GESTION DU PERSONNEL

Application permettant de gérer des employés dans des ligues et leur administrateur.

Diariyata & Edomiyas
SIO2 SLAM

SOMMAIRE

PRÉSENTATION DU PROJET

- Le projet consiste à mettre en place une application pour gérer les employés des ligues. Cette application, très simple, n'existe qu'en ligne de commande et est mono-utilisateur.
- Grâce à la création et la mise en place d'une base de données, L'application doit être rendue multi-utilisateur.

CAHIER DES CHARGES 1/2

- **Itération 1 :**

- 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).

- **Itération 2 :**

- Création de la base de données.
- Gestion des dates dans le dialogue en ligne de commande.
- Dans le dialogue en ligne de commande, un employé doit être sélectionné avant que l'on puisse choisir de modifier ou de supprimer.
- Possibilité de changer l'administrateur d'une ligue en ligne de commande.

CAHIER DES CHARGES 2/2

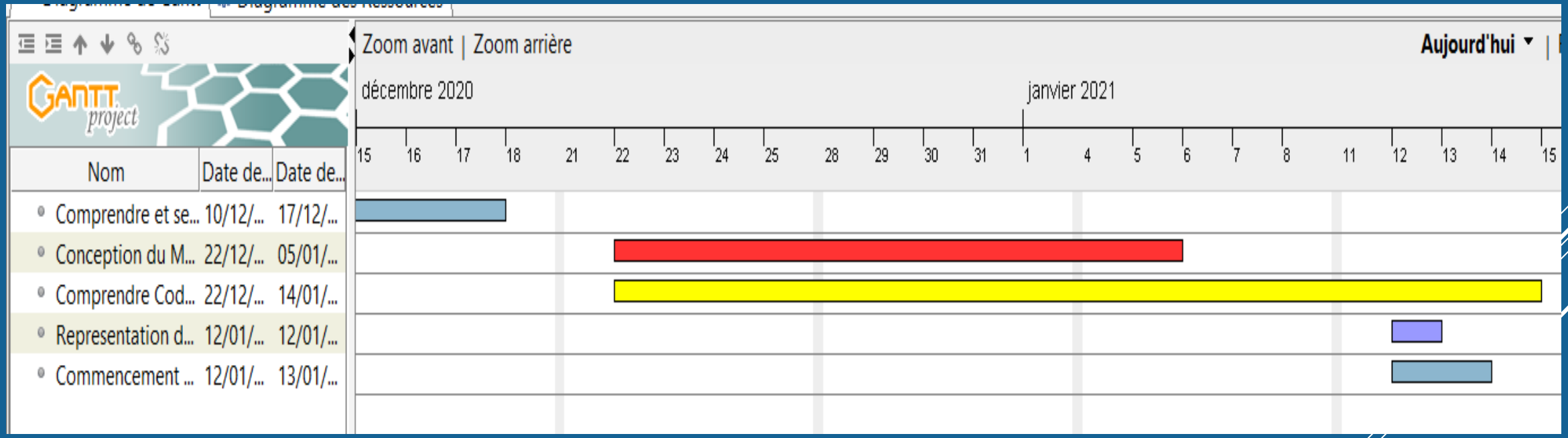
- **Itération 3 :**
- Création d'une classe fille de Passerelle permettant de gérer la connexion à la base de données avec JDBC (ou avec Hibernate si vous le souhaitez).
- Modélisation de l'interface graphique avec des maquettes.
- Possibilité de changer l'administrateur d'une ligue en ligne de commande.

- **Itération 4 :**
- 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.

