Samir Es safi

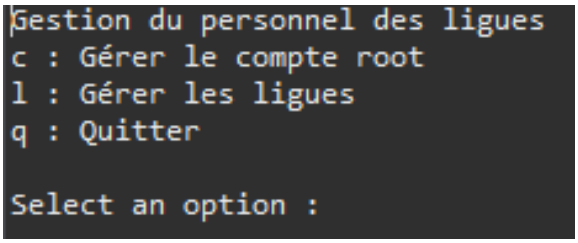
Documentation Projet Personnel

Sommaire :

Explication :

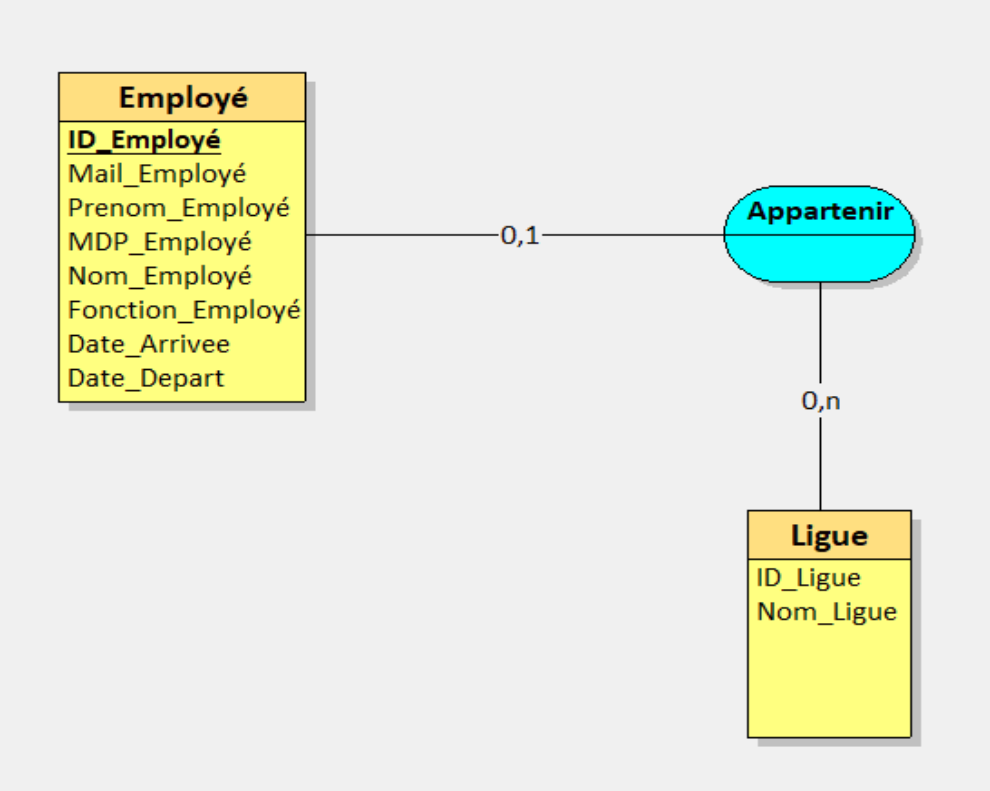
Cette application permet de gérer des ligues et des employés via une interface en ligne de commande. Elle est conçue pour permettre à un administrateur de créer et de gérer des ligues,
ajouter, de modifier et de supprimer des employés dans ces ligues, ainsi que de gérer l'administrateur de chaque ligue.

Voici la vue depuis le terminal :



1- Création de la base de données (mcd, script)

Le MCD :

