Institut Francophone International

Cours : Génie Logiciel Avancé

TP1 : Un petit Gestionnaire de Tache en JAVA

Rapport préparé par :

ELIODOR Ednalson Guy Mirlin - P21

Août 2017

1. Introduction

Dans le cadre du cours de génie logiciel, il nous a été demander d'implémenter en langage JAVA un petit gestionnaire de taches. Ce travail nous a permis de faire un rappel sur les concepts de modélisation avec UML et Programmation Oriente Objet avec JAVA ou chaque objet dispose d'un ensemble d'attributs décrivant son état et l'état du système est décrit (de façon décentralisée) par l'état de l'ensemble. Ceci nous a permis aussi de comprendre l'importance de l'utilisation de GitHub pour la gestion de code source dans le cadre d'un projet informatique.

1.1 – Spécification de l'application

Le projet est donc de réaliser une application : « Gestionnaire de Tache » pour une équipe de travail. Ce qu' il fait retenir dans ce cas c'est qu'une tache est composée d'un identifiant (id), un nom, une description et de son status qui peut être dans ce cas soit Nouveau, En-Progrès ou Terminé. Et, un membre est donc composé d'un identifiant(id) et de son nom. Ainsi, le programme implémenté fournira a l'utilisateur les fonctionnalités suivantes :

- 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)

2. Exigences Fonctionnelles et Non Fonctionnelles

2.1- Exigences Fonctionnelles :

Les besoins fonctionnels répondent aux points précis du cahier des charges, et sont donc requis par le client. Dans notre cas, ces besoins fonctionnels doivent répondent aux critères d'acceptation de notre cher professeur qui a dans ce cas émis le sujet de TP. Il est a préciser que de manière générale que les exigences fonctionnelles ne sont pas négociables, c'est a dire que c'est le "besoin primaire" du client.

- Le système doit considérer qu'un objet de type tache doit avoir un identifiant (id), un nom, une description et un status qui peut être dans ce cas soit Nouveau, En-Progrès ou Terminé.
- Le système doit considérer qu'un objet de type membre doit avoir un identifiant(id) et un nom.
- Lors de la création ou l'ajout d'une tache le système doit demander et prendre en compte les informations nécessaires (id, nom, description, status) et les enregistre dans la base de données.
- Lors de la création ou l'ajout d'un membre le système doit demander et prendre en compte les informations relatives au membre(id, nom) et les enregistre dans la base de données.
- Le système doit permettre la modification et la sauvegarde d'une entrée (tache, membre)dans la base.
- Le système doit permettre la suppression d'une entrée (tache, membre).
- Le système doit permettre a l'utilisateur d'assigner une ou plusieurs tache(s) a un membre.

2.2- Diagramme de cas d’ utilisation :

Le **diagramme de cas d'utilisation** constitue la première étape UML d’analyse d’un système. Il permet donc de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités d'un système .

Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, les cas d'utilisation, ayant un sens pour les acteurs. Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système, ils sont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique.

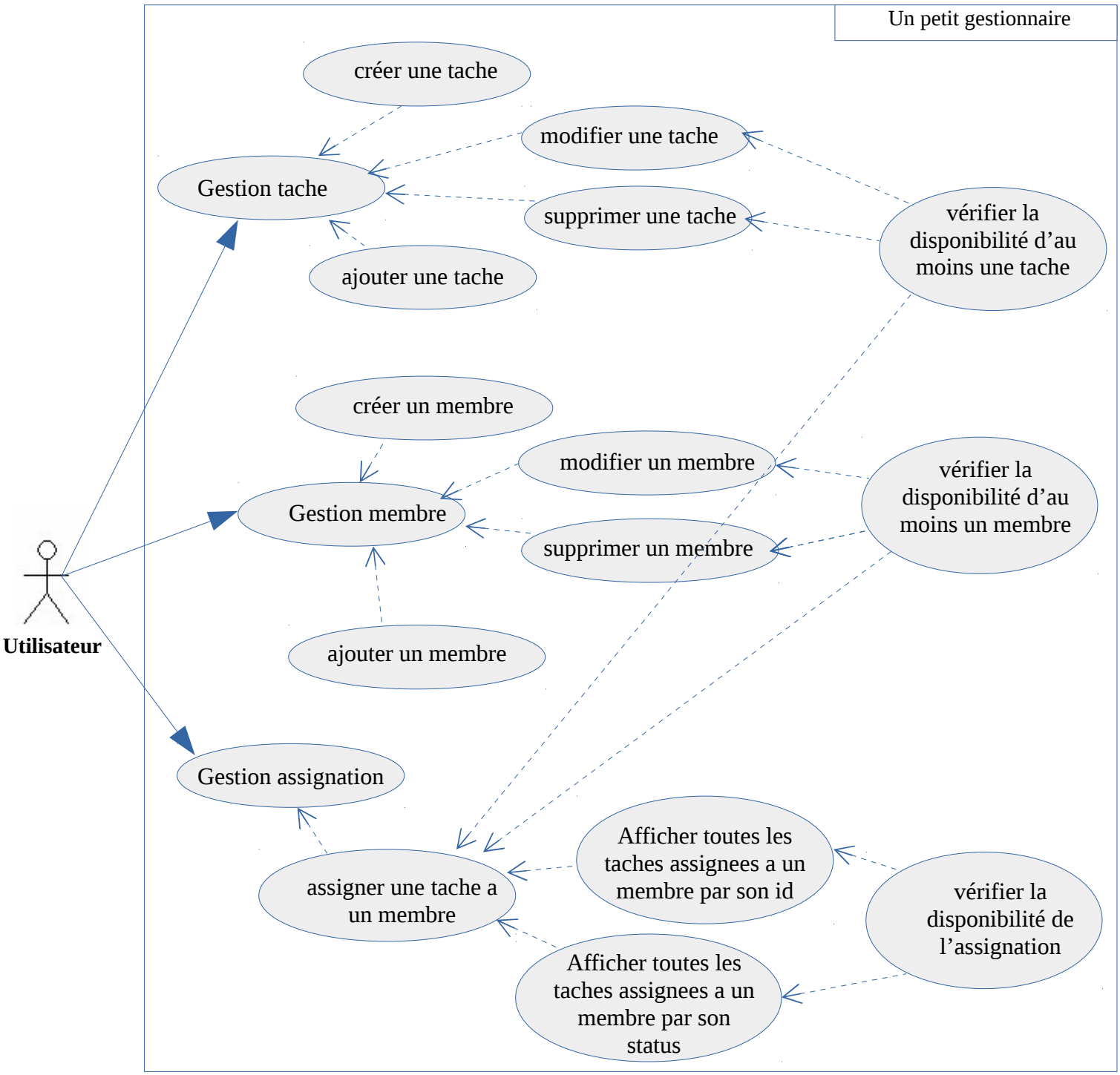


Figure : Diagramme de cas d’utilisation détaillé

Les besoins non-fonctionnels sont soit des besoins optionnels, soit des besoins/contraintes liés à l'implémentation. Ils peuvent être fixés par le client (fonctions optionnelles), ou par le développeur (contraintes d'implémentation).

Performance : Le temps de réponse du système suite a une requête du cote de l'utilisateur doit être essentiellement court (maximum 2 secondes).

Pertinence et exactitude : La réponse de la requête de l'utilisateur doit exactement celle l'opération demandée.

Compréhension : L'application doit être facilement compréhensible dans le cas de l'utilisation de ses fonctions.

Disponibilité : Les informations pré-enregistrées ne doivent subir aucun changement en cas d'erreur du coté utilisateur.

Maintenabilité : le code source de notre application doit être bien structure afin de faciliter les mises a jours.

3. Conception

A ce niveau, nous parlerons de l'analyse en terme de conception du système, c'est-à-dire la mise en place du diagramme de classe et ensuite nous choisirons 2 ou 3 scenarios pour lesquels nous préciserons les diagrammes de séquences.

3.1- Diagramme de classe

Le diagramme de classes est généralement considéré comme le plus important dans un développement orienté objet. Il représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboîtage conceptuel (héritage) ou une relation organique (agrégation).

