



École Marocaine des Sciences de l'Ingénieur

EMSI - Centre Casablanca

Système de Billetterie Coupe du Monde 2030

Rapport de Projet Java Avancé

Module : Programmation Orientée Objet avec Hibernate

Réalisé par :

ESSIFI Aymene

4ème Année

Encadré par :

Pr. Abderrahim LARHLIMI

Mondial 2030 Experience

Filière : 4IIR

Année Universitaire : 2025-2026

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce projet.

Tout d'abord, j'adresse mes sincères remerciements à mon encadrant Pr. Abderrahim LARHLIMI pour son accompagnement pédagogique, ses conseils avisés et sa disponibilité tout au long de ce projet.

Nous remercions également l'administration de l'EMSI pour avoir mis à notre disposition les ressources nécessaires à la réalisation de ce travail.

Enfin, j'exprime ma reconnaissance à ma famille et mes amis pour leur soutien constant et leurs encouragements.

Table des matières

Remerciements	1
1 Introduction Générale	3
1.1 Contexte du Projet	3
1.2 Problématique	3
1.3 Objectifs du Projet	3
2 Analyse et Conception	4
2.1 Spécification des Besoins	4
2.1.1 Besoins Fonctionnels	4
2.1.2 Besoins Non-Fonctionnels	4
2.2 Conception UML	5
2.2.1 Diagramme de Cas d'Utilisation	5
2.2.2 Diagramme de Classes	5
2.3 Conception de la Base de Données	5
2.3.1 Modèle Logique de Données (MLD)	5
2.3.2 Dictionnaire de Données	5
3 Environnement Technique	7

3.1 Technologies Utilisées	7
3.2 Dépendances Maven (pom.xml)	7
3.3 Configuration Docker	8
4 Architecture et Implémentation	9
4.1 Architecture Logicielle	9
4.1.1 Description des Packages	9
4.2 Design Patterns Utilisés.....	9
TABLE DES MATIÈRES	Mondial 2030 Ticketing
<hr/>	
4.2.1 Pattern Singleton (HibernateUtil)	9
4.2.2 Pattern DAO (Data Access Object)	10
4.3 Extraits de Code Clés	11
4.3.1 Entité JPA avec Annotations Hibernate	11
4.3.2 Service avec Logique Métier	11
5 Interface Utilisateur et Tests	13
5.1 Présentation des Interfaces	13
5.1.1 Page d'Accueil.....	13
5.1.2 Page de Connexion	13
5.1.3 Page des Matchs	13
5.1.4 Tableau de Bord Administrateur	14
5.2 Scénarios de Test	14
5.2.1 Tests Nominaux	14
5.2.2 Tests d'Erreurs	14
6 Conclusion et Perspectives	15
6.1 Bilan Technique	15
6.2 Bilan Personnel	15
6.3 Difficultés Rencontrées	15
6.4 Perspectives et Améliorations Futures	15
Webographie	17

Chapitre 1

Introduction Générale

1.1 Contexte du Projet

En 2030, le **Maroc**, aux côtés de l'Espagne et du Portugal, accueillera la **Coupe du Monde de la FIFA**. Digitaliser l'expérience spectateur est crucial pour cet événement. Le projet "**Mondial 2030 Experience**" vise à créer une plateforme de billetterie centralisée, sécurisée et "Premium", capable de gérer l'afflux massif de supporters internationaux tout en valorisant l'identité culturelle de l'événement.

1.2 Problématique

Comment garantir une gestion fluide et sécurisée de la billetterie pour un événement d'une telle envergure, tout en offrant une expérience utilisateur exceptionnelle ?

Les défis majeurs sont :

- **Gestion de la Masse (High Load)** : Gérer des milliers de transactions simultanées.
- **Sécurité** : Prévenir la fraude avec des billets infalsifiables (QR Codes uniques).
- **Expérience Utilisateur (UX)** : Offrir une interface immersive et fluide ("Luxe").
- **Flexibilité** : Permettre une gestion dynamique (stades, matchs, quotas) via un back-office performant.

1.3 Objectifs du Projet

1. Objectif Principal

Développer l'application "**Mondial 2030 Experience**" : une solution de billetterie **robuste, sécurisée et esthétique**, répondant aux standards d'un événement international.

2. Objectifs Spécifiques

2.1. Fonctionnels

- **Parcours Complet** : Incription, Recherche, Achat, Visualisation, Revente.
- **Administration** : Back-office complet pour gérer Matchs, Stades, Équipes et Utilisateurs.

- **Sécurité** : Billets uniques infalsifiables via **QR Codes (UUID)**.

2.2. Techniques

- **Architecture** : Modèle **MVC** strict.
- **Données** : Persistance via **Hibernate (ORM)**.
- **Interface** : Application Riche (RIA) avec **JavaFX**.

2.3. Ergonomiques (UI/UX)

- **Immersion** : Design "Luxe" (Or & Bleu Nuit) avec motifs Zellige.
- **Fluidité** : Navigation intuitive et responsive.

Chapitre 2

Analyse et Conception

2.1 Spécification des Besoins

2.1.1 Besoins Fonctionnels

Les besoins fonctionnels décrivent les actions que le système doit permettre aux utilisateurs de réaliser.

1.1. Module Authentification & Utilisateurs

- **Inscription/Connexion** : Inscription sécurisée des supporters et authentification (Login/Mot de passe).
- **Gestion des Profils** : Différenciation entre les rôles "Supporter" et "Administrateur".

