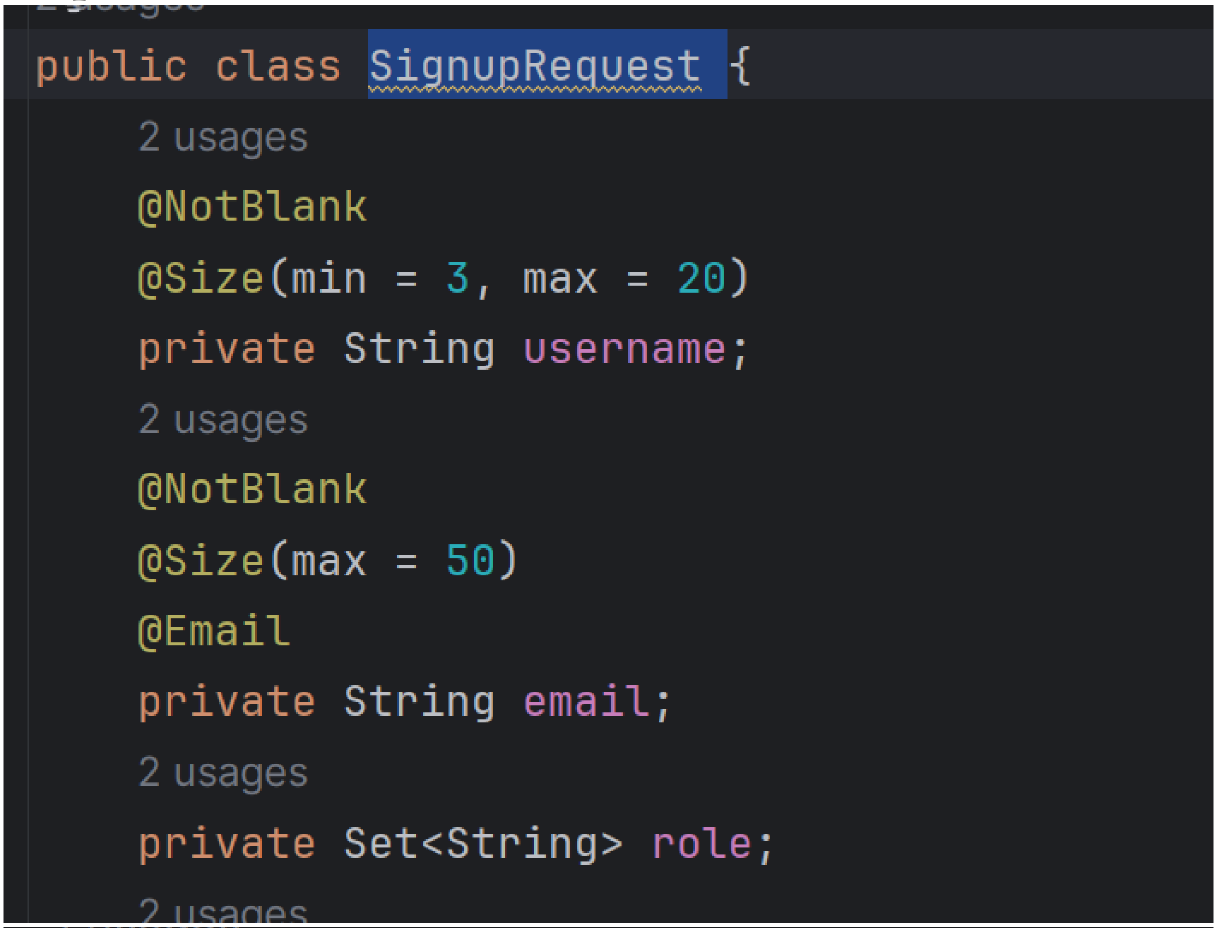
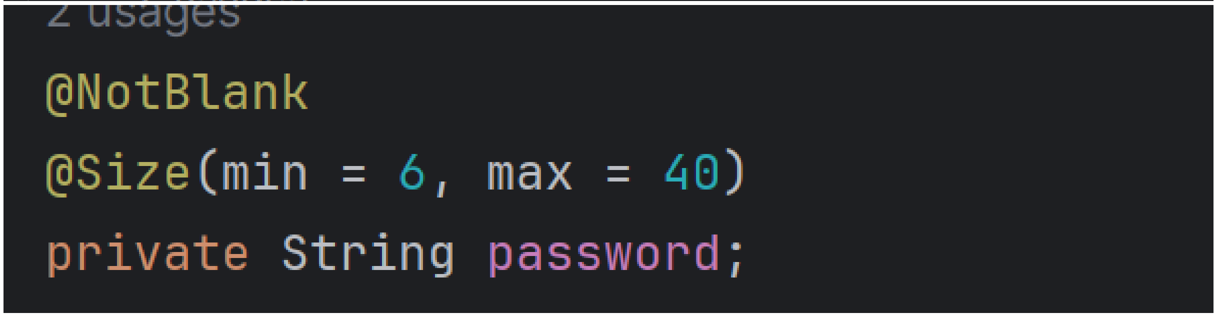
TP Sécurité

