Date: 10/03/2025 - 14/03/2025

SPRINT 3 BRIEF 5

Brief 5 : Plateforme de gestion d'une salle de sport avec rôles et réservation

Contexte professionnel

Dans le cadre de la digitalisation de ses services, une salle de sport souhaite mettre en place une **plateforme de gestion** permettant une **organisation fluide des membres, des entraîneurs et des sessions d'entraînement**.

L'objectif est de créer un **système complet et sécurisé** où :
✔ **Les membres** peuvent **réserver des séances** en ligne.
✔ **Les entraîneurs** ont une **vue d'ensemble** des séances et des membres qui y participent.
✔ **L'authentification et la gestion des rôles** sont mises en place pour **contrôler les accès**.
✔ **L'architecture MVC** est respectée pour une organisation claire du code.
✔ **Un utilisateur non inscrit est bloqué sur la page de connexion** tant qu'il ne s'est pas inscrit.
✔ **Un bouton de déconnexion** supprime le token et empêche toute navigation sans authentification.

Modélisation UML

Avant le développement, les diagrammes UML suivants devront être réalisés:

- **Diagramme de classes** : Relations entre **membres, entraîneurs et sessions d'entraînement**.

- **Diagramme de séquence** : Interaction entre les **utilisateurs et l'API REST** (inscription, réservation, consultation).
- **Diagramme de cas d'utilisation** : Définition des actions possibles pour chaque rôle (membre, entraîneur).

Conception de l'API REST

Le backend sera développé avec **Node.js** et **Express.js**, et communiquera avec une base de données **MongoDB** via **Mongoose**.

L'API devra permettre :

- **Authentification sécurisée** avec **JWT**, gestion des rôles (membre, entraîneur).
- **Gestion des membres** : Inscription, connexion, modification du profil.
- **Gestion des entraîneurs** : Consultation des sessions et des membres inscrits.
- **Gestion des sessions d'entraînement** : Création, modification, suppression, consultation.
- **Réservation des sessions** : Un membre peut réserver une session disponible.
- **Protection des routes** :
    - Un **membre non connecté** est bloqué sur la page de connexion.
    - Un **membre non autorisé** ne peut pas accéder aux routes des entraîneurs.
    - **Déconnexion** : Suppression du token, empêchant toute navigation sans reconnexion.

Validation des données :

- Vérification de l'unicité des emails.
- Format et sécurité des mots de passe (hash avec bcrypt).
- Vérification des **plages horaires** et de la disponibilité des sessions.

Mise à jour du Frontend

Le frontend sera développé avec **React.js** et devra inclure :

- **Page d'authentification (inscription et connexion)** : Blocage sur cette page si l'utilisateur n'est pas connecté.
- **Gestion des rôles** :
    - **Membre** : Voir et réserver les sessions disponibles.
    - **Entraîneur** : Voir les sessions et les membres inscrits.
- **Navigation protégée** avec **Protected Routes** (redirection si pas authentifié).
- **Bouton de déconnexion** qui **supprime le token** et bloque l'accès aux pages protégées.

Technologies utilisées

- **Frontend** : React JS, Tailwind CSS, React Router (protected routes)
- **Backend** : Node.js, Express.js, Mongoose
- **Base de données** : MongoDB
- **Authentification** : JWT (jsonwebtoken), bcrypt
- **Validation** : Yup (comparaison avec Joi et Validator.js)
- **Outils** : Postman, GitHub, Draw.io (UML)

Modalités pédagogiques

- **Travail** : binôme (2 personnes).
- **Durée** : **5 jours (10/03/2025 - 14/03/2025)**
- **Évaluation** : Présentation collective, démonstration des fonctionnalités, revue de code et questions techniques.

Livrables

1. **Code source** (GitHub).
2. **Lien de planification** (Trello, Jira, etc.)..
3. **Diagrammes UML** (Classes, Séquence, Cas d'utilisation).
4. **Présentation du projet** (Canva, PowerPoint).

Critères de performance

✅ **Gestion des rôles et authentification sécurisée** (JWT, bcrypt).
✅ **Système de réservation fonctionnel** avec gestion des disponibilités.
✅ **Protection des routes** empêchant la navigation sans connexion.
✅ **Expérience utilisateur fluide** avec React et React Router.
✅ **Architecture MVC respectée** pour une organisation claire du code.
✅ **Code propre et bien structuré** en respectant les bonnes pratiques MERN.

Ressources utiles

- 📌 **Express.js** → https://expressjs.com/
- 📌 **MongoDB & Mongoose** → https://mongoosejs.com/
- 📌 **React JS** → https://react.dev/
- 📌 **JWT - JSON Web Tokens** → https://jwt.io/
- 📌 **Yup - Validation** → https://www.npmjs.com/package/yup
- 📌 **Postman** → https://www.postman.com/
- 📌 **Tailwind CSS** → https://tailwindcss.com/

les compétences techniques visées : C1N2, C4N2, C3N2, C5N2,

C6N2, C7N2.
les compétences transversales visées : CT1N2, CT2N2, CT4N2, CT6N2, CT5N2, CT9N2.