FORMATION

- Management

2024

Bienvenue sur cette formation Spring

- Clément Hamon
- Développeur depuis 5 ans
- 3 ans de Java Spring Boot en entreprise
- 2 ans de formation
- clement.hamon35@gmail.com

Horaire et convocations

- 9h30 - 17h30
- 15 mn de pause le matin
- 1h de pause déjeuner
- 15 mn de pause l’après midi

- QCM à la fin pour vous évaluer

digi namic FORMATION

Déroulé et structure de la formation et formalités

- Emargement le matin et l’après-midi
- Google Forms de demi-journée pour la validation des acquis et l'adaptabilité
- Notions théoriques suivis de mise en pratique

Tour de table et pré-requis

- Qui êtes-vous et quel est votre expérience dans l'informatique ?
- Vos attentes à l'issue de cette formation ?

- Bonne connaissance de Java

Structure et versionning Github

- Pour chaque notion montrée, un commit sera disponible sur ce git
- https://github.com/ClementHamonDev/Formation-Spring

- Support de cours sur Teams et Github

Objectifs de la formation

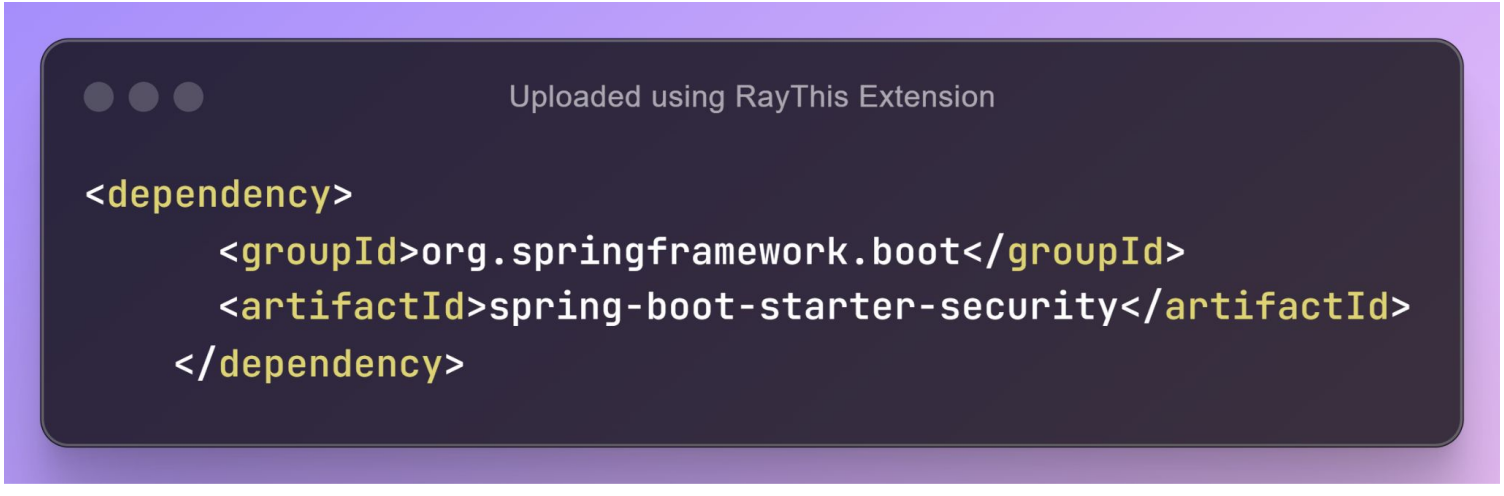
1. Mettre en œuvre le module Spring boot
2. Mettre en place une API et des échanges avec une base de données
3. Maîtriser la configuration et la sécurité

