Antoni M BIZI BANDOKI, Lucas AZER, Yannis N'GUESSAN

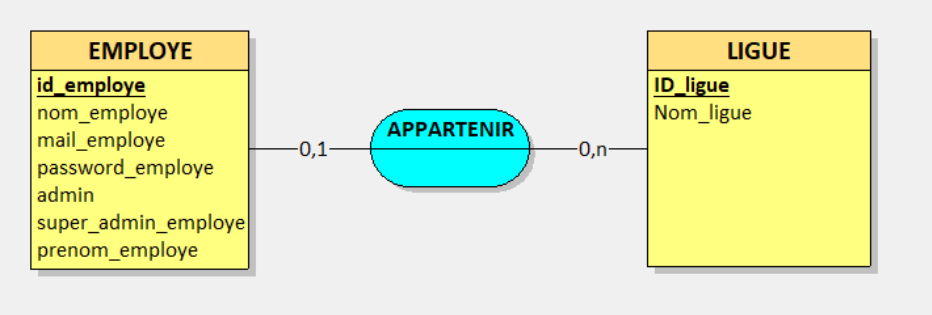
Personnel

Outils/Langages utilisés: Java, SQL, Looping

Personnel est un projet réalisé en Java, c'est une application qui permet de gérer les employés de la ligue M2L, en désignant un administrateur par ligue, qui lui, devra recencer les employés de sa ligue.
Cette application offre la possibilité de créer, modifier et supprimer des ligues ainsi que des employés, le tout sous la supervision de l'administrateur.

Base de donnée :

Voici le MCD :



Un employe peut avoir aucune ou une Ligue

Une Ligue peut avoir plusieurs employe

Après avoir modéliser la base de donnée, nous avons crée le script de création de table en SQL :

```sql
CREATE DATABASE IF NOT EXISTS  m2l;

USE m2l;

DROP TABLE IF EXISTS employe;
DROP TABLE IF EXISTS ligue;


CREATE TABLE ligue(
    Id_Ligue INT NOT NULL AUTO_INCREMENT,
    Nom_Ligue VARCHAR(40),
    PRIMARY KEY (Id_Ligue)
)ENGINE = INNODB;


CREATE TABLE employe(
    User_Id INT NOT NULL AUTO_INCREMENT,
    Id_Ligue INT,
    Nom VARCHAR(25),
    Prenom VARCHAR(25),
    Mdp VARCHAR (50),
    Date_Arrivee DATE,
    Date_Depart DATE,
    Mail VARCHAR(25),
    PRIMARY KEY (User_Id),
    FOREIGN KEY (Id_Ligue) REFERENCES Ligue (Id_Ligue)
)ENGINE = INNODB;
```

Code Java

Puis avec le code Java fournit par le professeur, on a dû effectuer quelque modification : tel que

```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 dateArrivee = LocalDate.of(0000, 01, 01);
    private LocalDate dateDepart = LocalDate.of(0000, 01, 01);
    private int id;
```

Constructeur de la class employe dans laquelle nous avons rajouté les date d'arrivée et de depart au format compris par MYSQL d'un employe.

```java
package personnel;

public class ExceptionArrivee extends Exception {
        public ExceptionArrivee()
        {
            System.out.println("Exception ExceptionArrivee has been raised...");
        }
        @Override
        public String toString()
        {
          return "La date d'arrivée ne peut pas etre avant la date de départ ";
        }


}
```

Class exception pour la date arrive

```java
package personnel;

public class ExceptionDepart extends Exception{
        public ExceptionDepart()
        {
            System.out.println("Exception ExceptionDepart has been raised...");
        }
        @Override
        public String toString()
        {
          return "La date de départ ne peut pas etre avant la date d'arrivée ";
        }


}
```

Class exception pour la date départ

```
Option editerEmploye(Employe employe)
{
        Menu menu = new Menu("Gérer le compte " + employe.getNom(), "c");
        menu.add(afficher(employe));
        /*menu.add(changerNom(employe));
        menu.add(changerPrenom(employe));
        menu.add(changerMail(employe));
        menu.add(changerPassword(employe)); */

        //ajout des options pour modifier et supprimer un employé
        menu.add(modifierEmploye(employe));
        menu.add(GererLesDates(employe));
        menu.add(supprimerEmploye(employe));
        menu.addBack("q");
        return menu;
}
```

