ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA

UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Rapport du Projet
Programmation Orientée Objet Java
Filière :
« Génie du Logiciel et des Systèmes Informatiques Distribués »
GLSID

La mise en place d'un système De gestion et de planification des projets

Réalisé par :
Abdelmalek **ENNANI**
Achraf **TAFFAH**

Encadré par :
M. Abdelmajid **BOUSSELHAM**

Année Universitaire : 2022-2023

Introduction

En tant qu'étudiants en deuxième année cycle d'ingénieur option Génie du Logiciel et des Systèmes Informatiques Distribués à l'Ecole Normale Supérieure de l'Enseignement Technique Mohammedia, nous sommes amenés à réaliser un projet en java, pour cela nous avons décidé de réaliser un projet sous le thème la mise en place d'un système de gestion et de planification des projets.

Dans ce sens, notre mission consiste à la mise en place et la réalisation d'un système de gestion des réservations des consultations en ligne entre les entreprises de vente et les acteurs commerciaux, donc nous somme charger d'assurer les étapes d'analyse, de conception et de la réalisation de l'application en utilisant le langage de modélisation UML, ainsi que le langages orienté objet Java avec les bibliothèques **JavaFx** et **JDBC**.

Ce rapport présente donc les phases que nous avons suivies pour la réalisation de ce projet.

Table des matières

Chapitre I. Conception de l'application

I.1 Identification des acteurs

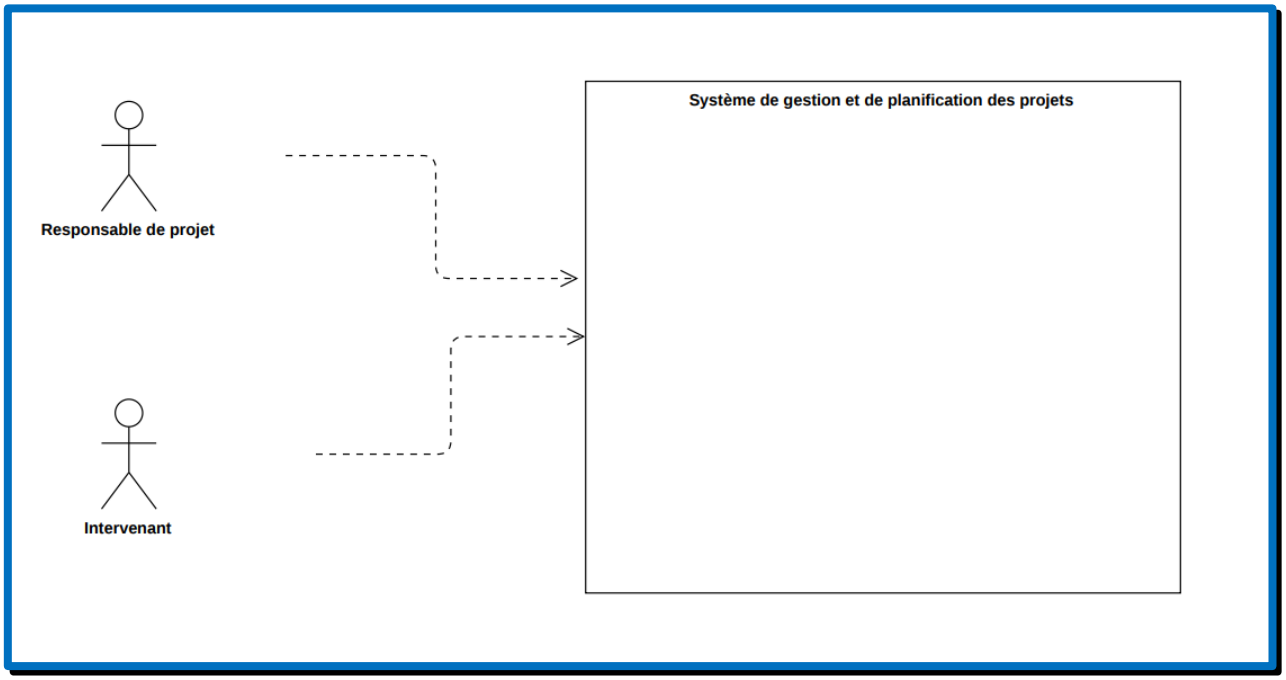
Un acteur d'un système est une entité externe à ce système qui interagit avec lui. Les acteurs permettent de cerner l'interface que le système va offrir à son environnement. Chaque acteur du système est un profil pouvant factoriser plusieurs personnes ayant les mêmes droits. Nous avons capturé les acteurs qui vont interagir avec notre système, comme le montre ce tableau pour l'identification des acteurs.

Nom de l'acteur	Rôle
Responsable de projet	Le responsable de projet est une personne qui peut : • Gérer ses informations personnelles. • Ajouter, modifier ou supprimer les comptes utilisateurs. • Ajouter, modifier ou supprimer une tâche. • Affecter des ressources humaines (intervenant) et matérielles à une tâche. • Planifier et visualiser dans un diagramme de Gantt les tâches du projet. • Importer et exporter toutes les données de l'application
Intervenant	L'intervenant est une personne qui peut : • Gérer ses informations personnelles. • Afficher ses ordres de travail. • Modifier l'état d'une tâche. • Recevoir une notification (Email et SMS).

I.2 Diagramme de contexte

Le diagramme de contexte est une représentation UML, il modélise le système sous forme d'une boîte noire en interaction avec les acteurs du système. Ce diagramme vient avant les diagrammes de cas d'utilisation. L'étude conceptuelle nous a permis d'élaborer les diagrammes de contexte ci-dessous :

Ce diagramme de contexte est pour système de gestion et de planification des projets. Il contient deux acteurs, le responsable de projet et l'intervenant.

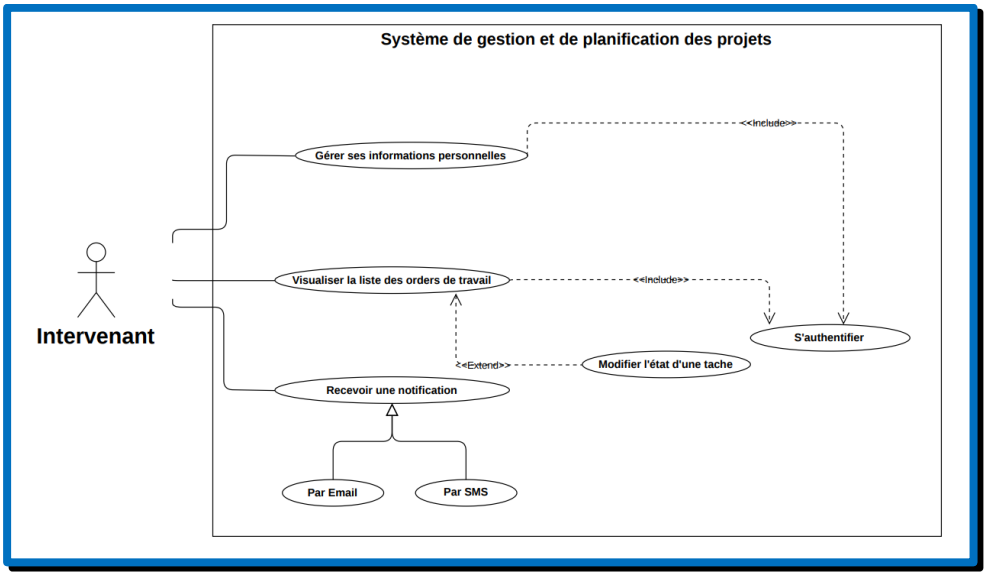


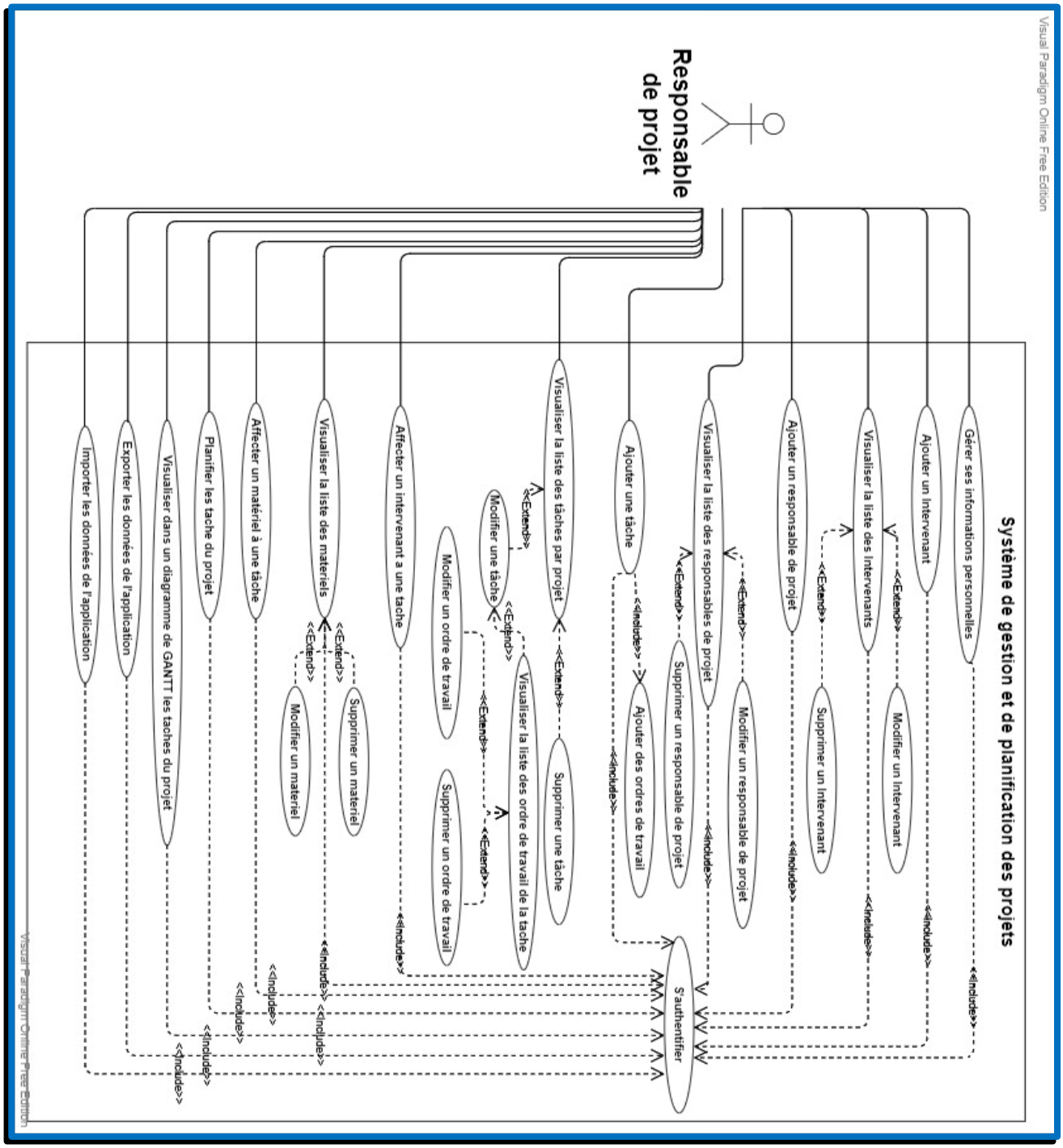
I.3 Diagramme de cas d'utilisation

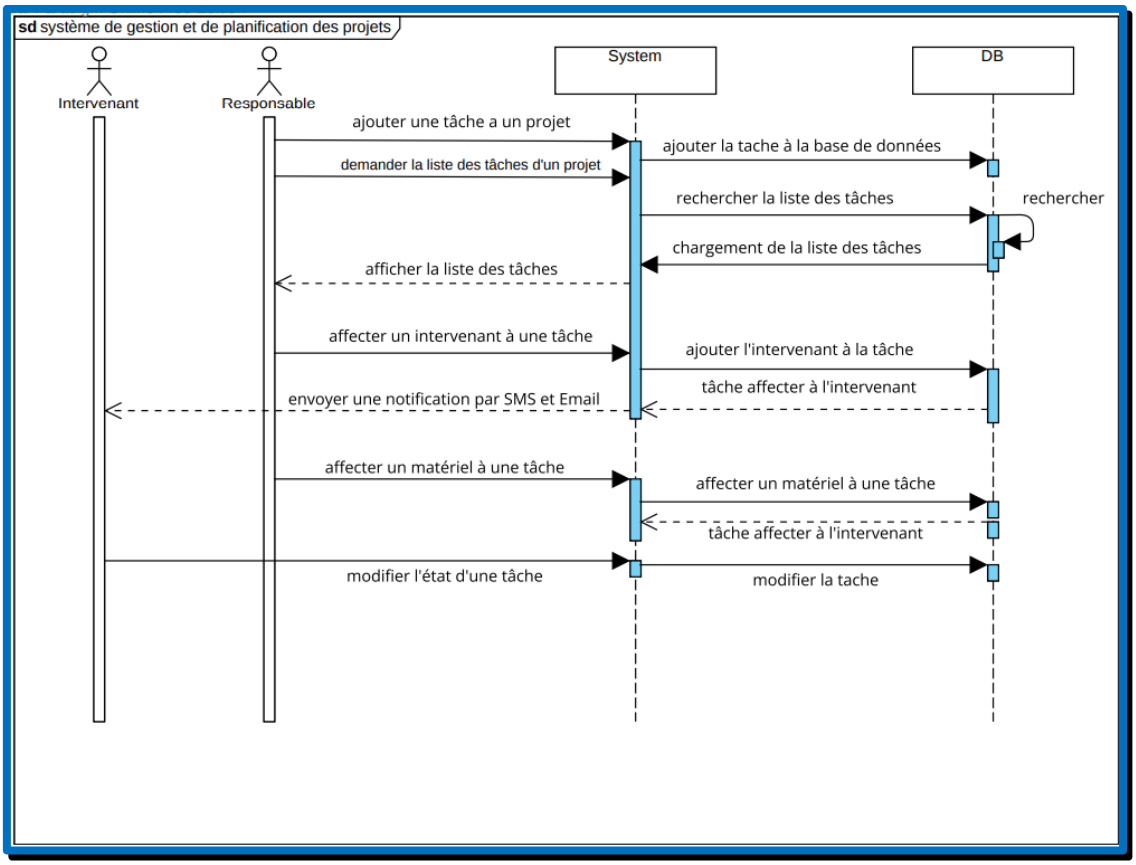
Les diagrammes de cas d'utilisation (DCU) sont des diagrammes UML utilisés pour une représentation du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés.

