

# RAPPORT TP1 GENIE LOGICIEL

## Réalisation d'un gestionnaire de tâches

Etudiant :

KAFADNDO Rodrique

Professeur:

- HO Tuong Vinh

*Promotion 21*

## Introduction

Ce projet est réalisé dans le cadre du cours Génie Logiciel. Le but est de revoir les techniques de conception et de modélisation de problèmes avec le langage UML (Unified Modeling Language) et de leur implémentation en utilisant Java comme langage sous l'environnement de développement intégré libre qui est ECLIPSE.

De nos jours, il est plus l'informatisation de systèmes de communication sont fortement recommandée au sein des institutions et organisations. Un système informatiser permet de gagner non seulement en temps, mais permet d'économiser les ressources afin de pouvoir les utiliser de manière efficace et efficiente. C'est ainsi qu'il nous a été proposé de développer une application qui permettra de bien gérer les membres d'une équipe avec leur tâches respectives.

## I. Les spécifications fonctionnelles

Conformément au cahier de charge, les besoins fonctionnels peuvent être décrits comme suit :

- Créer, modifier, supprimer, ajouter une tâche
- Créer, modifier, supprimer, ajouter un membre
- 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 status (avec le nom du assigné).

### I. 1) Les exigences fonctionnelles et non fonctionnelles de l'application

#### I.1.1) Les exigences fonctionnelles (EF)

Les exigences fonctionnelles décrites ci-dessous constituent les principes d'interaction entre le système et l'utilisateur afin de pouvoir exécuter les spécifications fonctionnelles présentées ci-dessus.

|            |  |
|------------|--|
| <b>EF1</b> | <b>Créer un membre</b>   |
|            | Le système doit demander de saisir les informations sur la personne                      |
|            | Le système doit enregistrer les informations saisies                                     |
|            | Le système doit attribuer un ID à chaque membre  |
| <b>EF2</b> | <b>Modifier les informations d'un membre</b>   |
|            | Le système doit demander à l'utilisateur de saisir l'ID de la personne à modifier        |
|            | Le système doit renvoyer les informations sur l'ID fournies                              |
|            | Le système doit permettre à l'utilisateur de faire le choix sur ce qu'il désire modifier |
|            | Le système doit enregistrer les nouvelles informations et faire la mise à jour           |
| <b>EF3</b> | <b>Supprimer un membre</b>   |
|            | Le système doit demander à l'utilisateur de saisir l'ID de la personne à supprimer       |
|            | Le système doit renvoyer les informations sur le membre concerné                         |
|            | Le système doit permettre à l'utilisateur de supprimer le membre ou d'ignorer            |

|             |   |
|-------------|---|
| <b>EF4</b>  | <b>Ajouter un membre</b>  |
|             | Le système doit demander de saisir les informations sur la personne                             |
|             | Le système doit enregistrer les informations saisies  |
|             | Le système doit attribuer un ID à chaque membre ajouté  |
| <b>EF5</b>  | <b>Créer une tâche</b>  |
|             | Le système doit demander de saisir les informations sur la tâche                                |
|             | Le système doit enregistrer les informations saisies  |
|             | Le système doit attribuer un ID à chaque tâche  |
| <b>EF6</b>  | <b>Modifier une tâche</b>   |
|             | Le système doit demander à l'utilisateur de saisir l'ID de la tâche à modifier                  |
|             | Le système doit renvoyer les informations sur l'ID fournies                                     |
|             | Le système doit permettre à l'utilisateur de faire le choix sur ce qu'il désire modifier        |
|             | Le système doit enregistrer les nouvelles informations et faire la mise à jour                  |
| <b>EF7</b>  | <b>Supprimer une tâche</b>  |
|             | Le système doit demander à l'utilisateur de saisir l'ID de la tâche à supprimer                 |
|             | Le système doit renvoyer les informations sur la tâche concerné                                 |
|             | Le système doit permettre à l'utilisateur de supprimer la tâche ou d'ignorer                    |
| <b>EF8</b>  | <b>Ajouter une tâche</b>  |
|             | Le système doit demander de saisir les informations sur la tâche                                |
|             | Le système doit enregistrer les informations saisies  |
|             | Le système doit attribuer un ID à chaque tâche ajouté   |
| <b>EF9</b>  | <b>Assigner une tache à un membre</b>   |
|             | Le système doit demander à l'utilisateur de saisir l'ID du membre                               |
|             | Le système doit demander à l'utilisateur de saisir l'ID de la tâche à assigner                  |
|             | Le système doit faire la liaison entre l'ID du membre et l'ID de la tâche                       |
|             | Le système doit enregistrer les opérations effectuées   |
| <b>EF10</b> | <b>Chercher et afficher tous les tâches assignées à un membre (par son ID)</b>                  |
|             | Le système doit demander à l'utilisateur de saisir l'ID du membre                               |
|             | Le système doit renvoyer les informations sur le membre concerné                                |
|             | Le système doit renvoyer les informations sur les tâches qui lui ont été assignées              |
| <b>EF11</b> | <b>Chercher et afficher tous les tâches en fonction de leur statut (avec le nom du assigné)</b> |
|             | Le système doit demander à l'utilisateur de saisir l'ID de la tâche qu'il veut afficher         |
|             | Le système doit renvoyer les informations sur le membre concerné                                |
|             | Le système doit renvoyer les informations sur le statut de chacune des tâches                   |

