocketConfig.java`, `mes/MesAccess.java`, `mes/MesEdgeSecurity.java`, `service/AuthenticationService.java` e nas anotações `@PreAuthorize` de cada controller.

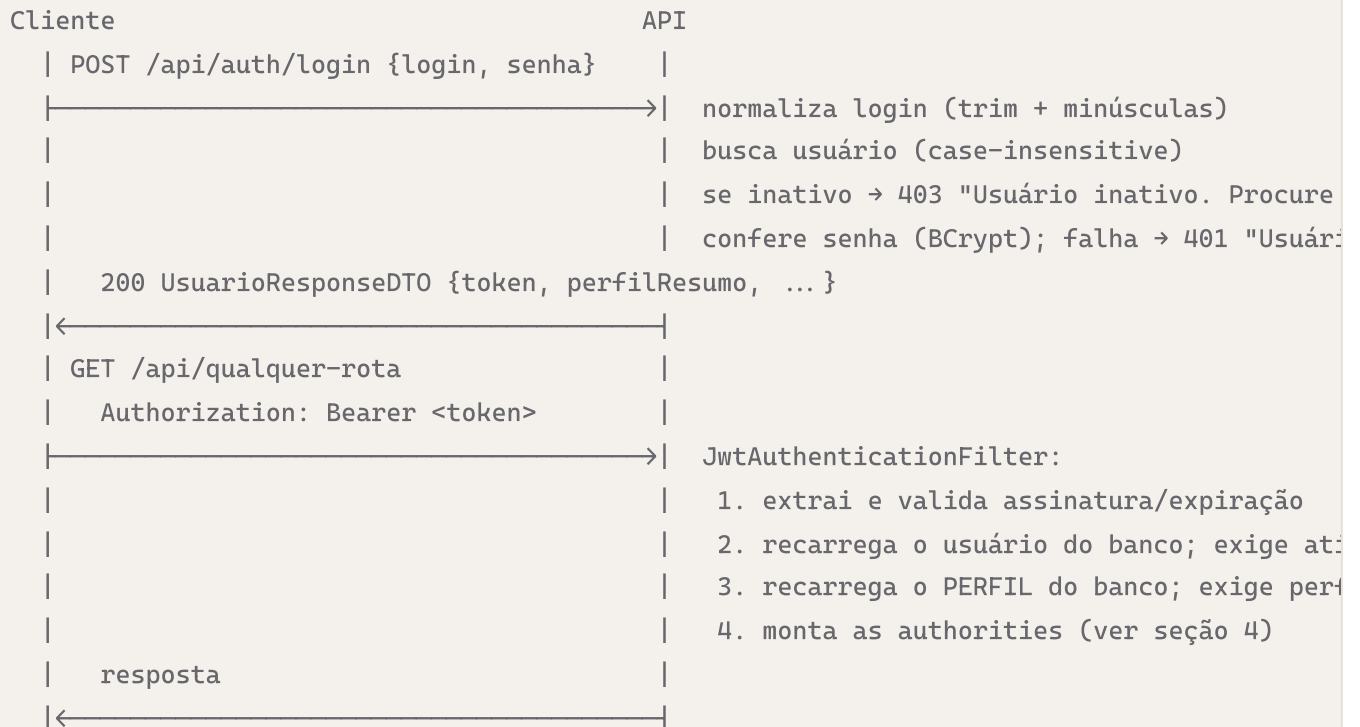
1. Visão geral

A API é **stateless**: não há sessão de servidor (`SessionCreationPolicy.STATELESS`) e o CSRF está desligado, coerente com um cliente que envia o token em toda chamada. Existem **duas cadeias de segurança independentes**:

CADEIA	ALCANCE	COMO AUTENTICA
Principal (<code>SecurityConfig</code>)	toda a API, exceto <code>/api/mes/edge/**</code>	JWT no cabeçalho <code>Authorization: Bearer <token></code>
Edge do MES (<code>MesEdgeSecurity</code> , <code>@Order(1)</code>)	<code>/api/mes/edge/**</code>	chave compartilhada no cabeçalho <code>X-Edge-Key</code>

São cadeias separadas de propósito: o coletor de chão de fábrica não é uma pessoa e não faz login.