Partie 3:

1) Créer une classe « **SignupRequest** » dans package **Request**

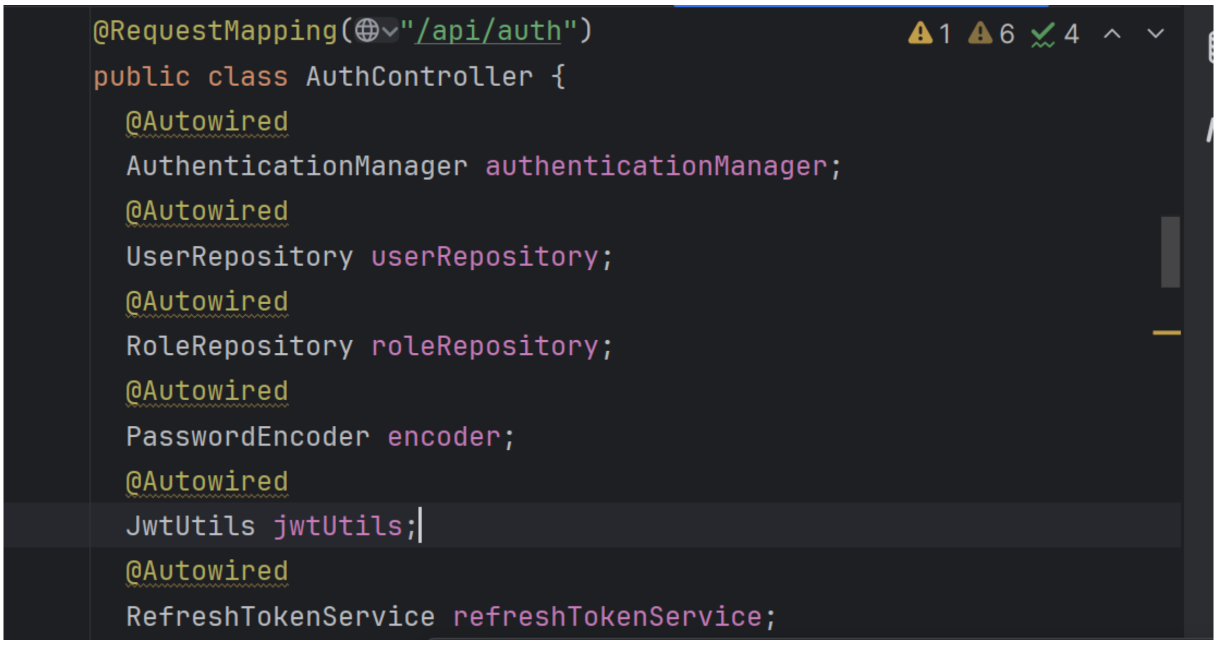


```java
public class SignupRequest {
    2 usages
    @NotBlank
    @Size(min = 3, max = 20)
    private String username;
    2 usages
    @NotBlank
    @Size(max = 50)
    @Email
    private String email;
    2 usages
    private Set<String> role;
    2 usages
```



