Fiche Projet CDA - Plateforme de réservation de créneaux ou de salles

Nom du projet : *à choisir*

Durée : 2 semaines

Objectifs pédagogiques :

- - Concevoir une application web en architecture REST
- - Maîtriser Spring Boot, Spring Data JPA, Spring Security
- - Travailler en mode projet avec gestion de versions (Git)
- - Développer une interface utilisateur avec Thymeleaf ou en SPA (Vue.js / React)
- - Appliquer les bonnes pratiques (tests, validation, …)

1. Présentation du projet

Création d'une plateforme de réservation permettant à des utilisateurs de réserver des créneaux horaires pour des ressources (salles, bureaux, prestations, coachs...).

Les utilisateurs peuvent s'inscrire, se connecter, réserver, consulter leurs réservations passées et futures. Les administrateurs peuvent gérer les ressources, les disponibilités et les comptes utilisateurs.

2. Fonctionnalités

Côté utilisateur :

- - Inscription, authentification, gestion du profil
- - Recherche de disponibilités par date / type de ressource
- - Réservation d'un créneau
- - Historique de réservations
- - Annulation / modification d'une réservation

Côté administrateur :

- - Gestion des utilisateurs (liste, suppression, changement de rôle)
- - CRUD complet sur les ressources
- - Gestion des créneaux disponibles (plages horaires, jours ouvrés, etc.)
- - Tableau de bord des réservations
- - Export des données (CSV / PDF)

3. Architecture technique

- - Backend : Java, Spring Boot, Spring Security, JPA/Hibernate
- - Base de données : MySQL
- - Frontend :
- Option 2 : SPA : React + appels API REST

4. Organisation des équipes

- - Travail en groupe de 2 à 4 étudiants
- - Un repo Git par groupe
- - Utilisation d'un board Trello ou GitHub Projects pour la gestion de tâches
- - Chaque groupe choisit un nom de projet

5. Contraintes techniques

- - Authentification obligatoire (Spring Security)
- - Gestion des rôles (USER, ADMIN)
- - Validation backend avec Bean Validation
- - Tests unitaires sur les services et répertoires clés
- - Utilisation des DTO et mapping entre couches
- - Documentation de l'API (Swagger ou README bien structuré)

6. Livrables attendus

- - Code source sur Git (avec README)
- - Base de données pré-remplie (dataset de test)
- - Documentation fonctionnelle (PDF ou Markdown)
- - Soutenance finale (15-20 minutes par groupe)

7. Extensions possibles (bonus)

- - Intégration calendrier externe (Google Calendar)
- - Paiement fictif (Stripe sandbox)
- - Notifications email (Spring Mail / SMTP)
- - Upload de documents ou images de salles
- - Statistiques (chart.js)