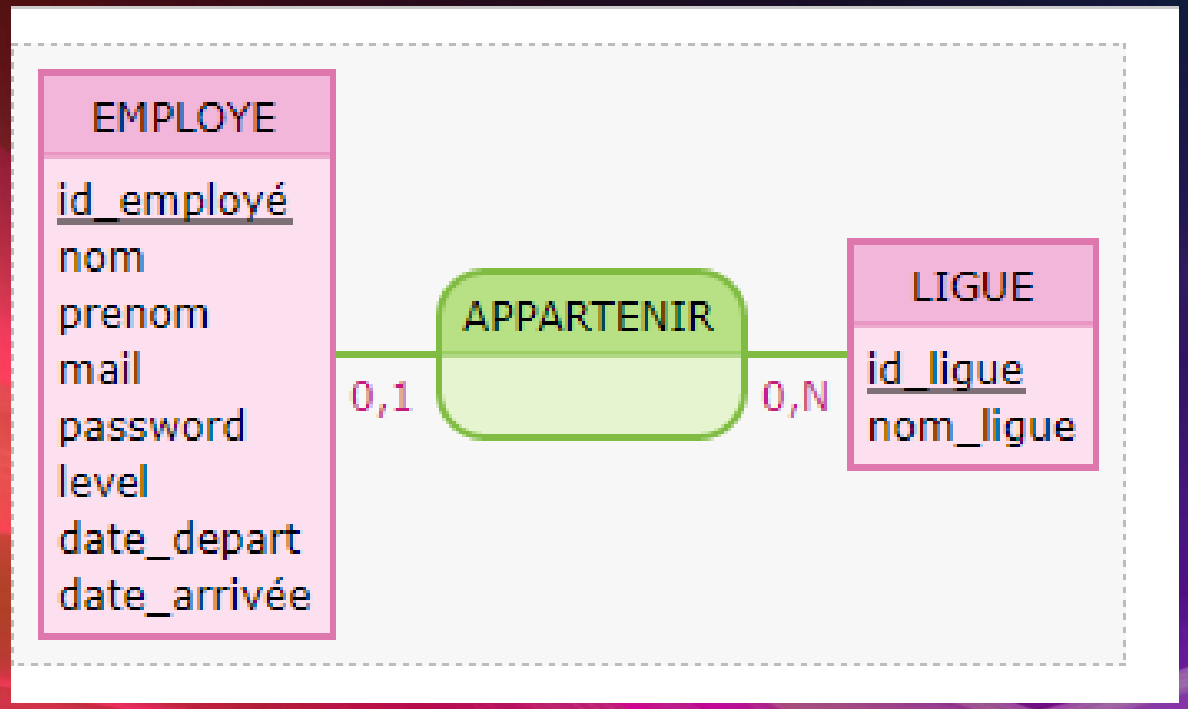
PPE - GESTION DU PERSONNEL DES LIGUES

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

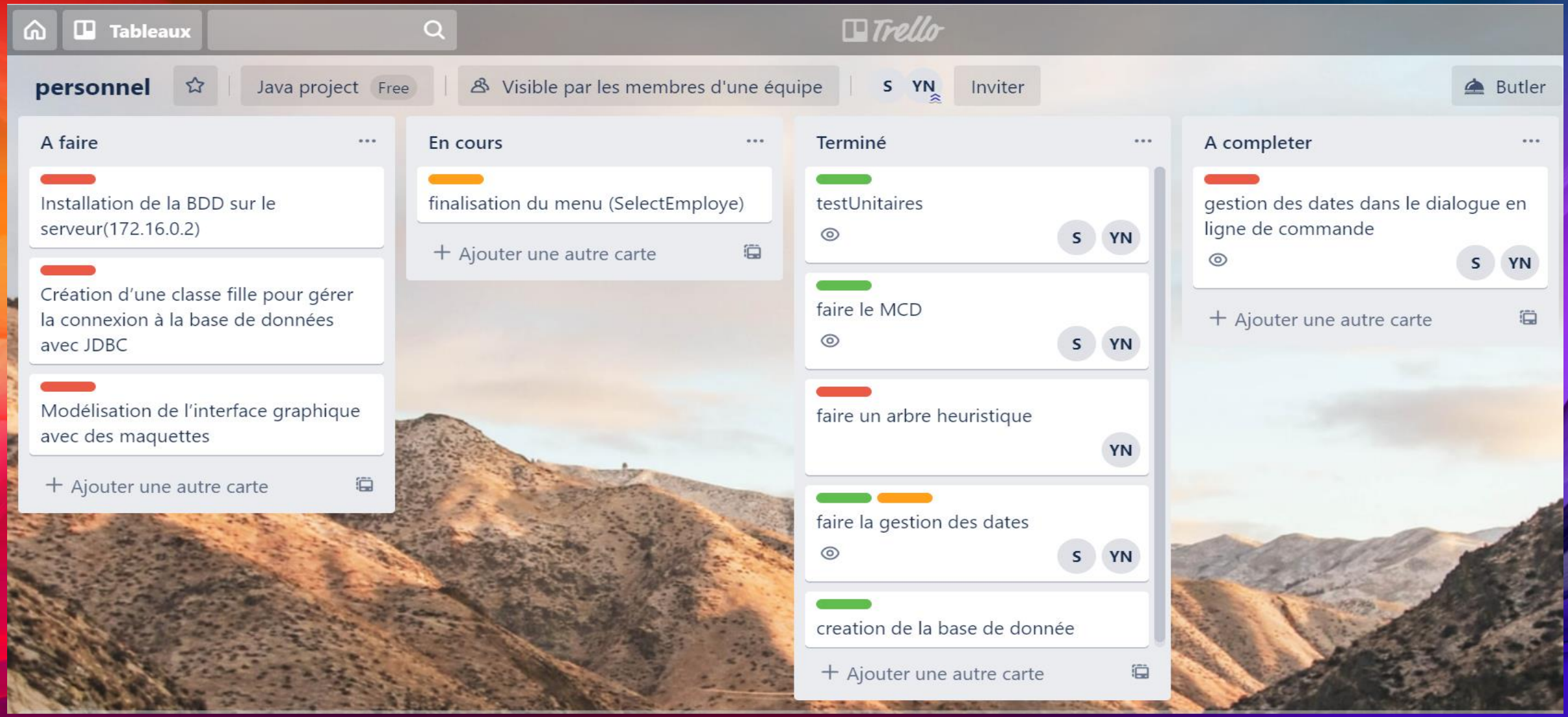
PPE - GESTION DU PERSONNEL DES LIGUES




```
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



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

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 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));
        menu.add(changerDateArrival(employe));
        menu.add(changerDateDepart(employe));
        menu.addBack("q");
        return menu;
}

private Option changerDateDepart( final Employe employe) {

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

private Option changerDateArrival(final Employe employe) {

        return new Option ("changer la date de départ ", "y",() ->  {employe.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;
}
```