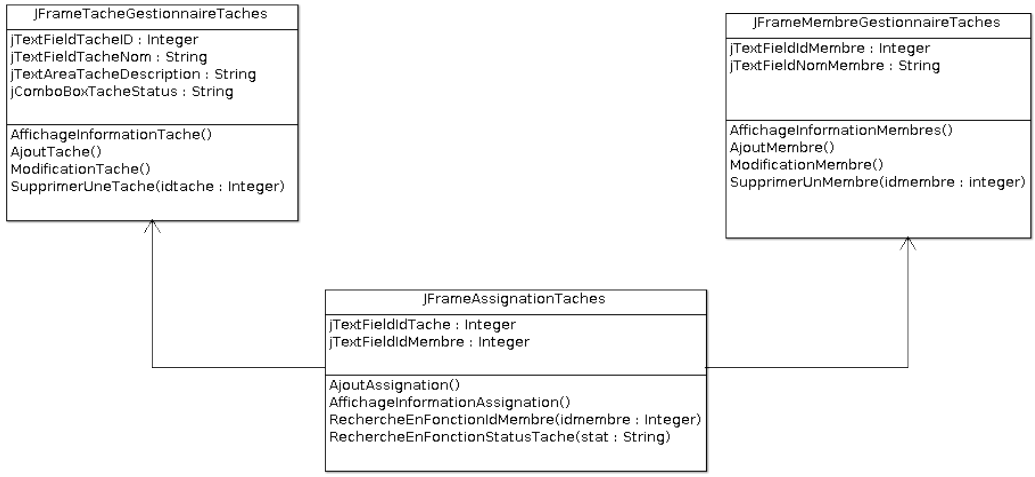


Figure : Diagramme de classe

3.2- Système Backload

Action #	Nom de l´ action	Processus de réalisation de l´ action
1	Créer un membre	Lancer l'application, Cliquer sur le bouton «Les Membres», Entrer les informations nécessaires a un membre (Id, Membre), Cliquer sur le bouton créer.
2	Ajouter un membre | *Ajouter un membre signifie que la base n'est plus vide donc on ajoute un ou des membres sur les informations des membres existantes.*	Lancer l'application, Cliquer sur le bouton «Les Membres», Entrer les informations nécessaires a un membre (Id, Membre), Cliquer sur le bouton « créer ».
3	Modifier un membre	Lancer l'application, Cliquer sur le bouton « Les Membres », Entrer l'Id du membre a modifier, Modifier le champ nom, Cliquer sur le bouton « Modifier ».
4	Supprimer un membre	Lancer l'application, Cliquer sur le bouton « Les Membres », Cliquer sur le bouton « Supprimer », Affichage d'un box demandant l'Id du membre a supprimer, Entrer l'Id du membre a supprimer, Cliquer sur Ok.
5	Créer une tache	Lancer l'application, Cliquer sur le bouton « **Les Taches** », Entrer les informations necessaires a une tache (Id, nom, description, Status), Cliquer sur le bouton « Créer ».
6	Ajouter une tache | *Ajouter une tache signifie que la base n'est plus vide donc on ajoute une ou des taches sur les informations des taches existantes.*	Lancer l'application, Cliquer sur le bouton « **Les Taches** », Entrer les informations nécessaires a une tache (Id, nom, description, Status), Cliquer sur le bouton « Créer ».
7	Modifier une tache	Lancer l'application, Cliquer sur le bouton « Les Taches », Entrer l'Id de la tache a modifier, Modifier les autres champs, Cliquer sur le bouton « Modifier »
8	Supprimer une tache	Lancer l'application, Cliquer sur le bouton « Les Taches », Cliquer sur le bouton « Supprimer », Affichage d'un box demandant l'Id de la tache a supprimer, Entrer l'Id de la tache a supprimer, Cliquer sur Ok.
9	Assigner une tache a un membre	Lancer l'application, Cliquer sur le bouton « Assignation Taches », Entrer l'Id de la tache, Entrer l'Id du membre,

		cliquer sur le bouton « Assignation »
10	Chercher et afficher toutes les taches assignées a un membre (par son id)	Lancer l’application, Cliquer sur le bouton « Assignation Taches », Entrer l’Id du membre, Cliquer sur « Rechercher »
11	Chercher et afficher toutes les taches en fonction de leur status (Avec le nom du assigné)	Lancer l’application, Cliquer sur le bouton « Assignation Taches », Entrer l’une des valeurs de l’attribut Status qui peut être soit (Nouveau, En-Progrès ou Termine), Cliquer sur le bouton « Rechercher »

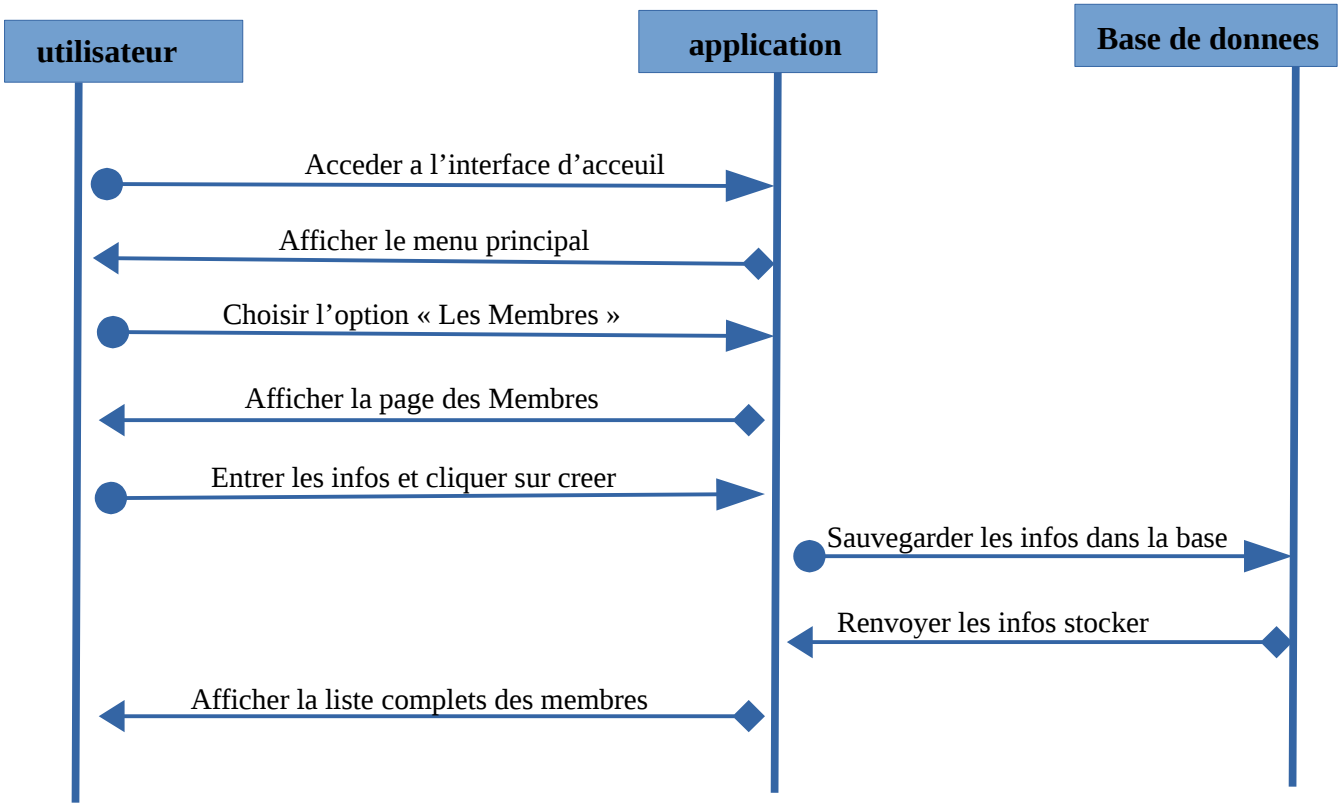
Diagrammes de Séquence

3.3- Diagrammes de séquence

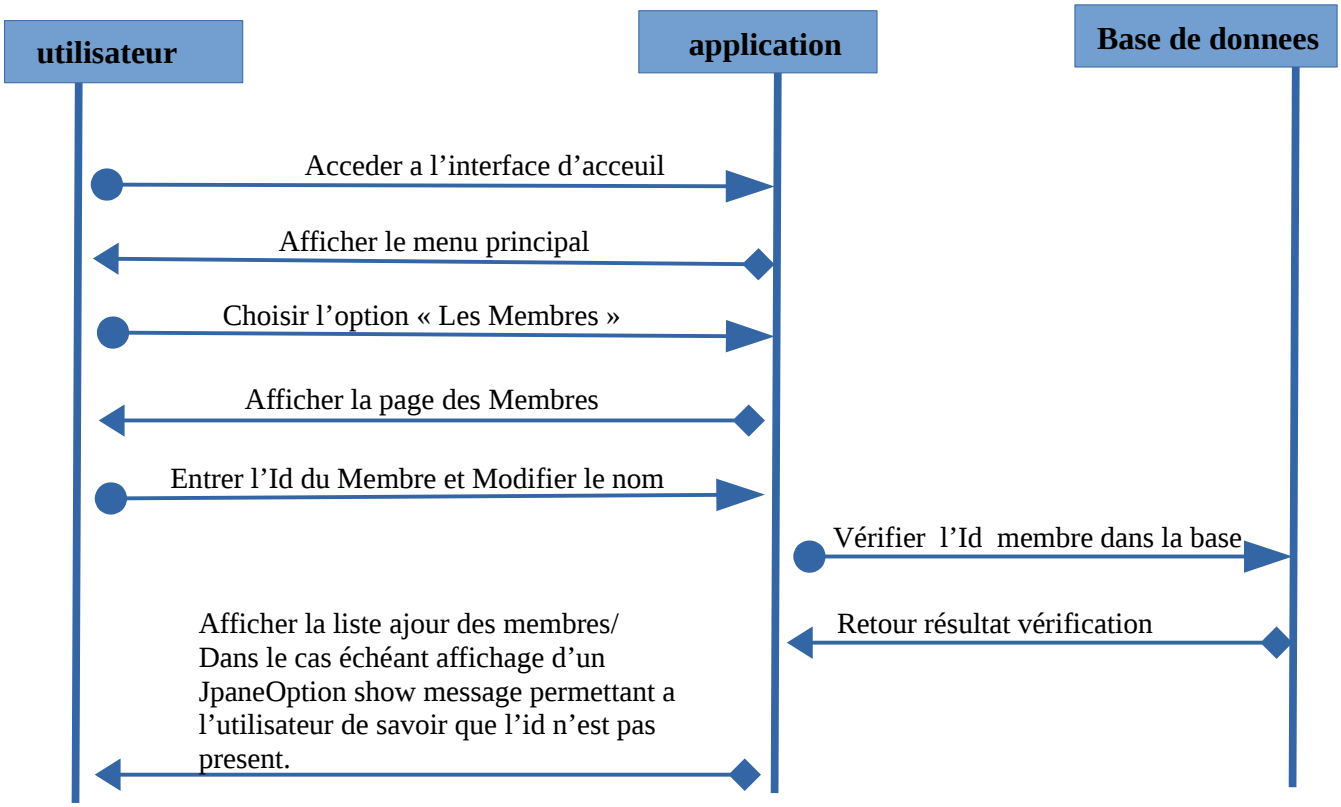
Les diagrammes de séquences permettent de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs :

- Les objets au cœur d'un système interagissent en s'échangent des messages.
- Les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).

