

CLASSES PREPARATOIRES INTEGREES (CPI)

2^{IEME} ANNEE

PROJET N° 30

Développement d'un client de messagerie électronique

EQUIPE DE PROJET :

- Benammar nassima
- Adjissa Ikram
- Bouamoucha nahla
- haddouche yasmine
- Korteby Fella
- Mechalikh amel

ENCADREE PAR :

- Mme Yessad Lamia

CLIENT :

- Mme Yessad Lamia

Remerciements

Il nous est spécialement agréable, d'exprimer toute notre reconnaissance envers les personnes qui de près ou de loin nous ont apporté leur soutien dans la réalisation de ce projet.

On tient à remercier notre encadrant **Mme Yessad Lamia**, son aide, ses conseils précieux, ses critiques constructives, ses explications et suggestions pertinentes nous était d'une grande aide pour réaliser notre application.

Sommaire

I.	Introduction générale.....	5
II.	Chapitre I : Etude théorique	
1.	Introduction	8
2.	Service de messagerie électronique.....	8
2.1	Définition.....	8
2.2	Différents agents d'un service de messagerie électronique.....	9
3.	Client de messagerie électronique.....	10
3.1	Définition.....	10
3.2	Avantages.....	10
4.	Structure et format d'un message électronique.....	10
4.1	Structure des messages.....	10
4.2	Formats standards des messages.....	12
4.3	Les principaux types MIME.....	13
5.	Protocoles de la messagerie électronique.....	13
5.1	Le protocole Simple Mail Transfer Protocol (SMTP)	13
5.2	Le protocole Post Office Protocol (POP)	14
5.3	Le protocole Internet Message Access Protocol (IMAP)	15
6.	Acheminement d'un message électronique.....	15
7.	Quelques clients de messagerie électronique.....	17
7.1	Microsoft Outlook.....	17
7.2	Mozilla Thunderbird.....	17
7.3	Comparaison entre Outlook et Thunderbird.....	18
8.	Conclusion.....	18
III.	Chapitre II : Etude conceptuelle	
1.	Introduction	20
2.	Architecture du système.....	20
3.	Diagrammes de cas d'utilisation.....	21
3.1	Diagramme général de cas d'utilisation.....	22

3.2 Diagrammes détaillés de cas d'utilisation.....	22
4. Identification des classes.....	24
4.1 Présentation des classes.....	24
4.2 Description des classes.....	27
5. Conclusion	29
 IV. Chapitre III : Etude pratique (Réalisation)	
1. Introduction.....	31
2. Environnement de travail.....	31
2.1 Environnement matériel.....	31
2.2 Environnement logiciel.....	31
3. Présentation des bibliothèques utilisées	32
4. Réalisation des fonctions de l'envoi et de la réception.....	33
5. Présentation de l'interface graphique	36
6. Conclusion.....	39
 V. Conclusion générale.....	40
 VI. Webographie.....	41

Table des figures

1. Diagramme de Gantt.....	6
2. Lancement de telnet.....	13
3. Connexion au serveur SMTP.....	14
4. Connexion au serveur POP.....	15
5. Schéma représentant l'acheminement d'un message électronique.....	16
6. Microsoft Outlook.....	17
7. Mozilla Thunderbird.....	17
8. Schéma général du système.....	20
9. Diagramme général de cas d'utilisation.....	22
10. Gestion des emails reçus.....	22
11. Gestion de l'envoi des emails	23
12. Gestion du carnet d'adresses.....	23
13. Fenêtre d'authentification.....	36
14. Fenêtre principale (l'accueil).....	37
15. Fenêtre de l'envoi d'un message (avec un éditeur HTML).....	37
16. Fenêtre du carnet d'adresses.....	38
17. Fenêtre d'ajout d'un contact.....	39

Liste des tableaux

1. Comparaison entre Outlook et Thunderbird.....	18
--	----

Introduction Générale

Il ne fait désormais plus aucun doute que les technologies de l'information et de la communication représentent la révolution la plus importante et la plus innovante qui a marqué la vie de l'humanité en ce siècle passé. En effet, loin d'être un éphémère phénomène de mode, ou une tendance passagère, ces technologies viennent apporter de multiples confort à notre mode de vie, car ils ont révolutionné le travail des individus par leur capacité de traitement d'information, d'une part, et de rapprochement des dimensions espace/temps, d'une autre.