L'étude de différentes fonctionnalités et l'analyse des actions que les acteurs peuvent exécuter nous a permis d'élaborer les diagrammes ci-dessous :

Ces deux Diagramme de cas d'utilisation pour système de gestion et de planification des projets, permet de voir les différentes fonctionnalités disponibles pour les différents acteurs, les cas d'utilisation montrés sur ces figures au-dessous.

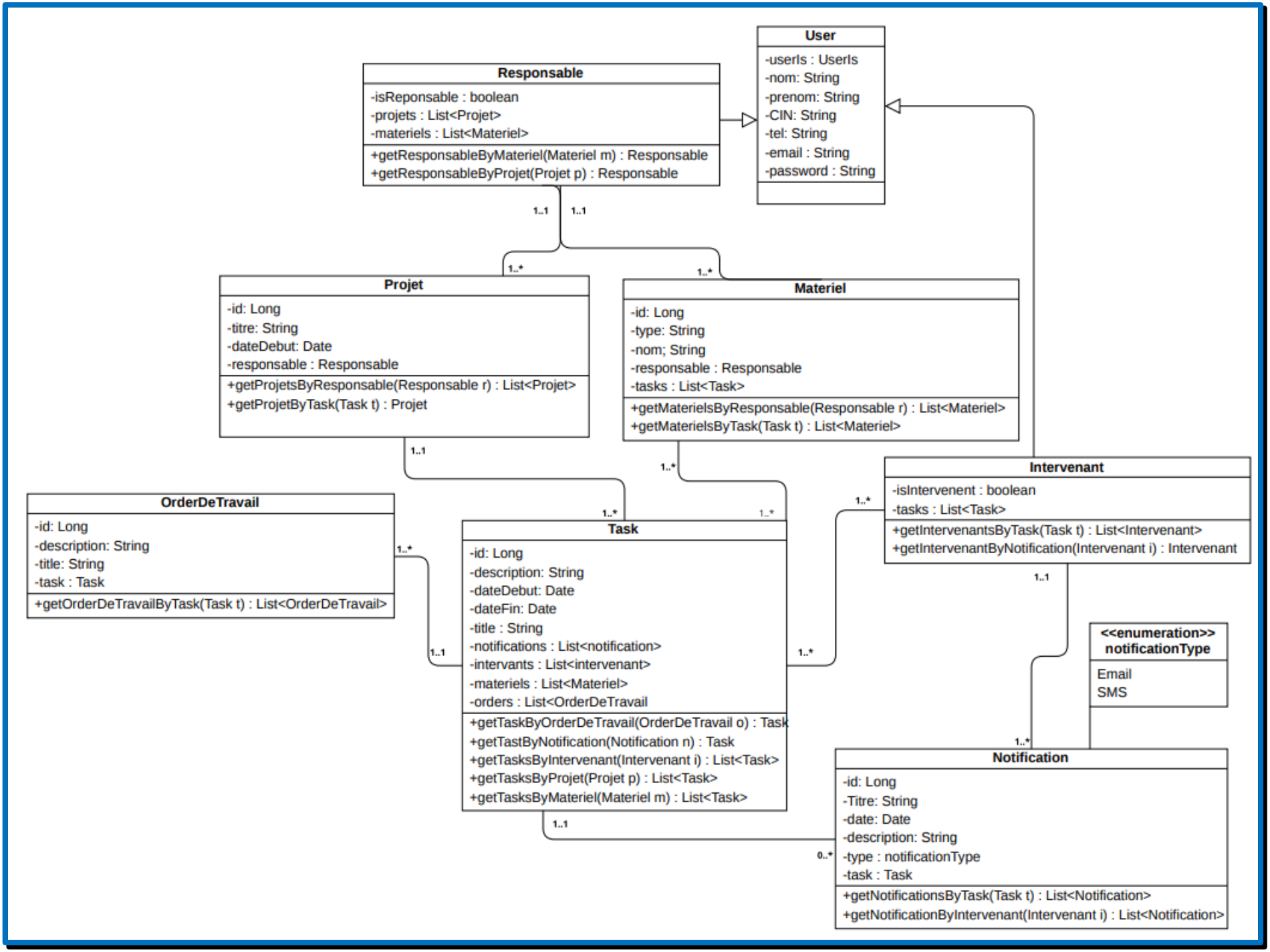



Système de gestion et de planification des projets
Responsable de projet
Gérer ses informations personnelles
Ajouter un intervenant
Visualiser la liste des Intervenants
Modifier un Intervenant
Supprimer un Intervenant
Ajouter un responsable de projet
Visualiser la liste des responsables de projet
Modifier un responsable de projet
Supprimer un responsable de projet
Ajouter une tâche
Ajouter des ordres de travail
Visualiser la liste des tâches par projet
Supprimer une tâche
Visualiser la liste des ordre de travail de la tache
Modifier une tâche
Modifier un ordre de travail
Supprimer un ordre de travail
Affecter un intervenant a une tache
Visualiser la liste des materiels
Supprimer un materiel
Modifier un materiel
Affecter un matériel à une tâche
Planifier les tache du projet
Visualiser dans un diagramme de GANTT les taches du projet
Exporter les données de l'application
Importer les données de l'application
S'authentifier
<<Include>>
<<Extend>>
Visual Paradigm Online Free Edition



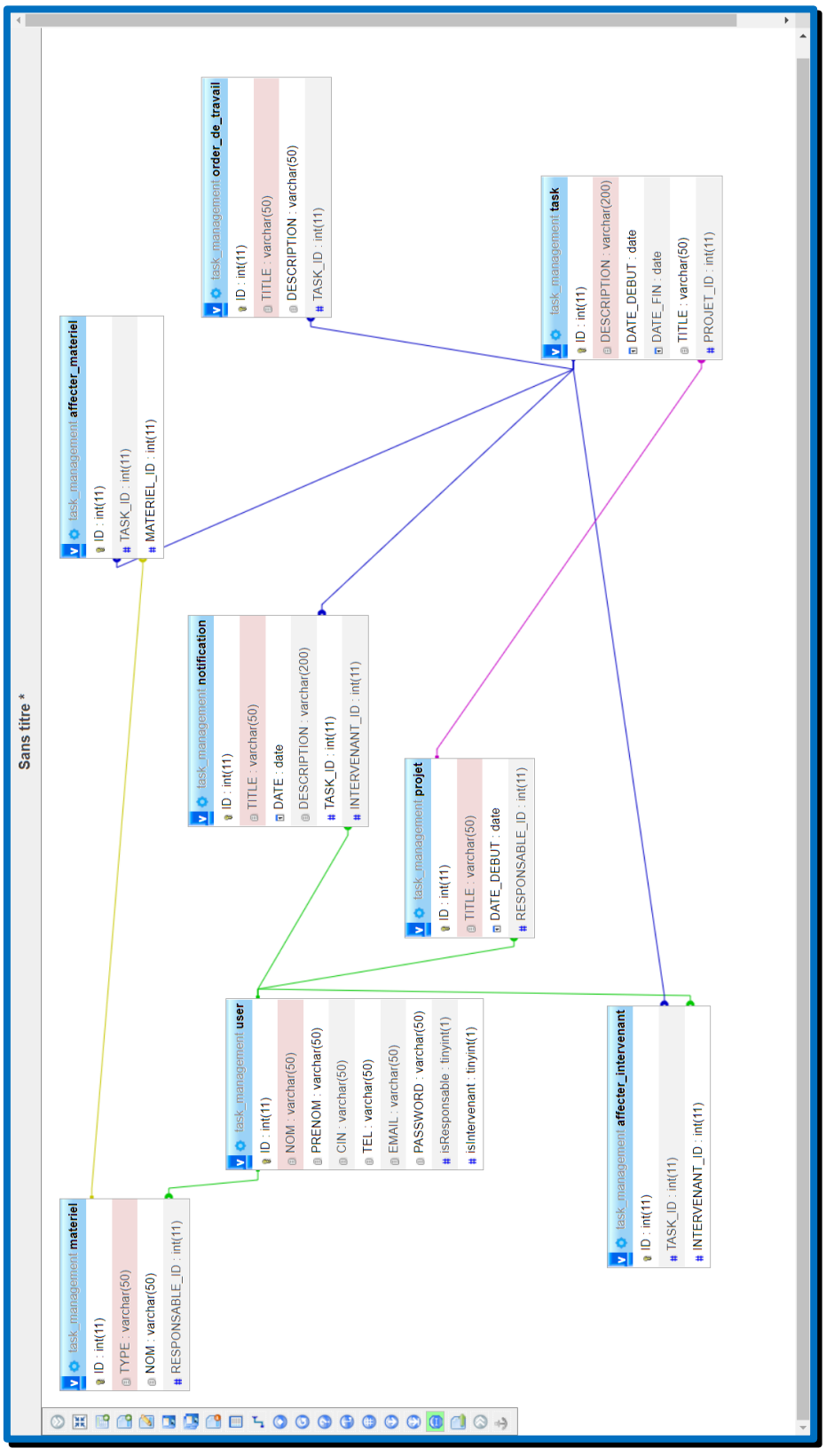
I.4 Diagramme de classe

Ce diagramme de classes pour le système de gestion et de planification des projets permet de visualiser les différentes classes existantes et les relations entre elles.


User
-userIs : UserIs
-nom: String
-prenom: String
-CIN: String
-tel: String
-email : String
-password : String
Responsable
-isReponsable : boolean
-projets : List<Projet>
-materiels : List<Materiel>
+getResponsableByMateriel(Materiel m) : Responsable
+getResponsableByProjet(Projet p) : Responsable
Projet
-id: Long
-titre: String
-dateDebut: Date
-responsable : Responsable
+getProjetsByResponsable(Responsable r) : List<Projet>
+getProjetByTask(Task t) : Projet
Materiel
-id: Long
-type: String
-nom; String
-responsable : Responsable
-tasks : List<Task>
+getMaterielsByResponsable(Responsable r) : List<Materiel>
+getMaterielsByTask(Task t) : List<Materiel>
Intervenant
-isIntervenent : boolean
-tasks : List<Task>
+getIntervenantsByTask(Task t) : List<Intervenant>
+getIntervenantByNotification(Intervenant i) : Intervenant
OrderDeTravail
-id: Long
-description: String
-title: String
-task : Task
+getOrderDeTravailByTask(Task t) : List<OrderDeTravail>
Task
-id: Long
-description: String
-dateDebut: Date
-dateFin: Date
-title : String
-notifications : List<notification>
-intervants : List<intervenant>
-materiels : List<Materiel>
-orders : List<OrderDeTravail
+getTaskByOrderDeTravail(OrderDeTravail o) : Task
+getTastByNotification(Notification n) : Task
+getTasksByIntervenant(Intervenant i) : List<Task>
+getTasksByProjet(Projet p) : List<Task>
+getTasksByMateriel(Materiel m) : List<Task>
<<enumeration>>
notificationType
Email
SMS
Notification
-id: Long
-Titre: String
-date: Date
-description: String
-type : notificationType
-task : Task
+getNotificationsByTask(Task t) : List<Notification>
+getNotificationByIntervenant(Intervenant i) : List<Notification>
1..1
1..*
0..*

I.5 Diagramme entité association

Ce diagramme pour le système de gestion et de planification des projets permet de visualiser les différentes relations entre les tables dans la base de données.



I.6 Conclusion

Dans ce chapitre, nous avons traité l'étude conceptuelle du projet. En premier lieu, nous avons précisé la méthodologie utilisée. Ensuite, nous avons identifié les différents acteurs et les diagrammes de contexte utilisés. Après, nous avons présenté les différents diagrammes de contexte, de cas d'utilisation et de classe pour le système.

Chapitre II. Réalisation et Mise en Œuvre

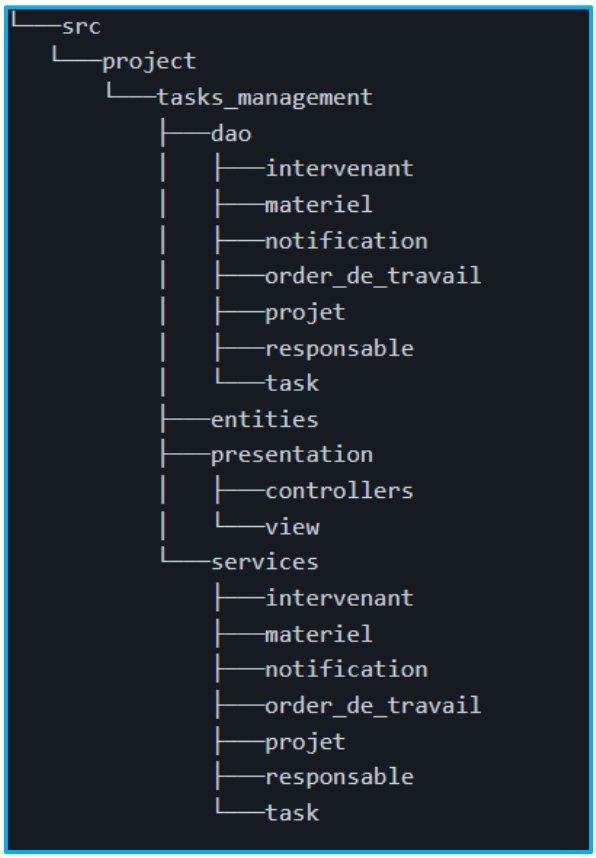
II.1 Introduction

Dans ce chapitre nous avons présenté les résultats de l'implémentation de l'ensemble des fonctionnalités citées dans le dernier chapitre pour le système de gestion et de planification des projets.

Nous avons devisé ce chapitre en deux parties, la premier pour présenter le code de l'application, et la deuxième partie pour présentation les interfaces de l'application.

