DOCUMENTATION TECHNIQUE

Système de Gestion du Personnel des Ligues

Version 2.0

Mars 2026

Projet Java — Architecture MVC + JDBC + Sérialisation

Sommaire

1. Présentation du projet

1.1 Contexte et objectif

Le projet Gestion du Personnel des Ligues est une application Java conçue pour gérer les employés de différentes ligues sportives hébergées par la M2L (Maison de la Ligue). Elle permet de créer, modifier, supprimer et consulter des ligues et leurs employés via une interface en ligne de commande.

1.2 Fonctionnalités principales

- Gestion des ligues : création, renommage, suppression
- Gestion des employés : ajout, modification (nom, prénom, mail, mot de passe, dates), suppression
- Gestion des rôles : root (super-utilisateur), administrateur de ligue, employé simple
- Persistance des données : via JDBC (MySQL) ou sérialisation Java
- Contrôle des dates d'arrivée et de départ avec validation métier

1.3 Technologies utilisées

Technologie	Version	Usage
Java	1.8+	Langage principal
Maven	3.x	Gestion des dépendances
MySQL + JDBC	8.0.16	Persistance base de données
JUnit Jupiter	5.8.2	Tests unitaires
CommandLineMenus	2.0	Interface console

2. Architecture du projet

2.1 Structure des packages

Le projet est organisé en quatre packages distincts, chacun ayant une responsabilité claire selon le principe de séparation des préoccupations :

Package	Rôle
personnel	Couche métier : classes Employe, Ligue, GestionPersonnel, exceptions
commandLine	Couche présentation : menus console (PersonnelConsole, LigueConsole, EmployeConsole)
jdbc	Couche persistance JDBC : connexion MySQL, opérations CRUD
serialisation	Couche persistance alternative : sérialisation Java binaire
testsUnitaires	Tests JUnit 5 : testLigue, testEmploye

2.2 Patron de conception

L'application applique un patron Passerelle (Gateway) pour découpler la logique métier de la couche de persistance. L'interface Passerelle définit le contrat CRUD, et deux implémentations concrètes sont disponibles :

- jdbc.JDBC — connexion MySQL via JDBC, activée par défaut (TYPE_PASSERELLE = JDBC)
- serialisation.Serialization — sauvegarde binaire Java, pour les environnements sans base de données

Le basculement entre les deux se fait par une simple constante dans GestionPersonnel :

```
public final static int SERIALIZATION = 1, JDBC = 2, TYPE_PASSERELLE = JDBC;
```

2.3 Diagramme de flux simplifié

Le flux d'exécution suit le chemin suivant : l'interface console capture les actions de l'utilisateur, les délègue à la couche métier (GestionPersonnel / Ligue / Employe), qui interagit avec la passerelle de persistance choisie.

3. Modèle de données

3.1 Schéma relationnel

La base de données est composée de deux tables : LIGUE et EMPLOYE. Un employé appartient à au plus une ligue (cardinalité 0,1 côté EMPLOYE, 0,n côté LIGUE), ce qui correspond au diagramme Entité-Association fourni.

Table LIGUE

Colonne	Type SQL	Contrainte	Description
id_ligue	INT	PK, AUTO_INCREMENT	Identifiant unique
nom_ligue	VARCHAR(100)	NOT NULL	Nom de la ligue

Table EMPLOYE

Colonne	Type SQL	Contrainte	Description
id_employe	INT	PK, AUTO_INCREMENT	Identifiant unique
nom_employe	VARCHAR(100)	NOT NULL	Nom de l'employé
prenom_employe	VARCHAR(100)	NOT NULL	Prénom de l'employé
mail_employe	VARCHAR(150)	NOT NULL	Adresse e-mail
password_employe	VARCHAR(255)	NOT NULL	Mot de passe (non haché)
date_arrivee_employe	DATE	NULLABLE	Date d'arrivée (≤ aujourd'hui)
date_départ_employe	VARCHAR / DATE	NULLABLE	Date de départ (≥ date arrivée)
id_ligue	INT	FK → LIGUE, NULLABLE	NULL si root, sinon ligue de l'employé
rôle_employe	VARCHAR(20)	NOT NULL	'admin' ou 'employe'

3.2 Script SQL de création

```sql
CREATE TABLE LIGUE (
  id_ligue  INT           AUTO_INCREMENT PRIMARY KEY,
  nom_ligue VARCHAR(100)  NOT NULL
);

CREATE TABLE EMPLOYE (
  id_employe          INT          AUTO_INCREMENT PRIMARY KEY,
  nom_employe         VARCHAR(100) NOT NULL,
  prenom_employe      VARCHAR(100) NOT NULL,
  mail_employe        VARCHAR(150) NOT NULL,
  password_employe    VARCHAR(255) NOT NULL,
  date_arrivee_employe DATE,
```

```
  date_départ_employe  VARCHAR(20),
  id_ligue             INT REFERENCES LIGUE(id_ligue),
  rôle_employe         VARCHAR(20) NOT NULL DEFAULT 'employe'
);
```

4. Description des classes

4.1 Classe GestionPersonnel

Classe centrale du système (singleton). Elle orchestre l'ensemble des opérations sur les ligues et employe les délègue à la passerelle de persistance. Elle ne peut être instanciée qu'une seule fois grâce au contrôle dans le constructeur.

