

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Benyoucef BENKHEDDA- Alger1

Faculté des Sciences

Département Mathématiques et Informatique



Master Ingénierie des Systèmes Informatiques Intelligents (ISII)

Rapport du TP

Conception et implémentation d'un réseau social d'échange de documents avec la méthode SCRUM et les systèmes multi-agents.

Module : *Principes fondamentaux de la gestion des processus métier*

Réalisé par :

- *ALILI Houda.*
- *BENBABA Rym Amina.*
- *DAHDOUH Ahmed.*

Groupe 02

2019/2020

Table des matières :

Liste des figures :.....	3
Liste des tableaux :.....	4
Introduction :	5
Méthodologies de conception & développement :	6
a) Les rôles :.....	7
b) Le processus :.....	7
c) Planification :.....	7
d) La revue de sprint :.....	8
Analyse et Conception :.....	9
I. Identification des besoins :	9
1. Identification des acteurs :.....	9
2. Identification des besoins fonctionnels :.....	9
3. Identification des besoins non fonctionnels :.....	10
II. Architecture générale du système :	10
III. Spécification des besoins UML :.....	11
1. Diagramme de cas d'utilisation global :	11
2. Les rôles :	11
3. Backlog du produit : (user story) : C'est le catalogue des besoins.....	12
Implémentation :.....	31
I. Environnement et Plateforme de travail :.....	31
II. Présentation de la plateforme JADE :.....	31
III. Structure Logique du Système :	33
IV. Diagramme de classe :	34
V. Mise en Œuvre du projet :.....	35
VI. Capture d'écran :.....	37
Conclusion :.....	45
Références :	46

Liste des figures :

Figure 1: Architecture générale de Scrum.	6
Figure 2: Architecture général du système.....	10
Figure 3: Diagramme de cas d'utilisation global du système.	11
Figure 4: Diagramme de cas d'utilisation détaillé de premier sprint.	13
Figure 5: Diagramme de séquence de l'inscription.....	14
Figure 6: Diagramme de séquence de l'authentification.	15
Figure 7: Diagramme de séquence de modification de profil.	16
Figure 8: Le diagramme de cas d'utilisation détaillé de deuxième sprint.	17
Figure 9: Diagramme de séquence pour l'envoi d'un document.....	18
Figure 10: Diagramme de séquence pour le cas de réception de document.....	19
Figure 11: interface de l'inscription.	20
Figure 12: code pour le cas d'inscription.....	22
Figure 13: Interface de l'authentification.	22
Figure 14: code pour le cas d'authentification.	24
Figure 15: Interface de l'affichage de profil (page d'accueil).	24
Figure 16: Code pour la page d'accueil.	25
Figure 17: Code pour le cas ' modifier le profil'.....	27
Figure 18: Code pour l'envoi et la réception des messages.	30
Figure 19: la répartition de la plateforme JADE en conteneurs.....	32
Figure 20: : Architecture modulaire des agents sous JADE	33
Figure 21: Diagramme de classe.	34
Figure 22: création de main container.	35
Figure 23: Création des containers secondaires.	36
Figure 24: Gestion des erreurs pour le cas d'inscription.....	38
Figure 25: gestion d'erreur si la création dépasse 2 utilisateurs.	39
Figure 26: La gestion d'erreur pour le cas d'authentification.....	40
Figure 27: la page d'accueil.....	41
Figure 28: créations d'user coté Eclipse	41
Figure 29: la représentation des agents sous la plateforme jade.	41
Figure 30: interface de l'envoi.	42
Figure 31: l'envoi de message sous jade.	42
Figure 32: Contenu de message envoyé.....	43
Figure 33: Interface qui contient le fichier envoyé.	44
Figure 34: l'interface de réception de document.	44

Liste des tableaux :

Tableau 1: Les besoins fonctionnels.....	9
Tableau 2: Les rôles de ce système.	11
Tableau 3: Détail de sprint 1.	12
Tableau 4: Détail de sprint 2.	13
Tableau 5: L'environnement logiciel et matériel de mise en œuvre de cette application.	31

Introduction :

Le service des réseaux sociaux aujourd'hui est l'un des services prédominant du web. Ces réseaux visent un large éventail d'utilisateurs de toutes tranches d'âges et de sexes et cela pour diverses raisons tant personnelles que professionnelles. Donc, ces réseaux sociaux sont tout simplement des communautés en ligne focalisées sur les membres qui se nouent des relations basées sur des intérêts communs entre eux.

L'avantage qu'offre ces réseaux est leur facilité d'utilisation pour publier des informations personnelles et communiquer avec aisance, même par des utilisateurs aux compétences techniques limitées.

La motivation principale pour un membre à joindre un tel réseau est la facilité de créer un profil, d'utiliser les différentes applications offertes par le service ainsi que la possibilité de partager facilement des informations avec des contacts sélectionnés ou le public et cela à des fins personnelles ou professionnelles.

En effet, La simulation Multi-Agents nous apparaît être la démarche la plus adéquate à la modélisation de ce type de systèmes car elle est un domaine de recherche relativement récent, qui se base sur la notion d'agent.

En effet, un système Multi-Agents est naturellement adapté pour décrire et simuler un système composé d'entités, il permet au modèle de paraître plus proche de la réalité. D'une part, un système Multi-Agents est particulièrement bien adapté à la description d'un système du point de vue de l'activité de ses composantes, c'est-à-dire lorsque le comportement des individus est complexe (difficile à décrire avec des équations). D'une autre part, la modélisation est plus facilement interprétable par un observateur humain, car la description par un SMA est plus naturelle que par de simples processus ; la validation par un expert en sera donc facilitée, car il pourra facilement se rapporter au monde réel.

Pour cela, notre mini projet inscrit dans le domaine des réseaux sociaux qui vise à concevoir un réseau social d'échange documents avec la méthode SCRUM et l'implémenter avec les systèmes multi- agents afin de faire communiquer plusieurs agents entre eux tels que chacun représentant un usager du réseau social.

Méthodologies de conception & développement :

Une méthode d'analyse et de conception est un procédé qui a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client. On distingue deux familles de méthodes : Les méthodes agiles et les méthodes du processus unifié.

Dans notre projet nous nous consacrerons sur la méthode Agile : **SCRUM**.

- Une **méthode agile** est une méthode de développement informatique permettant qui repose sur une structure commune (**itérative, incrémentale et adaptative**) et qui consiste à découper le projet en plusieurs étapes qu'on appelle « **itérations** ».
- L'utilisation de la méthodologie **SCRUM** offre la possibilité de développer uniquement les fonctionnalités qui apportent une valeur ajoutée au produit, en toute transparence, avec le souci de la qualité et du respect des délais et avec un retour rapide de la part du client.

Le principe de base de Scrum est de focaliser l'équipe de façon itérative sur un ensemble de fonctionnalités à réaliser, dans des itérations de durée fixe d'une à quatre semaines, appelées « *sprints* », ainsi que l'auto-organisation de l'équipe de développement tel que chaque sprint commence par une estimation suivie d'une planification opérationnelle et se termine par une démonstration de ce qui a été achevé (Un sprint aboutit toujours sur la livraison d'un produit partiel fonctionnel) et contribue à augmenter la valeur d'affaires du produit [1].

Le processus de la méthode SCRUM est illustré par la figure suivante :

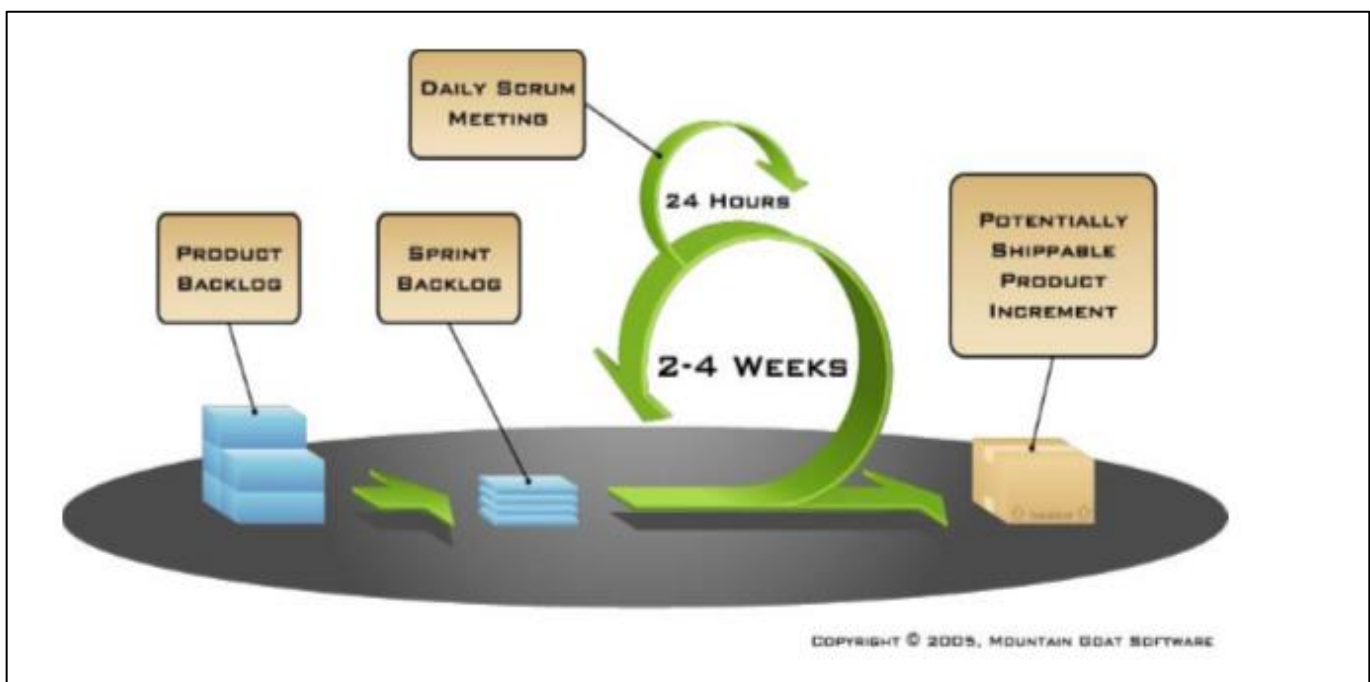


Figure 1: Architecture générale de Scrum.

a) Les rôles :

On distingue plusieurs rôles, dont :

- Le directeur de produit (Product owner) est le représentant des clients et des utilisateurs, et fait également parti de l'équipe.
- Le Scrum Master veille à l'application de la méthodologie Scrum au sein de l'équipe.
- L'équipe qui contribue à la réalisation des fonctionnalités du projet (planification, développement, test et documentation).

b) Le processus :

Tous les critères ou exigences du produit sont regroupées dans des journaux ou backlogs dont on distingue 2 types :

- Le backlog de produit ou < **Product backlog** > qui regroupe la liste des fonctionnalités du produit.
- Le backlog de sprint ou < **sprint backlog** > en fonction des fonctionnalités du produit. Il regroupe la liste des tâches qui devra être réalisées à l'itération en cours. Chaque tâche aurait fait l'objet d'une estimation préalable de charge par l'ensemble des membres de l'équipe afin d'estimer au mieux les tâches qui peuvent réaliser un sprint.

c) Planification :

Le sprint : Dès le début d'un projet, la première planification permet de définir le périmètre de chaque itération appelé sprint. Chaque sprint dure quelques semaines et regroupe une liste de tâches (défini dans le backlog).

La mêlée quotidienne : on appelle une mêlée quotidienne d'un quart d'heure qui consiste chaque jour avec les membres de l'équipe ainsi que le directeur de produit de se tenir au courant de l'avancement du projet, notamment en :

- Faisant le point sur le travail effectué la veille par chacun.
- Définissant les tâches qui sont réalisées durant la journée.
- Résolvant les éventuels problèmes qui avait ou qui pourrait être rencontré par chacun.
- Le développement suit un processus itératif et incrémental : de nouvelles fonctionnalités sont rajoutées au produit.

d) La revue de sprint :

La fin d'un sprint aboutit à la réalisation d'un produit avec des fonctionnalités partielles avec la documentation associée. Dans la plupart des cas, cela conduit à une revue de sprint consistant à faire une démonstration de la réalisation du sprint devant le client afin de valider le travail réalisé et d'avoir un retour pour éventuellement ajuster le backlog de produit [1].

Maintenant après avoir clôturer l'architecture de la méthode agile, nous appliquons ces notions et concepts dans notre mini projet afin de satisfaire le but souligné au début.

Analyse et Conception :

Nous allons présenter la conception de notre système qui est nécessaire pour comprendre son mécanisme de fonctionnement, puisqu'elle donne une généralisation de ce travail, et établit un couplage explicite entre les concepts et les mécanismes internes. Nous marquons ainsi les outils utilisés pour implémenter notre système et les résultats obtenus.

I. Identification des besoins :

L'identification des besoins est une étape primordiale et importante afin de déterminer les besoins et les attentes exactes d'un client. Ces besoins se déclinent en deux types de besoins : **Fonctionnels** et **non fonctionnels**.

1. Identification des acteurs :

Un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent directement avec le système étudié.

Notre système est constitué d'un seul acteur qui est :

- **L'utilisateur** qui possède un compte et identifié par un nom et un mot de passe qui lui permettent d'accéder à l'application et de bénéficier de ses services dans le centre d'intérêt qui lui appartient.

2. Identification des besoins fonctionnels :

Le système doit être capable de :

Fonctionnalité	Description
Inscription et authentification	Ces deux fonctionnalités permettent aux utilisateurs d'accéder à ce système et leurs profils.
Gestion de profil.	Permet aux utilisateurs de gérer leurs propres comptes (l'afficher et le modifier).
Partager les documents.	Partager les documents avec les différents utilisateurs de système.

Tableau 1: Les besoins fonctionnels.

3. Identification des besoins non fonctionnels :

Les besoins non fonctionnels représentent les exigences implicites auquel le système doit répondre. Parmi ces besoins nous citons les principaux pour notre cas :

- **Sécurité** : La solution proposée permet à l'utilisateur une navigation sécurisée. Elle n'est accessible qu'avec une authentification.
- **Ergonomie** : L'application doit satisfaire les critères de l'ergonomie suivants : La lisibilité des interfaces, le guidage et la facilité d'utilisation par les feed-back.
- **Portabilité l'application** : doit être accessible sans bugs.