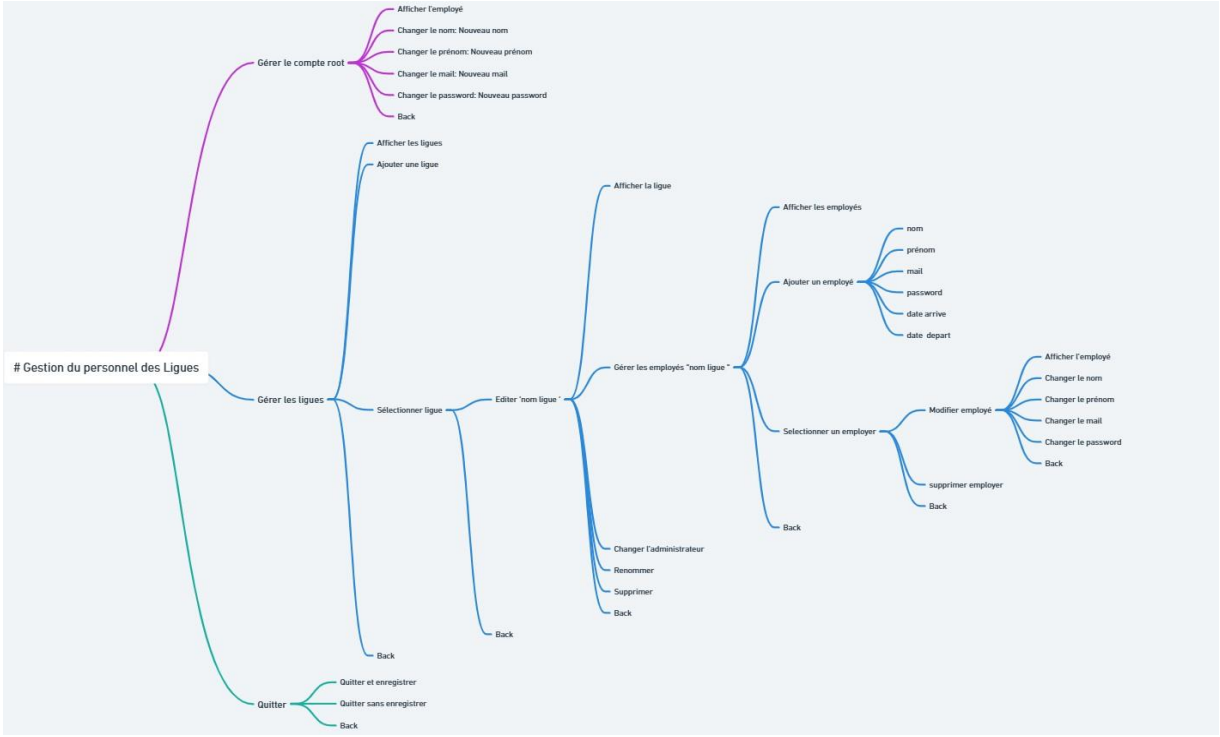


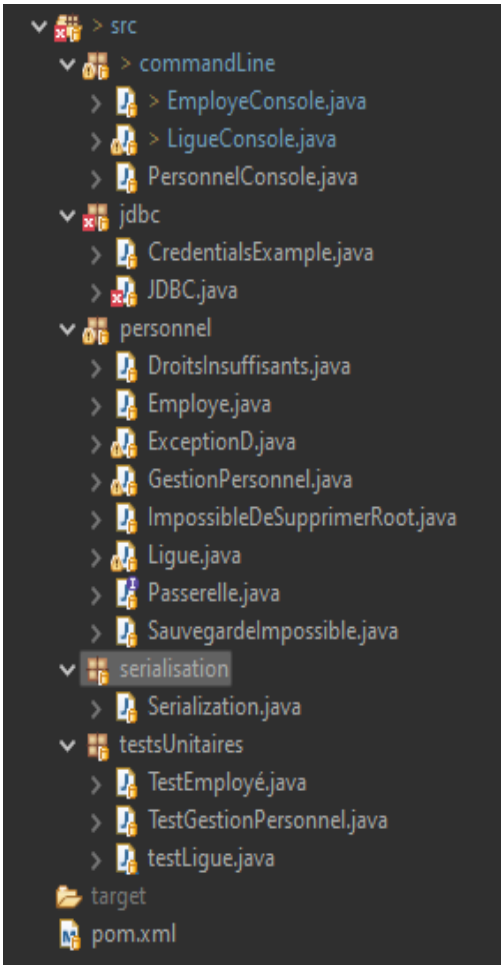
Voici le script :

```
CREATE TABLE Ligue (
    ID_Ligue INT ,
    Nom_Ligue VARCHAR(50),
    PRIMARY KEY(ID_Ligue)
);


CREATE TABLE Employé (
    ID_Employé INT ,
    Nom_Employé VARCHAR(20),
    Prenom_Employé VARCHAR(20),
    Mail_Employé VARCHAR(50),
    MDP_Employé VARCHAR(50),
    Fonction_Employé VARCHAR(50),
    Date_Depart DATE,
    Date_Arrivee DATE,
    ID_Ligue INT,
    PRIMARY KEY(ID_Employé),
    FOREIGN KEY(ID_Ligue) REFERENCES Ligue(ID_Ligue)
);
```

2- Arbre heuristique





Voici la vue des différentes classes

```java
public class Employe implements Serializable, Comparable<Employe>
{
    private static final long serialVersionUID = 4795721718037994734L;
    private String nom, prenom, password, mail;
    private Ligue ligue;
    private GestionPersonnel gestionPersonnel;
    private LocalDate dateArrive;
    private LocalDate dateDepart;
    private int id;

    Employe(GestionPersonnel gestionPersonnel, Ligue ligue, String nom, String prenom, String mail, String password, LocalDate dateArrive, LocalDate dateDepart)
            throws ExceptionD
    {
        this.gestionPersonnel = gestionPersonnel;
        this.nom = nom;
        this.prenom = prenom;
        this.password = password;
        this.mail = mail;
        this.ligue = ligue;

        if (dateArrive == null || dateDepart == null || dateDepart.isBefore(dateArrive) ) {
            throw new ExceptionD();
        }

        this.dateArrive = dateArrive;
        this.dateDepart = dateDepart;
    }

    Employe(GestionPersonnel gestionPersonnel, Ligue ligue, String nom, String prenom, String mail, String password)
```

```java
public LocalDate getDateArrivee()
{
    return this.dateArrive;
}

/**
 * Retourne la date d'arrivée de l'employé.
 * @return la date d'arrivée de l'employé.
 */

public void setDateArrivee(LocalDate dateArrive)
        throws ExceptionD
{
    if (dateArrive == null  || dateDepart.isBefore(dateArrive)) {
        throw new ExceptionD();
    }
    else {

        this.dateArrive = dateArrive;}
}

/**
 * Modifie la date de départ de l'employé.
 * @return la date de départ de l'employé.
 */

public LocalDate getDateDepart()
{
    return this.dateDepart;
}

/**
 * Modifie la date de départ de l'employé.
 * @return la date de départ de l'employé.
 */

public void setDateDepart(LocalDate dateDepart)
        throws ExceptionD
{
    if (dateDepart == null || dateDepart.isBefore(dateArrive)) {
        throw new ExceptionD();
    }
    else {
        this.dateDepart = dateDepart;}
}
```

Constructeur de la classe employé dans laquelle nous avons rajouté les dates d'arrivée et de départ

Ajout des setter et getter pour les dates ainsi que gestion des exceptions si la saisie n'est pas conforme