Méthode	Retour	Description
getGestionPersonnel()	GestionPersonnel	Retourne l'unique instance (pattern Singleton)
addLigue(String nom)	Ligue	Crée une ligue et l'enregistre en base
addLigue(int id, String nom)	Ligue	Recréé une ligue depuis la BDD (chargement)
getLigues()	SortedSet<Ligue>	Retourne toutes les ligues (lecture seule)
getRoot()	Employe	Retourne le super-utilisateur root
addRoot(String, String)	void	Crée le root (nom + password)
sauvegarder()	void	Délègue la sauvegarde à la passerelle
delete(Employe)	void	Supprime un employé via la passerelle

4.2 Classe Ligue

Représente une ligue sportive. Contient un ensemble trié d'employés et un administrateur. Par défaut, c'est le root qui est administrateur jusqu'à affectation explicite.

Méthode	Retour	Description
getNom() / setNom()	String / void	Lecture/écriture du nom, persisté en base
getId()	int	Identifiant en base de données
getEmployes()	SortedSet<Employe>	Liste triée des employés (lecture seule)
addEmploye(...)	Employe	Crée et ajoute un employé à la ligue
setAdministrateur(Employe)	void	Change l'admin et met à jour les rôles en BDD
setAdministrateurFromJDBC()	void	Affecte l'admin au chargement sans appel BDD
remove()	void	Supprime la ligue de GestionPersonnel

4.3 Classe Employe

Représente un employé d'une ligue. Instanciable uniquement via Ligue.addEmploye(). Contient la logique de validation des dates et les règles de suppression (protection du root).

Méthode	Retour	Description
setDateArrivee(LocalDate)	void	Lève DateInvalide si date > aujourd'hui
setDateDepart(LocalDate)	void	Lève DateInvalide si date < dateArrivee
estAdmin(Ligue)	boolean	Vrai si this est l'admin de la ligue donnée
estRoot()	boolean	Vrai si this est le root
checkPassword(String)	boolean	Compare le mot de passe fourni
remove()	void	Supprime l'employé ; lève ImpossibleDeSupprimerRoot si root

4.4 Interface Passerelle

Contrat d'accès aux données. Toute implémentation de persistance doit implémenter ces sept méthodes :

Méthode	Description
getGestionPersonnel()	Charge et retourne l'état complet depuis le support
sauvegarderGestionPersonnel()	Sauvegarde globale (utilisée par la sérialisation)
insert(Ligue) / insert(Employe)	Insère et retourne l'id généré
update(Ligue) / update(Employe)	Met à jour un enregistrement existant
delete(Employe)	Supprime un employé du support

5. Couche JDBC

5.1 Connexion à la base de données

La connexion est établie dans le constructeur de la classe JDBC, via le fichier Credentials.java (non versionné pour des raisons de sécurité). Un fichier CredentialsExample.java est fourni comme modèle.

Paramètre	Valeur par défaut
driver	mysql
driverClassName	com.mysql.cj.jdbc.Driver
host	localhost
port	3306
database	(à compléter)

5.2 Chargement des données (getGestionPersonnel)

Le chargement se fait en trois phases distinctes :

- Phase 1 : Sélection de toutes les ligues (SELECT * FROM LIGUE) et création des objets Ligue correspondants.
- Phase 2 : Pour chaque ligue, chargement de ses employés par requête paramétrée (SELECT * FROM EMPLOYE WHERE id_ligue = ?). Si le rôle est 'admin', l'employé est désigné administrateur via setAdministrateurFromJDBC().
- Phase 3 : Chargement du root (SELECT * FROM EMPLOYE WHERE id_ligue IS NULL LIMIT 1). Si absent, un root par défaut est créé.

5.3 Opérations d'écriture

Les insertions utilisent Statement.RETURN_GENERATED_KEYS pour récupérer l'identifiant auto-généré et le propager dans les objets Java. Le rôle (admin / employe) est déterminé dynamiquement à chaque insert ou update selon l'état courant de l'objet.