II.2 Présentation du code

Notre projet se compose de trois couches, CAO, Service et Présentation, comme le montre l'image ci-dessus.



1) La couche **ENTITIES**

- La classe **Intervenant**

```
package project.tasks_management.entities;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class Intervenant extends User implements Serializable {
    private Boolean isIntervenant=true;
    private List<Task> TASK=new ArrayList<>();
    private List<Notification> NOTIFICATIONS=new ArrayList<>();
    public Intervenant(){
        super();
    }

    public Intervenant(long ID, String NOM, String PRENOM, String CIN, String
TEL,String EMAIL,String PASSWORD) {
        super(ID, NOM, PRENOM, CIN, TEL,EMAIL,PASSWORD);
    }

    public Boolean getIntervenant() {
        return isIntervenant;
    }

    public void setIntervenant(Boolean intervenant) {
        isIntervenant = intervenant;
    }

    public List<Task> getTASK() {
        return TASK;
    }

    public void setTASK(List<Task> TASK) {
        this.TASK = TASK;
    }

    public List<Notification> getNOTIFICATIONS() {
        return NOTIFICATIONS;
    }

    public void setNOTIFICATIONS(List<Notification> NOTIFICATIONS) {
        this.NOTIFICATIONS = NOTIFICATIONS;
    }

    @Override
    public String toString() {
        return super.toString()+"Intervenant{" +
                "isIntervenant=" + isIntervenant +
                ", TASK=" + TASK +
                ", NOTIFICATIONS=" + NOTIFICATIONS +
                '}';
    }
}
```

- La classe **Materiel**

```java
package project.tasks_management.entities;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class Materiel implements Serializable {
    private long ID=0;
    private String TYPE="";
    private String NOM="";
    private List<Task> TASK=new ArrayList<>();
    private Responsable RESPONSABLE=new Responsable();

    public Materiel() {}

    public Materiel(long ID, String TYPE, String NOM) {
        this.ID = ID;
        this.TYPE = TYPE;
        this.NOM = NOM;
    }

    public long getID() {
        return ID;
    }

    public void setID(long ID) {
        this.ID = ID;
    }

    public String getTYPE() {
        return TYPE;
    }

    public void setTYPE(String TYPE) {
        this.TYPE = TYPE;
    }

    public String getNOM() {
        return NOM;
    }

    public void setNOM(String NOM) {
        this.NOM = NOM;
    }

    public List<Task> getTASK() {
        return this.TASK;
    }

    public void setTASK(List<Task> TASK) {
        this.TASK = TASK;
    }

    public Responsable getRESPONSABLE() {
        return RESPONSABLE;
    }

    public void setRESPONSABLE(Responsable RESPONSABLE) {
        this.RESPONSABLE = RESPONSABLE;
    }
}
```

- La classe **Notification**

```java
package project.tasks_management.entities;

import java.io.Serializable;
import java.util.Date;

public class Notification implements Serializable {
    private long ID=0;
    private String TITLE="";
    private Date DATE=new Date();
    private String DESCRIPTION="";
    private notificationType type;
    private Task TASK=new Task();
    private Intervenant INTERVENENT=new Intervenant();

    public Notification() {}

    public Notification(long ID, String TITLE, Date DATE, String DESCRIP-
TION,Task TASK,Intervenant INTERVENENT) {
        this.ID = ID;
        this.TITLE = TITLE;
        this.DATE = DATE;
        this.DESCRIPTION = DESCRIPTION;
        this.TASK=TASK;
        this.INTERVENENT=INTERVENENT;
    }

    public enum notificationType{
        Email,
        SMS
    }

    public long getID() {
        return ID;
    }

    public void setID(long ID) {
        this.ID = ID;
    }

    public String getTITLE() {
        return TITLE;
    }

    public void setTITLE(String TITLE) {
        this.TITLE = TITLE;
    }

    public Date getDATE() {
        return DATE;
    }

    public void setDATE(Date DATE) {
        this.DATE = DATE;
    }

    public String getDESCRIPTION() {
        return DESCRIPTION;
    }

    public void setDESCRIPTION(String DESCRIPTION) {
        this.DESCRIPTION = DESCRIPTION;
    }

    public notificationType getType() {
```

```
            return type;
        }

        public void setType(notificationType type) {
            this.type = type;
        }

        public Task getTASK() {
            return TASK;
        }

        public void setTASK(Task TASK) {
            this.TASK = TASK;
        }

        public Intervenant getINTERVENENT() {
            return INTERVENENT;
        }

        public void setINTERVENENT(Intervenant INTERVENENT) {
            this.INTERVENENT = INTERVENENT;
        }
    }
```

- La classe **OrderDeTravail**

```
package project.tasks_management.entities;

import java.io.Serializable;
public class OrderDeTravail implements Serializable {
    private long ID=0;
    private String DESCRIPTION="";
    private String TITLE="";
    private Task task;

    public OrderDeTravail(){ }

    public OrderDeTravail(long ID, String DESCRIPTION, String TITLE, Task task) {
        this.ID = ID;
        this.DESCRIPTION = DESCRIPTION;
        this.TITLE = TITLE;
        this.task = task;
    }

    public long getID() {
        return ID;
    }

    public void setID(long ID) {
        this.ID = ID;
    }

    public String getDESCRIPTION() {
        return DESCRIPTION;
    }

    public void setDESCRIPTION(String DESCRIPTION) {
        this.DESCRIPTION = DESCRIPTION;
    }

    public String getTITLE() {
        return TITLE;
```

```
    }

    public void setTITLE(String TITLE) {
        this.TITLE = TITLE;
    }

    public Task getTask() {
        return task;
    }

    public void setTask(Task task) {
        this.task = task;
    }
}
```

- La classe **Projet**

```
package project.tasks_management.entities;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class Projet implements Serializable {
    private long ID=0;
    private String TITLE="";
    private Date date=new Date();
    private List<Task> TASKS=new ArrayList<>();
    private Responsable RESPONSABLE=new Responsable();

    public Projet() {}

    public Projet(long ID, String TITLE, Date date, User user,Responsable RE-
SPONSABLE) {
        this.ID = ID;
        this.TITLE = TITLE;
        this.date = date;
        this.RESPONSABLE=RESPONSABLE;
    }

    public long getID() {
        return ID;
    }

    public void setID(long ID) {
        this.ID = ID;
    }

    public String getTITLE() {
        return TITLE;
    }

    public void setTITLE(String TITLE) {
        this.TITLE = TITLE;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
```

```
        }

        public List<Task> getTASKS() {
            return TASKS;
        }

        public void setTASKS(List<Task> TASKS) {
            this.TASKS = TASKS;
        }

        public Responsable getRESPONSABLE() {
            return RESPONSABLE;
        }

        public void setRESPONSABLE(Responsable RESPONSABLE) {
            this.RESPONSABLE = RESPONSABLE;
        }
    }
```

- La classe **Responsable**

```
package project.tasks_management.entities;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class Responsable extends User implements Serializable {
    private boolean isResponsable=true;
    private List<Projet> PROJETS=new ArrayList<>();
    private List<Materiel> MATERIELS=new ArrayList<>();

    public Responsable(){
        super();
    }

    public Responsable(long ID, String NOM, String PRENOM, String CIN, String
TEL,String EMAIL,String PASSWORD) {
        super(ID, NOM, PRENOM, CIN, TEL,EMAIL,PASSWORD);
    }

    public boolean getIsResponsable() {
        return isResponsable;
    }

    public List<Projet> getPROJETS() {
        return PROJETS;
    }

    public void setPROJETS(List<Projet> PROJETS) {
        this.PROJETS = PROJETS;
    }

    public List<Materiel> getMATERIELS() {
        return MATERIELS;
    }

    public void setMATERIELS(List<Materiel> MATERIELS) {
        this.MATERIELS = MATERIELS;
    }
}
```

- La classe **Task**

```
package project.tasks_management.entities;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class Task implements Serializable {
    private long ID=0;
    private String DESCRIPTION="";
    private Date DATE_DEBUT=new Date();
    private Date DATE_FIN=new Date();
    private String TITLE="";
    private List<Materiel> MATERIELS=new ArrayList<>();
    private List<OrderDeTravail> ORDERS_DE_TRAVAIL=new ArrayList<>();
    private List<Notification> NOTIFICATIONS=new ArrayList<>();
    private List<Intervenant> INTERVENT=new ArrayList<>();
    private Projet PROJET=new Projet();

    public Task () {}

    public Task(long ID, String DESCRIPTION, Date DATE_DEBUT, Date DATE_FIN, String
TITLE) {
        this.ID = ID;
        this.DESCRIPTION = DESCRIPTION;
        this.DATE_DEBUT = DATE_DEBUT;
        this.DATE_FIN = DATE_FIN;
        this.TITLE = TITLE;
    }

    public long getID() {
        return ID;
    }

    public void setID(long ID) {
        this.ID = ID;
    }

    public String getDESCRIPTION() {
        return DESCRIPTION;
    }

    public void setDESCRIPTION(String DESCRIPTION) {
        this.DESCRIPTION = DESCRIPTION;
    }

    public Date getDATE_DEBUT() {
        return DATE_DEBUT;
    }

    public void setDATE_DEBUT(Date DATE_DEBUT) {
        this.DATE_DEBUT = DATE_DEBUT;
    }

    public Date getDATE_FIN() {
        return DATE_FIN;
    }

    public void setDATE_FIN(Date DATE_FIN) {
        this.DATE_FIN = DATE_FIN;
    }
```

```
    public String getTITLE() {
        return TITLE;
    }

    public void setTITLE(String TITLE) {
        this.TITLE = TITLE;
    }

    public List<Materiel> getMATERIEL() {
        return MATERIELS;
    }

    public void setMATERIEL(List<Materiel> MATERIEL) {
        this.MATERIELS = MATERIEL;
    }

    public List<OrderDeTravail> getORDERS_DE_TRAVAIL() {
        return ORDERS_DE_TRAVAIL;
    }

    public void setORDERS_DE_TRAVAIL(List<OrderDeTravail> ORDERS_DE_TRAVAIL) {
        this.ORDERS_DE_TRAVAIL = ORDERS_DE_TRAVAIL;
    }

    public List<Notification> getNOTIFICATIONS() {
        return NOTIFICATIONS;
    }

    public void setNOTIFICATIONS(List<Notification> NOTIFICATIONS) {
        this.NOTIFICATIONS = NOTIFICATIONS;
    }

    public List<Intervenant> getINTERVENT() {
        return INTERVENT;
    }

    public void setINTERVENT(List<Intervenant> INTERVENT) {
        this.INTERVENT = INTERVENT;
    }

    public Projet getPROJET() {
        return PROJET;
    }

    public void setPROJET(Projet PROJET) {
        this.PROJET = PROJET;
    }
}
```

- La classe **User**

```
package project.tasks_management.entities;

import java.io.Serializable;

public class User implements Serializable {
    private long ID=0;
    private String NOM="";
    private String PRENOM="";
    private String CIN="";
    private String TEL="";
    private String EMAIL="";
```

```
    private  String PASSWORD="";

    public User(){ }

    public User(long ID, String NOM, String PRENOM, String CIN, String TEL,String
EMAIL,String PASSWORD) {
        this.ID = ID;
        this.NOM = NOM;
        this.PRENOM = PRENOM;
        this.CIN = CIN;
        this.TEL = TEL;
        this.EMAIL=EMAIL;
        this.PASSWORD=PASSWORD;
    }

    public long getID() {
        return ID;
    }

    public void setID(long ID) {
        this.ID = ID;
    }

    public String getNOM() {
        return NOM;
    }

    public void setNOM(String NOM) {
        this.NOM = NOM;
    }

    public String getPRENOM() {
        return PRENOM;
    }

    public void setPRENOM(String PRENOM) {
        this.PRENOM = PRENOM;
    }

    public String getCIN() {
        return CIN;
    }

    public void setCIN(String CIN) {
        this.CIN = CIN;
    }

    public String getTEL() {
        return TEL;
    }

    public void setTEL(String TEL) {
        this.TEL = TEL;
    }

    public String getEMAIL() {
        return EMAIL;
    }

    public void setEMAIL(String EMAIL) {
        this.EMAIL = EMAIL;
    }

    public String getPASSWORD() {
```

```
        return PASSWORD;
    }

    public void setPASSWORD(String PASSWORD) {
        this.PASSWORD = PASSWORD;
    }

    @Override
    public String toString() {
        return "{" +
                "ID=" + ID +
                ", NOM='" + NOM + '\'' +
                ", PRENOM='" + PRENOM + '\'' +
                ", CIN='" + CIN + '\'' +
                ", TEL='" + TEL + '\'' +
                ", EMAIL='" + EMAIL + '\'' +
                ", PASSWORD='" + PASSWORD + '\'' +
                '}';
    }
}
```

2) La couche **DAO**

- La connexion à la base de données.

```
package project.tasks_management.dao;

import java.sql.*;

61 usages
public class SingletonConnexionDB {
    2 usages
    private static Connection connection;
    static {
        try {
            connection = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/task_management", user: "root", password: "");
        } catch (SQLException e){
            e.printStackTrace();
        }
    }
    54 usages
    public Connection getConnexion() { return connection; }
}
```

- Interface de base **Dao**

```java
package project.tasks_management.dao.intervenant;


import project.tasks_management.dao.Dao;
import project.tasks_management.entities.Intervenant;
import project.tasks_management.entities.Task;

import java.util.List;

2 usages  1 implementation
public interface IntervenantDAO extends Dao<Intervenant> {
    1 usage  1 implementation
    List<Intervenant> getIntervenantsByTask(Task t);
}
```

- Interface **IntervenantDAO**

```java
package project.tasks_management.dao.intervenant;


import project.tasks_management.dao.Dao;
import project.tasks_management.entities.Intervenant;
import project.tasks_management.entities.Task;

import java.util.List;

2 usages  1 implementation
public interface IntervenantDAO extends Dao<Intervenant> {
    1 usage  1 implementation
    List<Intervenant> getIntervenantsByTask(Task t);
}
```