Tableau 1: Liste des exigences fonctionnelles

### 1.1.2) Les exigences non-fonctionnelles (ENF) de l'application

Les exigences non-fonctionnelles décrites ci-dessous nous permettent d'évaluer les performances de notre système sur tous les exigences fonctionnelles précédemment définies.

|       | Description   | EF concernée         |
|-------|---|----------------------|
| ENF 1 | Le taux pour qu'un échec survienne lors d'une opération sur le système doit être très faible soit 0.0001%                                 | Toutes les exigences |
| ENF 2 | Le système doit être disponible durant toutes les opérations  | Toutes les exigences |
| ENF 3 | En cas de défaillance, le système ne devra pas causer de dommages matériels   | Toutes les exigences |
| ENF 4 | En cas d'erreurs venant de l'utilisateur, les données ne doivent subir aucun changement, la disponibilité des données doit être garantie. | Toutes les exigences |
| ENF 5 | Le système doit avoir au plus 5 secondes comme temps de réponse à une opération.  | Toutes les exigences |

Tableau 2: Les exigences non-fonctionnelles

### 1.1.3) Diagramme de cas d'utilisation

Le diagramme d'utilisation présenté ci-dessous nous montre les cas d'interactions entre l'utilisateur et le système.

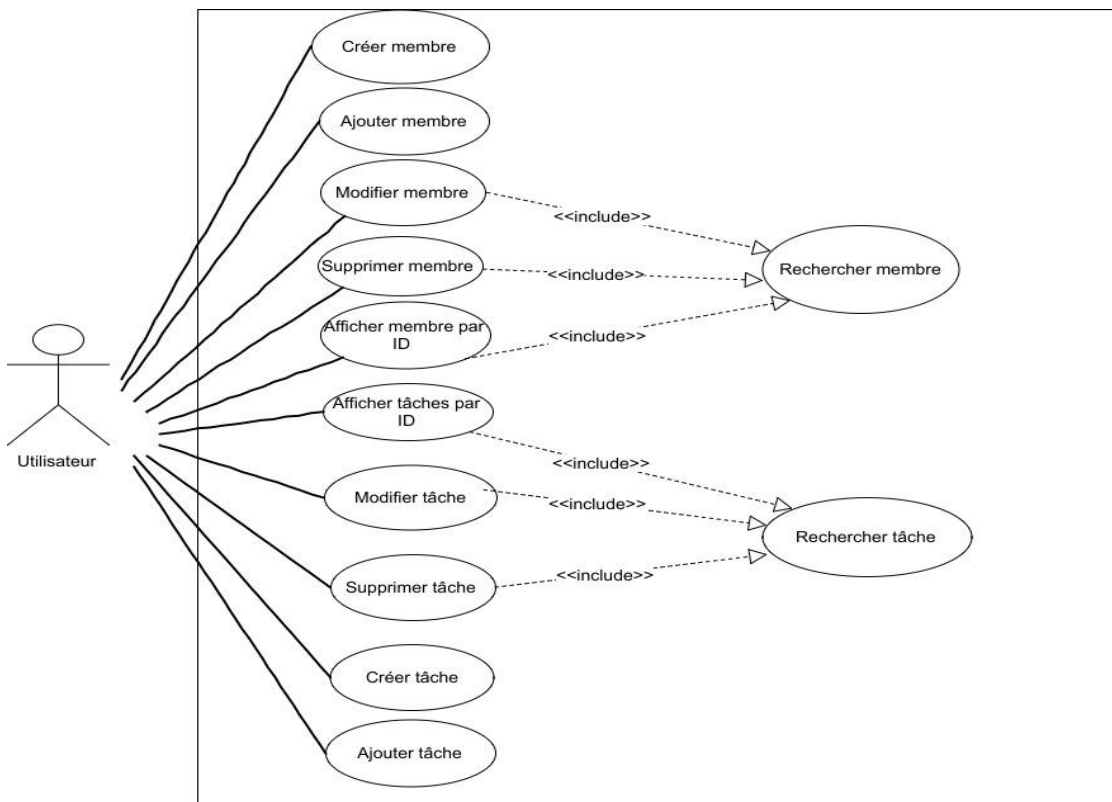


Figure 1: Diagramme de cas d'utilisation

## II. Conception

### II.1) Diagramme de classe

Le diagramme de classes de notre application est principalement composé de deux classes à savoir la classe « Taches » et la classe « Membre ».

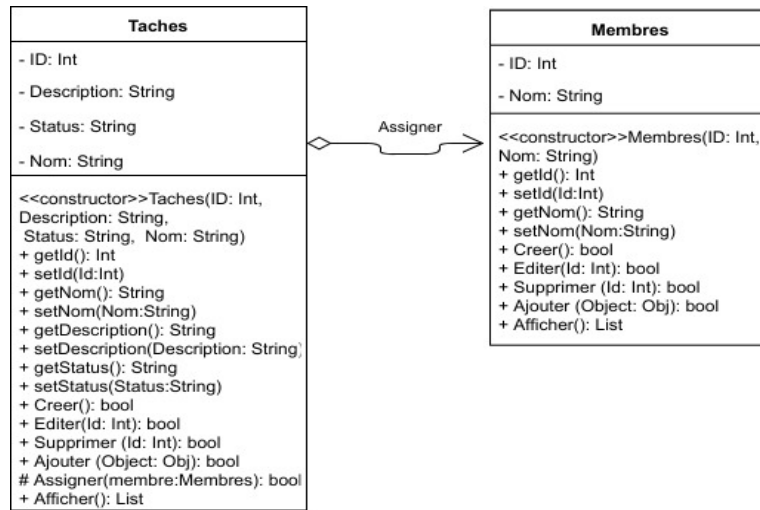


Figure 2: Diagramme de classe

### II.2) Diagramme de séquence

#### II.2.1) Suppression d'un membre

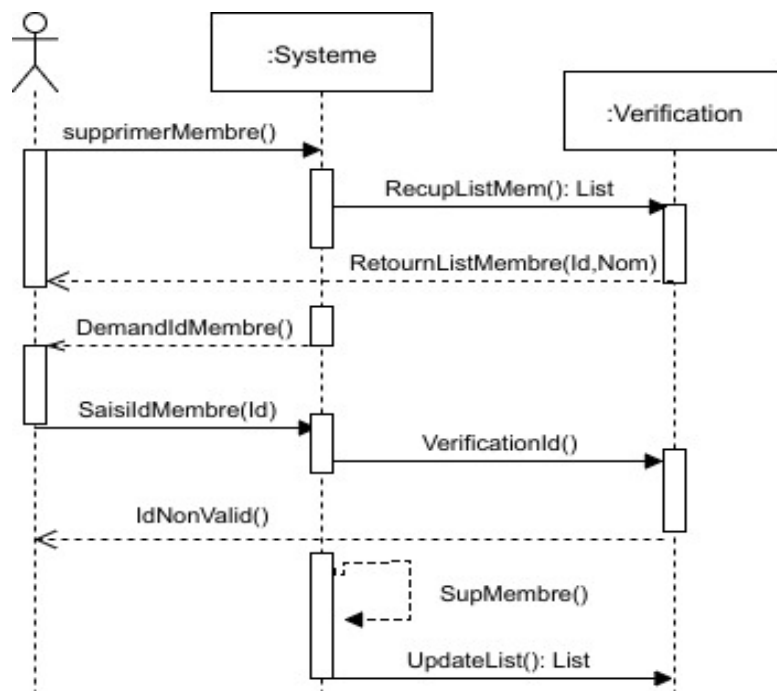


Figure 3: Diagramme de séquence (Suppression d'un membre)

