SEGURANÇA E AUTENTICAÇÃO

A segurança no backend envolve Spring Security e Token JWT. São gerados no login o access_token e o refresh_token, onde, o access_token expira a cada 5 minutos e um novo é gerado, desde que o refresh_token seja enviado via cookie httponly para o endpoint que gera o novo access_token. O refresh_token dura 1 hora.

🔓 Os Tokens

O access_token é armazenado em um cookie httponly e tem tempo de expiração de 5 minutos e seu payload contém as seguintes claims adicionais:

- **username:** como subject
- **roles:** papéis do usuário embutidos no token como array de strings

O refresh_token é armazenado em um cookie httponly e tem tempo de expiração de 30 minutos e seu payload contem as seguintes claims adicionais:

- **username**: como subject

🛡️ O Filtro de Autorização

O filtro de autorização funciona da seguinte forma: Se for feita a requisição para o endpoint de login, então o cookie que é responsável por armazenar o token de acesso é removido e o fluxo segue para o controller e service de login. Caso contrário, o filtro de autorização extrai o token do cookie, valida, extrai o username como subject e os roles embutidos e carrega o objeto Authentication com username e os roles para a autorização do spring security funcionar. Caso o token seja inválido ou esteja expirado, o backend retorna uma mensagem de erro.

Antes de cada requisição chegar ao devido endpoint, ela passa pelo filtro de autorização e são executados os seguintes passos:

- A requisição chega ao filtro de autorização
- Se a requisição vier de outro microserviço, o token vem como header Authorization: Bearer
- Se a requisição vier do frontend, o token vem como cookie http only
- Se o token vier como cookie, ele é extraído e o header de Authorization: Bearer token é adicionado a requisição
- Então, são extraídos do token o username e os roles e o SecurityContextHolder é carregado com esses dados de autenticação para a autenticação e autorização do endpoints funcionar.

🔑 O Login

Durante a autenticação, acontece o seguinte: O frontend envia uma requisição para o endpoint de login. O service do endpoint de login busca o usuário, seus grupos e roles no banco de dados pelo username e senha e gera o access token e o refresh token. Então, gera os cookies httponly para o access token e o refresh token e os retorna para o navegador armazenar os cookies como httponly e os reenviar a cada nova requisição aos endpoints do backend.

🛡️ O interceptor de API no frontend e o Refresh Token

Quando o frontend envia uma requisição e o filtro de autorização retorna uma mensagem 401 ou 403, o interceptor de API captura a resposta e supõe que o token expirou. Então, gera uma nova requisição para o endpoint de refresh token para que um novo token seja gerado. Por sua vez, o service utilizado pelo endpoint de refresh token faz o seguinte:

- Extrai o refresh token do cookie
- Extrai o username do refresh token
- Busca o usuário pelo username
- Carrega os roles do usuário
- Gera o novo token de acesso
- Gera um novo cookie httponly com o token de acesso e retorna para o navegador
- Retorna um DTO de LoginResponse com o novo token, o username e o nome do usuário

Após isto, o interceptor de API recebe o LoginResponse, acessa o access token e seta o access token no AuthContext de Context API.

O access token armazenado no estado da aplicação via Context API é utilizado pelo websocket de atualização de detalhes do dispositivo.

📡 A autenticação e autorização via Websocket

O websocket tem um interceptor configurado, onde, o token de acesso deve ser extraído do cabeçalho de token de acesso bearer. Então, carrega o username extraído do token para o envio de mensagens para o usuário específico pelo username funcionar. Isso garante que apenas os usuários com o token válido e o devido username recebam a mensagem enviada.

👷 Configuração de Cors

A configuração de cors está no backend da seguinte forma:

- A origem é o host com porta do local onde o frontend estiver hospedado ou localhost:5173 para desenvolvimento.
- O mapping path tem o seguinte padrão: /** englobando todos os endpoints da API e do websocket
- Todos os cabeçalhos são permitidos
- Todos os métodos são permitidos
- As credenciais são permitidas, o que necessita do '*withCredentials: true*' nas requisições via axios

🔗 A comunicação via REST entre microserviços

Os microserviços se comunicam via REST ou messageria. Na comunicação via REST, é utilizado o RestClient para consumo da API do microsserviço alvo. Cada microsserviço integrado possui componentes de integração com os controllers e endpoints dos microsserviços alvo. Os endpoints são configurados no application.properties, bem como o access_token utilizado para acesso a API do microsserviço alvo. Um único access_token é compartilhado pelos microsserviços para comunicação entre eles. Esse access_token não é gerado pela aplicação e é armazenado em local seguro. Isto é, como variável de ambiente dos containers dos microsserviços. Esse access_token de microsserviços tem o username: "microservice" e o role "microservice", necessário

para acessar a API compartilhada. O access_token de microsserviço também, teoricamente, não tem tempo de expiração. É vitalício. Dado que seu tempo de expiração em segundos corresponde ao valor inteiro máximo de 32 bits que deve corresponder a mais de um século.