Installation des outils

- Installation Java 21
- IDE Visual Code Studio
- Plugins
- Postman

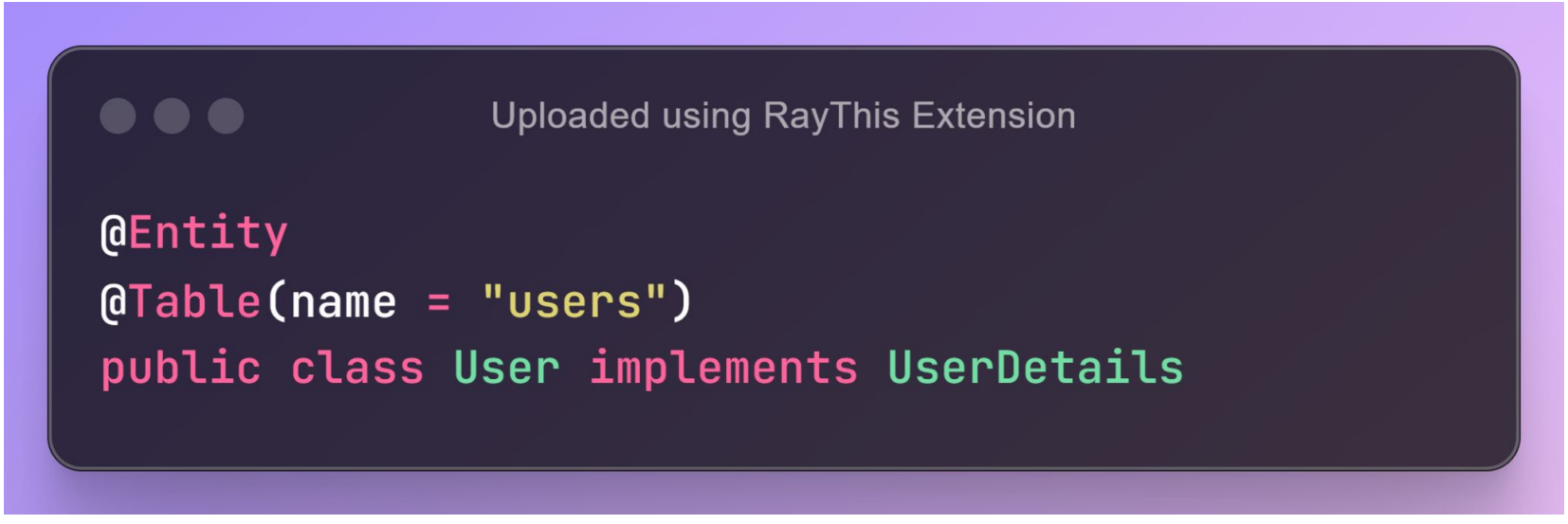
1. Authentification

pom.xml

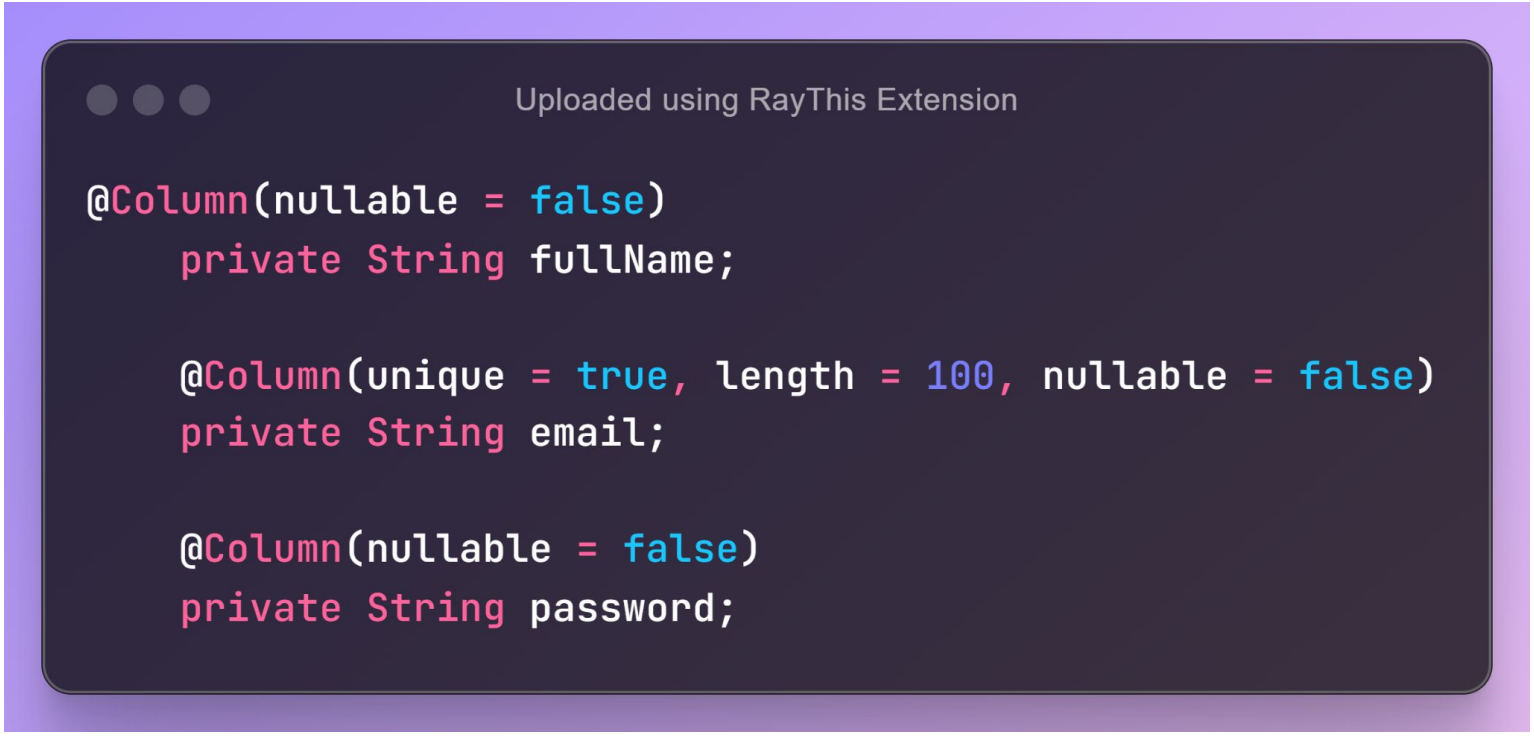


pom.xml

User


Uploaded using RayThis Extension
@Entity
@Table(name = "users")
public class User implements UserDetails

