

Pour ce projet, nous avons récupéré depuis Discord une application java permettant de gérer l'organisation de ligues et d'employés en ligne de commande.

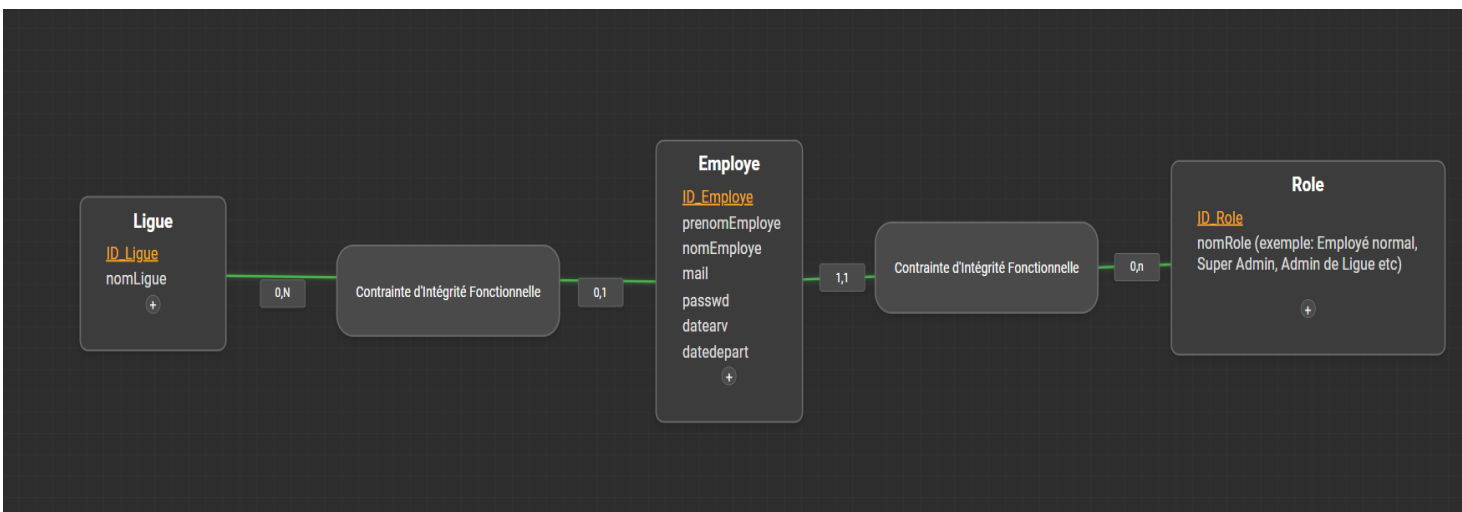
Actuellement dans l'itération 2, nous avons ici eu pour objectif d'ajouter les améliorations suivantes :

- Un MCD
- La création des tables sql
- Un arbre heuristique
- Le changement de l'administrateur en ligne de commande
- La saisie des dates en ligne de commande
- Le portfolio

Pour se faire nous avons utilisé GitHub comme environnement de travail ainsi qu'Eclipse, Looping, Canva et l'IA comme outils de travail.

Partie 1 - Base de données :

Dans un premier temps, nous avons commencé par imaginer la structure de la base donnée ce qui nous a permis de créer le [MCD](#) :