Parmi ces TIC (Technologies de l'information et de la communication), la messagerie électronique est rapidement développée dans les organisations aux cours de ces dix dernières années, par sa facilité d'utilisation et son utilité perçue. Désormais, elle représente l'outil de travail le plus utilisé.

C'est dans ce cadre que s'intègre ce projet qui consiste dans l'étude, la conception et le développement d'un client de messagerie électronique destiné au public. La messagerie électronique est un des moyens de communication les plus convoités cette dernière décennie. Elle permet d'établir la communication différée entre personnes géographiquement éloignées en échangeant des messages textuels, graphiques ou sonores. Ce moyen est souvent utilisé sur Internet grâce à son intégration dans différents sites. Sans passer par un navigateur Web, le client de messagerie électronique permet de se connecter à n'importe quel serveur de messagerie électronique pour envoyer ou lire des messages.

L'objectif de ce projet est de développer un client de messagerie destiné au grand public ou aux professionnels. Le logiciel construit doit assurer les deux fonctions les plus importantes, à savoir :

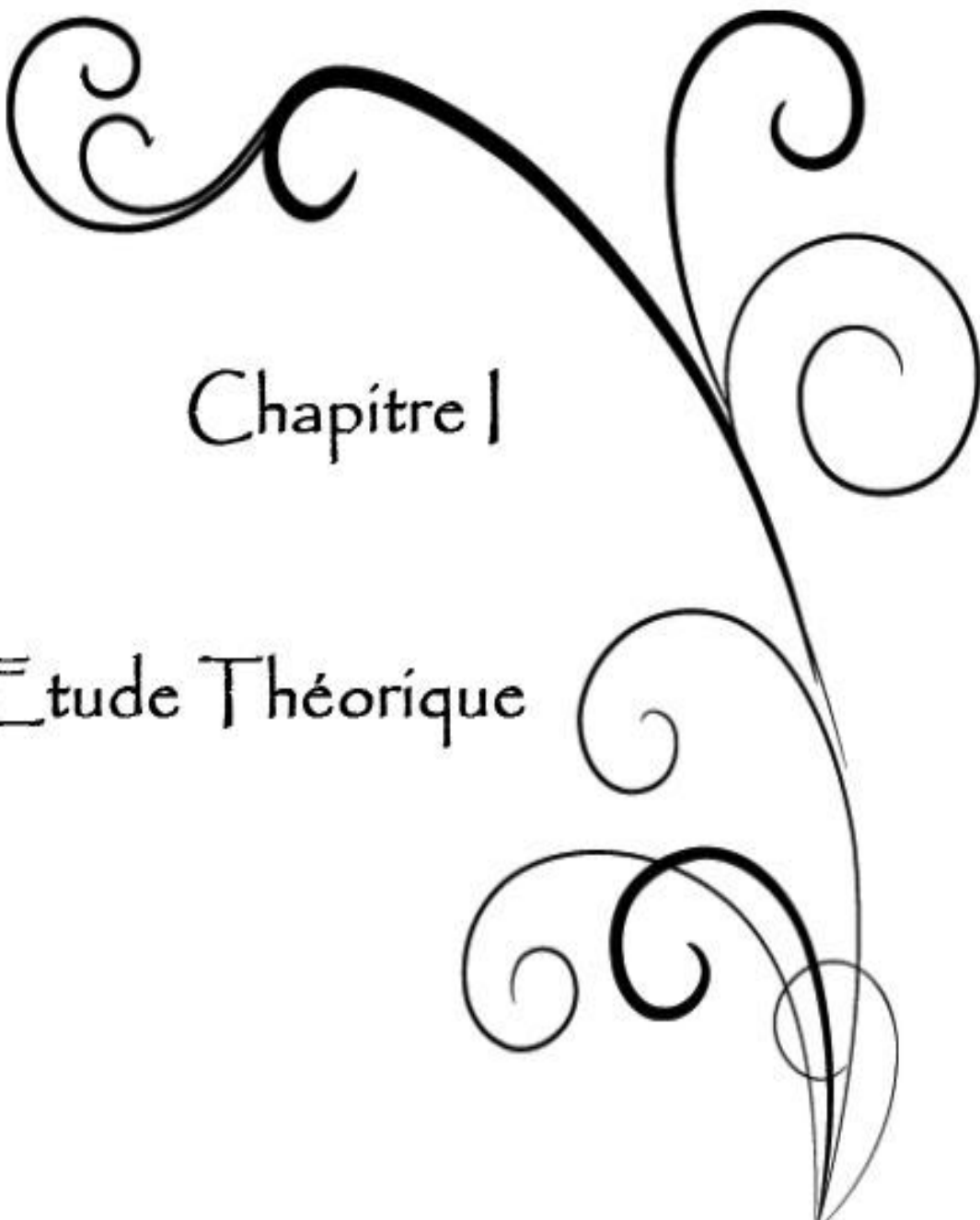
- 1) Envoi un nouveau message en utilisant le protocole SMTP ;
- 2) Récupération des messages d'un serveur de messagerie à l'aide du protocole POP3.