User



Créons ensuite le JpaRepository associé et la méthode findByEmail

Json Web Token (JWT)

https://www.akto.io/tools/hmac-sha-256-hash-generator

Exercice

- *Créer un nouveau projet, un User et mettre en place un service de JWT*

Bonus :

- *Tester les méthodes en dur dans un CommandLineRunner pour créer et récupérer un JWT*

Configuration

Filtre par JWT

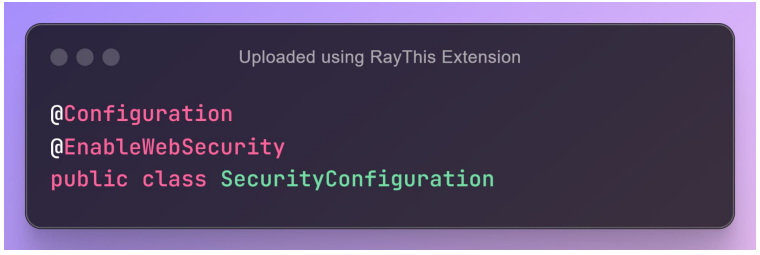
Uploaded using RayThis Extension

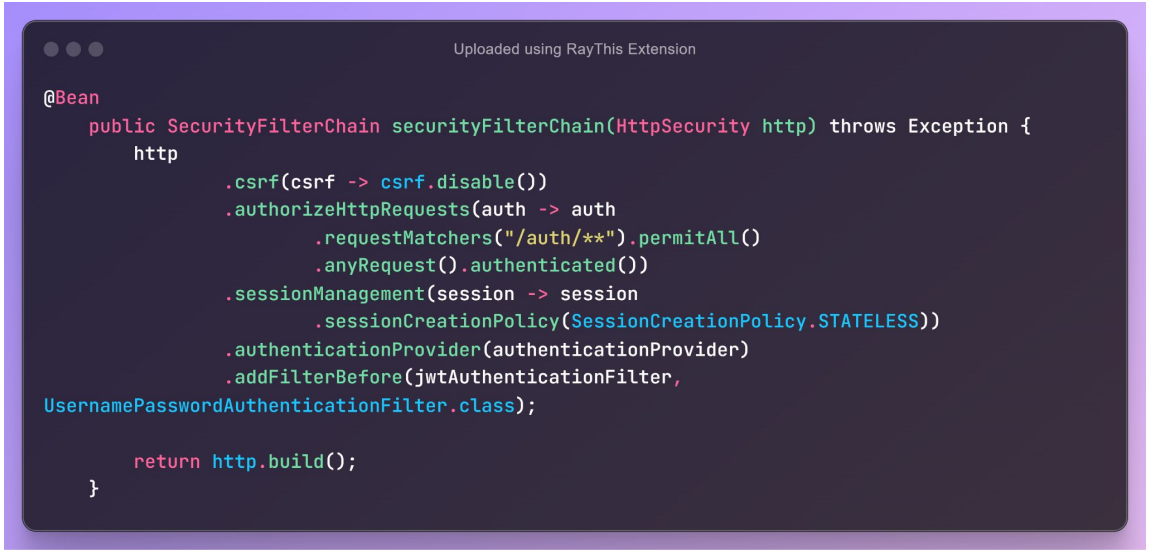
```
@Component
public class JwtAuthenticationFilter extends OncePerRequestFilter
```

Uploaded using RayThis Extension

```
@Override
   protected void doFilterInternal(
       @NonNull HttpServletRequest request,
       @NonNull HttpServletResponse response,
       @NonNull FilterChain filterChain
   ) throws ServletException, IOException {
       final String authHeader = request.getHeader("Authorization");

       if (authHeader == null || !authHeader.startsWith("Bearer ")) {
           filterChain.doFilter(request, response);
           return;
```

SecurityConfig et filterChain





Exercice

- *Mettre en place la config*

Bonus :

- *Créer un controller simple pour tester l'accès à une route protégée*

Data Transfer Object (DTO)