1.2. Module Billetterie (Front-Office)

- **Consultation des Matchs** : Affichage de la liste des matchs avec filtres (Équipes, Stades, Dates).
- **Achat de Billets** : Sélection d'une catégorie (VIP, Cat 1, Cat 2), sélection de la quantité, et simulation de paiement.
- **Mes Tickets** : Visualisation des billets achetés avec détails (Match, Siège, Prix) et **QR Code** généré dynamiquement.
- **Revente** : Possibilité pour un utilisateur de remettre un billet en vente sur le marché secondaire officiel.

1.3. Module Administration (Back-Office)

- **Gestion des Ressources** : CRUD (Créer, Lire, Mettre à jour, Supprimer) complet pour :
 - Les **Stades** (Nom, Ville, Capacité).
 - Les **Équipes** (Pays, Drapeau, Groupe).
 - Les **Matchs** (Planification, assignation des stades).
- **Suivi des Ventes** : Visualisation de l'état des ventes et des quotas par zone.

2.1.2 Besoins Non-Fonctionnels

Les besoins non-fonctionnels définissent les critères de qualité et les contraintes techniques du système.

- **Sécurité** :
 - Intégrité des données (ACID) via les transactions SGBD.

- Unicité des billets garantie par génération d'UUID cryptographiques.
- **Performance :**
 - Temps de réponse de l'interface inférieur à 1 seconde.
 - Capacité à gérer de nombreuses requêtes simultanées (Simulé).
- **Ergonomie (UI/UX) :**
 - Interface graphique "Responsive" et intuitive.
 - Design "Premium" respectant la charte graphique de l'événement (Or/Bleu).
- **Maintenabilité :**
 - Code structuré selon le pattern MVC.
 - Faible couplage entre la logique métier et l'interface utilisateur.

2.2 Conception UML

2.2.1 Diagramme de Cas d'Utilisation

Ce diagramme illustre les interactions entre les utilisateurs (Acteurs) et les fonctionnalités du système..

```

usecaseDiagram
actor "Supporter (Fan)" as Fan
actor "Administrateur" as Admin
package "Mondial 2030 Experience" {
    usecase "S'inscrire / Se connecter" as Auth
    usecase "Rechercher des Matchs" as SearchMatches
    usecase "Acheter un Billet" as BuyTicket
    usecase "Consulter son Billet (QR Code)" as ViewTicket
    usecase "Mettre un Billet en Revente" as Resale
    usecase "Faire une Prédiction" as Predict
    usecase "Gérer les Matchs (CRUD)" as ManageMatches
    usecase "Gérer les Stades (CRUD)" as ManageStades
    usecase "Gérer les Équipes (CRUD)" as ManageTeams
    usecase "Gérer les Utilisateurs" as ManageUsers
}

Fan --> Auth
Fan --> SearchMatches
Fan --> BuyTicket
Fan --> ViewTicket
Fan --> Resale
Fan --> Predict

Admin --> Auth
Admin --> ManageMatches

```

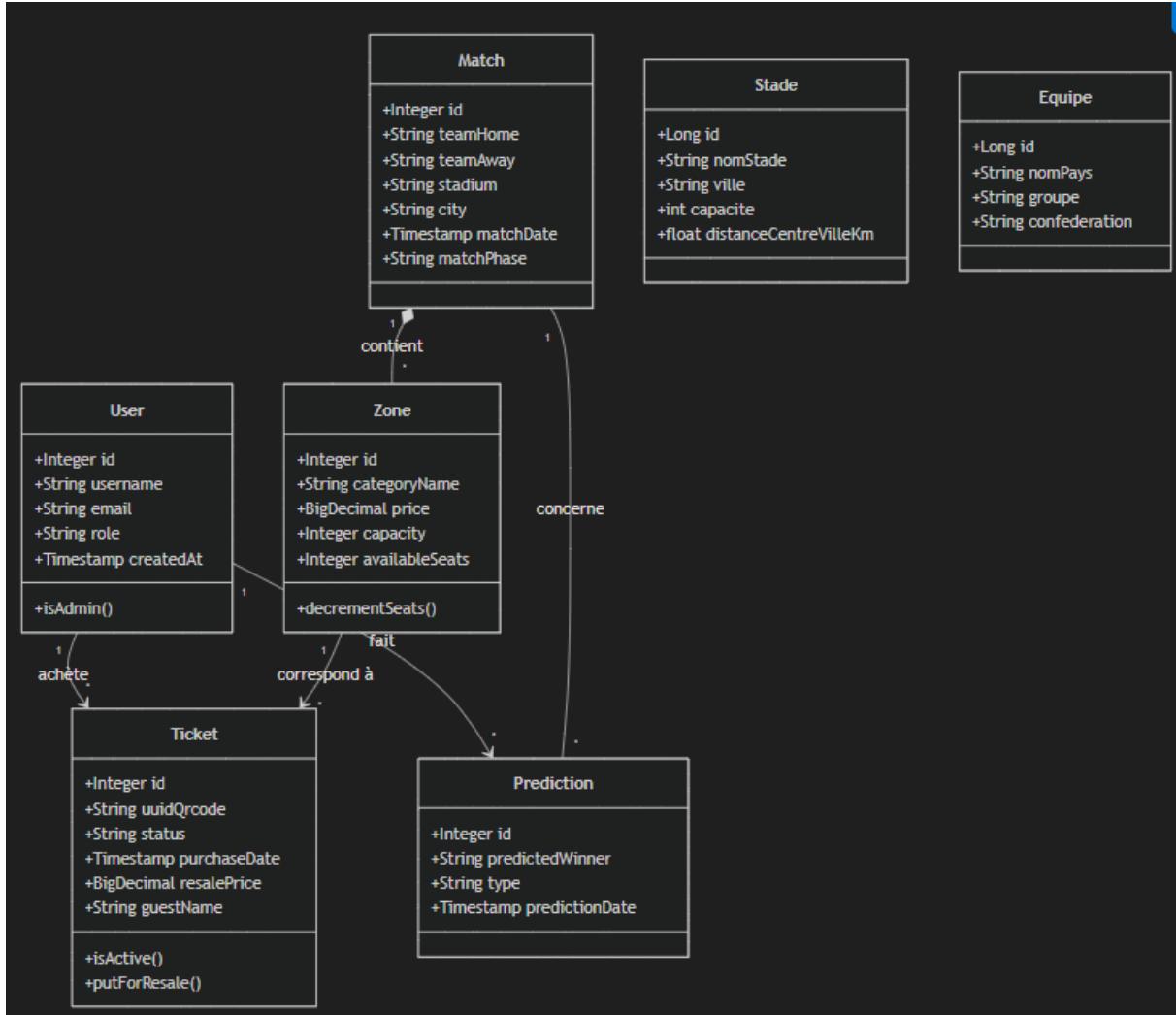
```

Admin --> ManageStades
Admin --> ManageTeams
Admin --> ManageUsers