- Interface **MaterielDAO**

```java
package project.tasks_management.dao.materiel;

import project.tasks_management.dao.Dao;
import project.tasks_management.entities.*;

import java.util.List;

1 usage  1 implementation
public interface MaterialDAO extends Dao<Materiel> {
    1 usage  1 implementation
    public List<Materiel> getMaterielsByResponsable(Responsable r);
    1 usage  1 implementation
    public List<Materiel> getMaterielsByTask(Task t);
}
```

- Interface **NotificationDAO**

```
package project.tasks_management.dao.notification;

import ...

1 usage  1 implementation
public interface NotificationDAO extends Dao<Notification> {
    1 usage  1 implementation
    public List<Notification> getNotificationByTask(Task t);
    1 usage  1 implementation
    public List<Notification> getNotificationByIntervenant(Intervenant i);
}
```

- Interface **OrderDeTravailDAO**

```
package project.tasks_management.dao.order_de_travail;


import ...

1 usage  1 implementation
public interface OrderDeTravailDAO extends Dao<OrderDeTravail> {
    1 usage  1 implementation
    List<OrderDeTravail> getOrderDeTravailByTask(Task t);
}
```

- Interface **ProjetDAO**

```java
package project.tasks_management.dao.projet;

import ...

1 usage  1 implementation
public interface ProjetDAO extends Dao<Projet> {
    1 usage  1 implementation
    public List<Projet> getProjetsByResponsable(Responsable r);
    1 usage  1 implementation
    public Projet getProjetByTask(Task t);
}
```

- Interface **ResponsableDAO**

```java
package project.tasks_management.dao.responsable;

import ...

1 usage  1 implementation
public interface ResponsableDAO extends Dao<Responsable> {
    1 usage  1 implementation
    Responsable getResponsablesByMateriel(Materiel m);
    1 usage  1 implementation
    Responsable getResponsablesByProjet(Projet p);
}
```

- Interface **TaskDAO**

```java
import java.sql.SQLException;
import java.util.List;

1 usage  1 implementation
public interface TaskDAO extends Dao<Task> {
    1 usage  1 implementation
    public Task getTaskByOrderDeTravail(OrderDeTravail o);
    1 usage  1 implementation
    public List<Task> getTasksByNotification(Notification n);
    1 usage  1 implementation
    public List<Task> getTasksByIntervenant(Intervenant i);
    1 usage  1 implementation
    public List<Task> getTasksByProjet(Projet p);
    1 usage  1 implementation
    public List<Task> getTasksByMateriel(Materiel m);
    1 usage  1 implementation
    public Materiel affecter_materiel(Task t,Materiel m);
    1 usage  1 implementation
    public Intervenant affecter_intervenant(Task t,Intervenant i);
    1 usage  1 implementation
    public boolean delete_affectation_intervenant(Task t,Intervenant i);
    1 usage  1 implementation
    public boolean delete_affectation_materiel(Task t,Materiel m);
}
```

- Implémentation de l'interface **IntervenantDAO**

```java
package project.tasks_management.dao.intervenant;

import project.tasks_management.dao.SingletonConnexionDB;
import project.tasks_management.dao.notification.NotificationDmpl;
import project.tasks_management.dao.task.TaskDmpl;
import project.tasks_management.entities.Intervenant;
import project.tasks_management.entities.Notification;
import project.tasks_management.entities.Task;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class IntervenantDmpl implements IntervenantDAO {
    @Override
    public List findAll() {
        List<Intervenant> Intervenants = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from user
```

```
where isIntervenant=1");
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                Intervenant i=new Intervenant();
                i.setID(rs.getInt("ID"));
                i.setNOM(rs.getString("NOM"));
                i.setPRENOM(rs.getString("PRENOM"));
                i.setTEL(rs.getString("TEL"));
                i.setEMAIL(rs.getString("EMAIL"));
                i.setPASSWORD(rs.getString("PASSWORD"));

                // affecter notifications
                /*NotificationDmpl n=new NotificationDmpl();
                if(n.getNotificationByIntervenant(i) != null) {
                    List<Notification> ns = n.getNotificationByIntervenant(i);
                    i.setNOTIFICATIONS(ns);
                }*/

                // affecter tasks
                /*TaskDmpl t=new TaskDmpl();
                if(t.getTasksByIntervenant(i) != null){
                    List<Task> ts=t.getTasksByIntervenant(i);
                    i.setTASK(ts);
                }*/

                Intervenants.add(i);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return Intervenants;
    }

    @Override
    public Intervenant findOne(long id) {
        Intervenant i=new Intervenant();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from user
where isIntervenant=1 and ID=?");
            pstm.setLong(1,id);
            ResultSet rs=pstm.executeQuery();
            if(rs.next()){
                i.setID(rs.getLong("ID"));
                i.setNOM(rs.getString("NOM"));
                i.setPRENOM(rs.getString("PRENOM"));
                i.setTEL(rs.getString("TEL"));
                i.setEMAIL(rs.getString("EMAIL"));
                i.setPASSWORD(rs.getString("PASSWORD"));

                // affecter notifications
                /*NotificationDmpl n=new NotificationDmpl();
                if(n.getNotificationByIntervenant(i) != null) {
                    List<Notification> ns = n.getNotificationByIntervenant(i);
                    i.setNOTIFICATIONS(ns);
                }*/

                // affecter tasks
                /*TaskDmpl t=new TaskDmpl();
                if(t.getTasksByIntervenant(i) != null){
                    List<Task> ts=t.getTasksByIntervenant(i);
                    i.setTASK(ts);
                }*/
            }
```

```
        }catch (SQLException e){
            e.printStackTrace();
        }
        return i;
    }

    @Override
    public Intervenant save(Intervenant i) {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("insert into user
(NOM,PRENOM,CIN,TEL,EMAIL,PASSWORD,isResponsable,isIntervenant) values
(?,?,?,?,?,?,0,1)");
            pstm.setString(1,i.getNOM());
            pstm.setString(2,i.getPRENOM());
            pstm.setString(3,i.getCIN());
            pstm.setString(4,i.getTEL());
            pstm.setString(5,i.getEMAIL());
            pstm.setString(6,i.getPASSWORD());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return i;
    }

    @Override
    public boolean delete(Intervenant i) {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("delete from user where
ID=? and isIntervenant=1");
            pstm.setLong(1,i.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return true;
    }

    @Override
    public Intervenant update(Intervenant i) {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("update user set
NOM=?,PRENOM=?,CIN=?,TEL=?,EMAIL=?,PASSWORD=? where ID=? and isIntervenant=1");
            pstm.setString(1,i.getNOM());
            pstm.setString(2,i.getPRENOM());
            pstm.setString(3,i.getCIN());
            pstm.setString(4,i.getTEL());
            pstm.setString(5,i.getEMAIL());
            pstm.setString(6,i.getPASSWORD());
            pstm.setLong(7,i.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return i;
    }

    @Override
    public List<Intervenant> getIntervenantsByTask(Task t){
        List<Intervenant> Intervenants = new ArrayList<>();
        try {
```

```
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select
i.ID,i.NOM,i.PRENOM,i.CIN,i.TEL,i.EMAIL,i.PASSWORD from user i,affecter_intervenant
aff,task t where isIntervenant=1 and i.ID=aff.ID_INTERVENANT and t.ID=aff.ID_TASK and
t.ID=?");
            pstm.setLong(1,t.getID());
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                Intervenant i=new Intervenant();
                i.setID(rs.getInt("ID"));
                i.setNOM(rs.getString("NOM"));
                i.setPRENOM(rs.getString("PRENOM"));
                i.setTEL(rs.getString("TEL"));
                i.setEMAIL(rs.getString("EMAIL"));
                i.setPASSWORD(rs.getString("PASSWORD"));

                // affecter notifications
                /*NotificationDmpl n=new NotificationDmpl();
                if(n.getNotificationByIntervenant(i) != null) {
                    List<Notification> ns = n.getNotificationByIntervenant(i);
                    i.setNOTIFICATIONS(ns);
                }*/

                // affecter stasks
                /*TaskDmpl tt=new TaskDmpl();
                if(tt.getTasksByIntervenant(i) != null){
                    List<Task> ts=tt.getTasksByIntervenant(i);
                    i.setTASK(ts);
                }*/

                Intervenants.add(i);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return Intervenants;
    }

}
```

- Implémentation de l'interface **MaterielDAO**

```
package project.tasks_management.dao.materiel;

import project.tasks_management.dao.SingletonConnexionDB;
import project.tasks_management.dao.responsable.ResponsableDmpl;
import project.tasks_management.dao.task.TaskDmpl;
import project.tasks_management.entities.Materiel;
import project.tasks_management.entities.Responsable;
import project.tasks_management.entities.Task;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class MaterialDmpl implements MaterialDAO {
    @Override
    public List<Materiel> findAll() throws SQLException {
```

```
        List<Materiel> Materiels = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from mate-
riel");
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                Materiel m=new Materiel();
                m.setID(rs.getLong("ID"));
                m.setTYPE(rs.getString("TYPE"));
                m.setNOM(rs.getString("NOM"));

                // affecter responsable
                /*ResponsableDmpl r=new ResponsableDmpl();
                if(r.getResponsablesByMateriel(m) != null) {
                    Responsable rrs = r.getResponsablesByMateriel(m);
                    m.setRESPONSABLE(rrs);
                }*/

                // affecter tasks
                /*TaskDmpl t=new TaskDmpl();
                if(t.getTasksByMateriel(m) != null) {
                    List<Task> ts = t.getTasksByMateriel(m);
                    m.setTASK(ts);
                }*/

                Materiels.add(m);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return Materiels;
    }

    @Override
    public Materiel findOne(long id) throws SQLException {
        Materiel m=new Materiel();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from materiel
where ID=?");
            pstm.setLong(1,id);
            ResultSet rs=pstm.executeQuery();
            if (rs.next()){
                m.setID(rs.getLong("ID"));
                m.setTYPE(rs.getString("TYPE"));
                m.setNOM(rs.getString("NOM"));

                // affecter responsable
                /*ResponsableDmpl r=new ResponsableDmpl();
                if(r.getResponsablesByMateriel(m) != null) {
                    Responsable rrs = r.getResponsablesByMateriel(m);
                    m.setRESPONSABLE(rrs);
                }*/

                // affecter tasks
                /*TaskDmpl t=new TaskDmpl();
                if(t.getTasksByMateriel(m) != null) {
                    List<Task> ts = t.getTasksByMateriel(m);
                    m.setTASK(ts);
                }*/
            }
        }catch (SQLException e){
            e.printStackTrace();
```

```
        }
        return m;
    }

    @Override
    public Materiel save(Materiel m) throws SQLException {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("insert into materiel
(ID,TYPE,NOM,RESPONSABLE_ID) values (?,?,?,?)");
            pstm.setLong(1,m.getID());
            pstm.setString(2,m.getTYPE());
            pstm.setString(3,m.getNOM());
            pstm.setLong(4,m.getRESPONSABLE().getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return m;
    }

    @Override
    public boolean delete(Materiel m) throws SQLException {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("delete materiel where
ID=?");
            pstm.setLong(1,m.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return true;
    }

    @Override
    public Materiel update(Materiel m) throws SQLException {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("update materiel set
ID=?,TYPE=?,NOM=?,RESPONSABLE_ID=?,TASK_ID=? where ID=?");
            pstm.setString(1,m.getTYPE());
            pstm.setString(2,m.getNOM());
            pstm.setLong(3,m.getRESPONSABLE().getID());
            pstm.setLong(4,0); // task id
            pstm.setLong(5,m.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return m;
    }

    @Override
    public List<Materiel> getMaterielsByResponsable(Responsable r){
        List<Materiel> Materiels = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select
m.ID,m.TYPE,m.NOM,m.RESPONSABLE_ID,m.TASK_ID from materiel m,user r where m.RESPON-
SABLE_ID=r.ID,r.ID=?");
            pstm.setLong(1,r.getID());
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
```

```
                Materiel m=new Materiel();
                m.setID(rs.getLong("ID"));
                m.setTYPE(rs.getString("TYPE"));
                m.setNOM(rs.getString("NOM"));

                // affecter responsable
                /*ResponsableDmpl rr=new ResponsableDmpl();
                if(rr.getResponsablesByMateriel(m) != null) {
                    Responsable rrs = rr.getResponsablesByMateriel(m);
                    m.setRESPONSABLE(rrs);
                }*/

                // affecter tasks
                /*TaskDmpl t=new TaskDmpl();
                if(t.getTasksByMateriel(m) != null) {
                    List<Task> ts = t.getTasksByMateriel(m);
                    m.setTASK(ts);
                }*/

                Materiels.add(m);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return Materiels;
    }

    @Override
    public List<Materiel> getMaterielsByTask(Task t){
        List<Materiel> Materiels = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select
m.ID,m.TYPE,m.NOM,m.RESPONSABLE_ID FROM materiel m,task t,affecter_materiel aff where
m.ID=aff.ID_MATERIEL and t.ID=aff.ID_TASK and t.ID=?");
            pstm.setLong(1,t.getID());
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                Materiel m=new Materiel();
                m.setID(rs.getLong("ID"));
                m.setTYPE(rs.getString("TYPE"));
                m.setNOM(rs.getString("NOM"));

                // affecter responsable
                /*ResponsableDmpl r=new ResponsableDmpl();
                if(r.getResponsablesByMateriel(m)!=null) {
                    Responsable rrs = r.getResponsablesByMateriel(m);
                    m.setRESPONSABLE(rrs);
                }*/

                // affecter tasks
                /*TaskDmpl tt=new TaskDmpl();
                if(tt.getTasksByMateriel(m)!=null) {
                    List<Task> ts = tt.getTasksByMateriel(m);
                    m.setTASK(ts);
                }*/

                Materiels.add(m);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return Materiels;
```

```
    }
}
```