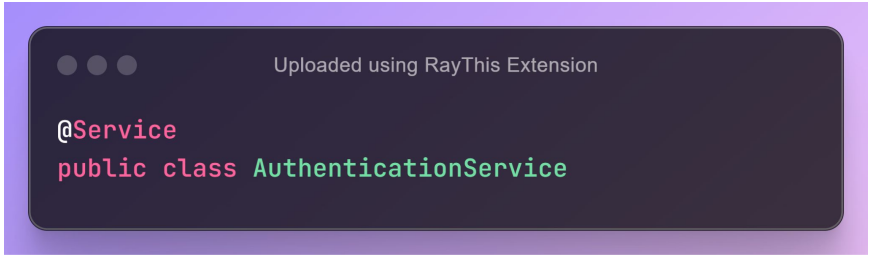
Uploaded using RayThis Extension

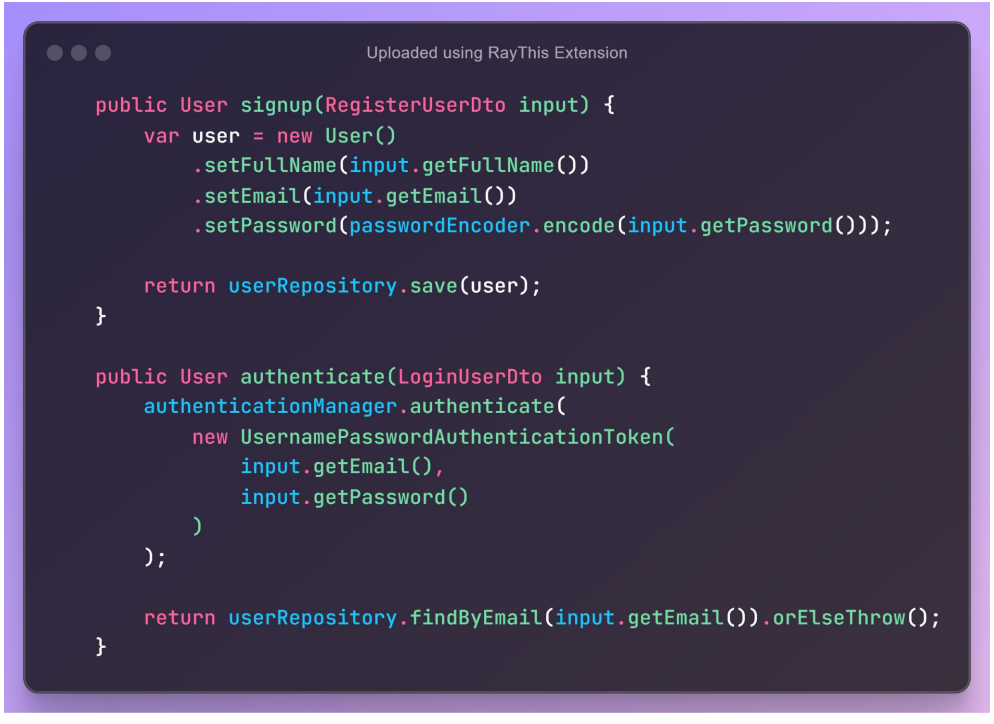
```java
public class LoginUserDto {
    private String email;
    private String password;

    public String getEmail() {
        return email;
    }

    public LoginUserDto setEmail(String email) {
        this.email = email;
        return this;
    }

    public String getPassword() {
        return password;
    }

    public LoginUserDto setPassword(String password) {
        this.password = password;
        return this;
    }

    @Override
    public String toString() {
        return "LoginUserDto{" +
                "email='" + email + '\'' +
                ", password='" + password + '\'' +
                '}';
    }
}
```

Uploaded using RayThis Extension

```java
public class RegisterUserDto {
    private String email;
    private String password;
    private String fullName;

    public String getEmail() {
        return email;
    }

    public RegisterUserDto setEmail(String email) {
        this.email = email;
        return this;
    }

    public String getPassword() {
        return password;
    }

    public RegisterUserDto setPassword(String password) {
        this.password = password;
        return this;
    }

    public String getFullName() {
        return fullName;
    }

    public RegisterUserDto setFullName(String fullName) {
        this.fullName = fullName;
        return this;
    }

    @Override
    public String toString() {
        return "RegisterUserDto{" +
                "email='" + email + '\'' +
                ", password='" + password + '\'' +
                ", fullName='" + fullName + '\'' +
                '}';
    }
}
```

Service





Auth Controller


```java
@RequestMapping("/auth")
@RestController
public class AuthenticationController {

    @Autowired
    private JwtService jwtService;
    @Autowired
    private AuthenticationService authenticationService;

    @PostMapping("/signup")
    public ResponseEntity<User> register(@RequestBody RegisterUserDto registerUserDto) {
        User registeredUser = authenticationService.signup(registerUserDto);

        return ResponseEntity.ok(registeredUser);
    }

    @PostMapping("/login")
    public ResponseEntity<LoginResponse> authenticate(@RequestBody LoginUserDto loginUserDto)
{
        User authenticatedUser = authenticationService.authenticate(loginUserDto);

        String jwtToken = jwtService.generateToken(authenticatedUser);

        LoginResponse loginResponse = new
LoginResponse().setToken(jwtToken).setExpiresIn(jwtService.getExpirationTime());

        return ResponseEntity.ok(loginResponse);
    }
}
```