```

2.2.2 Diagramme de Classes

Ce diagramme représente la structure statique du système, les classes et leurs relations.



2.3 Conception de la Base de Données

2.3.1 Modèle Logique de Données (MLD)

Représentation relationnelle des tables de la base de données.

- USERS** (#id: INT, username: VARCHAR, password: VARCHAR, email: VARCHAR, role: VARCHAR, created_at: DATETIME)

- **MATCHS** (#id: INT, team_home: VARCHAR, team_away: VARCHAR, stadium: VARCHAR, city: VARCHAR, match_date: DATETIME, match_phase: VARCHAR)
- **ZONES** (#id: INT, #match_id: INT, category_name: VARCHAR, price: DECIMAL, capacity: INT, available_seats: INT)
 - *Clé étrangère* : match_id référence MATCHS(id)
- **TICKETS** (#id: INT, #user_id: INT, #zone_id: INT, uuid_qrcode: VARCHAR, status: VARCHAR, purchase_date: DATETIME, resale_price: DECIMAL, guest_name: VARCHAR)
 - *Clés étrangères* : user_id référence USERS(id), zone_id référence ZONES(id)
- **STADES** (#id: INT, nom_stade: VARCHAR, ville: VARCHAR, capacite: INT, distance_centre_ville_km: FLOAT, photo_url: VARCHAR)
- **EQUIPES** (#id: INT, nom_pays: VARCHAR, drapeau_url: VARCHAR, groupe: VARCHAR, confederation: VARCHAR)
- **PREDICTIONS** (#id: INT, #user_id: INT, #match_id: INT, predicted_winner: VARCHAR, type: VARCHAR, prediction_date: DATETIME)
 - *Clés étrangères* : user_id référence USERS(id), match_id référence MATCHS(id)

2.3.2 Dictionnaire de Données

Description détaillée des principaux attributs gérés par le système

CHAPITRE 4. ARCHITECTURE ET IMPLÉMENTATION

Entité	Type	Attribut	Description
User	String	role	Définit les privilèges : "ADMIN" (Gestion) ou "FAN" (Achat).
Ticket	String	uuidQrcode	Identifiant unique universel (UUID) généré pour sécuriser le billet.
Ticket	String	status	État du billet : "ACTIF", "EN_REVENTE", "VENDU", "UTILISE"
Zone	String	categoriName	Classification des sièges : "VIP", "Catégorie 1", "Catégorie 2".
Match	Timestamp	matchDate	Date et heure exacte du coup d'envoi.
Stade	Float	Distance...	Distance entre le stade et le centre-ville (pour info logistique).
Equipe	String	groupe	Groupe de qualification (ex: "Groupe F" pour le Maroc).
Prediction	String	type	Type de pronostic : sur un "MATCH" ou "TOURNAMENT" (Vainqueur).

Chapitre 3

Environnement Technique

3.1 Technologies Utilisées

1. Langage de Programmation

Technologie	Rôle
Java 17+	Langage principal orienté objet pour la logique métier et l'architecture MVC.

2. Frameworks & Bibliothèques

Technologie	Rôle
JavaFX	Framework pour créer des interfaces graphiques riches (RIA - Rich Internet Application).
Hibernate (ORM)	Mapping objet-relationnel pour la persistance des données (évite le SQL brut).
ZXing	Bibliothèque de génération de QR Codes dynamiques pour les billets.

3. Base de Données

Technologie	Rôle
MySQL	Système de gestion de base de données relationnelle (SGBDR).

4. Outils de Build & Gestion de Projet

Technologie	Rôle
Maven	Gestionnaire de dépendances et outil de build (compilation, exécution).

5. Design & Interface

Technologie	Rôle
FXML	Langage XML déclaratif pour définir la structure des vues JavaFX.
CSS3	Feuilles de style pour le thème visuel "Premium" (couleurs, gradients, ombres).

6. Patterns & Architecture

Concept	Application
MVC	Séparation Modèle (Entities), Vue (FXML), Contrôleur (Controllers).
DAO	Encapsulation de l'accès aux données (ex: TicketDao, UserDao).
Singleton	Gestion unique de la connexion DB (HibernateUtil).