II. Architecture générale du système :

L'architecture générale de notre système est représentée dans la figure ci-dessus, telle que chaque communauté sociale est représentée par 2 agents seulement (comme indiquer dans l'énoncé) ; d'où les communautés créés sont les deux branches : 'Informatique' et 'Mathématique' :

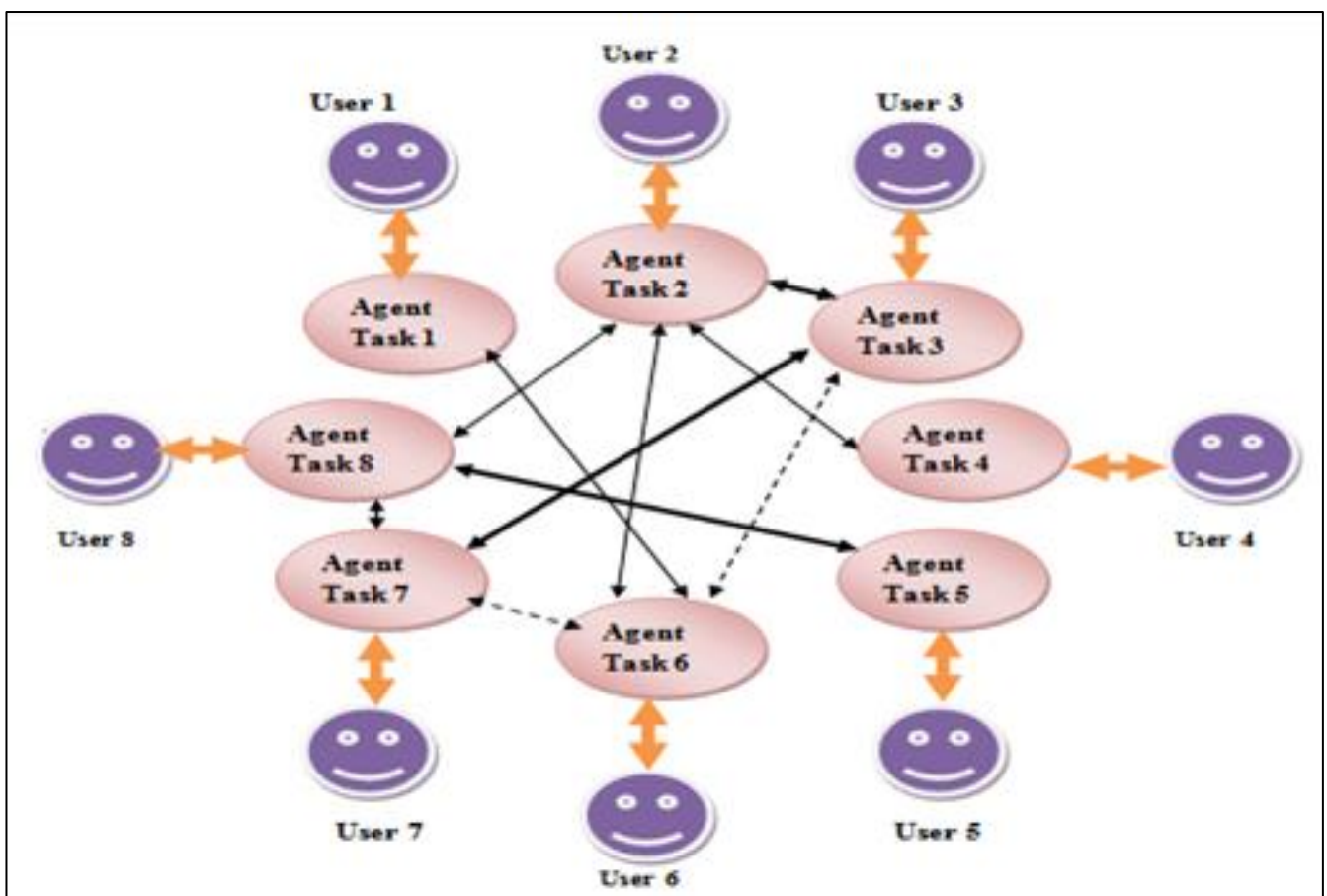


Figure 2: Architecture général du système.

III. Spécification des besoins UML :

1. Diagramme de cas d'utilisation global :

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

La figure suivante modélise le diagramme de cas d'utilisation global du système qui présente le rôle de **l'utilisateur** qui est le responsable sur l'opération de l'inscription, la gestion de son profil et la communication avec d'autres utilisateurs après avoir bien s'authentifier dans le système :

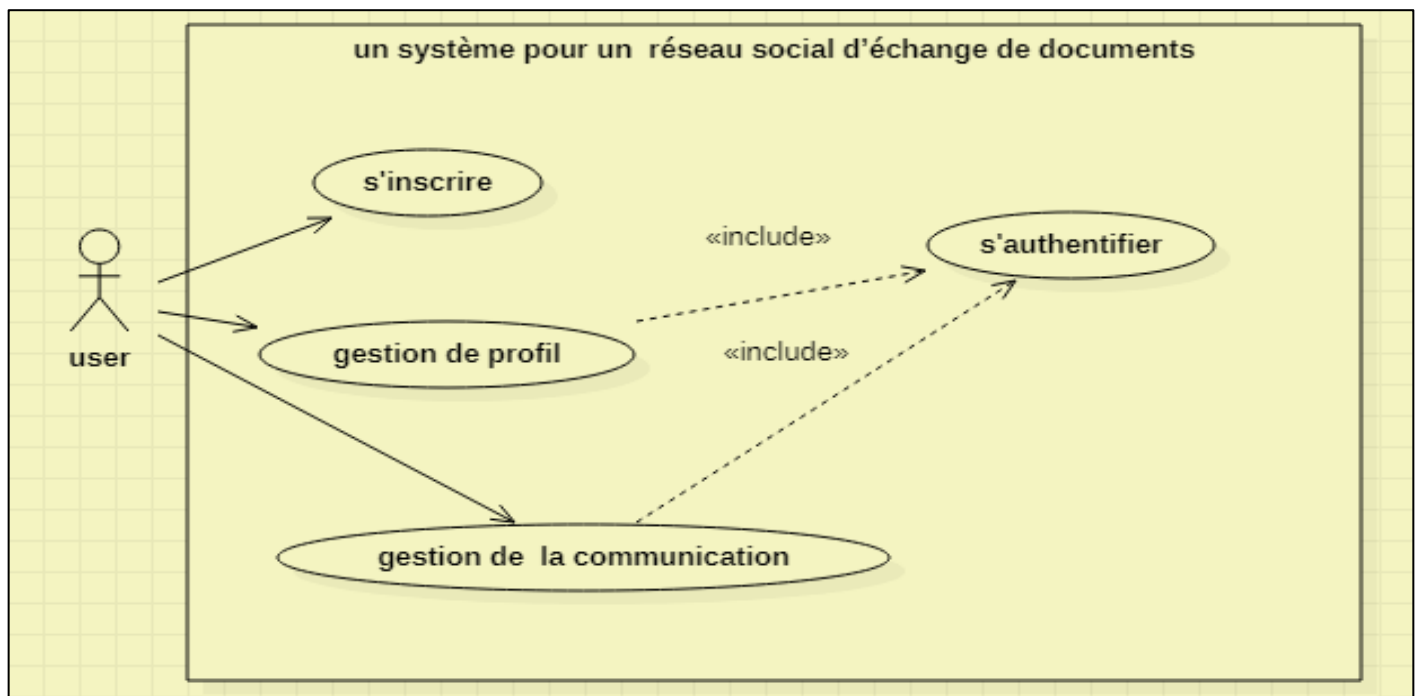


Figure 3: Diagramme de cas d'utilisation global du système.

2. Les rôles :

Rôle SCRUM	Personne (s) affectée (s)
Product owner	Université Alger 1 : Département MI ; Spécialité ISII.
Scrum master	L'enseignante : Madame « Chekkai »
Team	Les concepteurs et les développeurs : « Alili Houda », « Benbaba Rym Amina » et « Dahdouh Ahmed ».

Tableau 2: Les rôles de ce système.

3. **Backlog du produit : (user story)** : C'est le catalogue des besoins.

3.1. Définition du backlog du produit :

Sprint 1 :

- Inscription.
- Authentification.
- Gestion de profil.

Sprint 2 :

- Gestion de la communication (l'envoi et la réception des documents).

3.2. Détail de chaque sprint :

Les deux tableaux suivants représentent la description de chaque sprint de notre application tel que chaque user story est caractérisé par une priorité :

Sprint 1	User story	Entant que	Je veux	Priorité
- INSCRIPTION. - AUTHENTIFICATION. - GESTION DE PROFIL.	Inscription	Non utilisateur	M'inscrire	1
	Authentification	Utilisateur	M'authentifie	2
	Gestion de profil	Utilisateur	Afficher mon profil	3
	Gestion de profil	Utilisateur	Modifier mon profil	3

Tableau 3: Détail de sprint 1.

Sprint 2	User story	Entant que	Je veux	Priorité
-Gestion de la Communication (Envoi et réception)	Gestion de l'envoi.	Utilisateur	Consulter la liste des contacts	4
	Gestion de l'envoi	Utilisateur	Choisir le document à envoyer	4
	Gestion de l'envoi	Utilisateur	Décrire le document à envoyer	4
	Gestion de l'envoi	Utilisateur	Envoyer un document	4
	Gestion de la réception	Utilisateur	Afficher la liste des documents reçus	5
	Gestion de la réception	Utilisateur	Consulter les documents reçus	5

Tableau 4: Détail de sprint 2.

3.3. Analyse des sprints :

Sprint 1 : (Inscription, Authentification et Gestion des profils)

❖ Diagramme de cas d'utilisation détaillé :

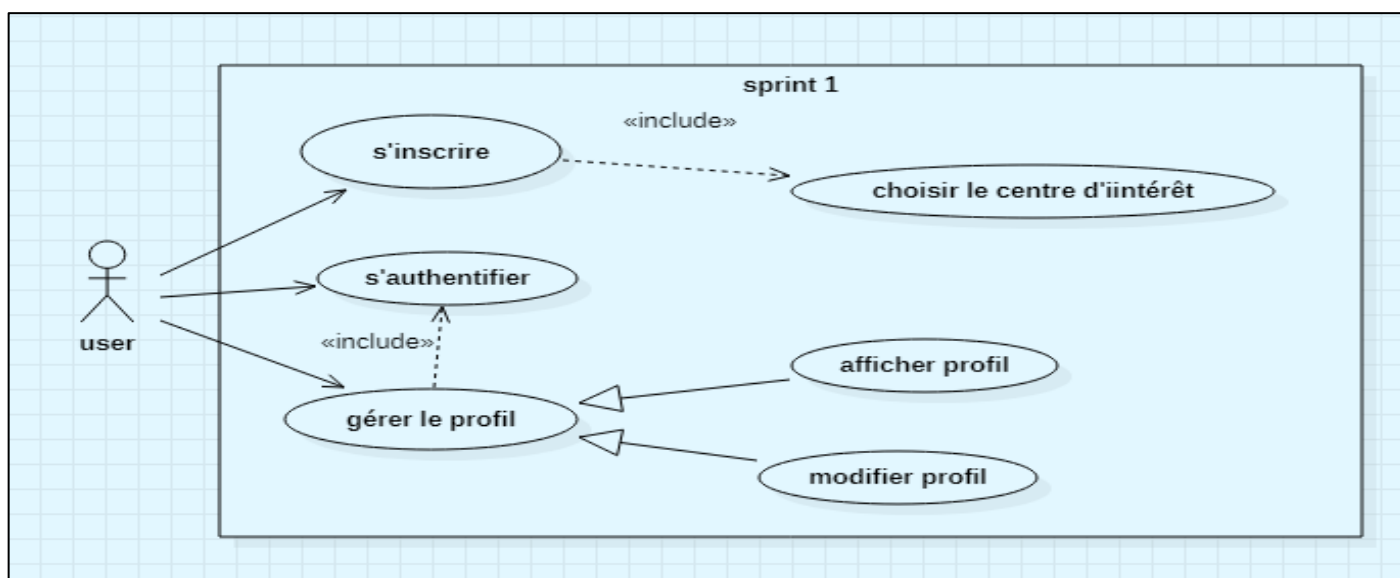


Figure 4: Diagramme de cas d'utilisation détaillé de premier sprint.

❖ Diagrammes de séquence :

Les diagrammes de séquences suivants sont la représentation graphique des interactions entre l'acteur et le système selon un ordre chronologique dans la formulation UML.

a. Inscription :

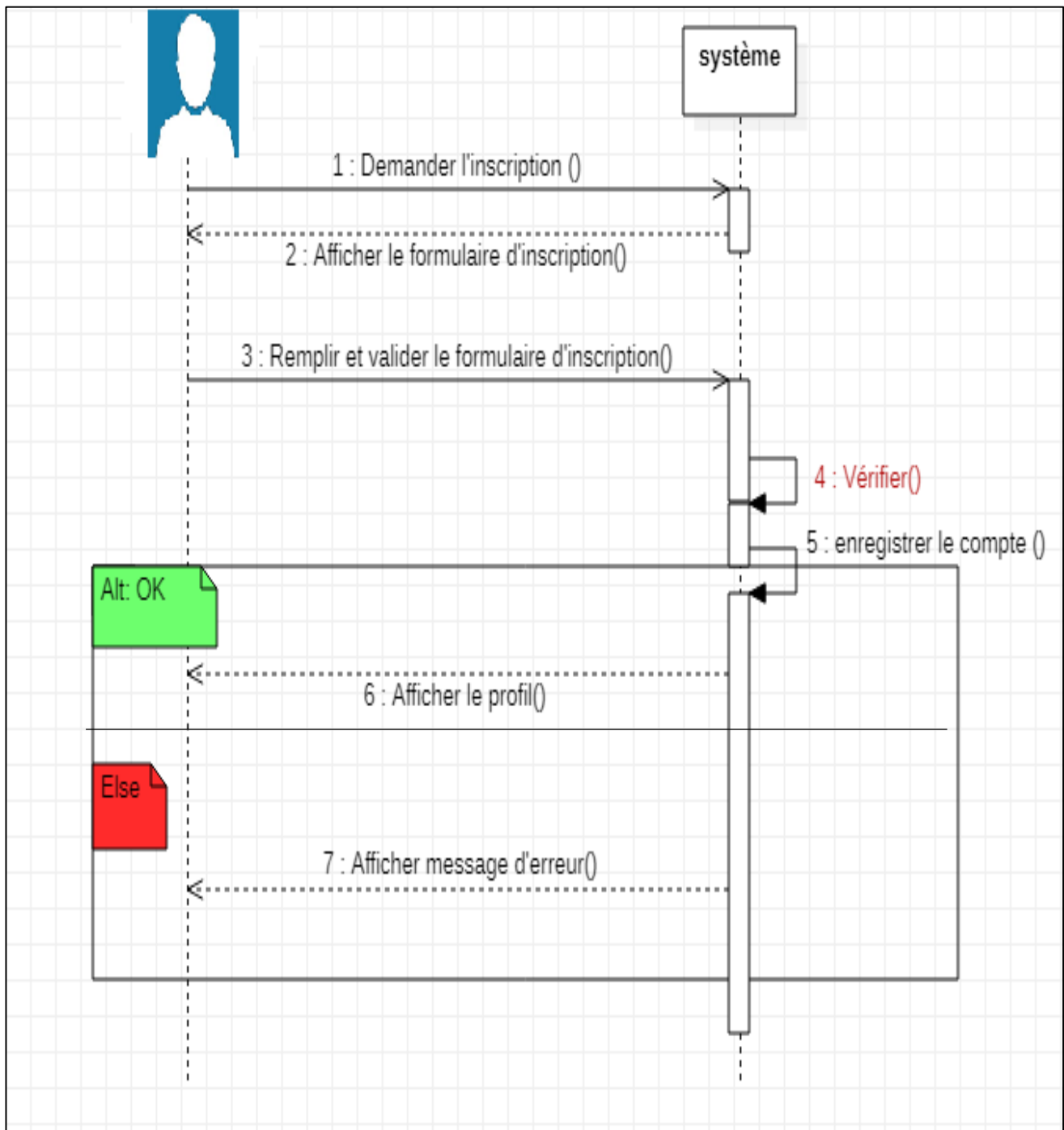


Figure 5: Diagramme de séquence de l'inscription.

