

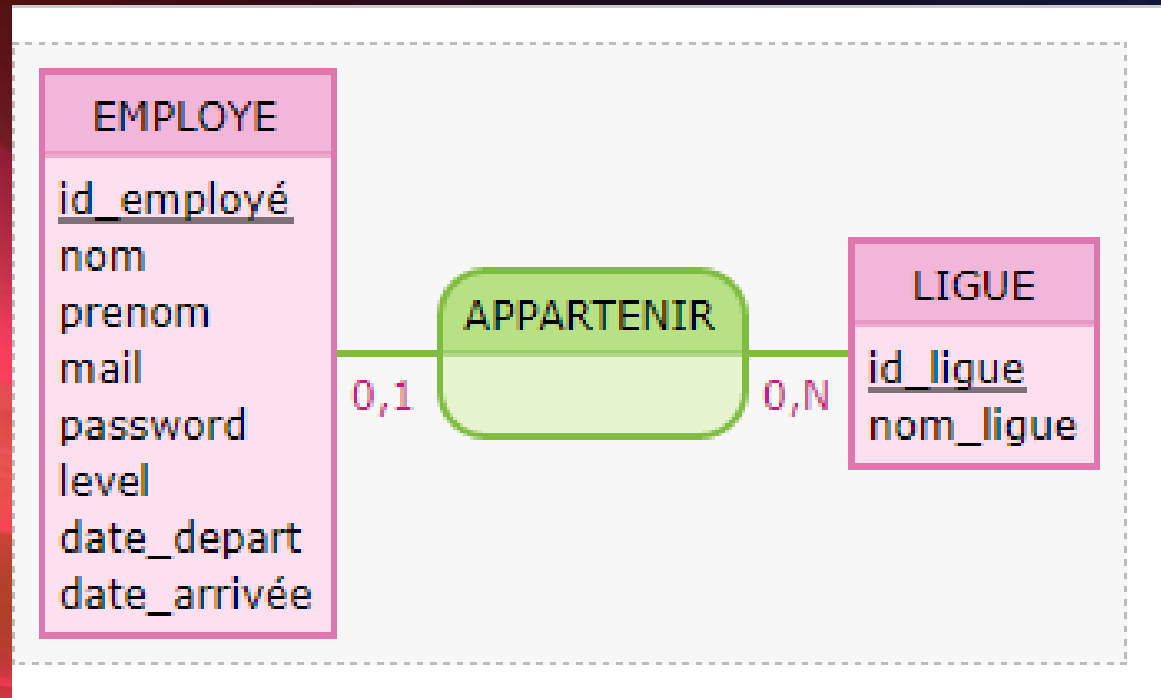
PPE - GESTION DU PERSONNEL DES LIGUES

La gestion des inscriptions aux compétitions est encore aujourd'hui assurée par les ligues avec des tableurs, et le développement d'un logiciel davantage adapté est en cours.

L'application en question permet de gérer un ensemble de compétitions, de personnes, et d'affecter des personnes à des compétitions. Il est possible que certaines compétitions soient réservées à des équipes et qu'il soit impossible à une personne seule de s'inscrire dans le cas, tous les membres de l'équipe doivent être enregistrés.

Cette application ne sera pas accessible au grand public, seuls des employés des ligues pourront y accéder pour saisir les inscrits. Les précédents informaticiens se sont penchés sur le problème et ont déjà commencé à développer la couche métier

PPE - GESTION DU PERSONNEL DES LIGUES



```
1  EMPLOYE : id_employé,nom,prenom,mail,password,level,date_depart,date_arrivée
2  APPARTENIR , 01 EMPLOYE, 0N LIGUE
3  LIGUE :id_ligue, nom_ligue
4
```


PPE - GESTION DU PERSONNEL DES LIGUES

The screenshot shows a Kanban board titled 'personnel' with a background image of a mountain landscape. The board is organized into four columns: 'A faire', 'En cours', 'Terminé', and 'A completer'. Each column contains task cards with progress bars and status indicators. The 'A faire' column has four tasks. The 'En cours' column has one task. The 'Terminé' column has four tasks. The 'A completer' column has one task. The board includes a top navigation bar with a search icon, a notification banner, and a user profile. A sidebar on the right contains a 'Butler' button and a menu toggle.

personnel ☆ | Java project Free | Visible par les membres d'une équipe | S YN Inviter | Butler ... Afficher le menu

A faire ...