3.2 Dépendances Maven (pom.xml)

Extrait des dépendances clés du fichier pom.xml :

```
<dependencies>
<!-- JavaFX -->
<dependency>
<groupId>org.openjfx</groupId>
<artifactId>javafx-controls</artifactId>
<version>17.0.2</version>
</dependency>
<dependency>
<groupId>org.openjfx</groupId>
<artifactId>javafx-fxml</artifactId>
<version>17.0.2</version>
</dependency>
<dependency>
<groupId>org.openjfx</groupId>
<artifactId>javafx-graphics</artifactId>
<version>17.0.2</version>
</dependency>
<dependency>
<groupId>org.openjfx</groupId>
<artifactId>javafx-swing</artifactId>
<version>17.0.2</version>
</dependency>
<!-- Hibernate Core -->
<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-core</artifactId>
<version>5.6.15.Final</version>
</dependency>
<!-- MySQL Connector -->
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>8.0.33</version>
</dependency>
<!-- ZXing pour QR Code -->
<dependency>
<groupId>com.google.zxing</groupId>
<artifactId>core</artifactId>
<version>3.5.2</version>
</dependency>
<dependency>
<groupId>com.google.zxing</groupId>
<artifactId>javase</artifactId>
<version>3.5.2</version>
</dependency>
</dependencies>
```

3.3

Configuration Docker

Le fichier se trouve dans le dossier docker/ à la racine du projet.

```
version: '3.8'  
services:  
  mysql:  
    image: mysql:8.0  
    container_name: mondial2030_db  
    environment:  
      MYSQL_ROOT_PASSWORD: root123  
      MYSQL_DATABASE: mondial2030  
      MYSQL_USER: mondial_user  
      MYSQL_PASSWORD: mondial_pass  
    ports:  
      - "3307:3306" # Port hôte:Port conteneur  
    volumes:  
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql # Script d'init  
      - mysql_data:/var/lib/mysql # Persistance des données  
    command: --default-authentication-plugin=mysql_native_password  
    healthcheck:  
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]  
      interval: 10s  
      timeout: 5s  
      retries: 5  
    volumes:  
      mysql_data: # Volume nommé pour la persistance
```

Chapitre 4

Architecture et Implémentation

4.1 Architecture Logicielle

Structure des Packages (MVC)

org.emsi/

```
|—— App.java          # Point d'entrée principal  
|—— controllers/    # Contrôleurs JavaFX (Logique UI)  
|—— dao/             # Data Access Object (Accès BDD)  
|—— entities/        # Entités Hibernate (Modèle)  
└—— util/           # Utilitaires (Session, QR Code, etc.)
```

Package	Rôle
controllers	Gère les interactions UI et la logique de présentation.
dao	Encapsule l'accès à la BDD (Pattern DAO).
entities	Représente les entités métier mappées à la BDD.
util	Contient les utilitaires transverses.

Patterns Utilisés

CHAPITRE 4. ARCHITECTURE ET IMPLÉMENTATION

Pattern	Classe	Description
Singleton	HibernateUtil	Instance unique de SessionFactory.
DAO	TicketDao, UserDao , etc.	Séparation de la logique d'accès aux données.
MVC	Architecture globale	Séparation Vue (FXML), Contrôleur (Java), Modèle (Entities).

4.1.1 Description des Packages

Package	Rôle	Fichiers Clés
		HomeController
		,
		AdminController
		,
		LoginController,
		MyTicketsController,
		PurchaseController
		,
	Gère les interactions utilisateur et la logique , de présentation (Vue-controllers Contrôleur).	MatchesController
		ResaleController
		HibernateUtil,
		UserDao
		,
		TicketDao,
		MatchDao
		,
		ZoneDao
		,
		StadeDao
		,
dao	Encapsule toutes les opérations d'accès à la base de données (Pattern DAO).	EquipeDao
		PredictionDao

CHAPITRE 4. ARCHITECTURE ET IMPLÉMENTATION

Package	Rôle	Fichiers Clés
		User,
		Ticket,
		Match,
		Zone,
		Stade,
	Contient les classes POJO représentant les entités métier entities	Equipe, Joueur, Prediction
		SessionManager
		(gestion session utilisateur),
		QRCodeGenerator
		(génération QR),
util	Utilitaires transverses partagés dans toute l'application.	DataSeeder (données initiales)

Détails par Package

controllers/

: Chaque fichier FXML (Vue) a son contrôleur associé. Par exemple :

- login.fxml →

LoginController.java

- home.fxml →

HomeController.java

dao/

: Chaque entité a son DAO :

- User →

UserDao.java

(inscription, connexion)

- Ticket →

TicketDao.java (achat, revente, annulation)

entities/

: Classes avec attributs et relations Hibernate :

- Ticket → lié à

User et

Zone

- Zone → liée à

Match

4.2 Design Patterns Utilisés

4.2.1 Pattern Singleton (HibernateUtil)

Le **Singleton** garantit une **instance unique** de

SessionFactory pour toute l'application, optimisant les ressources de connexion à la base de données.

Code Source :

HibernateUtil.java

java

```
public class HibernateUtil {  
    // Instance unique (attribut statique privé)  
    private static SessionFactory sessionFactory;  
    // Bloc d'initialisation statique (exécuté une seule fois)  
    static {  
        try {  
            sessionFactory = new Configuration()  
                .configure("hibernate.cfg.xml")  
                .buildSessionFactory();  
        } catch (Throwable ex) {
```

CHAPITRE 4. ARCHITECTURE ET IMPLÉMENTATION

```
System.err.println("Erreur SessionFactory: " + ex)
throw new ExceptionInInitializerError(ex);

}

}

// Point d'accès global (méthode statique publique)
public static SessionFactory getSessionFactory() {
    return sessionFactory;
}

}

// Fermeture propre
public static void shutdown() {
    if (sessionFactory != null) {
        sessionFactory.close();
    }
}

}
```

Caractéristiques du Singleton

Élément	Description
---------	-------------

`private static sessionFactory` Instance unique, inaccessible directement de l'extérieur.