b. Authentication :

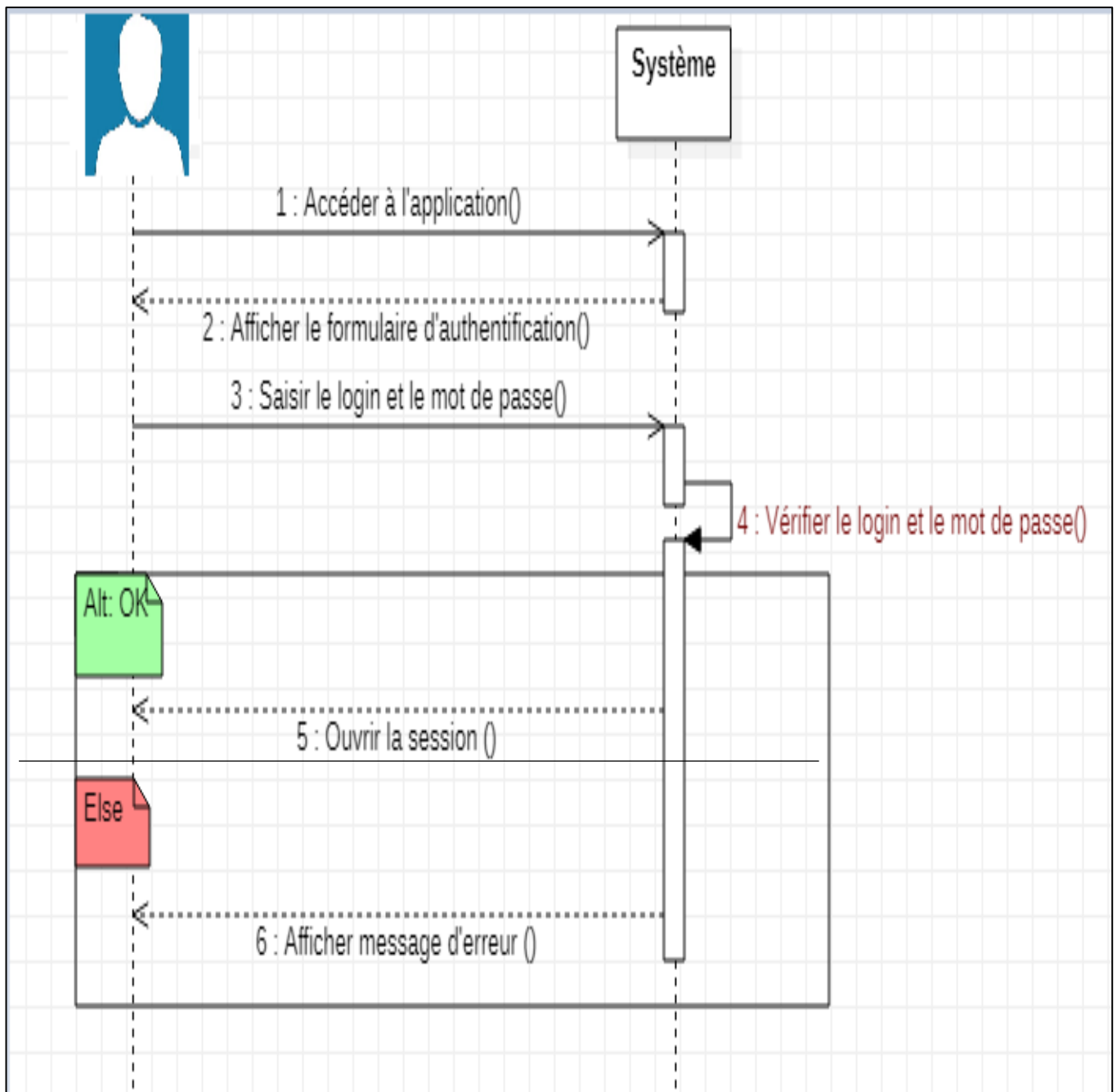


Figure 6: Diagramme de séquence de l'authentification.

c. **Modifier profile :**

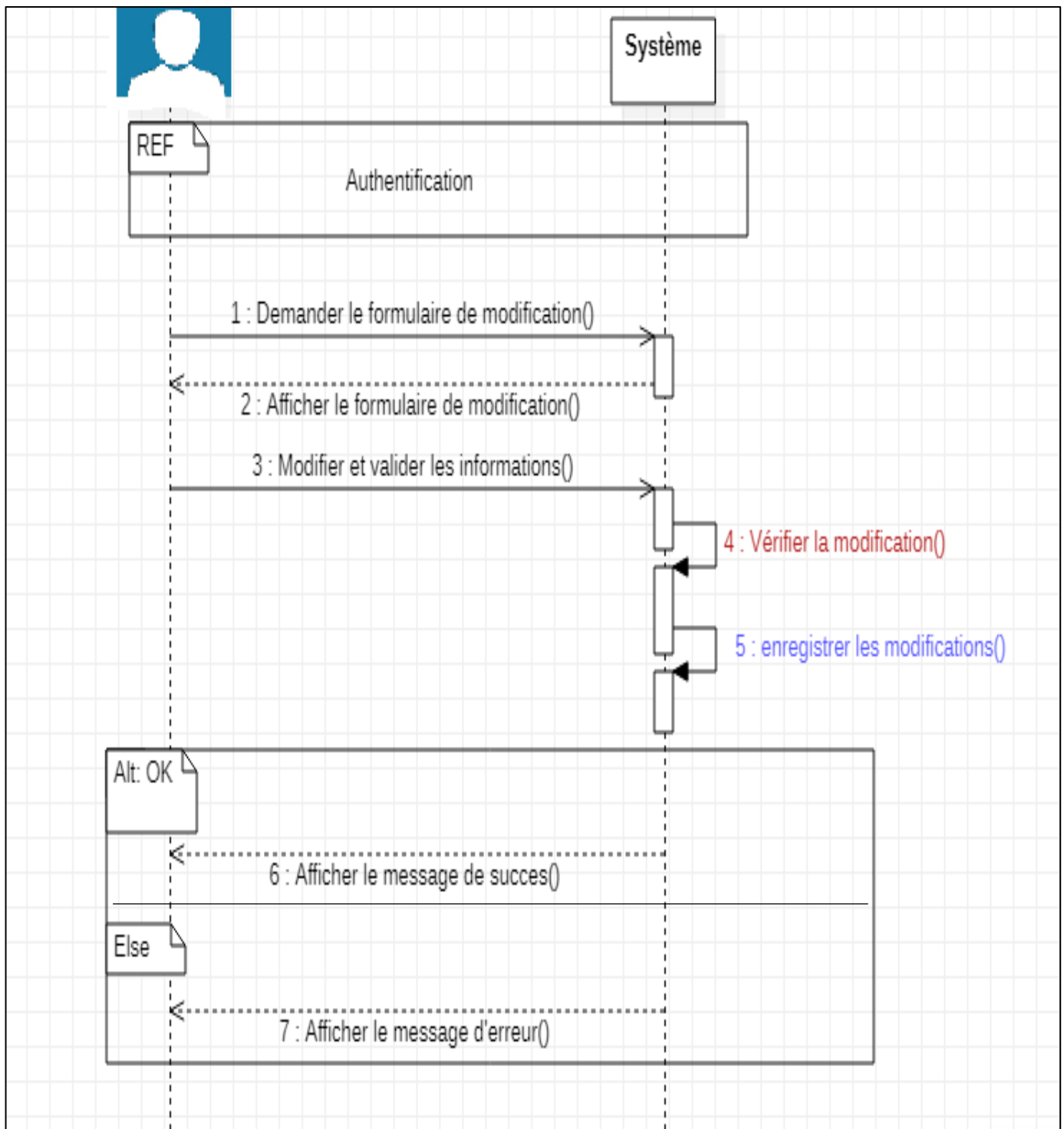


Figure 7: Diagramme de séquence de modification de profil.

Sprint2 : (Communication 'envoi et réception des documents')

❖ Diagramme de cas d'utilisation détaillé :

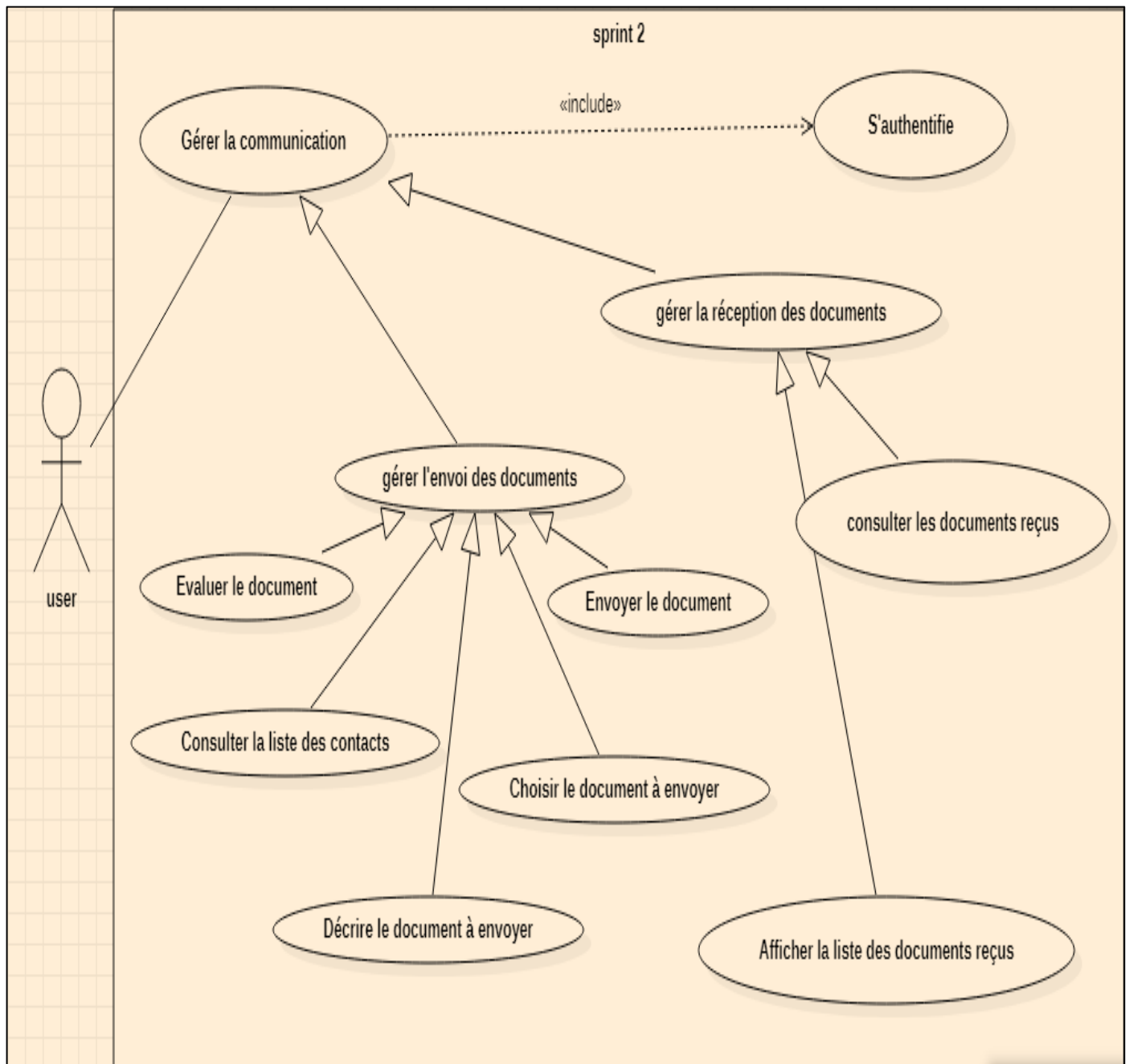


Figure 8: Le diagramme de cas d'utilisation détaillé de deuxième sprint.

❖ **Diagramme de séquence :**

a. Envoi d'un document :

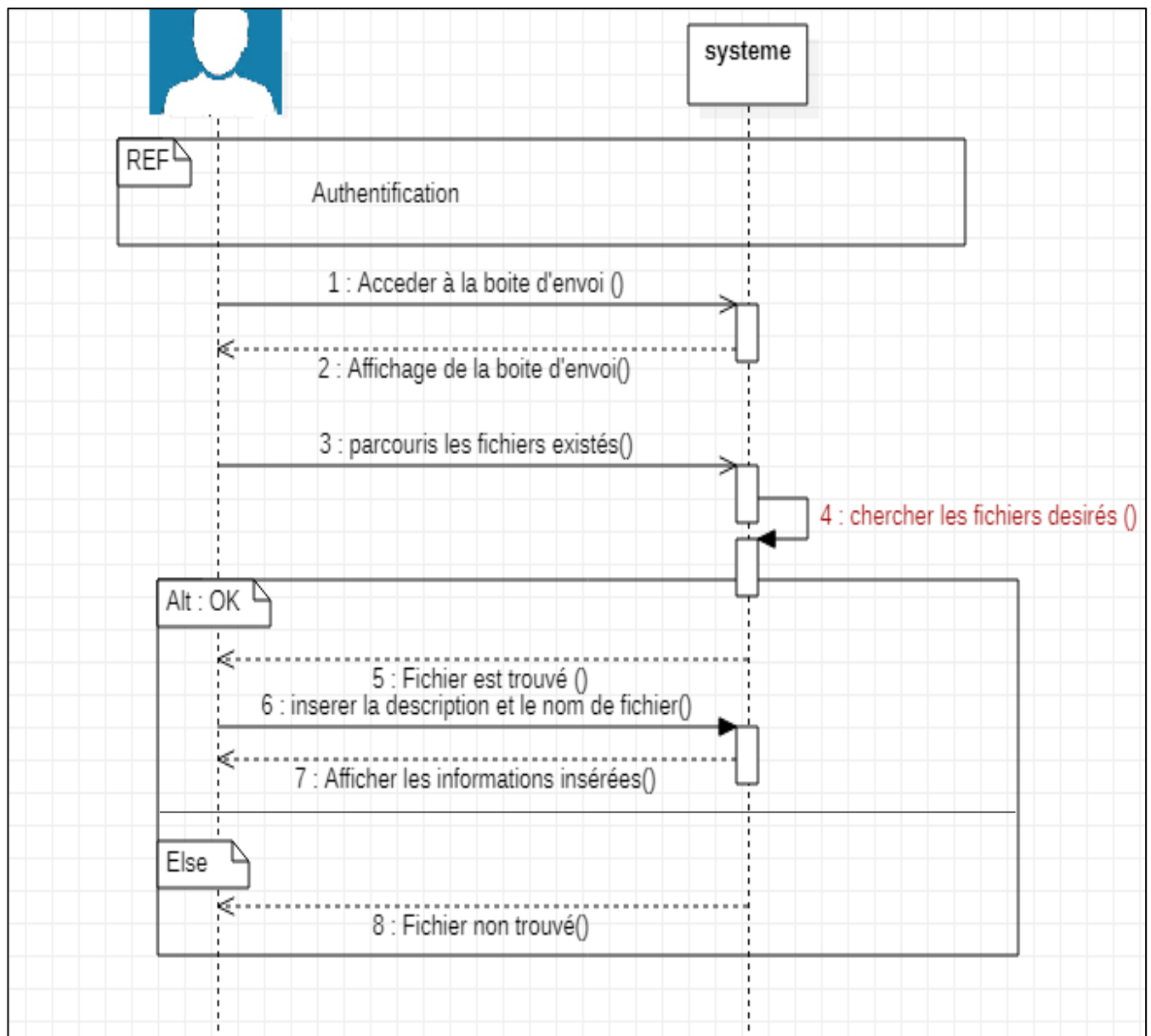


Figure 9: Diagramme de séquence pour l'envoi d'un document.

