INSTITUT FRANCOPHONE INTERNATIONAL

VNU ĐẠI HỌC QUỐC GIA HÀ NỘI Vietnam National University, Hanoi

MODULE : GENIE LOGICIEL AVANCE

TP1 :

PETIT GESTIONNAIRE DE TACHE

PRESENTER PAR : KOUADIO KOUAME OLIVIER

Etudiant en master 1 – promotion 21

Hanoï, Août 2017

SOMMAIRE

I- INTRODUCTION

Dans le but de nous mettre à jour sur les concepts de la modélisation avec la méthode UML et la programmation orientée objet, il nous a été demandé au cours du module de Génie Logiciel Avancé de concevoir et de réaliser une application de gestion des tâches. Pour réaliser ce TP nous avons créé notre application avec Java EE et utilisé GitHub pour la gestion de nos codes sources, nous avons utilisé l'environnement de développement intégré ECLIPS. Le présent rapport expose les différentes étapes de conception et de réalisation de l'application.

1- Spécifications de l'application

Les spécifications de l'application à réaliser sont les suivantes :

- **Créer, modifier, supprimer, ajouter une tâche**

Une tâche est composée des informations suivantes :

ID, Nom, Une description, Statut (nouveau, en-progrès, terminé).

- **Créer, modifier, supprimer, ajouter un membre**

Un membre est composé des informations suivantes :

ID, Nom.

- **Assigner une tâche à un membre**
- **Chercher et afficher tous les tâches assignées à un membre (par son ID)**
- **Chercher et afficher tous les tâches en fonction de leur statuts (avec le nom de l'assigné)**

2- Contrainte de programmation

- Il faut programmer de façon orientée objet
- Il faut assurer que la communication entre l'usager et le système est conviviale.
- Les consignes de codage doivent être respectées

II- LES EXIGENCES

1- Les exigences fonctionnelles et diagramme de cas d'utilisationDans cette partie nous allons à travers un tableau décrire les exigences fonctionnelles de notre application et nous allons utiliser le digramme de cas d'utilisation de ULM pour illustrer ces fonctionnalités de l'application.

1.1- Les exigences fonctionnelles

Par définition, les Exigences Fonctionnelles (EF) décrivent ce que le système doit pouvoir faire en termes d'actions et d'attentes. Pour ce qui concerne notre application elles sont décrites comme suit :

EF1 : Ajouter un Membre
Le système doit demander à l'utilisateur de renseigner le nom du membre qu'il veut créer
Le système doit enregistrer l'information renseignée par l'utilisateur dans la base de données
EF2 : Rechercher un Membre
Le système doit demander de saisir un mot clé ou le nom du membre à rechercher
Le système doit afficher la liste des membres ainsi leurs différentes informations pour le mot clé ou nom saisir par l'utilisateur.
EF3 : Modifier les informations d'un Membre
Le système doit permettre à l'utilisateur d'afficher la liste des membres afin qu'il choisisse le membre à modifier pour ne pas faire une erreur.
Le système doit pouvoir afficher les anciennes informations sur le membre
Le système doit permettre et demander de modifier les anciennes informations par les nouvelles que souhaite l'utilisateur.
Le système doit pouvoir enregistrer les nouvelles informations renseignées
EF4 : Supprimer les informations d'un Membre
Le système doit permettre à l'utilisateur d'afficher la liste des membres afin qu'il choisisse le membre à supprimer pour ne pas faire une erreur.
Le système doit permettre de supprimer les informations du membre enregistré
Le système doit pouvoir demander la confirmation de suppression
Le système doit être capable d'afficher à nouveau la liste pour constater la suppression du membre
EF5 : Afficher les informations d'un Membre
Le système doit permettre d'afficher les différentes informations renseignées du membre
Le système doit permettre d'afficher la liste de tous les membres enregistrées

Tableau1 : t*ableau d'exigence fonctionnelle pour le MEMBRE*

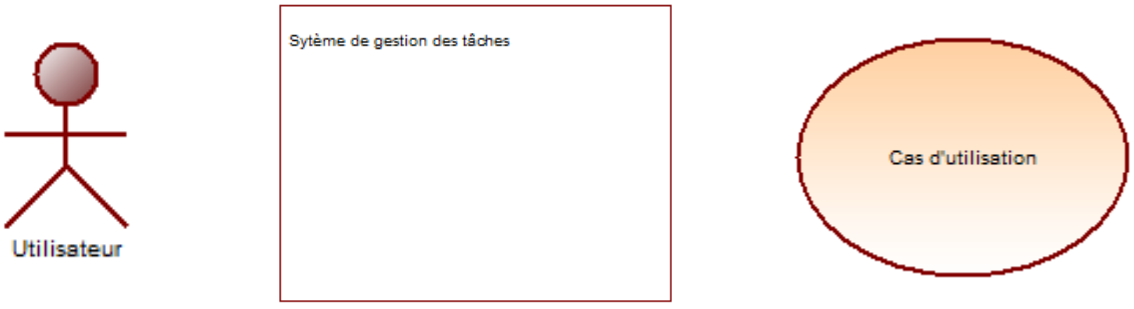
EF6 : Ajouter une Tâche Le système doit demander à l'utilisateur de renseigner les informations de la tâche qu'il veut créer Le système doit enregistrer les informations renseignées par l'utilisateur dans la base de données
EF7 : Rechercher une Tâche Le système doit demander de saisir un mot clé ou le nom de la tâche à rechercher Le système doit afficher la liste des tâches ainsi toutes différentes informations pour le mot clé ou nom saisir par l'utilisateur.
EF8 : Modifier les informations sur une Tâche Le système doit permettre à l'utilisateur d'afficher la liste des tâches afin qu'il choisisse la tâche à modifier pour ne pas faire une erreur. Le système doit pouvoir afficher les anciennes informations sur la tâche Le système doit permettre et demander de modifier les anciennes informations par les nouvelles que souhaite l'utilisateur. Le système doit pouvoir enregistrer les nouvelles informations renseignées
EF9 : Supprimer les informations d'une Tâche Le système doit permettre à l'utilisateur d'afficher la liste des tâches afin qu'il choisisse la tâche à supprimer pour ne pas faire une erreur. Le système doit permettre de supprimer les informations de la tâche enregistré Le système doit pouvoir demander la confirmation de suppression Le système doit être capable d'afficher à nouveau la liste des tâches pour constater la suppression du membre
EF10 : Afficher les informations d'une Tâche Le système doit permettre d'afficher les différentes informations renseignées de la tâche Le système doit permettre d'afficher la liste de toutes les tâche enregistrées.

Tableau2 : t*ableau d'exigence fonctionnelle pour la TACHE*

1.2- Diagramme de cas d'utilisation

Il nous semble bien important de donner le rôle spécifique de ce diagramme de la méthode UML. Il faut noter le diagramme de cas d'utilisateur sont utilisés dans la réalisation d'une application pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont très importants lorsque, en tant que chef de projets nous devons présenter notre nouvelle application à nos supérieurs hiérarchique. Ce digramme est très apprécié pour le développement d'une application car il décrit toutes les interactions entre un utilisateur et le système.

Notre diagramme de cas d'utilisation est composé des éléments suivants :



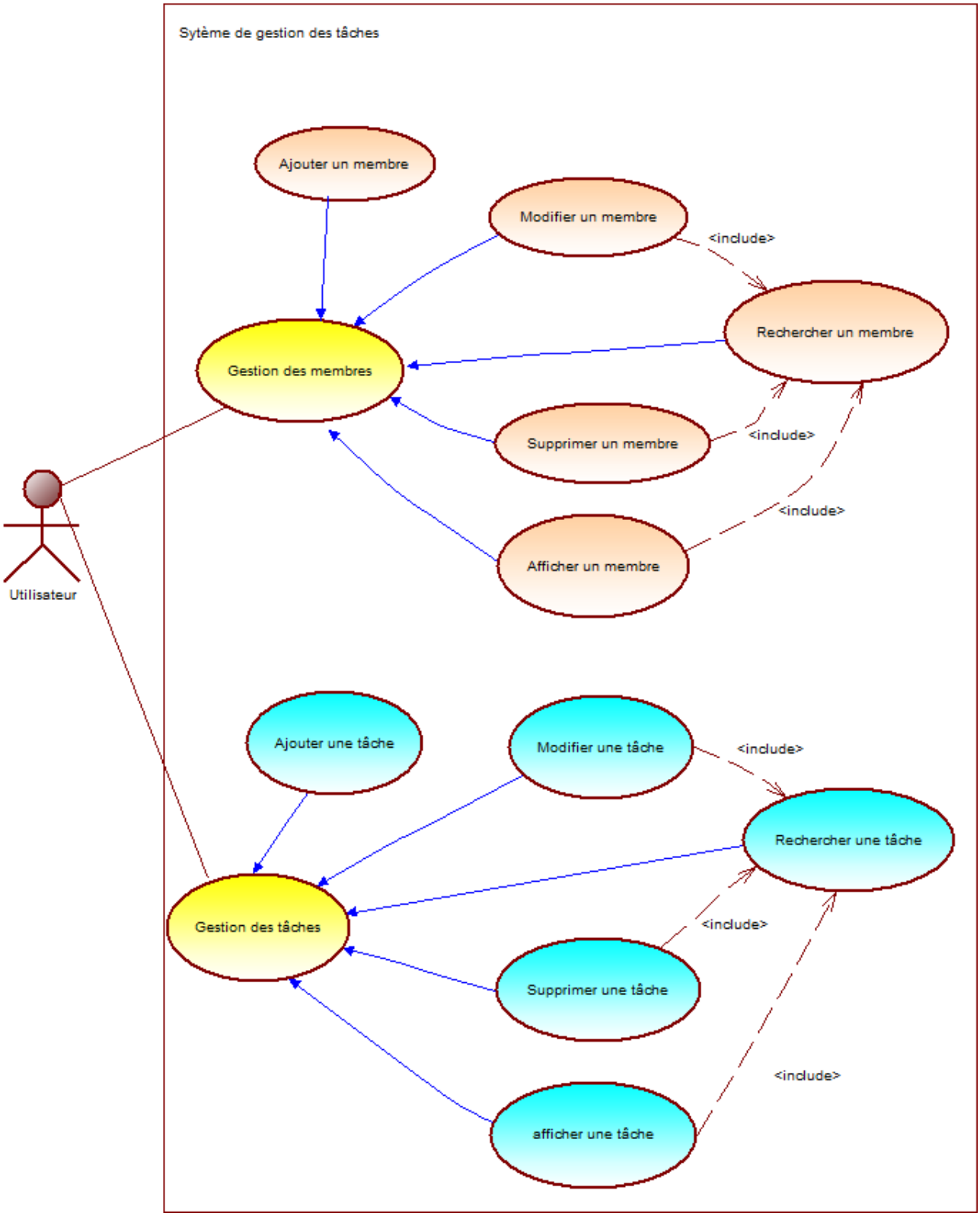


Figure1 *: Diagramme de cas d'utilisation*

3- Les exigences non-fonctionnelles

Les Exigences Non-Fonctionnelles, quant à elles caractérisent une qualité désirée du système telle que sa performance, sa robustesse, sa convivialité, sa maintenabilité, etc., les exigences non-fonctionnelles que nous avons défini pour notre application sont renseigner dans le tableau suivant :

Description	**Exigences fonctionnelles requises**	**Qualité/ Attribut**
Le système doit être disponible durant toutes les opérations et à tout moment ou l'utilisateur demande Le temps de réponse du système doit être essentiellement court, maximum 2 secondes pour exécuter la requête de l'utilisateur Notre application doit présenter des résultats précis et juste tels qu'attendus. Elle doit être capable d'effectuer la bonne opération demandée.	Toutes les exigences fonctionnelles	Performance Exactitude Pertinence
Les données doivent être disponibles et accessibles à tout moment où l'utilisateur en a besoin	Toutes les exigences fonctionnelles	Disponibilité
En cas de défaillance du système, les données ne doivent pas subir un changement Les fonctions de l'application doivent être facilement comprises et interprétés par l'utilisateur. Le code source de l'application est bien documenté pour la facilité d'entretien et de mise à jour du système à l'avenir	Toutes les exigences fonctionnelles	Facilité d'utilisation Facilité de la Maintenance

I- LA CONCEPTION

1- Diagramme de Classe

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation. Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour

réaliser les cas d'utilisation. Il est important de noter qu'un même objet peut très bien intervenir dans la réalisation de plusieurs cas d'utilisation.

