RAPPORT TECHNIQUE : GESTIONNAIRE DE SALLES DE SPORT (RABAT)

Date : 19 Janvier 2026

Technologie : Java / JavaFX

Objet : Gestionnaire de salles de sport

Auteur : Ilyas Essaidi , Aymane Lokman et Achraf Raouzi

1. INTRODUCTION ET OBJECTIFS DU PROJET

Ce projet consiste en le développement d'une application de bureau ("Desktop") visant à centraliser et afficher les informations de diverses salles de sport situées à Rabat. L'objectif principal est de fournir à l'utilisateur une vision immédiate et dynamique de l'état des établissements (Ouvert/Fermé) en fonction de l'heure actuelle.

L'application répond à trois besoins majeurs :

1. **Centralisation :** Regroupement des données (adresse, horaires, capacité) de multiples enseignes (ex: Fitness Park, Gold's Gym).
2. **Temps Réel :** Calcul dynamique du statut d'ouverture basé sur l'horloge système.
3. **Ergonomie :** Interface graphique fluide avec rafraîchissement automatique des données.

2. ARCHITECTURE LOGICIELLE

L'application suit une architecture modulaire inspirée du modèle **MVC (Modèle-Vue-Contrôleur)**, garantissant une séparation claire entre les données, la logique métier et l'interface utilisateur.

2.1 Le Modèle (`SalleDeSport.java`)

Ce fichier représente l'entité métier. Il s'agit d'une classe "POJO" encapsulant les propriétés d'une salle.

- **Attributs :** La classe définit le `nom`, l'`adresse`, la `capacite` (entier), ainsi que l'`heureOuverture` et l'`heureFermeture` (objets `LocalTime`).

- **Logique Métier :** La méthode `estOuverte()` encapsule la règle de gestion principale. Elle compare l'instant présent (`LocalTime.now()`) aux bornes horaires définies pour retourner un booléen.

2.2 La Source de Données (`SalleData.java`)

Ce composant agit comme une couche de persistance simulée (Mock Data).

- **Fonction :** Il fournit une méthode statique `getSalles()` qui instancie et retourne une `List<SalleDeSport>`.
- **Contenu :** La liste est pré-peuplée avec une vingtaine de salles réelles de Rabat, incluant des détails précis comme "Fitness Park Arribat Center" (6h-23h) ou "M-Fitness Agdal" (6h45-22h).

2.3 Le Contrôleur et la Vue (`Main.java`)

Ce fichier orchestre l'application. Il hérite de la classe `Application` de JavaFX.

- **Rôle :** Il initialise la fenêtre (`Stage`), configure la scène (`Scene`), place les composants graphiques et gère les événements (clics souris et mises à jour temporelles).

3. ANALYSE FONCTIONNELLE DÉTAILLÉE

3.1 Gestion du Temps et Statuts

La gestion du temps est critique pour ce projet. L'utilisation de l'API java.time.LocalTime permet une précision à la seconde.

Le statut d'une salle est déterminé par la condition suivante :

$$Maintenant > Ouverture \quad ET \quad Maintenant < Fermeture$$
Dans le code, cela se traduit par : maintenant.isAfter(heureOuverture) && maintenant.isBefore(heureFermeture).

3.2 Interface Graphique (GUI)

L'interface est construite autour d'un `BorderPane` divisé en deux zones principales :

1. **Zone Gauche (Liste) :** Une `ListView` affiche les salles. Une `CellFactory` personnalisée est utilisée pour concaténer le nom de la salle avec son statut textuel (">Ouverte" ou ">Fermée") directement dans la liste.
2. **Zone Centrale (Détails) :** Une `VBox` stylisée (CSS interne : `-fx-background-color: #f4f6f8`) affiche les informations détaillées de la salle sélectionnée (Nom, Adresse, Capacité, Horaires).

3.3 Interactivité

L'application utilise un écouteur (Listener) sur le modèle de sélection de la liste. Lorsqu'un utilisateur sélectionne une salle, la méthode `afficherInfosSalle(salle)` est déclenchée pour mettre à jour les labels d'information instantanément.

4. ASPECTS TECHNIQUES AVANCÉS

4.1 Rafraîchissement Automatique (Multithreading)

Pour garantir que l'heure affichée et le statut "Ouvert/Fermé" restent exacts sans intervention de l'utilisateur, l'application implémente un système de rafraîchissement automatique.

- **Mécanisme :** Un `Timer` planifie une `TimerTask` qui s'exécute toutes les 1000 millisecondes (1 seconde).
- **Gestion de la Concurrence :** Comme JavaFX ne permet pas de modifier l'interface (UI) depuis un thread secondaire (le Timer), le code utilise `Platform.runLater(() -> { ... })`. Cela place la mise à jour (`listView.refresh()` et `afficherInfosSalle`) dans la file d'attente du thread principal de l'application (JavaFX Application Thread).

4.2 Feedback Visuel

L'application améliore l'expérience utilisateur par des indicateurs visuels conditionnels :

- Si la salle est ouverte, le texte du statut s'affiche en **VERT** (`Color.GREEN`).
- Si la salle est fermée, le texte s'affiche en **ROUGE** (`Color.RED`).
- L'heure actuelle est affichée en bas de panneau avec un formatage précis `HH:mm:ss`.

5. CONCLUSION ET PERSPECTIVES

5.1 Bilan

L'application "Gestion des Salles de Sport - Rabat" est fonctionnelle et robuste. Elle respecte les principes de la programmation orientée objet et utilise efficacement les bibliothèques standard de Java (Collections, Time) ainsi que le framework JavaFX pour l'interface. La séparation entre `SalleData`, `SalleDeSport` et `Main` rend le code maintenable et lisible.

5.2 Améliorations Futures

Pour faire évoluer ce projet vers une solution professionnelle complète, les axes suivants sont recommandés :

1. **Base de Données :** Remplacer la classe statique `SalleData` par une connexion JDBC (MySQL ou SQLite) pour permettre l'ajout et la modification de salles par l'utilisateur.
2. **Recherche et Filtres :** Ajouter un champ de texte au-dessus de la liste pour filtrer les salles par nom ou par quartier (donnée présente dans l'adresse).
3. **Multimédia :** Intégrer un attribut "chemin image" dans la classe `SalleDeSport` pour afficher le logo de la salle dans le panneau de détails.