Pour atteindre cet objectif une répartition des tâches s'est faite dès la première semaine. Le travail durant les deux premières semaines était spécialement basé sur la documentation sur les clients de messageries (les différentes fonctionnalités offertes) ainsi que sur la programmation en langage Java et la Javamail API. A partir de la troisième semaine la partie programmation a débuté, nous avons commencé par faire les deux fonctions principales

de l'application autrement dit l'envoi la réception des messages tout en nous documentant. Après la réalisation de l'envoi et la réception nous avons traité le problème des pièces jointes et du stockage des messages dans un fichier. Nous avons enfin inclut le carnet d'adresse que nous avons réalisé.

Ce rapport s'articule autour de trois chapitres. Dans le premier chapitre, nous présentons le cadre général de notre application, nous nous intéresserons à l'étude des clients de messagerie existants ainsi que les différents protocoles permettant de les réaliser. Nous consacrons le second chapitre à la description de l'architecture de notre application. Le troisième et dernier chapitre portera sur la réalisation de l'application. Finalement nous terminons ce rapport par une conclusion générale qui présente le bilan de ce projet ainsi que d'éventuelles perspectives

Nous avons opté pour représenter les différents états d'avancement de notre projet avec un digramme de Gantt :

A large, elegant black decorative flourish with multiple loops and swirls, framing the text on the right side of the page.

Chapitre I

Étude Théorique

1. Introduction :

L'échange de courriers électroniques est certainement l'un des plus vieux et des plus utilisés de tous les services offerts sur Internet. Dans ce chapitre, nous allons faire l'état de l'art de ce service. Une première partie sera consacrée à la définition de quelques notions relatives au service de messagerie, sa structure et les différents agents intervenants. En une seconde partie, nous allons nous intéresser aux différents protocoles régulant la messagerie électronique.

2. Service de messagerie électronique :

2.1. Définition :

Un service de messagerie, dans sa forme la plus basique, est un service permettant essentiellement l'échange de messages textuels entre les différents utilisateurs enregistrés (ayant une adresse électronique valide) et connectés à un réseau informatique, que ce soit en local ou sur internet[W2]. Cet échange de messages peut s'effectuer en différé, c'est-à-dire il n'est pas nécessaire que le destinataire soit connecté au moment de l'envoi, son message sera enregistré sur un serveur et il pourra le consulter ultérieurement. On parle à ce moment de la messagerie électronique.

Actuellement, les services de messagerie sont beaucoup plus riches et présentent beaucoup plus de fonctionnalités; à savoir l'intégration des pièces jointes (joindre un fichier quelconque au message envoyé), la gestion du courrier indésirable (les SPAM) et la manipulation des listes de diffusion (permettre l'envoi multiple).

2.2. Différents agents d'un service de messagerie électronique :

Pour procéder à l'envoi et la réception des messages un service de messagerie électronique fait appel à plusieurs agents, et qui sont :

- **Le Mail User Agent (MUA)**

Le MUA est un programme qui sert à lire, écrire, répondre, recevoir des messages. Il présente généralement une interface graphique riche à la disposition de l'utilisateur. Le MUA ne permet pas de transférer le message au destinataire, de ce fait il fait appel à un MTA.