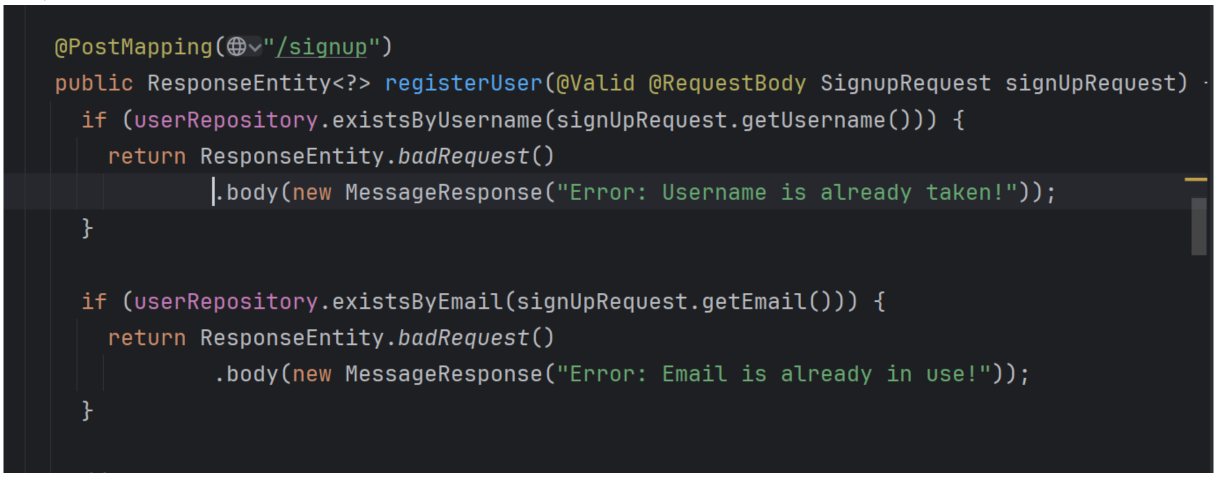
```java
2 usages
@NotBlank
@Size(min = 6, max = 40)
private String password;
```

2) Créer une Classe « **AuthController»**

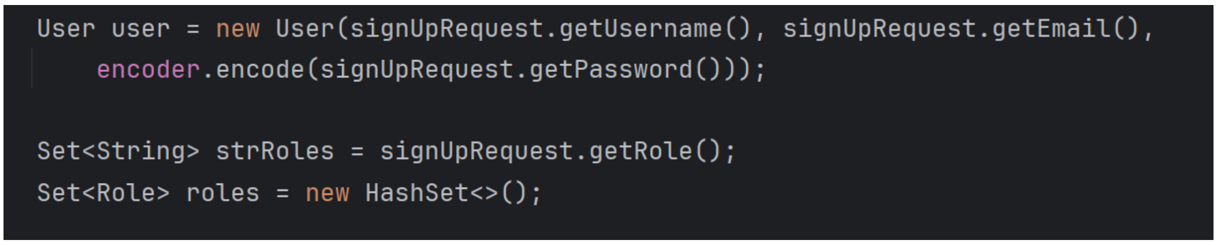


```java
@RequestMapping("/api/auth")
public class AuthController {
  @Autowired
  AuthenticationManager authenticationManager;
  @Autowired
  UserRepository userRepository;
  @Autowired
  RoleRepository roleRepository;
  @Autowired
  PasswordEncoder encoder;
  @Autowired
  JwtUtils jwtUtils;
  @Autowired
  RefreshTokenService refreshTokenService;
```

