Gestion du Personnel des Ligues

Itération 1

Table des matières

Introduction

Ce projet a été réalisé dans le cadre de ma formation de deuxième année de BTS SIO par notre groupe composé de 3 étudiants : Hadrien Schoeffler le chef de groupe, Racime HOUHOU et Marie Ucles. Nous avions 2 séances de 4h pour le mettre à bien.

Rappel du Contexte de la M2L

Cette ressource pédagogique est destinée aux enseignants. Elle constitue une matière d'œuvre à partir de laquelle ils pourront placer leurs étudiants dans différentes situations professionnelles décrites dans le référentiel. Le contexte proposé est celui de la Maison de Ligues de Lorraine (M2L) qui a pour mission de fournir des espaces et des services aux différentes ligues sportives régionales et à d'autres structures hébergées. Ce contexte est associé à différentes propositions de projets et missions susceptibles d'être réalisés par les étudiants dans l'horaire de PPE à différents moments de la formation. Il peut aussi être utilisé pour illustrer certains savoirs ou savoir-faire associés à différents modules d'enseignement, ceci aussi bien pour les modules communs (SI) que pour les modules spécifiques des parcours SISR et SLAM. Les choix d'exploitation pédagogique de ce contexte sont laissés à la libre initiative des professeurs.

Présentation de la Gestion du Personnel

Un des responsables de la M2L, utilise une application pour gérer les employés des ligues. L'application est mise à votre disposition par le biais des ressources suivantes :

- Le code source sur GitHub.
- La documentation.
- Une bibliothèque logicielle de dialogue en ligne de commande, disponible dans ce dépôt.

Cette application, très simple, n'existe qu'en ligne de commande et est mono-utilisateur. Nous souhaiterions désigner un administrateur par ligue et lui confier la tâche de recenser les employés de sa ligue. Une partie du travail est déjà faite mais vous allez devoir le compléter.

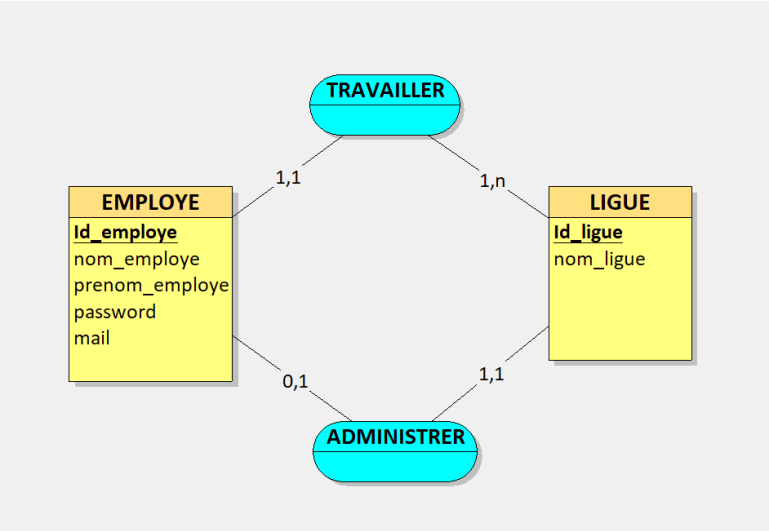
Les niveaux d'habilitation des utilisateurs sont les suivants :

- Un simple employé de ligue peut ouvrir l'application et s'en servir comme un annuaire, mais il ne dispose d'aucun droit d'écriture.
- Un employé par ligue est administrateur et dispose de droits d'écriture peut gérer la liste des employés de sa propre ligue avec une application bureau.
- Le super-administrateur a accès en écriture à tous les employés des ligues. Il peut aussi gérer les comptes des administrateurs des ligues avec une application accessible en ligne de commande.
- L'application doit être rendue multi-utilisateurs grâce à l'utilisation d'une base de données.
- Les trois niveaux d'habilitation ci-dessus doivent être mis en place.

Activités de l'Itération 1

Modélisation d'une base de données avec un MCD

Cette première activité a été réalisée par Racime et Marie, après étude des spécifications du besoin et les niveaux d'habilitation des utilisateurs ci-dessus.



Vérification du fonctionnement correct de l'application grâce à des tests unitaires.