- **Le Mail Transfer Agent (MTA)**

Le MTA est un programme dédié à la transmission des messages entre utilisateurs que ce soit en local ou sur des machines distantes. En général il existe un seul MTA par machine. Un MTA reçoit, habituellement par le protocole SMTP, les emails envoyés soit par des clients de messagerie électronique (MUA), soit par d'autres MTA. Son rôle est de redistribuer ces courriers à des Mail Delivery Agent (MDA) et/ou d'autres MTA.

- **Le Mail Delivery Agent (MDA)**

Le MDA Agent marque la fin de l'acheminement du mail vers la destination. C'est le MDA qui reçoit le message du MTA et se charge de le placer dans la boîte aux lettres de l'utilisateur. Il doit donc gérer les éventuels problèmes tels qu'un disque plein ou une boîte aux lettres corrompue et doit impérativement signaler au MTA toute erreur de délivrance. Parmi les MDA connus nous citons : Procmail, maildrop.

3. Client de messagerie électronique :

3.1. Définition :

Un client de messagerie électronique ou MUA (abréviation de l'anglais Mail User Agent) est un logiciel utilisé pour envoyer et recevoir des **courriers électroniques**. Ce terme peut référer à n'importe quel système capable d'accéder à la boîte de courriers électroniques d'un utilisateur quelconque.

Il existe une panoplie de clients de messagerie. Les plus connus d'entre eux sont **Microsoft Outlook** et **Mozilla Thunderbird** ils seront présentés et comparés à la fin de ce chapitre.

3.2. Avantages :

- Recherche automatique des messages une fois le logiciel ouvert.
- Gestion de plusieurs comptes de messagerie.
- L'archivage des messages des différents comptes.
- Gestion efficace des SPAM par l'anti-virus.
- Certains clients de messagerie comme Mozilla Thunderbird servent de lecteur de flux RSS, outil indispensable pour suivre les nouvelles des sites favoris.

4. Structure et format d'un message électronique :

4.1. Structure des messages :

Un message possède fondamentalement trois parties :

❖ L'enveloppe :

Un ensemble de lignes contenant les informations de transport telles que l'adresse de l'expéditeur, l'adresse du destinataire ou encore **l'horodatage** du traitement du

message par les serveurs intermédiaires nécessaires aux serveurs de transports (MTA) faisant office de bureaux de tri postal. L'enveloppe commence par une ligne **From** et est modifié par chaque serveur intermédiaire. Ainsi, grâce à l'enveloppe, il est possible de connaître le chemin parcouru par le courrier et le temps de traitement par chaque serveur.

Si le message est destiné à plusieurs destinataires, chacun a sa propre enveloppe mais le même en-tête et le même corps du message.

❖ Le message : composé à son tour de deux éléments :

- Les champs d'en-tête : (*header fields* en anglais)

Un ensemble de lignes décrivant les paramètres du message, tels que l'expéditeur, le destinataire, la date, etc. Chaque entête possède la forme suivante :

✓ Les champs nécessaires :

- **From**: adresse électronique de l'expéditeur.
- **To**: adresse électronique du destinataire.
- **Date**: date de création du courrier.

✓ Les champs facultatifs :

- **Received**: diverses informations sur les serveurs intermédiaires et la date de traitement du message associée.
- **Reply-To**: Une adresse pour la réponse.
- **Subject**: Le sujet du message.
- **Message-ID**: un identifiant unique du message.

- le corps du message :

Il contient les données utiles, c'est à dire le contenu du message à transmettre. Il doit être impérativement séparé de l'entête par une ligne vide. Il n'est pas toléré d'insérer des caractères non imprimables tels que les espaces ou les tabulations sinon la structure du message sera erronée.