Figure2 *: Diagramme de Classe*

2- Diagramme de séquence

Le diagramme de séquence quant à lui permet de representer les interaction entre objects et instances d'objets en précisant la chronologie des echanges de messages.Il répresente egalement une instance d'un cas d'utilisation(les sénario possibles d'un cas d'utilisation donné) . Ce diagramme est decrit parfaitement les interations entre l'utilisateur et le système. Dans notre contexte pour bien signifier cette interaction nous avons decider de présenter les opérations ***modifier un membre*** , ***supprimer une tâche*** *et* ***rechercher une tâche par l'ID du membre****.*

2.1- ***Modifier un membre***

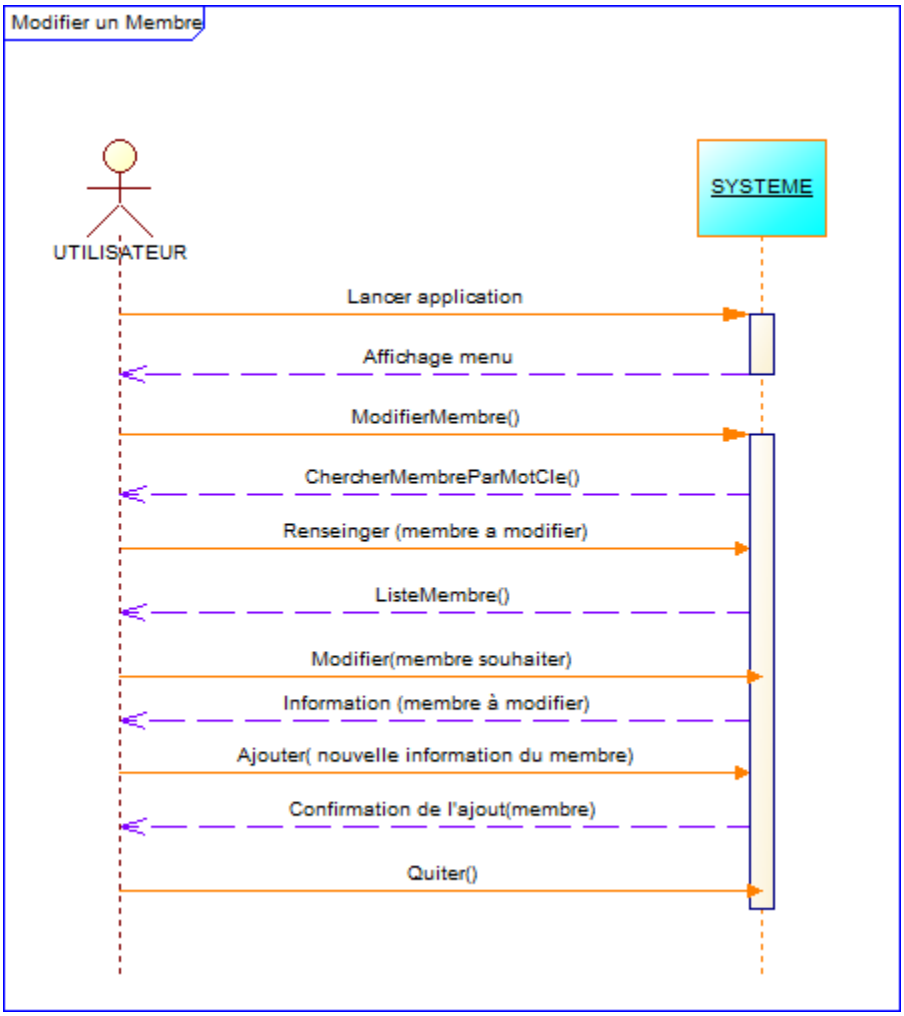


Figure3 :*Diagramme de cas d'utilisation : modifier un membre*

2.2- Supprimer une tâche

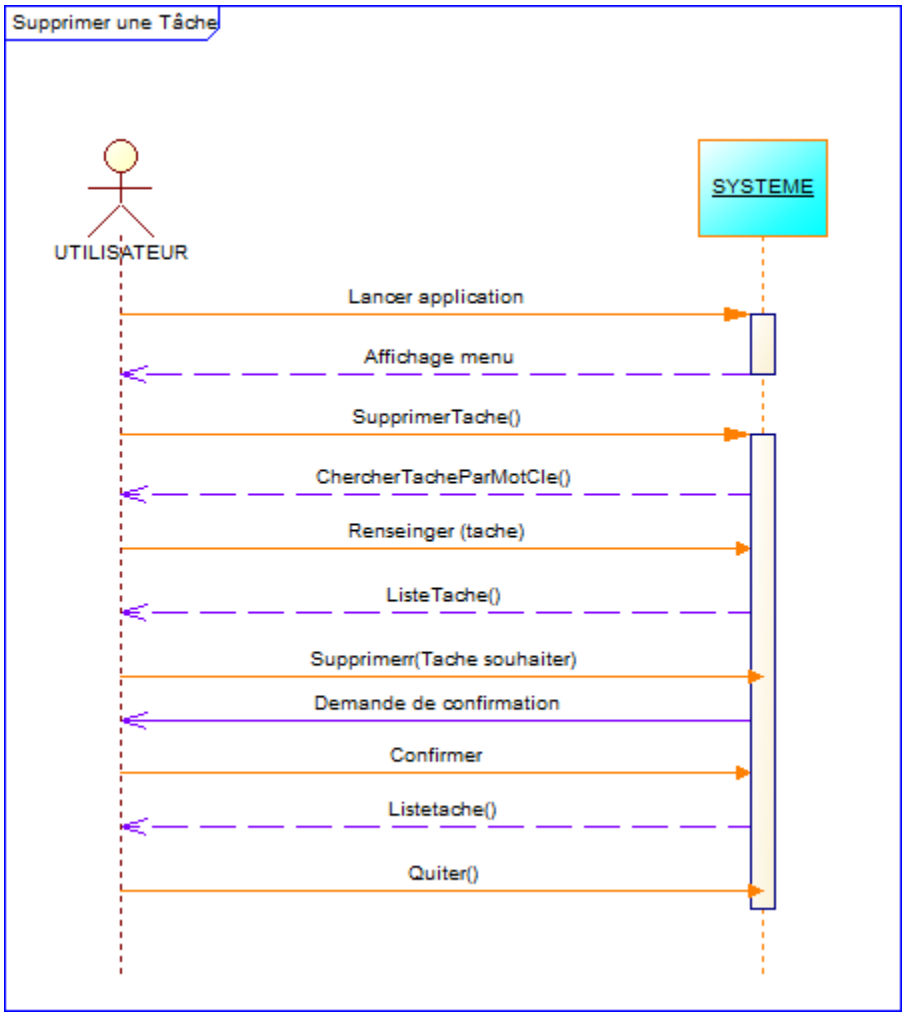


Figure4 :*Diagramme de cas d'utilisation :supprimer une tache*

2.3- Rechercher une tâche par l'ID du membre

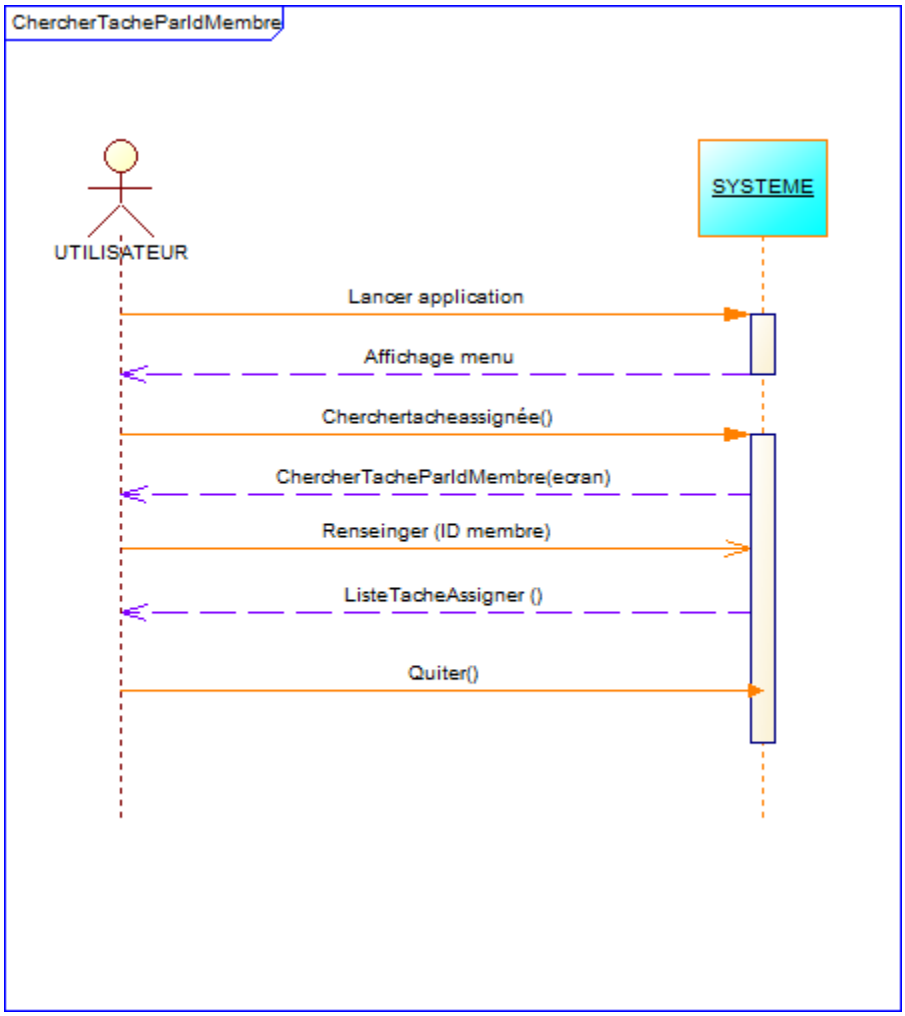


Figure5 : *diagramme cas d'utilisation : chercherTacheParIdMembre*

IV- IMPLEMENTATION

1- Plateforme et language

Notre application à été développer sous la plateforme *Eclipse IDE* qui est un environnement de developpement intégré. Nous avons utilisé le language JAVA précisement la Spécification *JAVA EE*, car notre application est accessible via une page web. Nous avons utilisé *MYSQL* comme système de gestion de base de donnée ainsi que *Tomcat 8.5* comme server d'application java. Et *XAMP* serveur qui contient apache et Mysql