b. Réception d'un document :

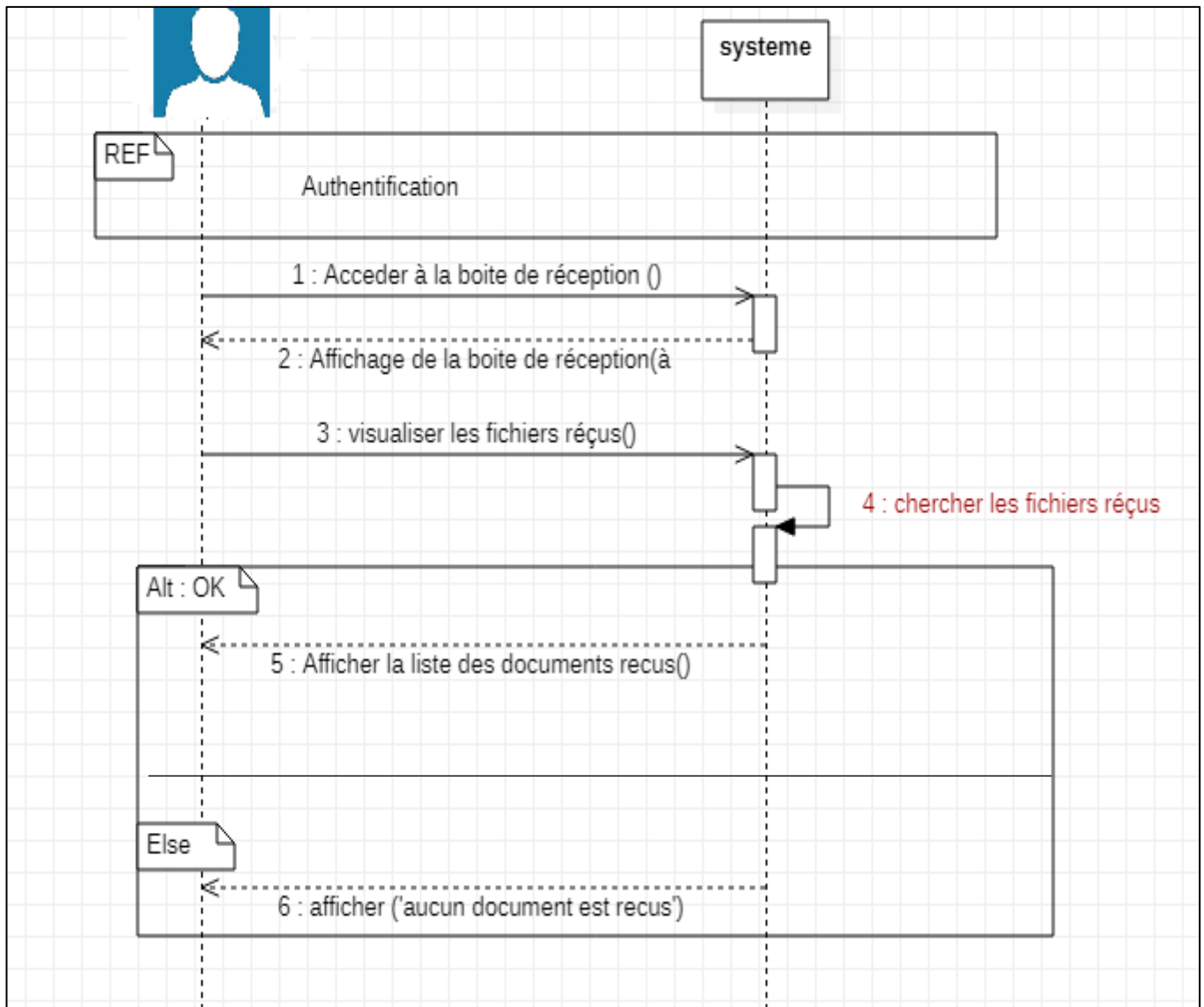


Figure 10: Diagramme de séquence pour le cas de réception de document

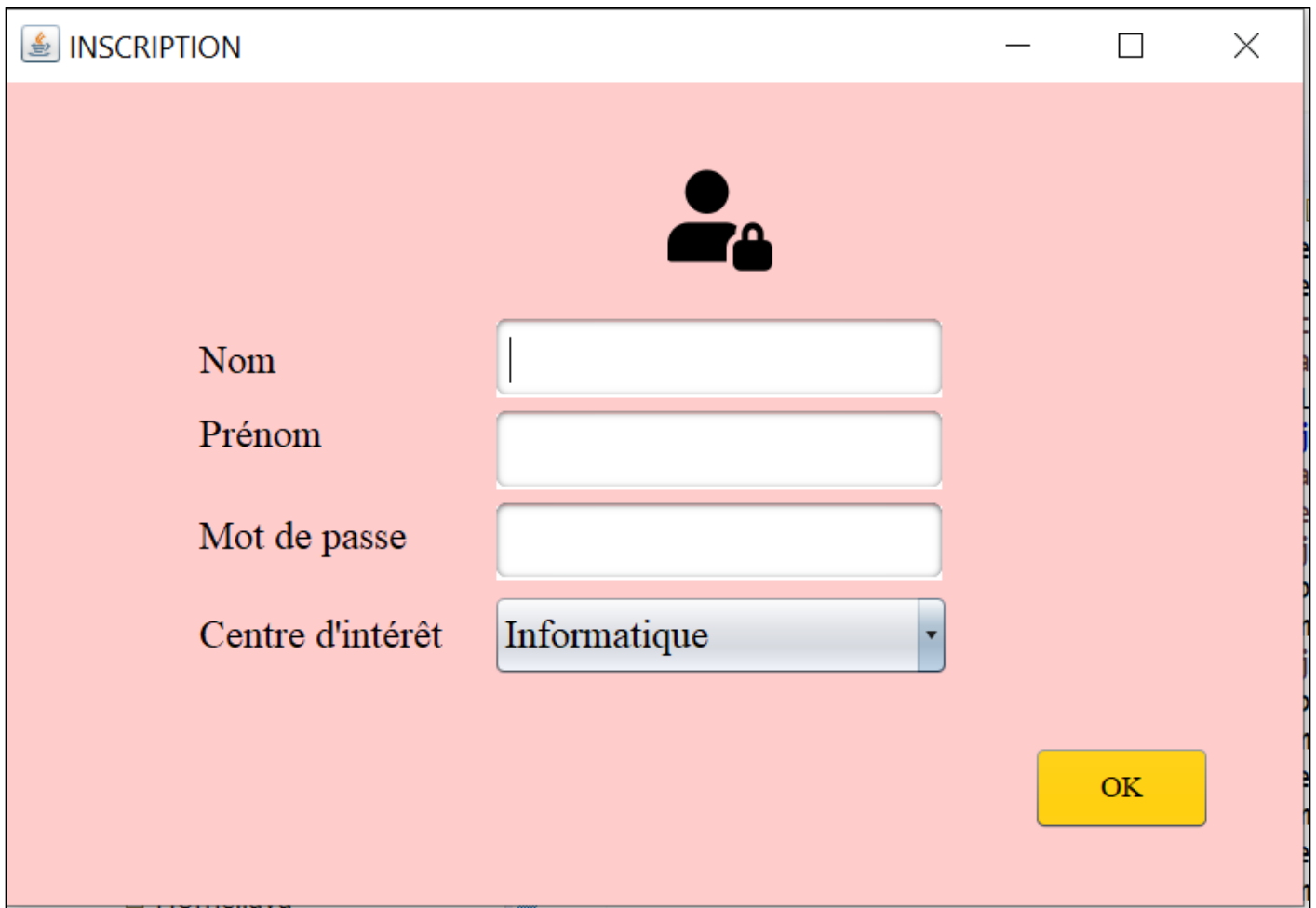
3.4. Développement des sprints : (interface +code)

Sprint 1 : (Inscription, authentification et gestion de profil).

- **Inscription :** cette interface pour effectuer l'inscription d'un non utilisateur de système pour qu'il puisse lui accéder, donc il doit remplir les champs indiqués et choisir obligatoirement son centre d'intérêt.

Par la suite un agent est affecté a cet utilisateur et créer un container pour cet agent.

NB : En prenant en consideration que notre système est distribué c'est-à-dire utiliser plusieurs containers (chaque agent dans un container).



The image shows a software window titled "INSCRIPTION". Inside the window, there is a light pink background. At the top center, there is a black icon of a person with a padlock. Below this icon, there are four input fields arranged vertically. The first field is labeled "Nom", the second "Prénom", the third "Mot de passe", and the fourth "Centre d'intérêt". The "Centre d'intérêt" field is a dropdown menu with "Informatique" selected. At the bottom right of the window, there is a yellow button labeled "OK".

Figure 11: interface de l'inscription.

Le code de ce cas d'utilisation est comme suit pour le cas d'informaticien :

```
// cas d'un informaticien
if (domaine.equals("Informatique")) {
    // on accepte au max 3 agents par communauté
    if (users.nombre_info < 2) {
        // ajouter dans la liste ds utilisateurs
        // users getUsers().add(user);
        AgentInfo info = new AgentInfo(user);
        // l'ajouter dans le vecteur informatique
        users.getInfo().add(info);

        try {
            Runtime rt = Runtime.instance();
            ProfileImpl pc = new ProfileImpl(false);
            pc.setParameter(ProfileImpl.MAIN_PORT, "localhost");
            AgentContainer ctr = rt.createAgentContainer(pc);
            AgentController agentController = ctr.createNewAgent(jTextField2.getText().toString(), "Domaine.AgentInfo",
                new Object[] { info });
            //Object[] args=((Agent) agentController).getArguments();

            agentController.start();
        } catch (ControllerException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        users.nombre_info++;
        new Connexion();
        this.setVisible(false);
        this.dispose();
    } else {
        JOptionPane.showMessageDialog(null, "Vous avez dépasser le nombre max!!!\n Utilisateur non créé");

        new Connexion();
        this.setVisible(false);
        this.dispose();
    }
}
```

Et pour le cas d'un mathématicien :

```
// cas de math
else {
    if (users.nombre_math < 2) {
        // ajouter dans la liste ds utilisateurs
        // users getUsers().add(user);

        // l'ajouter dans le vecteur mathematique

        Mathematique math = new Mathematique(user);
        users.getMath().add(math);

        try {
            Runtime rt = Runtime.instance();
            ProfileImpl pc = new ProfileImpl(false);
            pc.setParameter(ProfileImpl.MAIN_PORT, "localhost");
            AgentContainer ctr = rt.createAgentContainer(pc);
            AgentController agentController = ctr.createNewAgent(jTextField2.getText().toString(), "Domaine.Mathematique",
                new Object[] { math });
            //Object[] args=((Agent) agentController).getArguments();

            agentController.start();
        } catch (ControllerException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        users.nombre_math++;

        new Connexion();
        this.setVisible(false);
        this.dispose();
    } else {
        JOptionPane.showMessageDialog(null, "Vous avez dépasser le nombre max!!!\n Utilisateur non créé");

        new Connexion();
        this.setVisible(false);
        this.dispose();
    }
}
```

```
// créer l'utilisateur
if (jTextField2.getText().equals("") || jTextField1.getText().equals("")
    || jPasswordField1.getText().equals("")) {

    jLabel6.setText("Veuillez remplir tous les champs!!!");

} else {
    Utilisateur user = new Utilisateur(jTextField2.getText().toString(), jTextField1.getText().toString(),
        domaine, jPasswordField1.getText().toString(), new ArrayList<file>());
```

Figure 12: code pour le cas d'inscription.

- **Authentification :** Ce module constitue le point d'accès unique à l'application, tous les utilisateurs doivent s'inscrire pour pouvoir bénéficier de la totalité des services de cet application.



Figure 13: Interface de l'authentification.

```

@SuppressWarnings("serial")
public class Connexion extends javax.swing.JFrame {

    /**
     * Creates new form Connexion
     */
    public Connexion() {

        initComponents();
        this.setVisible(true);
    }

```

```

//pour authentification
@SuppressWarnings("deprecation")
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    int i=0;
    //parcourir tous les utilisateurs informaticiens
    if(!users.getInfo().isEmpty()) {
        while(i<users.getInfo().size())
        {
            System.out.println("user ----> "+users.getInfo().get(i).getUser().getNom());
            //vérification
            if(users.getInfo().get(i).getUser().getNom().equalsIgnoreCase(jTextField3.getText())
                &&(users.getInfo().get(i).getUser().getMot_de_passe().equals(jPasswordField1.getText())))
            {

                System.out.println("utilisateur trouvé ");
                Index.récupération( users.getInfo().get(i).getUser().getNom(),users.getInfo().get(i).getUser().getPrenom(),
                    "Informatique",users.getInfo().get(i).getUser().getMot_de_passe());
                Index.affecterAgentInfo(users.getInfo().get(i),i);
                new Index();
                this.setVisible(false);
                this.dispose();
                {
                    //System.out.println("ihhhhhhhh "+users.getUsers().get(i).getPrenom());

                }
                //this.setVisible(false);
            }
            i++;
        }
    }
}

```

```
//on click dans sing up
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new Inscription();
    this.setVisible(false);
}
}
```

Figure 14: code pour le cas d'authentification.

➤ Gestion de profil :

- 1- Afficher mon profil : Une fois authentifié, l'utilisateur accédera à l'ensemble des fonctionnalités de l'application.