4.2. Formats standards des messages : **La norme MIME**

Pour des raisons de compatibilité, plusieurs normes ont été élaborées afin d'uniformiser la structure des messages et de définir des formats standards. Parmi ces standards nous citons Multipurpose Internet Mail Extension ou MIME[W1]. Les principaux apports du standard MIME sont :

- Possibilité d'avoir plusieurs objets (pièces jointes) dans un même message.
- Une longueur de message illimitée.
- L'utilisation de jeux de caractères (alphabets) autres que le code ASCII.
- L'utilisation de texte enrichi (mise en forme des messages, polices de caractères, couleurs, etc.)
- Des pièces jointes binaires (exécutables, images, fichiers audio ou vidéo, etc.) comportant éventuellement plusieurs parties.

MIME utilise des directives d'entête spécifiques pour décrire le format utilisé dans le corps d'un message, afin de permettre au client de messagerie de pouvoir l'interpréter correctement :

- **MIME-Version** : Il s'agit de version du standard MIME utilisée dans le message. Actuellement seule la version 1.0 existe.
- **Content-type** : Décrit le type et les sous-types des données.
- **Content-Transfer-Encoding** : Définit l'encodage utilisé dans le corps du message.
- **Content-ID** : Représente un identificateur unique de partie de message.
- **Content-Description** : Donne des informations complémentaires sur le contenu du message.
- **Content-Disposition** : Définit les paramètres de la pièce jointe, notamment le nom associé au fichier grâce à l'attribut filename.

4.3. Les principaux types MIME :

Le type MIME, utilisé dans l'entête Content-Type, est utilisé pour typer les documents attachés à un message. Les principaux types de données, appelés parfois " types de données discrets ", sont les suivants :

- **Text:** données textuelles lisibles (simple, html)
- **Image:** données binaires représentant des images numériques (formats : jpeg, gif, png).
- **Audio :** données numériques sonores (basic, wav).
- **Video :** données vidéo (mpeg).
- **Application :** autres données binaires (PDF etc...).

5. Protocoles de la messagerie électronique :

5.1. Le protocole Simple Mail Transfer Protocol (SMTP) :

Le *Simple Mail Transfer Protocol* (littéralement « Protocole simple de transfert de courrier »), généralement abrégé **SMTP**, est un protocole de communication utilisé pour transférer le courrier électronique (courriel) vers les serveurs de messagerie électronique.

SMTP est un protocole assez simple (comme son nom l'indique). On commence par spécifier l'expéditeur du message puis, le ou les destinataires d'un message, puis, en général après avoir vérifié leur existence, le corps du message est transféré. Il est assez facile de tester un serveur SMTP en utilisant la commande **telnet** sur le port 25 d'un serveur distant, voilà en ce qui suit une petite démonstration :

- Lancez telnet :

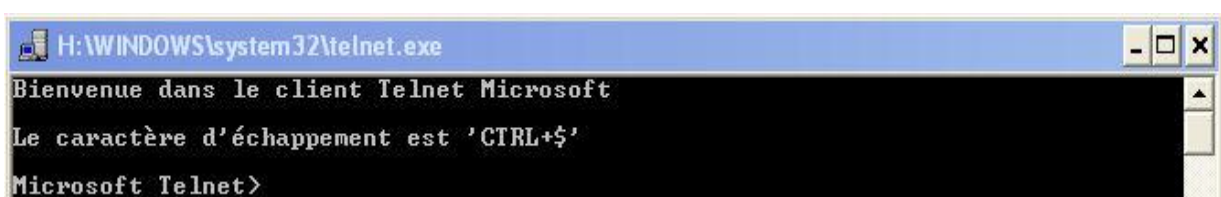


Figure 2 : Lancement de telnet

- Puis tapez :

```
open smtp.laposte.net 25
```

Figure 3 : Connexion au serveur SMTP

Cela signifie tout simplement que nous demandons d'établir une connexion avec le serveur SMTP de la Poste, à l'adresse *smtp.laposte.net* et sur le port 25

NB : Pensez à vérifier que votre serveur SMTP écoute bien sur le port 25. C'est généralement le cas, sauf pour les serveurs sécurisés (exemple : GMAIL = port 587).

5.2. Le protocole Post Office Protocol (POP) :

En informatique, le **POP (Post Office Protocol** littéralement *le protocole du bureau de poste*), est un protocole qui permet de récupérer les messages électroniques situés sur un serveur de messagerie électronique. Ce protocole a été réalisé en plusieurs versions respectivement POP1, POP2 et POP3. Actuellement, c'est **POP3**, ou *Post Office Protocol Version 3* qui est utilisé de façon standard.