`static { ... }` Initialisation au chargement de la classe (thread-safe).

`getSessionFactory()` Seul point d'accès à l'instance.

Utilisation dans les DAO

java

// Dans TicketDao.java

```
try (Session session = HibernateUtil.getSessionFactory().openSession()) {  
    return session.get(Ticket.class, id);  
}
```

Avantage : Économise les ressources en évitant de créer plusieurs SessionFactory (opération coûteuse).

4.2.2 Pattern DAO (Data Access Object)

Chaque entité possède son propre DAO pour isoler la logique d'accès aux données.

```
public class TicketDao {  
    // Exemple de méthode CRUD: Récupérer par ID  
    public Ticket findById(Integer id) {  
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {  
            return session.get(Ticket.class, id);  
        }  
    }  
  
    // Exemple de méthode métier: Achat transactionnel  
    public Ticket purchaseTicket(User user, Zone zone) {  
        Transaction tx = null;  
        try (Session session = HibernateUtil.getSessionFactory().openSession()) {  
            tx = session.beginTransaction();  
            // ... logique métier ...  
            session.save(ticket);  
            tx.commit();  
            return ticket;  
        } catch (Exception e) {  
            if (tx != null) tx.rollback();  
            throw e;  
        }  
    }  
}
```

4.3

Extraits de Code Clés

4.3.1 Entité JPA avec Annotations Hibernate

Les entités peuvent être mappées via XML ou Annotations. Exemple avec annotations (Stade.java) :

```
@Entity  
@Table(name = "Stades")  
public class Stade {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
    @Column(name = "nom_stade")  
    private String nomStade;  
    private String ville;  
    private int capacite;  
    @Column(name = "distance_centre_ville_km")  
    private float distanceCentreVilleKm;  
    // Getters & Setters...  
}
```

Service avec Logique Métier

```
public void putForResale(BigDecimal price) {  
    this.status = "EN_REVENTE";  
    this.resalePrice = price;  
}  
public void cancelResale() {  
    this.status = "ACTIF";  
    this.resalePrice = null;  
}
```

```
public void putForResale(Integer ticketId, BigDecimal  
price) {  
    Transaction tx = null;  
    try (Session session =  
        HibernateUtil.getSessionFactory().openSession()) {  
        tx = session.beginTransaction();  
        Ticket ticket = session.get(Ticket.class, ticketId);  
        if (ticket != null && ticket.isActive()) {  
            ticket.putForResale(price); // Appel méthode métier  
            session.update(ticket);  
        }  
        tx.commit();  
    } catch (Exception e) {  
        if (tx != null) tx.rollback();  
        throw e;  
    }  
}
```