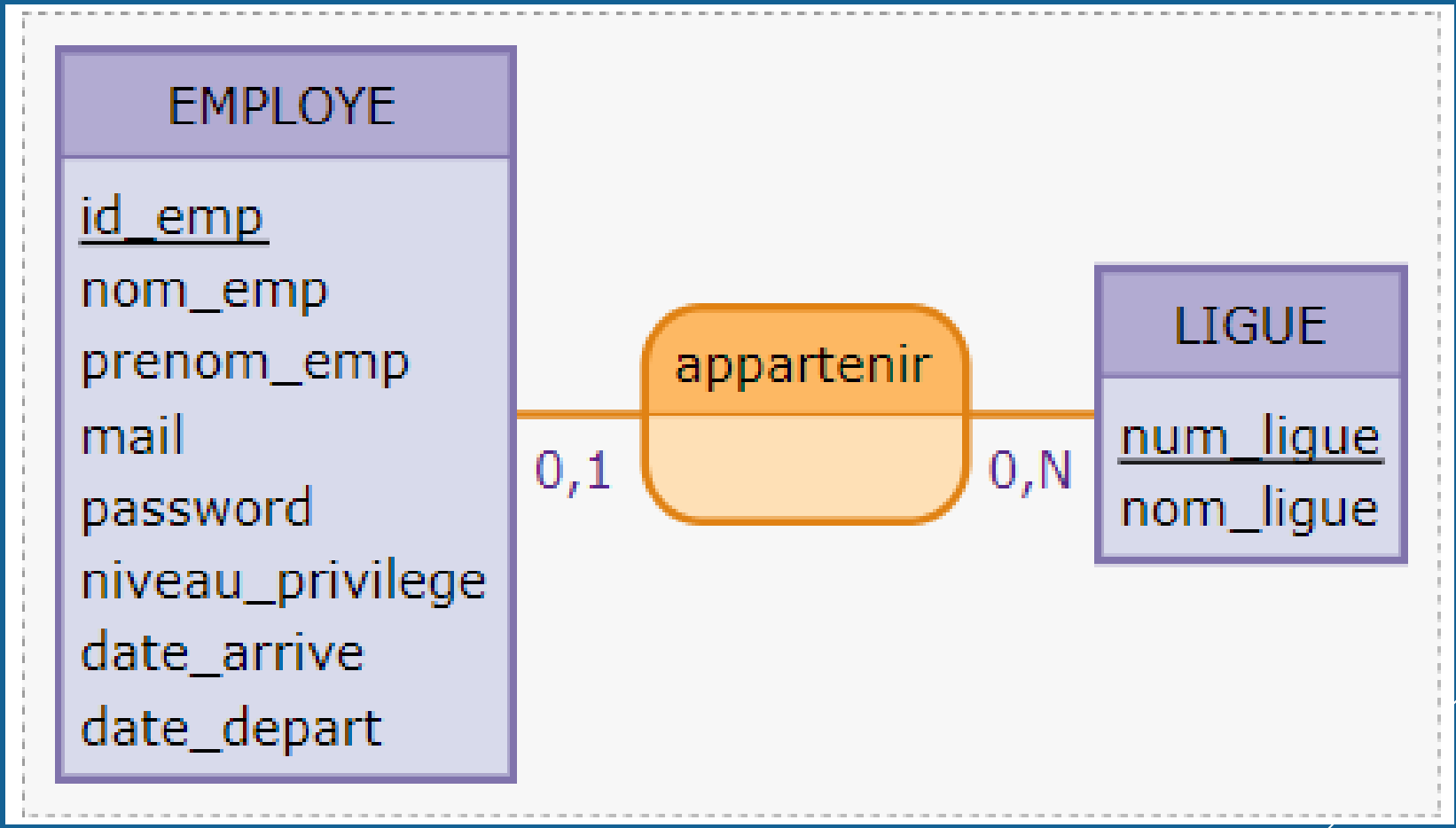
LES OUTILS TECHNOLOGIQUES À UTILISER

- Conserver java via eclipse pour l'application.
- Utiliser le versionnement avec git : Afin de communiquer au mieux entre nous concernant les modifications apporter au code, nous avons utiliser GITHUB pour faire les commit, push le code avec les messages de mises à jour afin que les autres puissent le pull avec compréhension.
- Utiliser la bibliothèque de dialogue en ligne de commande fournie.