2- Architecture Technique

Nous avons conçu notre application en **MVC (***modele vue controleur***)**

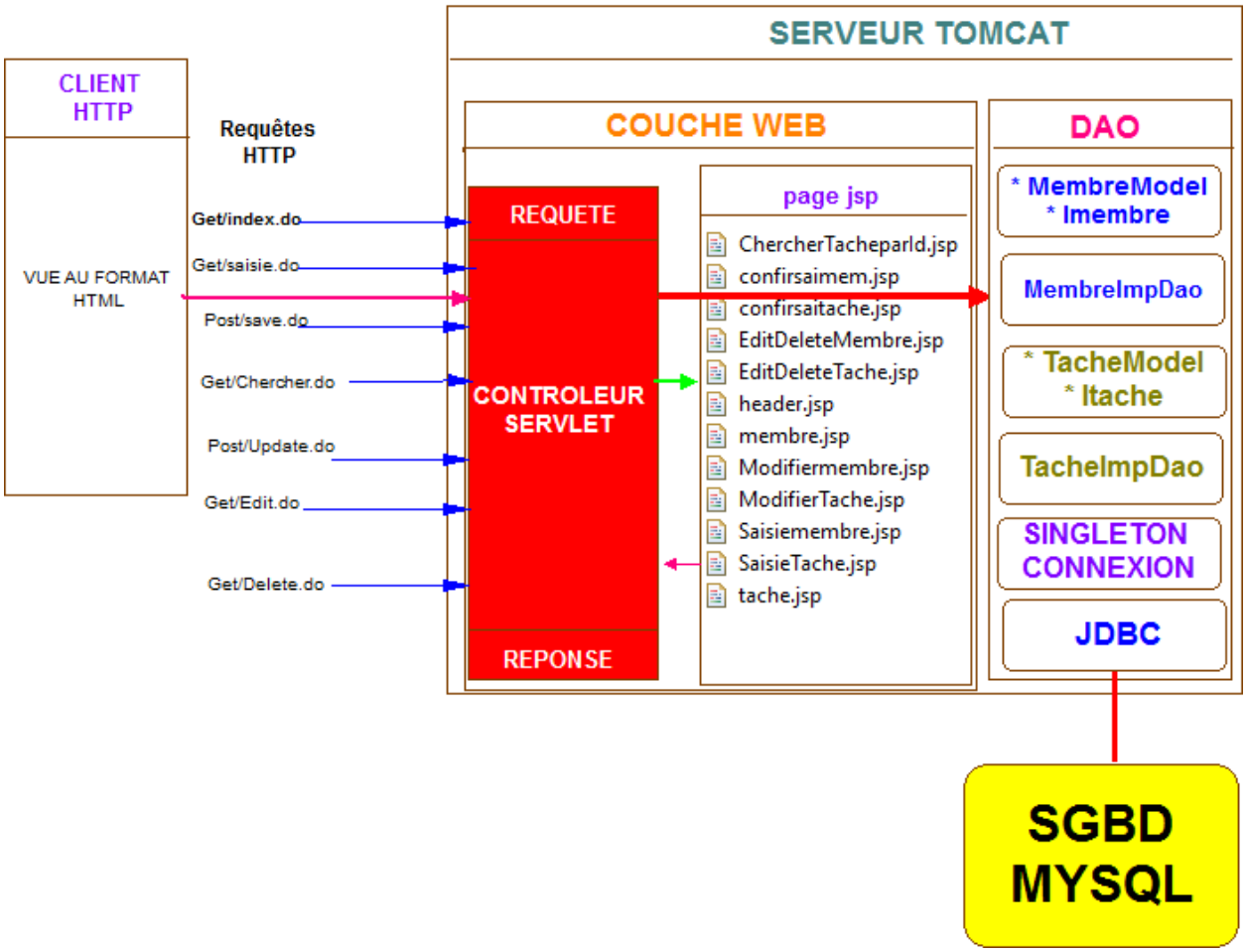
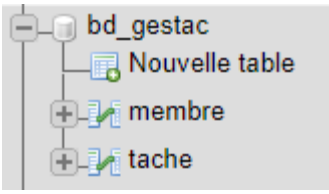


Figure6 : *Architecture technique*

3- Fonctionnement de l'application

Pour utiliser notre application il faut :

- Installer un server tomcat de preférence la version 8.5
- Installer le pilote JDBC(*mysql-connector-java*) vous le placer dans le *dossier lib* de *webContent*
- Installer server mysql et apache (wampserver, xampp, Mamp…)
- Créer la base de données nommée : **bd_gestac**



- Créér une table nommée **membre**
- Créér une table nommée **tache**

Le code sql nommée *bd_gestac.sql* est joint en à ce rapport vous pourrez créer les différentes tables automatiquement en exécutant les codes sql dans une nouvelle requête.

```sql
-- Base de données :  `bd_gestac`

CREATE TABLE `membre` (
  `ID_MEM` int(11) NOT NULL,
  `NOM_MEM` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=COMPACT;
CREATE TABLE `tache` (
  `ID` int(11) NOT NULL,
  `NOM_TAC` varchar(255) NOT NULL,
  `DESCRIPTION` varchar(255) NOT NULL,
  `STATUT` varchar(255) NOT NULL,
  `MEM_fk` int(11) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1 ROW_FORMAT=DYNAMIC;
ALTER TABLE `membre`
  ADD PRIMARY KEY (`ID_MEM`);
ALTER TABLE `tache`
  ADD PRIMARY KEY (`ID`),
  ADD KEY `MEM_fk` (`MEM_fk`);
ALTER TABLE `membre`
  MODIFY `ID_MEM` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `tache`
  MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT;
```

- Après avoir finir les configuration pré cité vous herbergez le projet sur le server tomcat en fait ***un click droit sur le server tomcat*** puis ***add and remove***
- Après cette étape vous demarrez le server, assurez que votre serveur apache et mysql a demarré (xampp mamp wamp…)
- vous lancer le projet en faisant ***clic droit sur le projet*** puis ***run as*** vous choisissez ***run on server***
- vous lancer dans votre navigateur http://localhost:8080/TP_GESTAC/index.do

4- Test unitaire

Le test unitaire est une prodcedure permettant au developpeur de verifier le bon fonctionnement d'une partie d'une portion de son programme. Pour notre cas nous

avons effectué ce test à l'aide de la librairie Junit. Alors nous présentons quelque test que nous avons éffectué à savoir : ***le test de l'ajout de tache***, ***le test de l'afficher de tache par un mot clé*** *et* ***le test de l'ajout d'un membre*** . La figure ci-après présente les resultats obtenus

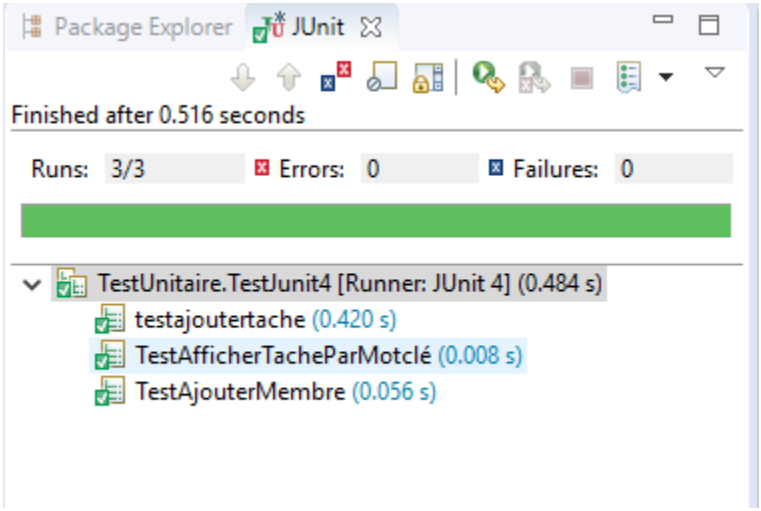


Figure7 : *Test Unitaire Junit.*

V- TEST D'ACCEPTATION

Dans cette partie de notre travail nous allons effectuer des tests sur notre application ces différentes tests sont resumés par les captutres d'ecrants suivants. Pour une question de logique nous allons presenter ces differents captures en fonction des exigence du TP.

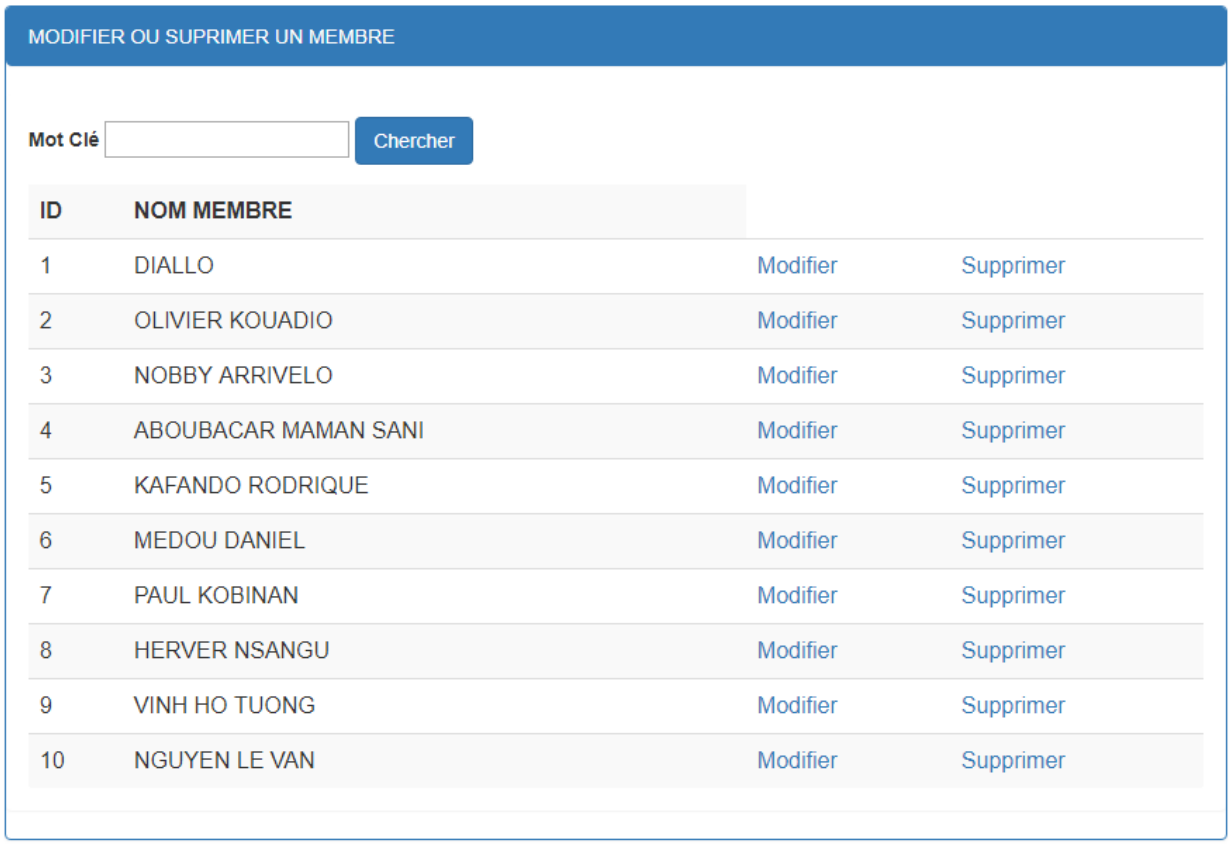
1- AJOUTER UN MEMBRE

Vous clickez sur le menu ***SAISIR MEMBRE*** vous remplissez le champ Nom

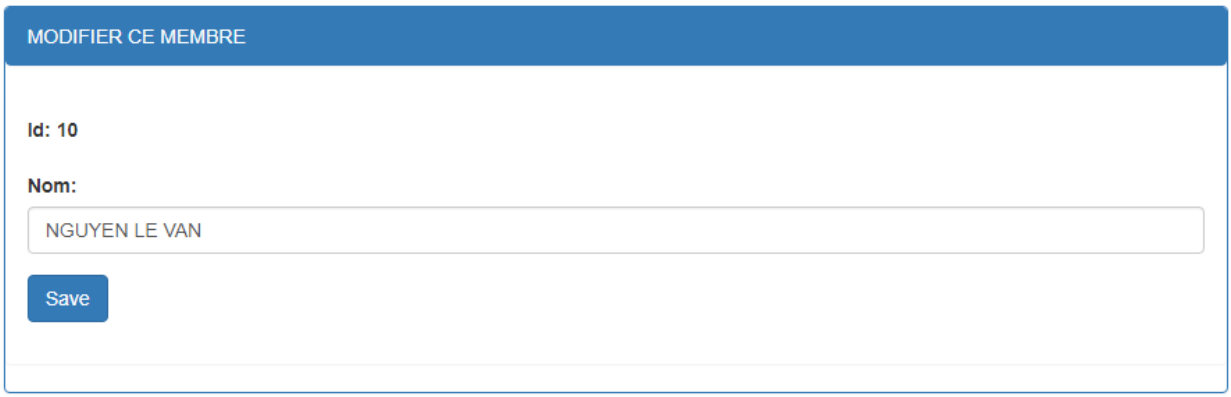
Après validation en appuyant sur le bouton save vous avez la confirmation dur membre ajouté

