Compte rendu de l'activité :

SPRING SECURITY

1. **Dépendance Maven**

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

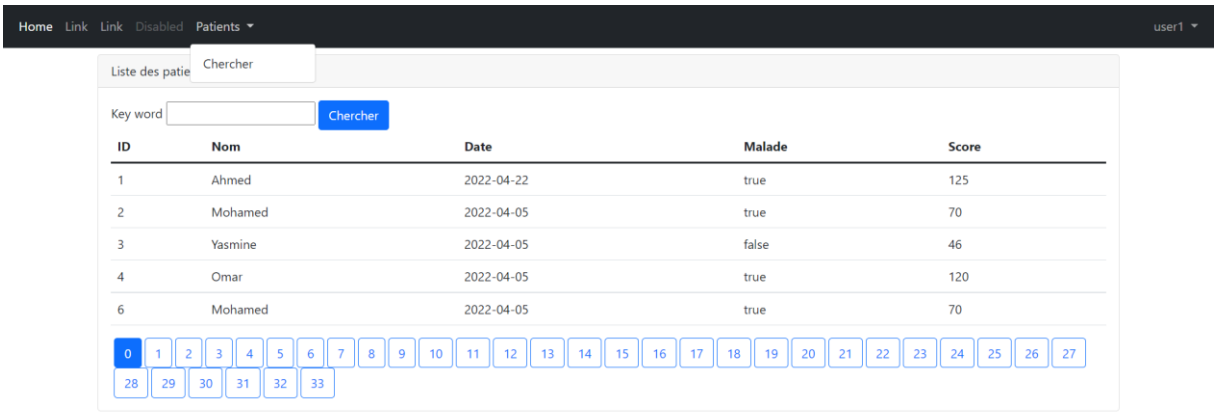
2. **Spring security filtre**

3. **Mot de passe généré par défaut de l'application user**

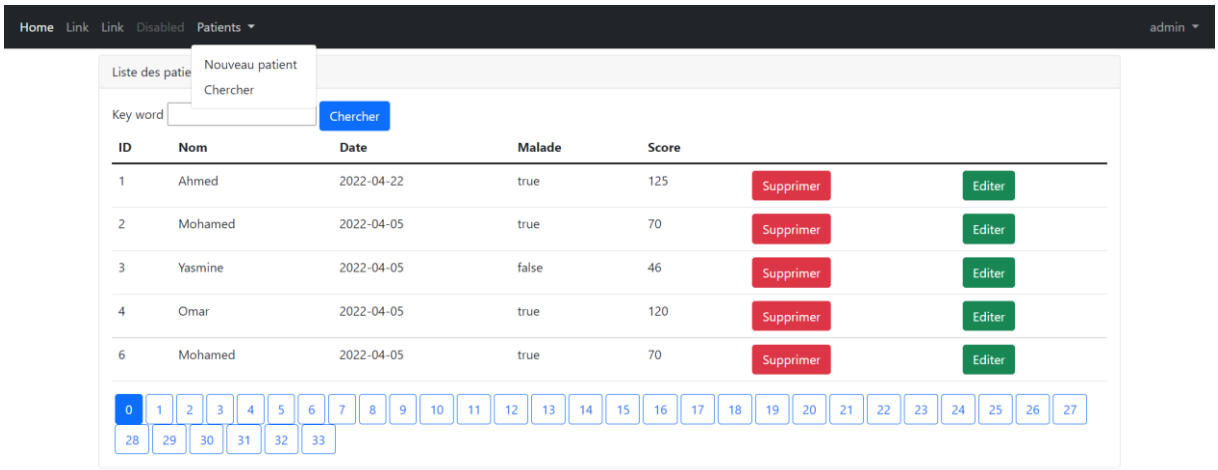
```
Using generated security password: 4bc2dee1-5249-47fb-b4c2-7084506deb27

This generated password is for development use only. Your security configuration must be updated before running your application in production.
```