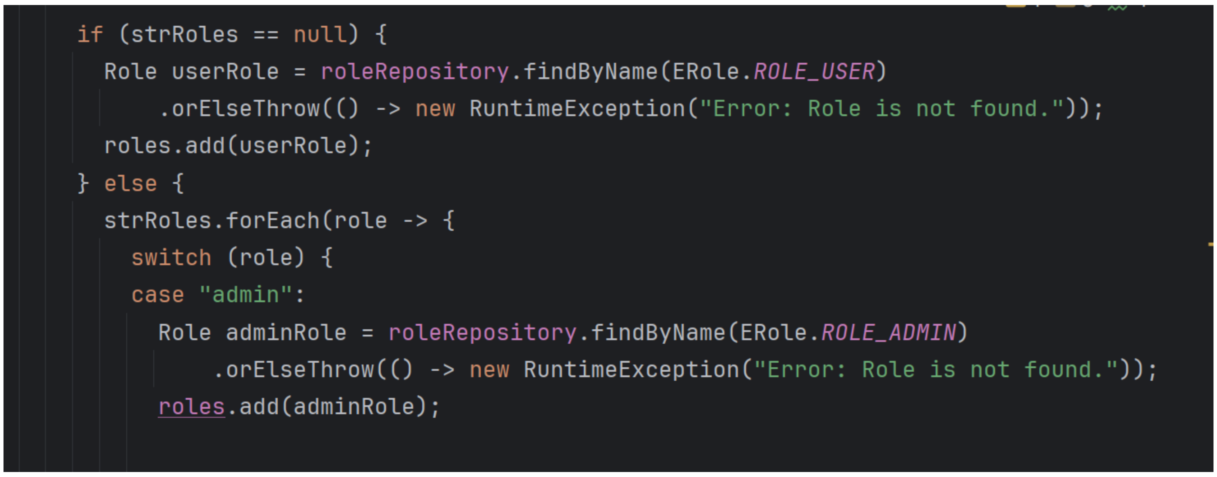
Ajouter la fonction **registrerUser**



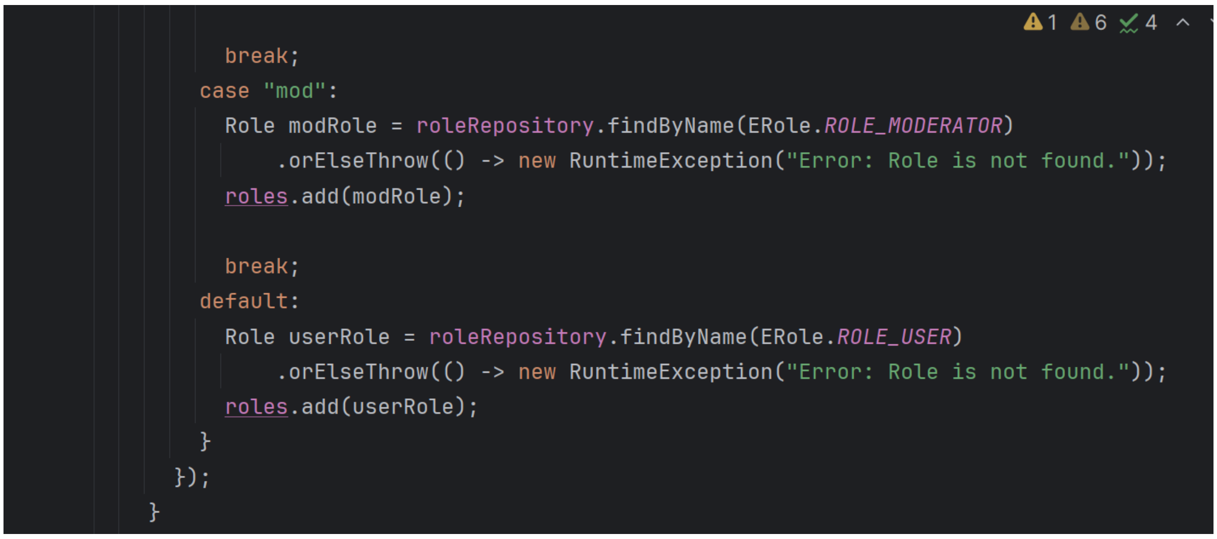
```java
@PostMapping("/signup")
public ResponseEntity<?> registerUser(@Valid @RequestBody SignupRequest signUpRequest)
  if (userRepository.existsByUsername(signUpRequest.getUsername())) {
    return ResponseEntity.badRequest()
            .body(new MessageResponse("Error: Username is already taken!"));
  }

  if (userRepository.existsByEmail(signUpRequest.getEmail())) {
    return ResponseEntity.badRequest()
            .body(new MessageResponse("Error: Email is already in use!"));
  }
```



```java
User user = new User(signUpRequest.getUsername(), signUpRequest.getEmail(),
    encoder.encode(signUpRequest.getPassword()));

Set<String> strRoles = signUpRequest.getRole();
Set<Role> roles = new HashSet<>();
```