- Implémentation de l'interface **NotificationDAO**

```
package project.tasks_management.dao.notification;

import project.tasks_management.dao.SingletonConnexionDB;
import project.tasks_management.entities.*;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class NotificationDmpl implements NotificationDAO {
    @Override
    public List findAll() {
        List<Notification> notifications = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from notifica-
tion");
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                Notification n=new Notification();
                n.setID(rs.getLong("ID"));
                n.setTITLE(rs.getString("TITLE"));
                n.setDATE(rs.getDate("DATE"));
                n.setDESCRIPTION(rs.getString("DESCRIPTION"));
                notifications.add(n);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return notifications;
    }

    @Override
    public Notification findOne(long id) {
        Notification n=new Notification();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from notifica-
tion where ID=?");
            pstm.setLong(1,id);
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                n.setID(rs.getLong("ID"));
                n.setTITLE(rs.getString("TITLE"));
                n.setDATE(rs.getDate("DATE"));
                n.setDESCRIPTION(rs.getString("DESCRIPTION"));
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return n;
    }

    @Override
    public Notification save(Notification n) {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
```

```
            PreparedStatement pstm=connection.prepareStatement("insert into notifica-
tion (ID,TITLE,DATE,DESCRIPTION,TASK_ID,INTERVENANT_ID) values (?,?,?,?,?,?)");
            pstm.setLong(1,n.getID());
            pstm.setString(2,n.getTITLE());
            pstm.setString(3, n.getDESCRIPTION());
            pstm.setLong(4, n.getTASK().getID());
            pstm.setLong(5, n.getINTERVENENT().getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return n;
    }

    @Override
    public boolean delete(Notification n) {
        Connection connection=new SingletonConnexionDB().getConnexion();
        try {
            PreparedStatement pstm=connection.prepareStatement("delete projet where
ID=?");
            pstm.setLong(1,n.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return true;
    }

    @Override
    public Notification update(Notification n) {
        Connection connection=new SingletonConnexionDB().getConnexion();
        try {
            PreparedStatement pstm=connection.prepareStatement("update notification set
TITLE=?,DATE=?,DESCRIPTION=?,TASK_ID=?,INTERVENANT=? where ID=?");
            pstm.setLong(1,n.getID());
            pstm.setString(2,n.getTITLE());
            pstm.setString(3, n.getDESCRIPTION());
            pstm.setLong(4, n.getTASK().getID());
            pstm.setLong(5, n.getINTERVENENT().getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return n;
    }

    @Override
    public List<Notification> getNotificationByTask(Task t){
        List<Notification> notifications = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select n.ID,n.TI-
TLE,n.DATE,n.DESCRIPTION,n.TASK_ID,n.INTERVENANT_ID from notification n,task t where
n.TASK_ID=t.ID and t.ID=?");
            pstm.setLong(1,t.getID());
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                Notification n=new Notification();
                n.setID(rs.getLong("ID"));
                n.setTITLE(rs.getString("TITLE"));
                n.setDATE(rs.getDate("DATE"));
                n.setDESCRIPTION(rs.getString("DESCRIPTION"));
                notifications.add(n);
            }
```

```
        }catch (SQLException e){
            e.printStackTrace();
        }
        return notifications;
    }

    @Override
    public List<Notification> getNotificationByIntervenant(Intervenant i) {
        List<Notification> notifications = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select n.ID,n.TI-
TLE,n.DATE,n.DESCRIPTION,n.TASK_ID,n.INTERVENANT_ID from notification n,user i where
i.isIntervenant=true and i.ID=n.INTERVENANT_ID and n.INTERVENANT_ID=?");
            pstm.setLong(1,i.getID());
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                Notification n=new Notification();
                n.setID(rs.getLong("ID"));
                n.setTITLE(rs.getString("TITLE"));
                n.setDATE(rs.getDate("DATE"));
                n.setDESCRIPTION(rs.getString("DESCRIPTION"));
                notifications.add(n);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return notifications;
    }
}
```

- Implémentation de l'interface **OrderDeTravailDAO**

```
package project.tasks_management.dao.order_de_travail;

import project.tasks_management.dao.SingletonConnexionDB;
import project.tasks_management.dao.task.TaskDmpl;
import project.tasks_management.entities.OrderDeTravail;
import project.tasks_management.entities.Task;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class OrderDeTravailDmpl implements OrderDeTravailDAO {
    @Override
    public List findAll() {
        List<OrderDeTravail> orders = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from or-
der_de_travail");
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                OrderDeTravail o=new OrderDeTravail();
                o.setID(rs.getInt("ID"));
                o.setTITLE(rs.getString("TITLE"));
                o.setDESCRIPTION(rs.getString("DESCRIPTION"));

                // affecter task
```

```
                /*TaskDmpl t=new TaskDmpl();
                if(t.getTaskByOrderDeTravail(o) != null) {
                    Task tt = t.getTaskByOrderDeTravail(o);
                    o.setTask(tt);
                }*/

                orders.add(o);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return orders;
    }

    @Override
    public OrderDeTravail findOne(long id) {
        OrderDeTravail o=new OrderDeTravail();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from or-
der_de_travail where ID=?");
            pstm.setLong(1,id);
            ResultSet rs=pstm.executeQuery();
            if(rs.next()){
                o.setID(rs.getInt("ID"));
                o.setTITLE(rs.getString("TITLE"));
                o.setDESCRIPTION(rs.getString("DESCRIPTION"));

                // affecter task
                /*TaskDmpl t=new TaskDmpl();
                if(t.getTaskByOrderDeTravail(o)!=null) {
                    Task tt = t.getTaskByOrderDeTravail(o);
                    o.setTask(tt);
                }*/
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return o;
    }

    @Override
    public OrderDeTravail save(OrderDeTravail o) {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("insert into or-
der_de_travail (ID,TITLE,DESCRIPTION,TASK_ID) values (?,?,?,?)");
            pstm.setLong(1,o.getID());
            pstm.setString(2,o.getTITLE());
            pstm.setString(3,o.getDESCRIPTION());
            pstm.setLong(4,o.getTask().getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return o;
    }

    @Override
    public boolean delete(OrderDeTravail o) {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("delete order_de_travail
where ID=?");
```

```
            pstm.setLong(1,o.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return true;
    }

    @Override
    public OrderDeTravail update(OrderDeTravail o) {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("update order_de_travail
set DESCRIPTION=?,TITLE=? where ID=?");
            pstm.setString(1,o.getTITLE());
            pstm.setString(2,o.getDESCRIPTION());
            pstm.setLong(3,o.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return o;
    }

    @Override
    public List<OrderDeTravail> getOrderDeTravailByTask(Task t){
        List<OrderDeTravail> orders = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select o.ID,o.TI-
TLE,o.DESCRIPTION,o.TASK_ID from order_de_travail o,task t where t.ID=o.TASK_ID and
t.ID=?");
            pstm.setLong(1,t.getID());
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                OrderDeTravail o=new OrderDeTravail();
                o.setID(rs.getInt("ID"));
                o.setTITLE(rs.getString("TITLE"));
                o.setDESCRIPTION(rs.getString("DESCRIPTION"));
                o.setTask(t);
                orders.add(o);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return orders;
    }

}
```

- Implémentation de l'interface **ProjetDAO**

```
package project.tasks_management.dao.projet;

import project.tasks_management.dao.SingletonConnexionDB;
import project.tasks_management.dao.responsable.ResponsableDmpl;
import project.tasks_management.dao.task.TaskDmpl;
import project.tasks_management.entities.*;

import java.sql.*;
import java.util.ArrayList;
```

```
import java.util.List;

public class ProjetDmpl implements ProjetDAO {
    @Override
    public List findAll() {
        List<Projet> Projets = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from projet");
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                Projet p=new Projet();
                p.setID(rs.getLong("ID"));
                p.setTITLE(rs.getString("TITLE"));
                p.setDate(rs.getDate("DATE_DEBUT"));

                // affecter responsable
                /*ResponsableDmpl r=new ResponsableDmpl();
                if(r.getResponsablesByProjet(p)!=null) {
                    Responsable rr = r.getResponsablesByProjet(p);
                    p.setRESPONSABLE(rr);
                }*/

                // affecter tasks
                /*TaskDmpl t=new TaskDmpl();
                if(t.getTasksByProjet(p)!=null) {
                    List<Task> ts = t.getTasksByProjet(p);
                    p.setTASKS(ts);
                }*/

                Projets.add(p);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return Projets;
    }

    @Override
    public Projet findOne(long id) {
        Projet p=new Projet();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from projet
where ID=?");
            pstm.setLong(1,id);
            ResultSet rs=pstm.executeQuery();
            if(rs.next()){
                p.setID(rs.getInt("ID"));
                p.setTITLE(rs.getString("TITLE"));
                p.setDate(rs.getDate("DATE_DEBUT"));

                // affecter responsable
                /*ResponsableDmpl r=new ResponsableDmpl();
                if(r.getResponsablesByProjet(p)!=null) {
                    Responsable rr = r.getResponsablesByProjet(p);
                    p.setRESPONSABLE(rr);
                }*/

                // affecter tasks
                /*TaskDmpl t=new TaskDmpl();
                if(t.getTasksByProjet(p)!=null) {
                    List<Task> ts = t.getTasksByProjet(p);
                    p.setTASKS(ts);
```

```
                }*/
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return p;
    }

    @Override
    public Projet save(Projet p) {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("insert into projet
(ID,TITLE,DATE_DEBUT,RESPONSABLE_ID) values (?,?,?,?)");
            pstm.setLong(1,p.getID());
            pstm.setString(2,p.getTITLE());
            pstm.setDate(3, (Date) p.getDate());
            pstm.setLong(4, p.getRESPONSABLE().getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return p;
    }

    @Override
    public boolean delete(Projet p) {
        Connection connection=new SingletonConnexionDB().getConnexion();
        try {
            PreparedStatement pstm=connection.prepareStatement("delete projet where
ID=?");
            pstm.setLong(1,p.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return true;
    }

    @Override
    public Projet update(Projet p) {
        Connection connection=new SingletonConnexionDB().getConnexion();
        try {
            PreparedStatement pstm=connection.prepareStatement("update projet set
TITLE=?,DATE_DEBUT=?,RESPONSABLE_ID=? where ID=?");
            pstm.setString(1,p.getTITLE());
            pstm.setDate(2, (Date) p.getDate());
            pstm.setLong(3, p.getRESPONSABLE().getID());
            pstm.setLong(4,p.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return p;
    }

    @Override
    public List<Projet> getProjetsByResponsable(Responsable r) {
        List<Projet> Projets = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select
p.TITLE,p.DATE_DEBUT,p.RESPONSABLE_ID from projet p,responsable r where p.ID_RESPON-
SABLE=r.ID and r.ID=?");
```

```
            pstm.setLong(1,r.getID());
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                Projet p=new Projet();
                p.setID(rs.getLong("ID"));
                p.setTITLE(rs.getString("TITLE"));
                p.setDate(rs.getDate("DATE_DEBUT"));

                // affecter responsable
                /*ResponsableDmpl rr=new ResponsableDmpl();
                if(rr.getResponsablesByProjet(p)!=null) {
                    Responsable rrr = rr.getResponsablesByProjet(p);
                    p.setRESPONSABLE(rrr);
                }*/

                // affecter tasks
                /*TaskDmpl t=new TaskDmpl();
                if(t.getTasksByProjet(p)!=null) {
                    List<Task> ts = t.getTasksByProjet(p);
                    p.setTASKS(ts);
                }*/

                Projets.add(p);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return Projets;
    }

    @Override
    public Projet getProjetByTask(Task t) {
        Projet p=new Projet();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select
p.ID,p.TITLE,p.DATE_DEBUT,p.RESPONSABLE_ID from projet p,task t where p.ID=t.PROJET_ID
and t.ID=?");
            pstm.setLong(1,t.getID());
            ResultSet rs=pstm.executeQuery();
            if (rs.next()){
                p.setID(rs.getLong("ID"));
                p.setTITLE(rs.getString("TITLE"));
                p.setDate(rs.getDate("DATE_DEBUT"));

                // affecter responsable
                /*ResponsableDmpl r=new ResponsableDmpl();
                if(r.getResponsablesByProjet(p)!=null) {
                    Responsable rr = r.getResponsablesByProjet(p);
                    p.setRESPONSABLE(rr);
                }*/

                // affecter tasks
                /*TaskDmpl tt=new TaskDmpl();
                if(tt.getTasksByProjet(p)!=null) {
                    List<Task> ts = tt.getTasksByProjet(p);
                    p.setTASKS(ts);
                }*/
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return p;
```

```
    }
}
```