Figure 15: Interface de l'affichage de profil (page d'accueil).


```

@SuppressWarnings("serial")
public class Index extends javax.swing.JFrame {
    // definie les deux types d'agents
    public static AgentInfo informaticien;
    public static Agent mathematicien;

    public static String nomAgent;
    public static String prenomAgent;
    public static String domaineAgent;
    public static String mtpAgent;
    public static int id;
    String nom = "";

    /**
     * Creates new form Index
     */
    public Index() {
        initComponents();
        this.setVisible(true);
    }
}

```

```

static void affecterAgentMath(Agent agent) {
    mathematicien = agent;
}

// ajouter l'utilisateur comme agent
// crée un container pour cet agent
static void affecterAgentInfo(AgentInfo agent, int id) {
    informaticien = agent;
    Index.id = id;
    System.out.println("taille " + users.getInfo().size());
    // System.out.println("Agent ID "+informaticien.getAID().getName());
}

```

```

static void récupération(String nomAgen, String prenomAgen, String domaineAgen, String mtpAgen) {


    nomAgent = nomAgen;
    prenomAgent = prenomAgen;
    domaineAgent = domaineAgen;
    mtpAgent = mtpAgen;
}

```

Figure 16: Code pour la page d'accueil.

- 2- Modifier mon profil : (par exemple je veux changer la communauté de mathématique à informatique) ou d'autre modification possible.

MODIFICATION



Nom

Prénom

Mot de passe


Centre d'intérêt

OK

Home

Bienvenu Benbaba Dans Votre Espace Personnel [Déconnexion](#)

[Accueil](#) [Boîte d'envoi](#) [Boîte de réception](#)



Informations de profil

Nom :

Prénom :

Centre d'intérêt :

[Modifier Profil](#)

```

public class Modification extends javax.swing.JFrame {
    static String nomAgent;
    static String prenomAgent;
    static String domaineAgent;
    static String mtpAgent;

    /**
     * Creates new form Inscription
     */
    public Modification() {
        initComponents();
        this.setVisible(true);
    }
}

```

```

static void Mrécuperation( String nomAgen,String prenomAgen, String domaineAgen,String mtpAgen) {

    nomAgent = nomAgen;
    prenomAgent=prenomAgen;
    domaineAgent=domaineAgen;
    mtpAgent=mtpAgen;
}

```

```

// modification de l'utilisateur
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Modification.Mrécuperation(nomAgent, prenomAgent, domaineAgent, mtpAgent);

    new Modification();
    this.dispose();
    this.setVisible(false);
}

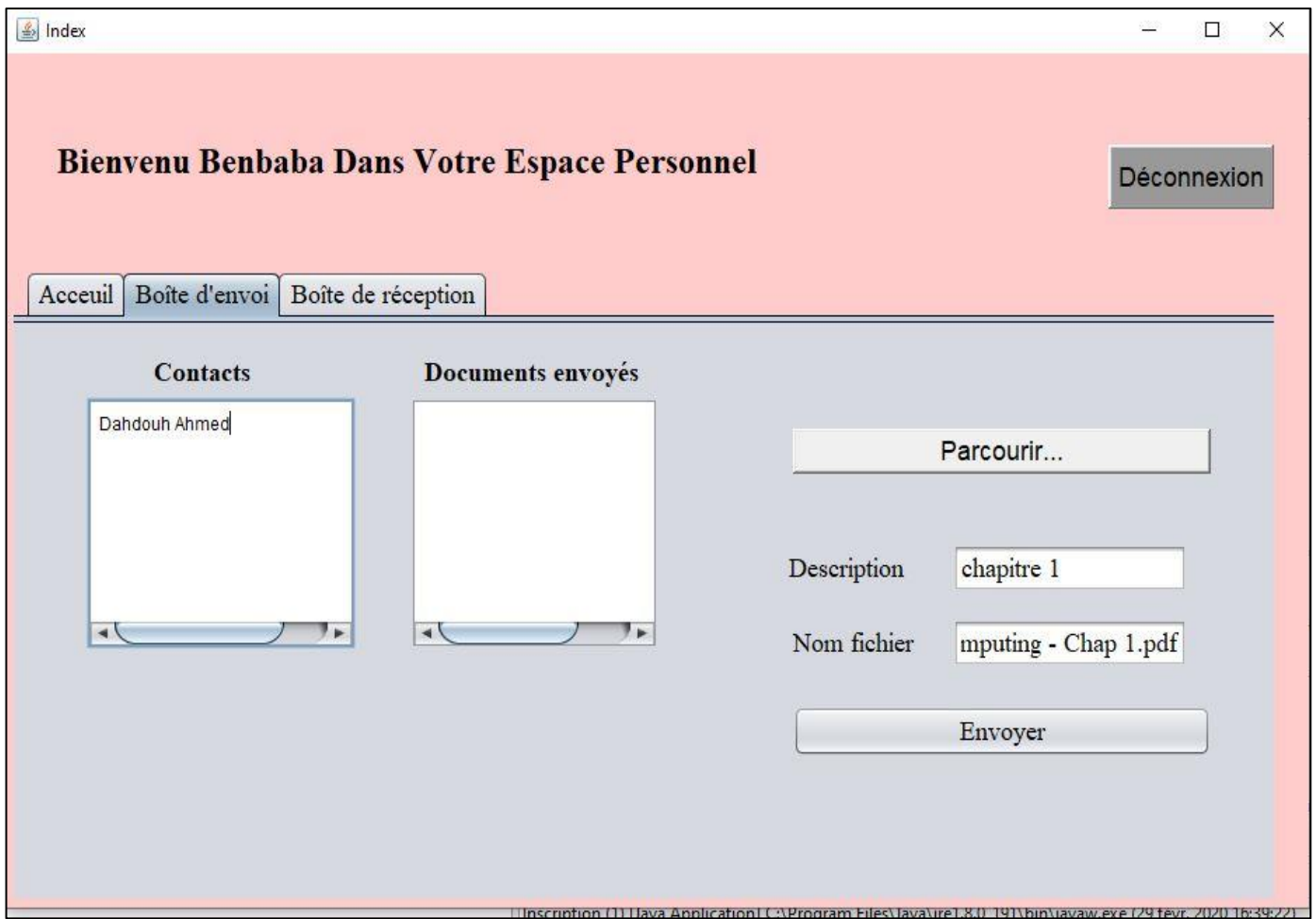
```

Figure 17: Code pour le cas ' modifier le profil'.

NB : les fonctions ci-dessous, les cas des erreurs et l'explication détaillée de l'implémentation est géré dans la section suivante.

Sprint 2 : (communication 'envoi et réception des documents').

➤ L'interface d'envoi :



1- La récupération des fichiers :

```
ACLMessage message = new ACLMessage(ACLMessage.INFORM);
if(Files.getF()!=(null)&&(Files.getDescription().equals(""))))
{
    System.out.println("file "+Files.getF());
    try {
        message.setContentObject(Files);
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    for (int i = 0; i < users.getInfo().size(); i++) {
        if (!users.getInfo().get(i).getUser().getNom().equals(myAgent.getAID().getName()))
            nom = users.getInfo().get(i).getUser().getNom();
        //System.out.println("cccccc "+nom);
    }
    message.addReceiver(new AID(nom, AID.ISLOCALNAME));
    send(message);
    Files.setF(null);
}
```

Dans la classe Index (la récupération) :

```
private void Button2ActionPerformed(ActionEvent evt) {
    //System.out.println("index taille " + AgentInfo.getMesAgents().size());
    ACLMessage message = new ACLMessage(ACLMessage.INFORM);
    /* init du filechooser */
    JFileChooser fc = new JFileChooser();
    file file = new file();
    /* affichage du dialog et test si le bouton ok est pressé */
    if (fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
        // récupération de fichier
        if (fc.getSelectedFile() != null) {
            String filename = fc.getSelectedFile().getName();
            jTextField2.setText(filename);
            FileInputStream fis;
            byte[] bytes = null;
            try {
                fis = new FileInputStream(fc.getSelectedFile());
                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                byte[] buf = new byte[1024];
                for (int readNum; (readNum = fis.read(buf)) != -1;) {
                    bos.write(buf, 0, readNum);
                }
                bytes = bos.toByteArray();
            } catch (FileNotFoundException ex) {
                // Logger.getLogger(Index.class.getName()).log(Level.SEVERE, null, ex);
            } catch (IOException ex) {
                // Logger.getLogger(Index.class.getName()).log(Level.SEVERE, null, ex);
                ex.printStackTrace();
            }
            file.setF(bytes);
            file.setFileName(filename);
            String desc = jTextField3.getText().toString();
            file.setDescription(desc);
            String a=jTextArea2.getText().toString();
            jTextArea2.setText(a+"\n"+file.getFileName());
        }
    }
}
```

```
        }
        bytes = bos.toByteArray();
    } catch (FileNotFoundException ex) {
        // Logger.getLogger(Index.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        // Logger.getLogger(Index.class.getName()).log(Level.SEVERE, null, ex);
        ex.printStackTrace();
    }
    file.setF(bytes);
    file.setFileName(filename);
    String desc = jTextField3.getText().toString();
    file.setDescription(desc);
    String a=jTextArea2.getText().toString();
    jTextArea2.setText(a+"\n"+file.getFileName());
}

try {
    // recupere le message
    message.setContentObject(file);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
AgentInfo.setFiles(file);
AgentInfo.set...
```

2- Lors la réception de message :

```
System.out.println("Je suis Informaticien : " + this.getAID().getName());
// comportement
addBehaviour(new CyclicBehaviour() {

    @Override
    public void action() {

        ACLMessage messageRecu = receive();
        if (messageRecu != null) {
            ArrayList<file> FilesListe = new ArrayList<file>();
            file f;
            try {
                f = (file) messageRecu.getContentObject();
                FilesListe.add(f);
                System.out.println("nom de fichier " + f.getFileName());
                user.getMessagerecu().add(f);
            } catch (UnreadableException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        else {
            block();
        }
    }
});
```

Figure 18: Code pour l'envoi et la réception des messages.

Implémentation :

Nous présentons, dans cette section les principales étapes de notre réalisation. Cette implémentation traduit le passage du modèle conceptuel décrit dans la partie précédente vers un produit informatique.

Pour l'implémentation de notre logiciel, il est tout naturel d'utiliser la plateforme JADE. La mise en œuvre des différentes entités vues dans la phase de conception se fait en utilisant les outils correspondants dans la cette plateforme. Les signaux de commande sont représentés par des messages et les comportements agents sont implémentés en utilisant du code JADE. Le système ainsi formé constitue le système Multi-Agents dont la dynamique de fonctionnement simule le processus industriel étudié.

- Pour notre cas le logiciel est un ensemble de classes (agents) écrites en java et exécutées sous la plateforme JADE.

I. Environnement et Plateforme de travail :

Type de plateforme	Nom de plateforme	Description
Equipement matériel	Ordinateurs	- Intel core i5-5200U. - RAM 4.00Go. - Disque dur 200Go SSD.
Système d'exploitation	Windows	Windows 10 professionnel
Environnement d'exécution	Plateforme JAVA	/
Plateforme Multi agent	JADE	Implémenté en java
Logiciel pour les diagrammes	StarUml	/
Environnement utilisé pour le développement	Eclipse	IDE pour java

Tableau 5: L'environnement logiciel et matériel de mise en œuvre de cette application.

II. Présentation de la plateforme JADE :

JADE est un middleware qui facilite le développement des systèmes multi agents (SMA).

JADE contient :

- **Un runtime Environnement** : l'environnement où les agents peuvent vivre. Ce runtime environnement doit être activé pour pouvoir lancer les agents.
- **Une librairie de classes** : que les développeurs utilisent pour écrire leurs agents.

- **Une suite d'outils graphiques** : qui facilitent la gestion et la supervision de la plateforme des agents.

Chaque instance du JADE est appelée conteneur " container ", et peut contenir plusieurs agents. Un ensemble de conteneurs constituent une plateforme. Chaque plateforme doit contenir un conteneur spécial appelé main-conteneur et tous les autres conteneurs s'enregistrent auprès de celui-là dès leur lancement.

La figure suivante illustre les concepts de base du jade en montrant un petit exemple dans notre cas (mini projet) de deux plateformes jade composées respectivement de trois et un conteneur. Chaque agent est identifié par un identifiant unique et peut communiquer avec n'importe quel autre agent sans avoir besoin de connaître son emplacement :

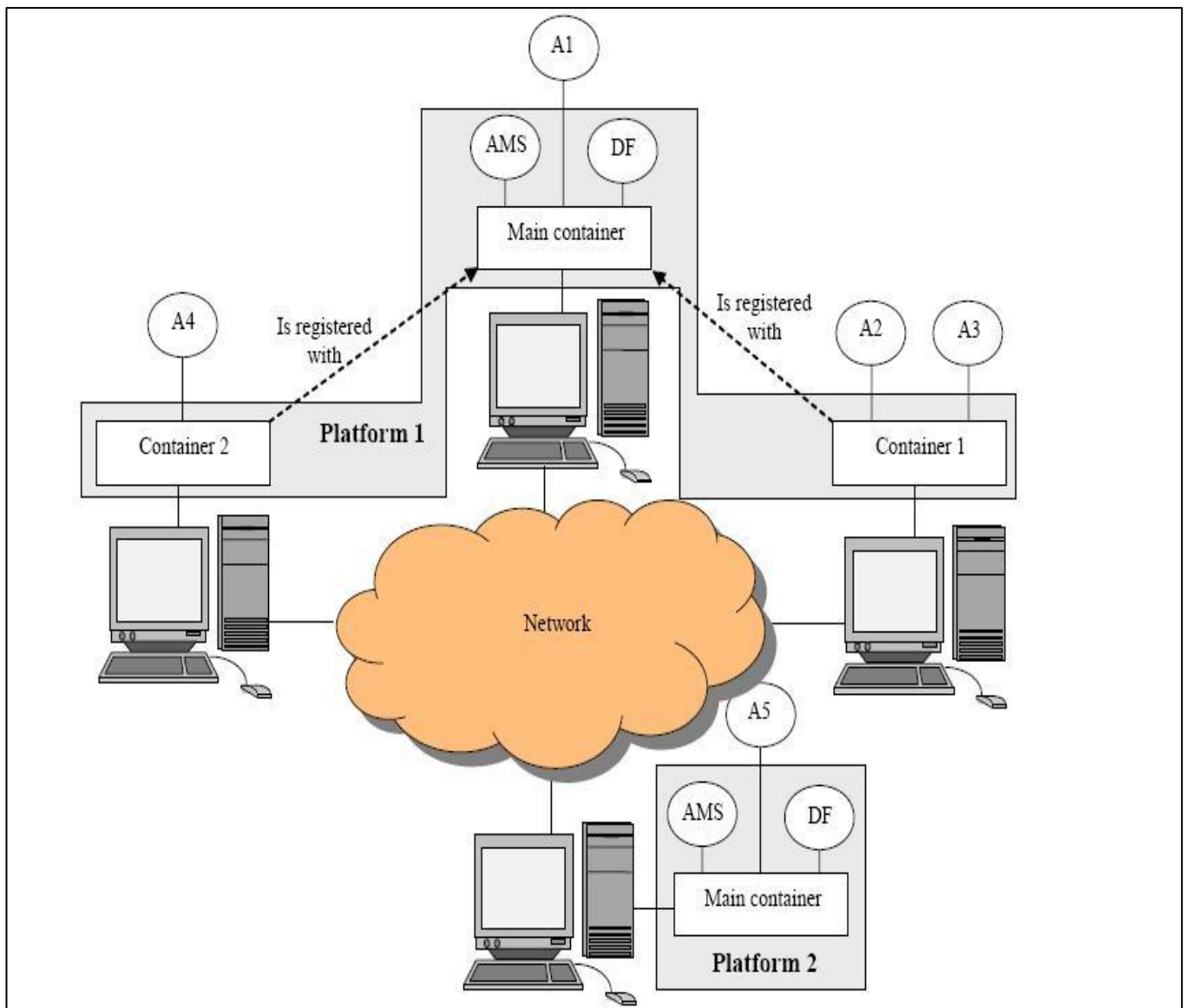


Figure 19: la répartition de la plateforme JADE en conteneurs.

