

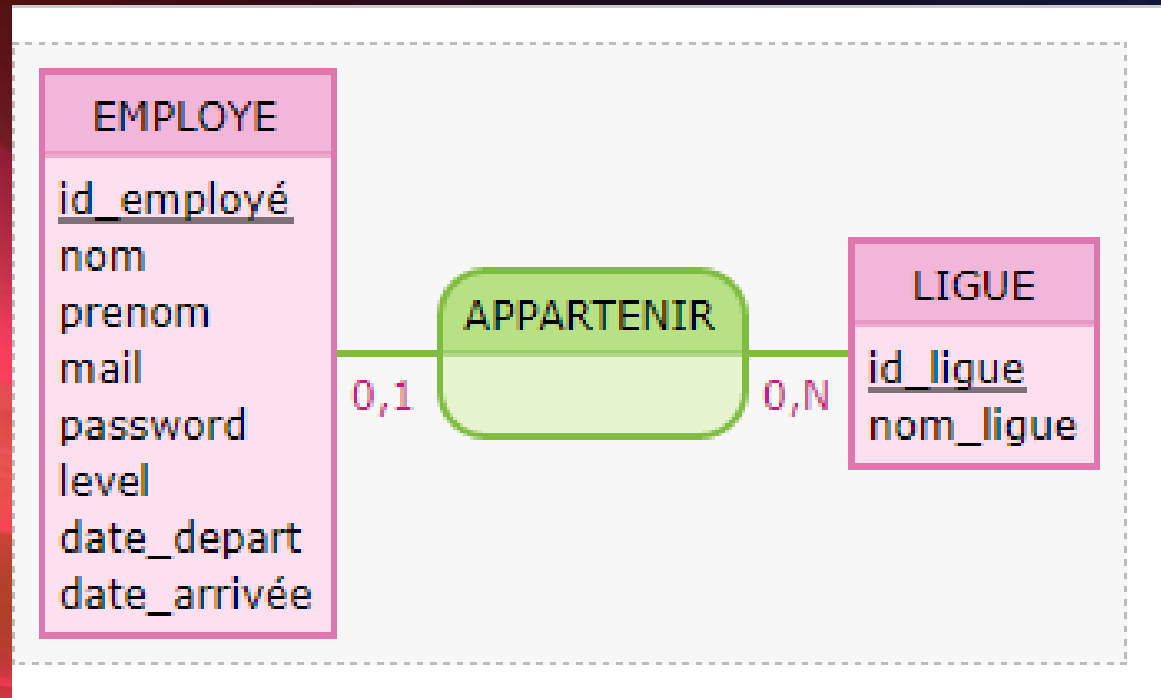
PPE - GESTION DU PERSONNEL DES LIGUES

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

PARTAGE DE TACHES

The screenshot displays a Trello board interface for a project named 'personnel'. The board is organized into four columns representing different stages of task completion: 'A faire' (To do), 'En cours' (In progress), 'Terminé' (Completed), and 'A completer' (To complete). Each column contains task cards with progress bars, titles, and status indicators (S for 'Seen', YN for 'Yet to be done').

Board Details:

- Board Name:** personnel
- Project:** Java project
- Visibility:** Visible par les membres d'une équipe
- Buttons:** S, YN, Inviter
- Butler:** Butler

Task Cards:

- A faire:**
 - 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
- En cours:**
 - finalisation du menu (SelectEmploye)
- Terminé:**
 - testUnitaires
 - faire le MCD
 - faire un arbre heuristique
 - faire la gestion des dates
 - creation de la base de donnée
- A completer:**
 - gestion des dates dans le dialogue en ligne de commande

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;
}
```