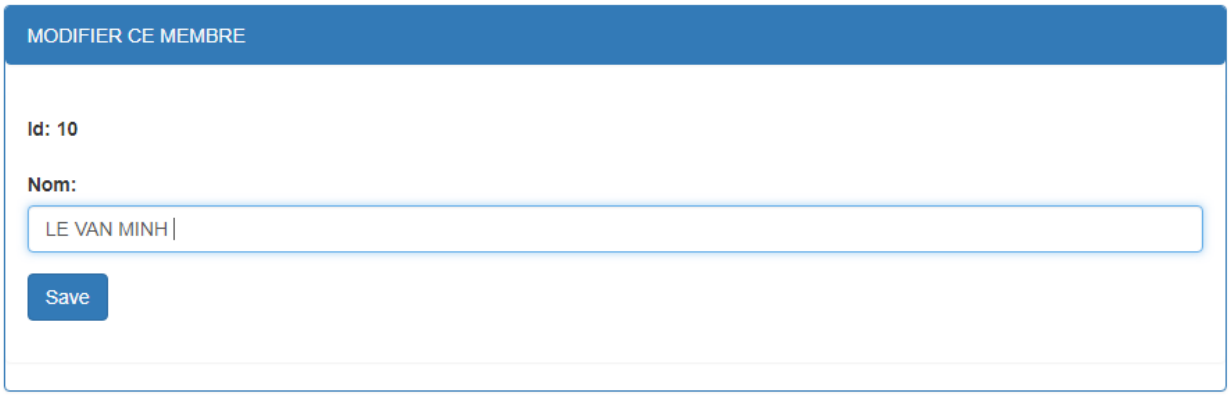
2- MODIFIER UN MEMBRE

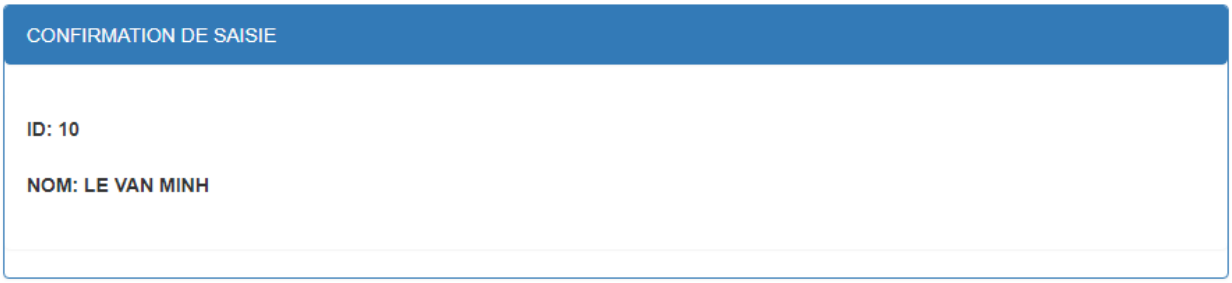
Pour modifier un membre vous affichez d'abord la liste des personnes pour être sûr que ce membre existe et que c'est lui que vous le modifier.Pour cela clikez sur le menu ***MODIFIER/SUPRIMER*** puis le bouton ***RECHERCHER***



A présent nous allons modifier pour exemple le ***membre 10*** au lieu de NGUYEN LE VAN nous allons mettre LE VAN MINH donc nous clickons sur modifier devant le membre en question et le système nous renvoi les information sur ce membre

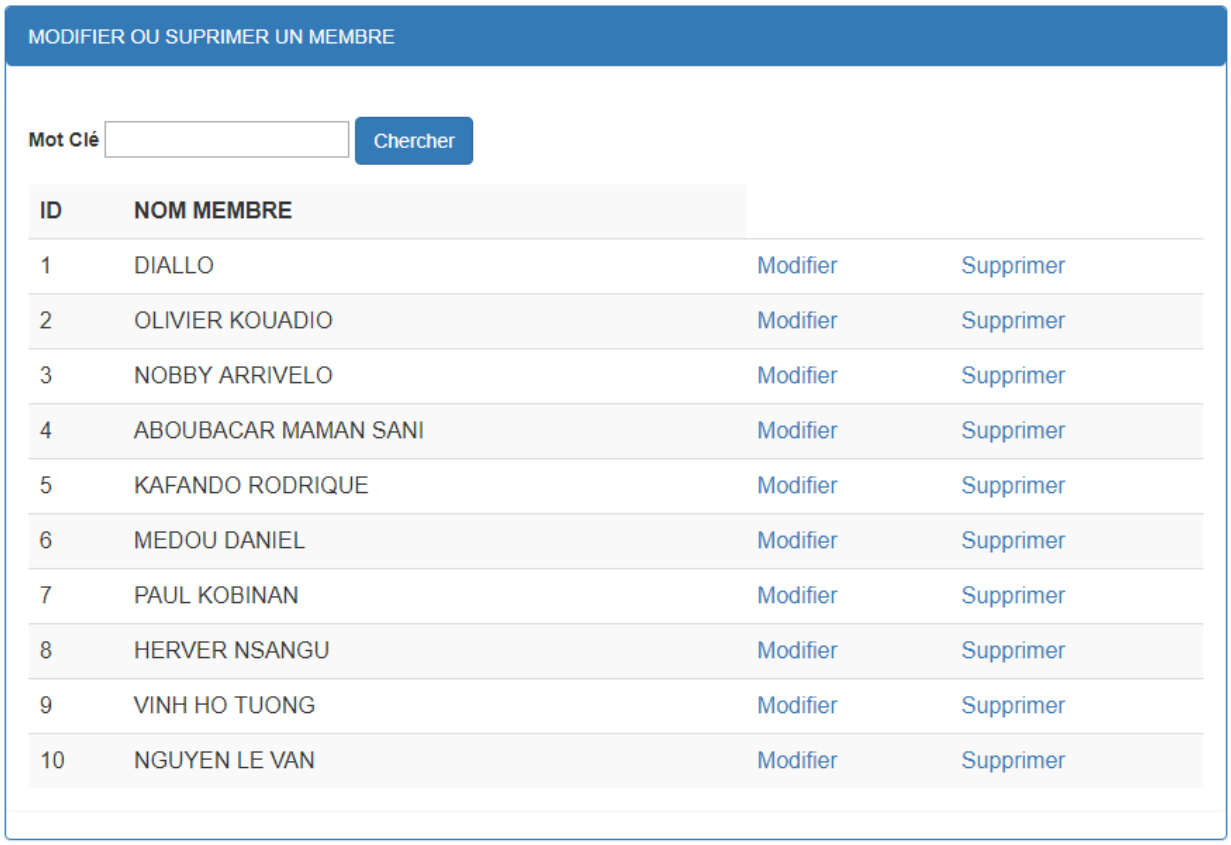




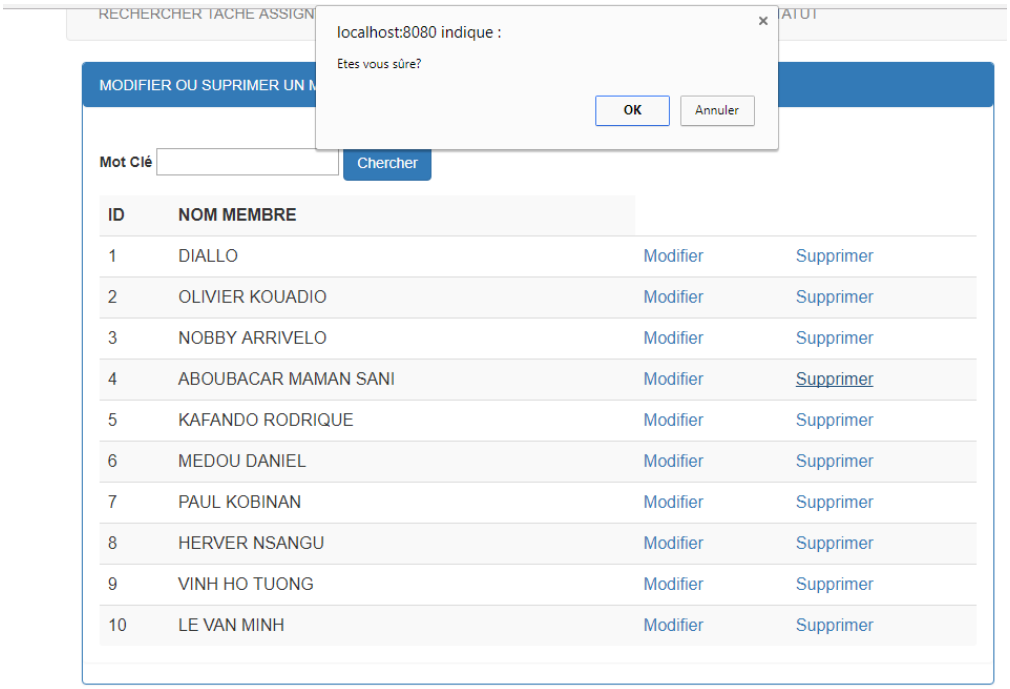


3- SUPPRIMER UN MEMBRE

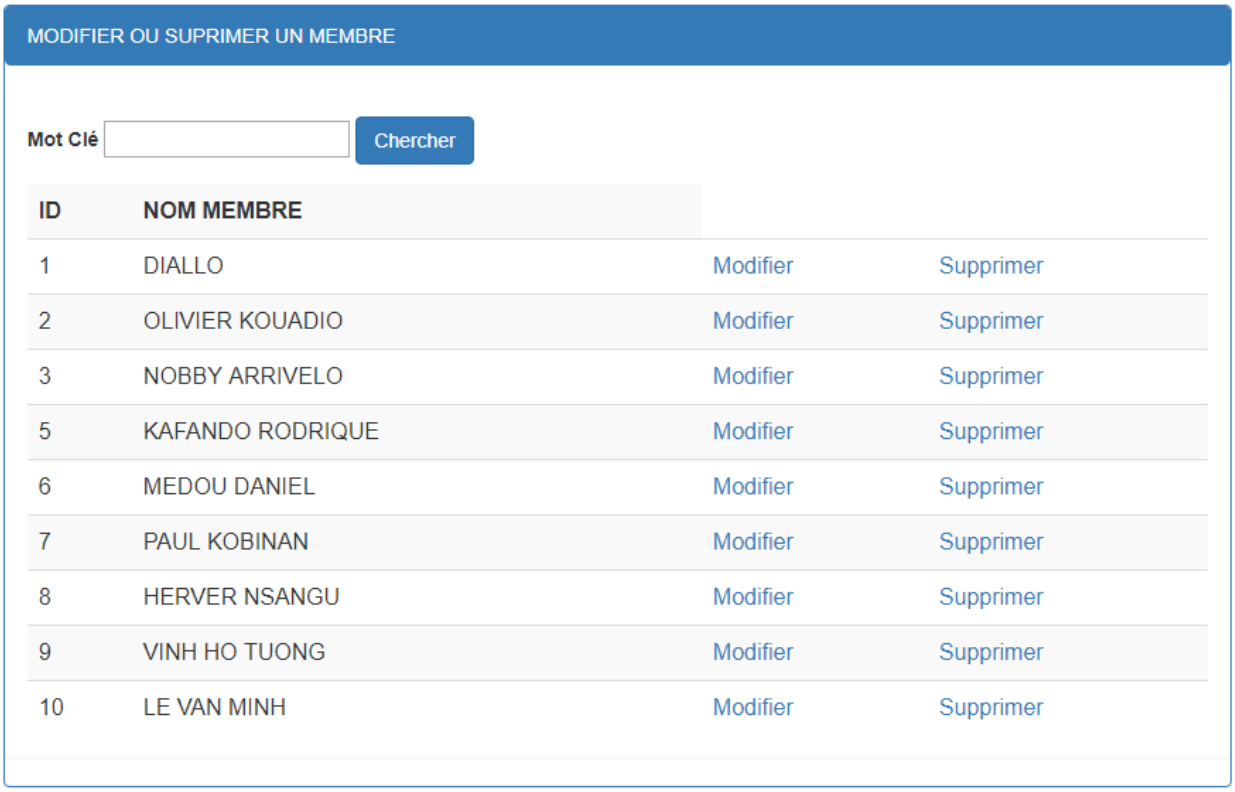
Pour supprimer un membre également vous affichez d'abord la liste des personnes pour être sûr que ce membre existe et que c'est lui que vous le supprimer.Pour cela clikez sur le menu ***MODIFIER/SUPRIMER*** puis le bouton ***RECHERCHER***



A présent nous allons suprimer pour exemple le ***membre 4*** donc nous clickons sur supprimer devant le membre en question et le système nous renvoi une fenêtre de confirmation.



Nous clickons sur ok pour confirmer la suppression et le membre est supprimer .