III. Structure Logique du Système :

La figure ci-dessus représente l'architecture modulaire d'un agent de notre application sous jade :

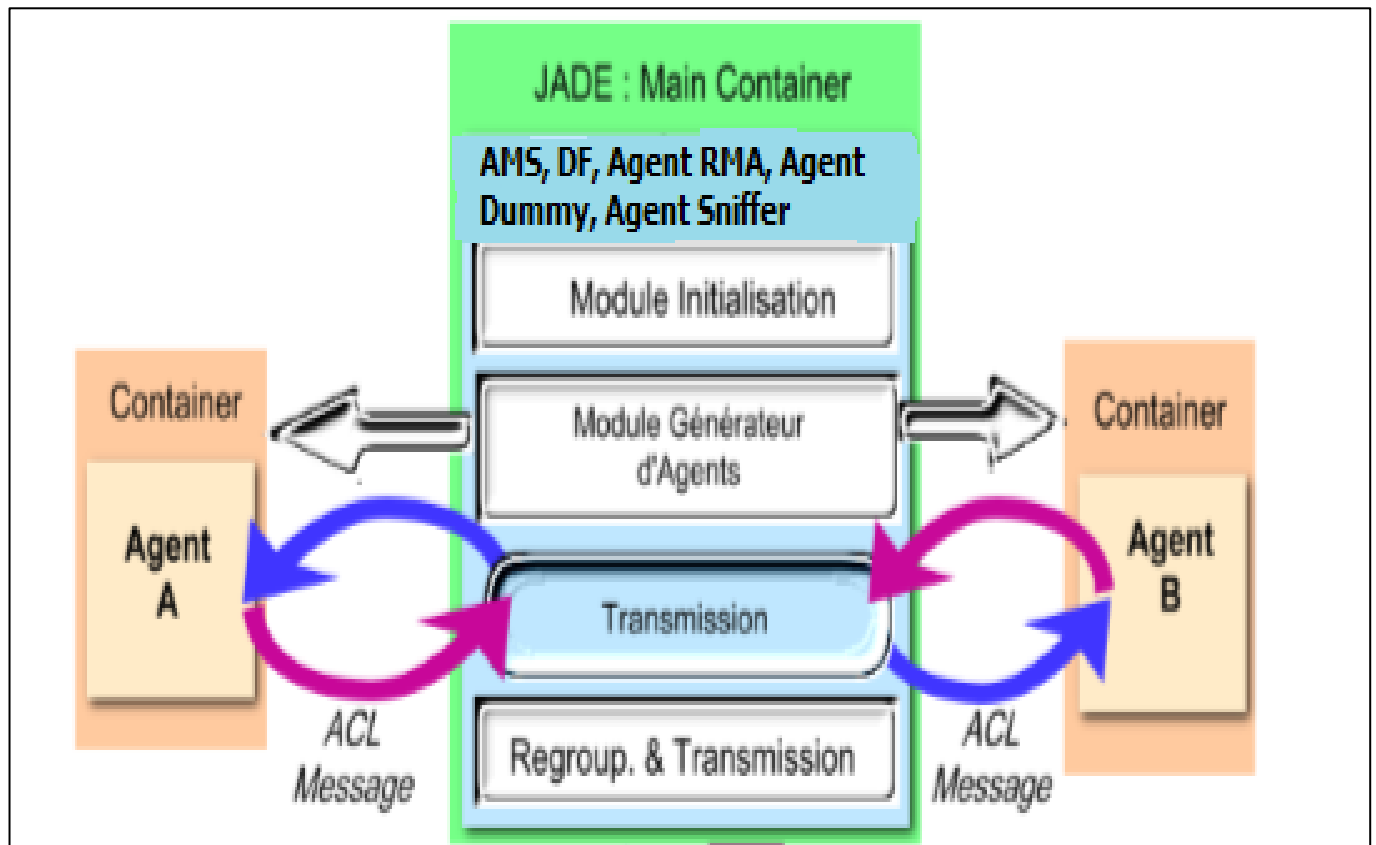


Figure 20: : Architecture modulaire des agents sous JADE

IV. Diagramme de classe :

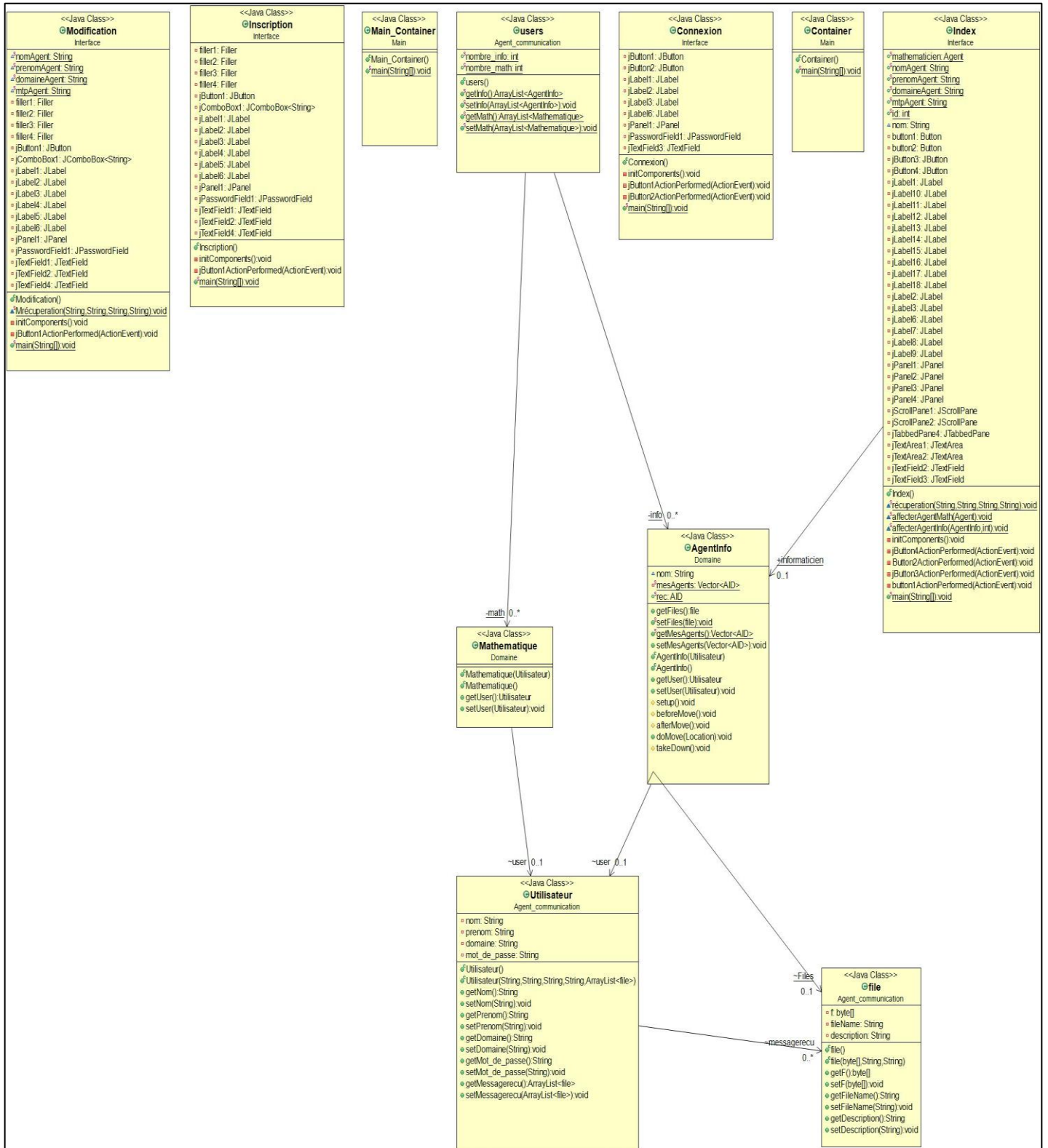


Figure 21: Diagramme de classe.

V. Mise en Œuvre du projet :

Le processus d'implémentation du projet est centré sur le concept clé qui réside sur l'implémentation de la plateforme JADE (création de conteneurs, création d'agents, gestion des comportements « behaviour »).

L'implémentation du système sur la plateforme JADE comprend à son tour l'appel aux concepts suivants :

- **La création du conteneur principale :** Ce module permet de faire appel au Conteneur principal de la plateforme JADE, il est nécessaire si on décide de lancer le mode multi-agents de notre système. C'est grâce à ce conteneur principal de la plateforme que seront créés les autres conteneurs qui joueront le rôle de sites pour héberger nos futurs agents.

Le conteneur principal de la plateforme Jade contient à son tour des agents prédisposés pour remplir des tâches nécessaires au fonctionnement de la plateforme elle-même et aussi aux systèmes conçus sur sa base.

```
1 package Main;
2
3 import jade.core.ProfileImpl;
4
5
6
7
8
9 public class Main_Container {
10
11     public static void main(String[] args) {
12         // TODO Auto-generated method stub
13
14         try {
15             Runtime rt=Runtime.instance();
16             Properties p= new Properties ();
17             p.setProperty("gui", "true");
18             // java jade.Boot -port 1097 -gui
19             //p.setProperty(Profile.MAIN_PORT, "1096");
20             ProfileImpl pc=new ProfileImpl(p);
21             //pc.setParameter(ProfileImpl.MAIN_PORT, "localhost");
22
23             AgentContainer ctr= rt.createMainContainer(pc);
24             //System.out.print("koko\n\n");
25             ctr.start();
26         } catch (ControllerException e) {
27             // TODO Auto-generated catch block
28             e.printStackTrace();
29         }
30     }
31 }
32 }
33 }
```

Figure 22: création de main container.

- **La création des conteneurs secondaires :** Ce module permet de créer un Conteneur non principal dans la plateforme JADE, il est nécessaire pour abriter et servir de milieu

d'exécution pour les agents de notre système. Un conteneur peut héberger un ou plusieurs agents à la fois, il peut représenter un site local (mode localhost) de la même façon qu'il peut représenter un site distant (en donne l'adresse IP), il est aussi nécessaire si on choisit le mode multi-agents de notre système. Dans notre cas c'est le cas local et chaque création est dans un container appart.

```
try {
    Runtime rt = Runtime.instance();
    ProfileImpl pc = new ProfileImpl(false);
    pc.setParameter(ProfileImpl.MAIN_PORT, "localhost");
    AgentContainer ctr = rt.createAgentContainer(pc);
    AgentController agentController = ctr.createNewAgent(jTextField2.getText().toString(), "Domaine.AgentInfo",
        new Object[] { info });
    //Object[] args=((Agent) agentController).getArguments();

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

Figure 23: Création des containers secondaires.

- **Les comportements « behaviours »** : Un comportement ou Behaviour est une tâche que l'agent doit effectuer, on peut la lui attribuer grâce à l'instruction :

addBehaviour(Behaviour b) ;

Il est composé de 4 méthodes :

- onStart () ;
- Action () ; elle désigne l'opération effectuée
- Done () ; elle retourne true si action () est terminée
- onEnd () ;

⇒ Les types de Behaviour :

- ❖ **One-shot Behaviour** : Classe jade.core.Behaviours.OneShotBehaviour Il s'exécute une seule fois puis se termine, elle implémente la méthode done() qui va toujours retourner true.
- ❖ **Cyclic Behaviour** : Classe jade.core.Behaviours.CyclicBehaviour Il s'exécute de manière répétitive, elle implémente la méthode done() qui va toujours retourner false.
- ❖ **Generic Behaviour** : Classe jade.core.Behaviours. Behaviour Il vient entre les deux premiers, il accepte la personnalisation, il laisse le choix de terminaison au programmeur.

- ❖ **Waker Behaviour** : Classe jade.core.Behaviours.WakerBehaviour Il s'exécute grâce à la méthode onWake() dans un délai ou une date en argument ;(comme un réveil).
- ❖ **Ticker Behaviour** : Classe jade. core.Behaviours.TickerBehaviour Il s'exécute grâce à la méthode onTick() dans un laps de temps répétitif ;(comme un timer).

- La Communication entre Agents :

Messages ACL :

ACL est une norme de transmission adaptée pour les SMA, mise au point par la FIPA-ACL (une Organisation pour la recherche des SMA). Package jade.lang.acl.

Envoyer un message :

```
...
ACLMessage message1=new ACLMessage(ACLMessage.INFORM) ;
message1.addReceiver(new AID(« vendeur1 »,AID.ISLOCALNAME)) ;
message1.setContent(« livre XML ») ; // message1.setObject(monObjet) ; pour envoyer
des objets
send(message1) ;
...
```

Recevoir et lire un message :

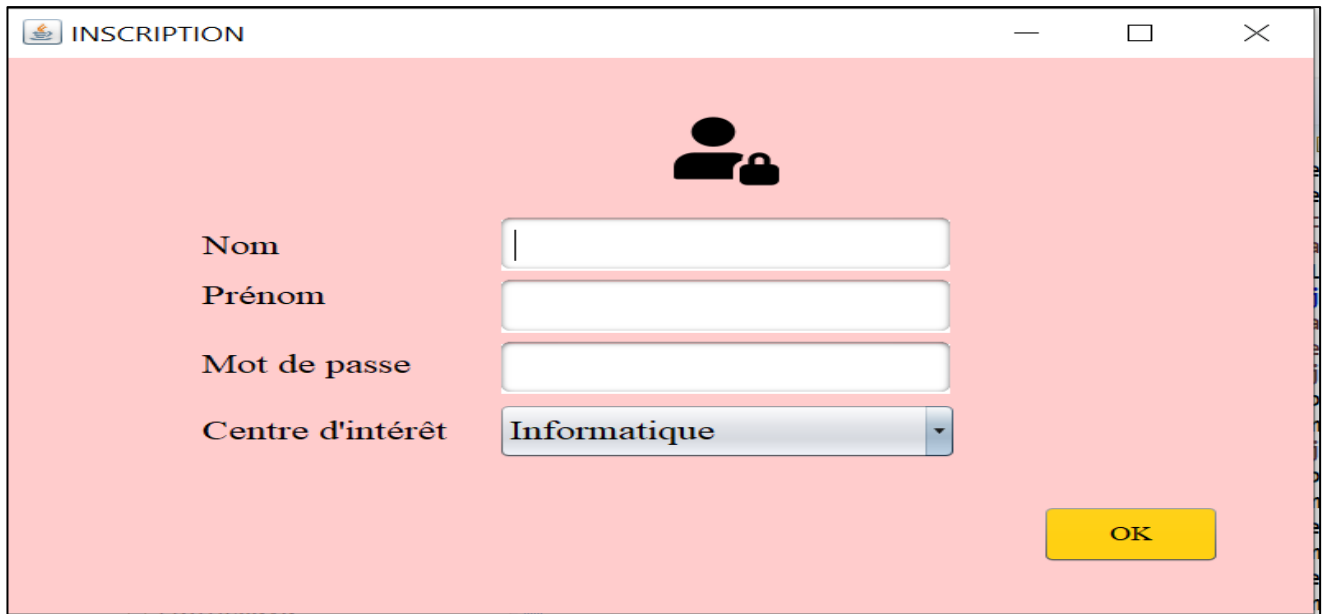
Créer un behaviour pour lire les messages :