User Controller


```java
@RequestMapping("/users")
@RestController
public class UserController {

    @Autowired
    private UserService userService;

    public UserController(UserService userService) {
        this.userService = userService;
    }

    @GetMapping("/me")
    public ResponseEntity<User> authenticatedUser() {
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();

        User currentUser = (User) authentication.getPrincipal();

        return ResponseEntity.ok(currentUser);
    }

    @GetMapping
    public ResponseEntity<List<User>> allUsers() {
        List <User> users = userService.allUsers();

        return ResponseEntity.ok(users);
    }
}
```

Exercice

- *Finir de mettre en place l'authentification*
- *Tester avec Postman*

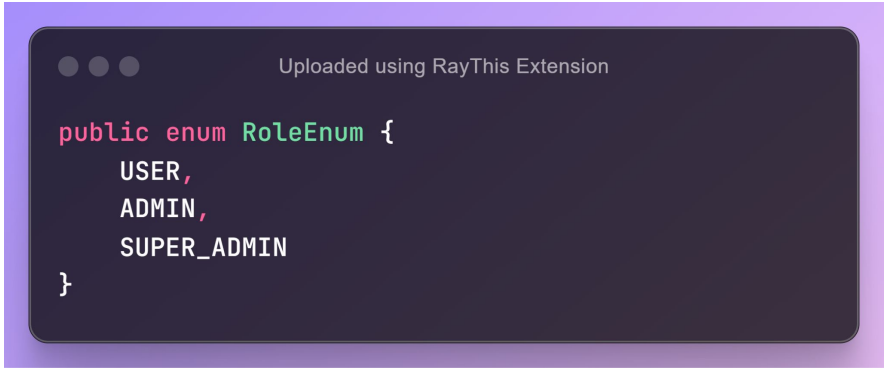
Bonus :