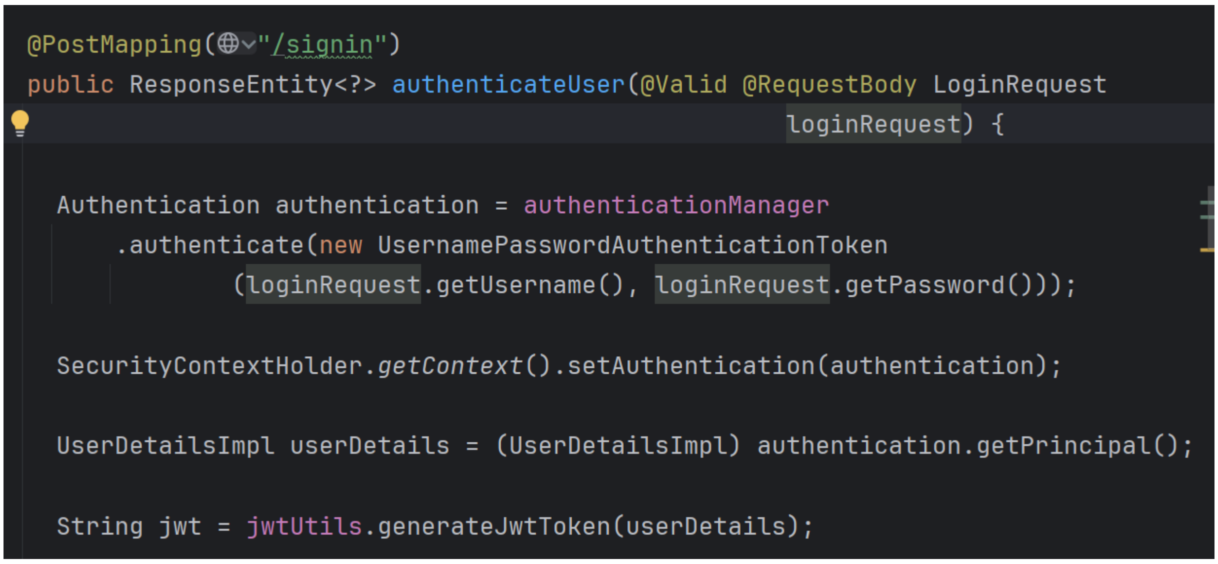


```
    if (strRoles == null) {
      Role userRole = roleRepository.findByName(ERole.ROLE_USER)
          .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
      roles.add(userRole);
    } else {
      strRoles.forEach(role -> {
        switch (role) {
        case "admin":
          Role adminRole = roleRepository.findByName(ERole.ROLE_ADMIN)
              .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
          roles.add(adminRole);
```

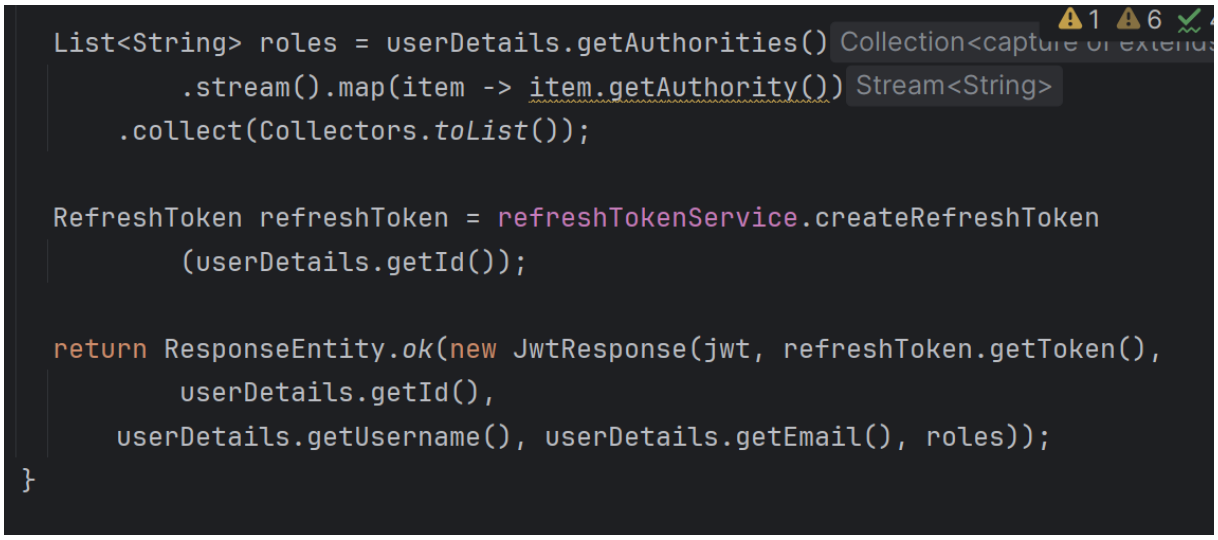


```
          break;
        case "mod":
          Role modRole = roleRepository.findByName(ERole.ROLE_MODERATOR)
              .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
          roles.add(modRole);

          break;
        default:
          Role userRole = roleRepository.findByName(ERole.ROLE_USER)
              .orElseThrow(() -> new RuntimeException("Error: Role is not found."));
          roles.add(userRole);
        }
      });
    }
```