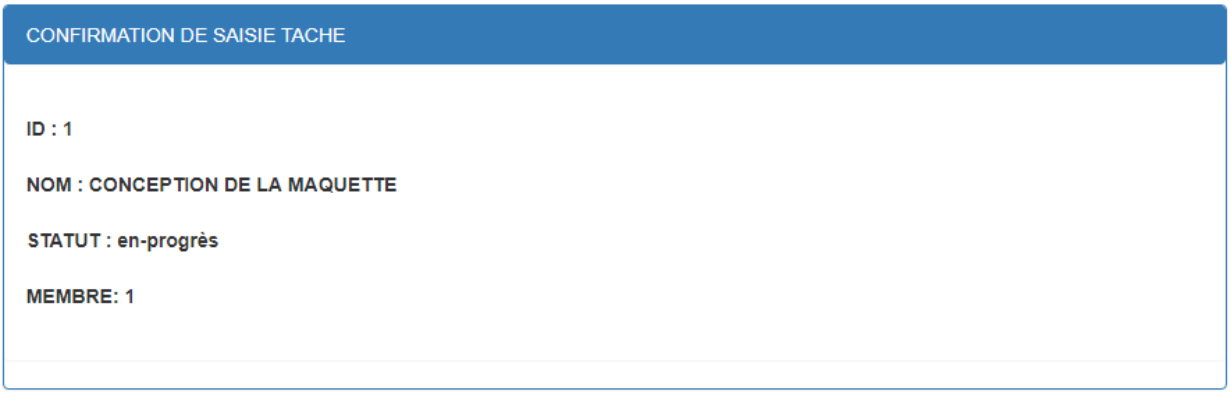
NB : **la suppression et la modification de la tâche se font de la même façon que les processus que nous venons décrire pour le membre.**

4- AJOUTER UNE TACHE

La petite précisionà faire c'est nous assignons la tâche à un membre de deux manière.

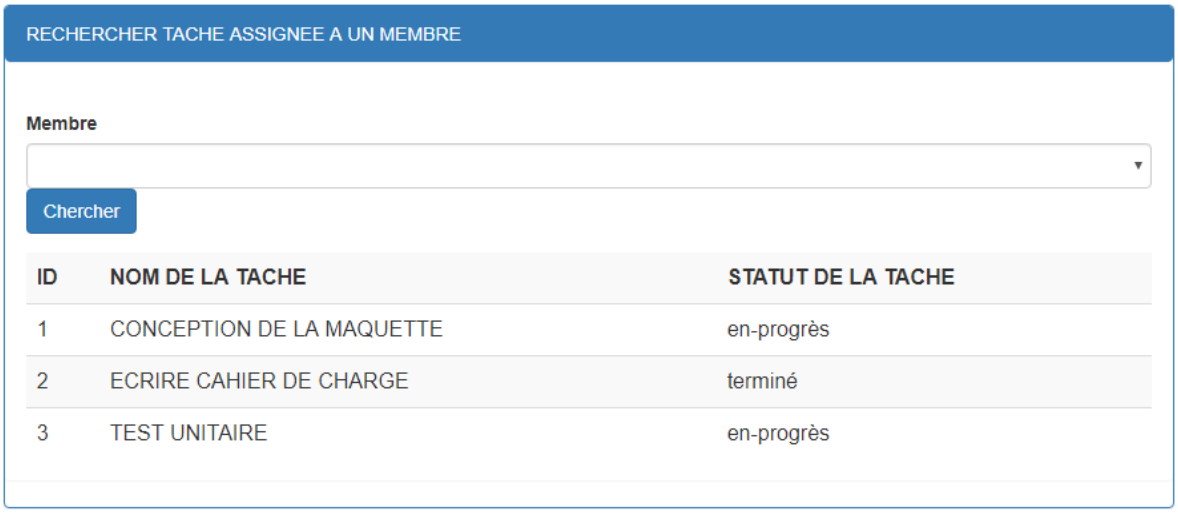
- Soit a la creation de la tâche nous affiche la liste des membres et l'utilisateur peut choisir le membre à qui il veut le lui assiger.
- Soit quqnt on créer la tâche elle reste non assignée et après on l'assigne a un membre en faisant simplement la modification de la tâche.

Nous montre dans ce cadre première option la deuxième ce fait comme dans la modification décrite précedement. Donc vous clikez sur le menu **SAISIR TACHE**, vous remplissez le nom de la tâche , la description et vous choisissez dans la liste deroulante le statut et le membre à qui vous voulez assigner la tache.



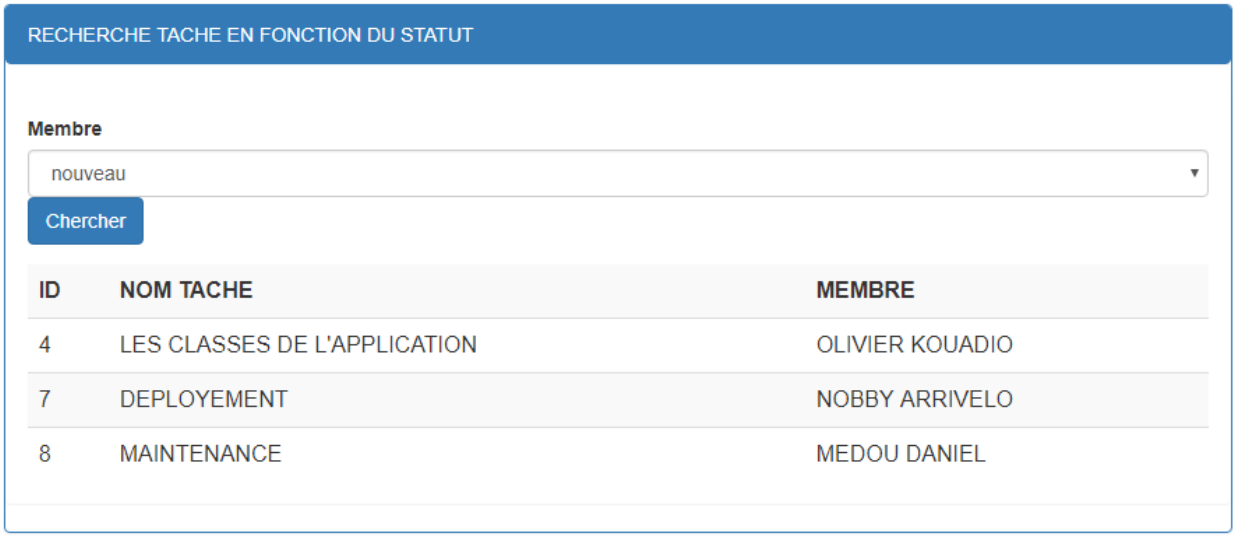
5- RECHERCHER UNE TACHE ASSIGNER A UN MEMBRE

Clickez sur le menu correspondant à cette opération puis vous choisissez le membre dans la liste deroulante et vous clickez sur chercher



6- RECHERCHER UNE TACHE EN FONCTION DU STATUT

Clickez sur le menu correspondant à cette opération puis vous choisissez le le statut dans la liste deroulante et vous clickez sur chercher

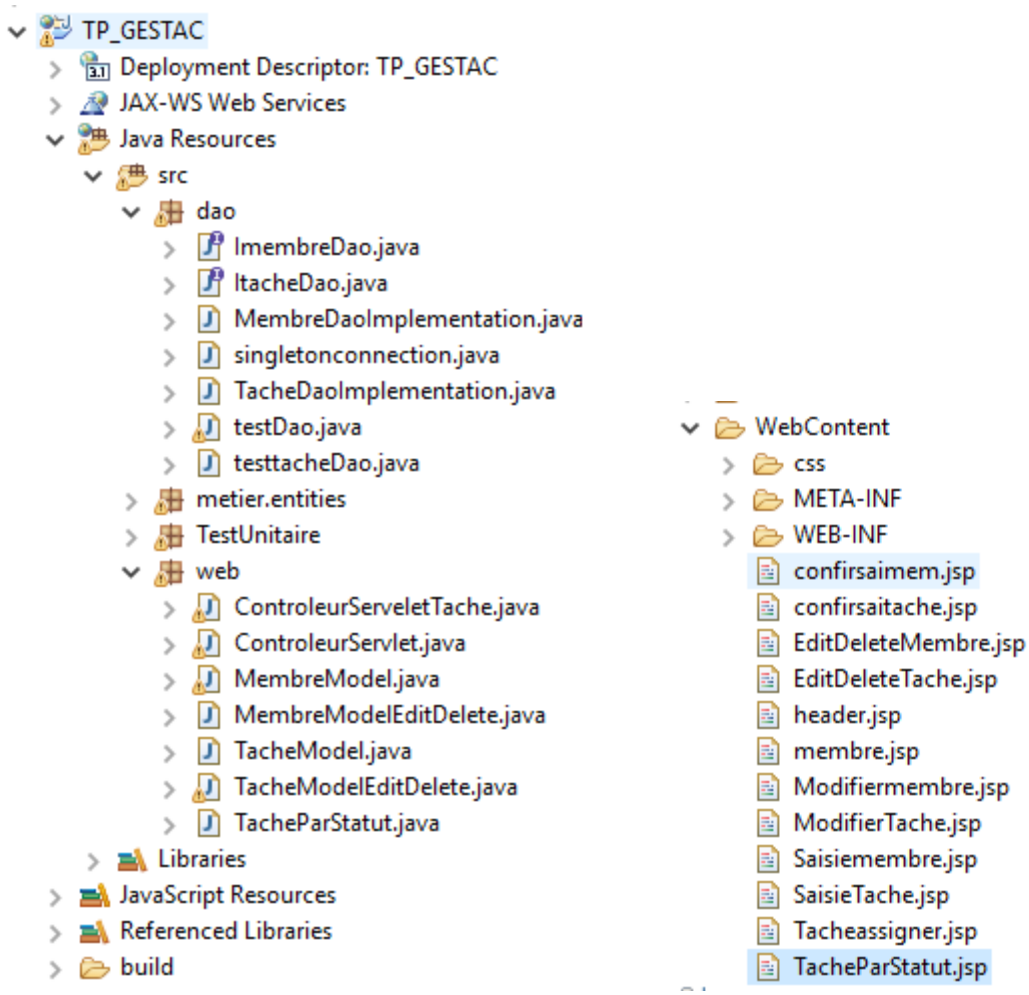


V- CONCLUSION

Au terme de ce TP nous pensons avoir atteint les objectifs fixés en ayant amélioré notre compréhension de la programmation orientée objet avec java. Par ailleurs les spécifications énumérées plus haut ont toutes été respectées et notre application est fonctionnelle. Toutefois des efforts restent à fournir dans le sens de la réflexion orientée objet et de la maîtrise du langage java mais aussi de la mise en application des bonnes pratiques de programmation notamment en programmation MVC. Nous retenons en conclusion que ce TP a été un excellent exercice pour l'assimilation des concepts de base du cours de génie logiciel.