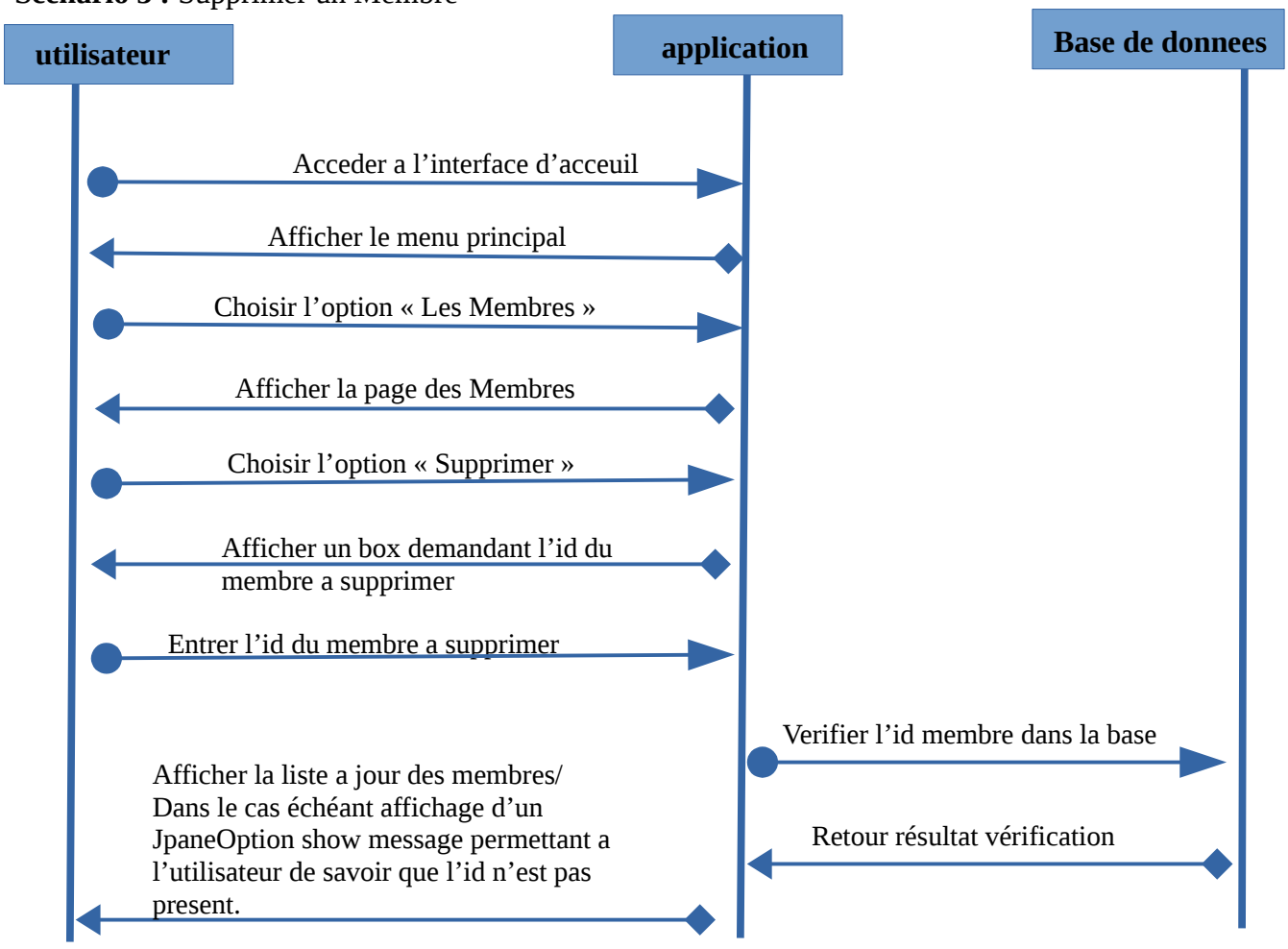
Scenario 1 : Créer un membre



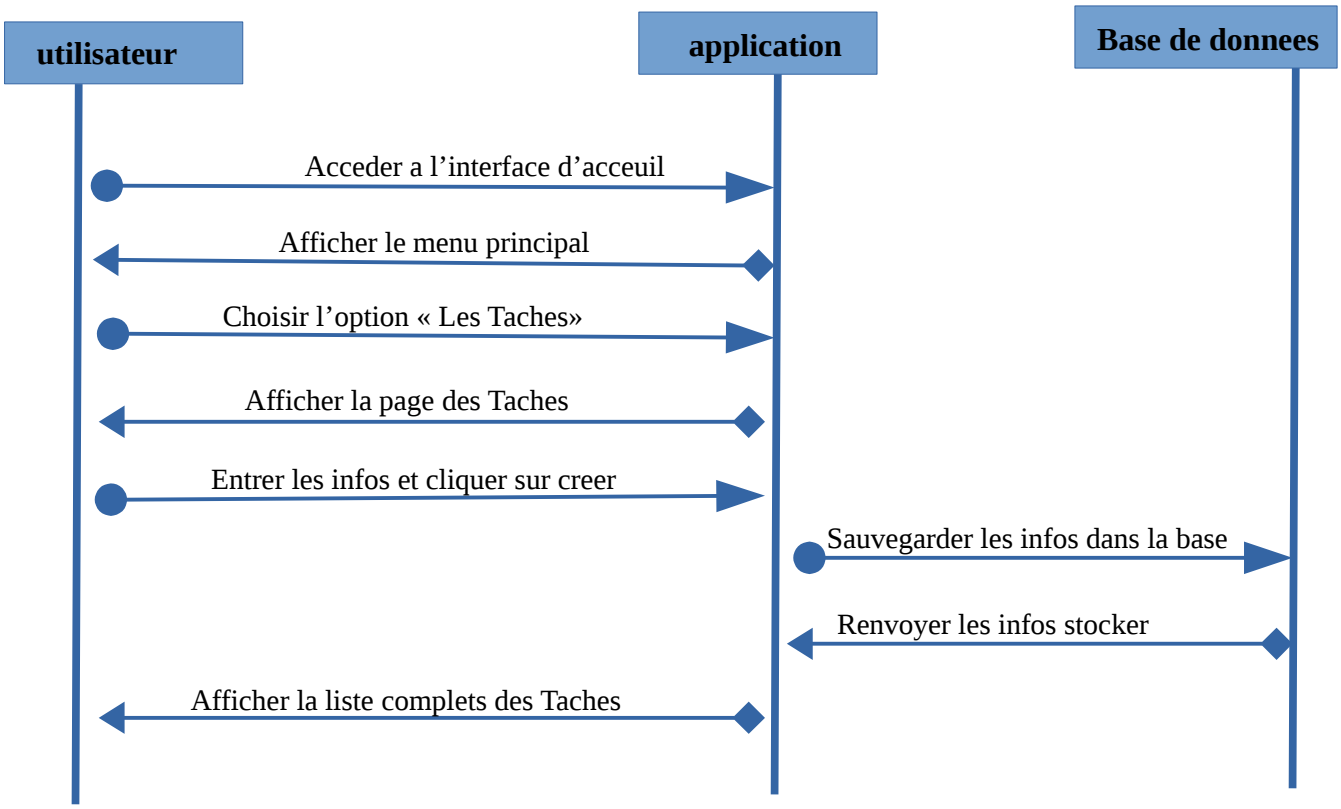
Scenario 2 : Modifier un membre



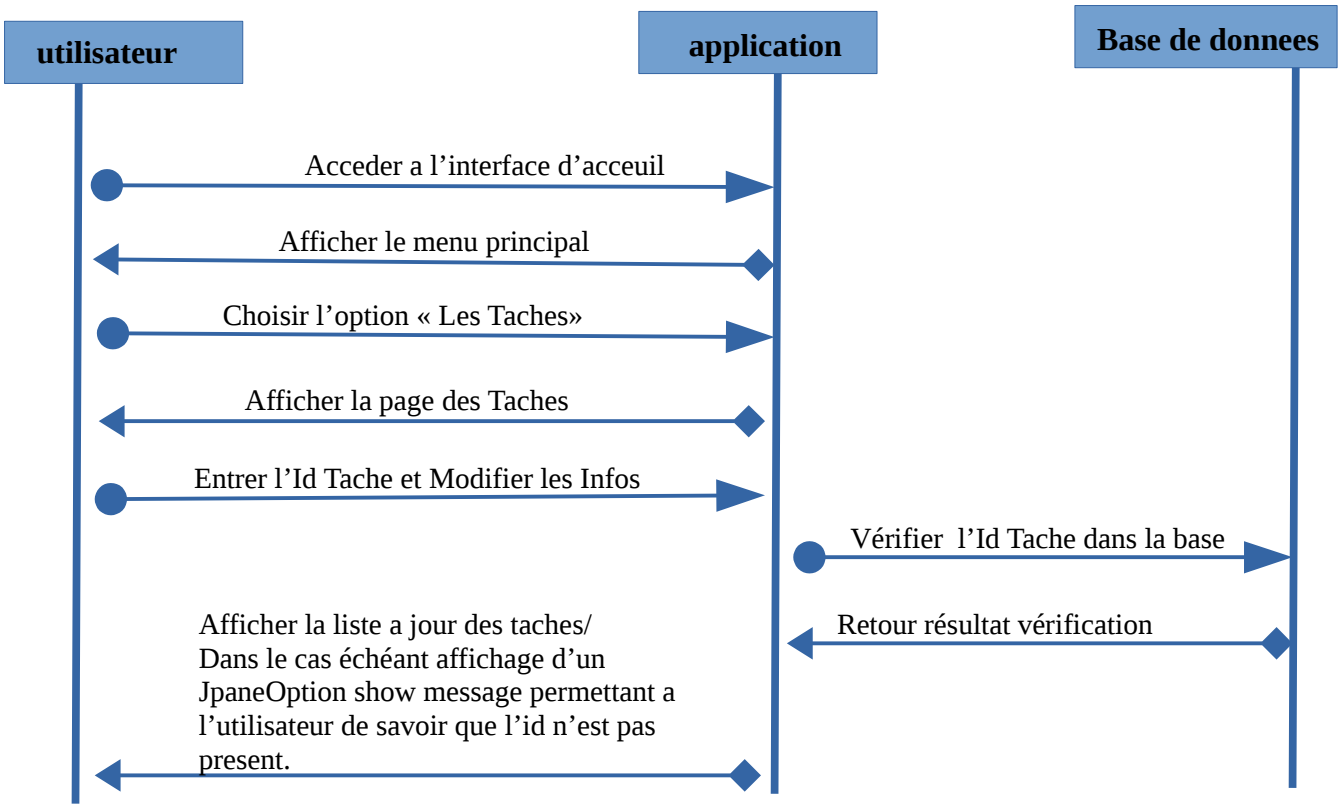
Scenario 3 : Supprimer un Membre



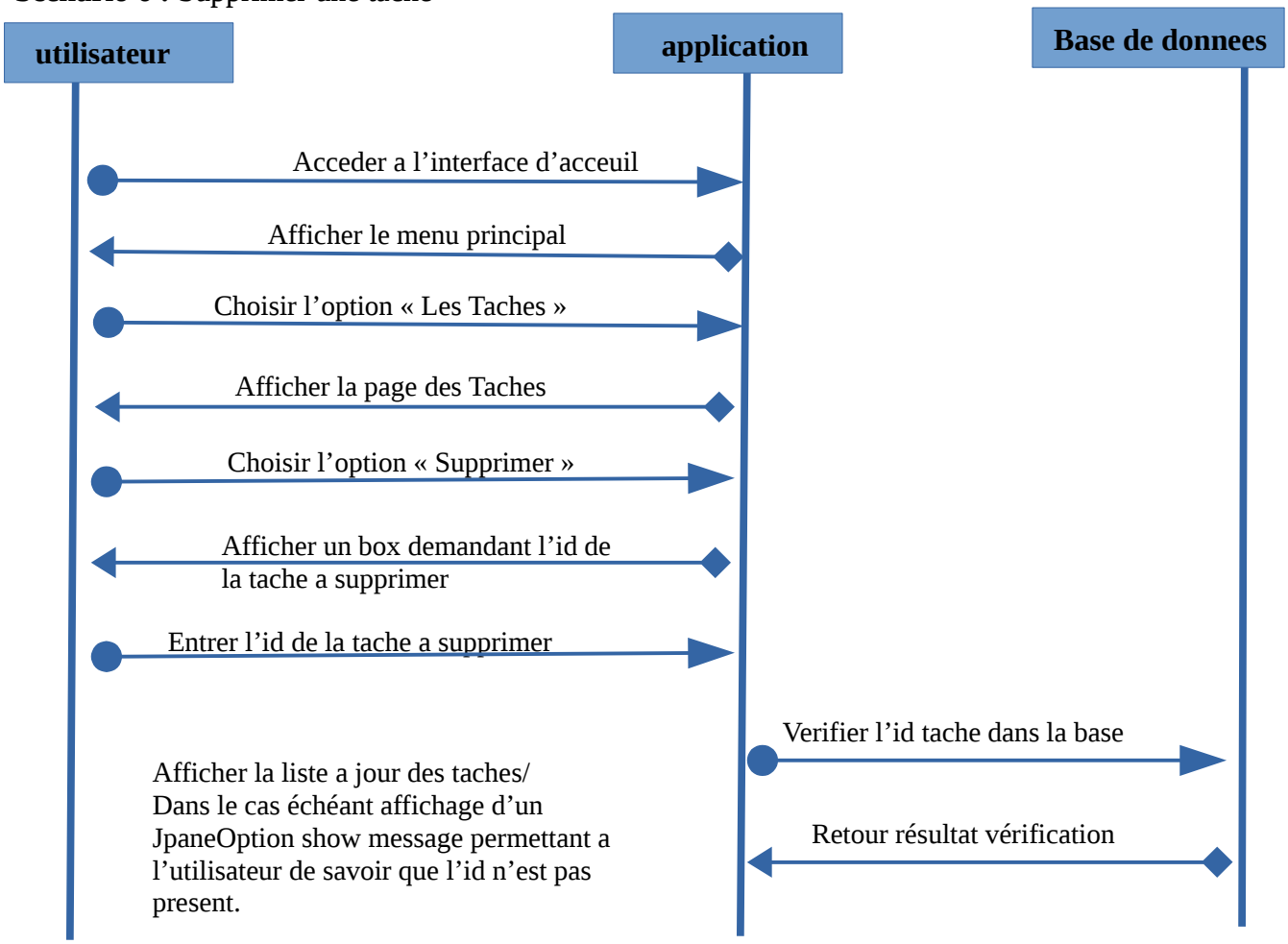
Scenario 4 : Créer une tache



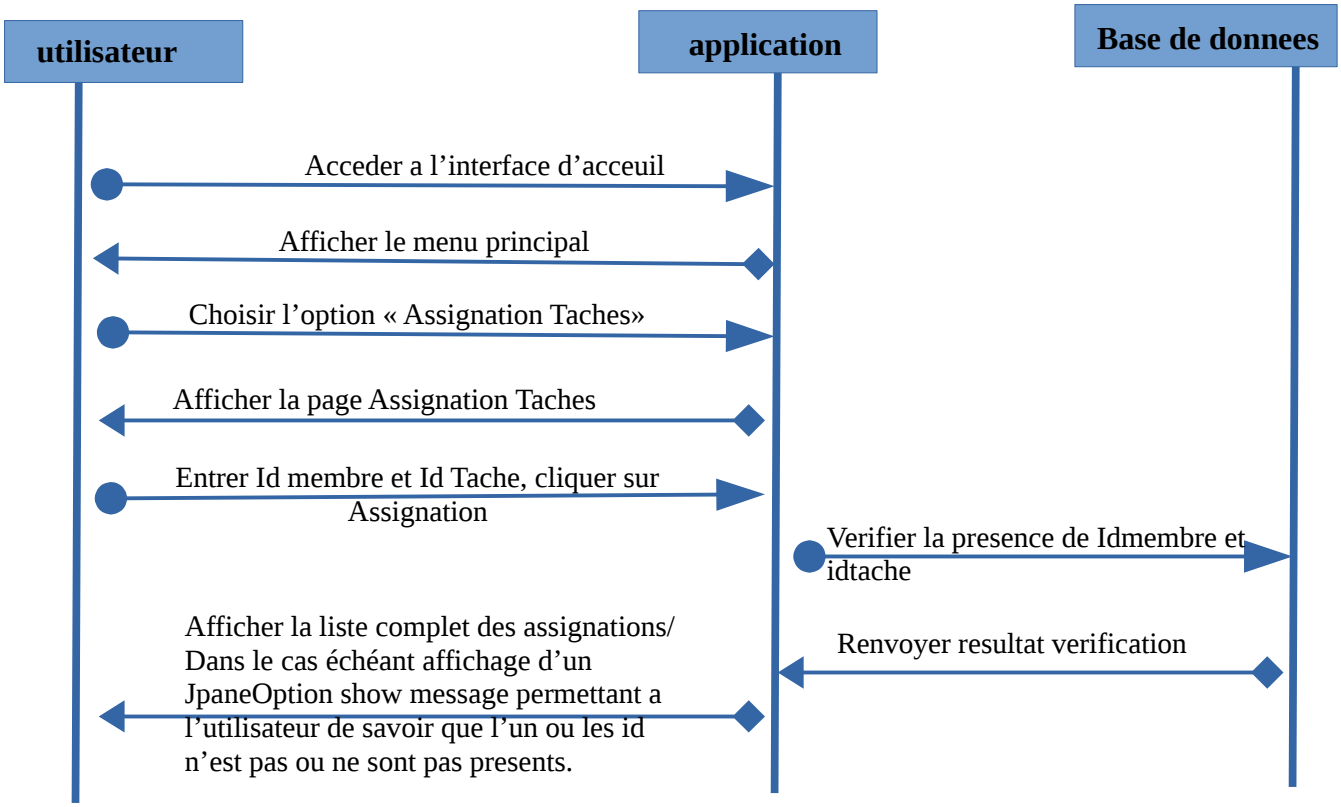
Scenario 5 : Modifier une tache



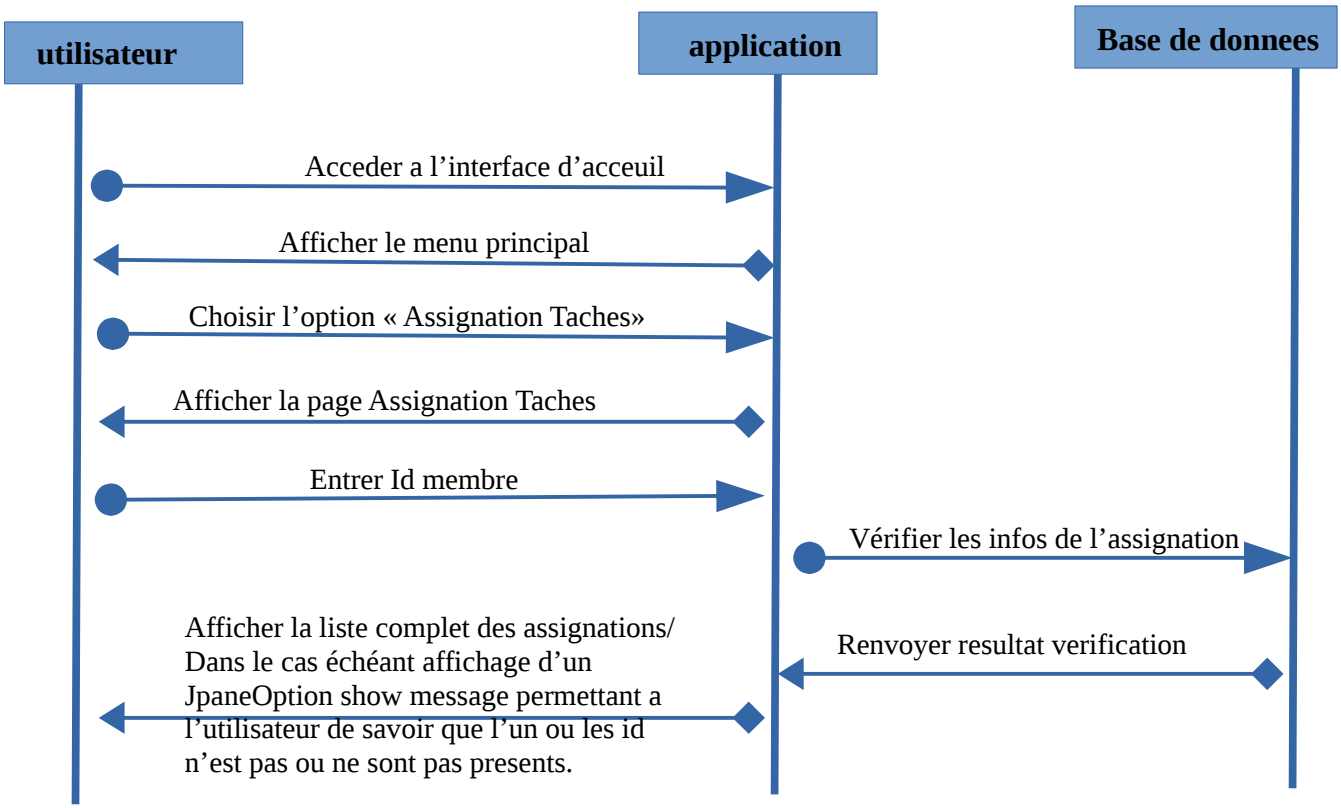
Scenario 6 : Supprimer une tache



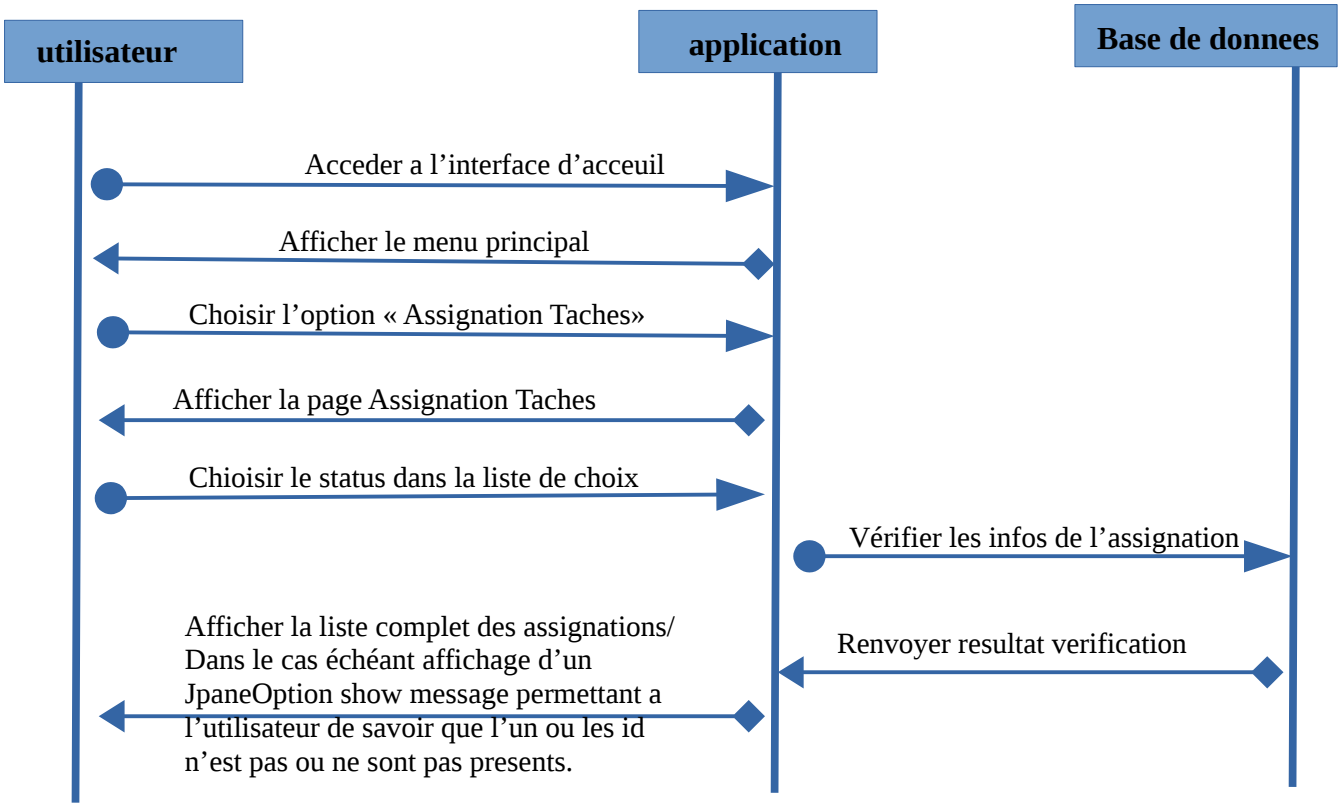
Scenario 7 : Affecter une tache a un membre



Scenario 8 : Chercher et afficher toutes les taches assignées a un membre (par son id)



Scenario 9 : Chercher et afficher toutes les taches en fonction de leur Status (Avec le nom du assigné)



Implémentation
&
Test d’acceptation

4. Implémentation et Test

4.1- Environnement de développement

L' application a été développé en local sur notre ordinateur portable (acer Aspire E15, Intel Celeron Processor N2830 up to 2.41Ghz, 4GB DDR3 Memory) avec utilisation exclusive des plateformes libres (Ubuntu 17.04, Netbeans IDE 8.2). Pour la modélisation, nous avons utilisé UML (Unified Modeling Language).

4.2- Sauvegarde des données

Afin de sauvegarder les données nous avons configurer PHPMyAdmin tout en faisant usage du langage SQL afin de mettre en place une base de données et gérer les procédures a l'interne (ie dans la base de données) afin d'éviter de laisser en clair les requêtes dans la code sources.