DIAGRAMME DE GANTT



CONCEPTION DE MCD POUR LA STRUCTURE DE LA BASE DE DONNÉES



GESTION DES DATES

La consigne nous demandais de gérer les dates de départ et d'arrivée de chaque employé (couche métier + tests unitaires). Pour cela nous avons :

- Ajouter les objets Date dans le classes Employé en précisant leur type LocalDate
- Ensuite, lors de l'implémentation nous avons désigné explicitement les objets à l'aide du mot clé « THIS » qui lui est remplacé au moment de l'appel par l'objet auquel est adressé l'appel. (exception, quand un terme n'est ni une variable ni un paramètre)
- Pour finir, nous avons fait l'accès au attributs grâce aux accesseurs GET en lecture, pour ouvrir & SET en écriture pour modifier

```java
private LocalDate dateDepart, dateArrivee;

Employe(GestionPersonnel gestionPersonnel, Ligue ligue, String nom, String prenom, String mail, String pas
{
    this.gestionPersonnel = gestionPersonnel;
    this.nom = nom;
    this.prenom = prenom;
    this.password = password;
    this.mail = mail;
    this.ligue = ligue;
    this.dateArrivee = dateArrivee;  //Itération 1
    this.dateDepart = dateDepart;  //Itération 1


/*Exception pour les dates */
    try {
        this.dateArrivee = dateArrivee;
        this.dateDepart = dateDepart;
    }
    catch(Exception e){
        System.out.println(e.getMessage());
    }
```

```java
public LocalDate getDateArrivee() {  //Getter pour la date d'arrivée
    return dateArrivee;
}

public void setDateArrivee(LocalDate dateArrivee){  //Setter pour la date d'arrivée
    this.dateArrivee = dateArrivee;
}

public LocalDate getDateDepart() {  //Getter pour la date de départ
    return dateDepart;
}

public void setDateDepart(LocalDate dateDepart){  //Setter pour la date de départ
    this.dateDepart = dateDepart;
}
```

