**TP Securite**

**A continuer sur votre API d’étudiants**

**1/ Authentification Basique en mémoire**

Premièrement vous allez commencer par sécuriser les routes avec une authentification Basic.

Pour cela vous devrez créer une classe de configuration étendant WebSecurityConfigurerAdapter. N’oubliez pas @EnableWebSecurity !

La sécurisation des routes se fera via la méthode configure(HttpSecurity http) à surcharger:

* Désactiver la protection CSRF
* Gérer les requêtes autorisées pour que les méthodes POST, PUT et DELETE ne soient accessibles qu’au rôle ADMIN.
* Toutes les autres requêtes, en GET, nécessitent le rôle USER
* Activer l’authentification Basic

Utiliser ensuite un AuthenticationManagerBuilder, via la méthode configureGlobal(...) pour créer des utilisateurs en mémoire.

Il vous en faudra deux : un ayant le rôle ADMIN et un autre ayant le rôle USER.

Pour la définition de ces utilisateurs, n’oubliez pas de définir un PasswordEncoder, et de l’utiliser dans la méthode password :

@Bean

**public** PasswordEncoder passwordEncoder() {

**return new** BCryptPasswordEncoder();

}

password(passwordEncoder().encode(**"password2"**))

**2/ Lire les utilisateurs en base de données**

Vous allez ajouter une table USERS à votre modèle de données.

Cette table contiendra les identifiants des utilisateurs ainsi que leurs mot de passe et leurs rôles. Créer un repository associé pour récupérer son contenu.

Pour associer l’AuthenticationManager et la table USERS, vous devez créer un service qui implémente UserDetailsService.

Surchargez ensuite la méthode loadUserByUsername(String username) qui convertit une entité de la table Users en un UserDetails.

Pour cela vous pouvez utiliser un UserBuilder contenu dans la classe User.

Vous devrez désactiver l’authentification en mémoire pour que tout cela fonctionne.