Ajouter la fonction **authenticateUser**

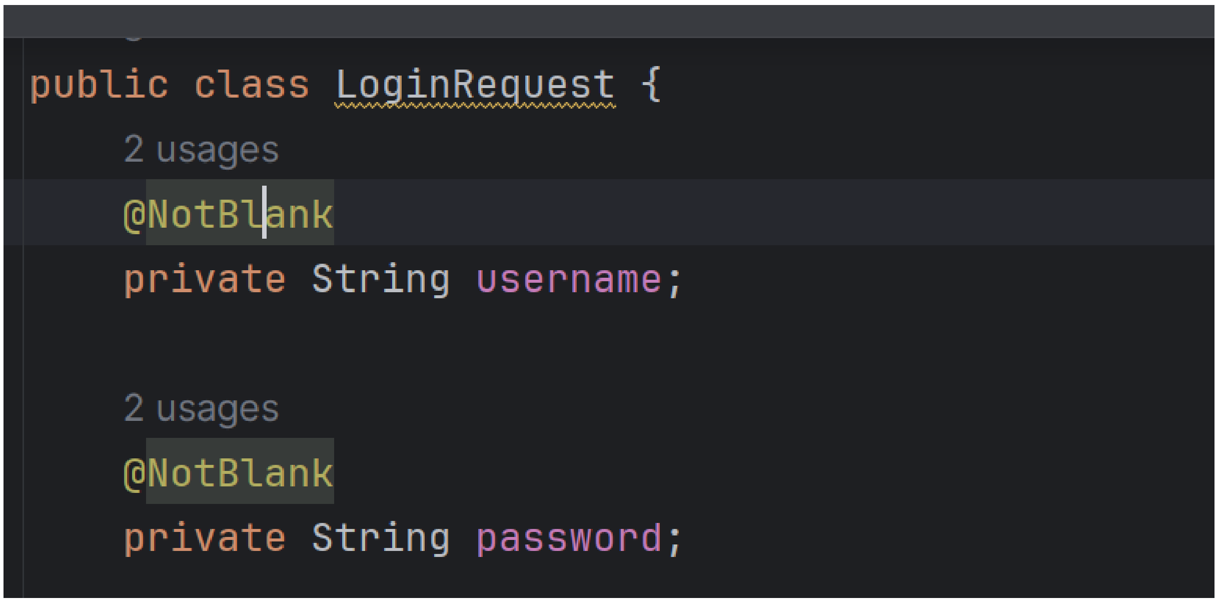


```
@PostMapping("/signin")
public ResponseEntity<?> authenticateUser(@Valid @RequestBody LoginRequest
                                                           loginRequest) {

  Authentication authentication = authenticationManager
      .authenticate(new UsernamePasswordAuthenticationToken
              (loginRequest.getUsername(), loginRequest.getPassword()));

  SecurityContextHolder.getContext().setAuthentication(authentication);

  UserDetailsImpl userDetails = (UserDetailsImpl) authentication.getPrincipal();

  String jwt = jwtUtils.generateJwtToken(userDetails);
```



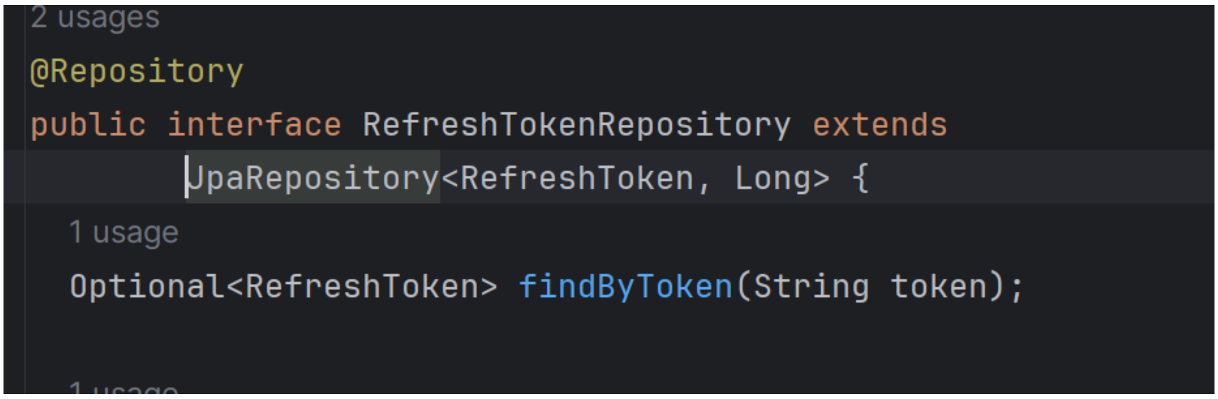
```
List<String> roles = userDetails.getAuthorities() Collection<capture of extends
        .stream().map(item -> item.getAuthority()) Stream<String>
    .collect(Collectors.toList());

RefreshToken refreshToken = refreshTokenService.createRefreshToken
        (userDetails.getId());

return ResponseEntity.ok(new JwtResponse(jwt, refreshToken.getToken(),
        userDetails.getId(),
    userDetails.getUsername(), userDetails.getEmail(), roles));
}
```