## II.2.2) Assignment d'une tâche à un membre

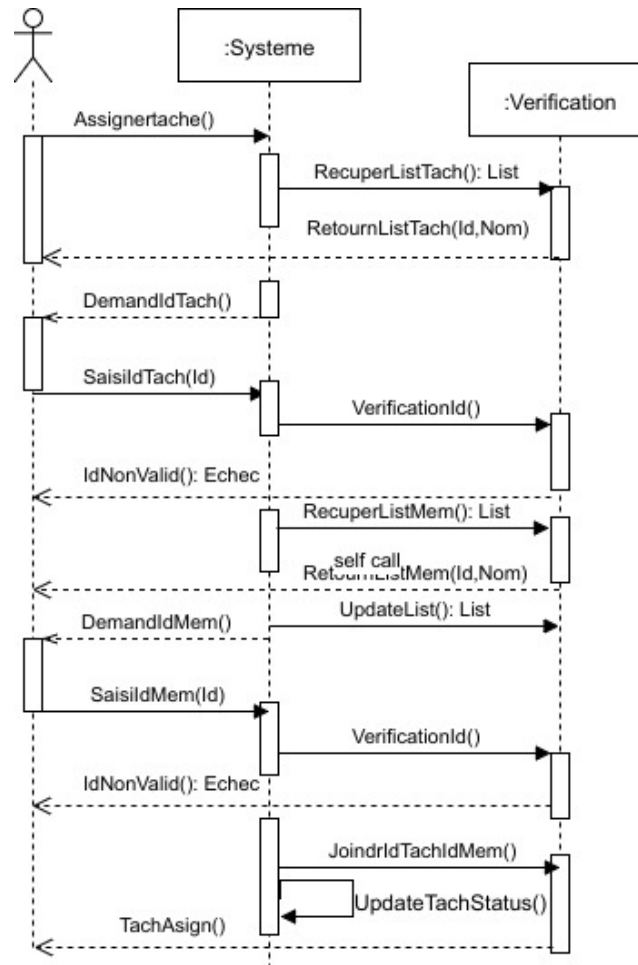


Figure 4: Diagramme de séquence (Assignment d'une tâche à un membre)

## III. Implémentation et Test

### III.1) Solution proposée

#### III.1.1) Outils de développement

L'implémentation de notre application s'est effectuée sur la plateforme d'Eclipse IDE. Il s'agit d'un environnement libre, extensible et polyvalent permettant de créer des projets de développement.

#### III.1.2) Le langage de programmation

Notre programme est implémenté en utilisant le langage Java. C'est un langage très performant car il est orienté objet. Avec le langage Java, nous pouvons développer des applications bien structurées et modulables rendant ainsi une grande flexibilité pour son usage.

#### III.1.3) Les tests d'acceptations

Afin de vérifier la conformité de notre programme par rapport aux exigences demandées, nous allons procéder à une série de tests sur quelques fonctionnalités.

### III.2) Expérimentation

#### III.2.1) Vue globale du programme

Après l'exécution du programme, nous avons un menu principal présenté comme suit:

```

*****
MENU PRINCIPAL
*****
1. Créer, editer, supprimer, ajouter une Tâche
2. Créer, editer, supprimer, ajouter un memebre
3. Assigner une tâches à un membre
4. Retrouver toutes les tâches assignées à un membre par son ID
5. Retrouver toutes les tâches selon leur ID avec le membre assigné
6. Quitter le programme

Entrer votre choix:

```

Figure 5: Menu principal de l'application

Dans ce menu nous pouvons voir les différentes options qui illustrent les exigences fonctionnelles de notre application. Numéroté de 1 à 6, l'utilisateur doit choisir un des chiffre et suivre les instructions pas à pas jusqu'à finir l'opération qu'il désire réaliser.