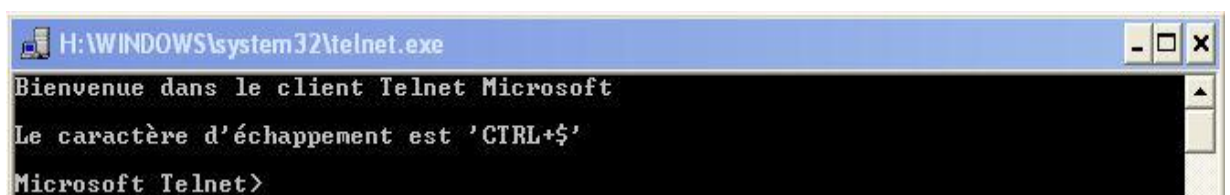
Le POP procède comme suit pour la récupération des messages :

- Il se connecte au serveur.
- Il télécharge tous les messages sur la machine locale en les marquant comme nouveaux messages.
- Il efface les messages du serveur et se déconnecte

Du coup l'utilisateur peut consulter ses messages hors connexion.

On va voir dans ce qui suit une démonstration de la connexion au serveur POP3 en utilisant telnet :

- Lancez d'abord telnet :



- Puis tapez :

```
open pop.laposte.net 110
```

Figure 4 : Connexion au serveur POP

Cela signifie nous demandons d'établir une connexion avec le serveur **pop.laposte.net** sur le port **110**.

Si tout se passe bien, un message comme celui qui suit devrait s'afficher (en rouge) :

```
open pop.laposte.net 110  
+OK connected to pop3 on 8201
```

5.3. Le protocole Internet Message Access Protocol (IMAP) :

Internet Message Access Protocol (IMAP) est un protocole qui permet de récupérer les courriers électroniques sur des serveurs de messagerie mais contrairement au POP l'IMAP consulte les messages sur le serveur, rien n'est téléchargé en local et rien n'est effacé du serveur sauf suite à la demande explicite de l'utilisateur.

Remarque :

Dans notre application nous avons seulement utilisé le protocole POP3 et pas l'IMAP4.

6. Acheminement d'un message électronique :

L'acheminement des courriels est régi par plusieurs standards : **SMTP** est dédié à l'envoi d'un message, **POP** et **IMAP** servent à rapatrier des messages pour leur lecture.

- ❖ Le **MUA** (*Mail User Agent* ou client de messagerie) de l'expéditeur envoie par SMTP le message à un serveur de courriel (celui de son fournisseur d'accès en général) ou MTA, *Mail Transfer Agent*.
- ❖ Le premier **MTA** route le message vers le **MTA** hébergeant le domaine du destinataire. Le MTA final délivre au **MDA** (**Message Delivery Agent**) qui est chargé de la gestion des boîtes aux lettres.

- ❖ Le destinataire, par l'intermédiaire de son **MUA**, demande à son serveur de courrier (**MDA**) les nouveaux messages par l'utilisation des protocoles IMAP ou POP.
- ❖ Le destinataire, par l'intermédiaire de son navigateur, demande au serveur web de retrouver les nouveaux messages sur le **MDA**.
- ❖ Le serveur envoie le message au **MUA** du destinataire.