Opération	Comportement notable
insert(Employe)	id_ligue = NULL si root ; rôle calculé dynamiquement
insert(Ligue)	Retourne l'id généré pour l'affecter à l'objet Java
update(Employe)	Recalcule le rôle ; met à jour l'ancien et le nouvel admin lors d'un changement
update(Ligue)	Met à jour le nom uniquement
delete(Employe)	Suppression par id_employe

6. Règles métier et validations

6.1 Gestion des dates

Règle	Exception levée	Message
Date d'arrivée > aujourd'hui	DateInvalide	«La date d'arrivée ne peut pas être au plus tard que la date actuelle.»
Date de départ < date d'arrivée	DateInvalide	«La date de départ ne peut pas être avant la date d'arrivée»
Dates nulles	Aucune	Autorisé (champs optionnels)

6.2 Gestion des rôles

- Root : employé sans ligue (id_ligue = NULL). Il ne peut pas être supprimé (ImpossibleDeSupprimerRoot). Il est l'administrateur par défaut de toute nouvelle ligue.
- Administrateur : employé d'une ligue désigné via setAdministrateur(). Un seul admin par ligue. Le changement d'admin met à jour le rôle de l'ancien admin en base.
- Employe simple : tout employé non root et non désigné administrateur.

6.3 Règles de suppression

- Suppression d'un employé admin : le root récupère automatiquement l'administration de la ligue avant suppression.
- Suppression du root : impossible, lève ImpossibleDeSupprimerRoot (RuntimeException).
- Suppression d'une ligue : retire la ligue de GestionPersonnel. Attention : les employés ne sont pas supprimés automatiquement en base dans l'implémentation actuelle.

7. Tests unitaires

7.1 Cadre de test

Les tests utilisent JUnit Jupiter 5.8.2. Deux classes de test couvrent les entités principales du domaine métier.

7.2 Classe testLigue

Méthode de test	Ce qui est vérifié
createLigue()	La ligue créée porte bien le nom fourni
addEmploye()	L'employé est bien le premier de la liste triée
ExceptionDateInvalideDA()	DateInvalide levée si date arrivée > aujourd'hui
ExceptionDateInvalideDD()	DateInvalide levée si date départ < date arrivée
TestsetNom() / TestgetNom()	Lecture et modification du nom de ligue
TestChangementAdmin()	setAdministrateur() + getAdministrateur() cohérents
TestRemoveLigue()	Ligue absente de getLigues() après suppression
TestRemoveAdmin()	Suppression admin : absent de getEmployes()

7.3 Classe testEmploye

Méthode de test	Ce qui est vérifié
TestgetNom() / TestsetNom()	Getter et setter du nom
TestgetPrenom() / TestsetPrenom()	Getter et setter du prénom
TestgetMail() / TestsetMail()	Getter et setter du mail
TestsetPassword()	checkPassword() retourne true après setPassword()
TestgetLigue()	getLigue() retourne la bonne ligue d'appartenance
TestgetDateArrivee() / TestsetDateArrivee()	Getter et setter de la date d'arrivée
TestgetDateDepart() / TestsetDateDepart()	Getter et setter de la date de départ
TestRemoveEmploye()	Employé absent de getEmployes() après remove()

8. Points d'amélioration identifiés

8.1 Problème identifié dans EmployeConsole

Dans la méthode changerDateDepart() de la classe EmployeConsole, un bug est présent : la méthode appelle setDateArrivee() au lieu de setDateDepart(). Ce qui signifie que la modification de la date de départ écrase en réalité la date d'arrivée.

Code actuel (bugué)	Correction attendue
`employe.setDateArrivee(LocalDate.parse(strdd));`	`employe.setDateDepart(LocalDate.parse(strdd));`

8.2 Autres améliorations recommandées

- Hashage des mots de passe : les mots de passe sont actuellement stockés en clair en base de données. L'utilisation de BCrypt ou SHA-256 est fortement recommandée.
- Suppression en cascade : la suppression d'une ligue ne supprime pas ses employés en base. Une contrainte ON DELETE CASCADE ou une gestion explicite est nécessaire.
- Colonne date_départ_employe : le type est VARCHAR au lieu de DATE, ce qui nuit à la cohérence et aux comparaisons SQL. Une migration de type est recommandée.
- Couverture de tests : les tests ne couvrent pas les cas d'erreur JDBC ni les scénarios de concurrence. Des tests d'intégration avec une base H2 in-memory pourraient être ajoutés.

9. Guide d'installation rapide

9.1 Prérequis

- Java 1.8 ou supérieur
- Maven 3.x
- MySQL 5.7+ ou 8.x avec un utilisateur ayant les droits sur la base cible

9.2 Configuration

- Copier Personnel/src/jdbc/CredentialsExample.java vers Personnel/src/jdbc/Credentials.java
- Renseigner host, port, database, user et password dans le fichier Credentials.java
- Exécuter le script SQL de création des tables (cf. section 3.2)

9.3 Compilation et exécution

```
# Compilation
cd Personnel && mvn compile

# Lancement
mvn exec:java -Dexec.mainClass="commandLine.PersonnelConsole"

# Tests
mvn test
```