### III.2.2) Création d'une tâche

Pour créer une tâche, nous choisissons le menu «1» et nous obtenons un autre menu consacré à tâche dans lequel différent opération peuvent être effectué. Ainsi nous donnons un nom et une description cette nouvelle tâche. La capture ci-dessous nous montre un exemple.

```

1. Créer, editer, supprimer, ajouter une Tâche
2. Créer, editer, supprimer, ajouter un memebre
3. Assigner une tâches à un membre
4. Retrouver toutes les tâches assignées à un membre par son ID
5. Retrouver toutes les tâches selon leur ID avec le membre assigné
6. Quitter le programme

Entrer votre choix: 1

*****
MENU CHOIX DE TACHES
*****
1. CREER
2. EDITER
3. SUPPRIMER
4. AJOUTER
5. SUPPRIMER REQUETTE

Entrer votre choix: 1

Creer une tache:
Entrer le nom de la tache: Tâche A
Entrer la description de tache : Ceci est la description de la tâche A
|CREER SUCCES

Voulez-vous creer autres taches. <0/N>:

```

Figure 6: Cas de création d'une tâche

La création d'un membre se déroule de la même façon en respectant les entrées du menu.

### III.2.1) Assignment d'une tâche à un membre

Le bouton **3** du menu principal permet d voir la liste des tâches créer et de les assigner à un membre existant. En procédant ainsi, nous obtenons les résultats suivants:

Taper ENTRE pour voir la liste des taches:

\*\*\*\*\*

ID: 1

Name: Planification

Description: Donner un planning pour la progrssion du projet

Status: DISPONIBLE

\*\*\*\*\*

ID: 2

Name: Transformation

Description: transformation des besoins fonctionnels

Status: DISPONIBLE

|

\*\*\*\*\*

ID: 3

Name: Tâche A

Description: Ceci est la description de la tâche A

Status: DISPONIBLE

Taper ENTRE si vous désirez annuler

Saisir l'ID de la tache dont vous désirez assigner à un membre

Entrer votre choix: 1

Figure 7: Cas de choix d'une tâche pour assignation

Les tâches ainsi listés viennent d'être créées, elles possèdent toutes les statu **DISPONIBLE**, ce qui veut dire qu'elles peuvent être assignées. Nous choisissons la tâche à l'ID=1 pour assigner à un membre. Après avoir choisi la tâche à assigner, le système nous présente la liste des membres. Chaque membre est présenté par son ID et son nom. Pour assigner, le système nous demande de choisir l'ID du membre auquel nous voulons assigner la tâche choisie.

Après exécution, le système nous indique que l'assignation a été bien effectuée. Nous pouvons par la suite voir la tâche après assignation en tapant juste sur ENTRE. Le résultat après assignation nous montre que le statut de la tâche assignée est passé de **DISPONIBLE** à **EN\_COURS** pour montrer que la tâche est déjà assignée et est en cours d'exécution.



```

Taper ENTRE pour voir la liste des membres:

*****
ID: 1
Name: Kafando

*****
ID: 2
Name: Phuong

*****
ID: 3
Name: Kouadio

*****
ID: 4
Name: Quy

Taper ENTRE si vous désirez annuler
Saisir l'ID du membre dont vous désirez assigner cette tache

Entrer votre choix: 2
ASSIGNER SUCCES

```

Figure 8: Cas de choix du membre auquel sera assignée la tâche

La capture ci-dessous montre la mise en correspondance entre la tâche et le membre choisi. Ceci indique que la tâche à l'ID=1 est assignée au membre dont l'ID=2.

```

Taper ENTRE pour voir les taches après assignation:

*****
ID: 1
Name: Planification
Description: Donner un planning pour la progrssion du projet
Status: EN_COURS
Member:
*****
ID: 2
Name: Phuong

```

Figure 9: Phase finale de l'assignation

### III.3) Présentation de quelques tests unitaires

Un test unitaire une opération qui consiste à soumettre les portions du code un ensemble de tests qui permettront de s'assurer que le code fonctionne en entièreté comme en ses différentes parties qui le compose. Il s'agit essentiellement les des classes et méthodes qui le compose. Afin de pouvoir effectuer les tests unitaires sur notre code, nous avons procédé à la création d'un nouveau fichier JUnit dans lequel nous avons procédé au test de nos méthodes.