Cette deuxième activité a été réalisé par Hadrien Schoeffler et Racime Houhou. Le but de cette activité est de créer des tests unitaires vérifiant chaque fonction des fichiers Employe.java, Ligue.java et GestionPersonnel.java afin de vérifier plus tard le bon fonctionnement du programme quand on aura l'occasion d'ajouter ou de changer certaines fonctions.

Par exemple : on teste le getNom() de TestLigue.java

```
/**
 * Retourne le nom de la ligue.
 * @return le nom de la ligue.
 */

public String getNom()
{
    return nom;
}
```

```
@Test
void getNom() throws SauvegardeImpossible
{
    Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
    assertEquals("Fléchettes", ligue.getNom());
}
@Test
```

Gestion de la date de départ et de celle d'arrivée de chaque employé (couche métier + tests unitaires)

Cette activité a été réalisé par Hadrien Schoeffler. Elle consiste à ajouter 2 paramètres dans la variable employe du fichier Employe.java. On peut distinguer 2 types d'employés, ceux en CDD dont la date de départ est à rentrer, et ceux en CDI dont la date de départ est inexistante. On obtient donc :

```
public class Employe implements Serializable, Comparable<Employe>
{
    private static final long serialVersionUID = 47957217180379947 34L;
    private String nom, prenom, password, mail;
    private Ligue ligue;
    private GestionPersonnel gestionPersonnel;
    private LocalDate dateArrivee, dateDepart;


    /*Employé en CDI*/
    Employe(GestionPersonnel gestionPersonnel, Ligue ligue, String nom, String prenom, String mail, String password)
    {
        this.gestionPersonnel = gestionPersonnel;
        this.nom = nom;
        this.prenom = prenom;
        this.password = password;
        this.mail = mail;
        this.ligue = ligue;
        /*Date de départ fixée*/
        this.dateArrivee = LocalDate.now();
        this.dateDepart = null;
    }

    /*Emplové en CDD*/
    Employe(GestionPersonnel gestionPersonnel, Ligue ligue, String nom, String prenom, String mail, String password, LocalDate dateDepart)
    {
        this.gestionPersonnel = gestionPersonnel;
        this.nom = nom;
        this.prenom = prenom;
        this.password = password;
        this.mail = mail;
        this.ligue = ligue;
        /*Date de départ fixée*/
        this.dateArrivee = LocalDate.now();
        this.dateDepart = dateDepart;
    }
```

S'ensuit les fonctions permettant d'attribuer les dates de départ et d'afficher les dates de départs et d'arrivées,

```
public LocalDate getDateArrivee()
{
    return this.dateArrivee;
}
public void setDateDepart(LocalDate dateDepart)
{
    this.dateDepart = dateDepart;
}
public LocalDate getDateDepart()
{
     return this.dateDepart;
}
```

En faisant évidemment attention à créer une 2ème fonction addEmploye avec la LocalDate en plus :

```
public Employe addEmploye(String nom, String prenom, String mail, String password, LocalDate dateDepart)
{
    Employe employe = new Employe(this.gestionPersonnel, this, nom, prenom, mail, password, dateDepart);
    employes.add(employe);
    return employe;
}
```

On peut alors vérifier le fonctionnement des fonctions avec les tests unitaires :

```java
@Test
void getDateArrivee() throws SauvegardeImpossible
{
    Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
    Employe employe = ligue.addEmploye("Bouchard", "Gérard", "g.bouchard@gmail.com", "azerty");
    assertEquals(LocalDate.now(), employe.getDateArrivee());
}
@Test
void getDateDepart() throws SauvegardeImpossible
{
    Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
    Employe employe = ligue.addEmploye("Bouchard", "Gérard", "g.bouchard@gmail.com", "azerty", LocalDate.of(2022, 12, 20));
    assertEquals(LocalDate.of(2022, 12, 20), employe.getDateDepart());
}
@Test
void setDateDepart() throws SauvegardeImpossible
{
    Ligue ligue = gestionPersonnel.addLigue("Fléchettes");
    Employe employe = ligue.addEmploye("Bouchard", "Gérard", "g.bouchard@gmail.com", "azerty", LocalDate.of(2022, 12, 20));
    employe.setDateDepart(LocalDate.of(2023, 01, 23));
    assertEquals(LocalDate.of(2023, 01, 23), employe.getDateDepart());
}
```

