LabXpert - Système de Gestion de Laboratoire

LabXpert est un système complet de gestion de laboratoire médical développé en Java avec Spring Boot, intégrant une API RESTful et utilisant PostgreSQL comme base de données. L'application vise à améliorer l'efficacité et la précision dans le traitement des analyses médicales pour offrir un service plus rapide et plus précis aux patients.

Fonctionnalités

1. Enregistrement des Échantillons :

- Les techniciens peuvent enregistrers de nouveaux échantillons en spécifiant les informations pertinentes telles que le patient, le type d'analyse et la date de prélèvement.

2.Suivi des Analyses en Cours :

- Interface conviviale pour suivre en temps réel l'état d'avancement des analyses en cours, avec des détails spécifiques pour chaque échantillon.

3. Gestion des Résultats :

- Consignation systématique des résultats d'analyses pour un accès rapide et la possibilité de partager les résultats avec les professionnels de la santé concernés.

4. Gestion des Patients :

- Module dédié pour gérer les informations relatives aux patients, assurant une centralisation des données et une navigation facilitée.

5. Inventaire des Réactifs :

- Intégration d'un suivi des stocks pour garantir la disponibilité des réactifs nécessaires aux différentes analyses.

6. Gestion des Utilisateurs :

- Interface d'administration pour gérer les droits d'accès et les informations des utilisateurs, assurant une sécurité accrue des données.

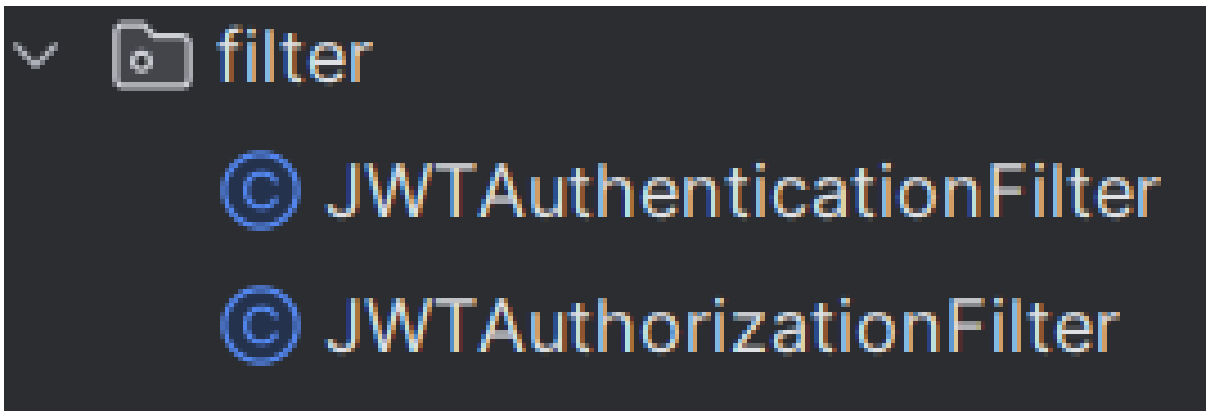
7. Planification des Analyses :

- Possibilité de planifier les analyses en fonction de la charge de travail, optimisant ainsi l'utilisation des ressources du laboratoire.

8. Rapports Statistiques :

- Génération de rapports statistiques pour évaluer les performances du laboratoire, identifier les tendances et prendre des décisions basées sur les données.

Security-JWT-Oauth2.0 :



JWTAuthenticationFilter

La classe JWTAuthenticationFilter étend UsernamePasswordAuthenticationFilter de Spring Security. Elle gère l'authentification en utilisant des jetons JWT. Les principales méthodes incluent:

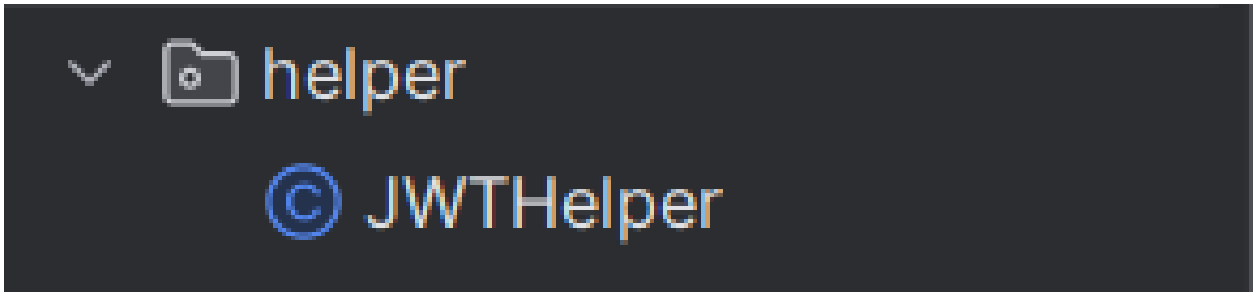
attemptAuthentication: Récupère les informations d'identification de la requête et transmet un jeton d'authentification à l'AuthenticationManager.

successfulAuthentication: En cas d'authentification réussie, génère des jetons d'accès et de rafraîchissement JWT, puis écrit la réponse en format JSON.

JWTAuthorizationFilter

La classe JWTAuthorizationFilter étend OncePerRequestFilter de Spring Security. Elle gère l'autorisation en utilisant des jetons JWT. Les principales méthodes incluent:

doFilterInternal: Extrait, valide et décode le jeton d'accès JWT, configure le contexte de sécurité, puis poursuit la chaîne de filtres.



JWTHelper

La classe JWTHelper est un composant Spring (@Component) fournissant des méthodes utilitaires pour la manipulation des jetons JWT.

Méthodes :

generateAccessToken(String email, List<String> roles):

Génère un jeton d'accès JWT avec le sujet (email), la date d'expiration et les rôles spécifiés.

Signe le jeton avec l'algorithme HMAC256.

generateRefreshToken(String email):

Génère un jeton de rafraîchissement JWT avec le sujet (email) et la date d'expiration.

Signe le jeton avec l'algorithme HMAC256.

extractTokenFromHeaderIfExists(String authorizationHeader):

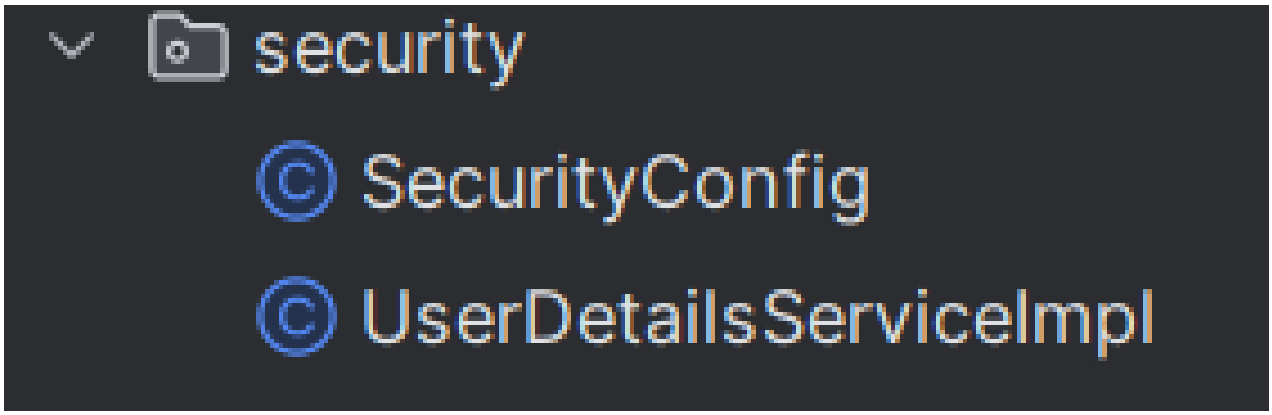
Extrait le jeton d'un en-tête d'autorisation s'il commence par le préfixe défini (Bearer).