```
ACLMessage message2=receive() ;
If (message2 !=null){
System.out.println(“message reçu :“+message2.getContent) ; }
else {
Block(); // arrêter le behaviour de lecture }
```

VI. Capture d'écran :

Nous présentons dans cette section les plus importantes interfaces qui illustrent les différents cas d'utilisation déjà vus dans la partie précédente et l'enchaînement avec la plateforme jade.

❖ Inscription :



The screenshot shows a window titled "INSCRIPTION" with a light pink background. At the top center is an icon of a person with a padlock. Below it are four labels on the left: "Nom", "Prénom", "Mot de passe", and "Centre d'intérêt". To the right of each label is an input field. The "Centre d'intérêt" field is a dropdown menu showing "Informatique". At the bottom right is a yellow "OK" button.

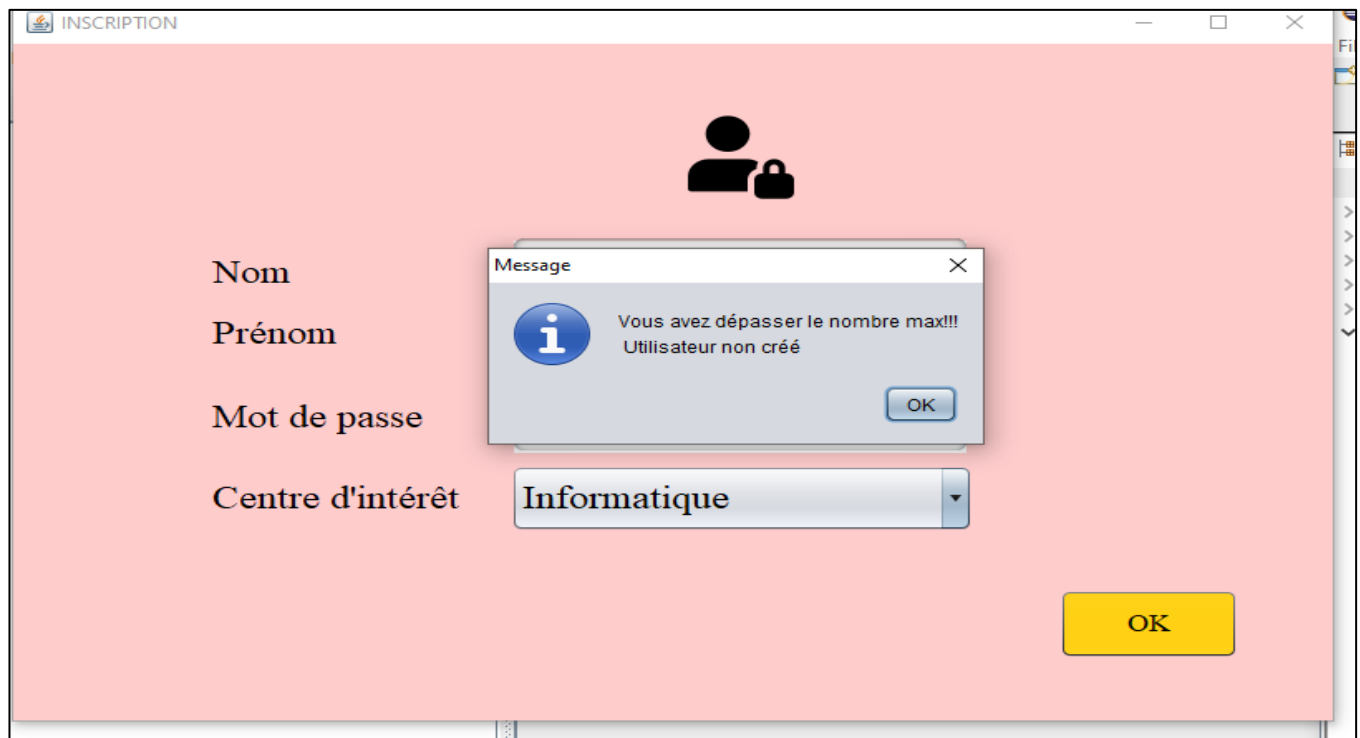
La gestion des erreurs : par exemple un champ n'est pas rempli.



The screenshot shows the same "INSCRIPTION" window. The input fields for "Nom" and "Prénom" now contain the text "nom" and "prénom" respectively. The "Mot de passe" field is still empty. The "Centre d'intérêt" dropdown still shows "Informatique". At the bottom, a red error message "Veuillez remplir tous les champs!!!" is displayed. The yellow "OK" button remains at the bottom right.

Figure 24: Gestion des erreurs pour le cas d'inscription.

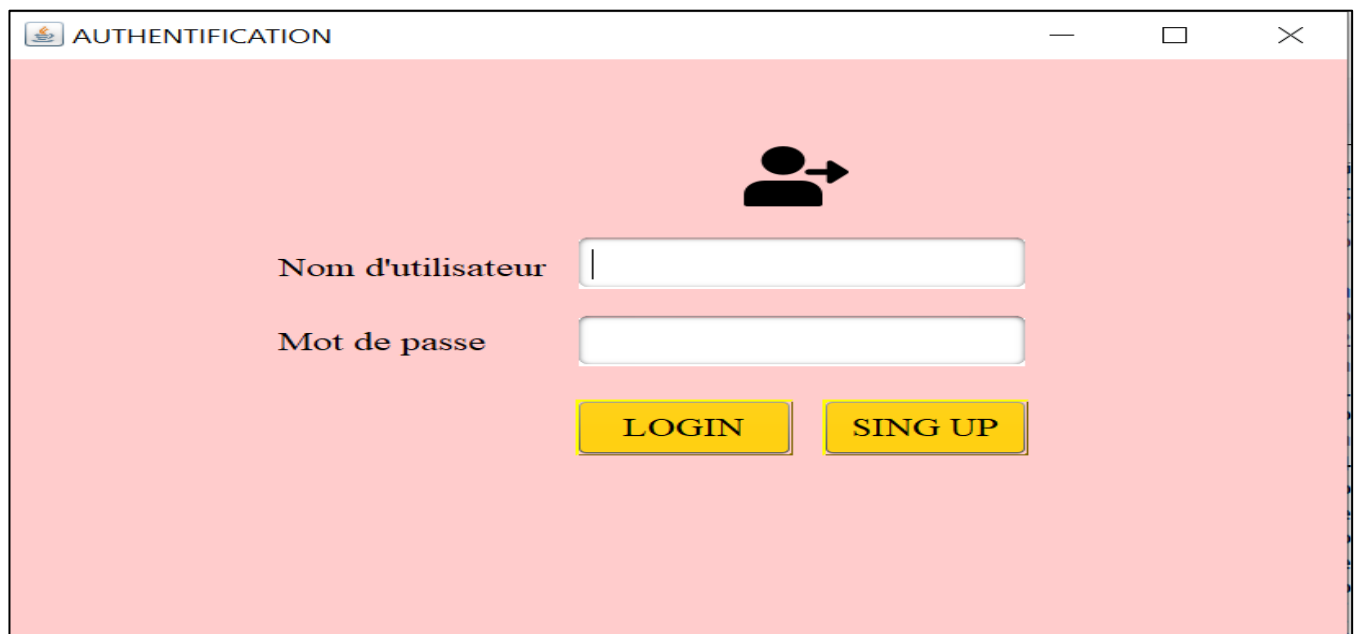
Ou bien dépasser le nombre de création définie.



The screenshot shows a window titled "INSCRIPTION" with a pink background. At the top center is an icon of a person with a lock. Below it are four labels: "Nom", "Prénom", "Mot de passe", and "Centre d'intérêt". The "Centre d'intérêt" label is followed by a dropdown menu showing "Informatique". A yellow "OK" button is at the bottom right. A modal dialog box titled "Message" is centered over the form. It contains an information icon (i in a blue circle) and the text "Vous avez dépasser le nombre max!!! Utilisateur non créé". An "OK" button is at the bottom right of the dialog.

Figure 25: gestion d'erreur si la création dépasse 2 utilisateurs.

❖ Authentification :



The screenshot shows a window titled "AUTHENTIFICATION" with a pink background. At the top center is an icon of a person with an arrow pointing right. Below it are two labels: "Nom d'utilisateur" and "Mot de passe". Each label is followed by a white text input field. Below the input fields are two yellow buttons: "LOGIN" and "SIGN UP".

La gestion des erreur : nom d'utilisateur ou mot de passe incorrecte.



The screenshot shows a window titled "AUTHENTIFICATION" with a light red background. At the top center is an icon of a person with an arrow pointing right. Below this are two input fields: "Nom d'utilisateur" containing the text "nom" and "Mot de passe" containing "****". There are two yellow buttons labeled "LOGIN" and "SING UP". At the bottom, a red error message reads "Utilisateur non trouvé !!!".

Figure 26: La gestion d'erreur pour le cas d'authentification.

Page d'accueil :



The screenshot shows a window titled "Home" with a light red background. At the top left, it says "Bienvenu Benbaba Dans Votre Espace Personnel". At the top right is a button labeled "Déconnexion". Below this are three tabs: "Accueil", "Boîte d'envoi", and "Boîte de réception". The "Accueil" tab is active. On the left is a cartoon illustration of a man with orange hair, wearing a blue shirt and dark pants, with a yellow speech bubble saying "HI!". On the right is a section titled "Informations de profil" with three input fields: "Nom : Benbaba", "Prénom : Rym", and "Centre d'intérêt : Mathématique". Below these fields is a button labeled "Modifier Profil".

Figure 27: la page d'accueil.

L'exécution de la création d'agent sous Eclipse.

