Información como esta implementado el 2FACTOR

**CLASES**

* **User** : Está relacionada con el template Profile.html hoy no carga.
* **UserControler**: maneja los datos del usuario
* **TwoFactorAuthenticationFilter:** Maneja la lógica con servicios para doble factor
* **UserService:**
  + Maneja la lógica de negocio con la clase User maneja el CRUD(mirar cual no se utiliza y cual e podría reutilizar)
    - **Authenticate**
    - **registerUser**
    - **updateUser**
    - **userExists**
    - **deleteUserById**
    - **isUserInactive**
    - **findByUsernameisActive**
    - **isTwoFactorEnabled (mirar para implantar la necesidad de activar y desactivar)**
    - **getUserEmail**
    - **findByUsername**
    - **sabe**
* **SecurityConfig:**
  + Maneja la seguridad de la aplicación y se utiliza también par el doble factor
    - .addFilterBefore(twoFactorAuthenticationFilter(), UsernamePasswordAuthenticationFilter.class);
    - .antMatchers("/verify\_2fa").permitAll()
    - public TwoFactorAuthenticationFilter twoFactorAuthenticationFilter() {  
       return new TwoFactorAuthenticationFilter();  
      }
    - Conecta con la clase TwoFactorAuthenticationFilter
* **TwoFactorController**:
  + Maneja la verificación del doble factor
    - // Establecer cookie opcional  
      Cookie cookie = new Cookie("2FA\_STATUS", "VERIFIED");  
      cookie.setMaxAge(3600); // 1 hora  
      cookie.setPath("/");  
      response.addCookie(cookie);  
        
      // Redirigir al usuario a la página de destino  
      return "redirect:/listar";
    - log.warn("Verificación fallida para el usuario: " + username);  
        
      // Agregar el mensaje de error a RedirectAttributes  
      redirectAttributes.addFlashAttribute("error", "El código es incorrecto");  
        
      // Redirigir a la página de verificación  
      return "redirect:/verify\_2fa";
* **CustomUserDetailsService:**
  + Implementa a **UserDetailsService librería de java**
  + Hay un log que dice si el Usuario activo: null username: juanma40 ocurre cuando el usuario no está activo
* **UserRepository:**
  + Se utiliza en CustomUserDetailsService
* **LoginController:**
  + Utiliza la lógica para la autentificación del usuario y el doble factor(VERIFICAR PAR ACONTROALR QUE SI ETA ACTIVA O NO EL DOBLE FACTOR QUE NO ENVIE EL COREO)
* **TwoFactorMessageService:**
  + Maneja el envio de código si es loguedo correctamente VERIFICAR PAR ACONTROALR QUE SI ETA ACTIVA O NO EL DOBLE FACTOR QUE NO ENVIE EL COREO
  + Se usa **CustomUserDetailsService en el método doFilterInternal**
  + Se maneja la lógica para el envio del codigo

**TEMPLATES**

* **Login.html:**
  + Es la primera pagina que ve el usuario y se verifica si esta activa y se envía el doble factor por email.
  + Se redirige a **LoginController** donde está la lógica para la verificación y él envió del email (VERIFICAR PAR ACONTROALR QUE SI ETA ACTIVA O NO EL DOBLE FACTOR QUE NO ENVIE EL COREO)
* **Verify-totp.html:**
  + Es la pantalla que maneja la introducción del código doble factor conecta con el contrlador
* **Profile.html:**
  + Profile desde el icono del personaje en la aplicación se puede ver los datos del usuario logado. relación con clase **user** tiene un campo twofactor psrece que no se usa esta template
* **Créate\_user\_with\_role.html:**
  + Plantilla que crea los usuarios para la aplicación añadiendo un rol dentro.
  + Llama su formulario al controlador /registre @PostMapping("/register")  
    public String registerUser(@ModelAttribute("user") User user, Model model, @RequestParam List<Long> roles) {
  + Se accede desde
* Administracion/Crear usuarios
  + Falta poner el modelo como los formularios modernos(mejora)
  + Que funcione el doblefactor(incidencia)
  + No abre el menú a las en el icono del admin para cerrar sesión y desplegar su menú (incidencia)  
    Interfaz de usuario gráfica, Aplicación

    El contenido generado por IA puede ser incorrecto.
* **List**\_**user.html**:
  + Muestra la lista de usuarios con opción de dar de baja, se accede del menú
    - Administracion/listar usuarios
  + La opción de editar no funciona no carga (incidencia)
  + Modificar la pantalla con los cambios actuales(mejora)

Interfaz de usuario gráfica, Teams

El contenido generado por IA puede ser incorrecto.

* **Setup-toto.html:**
  + No se utiliza se genera el doble factor por código QR (fue una prueba) considerar eliminar esta template

Necesidades en la implantación:

* añadir un checkbox para activar y desactivar el doble factor y solo entrar por contraseña.