Créer une classe « **LoginRequest** »



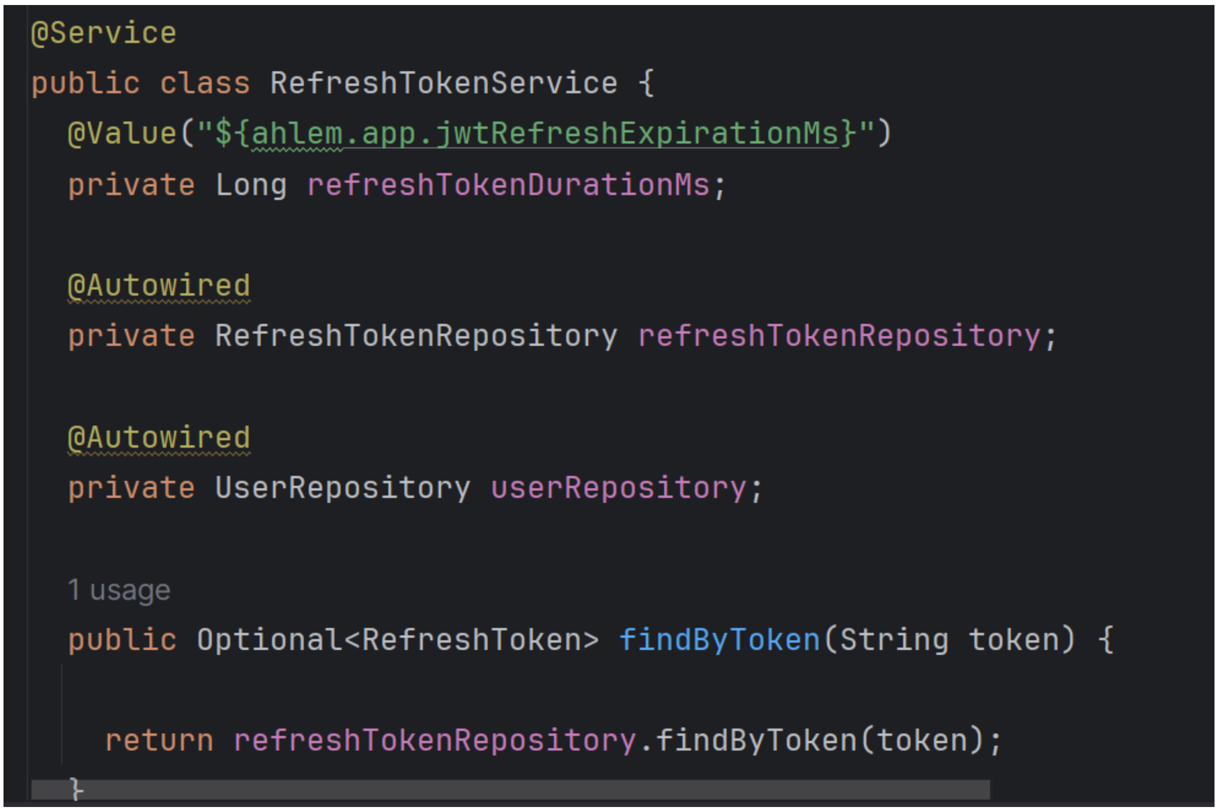
```
public class LoginRequest {
    2 usages
    @NotBlank
    private String username;

    2 usages
    @NotBlank
    private String password;
```