4.3- Méthode de test

Nous avons utilisé JUnit, pour faire le test unitaire et s'assurer le bon fonctionnement des méthodes utilisés dans la classe JframeTacheGestionnaireTache.java

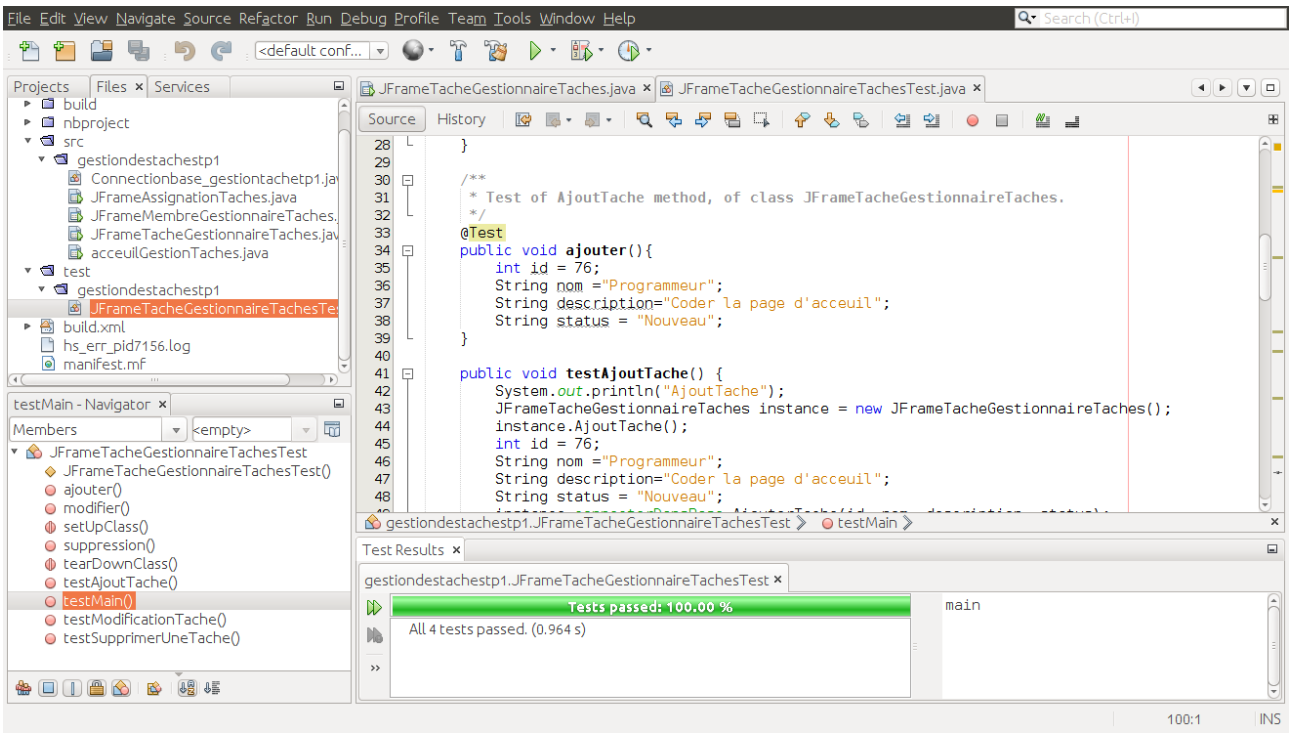


Figure : Test unitaire avec Junit dans Netbeans passé a 100 %

Dans ce test nous avons pris en compte les différentes méthodes de la classe précitée.

4.4- Test d’acceptation

La recette (ou test d'acceptation) est une phase de développement des projets, visant à assurer formellement que le produit est conforme aux spécifications (réponse donnée à un instant t aux attentes formulées). Elle s'inscrit dans les activités plus générales de qualification.

Dans ce cas, nous allons donc choisir certain cas d’utilisation dans le ***système backload*** afin de mener a point le test d’acceptation. Ainsi, nous avons fait choix de l’**action 3 : *Modifier un membre.***

- Lancer l’application
- Cliquer sur le bouton « Les Membres »
- Entrer l’Id du membre a modifier
- Modifier le champ nom
- Cliquer sur le bouton « Modifier »

Figure : Page accueil de l’application

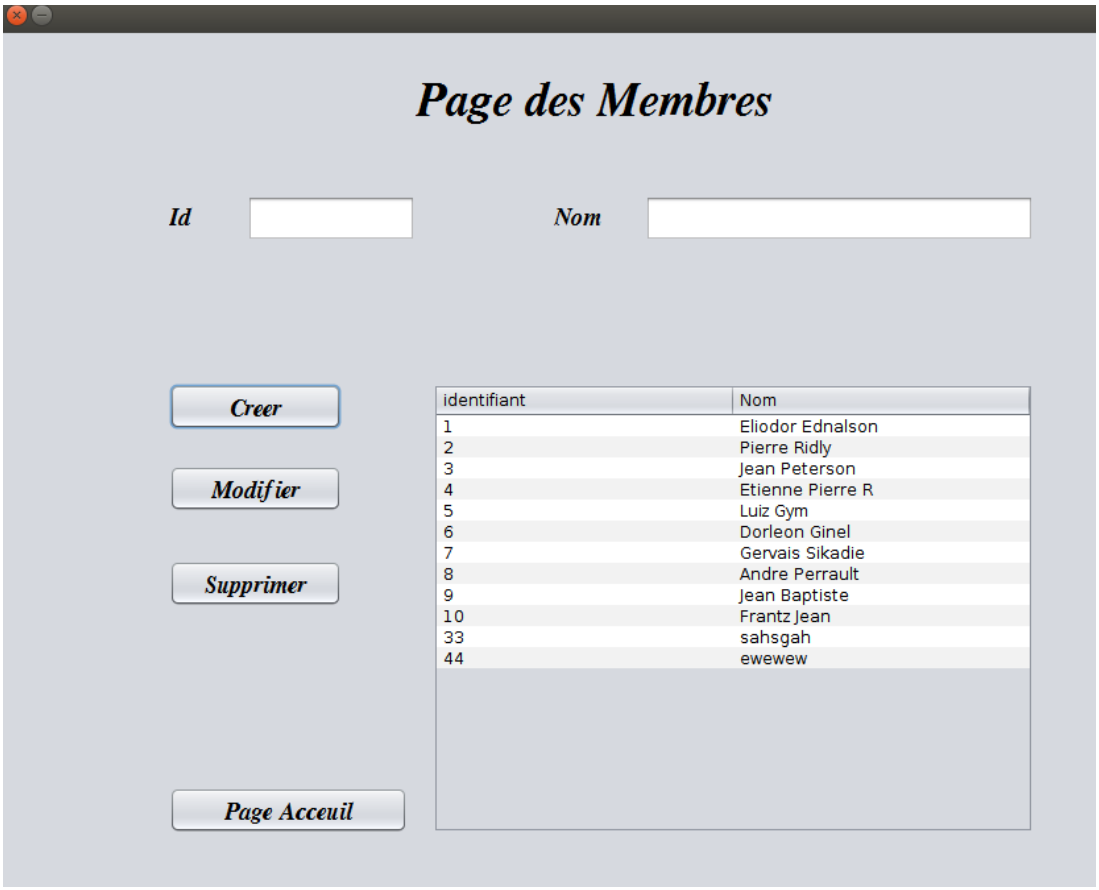


Figure : accès a la Page des Membres après avoir cliquer sur « Les Membres »