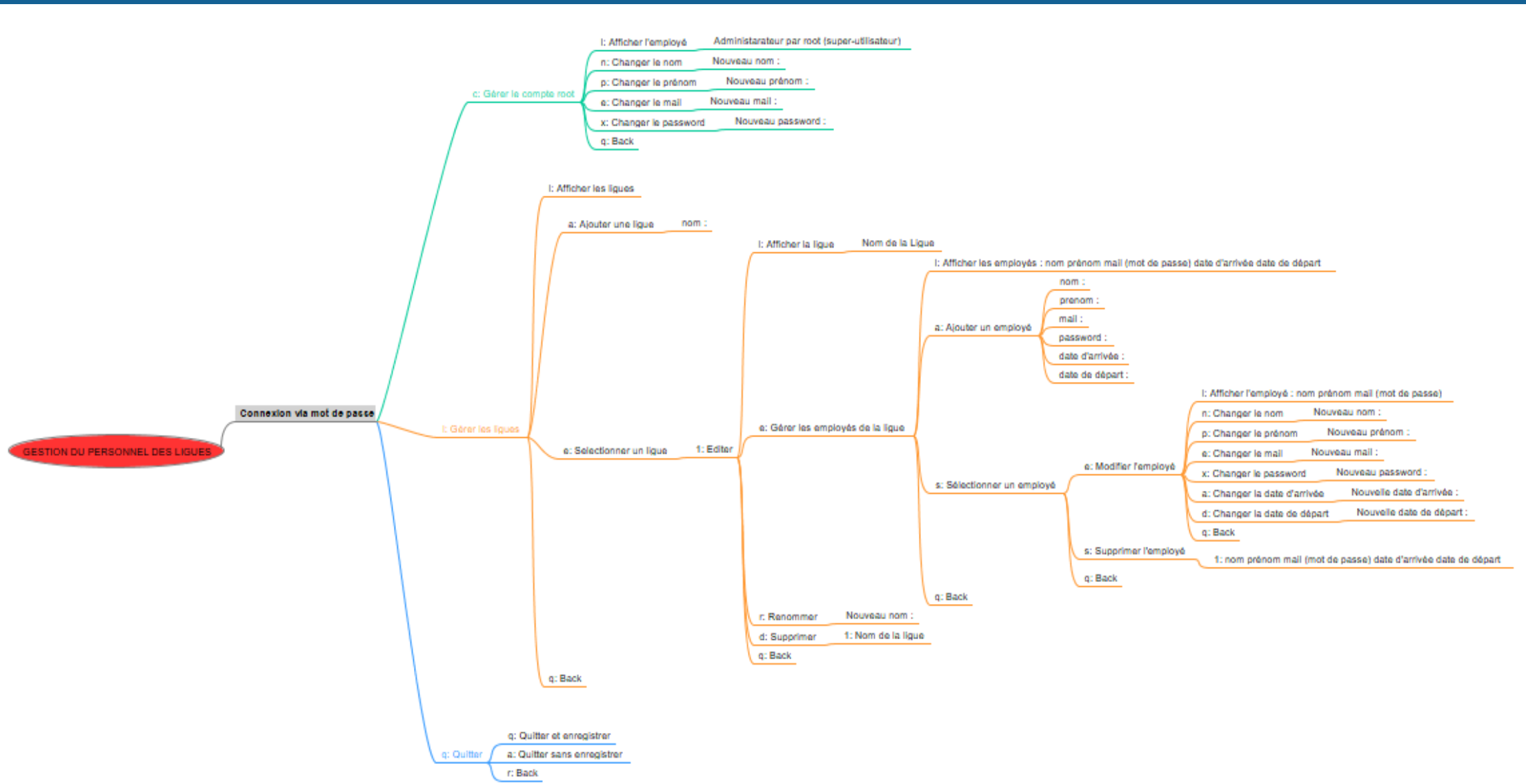
TESTS ET OPTIMISATION DU CODE SOURCE JAVA

Les tets unitaires nous on permis de vérifier le bon fonctionnement d'une partie précise du programme de gestion des ligues.

```java
class testGestionPersonnel {
    GestionPersonnel gestionPersonnel = GestionPersonnel.getGestionPersonnel();

    // Itération 1 test unitaire Gestion personnel
    @Test
    void testAddLigue() throws SauvegardeImpossible{
        Ligue ligue = new Ligue(gestionPersonnel, "ligueOne");
        assertEquals("ligueOne", ligue.getNom());
    }

}
```

```java
class testEmploye {
    GestionPersonnel gestionPersonnel = GestionPersonnel.getGestionPersonnel();

    // Itération 1 test unitaire employé
    @Test
    void getPrenom() throws SauvegardeImpossible{
        Ligue ligue = gestionPersonnel.addLigue("ligueOne");
        Employe employe = ligue.addEmploye("John", "Doe", "Johndoe@mail.com", "admin", null, null);
        assertEquals("Doe", employe.getPrenom());
    }

    @Test
    void setPrenom() throws SauvegardeImpossible{
        Ligue ligue = gestionPersonnel.addLigue("ligueOne");
        Employe employe = ligue.addEmploye("John", "Doe", "Johndoe@mail.com", "admin", null, null);
        employe.setPrenom("Bale");
        assertEquals("Bale", employe.getPrenom());
    }
```