- *Faire un frontend sommaire pour tester*

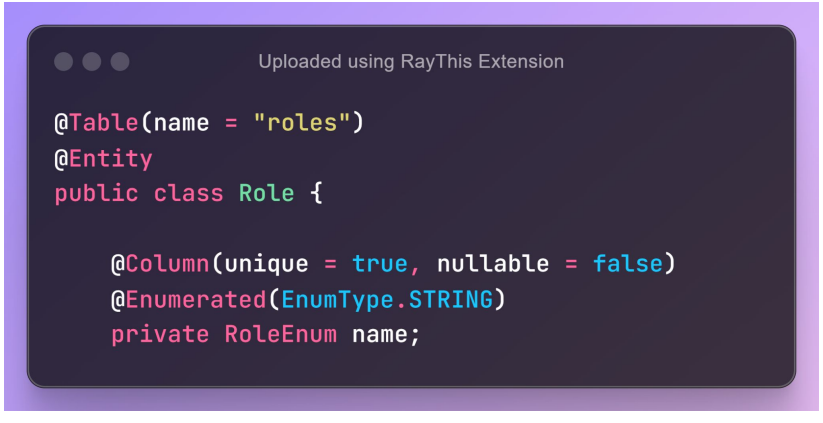
2. Ajout des Roles

Role

Liste des rôles

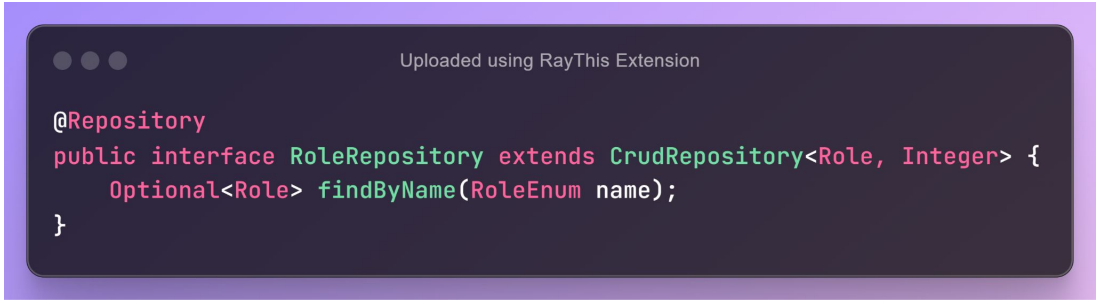


Entity de Role



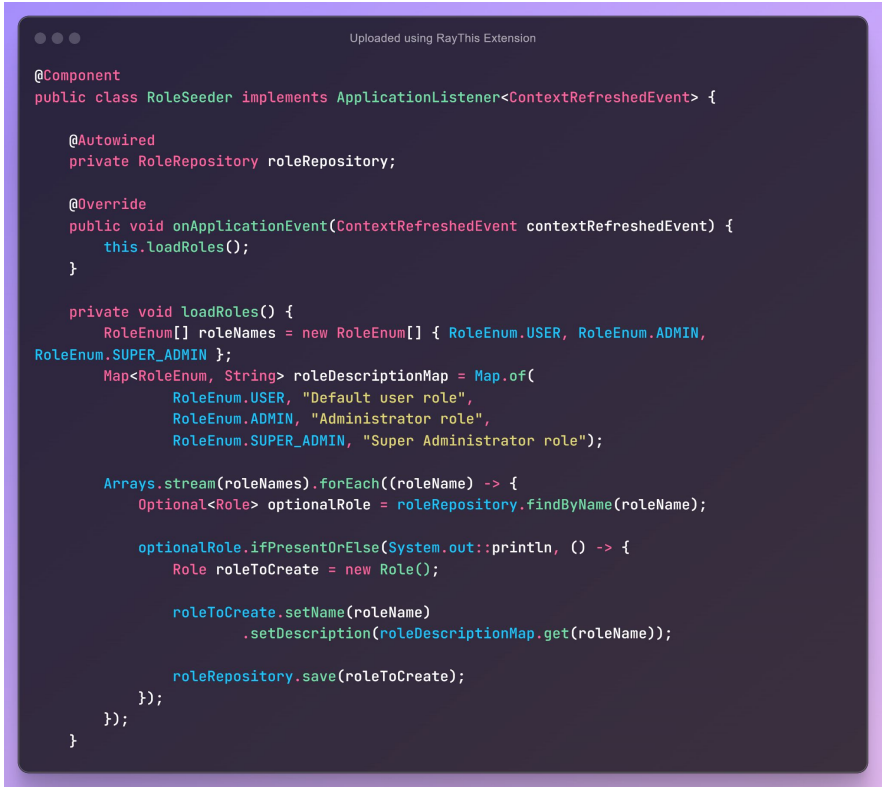
Role

Repo



Role

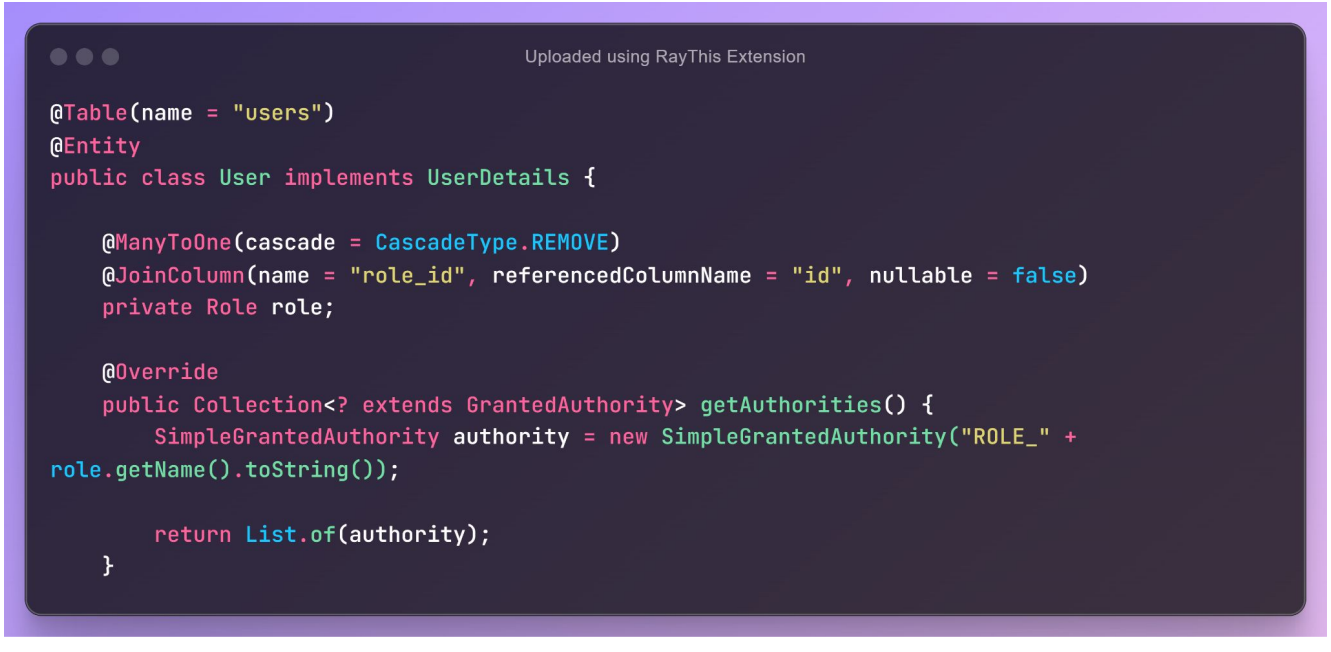
Seeder pour rentrer automatiquement les rôles en DB au lancement