- Création d'une interface graphique (Swing ou JavaFx) pour que les administrateurs puissent gérer les ligues.
- Si une ligue n'a pas d'administrateur, c'est automatiquement le root qui devient l'administrateur de la ligue (avec les tests unitaires correspondants).
- Installation de la base de données sur un serveur accessible dans le réseau local de la société.
- Rédaction d'un mode opératoire à l'usage des administrateurs.
- Saisissez un titre pour cette carte...

En cours ...

- + Ajouter une carte

Terminé ...

- gestion des dates dans le dialogue en ligne de commande
1 ⓘ S YN
- changer l'administrateur d'une ligue
- Installation de la BDD sur le serveur(172.16.0.2)
- Création d'une classe fille pour gérer la connexion à la base de données avec JDBC
- Modélisation de l'interface graphique avec des maquettes
- + Ajouter une autre carte

A completer ...

- faire un arbre heuristique
YN
- + Ajouter une autre carte

+ Ajoutez une autre l

LES TESTS UNITAIRES

Un test unitaire est une procédure visant à vérifier le bon fonctionnement d'un programme.

```
package testsUnitaires;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

import personnel.*;

class testLigue
{
    @Test
    void testCreateLigue()
    {
        Ligue ligue = new Ligue("Fléchettes");
        assertEquals("Fléchettes", ligue.getNom());
    }

    @Test
    void testAddEmploye()
    {
        Ligue ligue = new Ligue("Fléchettess");
        Employe employe = ligue.addEmploye("Bouchard", "Gérard", "g.bouchard@gmail.com", "azerty");
        assertEquals(employe, ligue.getEmployes().first());
    }
}
```

GESTION DES DATES

GESTION DE LA DATE DE DÉPART ET DE CELLE D'ARRIVÉE DE CHAQUE EMPLOYÉ

@Test

```
void testGetArrive() {  
    Ligue ligue = new Ligue("Fléchettes");  
    LocalDate dateArrive = LocalDate.of(2020, 01, 01);  
    Employe employe = ligue.addEmploye("selima", "bk", "sbk@gmail.com", "pass", dateArrive);  
    LocalDate date = LocalDate.of(2020, 01, 21);  
    employe.setDateDepart(date);  
    assertEquals(employe.getDateArrive(), dateArrive);  
}
```

@Test

```
void testGetDepart() {  
    Ligue ligue = new Ligue("Fléchettes");  
    Employe employe = ligue.addEmploye("selima", "bk", "sbk@gmail.com", "pass", null);  
    LocalDate date = LocalDate.of(2020, 01, 21);  
    employe.setDateDepart(date);  
    assertEquals(employe.getDateDepart(), date);  
}
```

}

CREATION DE LA BASE DE DONNEE

PHPMYADMIN

| Table | Action | Lignes | Type | Interclassement | Taille | Perte |
|----------------------------------|---|--------|--------|-----------------|--------|-------|
| <input type="checkbox"/> employe | Parcourir Structure Rechercher Insérer Vider Supprimer | 0 | MyISAM | utf8_general_ci | 1 kio | - |
| <input type="checkbox"/> ligue | Parcourir Structure Rechercher Insérer Vider Supprimer | 0 | MyISAM | utf8_general_ci | 1 kio | - |
| 2 tables | Somme | 0 | MyISAM | utf8_general_ci | 2 kio | 0 0 |



☐ Tout cocher

Avec la sélection :



Imprimer



Dictionnaire de données



Nouvelle table

Nom :