- Implémentation de l'interface **ResponsableDAO**

```
package project.tasks_management.dao.responsable;

import project.tasks_management.dao.SingletonConnexionDB;
import project.tasks_management.entities.*;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class ResponsableDmpl implements ResponsableDAO {

    @Override
    public List findAll() {
        List<Responsable> Responsables = new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from user
where isResponsable=true");
            ResultSet rs=pstm.executeQuery();
            while (rs.next()){
                Responsable r=new Responsable();
                r.setID(rs.getInt("ID"));
                r.setNOM(rs.getString("NOM"));
                r.setPRENOM(rs.getString("PRENOM"));
                r.setCIN(rs.getString("CIN"));
                r.setTEL(rs.getString("TEL"));
                r.setEMAIL(rs.getString("EMAIL"));
                r.setPASSWORD(rs.getString("PASSWORD"));
                Responsables.add(r);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return Responsables;
    }

    @Override
    public Responsable findOne(long id) {
        Responsable r=new Responsable();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select * from user
where isResponsable=true and ID=?");
            pstm.setLong(1,id);
            ResultSet rs=pstm.executeQuery();
            if(rs.next()){
                r.setID(rs.getInt("ID"));
                r.setNOM(rs.getString("NOM"));
                r.setPRENOM(rs.getString("PRENOM"));
                r.setCIN(rs.getString("CIN"));
                r.setTEL(rs.getString("TEL"));
                r.setEMAIL(rs.getString("EMAIL"));
                r.setPASSWORD(rs.getString("PASSWORD"));
            }
        }catch (SQLException e){
```

```
            e.printStackTrace();
        }
        return r;
    }

    @Override
    public Responsable save(Responsable r) {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("insert into user
(NOM,PRENOM,CIN,TEL,EMAIL,PASSWORD,true,false) values (?,?,?,?,?,?)");
            pstm.setString(1,r.getNOM());
            pstm.setString(2,r.getPRENOM());
            pstm.setString(3,r.getCIN());
            pstm.setString(4,r.getTEL());
            pstm.setString(5,r.getEMAIL());
            pstm.setString(6,r.getPASSWORD());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return r;
    }

    @Override
    public boolean delete(Responsable r) {
        Connection connection=new SingletonConnexionDB().getConnexion();
        try {
            PreparedStatement pstm=connection.prepareStatement("delete user where ID=?
and isResponsable=true");
            pstm.setLong(1,r.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return true;
    }

    @Override
    public Responsable update(Responsable r) {
        Connection connection=new SingletonConnexionDB().getConnexion();
        try {
            PreparedStatement pstm=connection.prepareStatement("update user set
NOM=?,PRENOM=?,CIN=?,TEL=?,EMAIL=?,PASSWORD=? where ID=? and isResponsable=?");
            pstm.setString(1,r.getNOM());
            pstm.setString(2,r.getPRENOM());
            pstm.setString(3,r.getCIN());
            pstm.setString(4,r.getTEL());
            pstm.setString(5,r.getEMAIL());
            pstm.setString(6,r.getPASSWORD());
            pstm.setLong(7,r.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return r;
    }

    @Override
    public Responsable getResponsablesByMateriel(Materiel m) {
        Responsable r = new Responsable();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select
```

```
r.ID,r.NOM,r.PRENOM,r.CIN,r.TEL,r.EMAIL,r.PASSWORD from user r,materiel m where
r.isResponsable=true and r.ID=m.RESPONSABLE_ID and m.RESPONSABLE_ID=?");
            pstm.setLong(1,m.getID());
            ResultSet rs=pstm.executeQuery();
            if (rs.next()){
                r.setID(rs.getInt("ID"));
                r.setNOM(rs.getString("NOM"));
                r.setPRENOM(rs.getString("PRENOM"));
                r.setCIN(rs.getString("CIN"));
                r.setTEL(rs.getString("TEL"));
                r.setEMAIL(rs.getString("EMAIL"));
                r.setPASSWORD(rs.getString("PASSWORD"));
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return r;
    }

    @Override
    public Responsable getResponsablesByProjet(Projet p) {
        Responsable r=new Responsable();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("select
r.ID,r.NOM,r.PRENOM,r.CIN,r.TEL,r.EMAIL,r.PASSWORD from user r,projet p where r.isRes-
ponsable=1 and r.ID=p.RESPONSABLE_ID and p.ID=?");
            pstm.setLong(1,p.getID());
            ResultSet rs=pstm.executeQuery();
            if (rs.next()){
                r.setID(rs.getInt("ID"));
                r.setNOM(rs.getString("NOM"));
                r.setPRENOM(rs.getString("PRENOM"));
                r.setCIN(rs.getString("CIN"));
                r.setTEL(rs.getString("TEL"));
                r.setEMAIL(rs.getString("EMAIL"));
                r.setPASSWORD(rs.getString("PASSWORD"));
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return r;
    }
}
```

- Implémentation de l'interface **TaskDAO**

```
package project.tasks_management.dao.task;

import project.tasks_management.dao.SingletonConnexionDB;
import project.tasks_management.dao.intervenant.IntervenantDAO;
import project.tasks_management.dao.intervenant.IntervenantDmpl;
import project.tasks_management.dao.materiel.MaterialDmpl;
import project.tasks_management.dao.notification.NotificationDmpl;
import project.tasks_management.dao.order_de_travail.OrderDeTravailDmpl;
import project.tasks_management.dao.projet.ProjetDmpl;
import project.tasks_management.entities.*;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class TaskDmpl implements TaskDAO {
```

```
    @Override
    public List<Task> findAll() throws SQLException {
        List<Task> tasks=new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement statement = connection.prepareStatement("SELECT * FROM
task ");
            ResultSet resultSet = statement.executeQuery();
            if (resultSet.next()) {
                Task task = new Task();
                task.setID(resultSet.getLong("ID"));
                task.setDESCRIPTION(resultSet.getString("DESCRIPTION"));
                task.setDATE_DEBUT(resultSet.getDate("DATE_DEBUT"));
                task.setDATE_FIN(resultSet.getDate("DATE_FIN"));
                task.setTITLE(resultSet.getString("TITLE"));

                // affecter projet
                /*ProjetDmpl p=new ProjetDmpl();
                if(p.getProjetByTask(task)!=null) {
                    Projet pp = p.getProjetByTask(task);
                    task.setPROJET(pp);
                }*/

                // affecter materiels
                /*MaterialDmpl m=new MaterialDmpl();
                if(m.getMaterielsByTask(task)!=null) {
                    List<Materiel> ms = m.getMaterielsByTask(task);
                    task.setMATERIEL(ms);
                }*/

                // affecter intervenants
                /*IntervenantDAO i=new IntervenantDmpl();
                if(i.getIntervenantsByTask(task)!=null) {
                    List<Intervenant> is = i.getIntervenantsByTask(task);
                    task.setINTERVENT(is);
                }*/

                // affecter notifications
                /*NotificationDmpl n=new NotificationDmpl();
                if(n.getNotificationByTask(task)!=null) {
                    List<Notification> ns = n.getNotificationByTask(task);
                    task.setNOTIFICATIONS(ns);
                }*/

                // affecter orders_de_travail
                /*OrderDeTravailDmpl o=new OrderDeTravailDmpl();
                if(o.getOrderDeTravailByTask(task)!=null) {
                    List<OrderDeTravail> os = o.getOrderDeTravailByTask(task);
                    task.setORDERS_DE_TRAVAIL(os);
                }*/

                tasks.add(task);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }

        return tasks;
    }


    @Override
    public Task findOne(long id) throws SQLException {
```

```
        Task task = new Task();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement statement = connection.prepareStatement("SELECT * FROM
task");
            ResultSet resultSet = statement.executeQuery();
            if (resultSet.next()) {
                task.setID(resultSet.getLong("ID"));
                task.setDESCRIPTION(resultSet.getString("DESCRIPTION"));
                task.setDATE_DEBUT(resultSet.getDate("DATE_DEBUT"));
                task.setDATE_FIN(resultSet.getDate("DATE_FIN"));
                task.setTITLE(resultSet.getString("TITLE"));

                // affecter projet
                /*ProjetDmpl p=new ProjetDmpl();
                if(p.getProjetByTask(task)!=null) {
                    Projet pp = p.getProjetByTask(task);
                    task.setPROJET(pp);
                }*/

                // affecter materiels
                /*MaterialDmpl m=new MaterialDmpl();
                if(m.getMaterielsByTask(task)!=null) {
                    List<Materiel> ms = m.getMaterielsByTask(task);
                    task.setMATERIEL(ms);
                }*/

                // affecter intervenants
                /*IntervenantDAO i=new IntervenantDmpl();
                if(i.getIntervenantsByTask(task)!=null) {
                    List<Intervenant> is = i.getIntervenantsByTask(task);
                    task.setINTERVENT(is);
                }*/

                // affecter notifications
                /*NotificationDmpl n=new NotificationDmpl();
                if(n.getNotificationByTask(task)!=null) {
                    List<Notification> ns = n.getNotificationByTask(task);
                    task.setNOTIFICATIONS(ns);
                }*/

                // affecter orders_de_travail
                /*OrderDeTravailDmpl o=new OrderDeTravailDmpl();
                if(o.getOrderDeTravailByTask(task)!=null) {
                    List<OrderDeTravail> os = o.getOrderDeTravailByTask(task);
                    task.setORDERS_DE_TRAVAIL(os);
                }*/
            }
        }catch (SQLException e){
            e.printStackTrace();
        }

        return task;
    }

    @Override
    public Task save(Task t) throws SQLException {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("insert into task (DE-
SCRIPTION,DATE_DEBUT,DATE_FIN,TITLE,PROJET_ID) values (?,?,?,?,?)");
            pstm.setString(1,t.getDESCRIPTION());
            pstm.setDate(2, (Date) t.getDATE_DEBUT());
            pstm.setDate(3, (Date) t.getDATE_FIN());
```

```
            pstm.setLong(5,t.getPROJET().getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return t;
    }

    @Override
    public boolean delete(Task t) throws SQLException {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("delete task where
ID=?");
            pstm.setLong(1,t.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return true;
    }

    @Override
    public Task update(Task t) throws SQLException {
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("update task set DE-
SCRIPTION=?,DATE_DEBUT=?,DATE_FIN=?,TITLE=?,PROJET_ID=?) where ID=?");
            pstm.setString(1,t.getDESCRIPTION());
            pstm.setDate(2, (Date) t.getDATE_DEBUT());
            pstm.setDate(3, (Date) t.getDATE_FIN());
            pstm.setString(4, t.getTITLE());
            pstm.setLong(5,t.getPROJET().getID());
            pstm.setLong(8,t.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return t;
    }

    @Override
    public Task getTaskByOrderDeTravail(OrderDeTravail o){
        Task task=new Task();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("SELECT t.DESCRIP-
TION,t.DATE_DEBUT,t.DATE_FIN,t.TITLE,t.PROJET_ID,t.INTERVENANT_ID,t.MATERIEL_ID FROM
task t,order_de_travail o where t.ID=o.TASK_ID and o.TASK_ID=?");
            pstm.setLong(1,o.getTask().getID());
            ResultSet resultSet = pstm.executeQuery();
            if (resultSet.next()) {
                task.setID(resultSet.getLong("ID"));
                task.setDESCRIPTION(resultSet.getString("DESCRIPTION"));
                task.setDATE_DEBUT(resultSet.getDate("DATE_DEBUT"));
                task.setDATE_FIN(resultSet.getDate("DATE_FIN"));
                task.setTITLE(resultSet.getString("TITLE"));

                // affecter projet
                /*ProjetDmpl p=new ProjetDmpl();
                if(p.getProjetByTask(task)!=null) {
                    Projet pp = p.getProjetByTask(task);
                    task.setPROJET(pp);
                }*/
```

```
                // affecter materiels
                /*MaterialDmpl m=new MaterialDmpl();
                if(m.getMaterielsByTask(task)!=null) {
                    List<Materiel> ms = m.getMaterielsByTask(task);
                    task.setMATERIEL(ms);
                }*/

                // affecter intervenants
                /*IntervenantDAO i=new IntervenantDmpl();
                if(i.getIntervenantsByTask(task)!=null) {
                    List<Intervenant> is = i.getIntervenantsByTask(task);
                    task.setINTERVENT(is);
                }*/

                // affecter notifications
                /*NotificationDmpl n=new NotificationDmpl();
                if(n.getNotificationByTask(task)!=null) {
                    List<Notification> ns = n.getNotificationByTask(task);
                    task.setNOTIFICATIONS(ns);
                }*/

                // affecter orders_de_travail
                /*OrderDeTravailDmpl ord=new OrderDeTravailDmpl();
                if(ord.getOrderDeTravailByTask(task)!=null) {
                    List<OrderDeTravail> os = ord.getOrderDeTravailByTask(task);
                    task.setORDERS_DE_TRAVAIL(os);
                }*/
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return task;
    }
    @Override
    public List<Task> getTasksByNotification(Notification n){
        List<Task> tasks=new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement statement = connection.prepareStatement("SELECT t.DE-
SCRIPTION,t.DATE_DEBUT,t.DATE_FIN,t.TITLE,t.PROJET_ID,t.INTERVENANT_ID,t.MATERIEL_ID
FROM task t,notification n where t.ID=n.TASK_ID and n.TASK_ID=?");
            statement.setLong(1,n.getTASK().getID());
            ResultSet resultSet = statement.executeQuery();
            if (resultSet.next()) {
                Task task = new Task();
                task.setID(resultSet.getLong("ID"));
                task.setDESCRIPTION(resultSet.getString("DESCRIPTION"));
                task.setDATE_DEBUT(resultSet.getDate("DATE_DEBUT"));
                task.setDATE_FIN(resultSet.getDate("DATE_FIN"));
                task.setTITLE(resultSet.getString("TITLE"));

                // affecter projet
                /*ProjetDmpl p=new ProjetDmpl();
                if(p.getProjetByTask(task)!=null) {
                    Projet pp = p.getProjetByTask(task);
                    task.setPROJET(pp);
                }*/

                // affecter materiels
                /*MaterialDmpl m=new MaterialDmpl();
                if(m.getMaterielsByTask(task)!=null) {
                    List<Materiel> ms = m.getMaterielsByTask(task);
                    task.setMATERIEL(ms);
```

```
                }*/

                // affecter intervenants
                /*IntervenantDAO i=new IntervenantDmpl();
                if(i.getIntervenantsByTask(task)!=null) {
                    List<Intervenant> is = i.getIntervenantsByTask(task);
                    task.setINTERVENT(is);
                }*/

                // affecter notifications
                /*NotificationDmpl not=new NotificationDmpl();
                if(not.getNotificationByTask(task)!=null) {
                    List<Notification> ns = not.getNotificationByTask(task);
                    task.setNOTIFICATIONS(ns);
                }*/

                // affecter orders_de_travail
                /*OrderDeTravailDmpl o=new OrderDeTravailDmpl();
                if(o.getOrderDeTravailByTask(task)!=null) {
                    List<OrderDeTravail> os = o.getOrderDeTravailByTask(task);
                    task.setORDERS_DE_TRAVAIL(os);
                }*/

                tasks.add(task);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }

        return tasks;
    }
    @Override
    public List<Task> getTasksByIntervenant(Intervenant i){
        List<Task> tasks=new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement statement = connection.prepareStatement("SELECT
t.ID,t.DESCRIPTION,t.DATE_DEBUT,t.DATE_FIN,t.TITLE,t.PROJET_ID FROM task t,user u ,af-
fecter_intervenant aff where t.ID=aff.ID_TASK and u.ID=aff.ID_INTERVENANT and u.ID=?");
            statement.setLong(1,i.getID());
            ResultSet resultSet = statement.executeQuery();
            if (resultSet.next()) {
                Task task = new Task();
                task.setID(resultSet.getLong("ID"));
                task.setDESCRIPTION(resultSet.getString("DESCRIPTION"));
                task.setDATE_DEBUT(resultSet.getDate("DATE_DEBUT"));
                task.setDATE_FIN(resultSet.getDate("DATE_FIN"));
                task.setTITLE(resultSet.getString("TITLE"));

                // affecter projet
                /*ProjetDmpl p=new ProjetDmpl();
                if(p.getProjetByTask(task)!=null) {
                    Projet pp = p.getProjetByTask(task);
                    task.setPROJET(pp);
                }*/

                // affecter materiels
                /*MaterialDmpl m=new MaterialDmpl();
                if(m.getMaterielsByTask(task)!=null) {
                    List<Materiel> ms = m.getMaterielsByTask(task);
                    task.setMATERIEL(ms);
                }*/

                // affecter intervenants
```

```
                /*IntervenantDAO in=new IntervenantDmpl();
                if(in.getIntervenantsByTask(task)!=null) {
                    List<Intervenant> is = in.getIntervenantsByTask(task);
                    task.setINTERVENT(is);
                }*/

                // affecter notifications
                /*NotificationDmpl n=new NotificationDmpl();
                if(n.getNotificationByTask(task)!=null) {
                    List<Notification> ns = n.getNotificationByTask(task);
                    task.setNOTIFICATIONS(ns);
                }*/

                // affecter orders_de_travail
                /*OrderDeTravailDmpl o=new OrderDeTravailDmpl();
                if(o.getOrderDeTravailByTask(task)!=null) {
                    List<OrderDeTravail> os = o.getOrderDeTravailByTask(task);
                    task.setORDERS_DE_TRAVAIL(os);
                }*/

                tasks.add(task);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }

        return tasks;
    }
    @Override
    public List<Task> getTasksByProjet(Projet p){
        List<Task> tasks=new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement statement = connection.prepareStatement("SELECT
t.ID,t.DESCRIPTION,t.DATE_DEBUT,t.DATE_FIN,t.TITLE FROM task t where t.PROJET_ID=?");
            statement.setLong(1,p.getID());
            ResultSet resultSet = statement.executeQuery();
            if (resultSet.next()) {
                Task task = new Task();
                task.setID(resultSet.getLong("ID"));
                task.setDESCRIPTION(resultSet.getString("DESCRIPTION"));
                task.setDATE_DEBUT(resultSet.getDate("DATE_DEBUT"));
                task.setDATE_FIN(resultSet.getDate("DATE_FIN"));
                task.setTITLE(resultSet.getString("TITLE"));

                // affecter projet
                /* ProjetDmpl pr=new ProjetDmpl();
                if(pr.getProjetByTask(task)!=null) {
                    Projet pp = pr.getProjetByTask(task);
                    task.setPROJET(pp);
                } */

                // affecter materiels
                /*MaterialDmpl m=new MaterialDmpl();
                if(m.getMaterielsByTask(task)!=null) {
                    List<Materiel> ms = m.getMaterielsByTask(task);
                    task.setMATERIEL(ms);
                }*/

                // affecter intervenants
                /*IntervenantDAO i=new IntervenantDmpl();
                if(i.getIntervenantsByTask(task)!=null) {
                    List<Intervenant> is = i.getIntervenantsByTask(task);
                    task.setINTERVENT(is);
```

```
                }*/

                // affecter notifications
                /*NotificationDmpl n=new NotificationDmpl();
                if(n.getNotificationByTask(task)!=null) {
                    List<Notification> ns = n.getNotificationByTask(task);
                    task.setNOTIFICATIONS(ns);
                }*/

                // affecter orders_de_travail
                /*OrderDeTravailDmpl o=new OrderDeTravailDmpl();
                if(o.getOrderDeTravailByTask(task)!=null) {
                    List<OrderDeTravail> os = o.getOrderDeTravailByTask(task);
                    task.setORDERS_DE_TRAVAIL(os);
                }*/

                tasks.add(task);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }

        return tasks;
    }
    @Override
    public List<Task> getTasksByMateriel(Materiel m){
        List<Task> tasks=new ArrayList<>();
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement statement = connection.prepareStatement("SELECT
t.ID,t.DESCRIPTION,t.DATE_DEBUT,t.DATE_FIN,t.TITLE FROM materiel m,task t,affecter_ma-
teriel aff where aff.ID_MATERIEL=m.ID and aff.ID_TASK=t.ID and m.ID=?");
            statement.setLong(1,m.getID());
            ResultSet resultSet = statement.executeQuery();
            if (resultSet.next()) {
                Task task = new Task();
                task.setID(resultSet.getLong("ID"));
                task.setDESCRIPTION(resultSet.getString("DESCRIPTION"));
                task.setDATE_DEBUT(resultSet.getDate("DATE_DEBUT"));
                task.setDATE_FIN(resultSet.getDate("DATE_FIN"));
                task.setTITLE(resultSet.getString("TITLE"));

                // affecter projet
                /*ProjetDmpl p=new ProjetDmpl();
                if(p.getProjetByTask(task)!=null) {
                    Projet pp = p.getProjetByTask(task);
                    task.setPROJET(pp);
                }*/

                // affecter materiels
                /*MaterialDmpl mm=new MaterialDmpl();
                if(mm.getMaterielsByTask(task)!=null) {
                    List<Materiel> ms = mm.getMaterielsByTask(task);
                    task.setMATERIEL(ms);
                }*/

                // affecter intervenants
                /*IntervenantDAO i=new IntervenantDmpl();
                if(i.getIntervenantsByTask(task)!=null) {
                    List<Intervenant> is = i.getIntervenantsByTask(task);
                    task.setINTERVENT(is);
                }*/

                // affecter notifications
```

```
                /*NotificationDmpl n=new NotificationDmpl();
                if(n.getNotificationByTask(task)!=null) {
                    List<Notification> ns = n.getNotificationByTask(task);
                    task.setNOTIFICATIONS(ns);
                }*/

                // affecter orders_de_travail
                /*OrderDeTravailDmpl o=new OrderDeTravailDmpl();
                if(o.getOrderDeTravailByTask(task)!=null) {
                    List<OrderDeTravail> os = o.getOrderDeTravailByTask(task);
                    task.setORDERS_DE_TRAVAIL(os);
                }*/

                tasks.add(task);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }

        return tasks;
    }

    @Override
    public Materiel affecter_materiel(Task t,Materiel m){
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("insert into affec-
ter_materiel values (0,?,?)");
            pstm.setLong(1,t.getID());
            pstm.setLong(2,m.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return m;
    }

    @Override
    public Intervenant affecter_intervenant(Task t,Intervenant i){
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("insert into affec-
ter_intervenant values (0,?,?)");
            pstm.setLong(1,t.getID());
            pstm.setLong(2,i.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return i;
    }

    @Override
    public boolean delete_affectation_materiel(Task t,Materiel m){
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("delete from affec-
ter_materiel where TASK_ID=? and MATERIEL_ID=?");
            pstm.setLong(1,t.getID());
            pstm.setLong(2,m.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
```

```
        return true;
    }

    @Override
    public boolean delete_affectation_intervenant(Task t,Intervenant i){
        try {
            Connection connection=new SingletonConnexionDB().getConnexion();
            PreparedStatement pstm=connection.prepareStatement("delete from affec-
ter_intervenant where TASK_ID=? and INTERVENANT_ID=?");
            pstm.setLong(1,t.getID());
            pstm.setLong(2,i.getID());
            pstm.executeUpdate();
        }catch (SQLException e){
            e.printStackTrace();
        }
        return true;
    }
}
```