Nous avons pu remarquer que lorsque des incohérences se présentent à l'intérieur de ces méthodes, le test passe à l'échec. Il est donc obligatoire de respecter les paramètres d'entrées de chaque méthode pour que le test soit accepté ou positif.

Nous présentons ci-dessous les résultats issus de quelques tests à savoir le test porté sur la méthode **Creer**, **Supprimer**, **AfficherListDesMembres**, et la méthode **Editer** de notre entité **Membres** que nous avons effectué. La capture présente le code de test JUnit de la classe « supprimer » et « AfficherListDesMembres ».

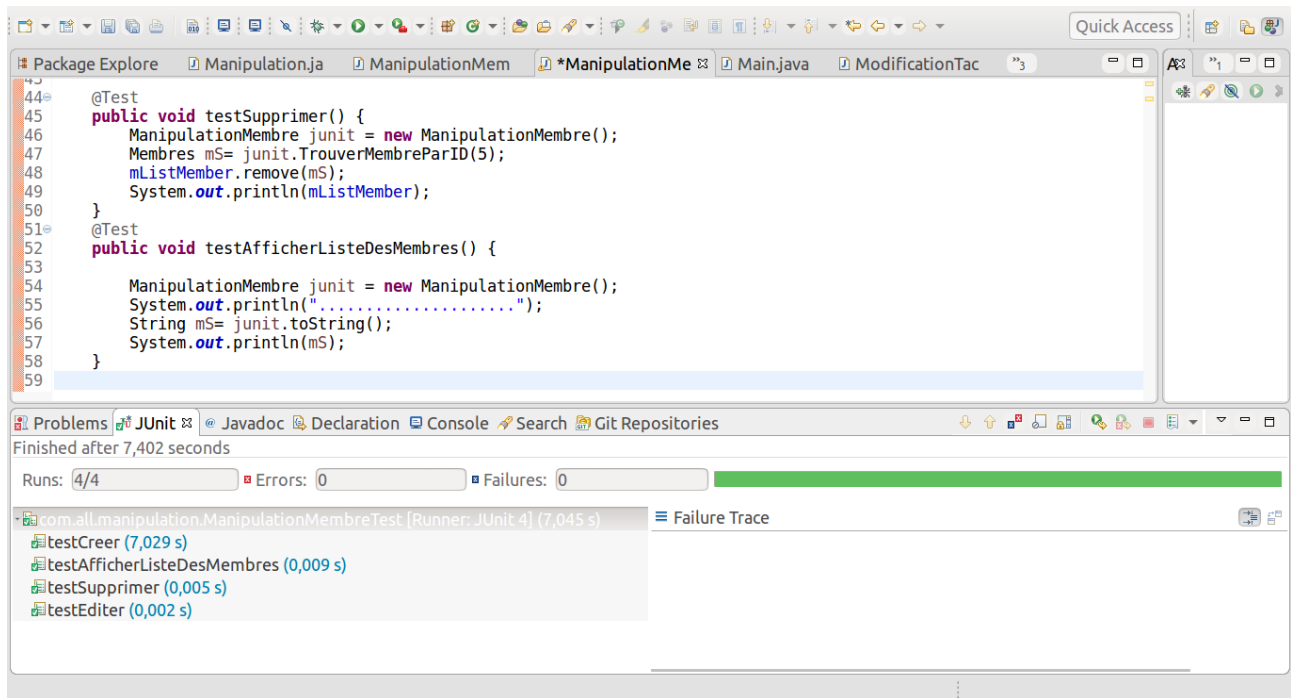


Figure 10: Exemple de test Unitaire

## Conclusion

Ce travail a consisté à mettre en œuvre une application de gestion de tâches. La modélisation a été faite par le langage UML et l'implémentation en Java sous l'IDE Eclipse. Pour tester le bon fonctionnement de notre programme, nous avons procédé à l'utilisation de JUnit.

A ce stade de notre travail, nous avons pu réaliser notre application de gestion de tâches. Toutes les exigences fonctionnelles ont été bien testées et donne de résultats satisfaisants. Cependant, notre application peut toujours subir des améliorations pour plus d'optimisation et d'efficacité.