Nombre de colonnes :

4

Exécuter

GESTION DES DATES

GESTION DES DATES DANS LE DIALOGUE EN LIGNE DE COMMANDE.

```
private LocalDate getDate() {/// rapeler la fonction getDate

int year = getInt("année : ");
int month = getInt("mois : ");
int dayOfMonth = getInt(" jour :");

LocalDate date = LocalDate.of(year, month, dayOfMonth);

return date;
```

```
Menu editEmployee(Employee employee)
{
    Menu menu = new Menu("Gérer le compte " + employee.getNom(), "c");
    menu.add(afficher(employee));
    menu.add(changerNom(employee));
    menu.add(changerPrenom(employee));
    menu.add(changerMail(employee));
    menu.add(changerPassword(employee));
    menu.add(changerDateArrival(employee));
    menu.add(changerDateDepart(employee));
    menu.addBack("q");
    return menu;
}

private Option changerDateDepart( final Employee employee) {

    return new Option ("changer la date d'arrivée ", "z", () -> {employee.setArrival(getDate());});
}

private Option changerDateArrival(final Employee employee) {

    return new Option ("changer la date de départ ", "y",() -> {employee.setDepart(getDate());});
}

private LocalDate getDate() {

int year = getInt("année : ");

int month = getInt("mois : ");

int dayOfMonth = getInt(" jour :");

LocalDate date = LocalDate.of(year,month, dayOfMonth);

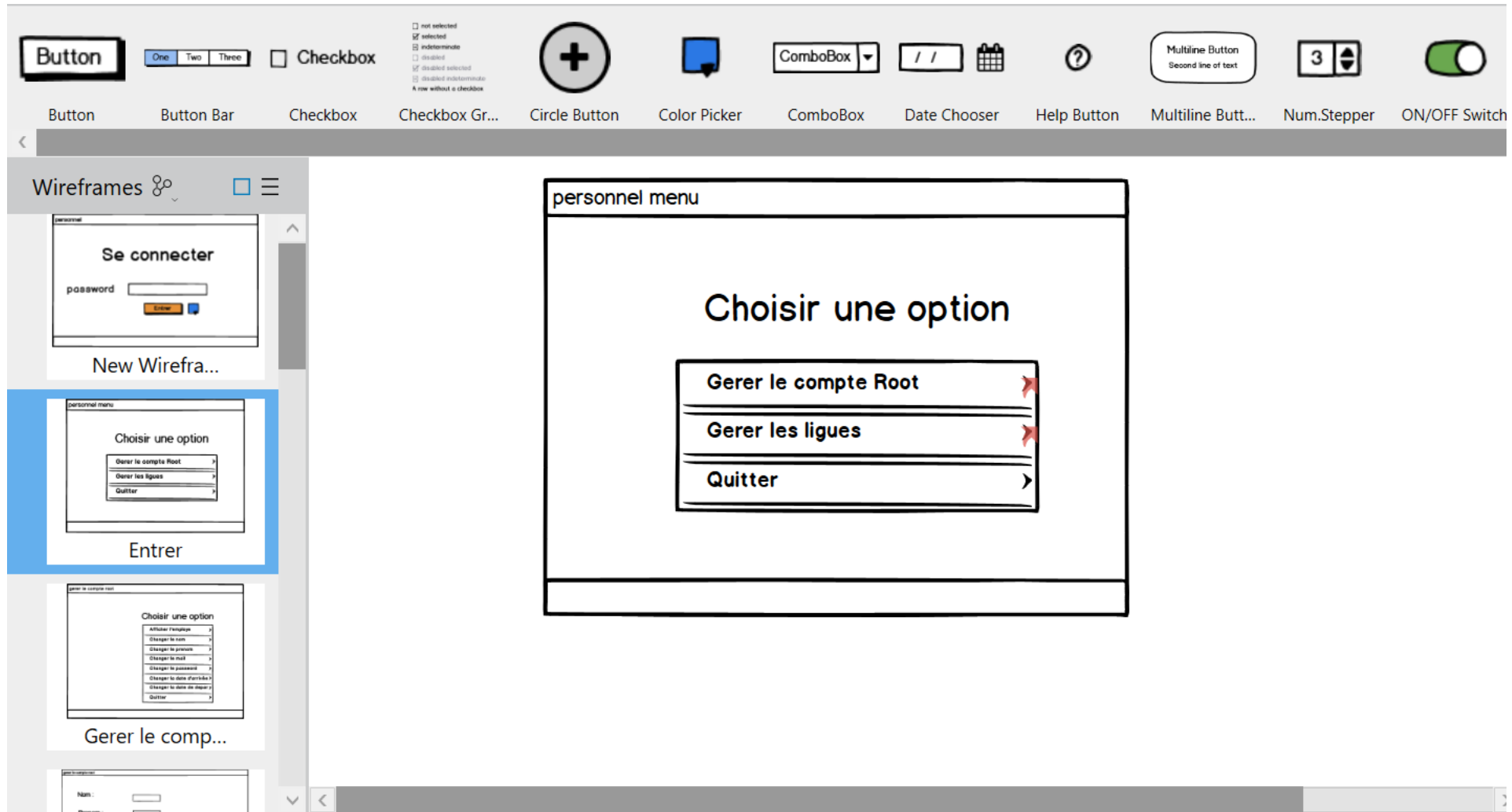
return date;
```