VI- CODES SOURCES

1- Présentation de l'explorateur package de l'application



2- La classe connexion à la base de données

```
3- package dao;
4- import java.sql.Connection;
5- import java.sql.DriverManager;
6- public class singletonconnection {
7-        private static Connection connection;
8-        static{
9-              try {
10-                    Class.forName("com.mysql.jdbc.Driver");
11-connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/BD_G
   ESTAC", "root", "");
12-             } catch (Exception e) {
13-                   // TODO Auto-generated catch block
14-                   e.printStackTrace();
15-             }
16-       }
17-       public static Connection getConnection() {
18-             return connection;
19-       }
20- }
```

3- L'OBJET MEMBRE

❖ *Interface : **ImembreDao***

```
package dao;
import java.util.ArrayList;
import metier.entities.Membre;
public interface ImembreDao {
	public ArrayList<Membre> membresearch(String ms) ;
	public Membre save(Membre m);
	public ArrayList<Membre> membresparMC(int mc);
	public ArrayList<Membre> tousmembresparMC(String mced);
	public Membre getMembre(int id);
	public Membre updateMembre (Membre m);
	public void deleteMembre(int id);
	public ArrayList<Membre> getListMembre();
}
```

❖ *Implementation de l'interface : **MembreDaoImplementation***

```
package dao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import metier.entities.Membre;
public class MembreDaoImplementation implements ImembreDao {
	@Override
	public Membre save(Membre m) {
	Connection connection=singletonconnection.getConnection();
	try {
		PreparedStatement ps =connection.prepareStatement
				("INSERT INTO MEMBRE(NOM_MEM) VALUES (?)");
		ps.setString(1, m.getNom());
		ps.executeUpdate();
		PreparedStatement ps2=connection.prepareStatement
				("SELECT MAX(ID_MEM) AS MAX_ID FROM MEMBRE");
		ResultSet rs=ps2.executeQuery();
		if(rs.next()){

			m.setId(rs.getInt("MAX_ID"));
		}
		ps.close();
	} catch (SQLException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
		return m;
```

```
}
@Override
public ArrayList<Membre> membresparMC(int mc) {
	ArrayList<Membre> membre =new ArrayList<Membre>();
	Connection connection=singletonconnection.getConnection();
	try {
		PreparedStatement ps =connection.prepareStatement
				("SELECT * FROM MEMBRE WHERE ID_MEM LIKE ?");
		ps.setInt(1, mc);
		ResultSet rs=ps.executeQuery();
		while(rs.next()){
			Membre m=new Membre();
		    m.setId(rs.getInt("ID_MEM"));
		    m.setNom(rs.getString("NOM_MEM"));
		    membre.add(m);
		}
	} catch (SQLException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
	return membre;
}
@Override
public ArrayList<Membre> getListMembre() {
	ArrayList<Membre> membre =new ArrayList<>();
	Connection connection=singletonconnection.getConnection();
	try {
		PreparedStatement ps =connection.prepareStatement
				("SELECT * FROM MEMBRE ");
		ResultSet rs=ps.executeQuery();
		while(rs.next()){

			Membre m=new Membre();
		    m.setId(rs.getInt("ID_MEM"));
		    m.setNom(rs.getString("NOM_MEM"));
		    membre.add(m);
		}
	} catch (SQLException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
	return membre;
}
@Override
public ArrayList<Membre> tousmembresparMC(String mced) {
	ArrayList<Membre> membre =new ArrayList<Membre>();
	Connection connection=singletonconnection.getConnection();
	try {
		PreparedStatement ps =connection.prepareStatement
				("SELECT * FROM MEMBRE WHERE NOM_MEM LIKE ?");
		ps.setString(1, mced);
		ResultSet rs=ps.executeQuery();
		while(rs.next()){
```

```
                    Membre m=new Membre();
                    m.setId(rs.getInt("ID_MEM"));
                    m.setNom(rs.getString("NOM_MEM"));
                    membre.add(m);
                }
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        return membre;
}
@Override
public Membre getMembre(int id) {
        Membre m=null;
        Connection connection=singletonconnection.getConnection();
        try {
                PreparedStatement ps =connection.prepareStatement
                                ("SELECT * FROM MEMBRE WHERE ID_MEM=?");
                ps.setInt(1, id);
                ResultSet rs=ps.executeQuery();
                if(rs.next()){
                        m=new Membre();
                    m.setId(rs.getInt("ID_MEM"));
                    m.setNom(rs.getString("NOM_MEM"));
                }
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        return m;
}
@Override
public Membre updateMembre(Membre m) {
        Connection connection=singletonconnection.getConnection();
        try {
                PreparedStatement ps =connection.prepareStatement
                                ("UPDATE  MEMBRE SET NOM_MEM=? WHERE ID_MEM=?");
                ps.setString(1, m.getNom());
                ps.setInt(2, m.getId());
                ps.executeUpdate();
                ps.close();

        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
                return m;
}
@Override
public void deleteMembre(int id) {
        Connection connection=singletonconnection.getConnection();
        try {
                PreparedStatement ps =connection.prepareStatement
                                ("DELETE FROM MEMBRE WHERE ID_MEM=?");
```

```
                ps.setInt(1, id);
                ps.executeUpdate();
                ps.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        @Override
        public ArrayList<Membre> membresearch(String ms) {
            // TODO Auto-generated method stub
            return null;
        }

}
```

❖ *Classe membre : **Membre***

```
package metier.entities;
import java.io.Serializable;
public class Membre implements Serializable {
    private int id;
    private String nom;
    public Membre() {
        super();
        // TODO Auto-generated constructor stub
    }
    public Membre(String nom) {
        super();
        this.nom = nom;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
    @Override
    public String toString() {
        return "Membre [id=" + id + ", nom=" + nom + "]";
    }
}
```

❖ *Le modèle1 : **MembreModelEditDelete***

```
package web;
```

```
import java.util.ArrayList;
import metier.entities.Membre;
public class MembreModelEditDelete {
	private String motcle;
	private ArrayList<Membre> membre=new ArrayList<>();
	public String getMotcle() {
		return motcle;
	}
	public void setMotcle(String motcle) {
	this.motcle = motcle;
	}
	public ArrayList<Membre> getMembre() {
		return membre;
	}
	public void setMembre(ArrayList<Membre> membre) {
		this.membre = membre;
	}
 }
```

❖ *Le modèle2 : **MembreModel***

```
package web;
    import java.util.ArrayList;
    import java.util.List;
    import metier.entities.Membre;
    public class MembreModel {
	private int motcle;
	private ArrayList<Membre> membre=new ArrayList<Membre>();
	public int getMotcle() {
		return motcle;
	}
	public void setMotcle(int motcle) {
		this.motcle = motcle;
	}
	public ArrayList<Membre> getMembre() {
		return membre;
	}
	public void setMembre(ArrayList<Membre> membre) {
		this.membre = membre;
	}
    }
```

❖ *LE CONTROLEUR SERVLET*

```
package web;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import dao.ImembreDao;
import dao.ItacheDao;
import dao.MembreDaoImplementation;
import dao.TacheDaoImplementation;
import metier.entities.Membre;
import metier.entities.Tache;
@SuppressWarnings("serial")

//////deployer le controleur sur le serveur tomcat//////
@WebServlet(name = "serveletmembre", urlPatterns = { "*.do" })

public class ControleurServlet extends HttpServlet {
	private ImembreDao metier;
	private ItacheDao metier2;
	@Override
	public void init() throws ServletException {
		metier = new MembreDaoImplementation();
		metier2 = new TacheDaoImplementation();
	}
	@Override
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
			throws ServletException, IOException {
		String path = request.getServletPath();

/////////////////////////// CONTROLEUR DE MEMBRE//////////////////////////////////
		if (path.equals("/index.do")) {
request.getRequestDispatcher("EditDeleteMembre.jsp").forward(request, response);
		}
		else if (path.equals("/chercher.do")) {

			int motcle;
			motcle = Integer.parseInt(request.getParameter("motCle"));
			MembreModel model = new MembreModel();
			model.setMotcle(motcle);
			ArrayList<Membre> membre = metier.membresparMC(motcle);
			model.setMembre(membre);
			request.setAttribute("model", model);
			request.getRequestDispatcher("membre.jsp").forward(request,
response);
		} else if (path.equals("/Saisie.do")) {
			request.getRequestDispatcher("Saisiemembre.jsp").forward(request,
response);
		}
		if (path.equals("/index.do")) {

	request.getRequestDispatcher("EditDeleteMembre.jsp").forward(request,
response);
		} else if (path.equals("/EditDeleteMembre.do")) {
		String motcle = request.getParameter("motCle");
		MembreModelEditDelete model = new MembreModelEditDelete();
		model.setMotcle(motcle);
		ArrayList<Membre> membre = metier.tousmembresparMC("%" + motcle + "%");
	    model.setMembre(membre);
	    request.setAttribute("model", model);
```

```
            request.getRequestDispatcher("EditDeleteMembre.jsp").forward(request,
response);
            }
            else if (path.equals("/Saisie.do")) {
            request.getRequestDispatcher("Saisiemembre.jsp").forward(request,
response);
            }
else if (path.equals("/SaveMembre.do") && (request.getMethod().equals("POST"))) {
                  String noms = request.getParameter("nom");
                  Membre m = metier.save(new Membre(noms));
                  request.setAttribute("membre", m);
                  request.getRequestDispatcher("confirsaimem.jsp").forward(request,
response);
            } else if (path.equals("/Supprimer.do")) {
                  int id = Integer.parseInt(request.getParameter("id"));
                  metier.deleteMembre(id);

      request.getRequestDispatcher("EditDeleteMembre.do?motCle=").forward(request,
response);

            } else if (path.equals("/Modifier.do")) {
                  int id = Integer.parseInt(request.getParameter("id"));
                  Membre m = metier.getMembre(id);
                  request.setAttribute("membre", m);
            request.getRequestDispatcher("Modifiermembre.jsp").forward(request,
response);
            } else if (path.equals("/UpdateMembre.do") &&
(request.getMethod().equals("POST"))) {
                  int id = Integer.parseInt(request.getParameter("id"));
                  String noms = request.getParameter("nom");
                  Membre m = new Membre(noms);
                  m.setId(id);
                  metier.updateMembre(m);
                  request.setAttribute("membre", m);
                  request.getRequestDispatcher("confirsaimem.jsp").forward(request,
response);
            }
////////////// FIN CONTROLEUR DE MEMBRECONTROLEUR DE TACHE//////////////////////////
            if (path.equals("/index.do")) {
            request.getRequestDispatcher("EditDeleteMembre.jsp").forward(request,
response);
            } else if (path.equals("/EditDeleteTache.do")) {
            String motcletache = request.getParameter("motCle");
            TacheModelEditDelete model = new TacheModelEditDelete();
            model.setMotcletache(motcletache);
            ArrayList<Tache> tache = metier2.tacheparMC("%" + motcletache + "%");
            model.setTaches(tache);
            request.setAttribute("model", model);
            request.getRequestDispatcher("EditDeleteTache.jsp").forward(request,
response);
            }
            else if (path.equals("/SaisieTache.do")) {
            ArrayList<Membre> membre = metier.getListMembre();
            MembreModelEditDelete model = new MembreModelEditDelete();
```

```
            model.setMembre(membre);
            request.setAttribute("model", model);
            request.getRequestDispatcher("SaisieTache.jsp").forward(request,
response);
            } else if (path.equals("/SaveTache.do") &&
(request.getMethod().equals("POST"))) {
            String nom = request.getParameter("nom");
            String description = request.getParameter("description");
            String statut = request.getParameter("statut");
            int mem_fk = Integer.parseInt(request.getParameter("mem_fk"));
            Tache t = metier2.save(new Tache(nom, description, statut, mem_fk));
            request.setAttribute("tache", t);

      request.getRequestDispatcher("confirsaitache.jsp").forward(request, response);
            }
            else if (path.equals("/SupprimerTache.do")) {
                  int id = Integer.parseInt(request.getParameter("id"));
                  metier2.deleteTache(id);
request.getRequestDispatcher("EditDeleteTache.do?motCle=").forward(request,
response);
            }
            else if (path.equals("/ModifierTache.do")) {
                  int id = Integer.parseInt(request.getParameter("id"));
                  Tache t = metier2.getTache(id);
                  request.setAttribute("tache", t);

request.getRequestDispatcher("ModifierTache.jsp").forward(request, response);
            } else if (path.equals("/UpdateTache.do") &&
(request.getMethod().equals("POST"))) {
                  int id = Integer.parseInt(request.getParameter("id"));
                  String nom = request.getParameter("nom");
                  String description = request.getParameter("description");
                  String statut = request.getParameter("statut");
                  int mem_fk = Integer.parseInt(request.getParameter("mem_fk"));
                  Tache t = new Tache(nom, description, statut, mem_fk);
                  t.setId(id);
                  metier2.updateTache(t);
                  request.setAttribute("tache", t);

request.getRequestDispatcher("confirsaitache.jsp").forward(request, response);
            }
            if (path.equals("/Tacheassignation.do")) {
                  ArrayList<Membre> membre = metier.getListMembre();
                  MembreModelEditDelete model = new MembreModelEditDelete();
                  model.setMembre(membre);
                  request.setAttribute("model", model);

request.getRequestDispatcher("Tacheassigner.jsp").forward(request, response);
            }
            else if (path.equals("/Tacheassigner.do")) {
                  int idmembre;
                  idmembre = Integer.parseInt(request.getParameter("idmembre"));
                  TacheModel model1 = new TacheModel();
                  model1.setIdmembre(idmembre);
```

```
  ArrayList<Tache> tache = metier2.getListTachebyIdMembre(idmembre);
                    model1.setTache(tache);
                    request.setAttribute("model1", model1);

        request.getRequestDispatcher("Tacheassigner.jsp").forward(request, response);
              }
          else if (path.equals("/TacheStatut.do")) {

                String motcletache = request.getParameter("statut");
                    TacheParStatut model2 = new TacheParStatut();
                    model2.setMotcletacheps(motcletache);
                    ArrayList<Tache> tache= metier2.TacheParStatut(motcletache);
                    model2.setTaches(tache);
                    request.setAttribute("model2", model2);

        request.getRequestDispatcher("TacheParStatut.jsp").forward(request, response);
              }
////////////////////// FIN CONTROLEUR DE TACHE///// ////////////////////////////////
              else {
                    response.sendError(response.SC_NOT_FOUND);
              }
        }

        @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse
response)
                    throws ServletException, IOException {
              doGet(request, response);
        }
    }
```