Représentation des menus du dialogue en ligne de commande avec un arbre heuristique

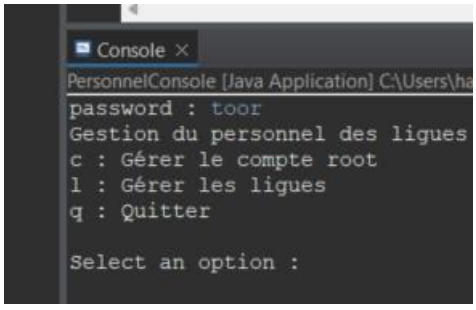
Cette dernière activité a été réalisé par Marie Ucles. Pour obtenir la représentation des menus du dialogue, nous devions créer un fichier Credentials.java pour faire fonctionner le jdbc.java.

JDBC.java Credentials.java

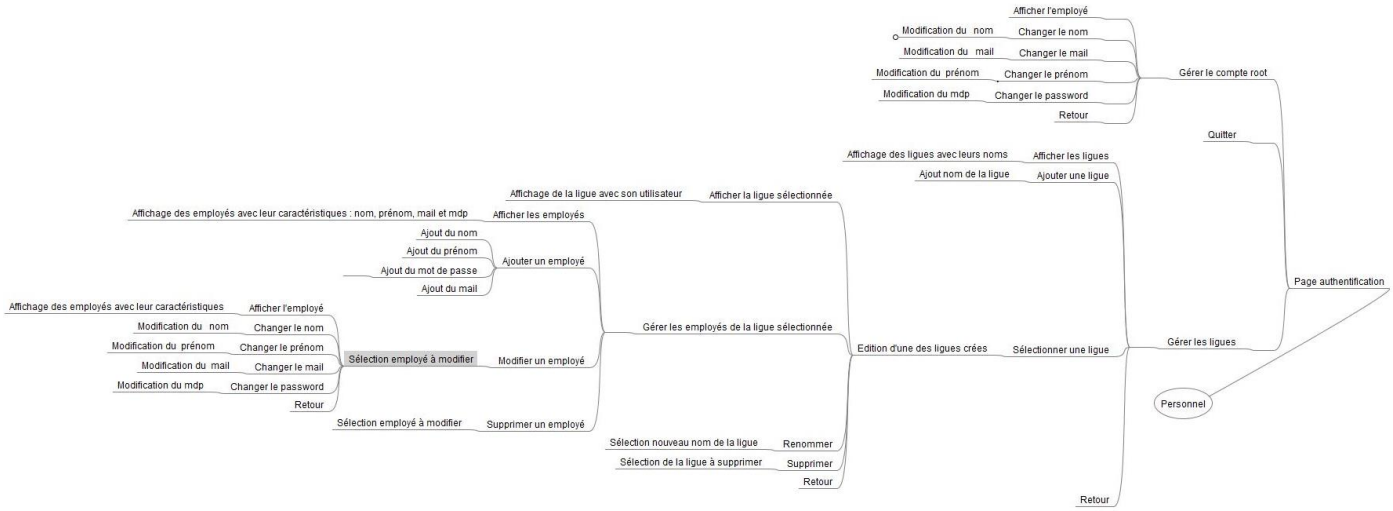
```java
package jdbc;

public class Credentials
{
    private static String driver = "mysql";
    private static String driverClassName = "com.mysql.cj.jdbc.Driver";
    private static String host = "localhost";
    private static String port = "3306";
    private static String database = "personnel";
    private static String user = "root";
    private static String password = "toor";

    static String getUrl()
    {
        return "jdbc:" + driver + "://" + host + ":" + port + "/" + database ;
    }

    static String getDriverClassName()
    {
        return driverClassName;
    }

    static String getUser()
    {
        return user;
    }

    static String getPassword()
    {
        return password;
    }
}
```

Ainsi, JDBC.java, servant d'interface pour se connecter à la base de données, permet faire ainsi fonctionner les 3 fichiers EmployeConsole.java, LigueConsole.java et PersonnelConsole.java de CommandLine.

Pour accéder aux menus, il faut démarrer le fichier PersonnelConsole.java. On obtient ceci :



En suivant chaque branche, on peut alors identifier la carte heuristique des menus du dialogue :



Compétences acquises