ARBORESCENCE DES MENUS



CRÉATION DE LA BASE DE DONNÉES

```
--                Génération d'une base de données pour
--                            MySQL
--                        (20/1/2021 13:00:00)
-- ------------------------------------------------------
--       Nom de la base : GestionPersonnel
--       Projet : PPE application JAVA
--       Auteur : Diariyata & Edomiyas
--       Date de dernière modification :
-- ------------------------------------------------------


-- ------------------------------------------------------
--         CREATION DE LA BASE
-- ------------------------------------------------------

CREATE DATABASE
GestionPersonnel;

-- ------------------------------------------------------
--         TABLE : Employe
-- ------------------------------------------------------

CREATE TABLE Employe
    (
     id_employe INT(11) NOT NULL AUTO_INCREMENT,
     nom_employe VARCHAR(40) NULL   ,
     prenom_employe VARCHAR(50) NULL   ,
     mail VARCHAR(100) NULL   ,
     password VARCHAR(11) NULL   ,
     niveau_privilège VARCHAR(25) NULL   ,
     date_arrivee DATE NOT NULL   ,
     date_depart DATE NULL ,
      PRIMARY KEY (id_employe)
    );

-- ------------------------------------------------------
--         TABLE : Ligue
-- ------------------------------------------------------

CREATE TABLE Ligue
  (
    num_ligue INT(11) NOT NULL AUTO_INCREMENT,
    nom_ligue VARCHAR(35) NULL   ,
     PRIMARY KEY (num_ligue)
  );
```

	Table	Action	Lignes	Type	Interclassement	Taille	Perte
☐	employe	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	16,0 kio	-
☐	ligue	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	16,0 kio	-
	2 tables	Somme	0	InnoDB	utf8mb4_general_ci	32,0 kio	0 o

GESTION DES EMPLOYÉS

- Dans le dialogue en ligne de commande, un employé doit être sélectionné avant que l'on puisse choisir de modifier ou de supprimer.

```
// Itération 2
private Menu editerEmploye(Ligue ligue) {
    Menu menu = new Menu("Editer " + ligue.getEmployes());
    menu.add(modifierEmploye(ligue));
    menu.add(supprimerEmploye(ligue));
    return menu;
}
```

```
// Itération 2 Sélectionner un employé avant de modifier
private List<Employe> selectionnerEmploye(Ligue ligue)
{
    return new List<>("Sélectionner un employé", "s",
            () -> new ArrayList<>(ligue.getEmployes()),
            employeConsole.editerEmploye()
            );
}
```

GESTION DES DATES DANS LES MENUS

```java
private Option ajouterEmploye(final Ligue ligue)
{
    return new Option("ajouter un employé", "a",
            () ->
            {
                ligue.addEmploye(getString("nom : "),
                    getString("prenom : "), getString("mail : "),
                    getString("password : "), LocalDate.parse(getString("Date d'arrivée (YYYY-MM-DD) : ")),
                    LocalDate.parse(getString("Date de départ (YYYY-MM-DD) : ")));
        /*Itération 2 Ajout des dates */
                //  LocalDate.parse(getString("Date d'arrivée (YYYY-MM-DD) : ")), LocalDate.parse(getString("Date de départ (YYYY-MM-DD) : "));
            }
    );
}
```

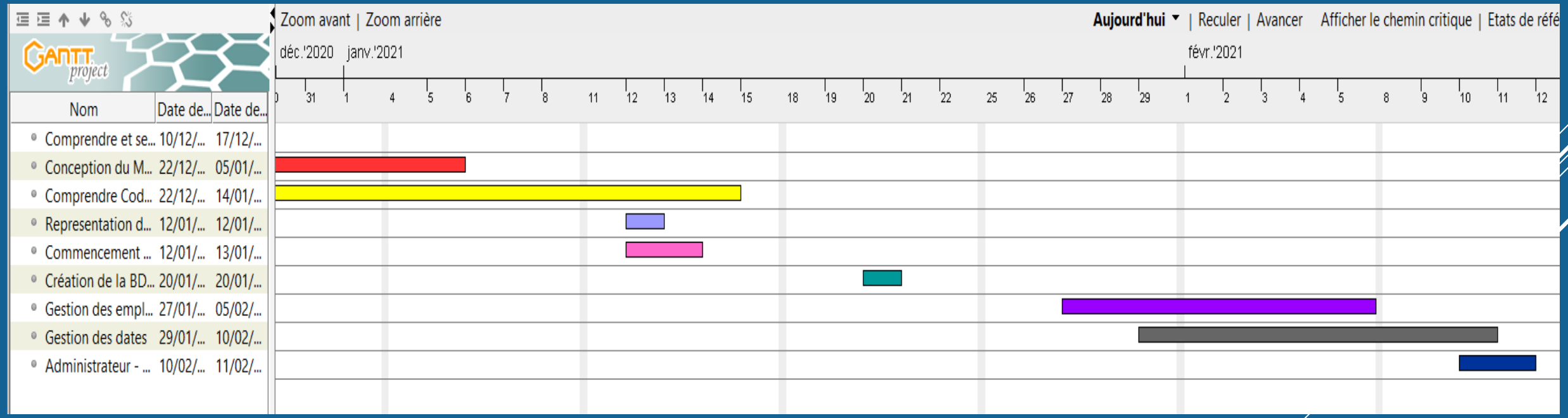
```
Select an option : a
nom : ed
prenom : te
mail : edte
password : eee
Date d'arrivée (YYYY-MM-DD) : 2020-02-02
Date de départ (YYYY-MM-DD) : 2020-03-01

---------------------------
```