❖ *Les différentes vues JSP de Membre :*

Header

```
<p></p>
<div class=" navbar navbar-default container col-md-8 col-md-offset-2">
      <ul class="nav navbar-nav">
            <li> <a href="Saisie.do" >SAISIR MEMBRE</a></li>
            <li> <a href="EditDeleteMembre.do">MODIFIER /SUPPRIMER MEMBRE</a></li>
            <li> <a href="SaisieTache.do">SAISIR TACHE</a></li>
            <li> <a href="EditDeleteTache.do">MODIFIER /SUPPRIMER TACHE</a></li>
            <li> <a href=Tacheassignation.do> RECHERCHER TACHE ASSIGNEE A UN MEMBRE
</a></li>
            <li> <a href="TacheStatut.do"> RECHERCHER TACHE EN FONCTION DE STATUT
</a></li>
      </ul>
</div>
```

Saisie membre

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html">
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Saisie membre</title>
<link rel="stylesheet" type="text/css"href="css/bootstrap.min.css">
</head>
<body>
<%@include file="header.jsp" %>
<div  class="container col-md-8 col-md-offset-2 col-xs-12">
	<div class="panel panel-primary">
		<div class=" panel panel-heading">SAISIE MEMBRE </div>
		<div class="panel panel-body">
			<form action="SaveMembre.do" method="post">
				<div class="form-group">
					<label class="control-label">Nom:</label>
					<input type="text" name="nom" value="" class="form-
control" required="required">
					<span></span>
				</div>
				<div>
					<button type="submit" class="btn btn-
primary">Save</button>
				</div>
			</form>
		</div>
	</div>
</div>
</body>
</html>
```

Confirmation saisie membre

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Confirmation saisie membre</title>
<link rel="stylesheet" type="text/css"href="css/bootstrap.min.css">
</head>
<body>
<%@include file="header.jsp" %>
<div  class="container col-md-8 col-md-offset-2 col-xs-12">
	<div class="panel panel-primary">
		<div class=" panel panel-heading">CONFIRMATION DE SAISIE</div>
		<div class="panel panel-body">
			<div class=form-group>
				<label>ID:</label>
				<label>${membre.id }</label>
			</div>
			<div class=form-group>
				<label>NOM:</label>
				<label>${membre.nom }</label>
			</div>
		</div>
```

```
		</div>
	</div>
	</body>
	</html>
```

Liste des membres

```
	<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
	<!DOCTYPE html">
	<html>
	<head>
	<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
	<title>Membre</title>
	<link rel="stylesheet" type="text/css"href="css/bootstrap.min.css">
	</head>
	<body>
	<%@include file="header.jsp" %>
	<div  class="container col-md-8 col-md-offset-2">
		<div class="panel panel-primary">
			<div class=" panel panel-heading">MODIFIER OU SUPRIMER UN MEMBRE</div>
			<div class="panel panel-body">
				<form action="EditDeleteMembre.do" method="get">
					<label>Mot Clé</label>
					<input type="text"name='motCle' value="${model.motcle}">
					<button type="submit" class="btn btn-primary">
	Chercher</button>
				</form>
				<table class="table table-striped">
					<tr>
						<th>ID</th><th>NOM MEMBRE</th>
					</tr>
					<c:forEach items="${model.membre}" var="m">
						<tr>
							<td>${m.id} </td>
							<td>${m.nom} </td>
							<td><a
	href="Modifier.do?id=${m.id}">Modifier</a> </td>
							<td><a onclick="return confirm('Etes vous
	sûre?')" href="Supprimer.do?id=${m.id}">Supprimer</a> </td>
						</tr>
					</c:forEach>
				</table>

			</div>
		</div>
	</div>
	</body>
	</html>
```

Modifer membre

```
<!DOCTYPE html">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

```
<title>Saisie membre</title>
<link rel="stylesheet" type="text/css"href="css/bootstrap.min.css">
</head>
<body>
<%@include file="header.jsp" %>
<div  class="container col-md-8 col-md-offset-2 col-xs-12">
    <div class="panel panel-primary">
          <div class=" panel panel-heading">MODIFIER CE MEMBRE </div>
          <div class="panel panel-body">
                <form action="UpdateMembre.do" method="post">
                      <div class="form-group">
                            <label class="control-label">Id:</label>
                            <input type="hidden" name="id" value="${membre.id }"
    class="form-control" required="required">
                            <label>${membre.id }</label>
                            <span></span>
                      </div>
                      <div class="form-group">
                            <label class="control-label">Nom:</label>
                            <input type="text" name="nom" value="${membre.nom }"
    class="form-control" required="required">
                            <span></span>
                      </div>
                      <div>
                            <button type="submit" class="btn btn-
    primary">Save</button>
                      </div>
                </form>
          </div>
    </div>
</div>
</body>
</html>
```

4-L'OBJET TACHE

❖ *Interface ItacheDao*

```
package dao;
import java.util.ArrayList;
import metier.entities.Membre;
import metier.entities.Tache;
public interface ItacheDao {
      public Tache save(Tache t);
      public ArrayList<Tache> tacheparMC(String mc);
      public ArrayList<Tache> getListTachebyIdMembre(int idmembre);
      public Tache updateTache (Tache t);
      public void deleteTache(int id);
      public ArrayList<Membre> tousmembresparMC(String mced);
      public Membre getMembre(int id);
      public Tache getTache(int id);
      public ArrayList<Tache> TacheParStatut(String tps);
}
```

❖ *Implementation de la tache: TacheDaoImplementation*

```
package dao;

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


import metier.entities.Membre;
import metier.entities.Tache;

public class TacheDaoImplementation implements ItacheDao {

	@Override
	public Tache save(Tache t) {
	Connection connection=singletonconnection.getConnection();
	try {
		PreparedStatement ps =connection.prepareStatement

		("INSERT INTO TACHE (NOM_TAC,DESCRIPTION,STATUT,MEM_fk) VALUES
(?,?,?,?)");
		ps.setString(1, t.getNom());
		ps.setString(2, t.getDescription());
		ps.setString(3, t.getStatut());
		ps.setInt(4, t.getMem_fk());
		ps.executeUpdate();
		PreparedStatement ps2=connection.prepareStatement
				("SELECT MAX(ID) AS MAX_ID FROM TACHE");
		ResultSet rs=ps2.executeQuery();
		if(rs.next()){
			t.setId(rs.getInt("MAX_ID"));
		}
		ps.close();
	} catch (SQLException e) {
		// TODO Auto-generated catch block
		e.printStackTrace();
	}
		return t;
	}
////////////////////////////////////////////////////////////////////////////
///// recherche globale
////////////////////////////////////////////////////
	@Override
	public ArrayList<Tache> tacheparMC(String mc) {
		ArrayList<Tache> tache =new ArrayList<Tache>();
		Connection connection=singletonconnection.getConnection();
		try {
```

```
                PreparedStatement ps =connection.prepareStatement
                            ("SELECT * FROM TACHE WHERE NOM_TAC LIKE ?");
                ps.setString(1, mc);
                ResultSet rs=ps.executeQuery();
                while(rs.next()){
                       Tache t=new Tache();
                    t.setId(rs.getInt("ID"));
                    t.setNom(rs.getString("NOM_TAC"));
                    t.setDescription(rs.getString("DESCRIPTION"));
                    t.setStatut(rs.getString("STATUT"));
                    t.setMem_fk(rs.getInt("MEM_fk"));
                    tache.add(t);
                }
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return tache;
        }
        @Override
        public ArrayList<Tache> TacheParStatut(String tps) {
            ArrayList<Tache> tache =new ArrayList<Tache>();
            Connection connection=singletonconnection.getConnection();
            try {
                PreparedStatement ps =connection.prepareStatement
                            ("SELECT t.*, m.NOM_MEM FROM TACHE t, MEMBRE m WHERE
t.MEM_fk = m.ID_MEM AND t.STATUT LIKE ?");
                ps.setString(1, "%"+tps+"%");
                ResultSet rs=ps.executeQuery();
                while(rs.next()){

                       Tache t=new Tache();
                    t.setId(rs.getInt("ID"));
                    t.setNom(rs.getString("NOM_TAC"));
                    t.setDescription(rs.getString("DESCRIPTION"));
                    t.setMem_fk(rs.getInt("MEM_fk"));
                    t.setMembre(rs.getString("NOM_MEM"));
                    tache.add(t);
                }
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return tache;
        }

////////////////////////////rechercher par id de
membre//////////////////////////////////////
        @Override
        public ArrayList<Tache> getListTachebyIdMembre(int idmembre) {
            ArrayList<Tache> tache =new ArrayList<Tache>();
            Connection connection=singletonconnection.getConnection();
            try {
                PreparedStatement ps =connection.prepareStatement
```

```
                        ("SELECT * FROM TACHE WHERE MEM_fk= ?");
                ps.setInt(1, idmembre);
                ResultSet rs=ps.executeQuery();
                while(rs.next()){
                        Tache t=new Tache();
                    t.setId(rs.getInt("ID"));
                    t.setNom(rs.getString("NOM_TAC"));
                    t.setDescription("DESCRIPTION");
                    t.setStatut(rs.getString("STATUT"));
                    tache.add(t);
                }
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        return tache;
}
@Override
public Tache getTache(int id) {
        Tache t=null;
        Connection connection=singletonconnection.getConnection();
        try {
                PreparedStatement ps =connection.prepareStatement
                ("SELECT * FROM TACHE WHERE ID=?");
                ps.setInt(1, id);
                ResultSet rs=ps.executeQuery();
                if(rs.next()){
                        t=new Tache();
                        t.setId(rs.getInt("ID"));
                        t.setNom(rs.getString("NOM_TAC"));
                        t.setDescription(rs.getString("DESCRIPTION"));
                        t.setStatut(rs.getString("STATUT"));
                        t.setMem_fk(rs.getInt("MEM_fk"));
                }
        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        return t;
}
@Override
public Tache updateTache (Tache t) {
        Connection connection=singletonconnection.getConnection();
        try {
                PreparedStatement ps =connection.prepareStatement
                                ("UPDATE TAHCE SET NOM_TAC=? WHERE ID_MEM=?");
                ps.setString(1, t.getNom());
                ps.setInt(2,t.getId());
                ps.executeUpdate();
                ps.close();

        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
```

```
		}
			return t;
	}
	@Override
	public void  deleteTache(int id) {
		Connection connection=singletonconnection.getConnection();
		try {
			PreparedStatement ps =connection.prepareStatement
					("DELETE FROM TACHE WHERE ID=?");
			ps.setInt(1, id);
			ps.executeUpdate();
			ps.close();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
	@Override
	public ArrayList<Membre> tousmembresparMC(String mced) {
		// TODO Auto-generated method stub
		return null;
	}
	@Override
	public Membre getMembre(int id) {
		// TODO Auto-generated method stub
		return null;
	}
}
```

❖ *Classe membre :* ***Tache***

```
package metier.entities;

import java.io.Serializable;

public class Tache implements Serializable {

	private int id;
	private String nom;
	private String description;
	private String statut;
	private int mem_fk;
	private String membre;
	public Tache() {
		super();
		// TODO Auto-generated constructor stub
	}
	public Tache(String nom, String description, String statut, int mem_fk) {
		super();
		this.nom = nom;
		this.description = description;
		this.statut = statut;
		this.mem_fk = mem_fk;
	}
```