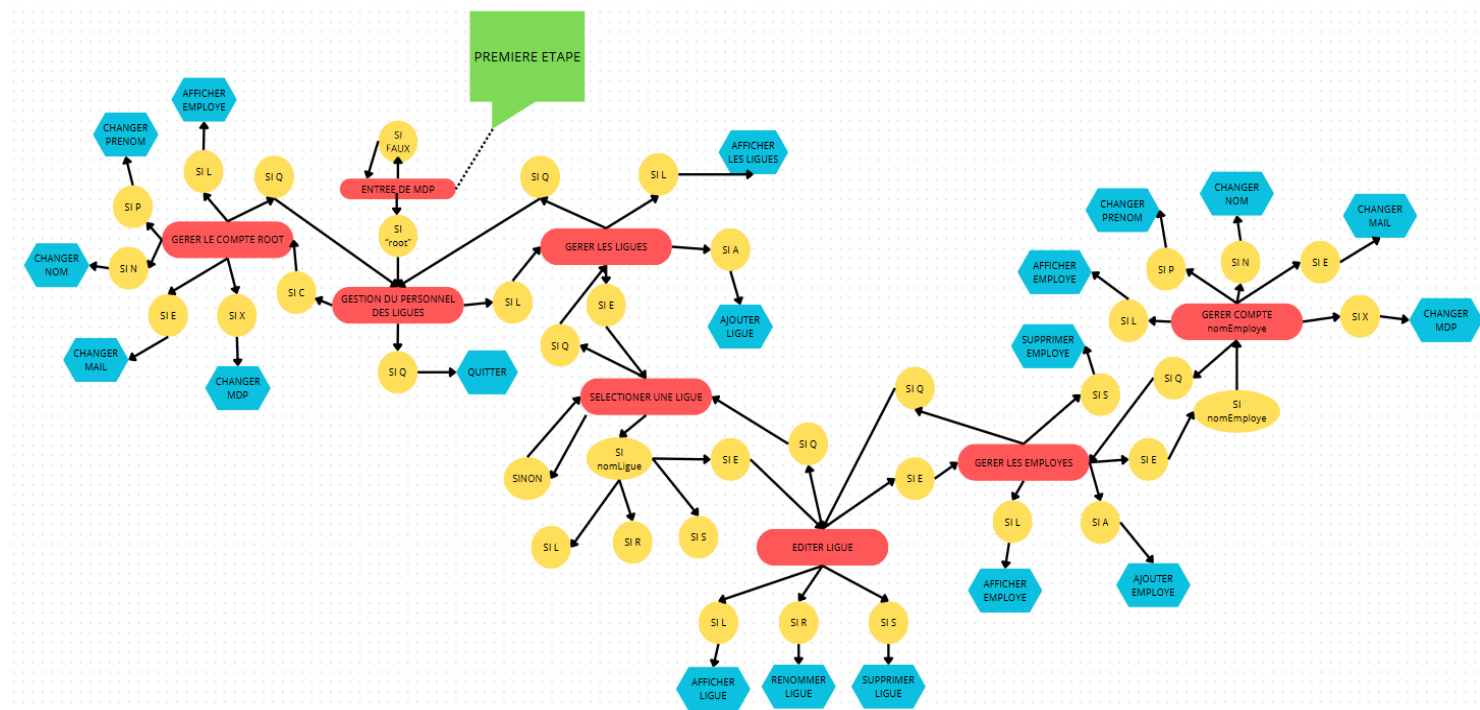
Une fois fait, nous avons commencé à travailler sur les scripts de [créations de table](#) et de [contraintes](#) :

```
1 ALTER TABLE Employe
2 ADD COLUMN ID_Ligue INT,
3 ADD COLUMN ID_Role INT;
4
5
6 ALTER TABLE Employe
7 ADD CONSTRAINT FK_ID_Ligue
8 FOREIGN KEY (ID_Ligue) REFERENCES Ligue(ID_Ligue);
9
10 ALTER TABLE Employe
11 ADD CONSTRAINT FK_ID_Role
12 FOREIGN KEY (ID_Role) REFERENCES Role(ID_Role);
```

```
1 DROP TABLE IF EXISTS Ligue;
2 DROP TABLE IF EXISTS Employe;
3 DROP TABLE IF EXISTS Role;
4
5
6 CREATE TABLE Ligue (
7     ID_Ligue int AUTO_INCREMENT PRIMARY KEY,
8     nomLigue varchar(100)
9 ) engine = InnoDB;
10
11 CREATE TABLE Employe (
12     ID_Employe int AUTO_INCREMENT PRIMARY KEY,
13     prenomEmploye varchar(100),
14     nomEmploye varchar(100),
15     mail varchar(100),
16     passwd varchar(100),
17     datearv date,
18     datedepart date
19 ) engine = InnoDB;
20
21 CREATE TABLE Role (
22     ID_Role int PRIMARY KEY,
23     nomRole varchar(100)
24 ) engine = InnoDB;
```