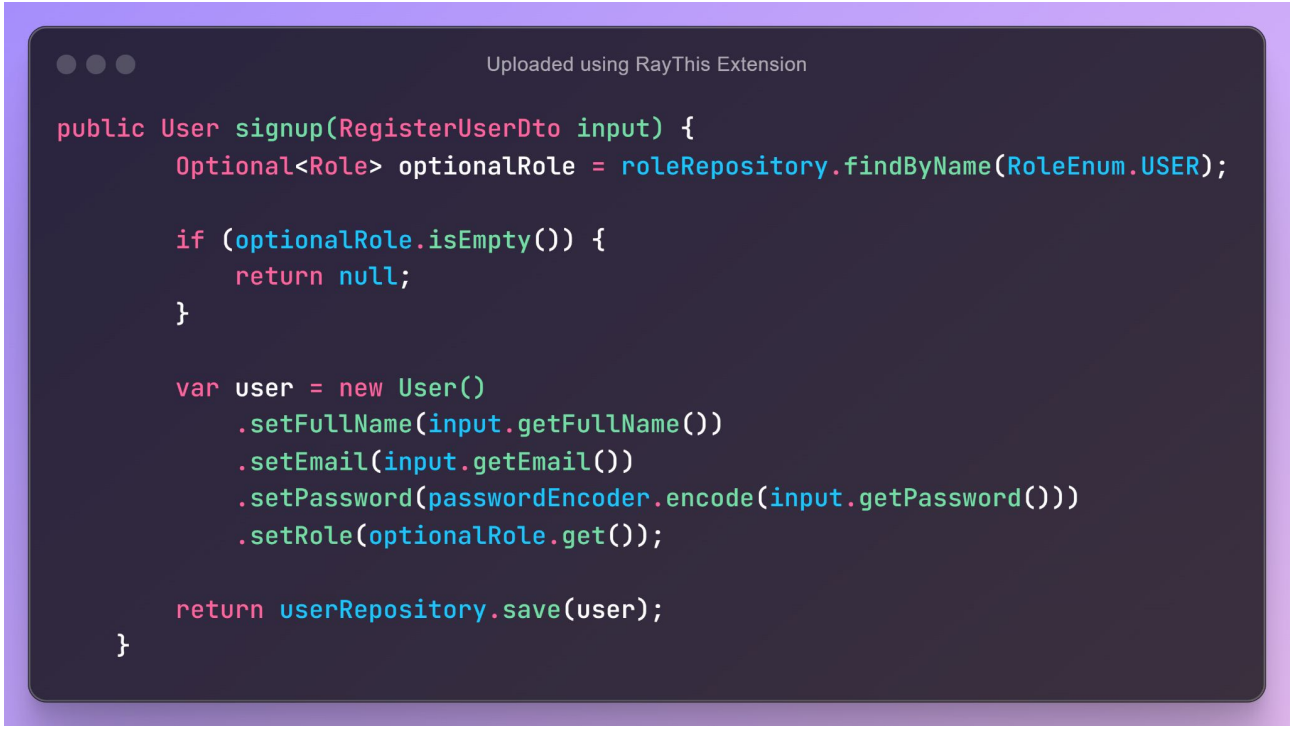
Exercice

- *Créer les fichiers relatifs aux Role et tester le seeder*

Mise à jour de User



Mise à jour de signup()



Restriction

Pour restreindre l'accès des utilisateurs en fonction de leurs rôles, activer cette fonctionnalité dans Spring Security, ce qui permet d'effectuer la vérification sans écrire de logique personnalisée.

Vous devez ajouter l'annotation **@EnableMethodSecurity** dans le fichier de configuration de la sécurité, nommé *SecurityConfiguration.java*.

Restriction

Ouvrez le fichier **UserController.java** et ajoutez les annotations suivantes pour les routes indiquées :