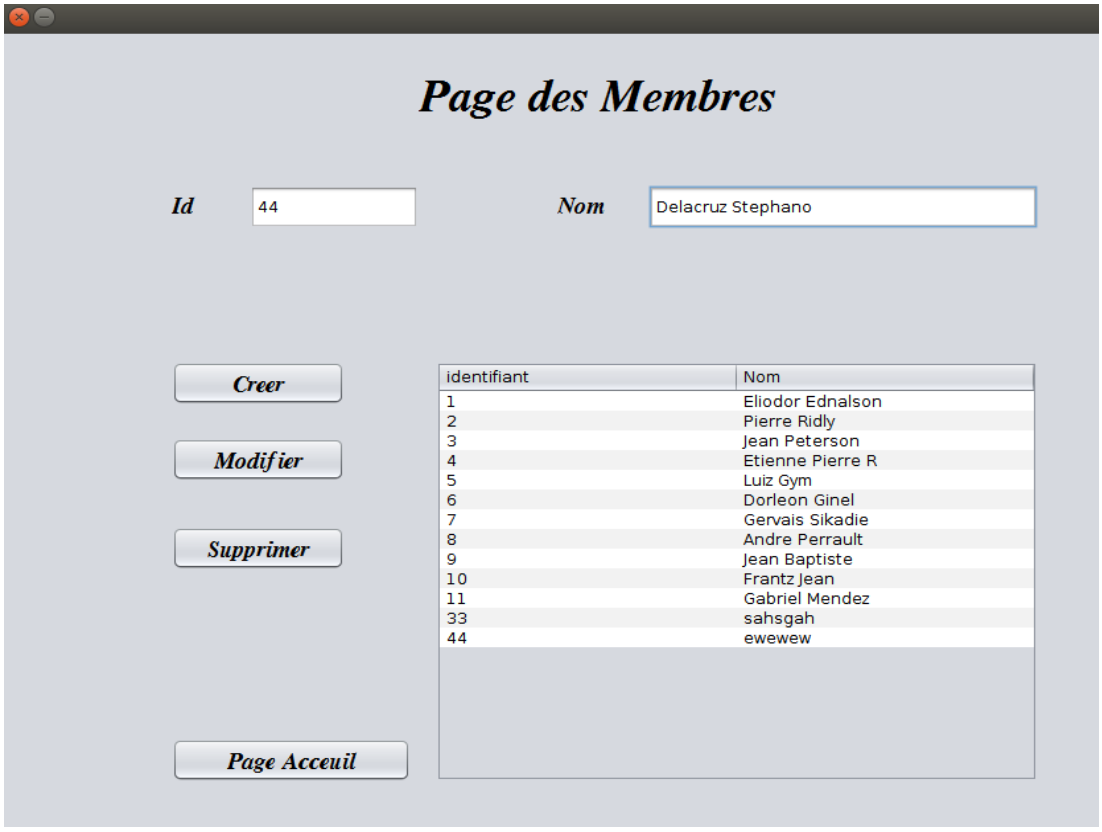


Figure : Entrer l'id du membre a modifier et modifier le nom

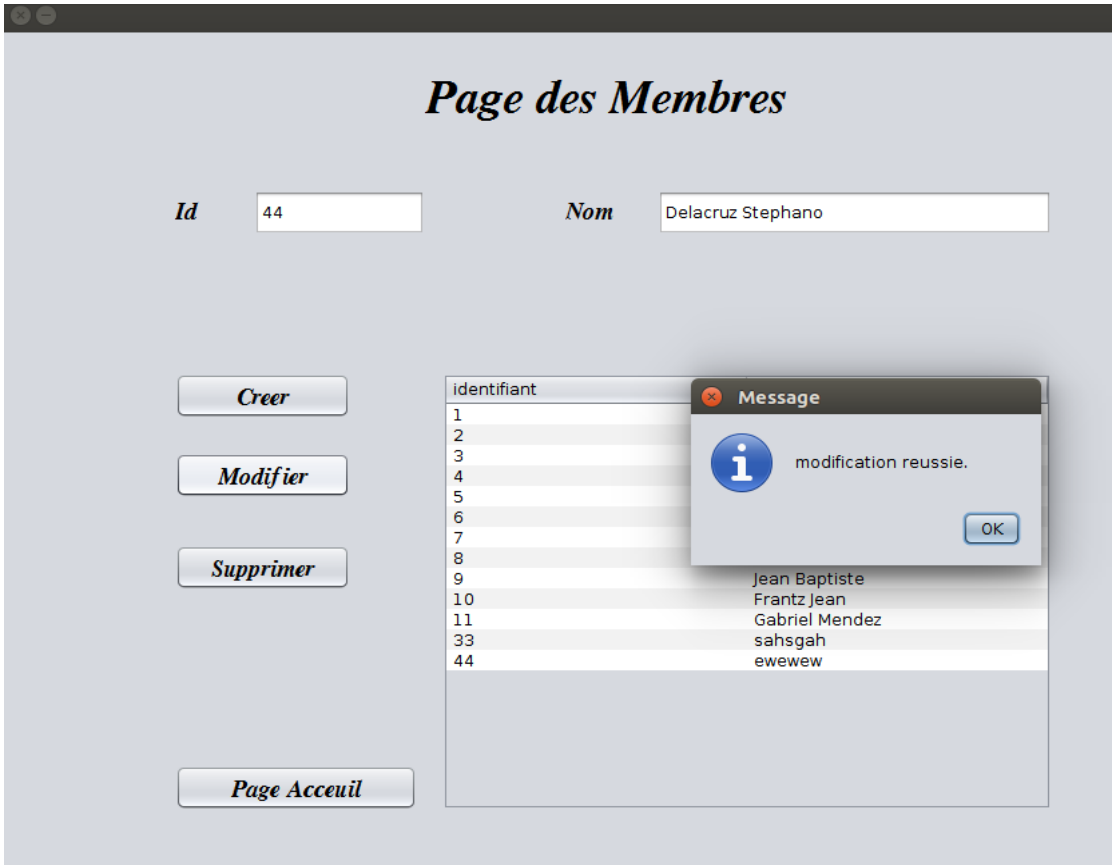


*Figure : modification réussie après avoir cliquer sur **ok**, vous aurez accès a la liste a jour.*

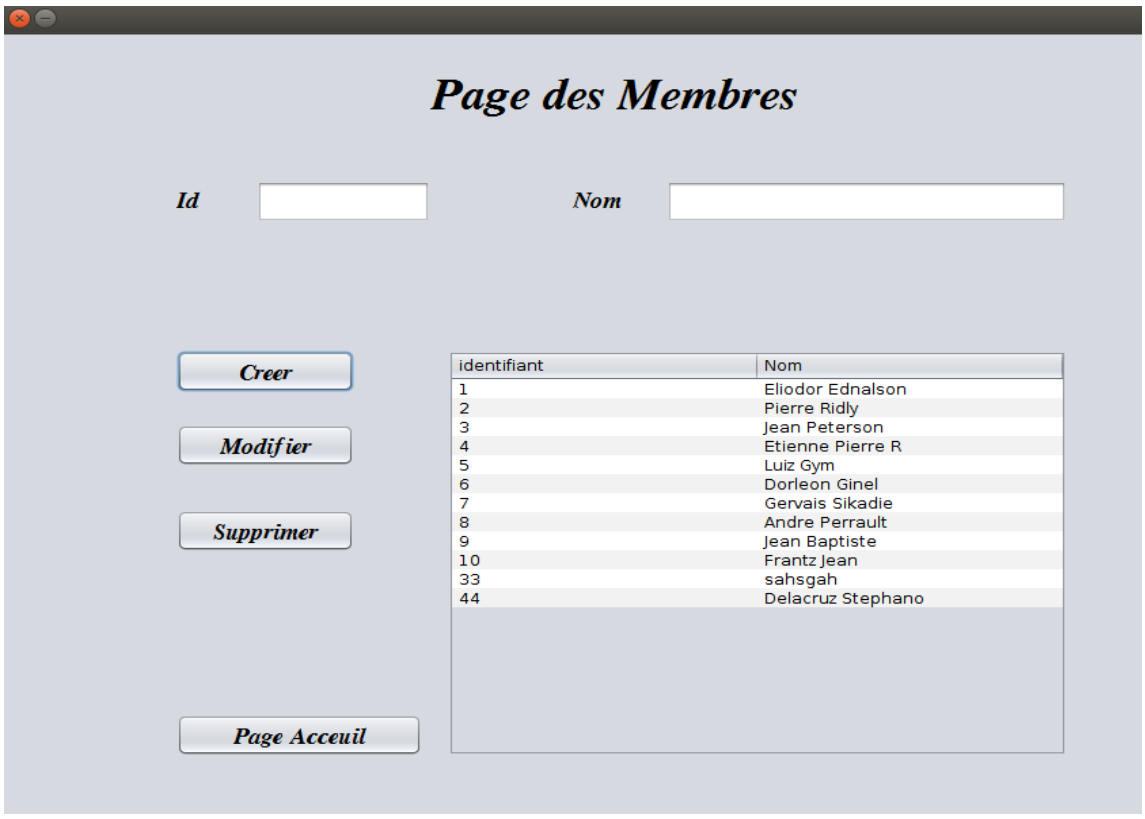


Figure : Liste mise a jour (Donc, le test d' acceptation a été réussi)

