**DOCUMENTATION PERSONNEL :**

# Gestion du personnel des ligues*:*

Ce projet a été réalisé en groupe de 3 personnes lors de ma 2ème année de BTS SIO durant lequel nous avons dû modifié et programmé en Java.

## Un des responsables de la M2L, utilise une application pour gérer les employés des ligues. 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.

## *Le besoin attendu :*

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 grace à l’utilisation d’une base de données.
* Les trois niveaux d’habilitation ci-dessus doivent être mis en place.

***Nous devions :***

* Conserver java pour l’application.
* Utiliser le versionnement avec git
* Utiliser la bibliothèque de dialogue en ligne de commande fournie.

***Caractéristiques à mettre en place sur le projet :***

* Modélisation d’une base de données avec un MCD.
* Vérification du fonctionnement correct de l’application grâce à des tests unitaires.
* Gestion de la date de départ et de celle d’arrivée de chaque employé (couche métier + tests unitaires).
* Représentation des menus du dialogue en ligne de commande avec un arbre heuristique (Utilisez un logiciel de type Freemind).
* Création de la base de données et production du script de création de tables.
* Gestion des dates dans le dialogue en ligne de commande.
* Dans le dialogue en ligne de commande, un employé doit être selectionné avant que l’on puisse choisir de modifier ou de supprimer.
* Possibilité de changer l’administrateur d’une ligue en ligne de commande.
* Création d’une classe fille de Passerelle permettant de gérer le dialogue avec la base de données avec JDBC (ou avec Hibernate si vous le souhaitez).
* Utilisation de la base de données pour réaliser les opérations d’ajout, de modification, de suppression des ligues et des employés.
* Modélisation de l’interface graphique avec des maquettes.

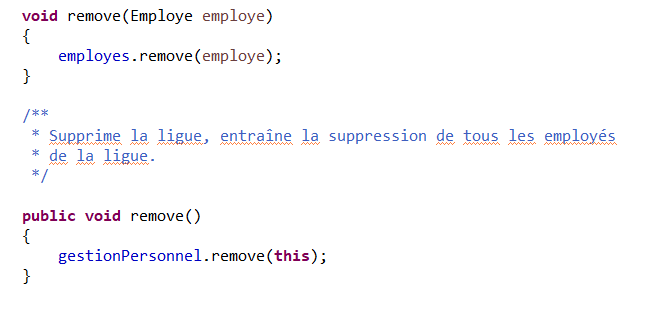
***MCD Réalisé pour cette application :***

Une image contenant diagramme

Description générée automatiquement

Tests unitaires :

Afin de vérifier le fonctionnement correct de l’application nous avons du effectuer des test unitaires.



Exemple de test pour vérifier la suppression d’un employé. :

Une image contenant texte

Description générée automatiquement

Voici l’arbre heuristique représentant les menus du dialogue en ligne de commande :

Une image contenant diagramme

Description générée automatiquement

Ensuite nous avons crée le script de la base de données et production du script de création de tables

Voici le script :

|  |
| --- |
| DROP DATABASE IF EXISTS personnel; |
|  |

|  |
| --- |
| CREATE DATABASE personnel ; |
|  |

|  |
| --- |
| USE personnel; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| DROP TABLE IF EXISTS Employe; |
|  |

|  |
| --- |
| DROP TABLE IF EXISTS Ligue; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| create table Employe ( |
|  |

|  |
| --- |
| id\_employe BIGINT(8) , |
|  |

|  |
| --- |
| nomEmploye varchar(25), |
|  |

|  |
| --- |
| prenomEmploye varchar(25) , |
|  |

|  |
| --- |
| password varchar(25) not null, |
|  |

|  |
| --- |
| mail varchar(70) not null, |
|  |

|  |
| --- |
| DateArrivee DATE, |
|  |

|  |
| --- |
| DateDepart DATE, |
|  |

|  |
| --- |
| id\_ligue BIGINT, |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| constraint PK\_EMP primary key(id\_employe) |
|  |

|  |
| --- |
| )ENGINE=INNODB; |
|  |

|  |
| --- |
| DESC Employe; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| create table Ligue( |
|  |

|  |
| --- |
| id\_ligue BIGINT(4), |
|  |

|  |
| --- |
| nom varchar(255) not null, |
|  |

|  |
| --- |
| administrateur varchar(50) not null, |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| constraint PK\_LIG primary key(id\_ligue) |
|  |

|  |
| --- |
| )ENGINE=INNODB; |
|  |

|  |
| --- |
| DESC Ligue; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ALTER TABLE `Employe` ADD FOREIGN KEY (`id\_ligue`) REFERENCES `ligue` (`id\_ligue`); |
|  |

ALTER TABLE `Ligue` ADD FOREIGN KEY (`id\_ligue`) REFERENCES `Employe` (`id\_employe`);

Puis nous avons géré les dates en ligne de commande avec :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Il y a 2 catégories d’employés CDI et CDD afin d’avoir une date d’arrivée et de fin pour celui en CDD. L’employé n’a pas de date de fin prédéfinie.

Voici les méthodes permettant de gérer les dates d’arrivée et les dates de départ.

Une image contenant texte, lettre

Description générée automatiquement

Ajout de dateDepart dans le fichier Ligue.java pour actualiser la date de départ d’un employé en CDD

Une image contenant texte

Description générée automatiquement

Voici les test unitaires correspondants :

Une image contenant texte

Description générée automatiquement

Afin de gérer le permettant de gérer le dialogue avec la base de données avec JDBC :

Une image contenant texte

Description générée automatiquement

Mise en place du fichier Credentials.java afin de relier à la base de données

Mise en place du fichier JBDC.java afin de relier à la base de données

Puis mise en place des insert et update