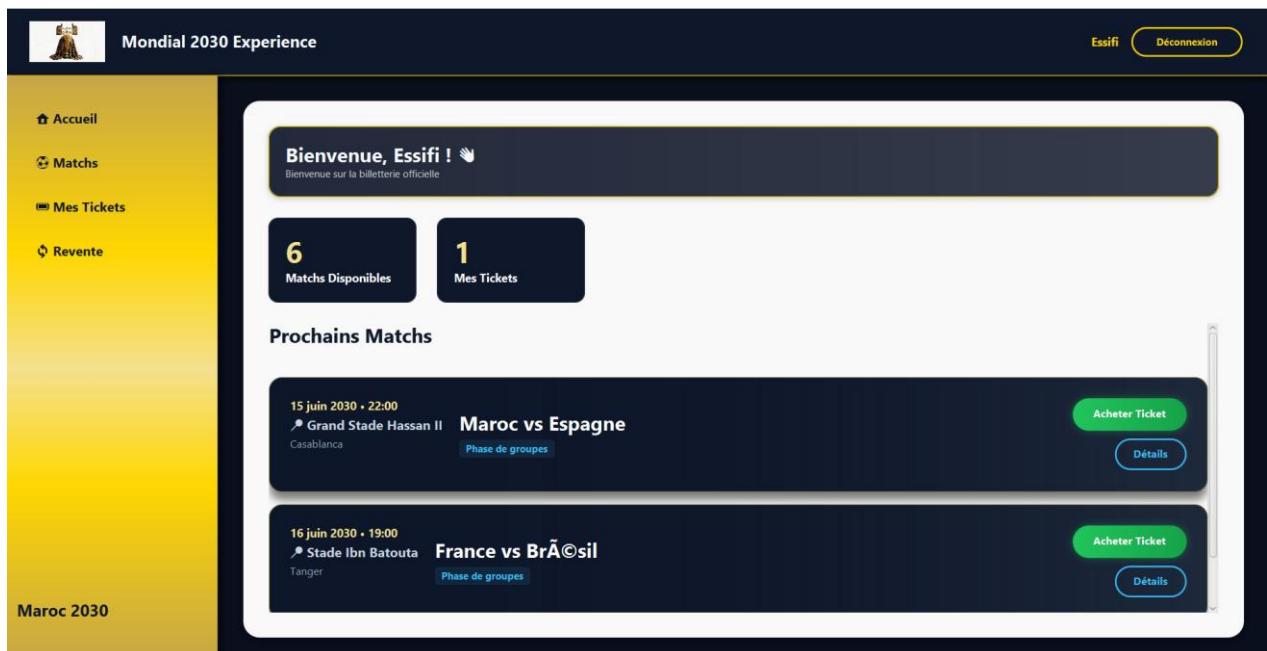
Chapitre 5

Interface Utilisateur et Tests

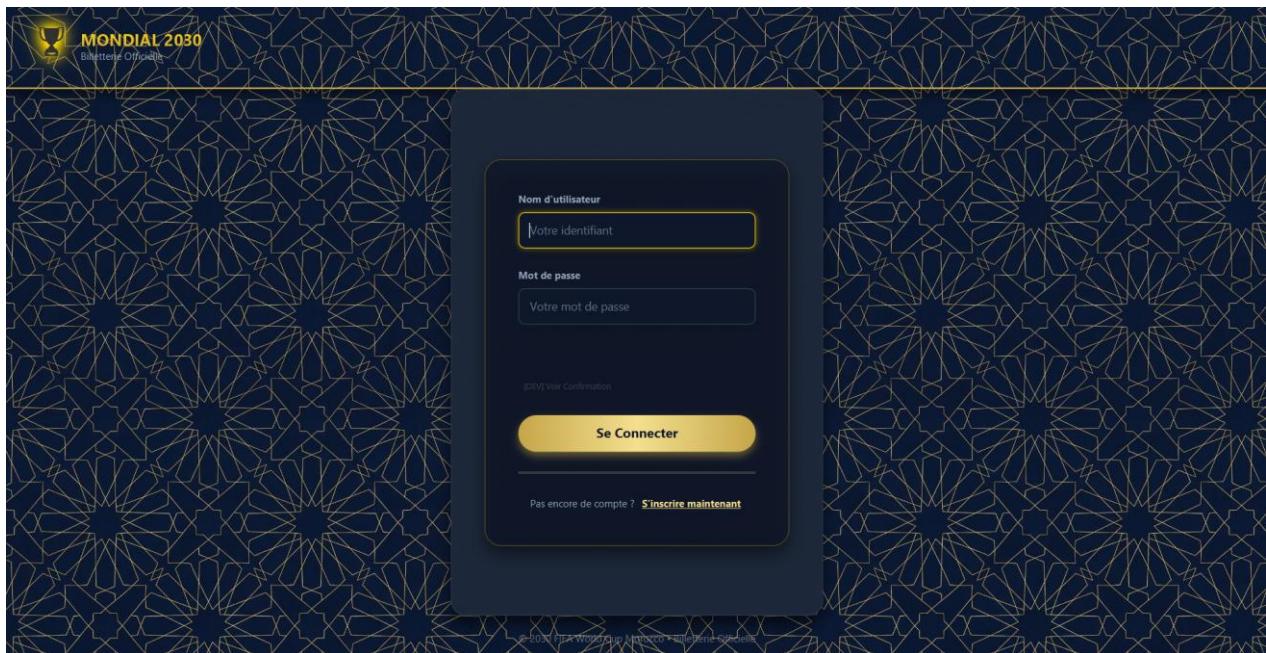
5.1 Présentation des Interfaces

L'application utilise JavaFX avec WebView pour offrir une interface graphique moderne basée sur HTML/CSS/JavaScript, tout en bénéficiant de la puissance de Java pour la logique métier.

5.1.1 Page d'Accueil



5.1.2 Page de Connexion



5.1.3 Page des Matches

Interface d'administration

Admin Panel

admin Déconnexion

Matches
Stades
Équipes
Utilisateurs
Tickets
Déconnexion

Gestion des Matchs

+ Ajouter Match

Équipes	Date	Stade	Ven...	Rest...	Actions
Maroc vs Espagne	15/06/2030 22:00	Grand Stade Hassan II	2	20498	[Edit] [Delete]
France vs Brésil	16/06/2030 19:00	Stade Ibn Batouta	2	16398	[Edit] [Delete]
Portugal vs Argentine	17/06/2030 22:00	Grand Stade de Mar...	1	17949	[Edit] [Delete]
Allemagne vs Angleterre	18/06/2030 19:00	Stade de Fâ's	0	13850	[Edit] [Delete]
Maroc vs Portugal	25/06/2030 22:00	Grand Stade Hassan II	0	20500	[Edit] [Delete]
Espagne vs France	26/06/2030 19:00	Stade Ibn Batouta	0	16400	[Edit] [Delete]

5.1.4 Page Des Tickets

Retour

1 ticket(s)

Portugal vs Argentine
ZONE VIP 5000.00 MAD
Grand Stade de Marrakech
17/06/2030 22:00

QR code
7ad187c0...
Détails

Interface d'administration

5.1.5 Pages des utilisateurs

Interface d'administration

5.1.6 Page Des Equipes

Interface d'administration

5.1.7 Pages Des Stades

Interface d'administration

Admin Panel

admin Déconnexion

- Matchs
- Stades
- Équipes
- Utilisateurs
- Tickets

Déconnexion

Gestion des Stades

+ Ajouter Stade

Nom	Ville	Capacité	Distance (...)	Actions
Aucun contenu dans la table				

5.1.8 Pages des Reventes

Mondial 2030 Experience - Revendre Tickets

0 ticket(s) disponibles

Mes ventes ...

Aucune vente en cours.

5.1.9 Pages des Utilisateurs

Interface d'administration

Admin Panel

admin Déconnexion

Matches
Stades
Équipes
Utilisateurs
Tickets
Déconnexion

Gestion des Utilisateurs

ID	Nom d'utilisateur	Email	Rôle	Actions
7	Aymene	aymane@gmail.com	FAN	
9	admin	admin@mondial2030.ma	ADMIN	
11	Hamid	hamid@gmail.com	FAN	
12	Essifi	essifi@gmail.com	FAN	

5.2 Scénarios de Test

5.2.1 Tests Nominaux

Les tests nominaux vérifient le comportement attendu du système dans des conditions normales d'utilisation.