Renvoie null si l'en-tête est nul ou ne commence pas par le préfixe.

getTokensMap(String jwtAccessToken, String jwtRefreshToken):

Crée une carte contenant les jetons d'accès et de rafraîchissement.

Les clés de la carte sont "accessToken" et "refreshToken".



SecurityConfig

La classe SecurityConfig est une configuration Spring Security qui définit la manière dont la sécurité doit être gérée dans l'application. Elle utilise des filtres personnalisés pour gérer l'authentification et l'autorisation par jeton JWT.

Méthodes :

filterChain(HttpSecurity http):

Configure la chaîne de filtres de sécurité.

Désactive la protection contre les attaques CSRF.

Configure la politique de création de sessions comme STATELESS (sans session).

Autorise toutes les requêtes authentifiées.

Ajoute le filtre JWTAuthenticationFilter pour gérer l'authentification.

Ajoute le filtre JWTAuthorizationFilter avant le filtre standard UsernamePasswordAuthenticationFilter pour gérer l'autorisation.

Active la connexion via OAuth2.

authenticationManager(AuthenticationConfiguration authConfig):

Récupère le gestionnaire d'authentification à partir de la configuration d'authentification.

UserDetailsServiceImpl

La classe UserDetailsServiceImpl est une implémentation de l'interface UserDetailsService de Spring Security. Elle est responsable de charger les détails d'un utilisateur à partir de la base de données lors de l'authentification.

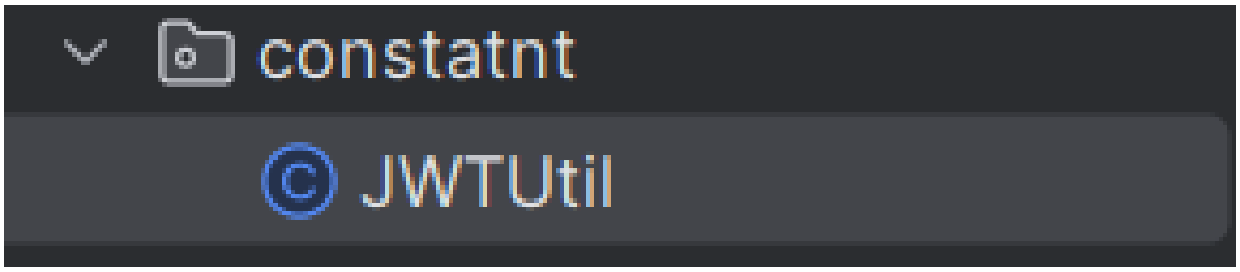
Méthodes :

loadUserByUsername(String email) :

Méthode de l'interface UserDetailsService qui charge les détails d'un utilisateur à partir de la base de données en utilisant l'e-mail comme identifiant.

Utilise le service UtilisateurServiceImpl pour charger l'utilisateur par son e-mail.

Crée un objet UserDetails à partir des informations de l'utilisateur récupérées.



JWTUtil

La classe JWTUtil est une classe utilitaire contenant des constantes liées à la gestion des jetons JWT.

Constantes :

EXPIRE_ACCESS_TOKEN :

Durée de validité du jeton d'accès en millisecondes (ici, 10 minutes).

ISSUER :

Émetteur du jeton JWT.

SECRET_KEY :

Clé secrète utilisée pour signer le jeton JWT.

BEARER_PREFIX :

Préfixe utilisé dans l'en-tête d'autorisation pour indiquer le type de jeton (Bearer).

EXPIRE_REFRESH_TOKEN :

Durée de validité du jeton de rafraîchissement en millisecondes (ici, 2 heures).

AUTH_HEADER :

Nom de l'en-tête d'autorisation.

SpringSecurOAUTH2Github

La classe SpringSecurOAUTH2Github est une configuration de sécurité dédiée à l'intégration de l'authentification OAuth2 avec GitHub.

Méthodes :

defaultSecurityFilterChain(HttpSecurity http) :

Configure la chaîne de filtres de sécurité pour gérer l'authentification avec OAuth2.

Autorise toutes les requêtes authentifiées.

Active l'authentification via OAuth2 pour permettre la connexion avec GitHub