```
févr. 29, 2020 10:33:29 PM jade.core.Runtime beginContainer
INFOS: -----
This is JADE snapshot - revision $WCREV$ of $WCDATE$
downloaded in Open Source, under LGPL restrictions,
at http://jade.tilab.com/
-----
févr. 29, 2020 10:33:30 PM jade.core.BaseService init
INFOS: Service jade.core.management.AgentManagement initialized
févr. 29, 2020 10:33:30 PM jade.core.BaseService init
INFOS: Service jade.core.messaging.Messaging initialized
févr. 29, 2020 10:33:30 PM jade.core.BaseService init
INFOS: Service jade.core.mobility.AgentMobility initialized
févr. 29, 2020 10:33:30 PM jade.core.BaseService init
INFOS: Service jade.core.event.Notification initialized
févr. 29, 2020 10:33:30 PM jade.core.messaging.MessagingService clearCachedSlice
INFOS: Clearing cache
févr. 29, 2020 10:33:30 PM jade.core.AgentContainerImpl joinPlatform
INFOS: -----
Agent container Container-1@Rym-PC is ready.
-----
Je suis Informaticien : Benbaba@Rym-PC:1099/JADE
element ( agent-identifiant :name Benbaba@Rym-PC:1099/JADE :addresses (sequence http://Rym-PC:7778/acc ))
```

Figure 28: créations d'user coté Eclipse

La connexion de ces agent sous jade :

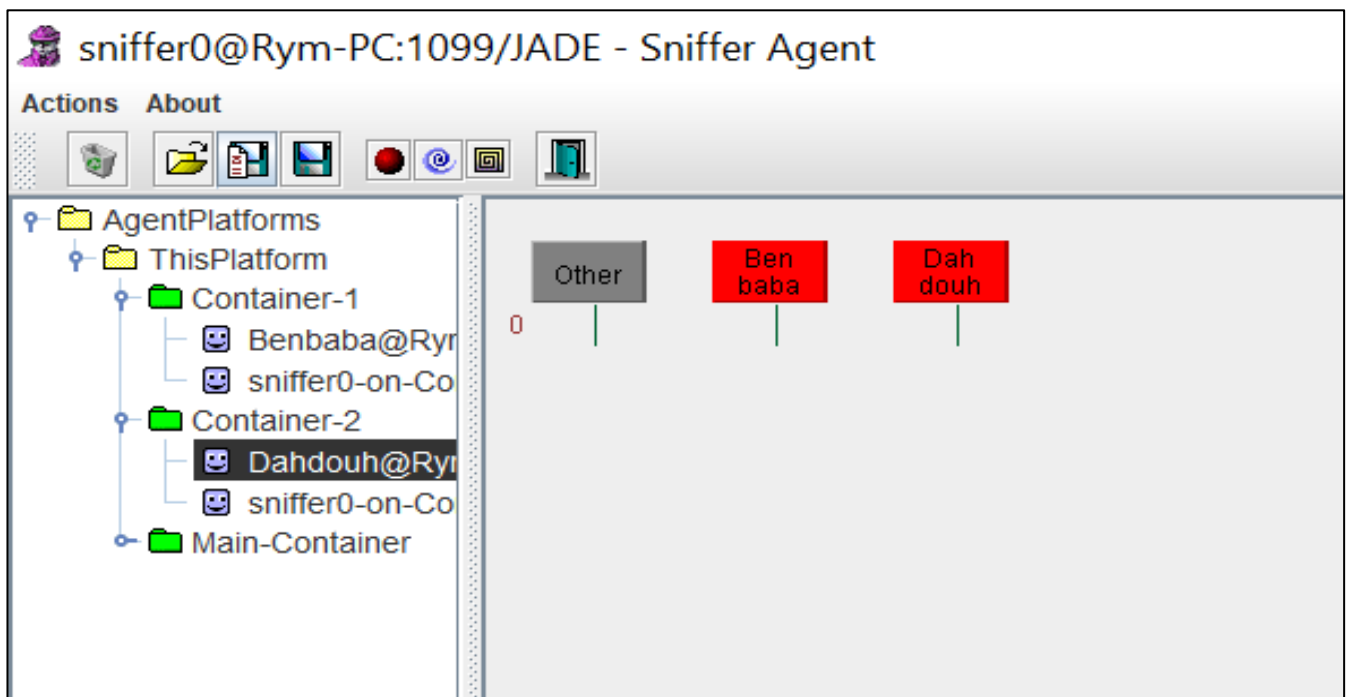


Figure 29: la représentation des agents sous la plateforme jade.

❖ L'envoi de message :

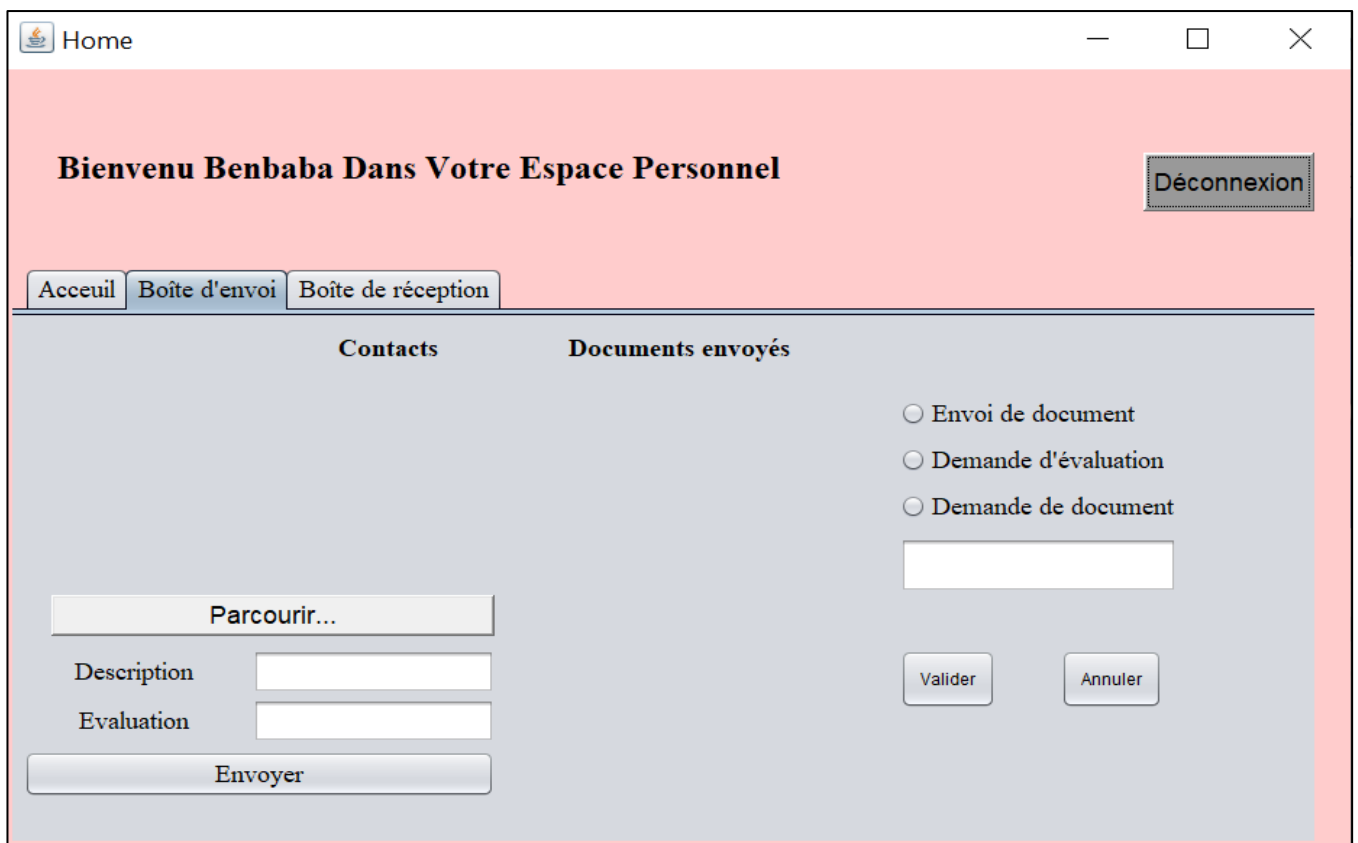


Figure 30: interface de l'envoi.

❖ Sous l'interface Jade (avec l'agent sniffer)

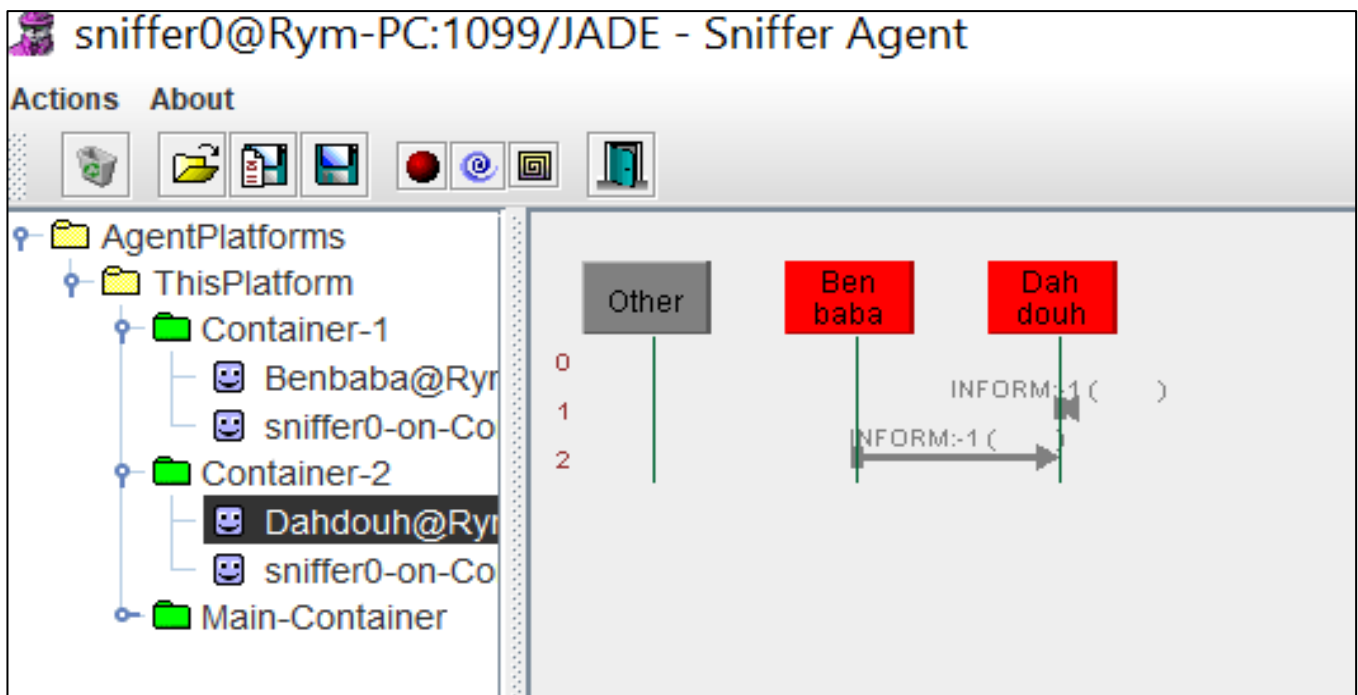


Figure 31: l'envoi de message sous jade.

Le contenu de message envoyé :

The screenshot shows a dialog box titled "ACL Message" with a close button (X) in the top right corner. It has two tabs: "ACLMessage" (selected) and "Envelope".

Fields and values:

- Sender:** View button, text: enbaba@Rym-PC:1099/JADE
- Receivers:** Dahdouh@Rym-PC:1099/JADE
- Reply-to:** (empty text field)
- Communicative act:** inform (dropdown menu)
- Content:** A text area containing:
;4H□N
□=F/Á□AgentsExemple/Agents/build/classes/agents/Neg
tsExemple/Agents/build/classes/agents/Agents.classPK
**@□\$□t&AgentsExemple-20200224T192003Z-001.zip
- Language:** (empty text field)
- Encoding:** (empty text field)
- Ontology:** (empty text field)
- Protocol:** Null (dropdown menu)
- Conversation-id:** (empty text field)
- In-reply-to:** (empty text field)
- Reply-with:** (empty text field)
- Reply-by:** View button, (empty text field)
- User Properties:** (empty text field)

An "OK" button is located at the bottom center of the dialog box.

Figure 32: Contenu de message envoyé.

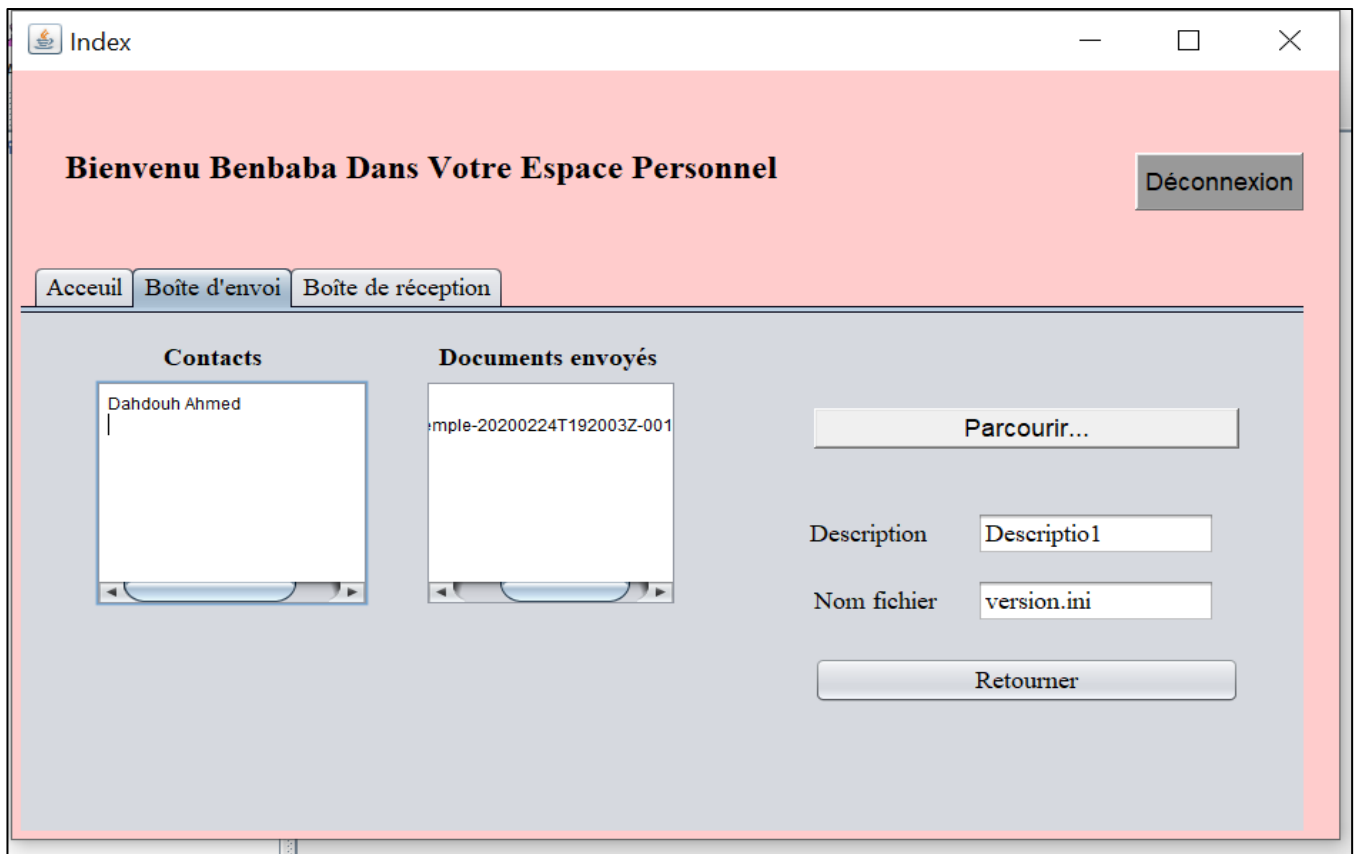


Figure 33: Interface qui contient le fichier envoyé.

Pour le bloc de réception :

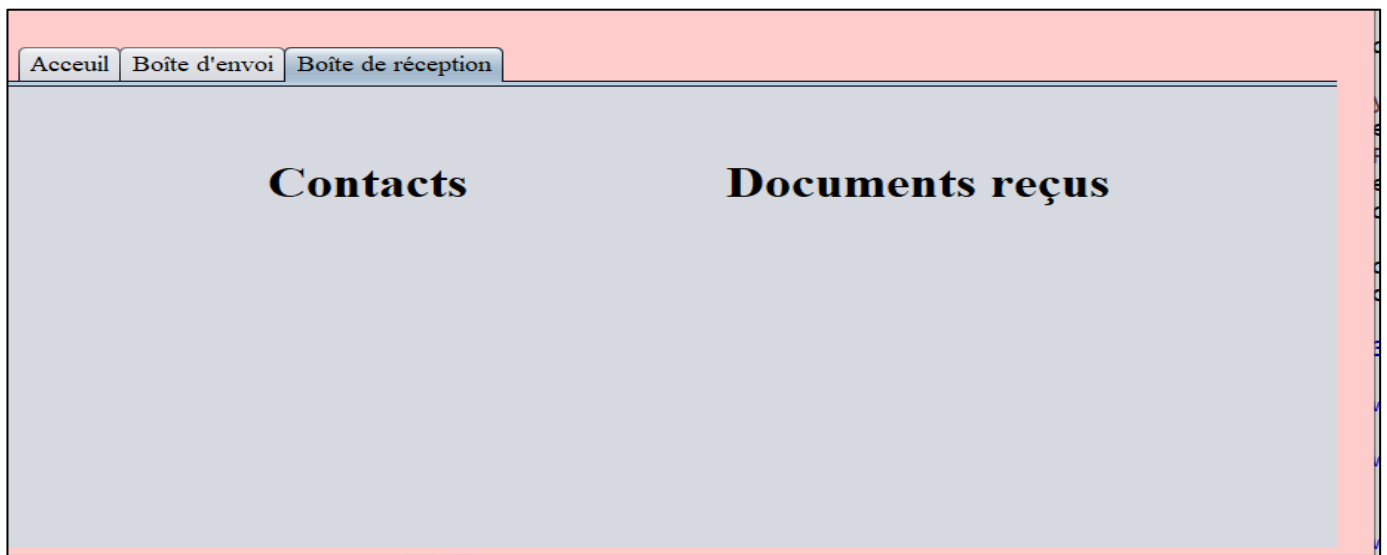


Figure 34: l'interface de réception de document.

Conclusion :

La motivation principale pour un membre à joindre un réseau social est la facilité de créer un profil, d'utiliser les différentes applications offertes par le service ainsi que la possibilité de partager facilement des informations avec des contacts sélectionnés ou le public et cela à des fins personnelles ou professionnelles.

Le SMA a toujours été un médiateur qui facilite les processus de programmation, et surtout qu'il est conforme à l'utilisateur actuel qui est toujours voulu d'accélérer et de préciser les opérations sur tout avec la propriété de mobilité qui créer un espace virtuel commun pour permettre de migrer à l'autre machine sans déplacement réel.

Des perspectives d'avenir' qui sera lancé à partir de ce travail pour se rendre à d'autres développements dans une vision plus complète, c'est d'identifier le leader d'opinion de chaque communauté.

Références :

- [1] YSDJmemoirePADMARAD2016.pdf.
- [2] GESTION DES PROCESSUS METIER OU BPM (BUSINESS PROCESS MANAGEMENT) : Livre blanc BlueWay .
- [3] Les Systèmes Multi-Agents : Université Larbi Ben M'hidi – Oum El Bouaghi.