Partie - 2 Arbre Heuristique :

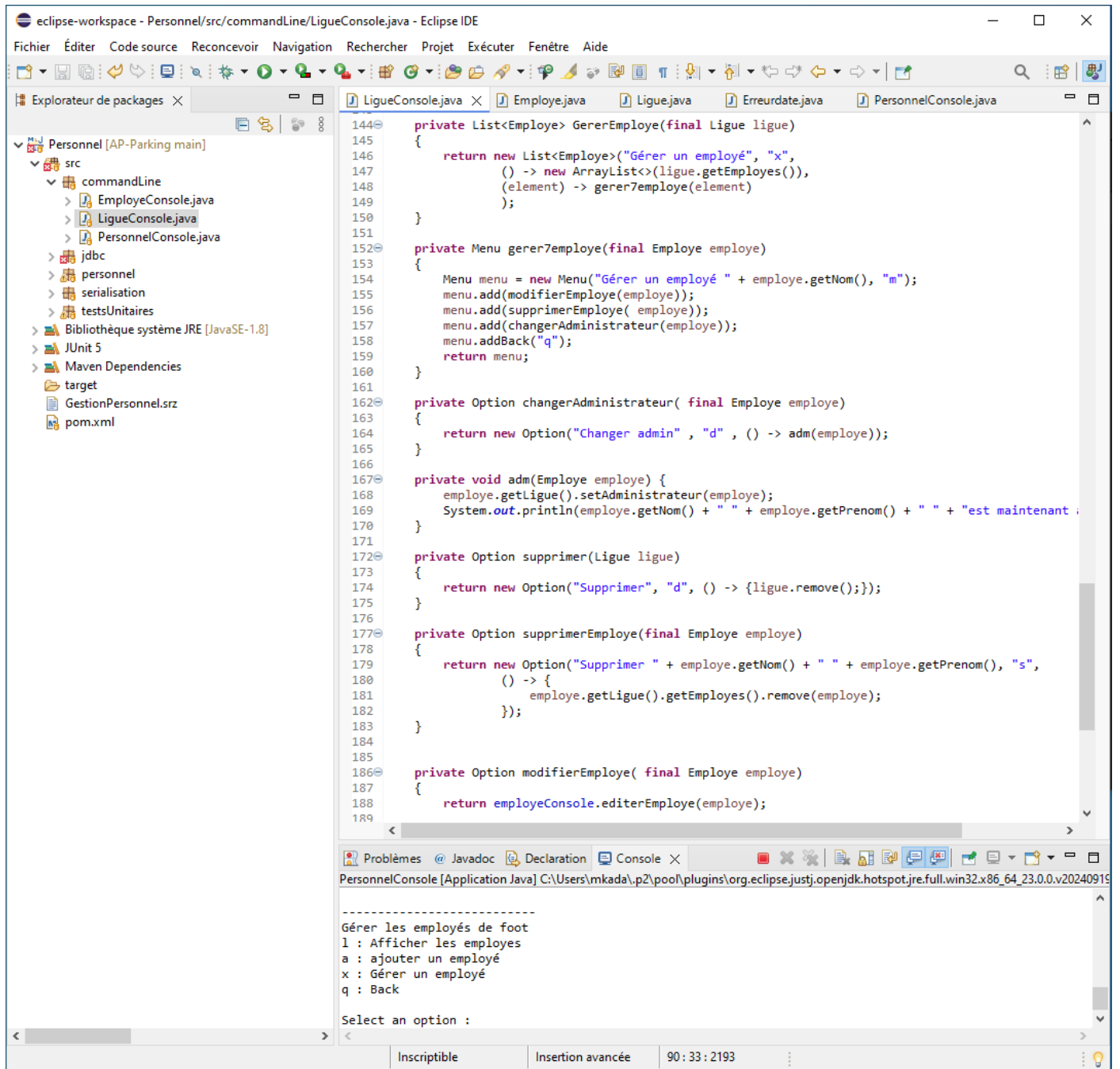
Pour la deuxième partie de la structure de notre projet, nous avons cloné l'application depuis GitHub l'avons lancé pour nous familiariser avec celle-ci. Par la suite nous avons réalisé un [arbre heuristique](#) retraçant toutes les fonctionnalités de l'application. :



Partie 3 – Java :

Enfin après avoir établis une structure au projet, nous avons utilisé Eclipse afin d'améliorer l'application.