2. Fluxo de autenticação (usuários)



Pontos confirmados no código:

- **Rotas de login:** `POST /api/auth/login` e `POST /api/usuarios/login` (mesma operação, dois caminhos).
- **Corpo do login:** `{"login": " ... ", "senha": " ... "}` (`LoginDTO` , validado com `@Valid`).
- **Sessão atual:** `GET /api/auth/me` devolve o usuário autenticado **sem** token novo, com `perfilResumo` .
- **Token:** JWT assinado com HS256. O `subject` é o login do usuário. **Não há claims de perfil nem de permissões no token.** Validade padrão: `28800000` ms (8 horas), configurável pela variável `SECURITY_JWT_EXPIRATION_MS` .
- **Segredo:** `SECURITY_JWT_SECRET` (ou `JWT_SECRET`); o serviço **não sobe** sem ele nem com menos de 32 caracteres. A chave efetiva é o SHA-256 do segredo informado.
- **Não existe** refresh token, logout no servidor, revogação de token nem troca de senha obrigatória. "Sair" é apenas descartar o token no cliente (ver documento interno de lacunas).
- **Efeito imediato de mudanças:** como usuário e perfil são relidos do banco a cada requisição, inativar um usuário, inativar um perfil ou mudar as telas de um perfil vale **na próxima chamada**, sem invalidar tokens.

- **Perfil ausente, inativo ou desconhecido gera zero authorities:** o usuário continua autenticado, mas toda rota que exige uma permissão específica responde `403`.

3. Rotas públicas (sem token)

Declaradas em `SecurityConfig`:

ROTA	OBSERVAÇÃO
<code>OPTIONS /**</code>	pré-voo de CORS
<code>POST /api/auth/login</code> , <code>POST /api/usuarios/login</code>	login (limitado a 10 tentativas por minuto por IP)
<code>POST /api/usuarios</code>	autocadastro (limitado a 5 por minuto por IP) — ver seção 6
<code>/swagger-ui.html</code> , <code>/swagger-ui/**</code> , <code>/v3/api-docs/**</code> , <code>/api-docs/**</code>	só respondem se <code>SWAGGER_ENABLED=true</code> (padrão <code>false</code>)
<code>/actuator/**</code> , <code>/error</code>	liberados na cadeia; o que o Actuator expõe depende da configuração de deploy (não verificada)
<code>/ws/**</code> , <code>/ws-native/**</code>	<i>handshake</i> de WebSocket liberado; a autenticação é exigida no comando STOMP <code>CONNECT</code> (seção 7)

Todo o resto cai em `anyRequest().authenticated()` (sem token → `401`).

4. Autorização: perfil, telas e tipo de acesso

O modelo é o de **perfil** (`perfil`): cada usuário tem um código de perfil (campo `perfil` do usuário, texto, **não é chave estrangeira**) que aponta para um registro com:

- `tipoAcesso`: `FULL_ACCESS` (pode alterar) ou `READ_ONLY` (só consulta);
- `telasPermitidas`: lista de "telas" (ex.: `WAREHOUSE`, `PRODUCTS`, `MES_PCP`);
- `ativo`.

O filtro JWT transforma isso em duas famílias de *authorities*:

AUTHORITY	ORIGEM
<code>ROLE_FULL_ACCESS</code> ou <code>ROLE_READ_ONLY</code>	<code>tipoAcesso</code> do perfil
<code>SCREEN_<TELA></code>	uma por item de <code>telasPermitidas</code> (maiúsculas)

As anotações `@PreAuthorize` combinam os dois eixos. O padrão que se repete é:

- **leitura:** exige só `SCREEN_<tela>`;
- **criar / alterar / excluir:** exige `SCREEN_<tela>` e `hasRole('FULL_ACCESS')`.

Exemplo real (`AreaEstoqueController`): `GET /api/areas` aceita `SCREEN_DASHBOARD` ou `SCREEN_WAREHOUSE`; `POST /api/areas` e `DELETE /api/areas/{id}` exigem `SCREEN_WAREHOUSE` e `FULL_ACCESS`.

