Ótima colocação — essa é **uma preocupação legítima e comum** em aplicações baseadas em páginas HTML estáticas (MPA) com autenticação por sessão.

Quando o usuário **fecha o navegador sem fazer logout**, alguns efeitos acontecem:

**💣 Problemas potenciais**

1. **Cookies persistem (dependendo do SameSite, Secure e tempo de vida)**
   * O token CSRF (XSRF-TOKEN) pode continuar existindo.
   * O JSESSIONID pode estar inválido (expirado no backend).
2. **Na próxima visita, o JS reutiliza o token do cookie**, mas:
   * O servidor rejeita o XSRF-TOKEN pois a SESSIONID não existe mais.
   * Isso resulta em erro 403, mesmo com um formulário de login válido.

**✅ Soluções preventivas eficazes**

**✅ 1. Sempre requisitar um novo CSRF token na tela de login**

Já discutido, mas reforçado aqui: **a tela de login não deve confiar no cookie CSRF**. Ela **sempre deve buscar um novo token do backend**.

Isso evita o uso de um token desassociado de sessão, mesmo que o navegador tenha sido fechado abruptamente.

**Vantagens:**

* Independente de limpeza do navegador.
* Compatível com Spring Security e backend com sessão.

**✅ 2. Backend deve invalidar sessão após inatividade**

Spring Boot já expira sessões inativas automaticamente, mas certifique-se de configurar:

properties

CopiarEditar

server.servlet.session.timeout=15m

Ou no application.yml:

yaml

CopiarEditar

server:

servlet:

session:

timeout: 15m

Assim, mesmo se o usuário **não fechar o navegador**, a sessão expira naturalmente e um novo token CSRF será necessário após esse tempo.

**✅ 3. Interceptar erro 403 no frontend e forçar fluxo limpo**

Nas páginas protegidas (que não são login), quando um fetch retornar **403**, redirecione para a tela de login e **ignore o token CSRF existente**:

js

CopiarEditar

if (resposta.status === 403) {

// Sessão possivelmente expirada

window.location.href = "tela\_login.html";

}

**✅ 4. Se quiser reforçar ainda mais: usar sessionStorage para controle em JS**

Como sessionStorage é **limpo automaticamente quando o navegador ou aba fecha**, você pode usá-lo como um “sinal” complementar de que a sessão foi limpa.

Exemplo (na tela de login):

js

CopiarEditar

if (!sessionStorage.getItem("session\_ativa")) {

// Força novo token, não usa o do cookie

obterNovoToken();

}

E ao logar com sucesso:

js

CopiarEditar

sessionStorage.setItem("session\_ativa", "sim");

Assim, **a sessão só é considerada ativa se o browser não tiver sido fechado**.

**🧠 Conclusão**

Você **não pode impedir que o usuário feche o navegador**, mas pode **construir o frontend e backend para reagir graciosamente** a esse comportamento e **não reaproveitar tokens inválidos**.

Assegurando:

* Sessões expiradas no servidor,
* Requisição de novo token ao acessar login,
* Redirecionamento em erro 403,
* E, se quiser, sinalização complementar via sessionStorage.

Essa abordagem combinada dá **resiliência contra abandono de sessão** sem depender da disciplina do usuário.