CHANGEMENT ADMIN

Pour cette étape, nous devions créer une méthode nous permettant de pouvoir changer l'administrateur d'une ligue en ligne de commande

```
private List<Employe> changerAdministrateur(final Ligue ligue, String key)
{
    List<Employe> liste = new List<>("Changer d'administrateur", key,
            () -> new ArrayList<>(ligue.getEmployes()),
            (index, element) -> {
                ligue.setAdministrateur(element);
                gestionPersonnel.changerAdmin(element);
            });
    liste.addBack("q");
    return liste;
}
```

GANTT PHASE 2

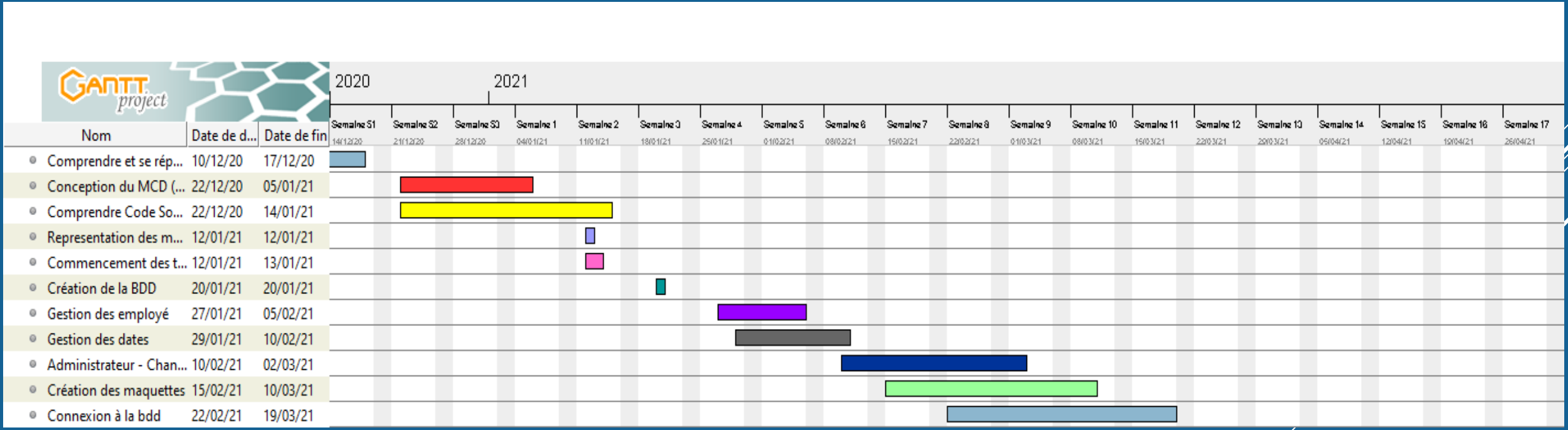


CRÉATION DE LA MAQUETTE

CONNEXION À LA BDD

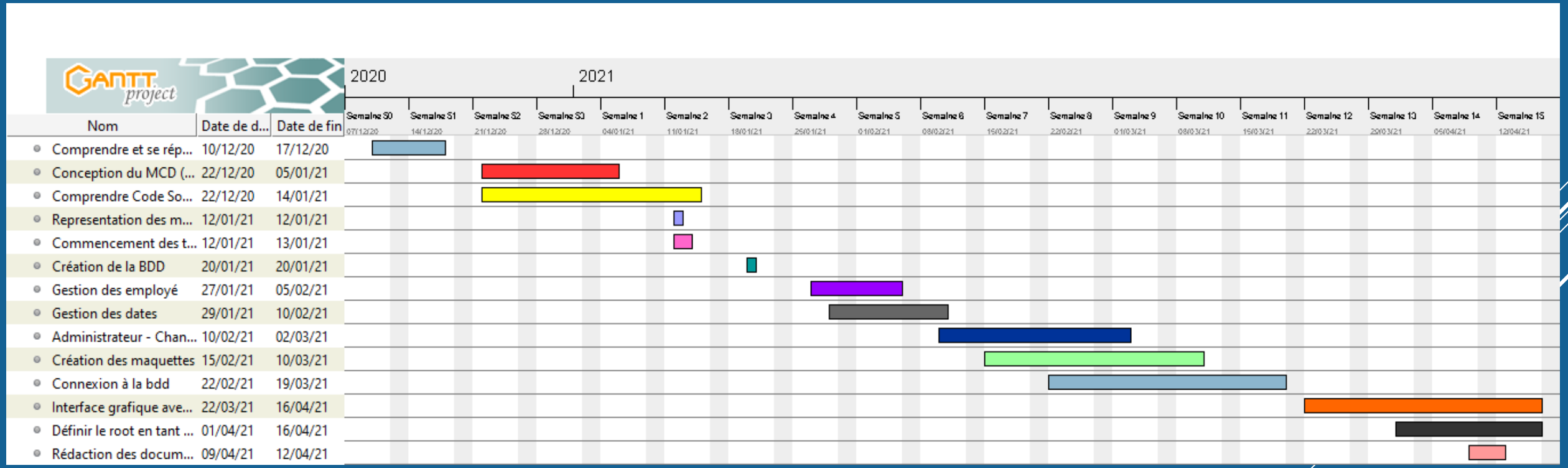
```
public class GestionPersonnel implements Serializable
{
    private static final long serialVersionUID = -10528311398788642 5L;
    private static GestionPersonnel gestionPersonnel = null;
    private SortedSet<Ligue> ligues;
    private Employe root = new Employe(this, null, "root", "", "", "toor", null, null);
    public final static int SERIALIZATION = 1, JDBC = 2,
            TYPE_PASSERELLE = JDBC;
    private static Passerelle passerelle = TYPE_PASSERELLE == JDBC ? new jdbc.JDBC() : new serialisation.Se
```

GANTT PHASE 3



ROOT DEVIENT ADMIN

GANTT PHASE 4



Nom	Date de d...	Date de fin
Comprendre et se rép...	10/12/20	17/12/20
Conception du MCD (...	22/12/20	05/01/21
Comprendre Code So...	22/12/20	14/01/21
Representation des m...	12/01/21	12/01/21
Commencement des t...	12/01/21	13/01/21
Création de la BDD	20/01/21	20/01/21
Gestion des employé	27/01/21	05/02/21
Gestion des dates	29/01/21	10/02/21
Administrateur - Chan...	10/02/21	02/03/21
Création des maquettes	15/02/21	10/03/21
Connexion à la bdd	22/02/21	19/03/21
Interface grafique ave...	22/03/21	16/04/21
Définir le root en tant ...	01/04/21	16/04/21
Rédaction des docum...	09/04/21	12/04/21