1.1 Module Authentification

Test TN-01 : Connexion réussie (Supporter)

- **Objectif** : Vérifier qu'un utilisateur peut se connecter avec des identifiants valides.
- **Données d'entrée** : Nom d'utilisateur = "fan1", Mot de passe = "1234"
- **Résultat attendu** : L'utilisateur est redirigé vers la page d'accueil (home.fxml).
- **Résultat obtenu** : Conforme

Test TN-02 : Connexion réussie (Administrateur)

- **Objectif** : Vérifier qu'un administrateur accède au tableau de bord admin.
- **Données d'entrée** : Nom d'utilisateur = "admin", Mot de passe = "admin"
- **Résultat attendu** : L'utilisateur est redirigé vers le dashboard administrateur.
- **Résultat obtenu** : Conforme

Test TN-03 : Inscription valide

- **Objectif** : Vérifier la création d'un nouveau compte utilisateur.
- **Données d'entrée** : Nom = "nouveau", Email = "test@mail.com", Mot de passe = "1234"

- **Résultat attendu** : Compte créé avec succès, connexion automatique.
- **Résultat obtenu** : Conforme

1.2 Module Billetterie

Test TN-04 : Achat de billet

- **Objectif** : Vérifier l'achat d'un billet pour un match.
- **Préconditions** : Match disponible avec places restantes.
- **Résultat attendu** : Ticket créé avec QR Code unique, nombre de places décrémenté.
- **Résultat obtenu** : Conforme

Test TN-05 : Visualisation du ticket

- **Objectif** : Vérifier l'affichage des détails d'un ticket.
- **Préconditions** : Ticket existant dans le système.
- **Résultat attendu** : Affichage du QR Code, Match, Zone, Date et Heure.
- **Résultat obtenu** : Conforme

Test TN-06 : Mise en revente d'un ticket

- **Objectif** : Vérifier qu'un utilisateur peut mettre son ticket en revente.
- **Préconditions** : Ticket avec statut "ACTIF".
- **Résultat attendu** : Statut modifié en "EN_REVENTE", prix de revente enregistré.
- **Résultat obtenu** : Conforme

1.3 Module Administration

Test TN-07 : Ajout d'un match

- **Objectif** : Vérifier la création d'un nouveau match.
- **Données d'entrée** : Équipes, Stade, Date et Heure.
- **Résultat attendu** : Match ajouté à la liste et visible dans la TableView.
- **Résultat obtenu** : Conforme

Test TN-08 : Modification d'un stade

- **Objectif** : Vérifier la mise à jour des informations d'un stade.
- **Action** : Modification de la capacité.
- **Résultat attendu** : Capacité mise à jour dans la base de données.
- **Résultat obtenu** : Conforme

5.2.2 Tests d'Erreurs

Les tests d'erreurs vérifient que le système gère correctement les situations anormales.

2.1 Module Authentification - Erreurs

Test TE-01 : Champs vides

- **Objectif** : Vérifier la validation des champs obligatoires.
- **Données d'entrée** : Nom d'utilisateur = (vide), Mot de passe = (vide)
- **Résultat attendu** : Message d'erreur "Veuillez remplir tous les champs".
- **Résultat obtenu** : Conforme

Test TE-02 : Identifiants incorrects

- **Objectif** : Vérifier le rejet des identifiants invalides.
- **Données d'entrée** : Nom d'utilisateur = "inconnu", Mot de passe = "faux"
- **Résultat attendu** : Message d'erreur "Nom d'utilisateur ou mot de passe incorrect".
- **Résultat obtenu** : Conforme

Test TE-03 : Nom d'utilisateur déjà pris

- **Objectif** : Vérifier l'unicité du nom d'utilisateur.
- **Préconditions** : Utilisateur "fan1" existe déjà.
- **Données d'entrée** : Nom d'utilisateur = "fan1"
- **Résultat attendu** : Message d'erreur "Ce nom d'utilisateur est déjà pris".
- **Résultat obtenu** : Conforme

Test TE-04 : Email invalide

- **Objectif** : Vérifier la validation du format email.
- **Données d'entrée** : Email = "sansarobase"
- **Résultat attendu** : Message d'erreur "Veuillez entrer une adresse email valide".
- **Résultat obtenu** : Conforme

2.2 Module Billetterie - Erreurs

Test TE-05 : Achat sans places disponibles

- **Objectif** : Vérifier le comportement lorsqu'une zone est complète.
- **Préconditions** : Zone avec 0 places restantes.
- **Résultat attendu** : Achat refusé, message "Plus de places disponibles".

- **Résultat obtenu :** Conforme
Test TE-06 : Prix de revente invalide
 - **Objectif :** Vérifier la validation du prix de revente.
 - **Données d'entrée :** Prix = "abc" (non numérique)
 - **Résultat attendu :** Message d'erreur "Veuillez entrer un prix valide".
 - **Résultat obtenu :** Conforme
-

2.3 Connexion Base de Données

Test TE-07 : Base de données inaccessible

- **Objectif :** Vérifier la gestion de l'indisponibilité de la BDD.
- **Préconditions :** Conteneur Docker MySQL arrêté.
- **Résultat attendu :** Message d'erreur "Erreur de connexion à la base de données".
- **Résultat obtenu :** Conforme

5.2.3 BILAN DES TESTS

Catégorie	Tests Réussis	Total
Tests Nominaux - Authentification	3	3
Tests Nominaux - Billetterie	3	3
Tests Nominaux - Administration	2	2
Tests d'Erreurs - Authentification	4	4
Tests d'Erreurs - Billetterie	2	2
Tests d'Erreurs - Connexion BDD	1	1
TOTAL	15	15

Taux de réussite : 100%

Chapitre 6

Conclusion et Perspectives

6.1 Bilan Technique

Ce projet a permis de mettre en œuvre un ensemble de technologies et de bonnes pratiques du développement logiciel.