```
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getNom() {
		return nom;
	}
	public String getMembre() {
		return membre;
	}
	public void setMembre(String membre) {
		this.membre = membre;
	}
	public void setNom(String nom) {
		this.nom = nom;
	}
	public String getDescription() {
		return description;
	}
	public void setDescription(String description) {
		this.description = description;
	}
	public String getStatut() {
		return statut;
	}
	public void setStatut(String statut) {
		this.statut = statut;
	}
	public int getMem_fk() {
		return mem_fk;
	}
	public void setMem_fk(int mem_fk) {
		this.mem_fk = mem_fk;
	}
	@Override
	public String toString() {
		return "Tache [id=" + id + ", nom=" + nom + ", description=" +
description + ", statut=" + statut + ", mem_fk="
				+ mem_fk + "]";
	}
}
```

❖ *Model1 :* ***TacheModel***

```
package web;
import java.util.ArrayList;
import metier.entities.Tache;
public class TacheModel {
	private int idmembre;
	private ArrayList<Tache> tache=new ArrayList<Tache>();
	public int getIdmembre() {
		return idmembre;
	}
```

```
	public void setIdmembre(int idmembre) {
		this.idmembre = idmembre;
	}
	public ArrayList<Tache> getTache() {
		return tache;
	}
	public void setTache(ArrayList<Tache> tache) {
		this.tache = tache;
	}
	}
```

❖ *Model2 : **TacheModelEditDelete***

```
package web;
import java.util.ArrayList;
import metier.entities.Tache;
import metier.entities.Membre;
public class TacheModelEditDelete {
	private String motcletache;
	private ArrayList<Tache> taches=new  ArrayList<Tache>();
	public String getMotcletache() {
		return motcletache;
	}
	public void setMotcletache(String motcletache) {
		this.motcletache = motcletache;
	}
	public ArrayList<Tache> getTaches() {
		return taches;
	}
	public void setTaches(ArrayList<Tache> taches) {
		this.taches = taches;
	}
}
```

❖ *Model3 : **TacheParStatut***

```
package web;
import java.util.ArrayList;
import metier.entities.Tache;
public class TacheParStatut{
	String motcletacheps;
	private ArrayList<Tache> taches=new  ArrayList<Tache>();
	public String getMotcletacheps() {
		return motcletacheps;
	}
	public void setMotcletacheps(String motcletacheps) {
		this.motcletacheps = motcletacheps;
	}
	public ArrayList<Tache> getTaches() {
		return taches;
	}
	public void setTaches(ArrayList<Tache> taches) {
		this.taches = taches;
	}
```

}

❖ *Les différentes vues JSP de taches :*

Saisie tache

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Saisie Tache</title>
<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
</head>
<body>
<%@include file="header.jsp"%>
<div class="container col-md-8 col-md-offset-2 col-xs-12">
 <div class="panel panel-primary">
  <div class=" panel panel-heading">SAISIE TACHE</div>
  <div class="panel panel-body">
<form action="SaveTache.do" method="post">
   <div class="form-group">
        <label class="control-label">Nom Tache:</label> <input type="text"
       name="nom" value="" class="form-control"required="required"> <span></span>
   </div>
   <div class="form-group">
      <label class="control-label">Description:</label>
      <textarea class="form-control" rows="3" name="description"></textarea>
      <span></span>
   </div>
   <div class="form-group">
      <label class="control-label">Statut:</label>
       <select class="form-control" name="statut"value="statut" >
             <option value=" nouveau"> nouveau</option>
             <option value=" en-progrès"> en-progrès</option>
             <option value=" terminé"> terminé</option>
       </select>
       <span></span>
   </div>
    <div class="form-group">
      <label class="control-label">Assigner tache a membre</label>
      <select class="form-control" name="mem_fk">
             <c:forEach items="${model.membre}" var="m">
                       <option value="${m.id}">${m.nom}</option>
              </c:forEach>
       </select>
  </div>
  <div>
      <button type="submit" class="btn btn-primary">Save</button>
  </div>
</form>
```

```
</div>
</div>
</div>
</body>
</html>
```

Confirmation Saisie tache

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Confirmation saisie tache</title>
<link rel="stylesheet" type="text/css"href="css/bootstrap.min.css">
</head>
<body>
<%@include file="header.jsp" %>
<div  class="container col-md-8 col-md-offset-2 col-xs-12">
      <div class="panel panel-primary">
            <div class=" panel panel-heading">CONFIRMATION DE SAISIE TACHE</div>
            <div class="panel panel-body">
                  <div class=form-group>
                        <label>ID :</label>
                        <label>${tache.id }</label>
                  </div>
                  <div class=form-group>
                        <label>NOM :</label>
                        <label>${tache.nom }</label>
                  </div>
                  <div class=form-group>
                        <label>STATUT :</label>
                        <label>${tache.statut }</label>
                  </div>
                  <div class=form-group>
                        <label>MEMBRE:</label>
                        <label>${tache.mem_fk }</label>
                  </div>
            </div>
      </div>
</div>
</body>
</html>
```

Liste des taches

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Tache</title>
<link rel="stylesheet" type="text/css"href="css/bootstrap.min.css">
</head>
<body>
```

```
<%@include file="header.jsp" %>
<div  class="container col-md-8 col-md-offset-2">
	<div class="panel panel-primary">
		<div class=" panel panel-heading">MODIFIER OU SUPRIMER UNE TACHE</div>
		<div class="panel panel-body">
			<form action="EditDeleteTache.do" method="get">
				<label>Mot Clé</label>
				<input type="text"name='motCle'
value="${model.motcletache}">
				<button type="submit" class="btn btn-primary">
Chercher</button>
			</form>
			<table class="table table-striped">
				<tr>
					<th>ID</th>
					<th>NOM TACHE</th>
					<th>STATUT</th>
					<th>ID MEMBRE</th>
				</tr>
				<c:forEach items="${model.taches}" var="t">
					<tr>
						<td>${t.id} </td>
						<td>${t.nom} </td>
						<td>${t.statut} </td>
						<td>${t.mem_fk} </td>
						<td><a
href="ModifierTache.do?id=${t.id}">Modifier</a> </td>
						<td><a onclick="return confirm('Etes vous
sûre?')" href="SupprimerTache.do?id=${t.id}">Supprimer</a> </td>
					</tr>
				</c:forEach>
			</table>

		</div>
	</div>
</div>
</body>
</html>
```

Modifier taches

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Saisie membre</title>
<link rel="stylesheet" type="text/css"href="css/bootstrap.min.css">
</head>
<body>
<%@include file="header.jsp" %>
<div  class="container col-md-8 col-md-offset-2 col-xs-12">
	<div class="panel panel-primary">
		<div class=" panel panel-heading">MODIFICATION DE LA TACHE </div>
		<div class="panel panel-body">
```

```
                    <form action="UpdateTache.do" method="post">

                            <div class="form-group">
                                  <label class="control-label">Id:</label>
                                  <input type="hidden" name="id" value="${tache.id }"
class="form-control" required="required">
                                  <label>${tache.id }</label>
                                  <span></span>
                            </div>
                            <div class="form-group">
                                  <label class="control-label">Nom:</label>
                                  <input type="text" name="nom" value="${tache.nom }"
class="form-control" required="required">
                                  <span></span>
                            </div>

                            <div class="form-group">
                                  <label class="control-label">Description:</label>
                                  <textarea class="form-control" rows="3"
name="description" value="${tache.description }">${tache.description }</textarea>
                                  <span></span>
                            </div>
                            <div class="form-group">
                                        <label class="control-label">Statut:</label>
                                        <select class="form-control"
name="statut"value="statut" >
                                        <option value=" nouveau"> nouveau</option>
                                        <option value=" en-progrès"> en-
progrès</option>
                                        <option value=" terminé"> terminé</option>
                                        </select>
                                        <span></span>
                                  </div>
                            <div class="form-group">
                                  <label class="control-label">Membre:</label>
                                  <input type="hidden" name="mem_fk"
value="${tache.mem_fk }" class="form-control" required="required">
                                  <label>${tache.mem_fk }</label>
                                  <span></span>
                            </div>
                            <div>
                                  <button type="submit" class="btn btn-
primary">Save</button>
                            </div>
                    </form>
              </div>
       </div>
</div>
</body>
</html>
```

Assigner une tache a un membre

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html">
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Membre</title>
<link rel="stylesheet" type="text/css"href="css/bootstrap.min.css">
</head>
<body>
<%@include file="header.jsp" %>
<div  class="container col-md-8 col-md-offset-2">
      <div class="panel panel-primary">
            <div class=" panel panel-heading">RECHERCHER TACHE ASSIGNEE A UN
MEMBRE</div>
            <div class="panel panel-body">
                  <form action="Tacheassigner.do" method="get">
                        <label class="control-label">Membre</label>
                              <select class="form-control" name='idmembre'>
                                    <c:forEach items="${model.membre}"
var="m">
                                          <option
value="${m.id}">${m.nom}</option>
                                    </c:forEach>
                              </select>
                        <button type="submit" class="btn btn-primary">
Chercher</button>
                  </form>
                  <table class="table table-striped">
                        <tr>
                              <th>ID</th><th>NOM  DE LA TACHE</th><th>STATUT DE LA
TACHE</th>
                        </tr>
                        <c:forEach items="${model1.tache}" var="t">
                              <tr>
                                    <td>${t.id} </td>
                                    <td>${t.nom} </td>
                                    <td>${t.statut} </td>
                                    <!--<td><a
href="Modifier.do?id=${m.id}">Modifier</a> </td>
                                    <td><a onclick="return confirm('Etes vous
sûre?')" href="Supprimer.do?id=${m.id}">Supprimer</a> </td>  -->
                              </tr>
                        </c:forEach>
                  </table>
            </div>
      </div>
</div>
</body>
</html>
```

Chercher tache par statut

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Membre</title>
<link rel="stylesheet" type="text/css"href="css/bootstrap.min.css">
</head>
<body>
<%@include file="header.jsp" %>
<div  class="container col-md-8 col-md-offset-2">
      <div class="panel panel-primary">
            <div class=" panel panel-heading">RECHERCHE TACHE EN FONCTION DU
STATUT</div>
            <div class="panel panel-body">
                  <form action="TacheStatut.do" method="get">
                        <label class="control-label">Membre</label>
                              <select class="form-control" name="statut" >
                              <option value=" nouveau"> nouveau</option>
                              <option value=" en-progres"> en-
progrès</option>
                              <option value=" termine"> terminé</option>
                              </select>
                        <button type="submit" class="btn btn-primary">
Chercher</button>
                  </form>
                  <table class="table table-striped">
                        <tr>
                              <th>ID</th><th>NOM TACHE </th><th>MEMBRE</th>
                        </tr>
                        <c:forEach items="${model2.taches}" var="t">
                              <tr>
                                    <td>${t.id} </td>
                                    <td>${t.nom} </td>
                                    <td>${t.membre}</td>
                                    <!--<td><a
href="Modifier.do?id=${m.id}">Modifier</a> </td>
                                    <td><a onclick="return confirm('Etes vous
sûre?')" href="Supprimer.do?id=${m.id}">Supprimer</a> </td>  -->
                              </tr>
                        </c:forEach>
                  </table>

            </div>
      </div>
</div>
</body>
</html>
```

5- TEST UNITAIRE

```
package TestUnitaire;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.List;
import org.junit.Assert;
import org.junit.Test;
import dao.TacheDaoImplementation;
import dao.MembreDaoImplementation;
import metier.entities.Membre;
import metier.entities.Tache;
public class TestJunit4 {
	@Test
	public void testajoutertache()
	{
		TacheDaoImplementation dao=new TacheDaoImplementation();
		Tache t8 =dao.save(new Tache("PROJET", "Developper une application de
gestion d'emploi de temps ", "en cours",3));
		System.out.println(t8.toString());
	}
	@Test
	public void TestAfficherTacheParMotclé(){
		TacheDaoImplementation dao=new TacheDaoImplementation();
		System.out.println("chercher des taches");
		List<Tache> taches=dao.tacheparMC("%D%");
		for(Tache t:taches){
			System.out.println(t.toString());
		}
	}

	@Test
	public void TestAjouterMembre(){
		MembreDaoImplementation dao=new MembreDaoImplementation();
		Membre m12=dao.save(new Membre("aboubacar sani"));
		System.out.println(m12.toString());
		System.out.println("chercher des membres");
		List<Membre> mem=dao.tousmembresparMC("%A%");
		for(Membre m:mem){
			System.out.println(m.toString());

		}
	}
}
```