Nous avons ajouté un sous-menu Gérer les employés au menu Gérer Ligue d'origine permettant une meilleure clarté. Nous avons également profité de ce nouveau sous-menu pour ajouter une option permettant de changer l'administrateur de la ligue concerné par l'employé concerné.



The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with packages like `Personnel`, `src`, `commandLine`, `EmployeConsole.java`, `LigueConsole.java`, and `PersonnelConsole.java`.
- Editor:** Displays the code for `LigueConsole.java`. The code includes methods for managing employees and leagues, such as `GererEmploye`, `gerer7employe`, `changerAdministrateur`, `adm`, `supprimer`, `supprimerEmploye`, and `modifierEmploye`.
- Console:** Shows the application output, including the menu structure and the prompt "Select an option :".

```
private List<Employe> GererEmploye(final Ligue ligue)
{
    return new List<Employe>("Gérer un employé", "x",
        () -> new ArrayList<>(ligue.getEmployes()),
        (element) -> gerer7employe(element)
    );
}

private Menu gerer7employe(final Employe employe)
{
    Menu menu = new Menu("Gérer un employé " + employe.getNom(), "m");
    menu.add(modifierEmploye(employe));
    menu.add(supprimerEmploye(employe));
    menu.add(changerAdministrateur(employe));
    menu.addBack("q");
    return menu;
}

private Option changerAdministrateur( final Employe employe)
{
    return new Option("Changer admin", "d", () -> adm(employe));
}

private void adm(Employe employe) {
    employe.getLigue().setAdministrateur(employe);
    System.out.println(employe.getNom() + " " + employe.getPrenom() + " " + "est maintenant ");
}

private Option supprimer(Ligue ligue)
{
    return new Option("Supprimer", "d", () -> {ligue.remove();});
}

private Option supprimerEmploye(final Employe employe)
{
    return new Option("Supprimer " + employe.getNom() + " " + employe.getPrenom(), "s",
        () -> {
            employe.getLigue().getEmployes().remove(employe);
        });
}

private Option modifierEmploye( final Employe employe)
{
    return employeConsole.editerEmploye(employe);
}
```

PersonnelConsole [Application Java] C:\Users\mkada\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.0.v20240919

Gérer les employés de foot
l : Afficher les employes
a : ajouter un employé
x : Gérer un employé
q : Back
Select an option :

Ensuite nous avons décidé [d'améliorer la prise en considération des erreurs](#), notamment concernant les dates d'arrivées et de départs, par exemple concernant le format de la date.

```
private Option ajouterEmploye(final Ligue ligue)
{
    return new Option("ajouter un employé", "a",
        () ->
        {
            // Saisie en chaîne - demande toutes les informations d'abord
            String nom = getString("nom : ");
            String prenom = getString("prenom : ");
            String mail = getString("mail : ");
            String password = getString("password : ");
            String dateArrStr = getString("date arrivée (AAAA-MM-JJ) : ");
            String dateDepStr = getString("date départ (AAAA-MM-JJ) : ");

            try {
                // Exception Format - vérifie le format des dates
                LocalDate dateArrivee = LocalDate.parse(dateArrStr);
                LocalDate dateDepart = LocalDate.parse(dateDepStr);

                // Exception ordre - vérifie l'ordre chronologique
                if (dateDepart.isBefore(dateArrivee)) {
                    throw new Erreurdate();
                }

                // Si tout est valide, crée l'employé
                ligue.addEmploye(nom, prenom, mail, password, dateArrivee, dateDepart);
                System.out.println("Employé ajouté avec succès");
            } catch (DateTimeParseException e) {
                // Gestion de l'exception de format
                System.out.println("Erreur : Format de date invalide. Utilisez le format AAAA-MM-JJ");
            } catch (Erreurdate e) {
                // Gestion de l'exception d'ordre chronologique
                System.out.println(e.getMessage());
            }
        }
    );
}
```