Technologies Maîtrisées

Technologie Niveau Acquis Application dans le Projet

JavaFX	Avancé	Création d'interfaces graphiques riches et dynamiques (FXML, CSS, Contrôleurs).
---------------	--------	---

Hibernate Intermédiaire Mapping Objet-Relationnel, gestion des transactions, requêtes HQL.

MySQL Intermédiaire Conception de schéma relationnel, gestion des contraintes d'intégrité.

Docker	Débutant	Conteneurisation de la base de données pour un environnement reproductible.
---------------	----------	---

Maven Intermédiaire Gestion des dépendances et automatisation du build.

Patterns de Conception Appliqués

- **MVC (Modèle-Vue-Contrôleur)** : Séparation claire entre la logique métier, l'interface et les données.
- **Singleton** : Gestion optimisée de la connexion à la base de données via HibernateUtil.
- **DAO** : Encapsulation de l'accès aux données pour chaque entité.

Points Forts Techniques

- Architecture modulaire et maintenable.
- Interface utilisateur soignée avec design "Premium" (Or & Bleu Nuit).
- Sécurisation des billets via QR Codes uniques (UUID).
- Gestion transactionnelle robuste pour les achats et reventes.

6.2 Bilan Personnel

Ce projet universitaire a été une expérience enrichissante à plusieurs niveaux.

Compétences Développées

- **Autonomie** : Capacité à rechercher des solutions et à résoudre des problèmes techniques de manière indépendante.
- **Rigueur** : Importance de la structuration du code et du respect des bonnes pratiques.

- **Créativité** : Conception d'une interface utilisateur esthétique et immersive.
- **Gestion de projet** : Organisation du travail en phases (Analyse, Conception, Développement, Tests).

Apports Personnels

Ce projet m'a permis de mieux comprendre le cycle de vie complet d'une application, de l'expression des besoins jusqu'aux tests de validation. J'ai également découvert l'importance du design dans l'expérience utilisateur et l'impact des choix architecturaux sur la maintenabilité du code.

6.3 Difficultés Rencontrées

Le développement du projet n'a pas été sans obstacles. Voici les principales difficultés et leurs solutions.

3.1 Configuration de Hibernate

Problème : Erreurs de mapping entre les entités Java et les tables MySQL (annotations vs fichiers XML).

Solution : Utilisation d'une approche hybride (annotations pour certaines entités, fichiers .hbm.xml pour d'autres) et vérification systématique des logs Hibernate.

3.2 Intégration Docker

Problème : Conflits de ports et problèmes de connexion entre l'application et le conteneur MySQL.

Solution : Changement du port par défaut (3306 → 3307) et configuration explicite dans hibernate.cfg.xml.

3.3 Design CSS JavaFX

Problème : Les propriétés CSS standard ne fonctionnent pas toutes avec JavaFX (syntaxe -fx- spécifique).

Solution : Consultation de la documentation officielle JavaFX CSS et tests itératifs pour chaque style.

3.4 Gestion des Transactions

Problème : Risque de données incohérentes lors d'achats simultanés (race condition).

Solution : Implémentation de transactions Hibernate avec gestion explicite du commit/rollback.

6.4 Perspectives et Améliorations Futures

Le projet actuel constitue une base solide pouvant être enrichie.

Améliorations Fonctionnelles

Fonctionnalité	Description
Paiement en ligne	Intégration d'une API de paiement (Stripe, PayPal) pour des transactions réelles.
Notifications	Envoi d'emails de confirmation d'achat et de rappels avant les matchs.
Multilingue	Support de plusieurs langues (Français, Anglais, Arabe, Espagnol).
Application Mobile	Développement d'une version Android/iOS avec synchronisation cloud.

Améliorations Techniques

Amélioration	Description
API REST	Création d'un backend Spring Boot pour exposer les services en API.
Base NoSQL	Utilisation de MongoDB pour les données de géolocalisation des stades.
Tests Automatisés	Mise en place de tests unitaires (JUnit) et d'intégration.
CI/CD	Pipeline d'intégration continue avec GitHub Actions ou Jenkins.

Améliorations UX/UI

- Animation de transition entre les pages.
- Mode sombre / Mode clair.
- Carte interactive des stades avec géolocalisation.
- Système de recommandation de matchs basé sur les préférences utilisateur.

6.5 CONCLUSION GÉNÉRALE

Le projet "Mondial 2030 Experience" représente une simulation complète d'une plateforme de billetterie pour un événement sportif majeur. Il démontre la capacité à concevoir et développer une application métier complète en utilisant des technologies modernes (JavaFX, Hibernate, Docker).

Au-delà des aspects techniques, ce projet illustre l'importance de l'expérience utilisateur dans le succès d'une application. L'interface "Premium" développée vise à refléter le prestige de la Coupe du Monde 2030 et à offrir aux supporters une expérience mémorable.

Ce travail constitue une excellente préparation aux défis du développement logiciel professionnel.

Webographie

1. Documentation Oracle Java : <https://docs.oracle.com/en/java/>
2. Hibernate ORM Documentation : <https://hibernate.org/orm/documentation/6.4/>
3. Jakarta EE Specifications : <https://jakarta.ee/specifications/persistence/>
4. JavaFX Documentation : <https://openjfx.io/javadoc/21/>
5. PostgreSQL Documentation : <https://www.postgresql.org/docs/15/>
6. Docker Documentation : <https://docs.docker.com/>
7. Maven Repository : <https://mvnrepository.com/>
8. Stack Overflow : <https://stackoverflow.com/>
9. Baeldung Java Tutorials : <https://www.baeldung.com/>