4.1 Telas (**SCREEN_***) e o que liberam

AUTHORITY	TELA / USO NO FRONTEND	O QUE O BACKEND EXIGE
SCREEN_DASHBOARD D	Dashboard	leitura de setores (/api/areas) e do estoque consolidado (/api/estoque/itens , /api/estoque/itens-consolidado), sempre "ou" SCREEN_WAREHOUSE
SCREEN_WAREHOUSE E	Armazém, Setores, estruturas, itens de estoque, listas (encerto)	leitura com a tela; criar/alterar/excluir com a tela e FULL_ACCESS
SCREEN_PRODUCT S	Produtos e categorias; Moldes	leitura com a tela; escrita com a tela e FULL_ACCESS
SCREEN_MOLDS	Moldes	aceita-se SCREEN_MOLDS ou SCREEN_PRODUCTS nas rotas de moldes
SCREEN_USERS	Usuários	listar/consultar com a tela; alterar com a tela e FULL_ACCESS
SCREEN_PROFILE S	Perfis	criar/alterar/ativar/inativar com a tela e FULL_ACCESS ; listar também aceita SCREEN_USERS
SCREEN_HISTORY	Histórico	consulta; o histórico por entidade também aceita SCREEN_USERS , SCREEN_PROFILES e SCREEN_PRODUCTS
SCREEN_REPORTS	Relatórios de estoque (tela desativada no menu atual)	consulta
SCREEN_ALERTS	Alertas (tela desativada no menu atual)	consulta
SCREEN_MOVEMENT S	Movimentações (tela desativada no menu atual)	consulta; registrar exige também FULL_ACCESS

Cada ficha de endpoint em modulos/ traz a expressão exata. A lista dos perfis iniciais está na seção 5.

4.2 MES / PCP: outro esquema

`MesController` não usa `@PreAuthorize("SCREEN_...")` direto. Usa o bean `mesAccess` (`mes/MesAccess.java`), que olha quatro authorities de tela e o papel `FULL_ACCESS`:

VERIFICAÇÃO	REGRA NO CÓDIGO
<code>read</code>	qualquer uma entre <code>SCREEN_MES_ADMIN</code> , <code>SCREEN_MES_OPERATOR</code> , <code>SCREEN_MES_MECHANIC</code> , <code>SCREEN_MES_PCP</code>
<code>admin</code>	<code>FULL_ACCESS</code> e <code>SCREEN_MES_ADMIN</code>
<code>operate</code>	<code>admin</code> ou (<code>FULL_ACCESS</code> e <code>SCREEN_MES_OPERATOR</code>)
<code>maintain</code>	<code>admin</code> ou (<code>FULL_ACCESS</code> e <code>SCREEN_MES_MECHANIC</code>)
<code>plan</code>	<code>admin</code> ou (<code>FULL_ACCESS</code> e <code>SCREEN_MES_PCP</code>)

Consequência importante: um perfil `READ_ONLY` com `MES_PCP` só consulta — não planeja. As fichas do módulo `producao.md` indicam qual verificação cada rota usa; `GET /api/mes/permissions` devolve `{read, operate, maintain, plan, admin}` do usuário logado, e é o que o frontend usa para mostrar/ocultar ações.

4.3 Autenticação do Edge (coletor industrial)

`MesEdgeSecurity` cria uma cadeia própria para `/api/mes/edge/**` que compara o cabeçalho `X-Edge-Key` com a variável de ambiente `MES_EDGE_KEY` (comparação em tempo constante, `MessageDigest.isEqual`):

SITUAÇÃO	RESPOSTA
<code>MES_EDGE_KEY</code> ausente ou com menos de 16 bytes	503
cabeçalho ausente ou diferente da chave configurada	401
chave correta	segue para o controller (principal <code>mes-edge</code> , authority <code>MES_EDGE</code>)

O corpo é `{"message":"Integracao edge indisponivel ou credencial invalida"}` nos dois casos — deliberadamente igual, para não revelar qual deles ocorreu. JWT de usuário não substitui a

chave, e a chave não abre nenhuma rota fora de `/api/mes/edge/**`. A chave é **global** (não por máquina).

5. Perfis iniciais (migração `V2__insert_default_data.sql`)

CÓDIGO	TIPO	TELAS
ADMINISTRADOR	FULL_ACCESS	DASHBOARD, ALERTS, MOVEMENTS, REPORTS, WAREHOUSE, PRODUCTS, USERS, HISTORY, PROFILES
OPERADOR	FULL_ACCESS	DASHBOARD, ALERTS, MOVEMENTS, REPORTS, WAREHOUSE, PRODUCTS, HISTORY
CONSULTOR	READ_ONLY	DASHBOARD, ALERTS, REPORTS, WAREHOUSE, PRODUCTS, HISTORY

Os perfis podem ser editados depois pela API/tela de Perfis (`/api/perfis`); o banco de produção pode estar diferente da semente. Essa semente não contém `MOLDS` nem as telas `MES_*`; elas passam a existir quando um perfil é criado ou editado com elas. A tela `MOLDS` foi criada depois: o frontend libera Moldes por `MOLDS` ou `PRODUCTS`, e o backend também aceita `SCREEN_PRODUCTS` ou `SCREEN_MOLDS` nas rotas de moldes.