3) La couche **Services**

- L'interface de base Service

```
package project.tasks_management.services;

public interface Service <T>{ }
```

- L'interface **intervenantService**

```java
package project.tasks_management.services.intervenant;

import project.tasks_management.entities.Intervenant;
import project.tasks_management.entities.Task;
import project.tasks_management.services.Service;

import java.sql.SQLException;
import java.util.List;

1 usage  1 implementation
public interface intervenantService extends Service<Intervenant> {
    3 usages  1 implementation
    List<Intervenant> retournerIntervenants() throws SQLException;
    1 implementation
    Intervenant retournerIntervenantParId(Intervenant i) throws SQLException;
    1 implementation
    List<Intervenant> retournerIntervenantsParTask(Task t) throws SQLException;
    2 usages  1 implementation
    void ajouterIntervenant(Intervenant i) throws SQLException;

    1 usage  1 implementation
    void supprimerIntervenant(Intervenant i) throws SQLException;

    1 implementation
    void modifierIntervenant(Intervenant i) throws SQLException;
    1 implementation
    void enregistrerIntervenant(Intervenant i) throws SQLException;

}
```

- L'interface **materielService**

```java
import java.sql.SQLException;
import java.util.List;

2 usages  1 implementation
public interface materielService extends Service<Materiel> {
    2 usages  1 implementation
    public List<Materiel> retournerMateriels() throws SQLException;
    1 implementation
    public Materiel retournerMaterielParId(Materiel m) throws SQLException;
    1 implementation
    public List<Materiel> retournerMaterielsParTask(Task t) throws SQLException;
    1 implementation
    public List<Materiel> retournerMaterielsParResponsable(Responsable r) throws SQLException;
    1 implementation
    public void ajouterMateriel(Materiel m) throws SQLException;

    1 implementation
    public void supprimerMateriel(Materiel m) throws SQLException;

    1 implementation
    public void modifierMateriel(Materiel m) throws SQLException;
    1 implementation
    public void enregistrerMateriel(Materiel m) throws SQLException;

}
```

- L'interface **notificationService**

```java
package project.tasks_management.services.notification;

import project.tasks_management.entities.Intervenant;
import project.tasks_management.entities.Notification;
import project.tasks_management.entities.Task;
import project.tasks_management.services.Service;

import java.sql.SQLException;
import java.util.List;

1 usage  1 implementation
public interface notificationService extends Service<Notification> {
    1 usage  1 implementation
    public List<Notification> retournerNotifications() throws SQLException;
    1 implementation
    public Notification retournerNotificationParId(Notification n) throws SQLException;
    1 implementation
    public List<Notification> retournerNotificationsParTask(Task t) throws SQLException;
    1 implementation
    public List<Notification> retournerNotificationsParIntervenant(Intervenant i) throws SQLException;
    1 implementation
    public void ajouterNotification(Notification n) throws SQLException;
    1 implementation
    public void supprimerNotification(Notification n) throws SQLException;

    1 implementation
    public void modifierNotification(Notification n) throws SQLException;
    1 implementation
    public void enregistrerNotification(Notification n) throws SQLException;
}
```

- L’interface **order_de_travailService**

```java
import project.tasks_management.entities.OrderDeTravail;
import project.tasks_management.entities.Task;
import project.tasks_management.services.Service;

import java.sql.SQLException;
import java.util.List;

1 usage  1 implementation
public interface order_de_travailService extends Service<OrderDeTravail> {
    1 usage  1 implementation
    public List<OrderDeTravail> retournerOrderDeTravails() throws SQLException;
    1 implementation
    public OrderDeTravail retournerOrderDeTravailParId(OrderDeTravail o) throws SQLException;
    1 implementation
    public List<OrderDeTravail> retournerOrdersDeTravailParTask(Task t) throws SQLException;
    1 implementation
    public void ajouterOrderDeTravail(OrderDeTravail o) throws SQLException;
    1 implementation
    public void supprimerOrderDeTravail(OrderDeTravail o) throws SQLException;

    1 implementation
    public void modifierOrderDeTravail(OrderDeTravail o) throws SQLException;
    1 implementation
    public void enregistrerOrderDeTravail(OrderDeTravail o) throws SQLException;
}
```

- L’interface **projetService**

```java
package project.tasks_management.services.projet;

import project.tasks_management.entities.Projet;
import project.tasks_management.entities.Responsable;
import project.tasks_management.entities.Task;
import project.tasks_management.services.Service;

import java.sql.SQLException;
import java.util.List;

1 usage  1 implementation
public interface projetService extends Service<Projet> {
    1 usage  1 implementation
    public List<Projet> retournerProjets() throws SQLException;
    1 implementation
    public Projet retournerProjetParId(Projet p) throws SQLException;
    1 implementation
    public List<Projet> retournerProjetsParResponsable(Responsable r) throws SQLException;
    1 implementation
    public Projet retournerProjetParTask(Task t) throws SQLException;
    1 implementation
    public void ajouterProjet(Projet p) throws SQLException;

    1 implementation
    public void supprimerProjet(Projet p) throws SQLException;

    1 implementation
    public void modifierProjet(Projet p) throws SQLException;
    1 implementation
    public void enregistrerProjet(Projet p) throws SQLException;
}
```

- L'interface **responsableService**

```java
package project.tasks_management.services.responsable;

import project.tasks_management.entities.Materiel;
import project.tasks_management.entities.Projet;
import project.tasks_management.entities.Responsable;
import project.tasks_management.services.Service;

import java.sql.SQLException;
import java.util.List;

1 usage  1 implementation
public interface responsableService extends Service<Projet> {
    1 usage  1 implementation
    public List<Responsable> retournerResponsables() throws SQLException;
    1 implementation
    public Responsable retournerResponsableParId(Responsable r) throws SQLException;
    1 implementation
    public Responsable retournerResponsableParMateriel(Materiel m) throws SQLException;
    1 implementation
    public Responsable retournerResponsableParProjet(Projet p) throws SQLException;
    1 implementation
    public void ajouterResponsable(Responsable r) throws SQLException;

    1 implementation
    public void supprimerResponsable(Responsable r) throws SQLException;

    1 implementation
    public void modifierResponsable(Responsable r) throws SQLException;
    1 implementation
    public void enregistrerResponsable(Responsable r) throws SQLException;
}
```