4. Restrictions de la stratégie InMemoryAuthentication

a. Rôle USER



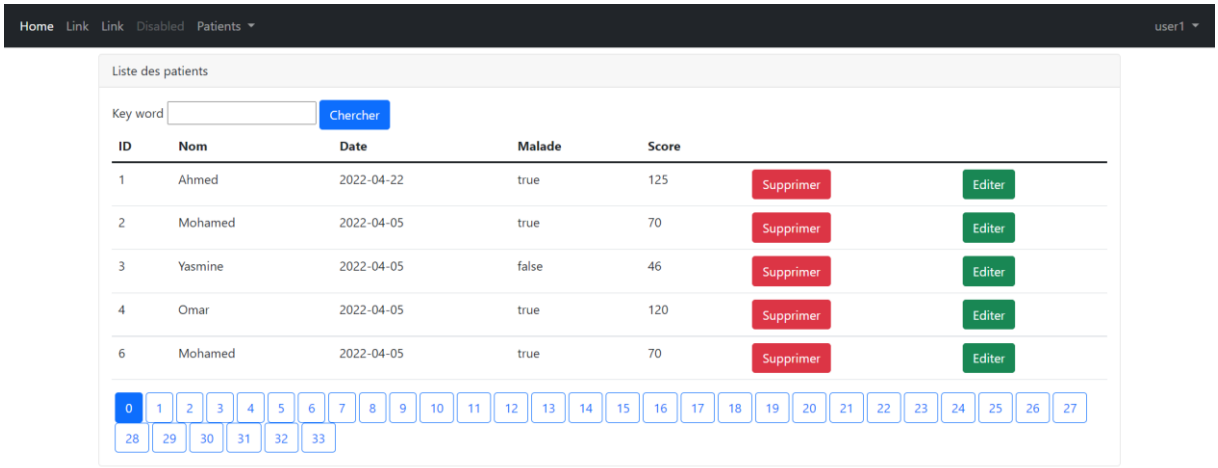
b. Rôle ADMIN



5. Stratégie JDBC Authentication

```java
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    PasswordEncoder passwordEncoder = passwordEncoder();
/*      String encodedPWD = passwordEncoder.encode("1234");
    System.out.println(encodedPWD);
    auth.inMemoryAuthentication().withUser("user1").password(encodedPWD).roles("USER");
    auth.inMemoryAuthentication().withUser("user2").password(passwordEncoder().encode("1111")).roles("USER");
    auth.inMemoryAuthentication().withUser("admin").password(passwordEncoder().encode("2345")).roles("ADMIN","USER");*/
    auth.jdbcAuthentication()
            .dataSource(dataSource)
            .usersByUsernameQuery("select username as principal, password as credentials, active from users where username=?")
            .authoritiesByUsernameQuery("select username as principal, role as role from users_roles where username=?")
            .rolePrefix("ROLE_")
            .passwordEncoder(passwordEncoder);
}
```