DANS LE DIALOGUE EN LIGNE DE COMMANDE, UN EMPLOYÉ DOIT ÊTRE SELECTIONNÉ AVANT QUE L'ON PUISSE CHOISIR DE MODIFIER OU DE SUPPRIMER.

```
private List<Employe> selectEmploye( Ligue ligue)
{
    return new List<Employe>("Sélectionner un employe", "e",
        () -> new ArrayList<>(ligue.getEmployes()),
        (element) -> employeConsole.selectEmployer(element)
    );
}

private Menu gererEmployes(Ligue ligue)
{
    Menu menu = new Menu("Gérer les employés de " + ligue.getNom(), "e");
    menu.add(afficherEmployes(ligue));
    menu.add(ajouterEmploye(ligue));
    menu.add(selectEmploye(ligue));
    menu.add(supprimerEmploye(ligue));
    menu.addBack("q");
    return menu;
}
```


MODÉLISATION DE L'INTERFACE GRAPHIQUE AVEC DES MAQUETTES



CRÉATION D'UNE CLASSE FILLE POUR GÉRER LA CONNEXION À LA BASE DE DONNÉES AVEC JDBC ET INSTALLATION DE LA BDD SUR LE SERVEUR(172.16.0.2)

```
1 + package sql;
2 +
3 + import java.sql.Connection;
4 + import java.sql.DriverManager;
5 + import java.sql.SQLException;
6 +
7 + public class DBConnect {
8 +
9 +     private static final String url="jdbc:mysql://localhost:3306/ynguetche?
    useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serve
    rTimezone=UTC";
10 +     private static final String username="root";
11 +     private static final String password="";
12 +
13 +     public static Connection getConnection() throws SQLException {
14 +         return DriverManager.getConnection(url, username, password);
15 +
16 +     }
17 + }
```

```
1 + package sql;
2 + import java.sql.*;
3 + import java.time.LocalDate;
4 +
5 + import personnel.Employe;
6 + import personnel.GestionPersonnel;
7 + import personnel.SauvegardeImpossible;
8 +
9 + public class DatabaseConnexion implements personnel.Passerelle{
10 +
11 +     @Override
12 +     public GestionPersonnel getGestionPersonnel() {
13 +         return null;
14 +     }
15 +
16 +     @Override
17 +     public void sauvegarderGestionPersonnel(GestionPersonnel gestionPersonnel)
    throws SauvegardeImpossible {
18 +
19 +
20 +     static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
```