6. Autocadastro: `POST /api/usuarios`

A rota é pública (usada pela tela "Criar conta"), mas o campo `perfil` do corpo só é respeitado quando quem chama é um usuário autenticado com `SCREEN_USERS` e `FULL_ACCESS` (`chamadorPodeEscolherPerfil`). Nos demais casos o perfil é forçado para `CONSULTOR` (`PERFIL_PADRAO_AUTOCADASTRO`), qualquer que seja o valor enviado. Por isso a pergunta "um anônimo consegue se criar como administrador?" tem resposta **não** no código atual.

7. WebSocket / STOMP

Endpoints `/ws` (com SockJS) e `/ws-native` (WebSocket puro). O *handshake* é sempre aceito, mas o comando STOMP `CONNECT` exige JWT válido (cabeçalho `Authorization` / `authorization` / `X-Authorization` / `token` do frame STOMP, ou parâmetro `?token=` no handshake); sem isso a conexão é rejeitada com `SecurityException`. Usuário inativo ou token inválido também são rejeitados. Detalhes dos tópicos em `integracoes.md`.

8. CORS

`CorsConfig` aceita apenas origens de navegador autorizadas (o frontend publicado, ambientes locais de desenvolvimento e hosts internos da Wester).

Métodos permitidos: `GET, POST, PUT, PATCH, DELETE, OPTIONS`; cabeçalhos permitidos **apenas** `Authorization` e `Content-Type`; credenciais habilitadas; cache de pré-voo de 3600 s. Um cliente de servidor (cURL, TUTOM, coletor) não é afetado por CORS; um navegador em outra origem só consegue chamar a API se a origem estiver nessa lista.

`X-Edge-Key` não está na lista de cabeçalhos permitidos: o coletor deve chamar de servidor/dispositivo, não de navegador.

9. Limites de taxa

`LoginRateLimitingFilter` (em memória, por IP, janela deslizante):

ROTA	LIMITE
<code>POST</code> <code>/api/auth/login</code>	10 por minuto
<code>POST</code> <code>/api/usuarios/login</code>	10 por minuto
<code>POST</code> <code>/api/usuarios</code>	5 por minuto

Excedido: `429` com `{"status":429,"error":"Too Many Requests","message":"Muitas tentativas. Aguarde um momento antes de tentar novamente."}`. O IP vem do primeiro valor de `X-Forwarded-For` quando existe, senão do endereço remoto. O contador **não sobrevive a reinício** e **não é compartilhado entre instâncias**.

10. Exemplos

```
bash
# 1) login - devolve o token em .token
curl -s -X POST "${BASE_URL}/api/auth/login" \
  -H "Content-Type: application/json" \
  -d '{"login":"usuario@exemplo.com","senha":"<SENHA>"}'

# 2) chamada autenticada
curl -s "${BASE_URL}/api/auth/me" -H "Authorization: Bearer ${TOKEN}"

# 3) chamada do coletor (não usa JWT)
curl -s "${BASE_URL}/api/mes/edge/config?machineId=<TAG_DA_INJETORA>" -H "X-Edge-Key: ${E
```

Resposta típica de login (estrutura de `UsuarioResponseDTO`; valores ilustrativos):

```
json
{
  "id": 12,
  "login": "usuario@exemplo.com",
  "nome": "Usuário Exemplo",
  "perfil": "OPERADOR",
  "ativo": true,
  "token": "<JWT>",
  "perfilResumo": {
    "id": 2,
    "code": "OPERADOR",
    "description": "Operador",
    "type": "FULL_ACCESS",
    "allowedScreens": ["DASHBOARD", "ALERTS", "MOVEMENTS", "REPORTS", "WAREHOUSE", "PRODU
    "active": true
  }
}
```

Exemplos acima não carregam credenciais nem endereços reais de produção.