- L'interface **taskService**

```java
1 usage  1 implementation
public interface taskService extends Service<Task> {
    4 usages  1 implementation
    public List<Task> retournerTasks() throws SQLException; public Task retournerTaskParId(Task t) throws SQLException;
    1 implementation
    public Task retournerTaskParOrderDeTravail(OrderDeTravail o) throws SQLException;
    1 implementation
    public List<Task> retournerTasksParNotification(Notification n) throws SQLException;
    1 implementation
    public List<Task> retournerTasksParIntervenant(Intervenant i) throws SQLException;
    1 implementation
    public List<Task> retournerTasksParProjet(Projet p) throws SQLException;
    1 implementation
    public List<Task> retournerTasksParMateriel(Materiel m) throws SQLException;
    1 usage  1 implementation
    public Materiel affecter_materiel(Task t,Materiel m) throws SQLException;
    1 usage  1 implementation
    public Intervenant affecter_intervenant(Task t,Intervenant i) throws SQLException;
    1 usage  1 implementation
    public boolean supprimer_affectation_intervenant(Task t,Intervenant i) throws SQLException;
    1 implementation
    public boolean supprimer_affectation_materiel(Task t,Materiel m) throws SQLException;
    1 implementation
    public void ajouterTask(Task t) throws SQLException;
    1 implementation
    public void supprimerTask(Task t) throws SQLException;
    1 implementation
    public void modifierTask(Task t) throws SQLException;
    1 implementation
    public void enregistrerTask(Task t) throws SQLException;
}
```

- Implémentation de l'interface **intervenantService**

```
package project.tasks_management.services.intervenant;

import project.tasks_management.dao.intervenant.IntervenantDmpl;
import project.tasks_management.entities.Task;

import java.sql.SQLException;
import java.util.List;

public class Intervenant implements intervenantService{
    IntervenantDmpl idmpl=new IntervenantDmpl();

    @Override
    public List<project.tasks_management.entities.Intervenant> retournerIntervenants()
throws SQLException {
        return idmpl.findAll();
    }

    @Override
    public project.tasks_management.entities.Intervenant retournerIntervenantParId(pro-
ject.tasks_management.entities.Intervenant i) throws SQLException {
        return idmpl.findOne(i.getID());
    }

    @Override
    public List<project.tasks_management.entities.Intervenant> retournerInterve-
nantsParTask(Task t) throws SQLException {
        return idmpl.getIntervenantsByTask(t);
    }

    @Override
    public void ajouterIntervenant(project.tasks_management.entities.Intervenant i)
throws SQLException {
        idmpl.save(i);
    }

    @Override
    public void supprimerIntervenant(project.tasks_management.entities.Intervenant i)
throws SQLException {
        idmpl.delete(i);
    }

    @Override
    public void modifierIntervenant(project.tasks_management.entities.Intervenant i)
throws SQLException {
        idmpl.update(i);
    }

    @Override
    public void enregistrerIntervenant(project.tasks_management.entities.Intervenant i)
throws SQLException {
        idmpl.save(i);
    }
}
```

- Implémentation de l'interface **materielService**

```
package project.tasks_management.services.materiel;

import project.tasks_management.dao.materiel.MaterialDmpl;
import project.tasks_management.entities.Responsable;
import project.tasks_management.entities.Task;
```

```
import java.sql.SQLException;
import java.util.List;

public class Materiel implements materielService{
    MaterialDmpl mdmpl=new MaterialDmpl();

    @Override
    public List<project.tasks_management.entities.Materiel> retournerMateriels() throws
SQLException {
        return mdmpl.findAll();
    }

    @Override
    public project.tasks_management.entities.Materiel retournerMaterielParId(pro-
ject.tasks_management.entities.Materiel m) throws SQLException {
        return mdmpl.findOne(m.getID());
    }

    @Override
    public List<project.tasks_management.entities.Materiel> retournerMaterielsPar-
Task(Task t) throws SQLException {
        return mdmpl.getMaterielsByTask(t);
    }

    @Override
    public List<project.tasks_management.entities.Materiel> retournerMaterielsParRes-
ponsable(Responsable r) throws SQLException {
        return mdmpl.getMaterielsByResponsable(r);
    }

    @Override
    public void ajouterMateriel(project.tasks_management.entities.Materiel m) throws
SQLException {
        mdmpl.save(m);
    }

    @Override
    public void supprimerMateriel(project.tasks_management.entities.Materiel m) throws
SQLException {
        mdmpl.delete(m);
    }

    @Override
    public void modifierMateriel(project.tasks_management.entities.Materiel m) throws
SQLException {
        mdmpl.update(m);
    }

    @Override
    public void enregistrerMateriel(project.tasks_management.entities.Materiel m)
throws SQLException {
        mdmpl.save(m);
    }
}
```

- Implémentation de l'interface **notificationService**

```
package project.tasks_management.services.notification;

import project.tasks_management.dao.notification.NotificationDmpl;
import project.tasks_management.entities.Intervenant;
import project.tasks_management.entities.Task;
```

```
import java.sql.SQLException;
import java.util.List;

public class Notification implements notificationService {
    NotificationDmpl ndmpl=new NotificationDmpl();

    @Override
    public List<project.tasks_management.entities.Notification> retournerNotifica-
tions() throws SQLException {
        return ndmpl.findAll();
    }

    @Override
    public project.tasks_management.entities.Notification retournerNotificationPa-
rId(project.tasks_management.entities.Notification n) throws SQLException {
        return ndmpl.findOne(n.getID());
    }

    @Override
    public List<project.tasks_management.entities.Notification> retournerNotifications-
ParTask(Task t) throws SQLException {
        return ndmpl.getNotificationByTask(t);
    }

    @Override
    public List<project.tasks_management.entities.Notification> retournerNotifications-
ParIntervenant(Intervenant i) throws SQLException {
        return ndmpl.getNotificationByIntervenant(i);
    }

    @Override
    public void ajouterNotification(project.tasks_management.entities.Notification n)
throws SQLException {
        ndmpl.save(n);
    }

    @Override
    public void supprimerNotification(project.tasks_management.entities.Notification n)
throws SQLException {
        ndmpl.delete(n);
    }

    @Override
    public void modifierNotification(project.tasks_management.entities.Notification n)
throws SQLException {
        ndmpl.update(n);
    }

    @Override
    public void enregistrerNotification(project.tasks_management.entities.Notification
n) throws SQLException {
        ndmpl.save(n);
    }
}
```

- Implémentation de l'interface **order_de_travailService**

```
package project.tasks_management.services.order_de_travail;

import project.tasks_management.dao.order_de_travail.OrderDeTravailDmpl;
import project.tasks_management.entities.OrderDeTravail;
import project.tasks_management.entities.Task;
```

```
import java.sql.SQLException;
import java.util.List;

public class Order_de_travail implements order_de_travailService {
    OrderDeTravailDmpl odmpl=new OrderDeTravailDmpl();

    @Override
    public List<OrderDeTravail> retournerOrderDeTravails() throws SQLException {
        return odmpl.findAll();
    }

    @Override
    public OrderDeTravail retournerOrderDeTravailParId(OrderDeTravail o) throws SQLEx-
ception {
        return odmpl.findOne(o.getID());
    }

    @Override
    public List<OrderDeTravail> retournerOrdersDeTravailParTask(Task t) throws SQLEx-
ception {
        return odmpl.getOrderDeTravailByTask(t);
    }

    @Override
    public void ajouterOrderDeTravail(OrderDeTravail o) throws SQLException {
        odmpl.save(o);
    }

    @Override
    public void supprimerOrderDeTravail(OrderDeTravail o) throws SQLException {
        odmpl.delete(o);
    }

    @Override
    public void modifierOrderDeTravail(OrderDeTravail o) throws SQLException {
        odmpl.update(o);
    }

    @Override
    public void enregistrerOrderDeTravail(OrderDeTravail o) throws SQLException {
        odmpl.save(o);
    }
}
```

- Implémentation de l'interface **projetService**

```
package project.tasks_management.services.projet;

import project.tasks_management.dao.projet.ProjetDmpl;
import project.tasks_management.entities.Responsable;
import project.tasks_management.entities.Task;

import java.sql.SQLException;
import java.util.List;

public class Projet implements projetService {
    ProjetDmpl pdmpl=new ProjetDmpl();

    @Override
    public List<project.tasks_management.entities.Projet> retournerProjets() throws
SQLException {
        return pdmpl.findAll();
```

```
    }

    @Override
    public project.tasks_management.entities.Projet retournerProjetParId(pro-
ject.tasks_management.entities.Projet p) throws SQLException {
        return pdmpl.findOne(p.getID());
    }

    @Override
    public List<project.tasks_management.entities.Projet> retournerProjetsParRespon-
sable(Responsable r) throws SQLException {
        return pdmpl.getProjetsByResponsable(r);
    }

    @Override
    public project.tasks_management.entities.Projet retournerProjetParTask(Task t)
throws SQLException {
        return pdmpl.getProjetByTask(t);
    }

    @Override
    public void ajouterProjet(project.tasks_management.entities.Projet p) throws SQLEx-
ception {
        pdmpl.save(p);
    }

    @Override
    public void supprimerProjet(project.tasks_management.entities.Projet p) throws
SQLException {
        pdmpl.delete(p);
    }

    @Override
    public void modifierProjet(project.tasks_management.entities.Projet p) throws
SQLException {
        pdmpl.update(p);
    }

    @Override
    public void enregistrerProjet(project.tasks_management.entities.Projet p) throws
SQLException {
        pdmpl.save(p);
    }
}
```

- Implémentation de l'interface **responsableService**

```
package project.tasks_management.services.responsable;

import project.tasks_management.dao.responsable.ResponsableDmpl;
import project.tasks_management.entities.Materiel;
import project.tasks_management.entities.Projet;

import java.sql.SQLException;
import java.util.List;

public class Responsable implements responsableService {
    ResponsableDmpl rdmpl=new ResponsableDmpl();


    @Override
    public List<project.tasks_management.entities.Responsable> retournerResponsables()
throws SQLException {
```

```
        return rdmpl.findAll();
    }

    @Override
    public project.tasks_management.entities.Responsable retournerResponsableParId(pro-
ject.tasks_management.entities.Responsable r) throws SQLException {
        return rdmpl.findOne(r.getID());
    }

    @Override
    public project.tasks_management.entities.Responsable retournerResponsableParMate-
riel(Materiel m) throws SQLException {
        return rdmpl.getResponsablesByMateriel(m);
    }

    @Override
    public project.tasks_management.entities.Responsable retournerResponsableParPro-
jet(Projet p) throws SQLException {
        return rdmpl.getResponsablesByProjet(p);
    }

    @Override
    public void ajouterResponsable(project.tasks_management.entities.Responsable r)
throws SQLException {
        rdmpl.save(r);
    }

    @Override
    public void supprimerResponsable(project.tasks_management.entities.Responsable r)
throws SQLException {
        rdmpl.delete(r);
    }

    @Override
    public void modifierResponsable(project.tasks_management.entities.Responsable r)
throws SQLException {
        rdmpl.update(r);
    }

    @Override
    public void enregistrerResponsable(project.tasks_management.entities.Responsable r)
throws SQLException {
        rdmpl.save(r);
    }
}
```

- Implémentation de l'interface **taskService**

```
package project.tasks_management.services.task;

import project.tasks_management.dao.task.TaskDmpl;
import project.tasks_management.entities.*;

import java.sql.SQLException;
import java.util.List;

public class Task implements taskService {
    TaskDmpl tmpl=new TaskDmpl();


    @Override
    public List<project.tasks_management.entities.Task> retournerTasks() throws SQLEx-
ception {
```

```
        return tmpl.findAll();
    }

    @Override
    public project.tasks_management.entities.Task retournerTaskParId(project.tasks_ma-
nagement.entities.Task t) throws SQLException {
        return tmpl.findOne(t.getID());
    }

    @Override
    public project.tasks_management.entities.Task retournerTaskParOrderDeTravail(Order-
DeTravail o){
        return tmpl.getTaskByOrderDeTravail(o);
    }

    @Override
    public List<project.tasks_management.entities.Task> retournerTasksParNotifica-
tion(Notification n){
        return tmpl.getTasksByNotification(n);
    }

    @Override
    public List<project.tasks_management.entities.Task> retournerTasksParInterve-
nant(Intervenant i){
        return tmpl.getTasksByIntervenant(i);
    }

    @Override
    public List<project.tasks_management.entities.Task> retournerTasksParProjet(Projet
p){
        return tmpl.getTasksByProjet(p);
    }

    @Override
    public List<project.tasks_management.entities.Task> retournerTasksParMateriel(Mate-
riel m){
        return tmpl.getTasksByMateriel(m);
    }

    @Override
    public Materiel affecter_materiel(project.tasks_management.entities.Task t, Mate-
riel m){
        return tmpl.affecter_materiel(t,m);
    }

    @Override
    public Intervenant affecter_intervenant(project.tasks_management.entities.Task t,
Intervenant i){
        return tmpl.affecter_intervenant(t,i);
    }

    @Override
    public boolean supprimer_affectation_intervenant(project.tasks_management.enti-
ties.Task t, Intervenant i){
        return tmpl.delete_affectation_intervenant(t,i);
    }

    @Override
    public boolean supprimer_affectation_materiel(project.tasks_management.enti-
ties.Task t, Materiel m){
        return tmpl.delete_affectation_materiel(t,m);
    }

    @Override
```

```
    public void ajouterTask(project.tasks_management.entities.Task t) throws SQLExcep-
tion {
        tmpl.save(t);
    }

    @Override
    public void supprimerTask(project.tasks_management.entities.Task t) throws SQLEx-
ception {
        tmpl.delete(t);
    }

    @Override
    public void modifierTask(project.tasks_management.entities.Task t) throws SQLExcep-
tion {
        tmpl.update(t);
    }

    @Override
    public void enregistrerTask(project.tasks_management.entities.Task t) throws SQLEx-
ception {
        tmpl.save(t);
    }
}
```

Conclusion

Dans cette partie nous avons présenté le code des couches ENTITIES, DAO et SERVICES mais pour la couche présentation vous trouverez dans ce lien GitHub le code de notre projet.

https://github.com/TAFFAHACHRAF/Tasks-management-project

Conclusion

Dans ce projet nous avons suivies toutes les étapes de développement d'une application desktop dynamique, en utilisant le langage orienté objet Java avec **JavaFx** dans la partie interface graphique, nous avons suivi une méthode de travail agile à l'aide de SCRUM qui nous a beaucoup aidé pour l'avancement dans notre projet même si notre groupe ne dépasse pas deux personnes. Lorsque nous avons développé les couches de zéro, nous avons bien touché l'importance de l'utilisation des Framework qui nous facilite beaucoup le travail, pour la présentation des interfaces ce rapport est inclus avec un vidéo qui présente le fonctionnement du projet.