Ajout de la gestion des dates dans le dialogue en ligne de commande.

```
//Modification de la methode changerAdministrateur pour qu'elle change l'administrateur d'une ligue
private List<Employe> changerAdministrateur(final Ligue ligue)
{
    return new List<>("Changer l'administrateur", "c",
            () -> new ArrayList<>(ligue.getEmployes()),
            (index, element) -> {ligue.setAdministrateur(element);}
            );
}
```

Option permettant de changer l'administrateur d'une ligue

```
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 = "superuser";
    private static String password = "root";

    static String getUrl()
    {
        return "jdbc:" + driver + "://" + host + ":" + port + "/" + database +"?zeroDateTimeBehavior=CONVERT_TO_NULL&serverTimezone=UTC";

    }

    static String getDriverClassName()
    {
        return driverClassName;
    }

    static String getUser()
    {
        return user;
    }

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

Class Credentials contenant les informations de connexion a la Base de donnees local.

```java
package jdbc;

import java.sql.DriverManager;

public class JDBC implements Passerelle
{
    Connection connection;

    public JDBC()
    {
        try
        {
            Class.forName(Credentials.getDriverClassName());
            connection = DriverManager.getConnection(Credentials.getUrl(), Credentials.getUser(), Credentials.getPassword());
        }
        catch (ClassNotFoundException e)
        {
            System.out.println("Pilote JDBC non installé.");
        }
        catch (SQLException e)
        {
            System.out.println(e);
        }
    }
```

Fonction se trouvant dans JDBC et permettant la connexion a la base de données

```java
@Override
public int insert(Ligue ligue) throws SauvegardeImpossible
{
    try
    {
        PreparedStatement instruction;
        instruction = connection.prepareStatement("insert into ligue (Nom_Ligue) values(?)", Statement.RETURN_GENERATED_KEYS);
        instruction.setString(1, ligue.getNom());
        instruction.executeUpdate();
        ResultSet id = instruction.getGeneratedKeys();
        id.next();
        return id.getInt(1);
    }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
```

Fonction insert récupérant le résultat de la requête SQL effectué dans la base pour la transmettre a l'application

```java
@Override
public void deleteLigue(Ligue ligue) throws SauvegardeImpossible
{
    try
    {
        System.out.println("Ca marche");
        PreparedStatement instruction;
        instruction = connection.prepareStatement("delete from ligue where Id_Ligue = ?", Statement.RETURN_GENERATED_KEYS);
        instruction.setInt(1, ligue.getID());
        instruction.executeUpdate();
    }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
@Override
public void deleteEmploye(Employe employe) throws SauvegardeImpossible
{
    try
    {
        System.out.println("Ca marche taggle");
        PreparedStatement instruction;
        instruction = connection.prepareStatement("delete from employe where User_Id = ?", Statement.RETURN_GENERATED_KEYS);
        instruction.setInt(1, employe.getID());
        instruction.executeUpdate();
    }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
```

Fonctions de suppression d'une ligue et d'un employé

```java
@Override
public int insert(Employe employe) throws SauvegardeImpossible
{
    try {
        Date dateArriveeSQL = Date.valueOf(employe.getDateArrivee());
        Date dateDepartSQL = Date.valueOf(employe.getDateDepart());
        PreparedStatement instruction;
        instruction = connection.prepareStatement("insert into employe (Nom, Prenom, Mdp, Date_Arrivee, Date_Depart, Mail) values(?, ?, ?, ?, ?, ?)"
        instruction.setString(1, employe.getNom());
        instruction.setString(2, employe.getPrenom());
        instruction.setString(3, employe.getPassword());
        instruction.setDate(4, dateArriveeSQL);
        instruction.setDate(5, dateDepartSQL);
        instruction.setString(6, employe.getMail());
        instruction.executeUpdate();
        ResultSet id = instruction.getGeneratedKeys();
        id.next();
        return id.getInt(1);
     }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
```

Fonction d'insertion d'un employe

```java
@Override
public int update(Employe employe) throws SauvegardeImpossible
{
    try
    {
        PreparedStatement instruction;
        instruction = connection.prepareStatement("update utilisateur set Nom = ?, Prenom =
        instruction.setString(1, employe.getNom());
        instruction.setString(2, employe.getPrenom());
        instruction.setString(3, employe.getPassword());
        instruction.setString(4, employe.getDateArrivee().toString());
        instruction.setString(5, employe.getDateDepart().toString());
        instruction.executeUpdate();
        ResultSet id = instruction.getGeneratedKeys();
        id.next();
        return id.getInt(1);
    }
    catch (SQLException exception)
    {
        exception.printStackTrace();
        throw new SauvegardeImpossible(exception);
    }
}
```

Fonction de modification d'un employé