Projet de fin de cursus

Pré-requis poste de développement :

- JDK 11+
- IDEs : STS 4.x, VSCode
- Librairie lombok : https://projectlombok.org/downloads/lombok.jar
- Git

Introduction

Le projet de fin de cursus doit permettre de mettre en application les principes vus lors des différentes formation et en particulier :

- Gestion de sources avec Git
- Les technologies back-end : Java, Junit, JPA, Spring
- Les technologies front-end : HTML/CSS, Angular ou React

Le projet peut se faire seul ou en binôme.

Il consiste en :

- Une application back-end SpringBoot offrant une API REST et s'appuyant sur une base de données relationnelles.
  L'application fournit une documentation au format OpenAPI.
  Si elle contient des tests automatisés, c'est un plus
- Une application front-end (React ou Angular) offrant une interface web destinée à des utilisateurs finaux. Les aspects présentation et ergonomie seront importants.

Livraison du projet :

- Dépôt de sources Git
- Package exécutable ou procédure de description pour construire l'exécutable à partir des sources

Calendrier du projet

- Finalisation de la spécification : A la fin de la dernière formation
- Constitution des équipes, mis en place du dépôt de sources : Avant le 7 Octobre

- Développement du projet. Démarrage dés la fin de la formation SpringBoot et surtout pendant les journées dédiées : 7 Octobre, 21 Octobre, 24 et 25 Octobre

- Présentation le 26 Octobre

Spécification fonctionnelle de l'application

L'application est destinée aux utilisateurs d'une médiathèque.

La médiathèque gère un stock de DVD, CD et Livres. Voir les données associées à chaque Item plus loin.

Les utilisateurs de l'application sont les usagers de la médiathèque et doivent s'authentifier par login/mot de passe pour accéder à l'application.

L'application Web leur permet ensuite plusieurs Use Case :

- Parcourir le catalogue du stock et visualiser les items disponibles à l'emprunt. Permettre de voir les nouveautés, uniquement les CD, etc..
- Effectuer un emprunt
- Restituer un emprunt
- Visualiser ses emprunts

L'application devra respecter quelques règles métier :

- Un utilisateur ne peut pas avoir plus de 3 items empruntés simultanément
- La date limite de restitution d'un item est d'1 semaine

Modèle de données des Items.

Les usagers sont représentés par :

- Un login (email)
- Un mot de passe
- Un nom et un prénom

Un DVD est constitué :

- D'un titre
- D'un nombre d'exemplaires
- D'une date de parution
- D'un réalisateur
- D'une durée
- D'un type (blue-ray ou normal)

Un CD est constitué :

- D'un titre
- D'un nombre d'exemplaires
- D'une date de parution
- D'un artiste/groupe
- D'une durée
- D'un nombre de titres

Un Livre est constitué :

- D'un titre

- D’un nombre d’exemplaires
- D’une date de parution
- D’un écrivain
- D’un numéro ISBN

Étapes de réalisation :

Back-end Spring

1) Mise en place dépôt Git : github ou gitlab,

2) Création de projet avec le SpringStarter, choix du nom du projet, du package racine, définition des dépendances.
   Commit

3) Stabilisation du modèle de données : Diagramme des classes modèles, génération automatique des tables par Hibernate, mise en place d'un jeu de données de tests dans un fichier import.sql

4) Définition de la couche Repository : Analyse des requêtes requises par le projet. Éventuellement, écriture de classes de test qui valident que les requêtes font ce que l'on pense.

5) Définition des services métiers (éventuellement dans une interface). Implémentation de la logique métier.
   Classe de tests validant que la logique métier est respectée.

6) Définition de l'API Rest : URLs, Méthodes, Code retour, (en suivant les conventions d'une API Rest). Format JSON des réponses et des corps de requêtes. Implémentation dans les contrôleurs, Tests via tests automatisés ou via Swagger.

7) Gestion des exceptions. Implémentation des gestionnaires d'exception métier

8) Mise en place de la sécurité

- Ajouter le starter spring-security, accèder à l'application et voir le formulaire de login

- Créer un bean WebSecurityConfigurer et surcharger la méthode définissant les ACLS :
  protected void configure(HttpSecurity http) throws Exception
  Tester en utilisant un realm mémoire avec des mots de passe en clair

- Implémenter UserDetailService pour que le realm soit celui de la base H2

- Crypter les mote dans la base (voir le site Bcrypt.org)

9) Classes de tests complet

Bien sûr le processus est itératif

Front-end

….

Rappels *conventions* d'API RestFul

URLS : Les URLs identifient des ressources (~équivalent aux objets du modèle)

Par exemple :
/users : La ressource Collection de User
/users/{id} : Un user particlier
*/u**sers/{id}/roles*** : Les rôles d'un utilisateur particulier

VERBES HTTP :

- GET : Lecture
- POST : Insertion, Ajout
- PUT : Remplacement
- PATCH : Mise à jour partielle
- DELETE : Suppression

Données d'entrées :

- Type simple Obligatoire (par exemple un id) : Via le chemin, ex ***/users/{id}***
- Type simple Optionnel : Via les paramètres de requête. Ex ***/users?q=dupont*** Recherche des utilisateurs via mot-clé dupont
- Données structurées : Via le corps de la requête en effectuant de la validation
- Sécurité via les entêtes

Données en retour :

- Données structurées en JSON
- Rien

Principaux Codes retours HTTP

- 200 OK
- 201 CREATED
- 204 NO-CONTENT
- 400 BAD REQUEST
- 404 NOT FOUND
- 401 FORBIDDEN (Nécessite une identification)
- 403 NOT-ALLOWED (Les droits ne suffisent pas)