Action 6 : Ajouter une tache

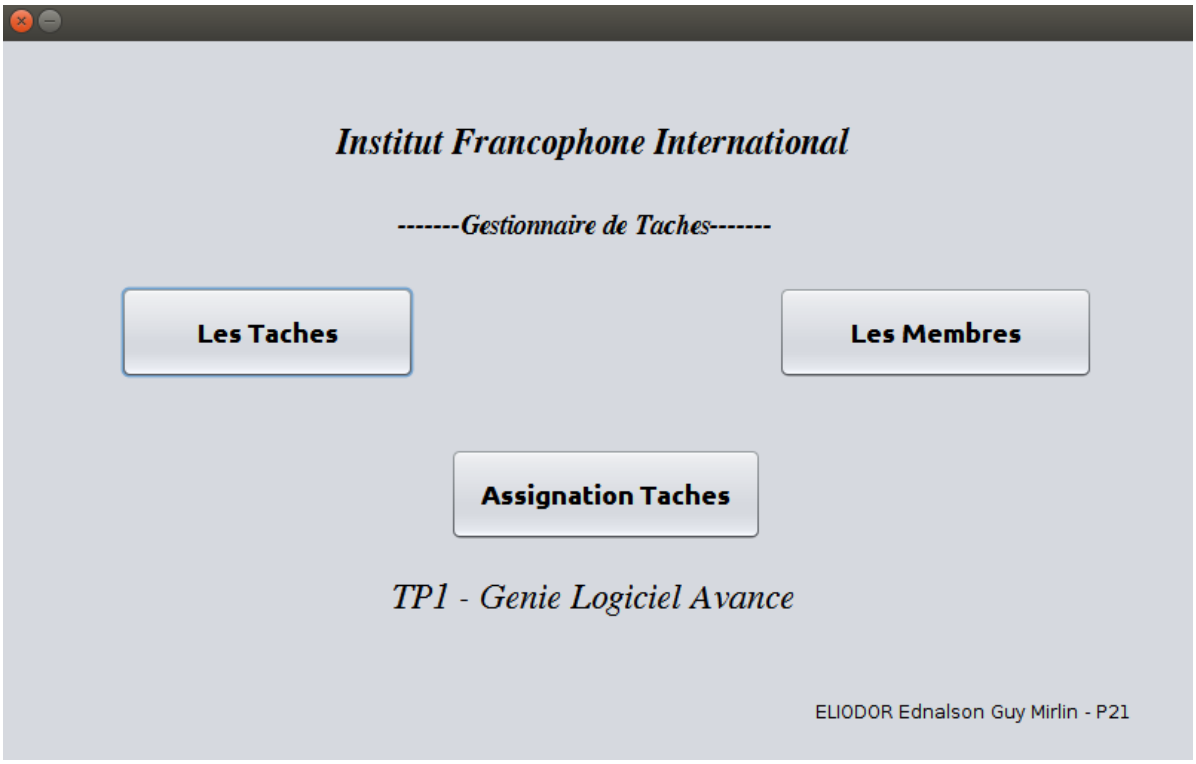


Figure : Page d'accueil du petit gestionnaire

Figure : Après avoir saisi les infos et cliquer sur créer, le pop up affichera ajout reussie

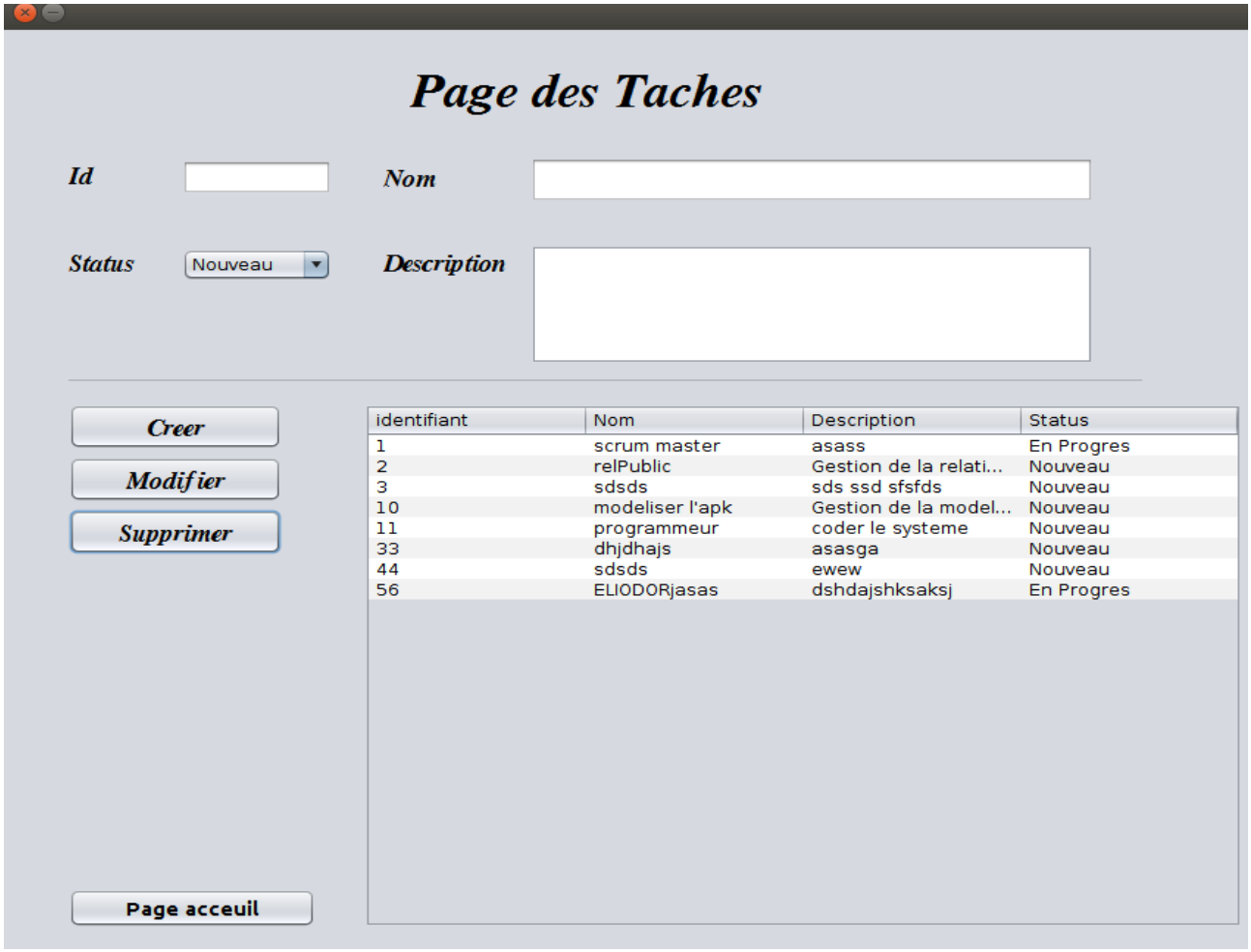


Figure : Au final on trouve la tache 11 bel et bien enregistré

Action 10 : Chercher et afficher toutes les taches assignées a un membre (par son id).

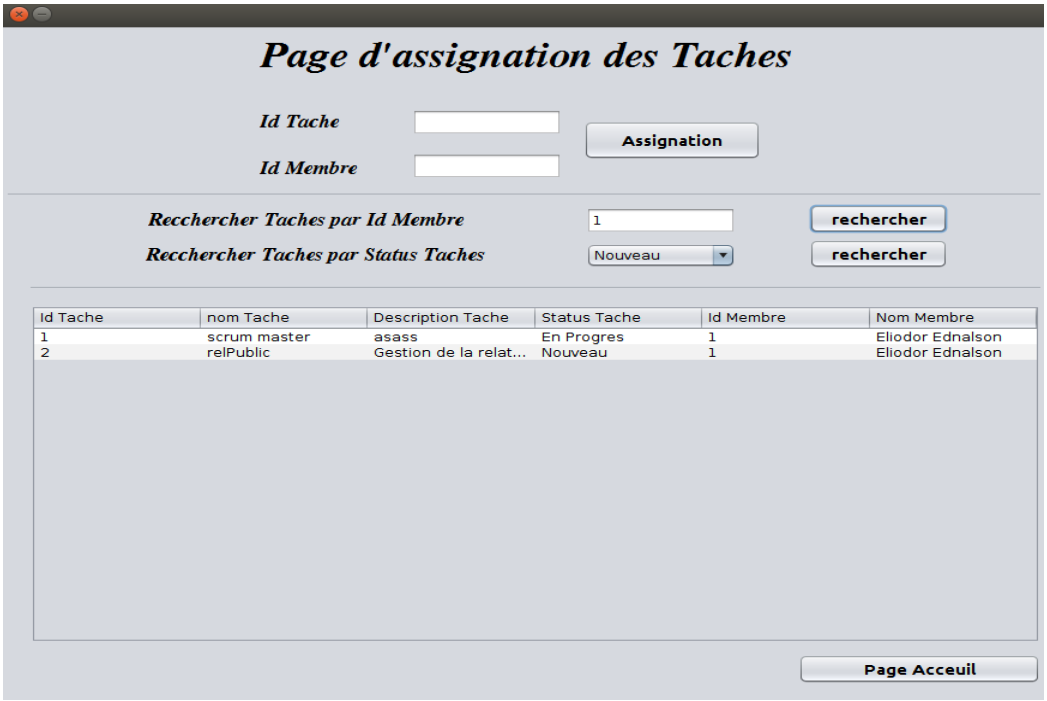


Figure : Liste des taches assignées au membre #1

Action 11 : Chercher et afficher toutes les taches en fonction de leur status (Avec le nom du assigné)

Figure : Recherche toutes les taches en fonction du statu ayant pour valeur: Nouveau

5. Conclusion

Dans le cadre de ce TP nous avons implémenté un petit gestionnaire de tache en Java. Réalisé ce TP, nous a permis de faire un rappel sur les notions de la programmation orientée objet et la modélisation avec UML. il nous a permis de mieux suivre une méthodologie de travail bien étudié, d'approfondir des connaissances sur les méthodes de développement des applications.

6. Références

[1] : Laurent Audibert, UML 2 - De l'apprentissage a la pratique (consulte le 12 Aout 2017), http://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-cas-utilisation

[2] : Pierre Gérard, UML – Diagramme de sequences (Consulte le 13 Aout 2017), http://lipn.univ-paris13.fr/~gerard/uml-s2/uml-cours05.html