`/users/me` :

```
@PreAuthorize("isAuthenticated()")
```

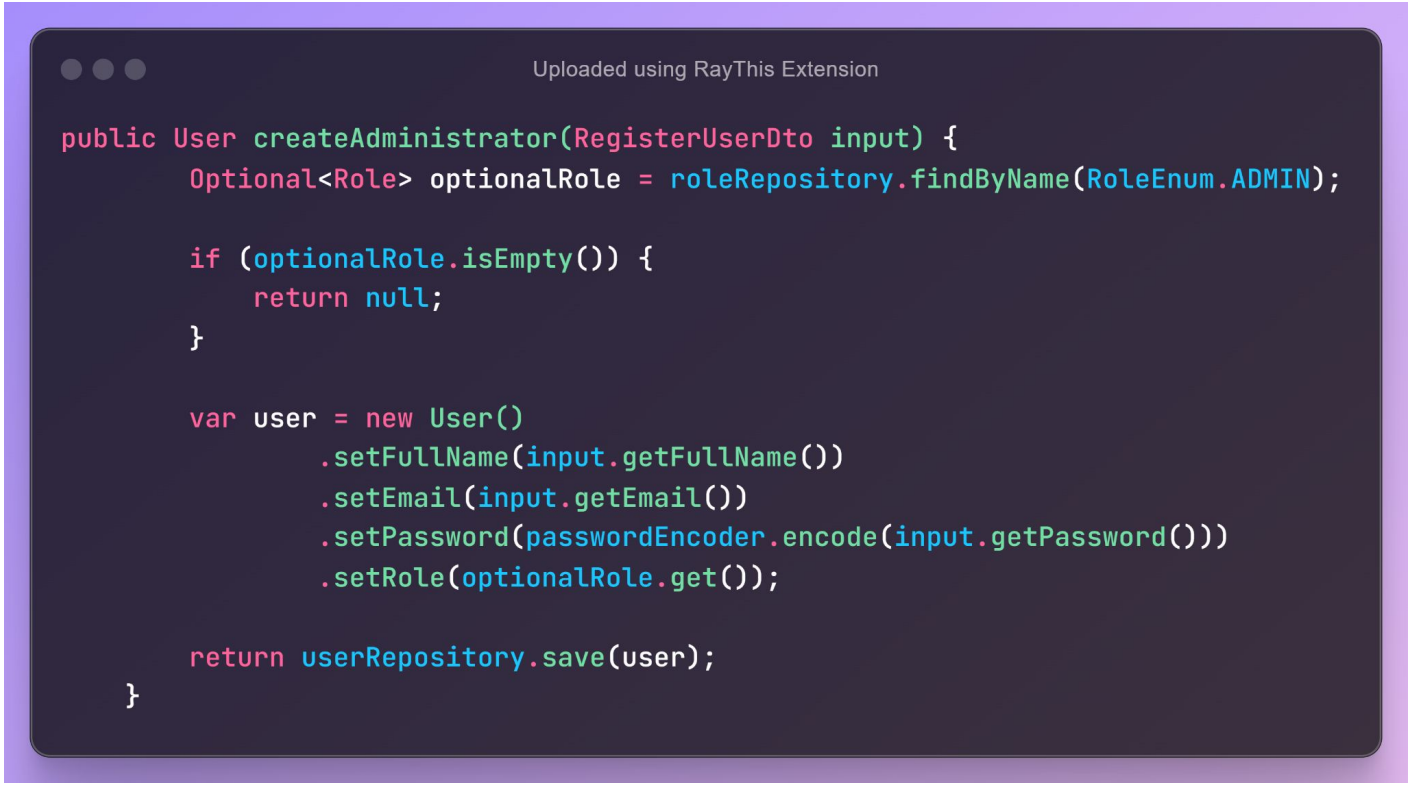
- Étant donné que cette route est accessible à tous les rôles, nous vérifions simplement que l'utilisateur est authentifié.

`/users` :

```
@PreAuthorize("hasAnyRole('ADMIN', 'SUPER_ADMIN')")
```

- Cette route est réservée aux utilisateurs ayant les rôles **ADMIN** ou **SUPER_ADMIN**.

Mise à jour UserService



AdminController


```java
@RequestMapping("/admins")
@RestController
public class AdminController {
    @Autowired
    private UserService userService;

    @PostMapping
    @PreAuthorize("hasRole('SUPER_ADMIN')")
    public ResponseEntity<User> createAdministrator(@RequestBody RegisterUserDto
registerUserDto) {
        User createdAdmin = userService.createAdministrator(registerUserDto);

        return ResponseEntity.ok(createdAdmin);
    }
}
```

Exercice

- *Mettre à jour les fichiers et créer des routes pour tester les différentes protections*

Validation des acquis

- *En reprenant le projet de gestion de magasin, ajoutez des accès sécurisés*

Objectif :

- *Seul un admin peut accèder à toutes les commandes et clients*
- *Tout le monde peut faire un panier*
- *Seul un admin peut ajouter un produit et une catégorie*
- *Seul le super admin peut ajouter un admin*

Bonus : Faire un frontend sommaire pour consommer l'API

Questions / réponses

- Revenons sur les questions hors plan de cours que vous m’avez posé durant la formation pour y répondre

Votre formateur

Clément Hamon
clement.hamon35@gmail.com
https://www.linkedin.com/in/clément-hamon-135485209/

Merci d’avoir suivi cette formation !