Ajouter le rôle « ADMIN« à « user1«



6. Stratégie UserDetailsService

Création d'un package sécurité nommé sec contenant les implémentations suivantes

```java
package ma.emsi.patientsmvc.sec.repositories;

import ...

public interface AppUserRepository extends JpaRepository<AppUser,String> {
    AppUser findByUsername(String username);
}
```

```java
package ma.emsi.patientsmvc.sec.repositories;

import ...

public interface AppRoleRepository extends JpaRepository<AppRole,Long> {
    AppRole findByRoleName(String roleName);
}
```

```java
package ma.emsi.patientsmvc.sec.entities;

import ...

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class AppUser {
    @Id
    private String userId;
    @Column(unique = true)
    private String username;
    private String password;
    private boolean active;
    @ManyToMany(fetch = FetchType.EAGER)
    private List<AppRole> appRoles = new ArrayList<>();
}
```

```java
package ma.emsi.patientsmvc.sec.entities;

import ...

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class AppRole {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long roleId;
    @Column(unique = true)
    private String roleName;
    private String description;
}
```

```java
//@Bean
CommandLineRunner saveUsers(SecurityService securityService){
    return args -> {
        securityService.saveNewUser( username: "mohamed", password: "1234", rePassword: "1234");
        securityService.saveNewUser( username: "yasmine", password: "1234", rePassword: "1234");
        securityService.saveNewUser( username: "hassan", password: "1234", rePassword: "1234");

        securityService.saveNewRole( roleName: "USER", description: "");
        securityService.saveNewRole( roleName: "ADMIN", description: "");

        securityService.addRoleToUser( username: "mohamed", roleName: "USER");
        securityService.addRoleToUser( username: "mohamed", roleName: "ADMIN");
        securityService.addRoleToUser( username: "yasmine", roleName: "USER");
        securityService.addRoleToUser( username: "hassan", roleName: "USER");
    };
}
```

```java
@Override
public AppUser saveNewUser(String username, String password, String rePassword) {
    if(!password.equals(rePassword)) throw new RuntimeException("Password does not match");
    String hashedPWD = passwordEncoder.encode(password);
    AppUser appUser = new AppUser();
    appUser.setUserId(UUID.randomUUID().toString()); // Génerer des identifiants en se référant à la date systeme
    appUser.setUsername(username);
    appUser.setPassword(hashedPWD);
    appUser.setActive(true);
    AppUser savedAppUser = appUserRepository.save(appUser);
    return savedAppUser;
}
```

```java
@Override
public AppRole saveNewRole(String roleName, String description) {
    AppRole appRole = appRoleRepository.findByRoleName(roleName);
    if (appRole != null) throw new RuntimeException("Role " +roleName+ " already exists.");
    appRole = new AppRole();
    appRole.setRoleName(roleName);
    appRole.setDescription(description);
    AppRole savedAppRole = appRoleRepository.save(appRole);
    return savedAppRole;
}

@Override
public void addRoleToUser(String username, String roleName) {
    AppUser appUser = appUserRepository.findByUsername(username);
    if (appUser == null) throw new RuntimeException("User not found.");
    AppRole appRole = appRoleRepository.findByRoleName(roleName);
    if (appRole == null) throw new RuntimeException("Role not found.");
    appUser.getAppRoles().add(appRole);
}
```

```java
@Override
public void removeRoleToUser(String username, String roleName) {
    AppUser appUser = appUserRepository.findByUsername(username);
    if (appUser == null) throw new RuntimeException("User not found.");
    AppRole appRole = appRoleRepository.findByRoleName(roleName);
    if (appRole == null) throw new RuntimeException("Role not found.");
    appUser.getAppRoles().remove(appRole);
}

@Override
public AppUser loadUserByUserName(String username) { return appUserRepository.findByUsername(username); }
```

```
auth.userDetailsService(userDetailsService);
```

```java
package ma.emsi.patientsmvc.sec.service;

import ...

@Service
public class UserDetailsServiceImpl implements UserDetailsService {
    private SecurityService securityService;

    public UserDetailsServiceImpl(SecurityService securityService) { this.securityService = securityService; }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        AppUser appUser = securityService.loadUserByUserName(username);
/*        Collection<GrantedAuthority> authorities = new ArrayList<>();
        appUser.getAppRoles().forEach(role ->{
            SimpleGrantedAuthority authority = new SimpleGrantedAuthority(role.getRoleName());
            authorities.add(authority);
        });*/
        Collection<GrantedAuthority> authorities1 =
                appUser.getAppRoles() List<AppRole>
                        .stream() Stream<AppRole>
                        .map(role ->
                        new SimpleGrantedAuthority(role.getRoleName())) Stream<SimpleGrantedAuthority>
                        .collect(Collectors.toList());

        User user = new User(appUser.getUsername(), appUser.getPassword(), authorities1);
        return user;
    }
}
```

```java
package ma.emsi.patientsmvc.sec.service;

import ...

public interface SecurityService {
    AppUser saveNewUser(String username, String password, String rePassword);
    AppRole saveNewRole(String roleName, String description);
    void addRoleToUser(String username, String roleName);
    void removeRoleToUser(String username, String roleName);
    AppUser loadUserByUserName(String username);
}
```