Para condicionar el flujo de verificación de doble factor (2FA) en tu Filter a si el usuario tiene activado twoFactorEnabled, debes hacer lo siguiente:

**Paso 1: Asegúrate de que tienes acceso al objeto User para cada usuario autenticado**

Ya lo estás obteniendo con:

java

CopiarEditar

User user = userRepository.findByUsername(username);

Ahora simplemente necesitas verificar si user.isTwoFactorEnabled() es true.

**Paso 2: Modifica tu filtro para que solo aplique el 2FA si está habilitado**

Aquí tienes el código **modificado** con esa condición añadida:

java

CopiarEditar

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)

throws ServletException, IOException {

Authentication auth = SecurityContextHolder.getContext().getAuthentication();

if (auth != null && auth.isAuthenticated()) {

HttpSession session = request.getSession();

String requestURI = request.getRequestURI();

if (requestURI.contains("/verify\_2fa")) {

filterChain.doFilter(request, response);

return;

}

String username = auth.getName();

User user = userRepository.findByUsername(username);

// Verificar si el usuario tiene 2FA activado

if (user != null && user.isTwoFactorEnabled()) {

Boolean isTwoFactorVerified = (Boolean) session.getAttribute("2FA\_VERIFIED");

if (Boolean.TRUE.equals(isTwoFactorVerified)) {

filterChain.doFilter(request, response);

return;

}

Boolean isCodeSent = (Boolean) session.getAttribute("2FA\_CODE\_SENT");

if (isCodeSent == null || !isCodeSent) {

String email = user.getEmail();

if (email != null) {

String generatedCode = twoFactorMessageService.generateVerificationCode();

user.setSecret(generatedCode);

userRepository.save(user);

try {

twoFactorMessageService.sendVerificationCode(email, generatedCode);

} catch (MessagingException e) {

log.severe("Error al enviar el código de verificación: " + e.getMessage());

response.sendError(HttpServletResponse.SC\_INTERNAL\_SERVER\_ERROR, "Error al enviar el código de verificación.");

return;

}

session.setAttribute("2FA\_CODE\_SENT", true);

}

}

response.sendRedirect("/verify\_2fa");

return;

}

// Si 2FA no está habilitado, continuar sin redirigir

filterChain.doFilter(request, response);

return;

}

filterChain.doFilter(request, response);

}

**Resumen de cambios clave:**

* Se obtiene el User desde el userRepository.
* Se verifica si user.isTwoFactorEnabled() es true **antes de aplicar cualquier lógica de 2FA**.
* Si 2FA no está habilitado, se permite el acceso normal (filterChain.doFilter()).

Con esto, solo los usuarios con 2FA activado serán redirigidos a /verify\_2fa, y se generará el código para ellos. Los demás iniciarán sesión sin pasar por el filtro de 2FA.

Gracias por compartir tu fragmento HTML y el controlador de registro. Ya tienes bien integrado el checkbox twoFactorEnabled en el formulario, y el @ModelAttribute("user") en tu @PostMapping("/register") debería mapear automáticamente su valor.

Aquí tienes un **checklist** y **mejoras sugeridas** para garantizar que se guarde correctamente el valor del checkbox en la base de datos:

**✅ Verificaciones que ya están bien:**

**1. ✅ En el formulario:**

html

CopiarEditar

<input type="checkbox" th:field="\*{twoFactorEnabled}" id="2fa-checkbox">

Este checkbox **se vincula directamente** con el atributo twoFactorEnabled del modelo User. Funciona perfectamente si User tiene:

java

CopiarEditar

private boolean twoFactorEnabled;

public boolean isTwoFactorEnabled() { return twoFactorEnabled; }

public void setTwoFactorEnabled(boolean twoFactorEnabled) { this.twoFactorEnabled = twoFactorEnabled; }

**2. ✅ En el controlador:**

En el método registerUser, al recibir el objeto User como @ModelAttribute, **el campo twoFactorEnabled se setea automáticamente** si el checkbox fue marcado.

**⚠️ Posibles problemas si el valor no se guarda:**

1. El checkbox **no envía ningún valor** si está **desmarcado**, lo que puede provocar que el campo twoFactorEnabled en User quede como false por defecto. Esto es el comportamiento correcto de HTML y Java.
2. En la vista, asegúrate de que th:object="${user}" esté definido en el <form> correctamente:

html

CopiarEditar

<form th:action="@{/register}" th:object="${user}" method="post">

**✅ Confirmación en el registerUser:**

Tu código:

java

CopiarEditar

userService.registerUser(user);

Asegúrate de que dentro de ese método no estés sobreescribiendo twoFactorEnabled. Por ejemplo:

java

CopiarEditar

public void registerUser(User user) {

user.setPassword(passwordEncoder.encode(user.getPassword()));

// NO hacer esto si quieres respetar el valor del checkbox:

// user.setTwoFactorEnabled(false);

userRepository.save(user);

}

**✅ En resumen:**

No necesitas cambiar nada importante. Solo asegúrate de:

* Que el formulario tenga th:object="${user}".
* Que no estés sobreescribiendo twoFactorEnabled al registrar.
* Que la entidad User tenga correctamente el campo con su getter/setter.
* Que la columna en la base de datos permita true/false (boolean, tinyint(1), etc., según tu base de datos).