Créer une interface RefreshTokenRepository

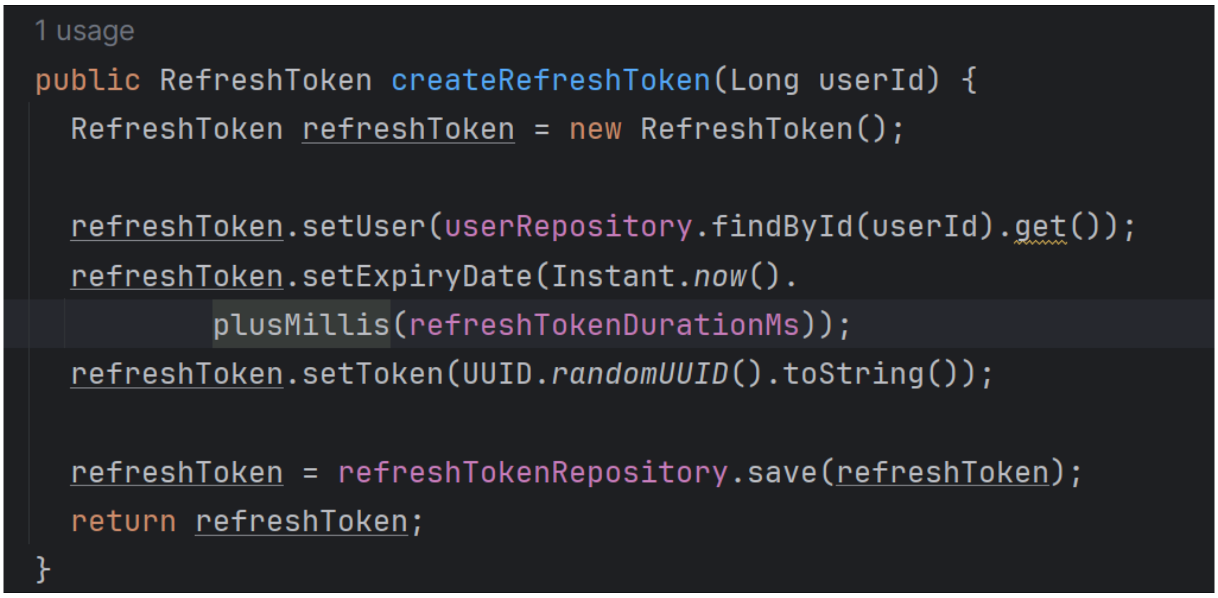


```
2 usages
@Repository
public interface RefreshTokenRepository extends
        JpaRepository<RefreshToken, Long> {
    1 usage
    Optional<RefreshToken> findByToken(String token);
```

Créer une Classe « **RefreshTokenService »** dans package **Service**

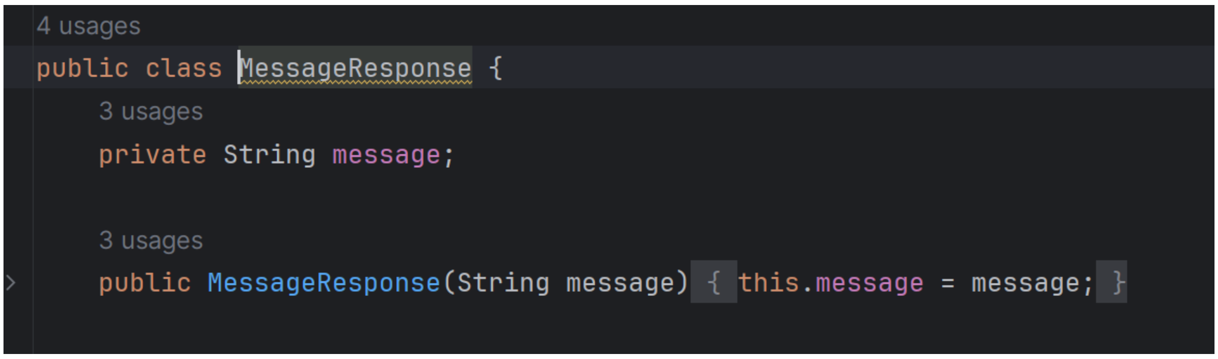


```
@Service
public class RefreshTokenService {
  @Value("${ahlem.app.jwtRefreshExpirationMs}")
  private Long refreshTokenDurationMs;

  @Autowired
  private RefreshTokenRepository refreshTokenRepository;

  @Autowired
  private UserRepository userRepository;

  1 usage
  public Optional<RefreshToken> findByToken(String token) {

    return refreshTokenRepository.findByToken(token);
  }
```



```
1 usage
public RefreshToken createRefreshToken(Long userId) {
  RefreshToken refreshToken = new RefreshToken();

  refreshToken.setUser(userRepository.findById(userId).get());
  refreshToken.setExpiryDate(Instant.now().
          plusMillis(refreshTokenDurationMs));
  refreshToken.setToken(UUID.randomUUID().toString());

  refreshToken = refreshTokenRepository.save(refreshToken);
  return refreshToken;
}
```

Créer une Classe « MessageResponse »



```
4 usages
public class MessageResponse {
    3 usages
    private String message;

    3 usages
    public MessageResponse(String message) { this.message = message; }
```