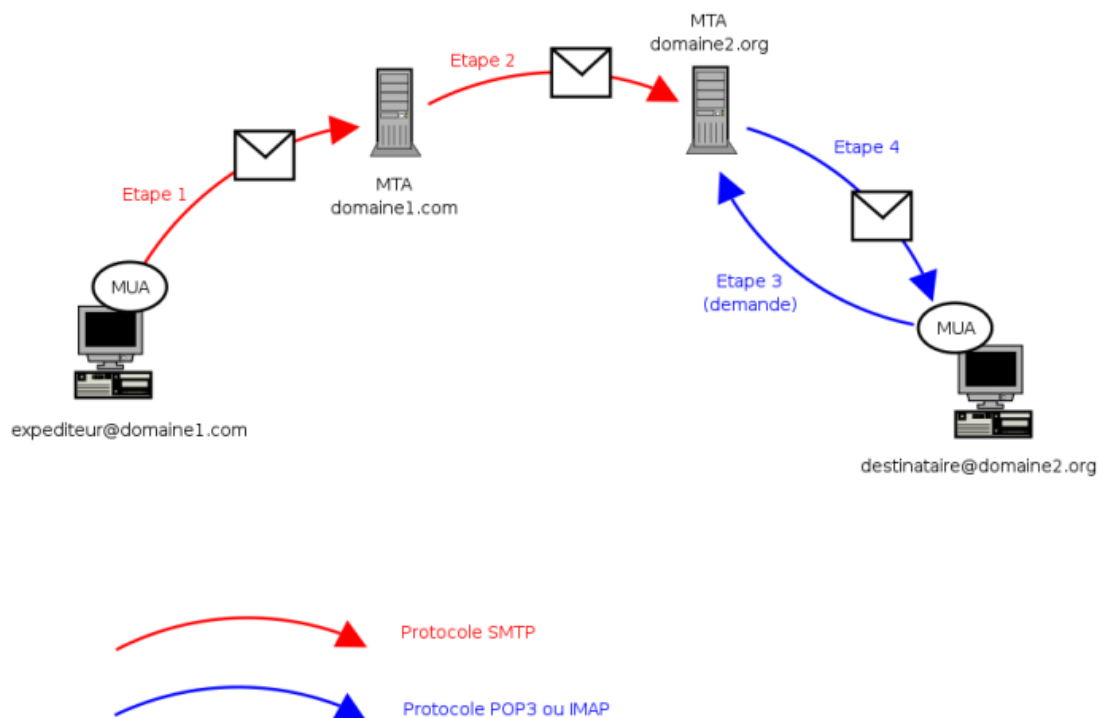


Figure 5 : Schéma représentant l'acheminement d'un message électronique.

7. Quelques clients de messagerie électronique :

7.1. Microsoft Outlook :

Microsoft Outlook (officiellement Microsoft Office Outlook) est un gestionnaire d'informations personnelles et un client de messagerie électronique propriétaire édité par Microsoft. Il fait partie de la suite bureautique Microsoft Office.

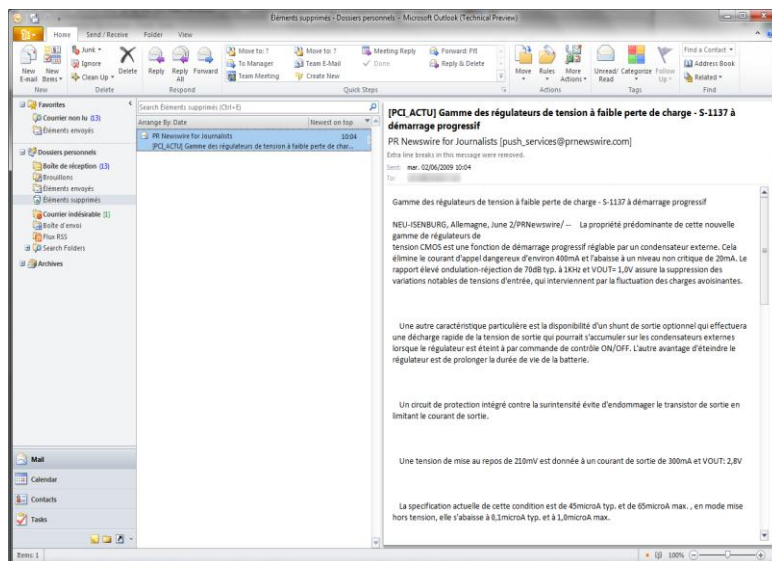


Figure 6 : Microsoft Outlook

Bien qu'il soit principalement utilisé en tant qu'application de courrier électronique, il propose aussi un calendrier et un gestionnaire de tâche et de contact.

La première version fut sortie avec Microsoft Office 97, la version actuelle (Outlook 2010) et la 14ème.

7.2. Mozilla Thunderbird :

Mozilla Thunderbird est un client de messagerie libre distribué gratuitement par la Fondation Mozilla uniquement consacré au courrier électronique, aux groupes de discussion et aux flux RSS.

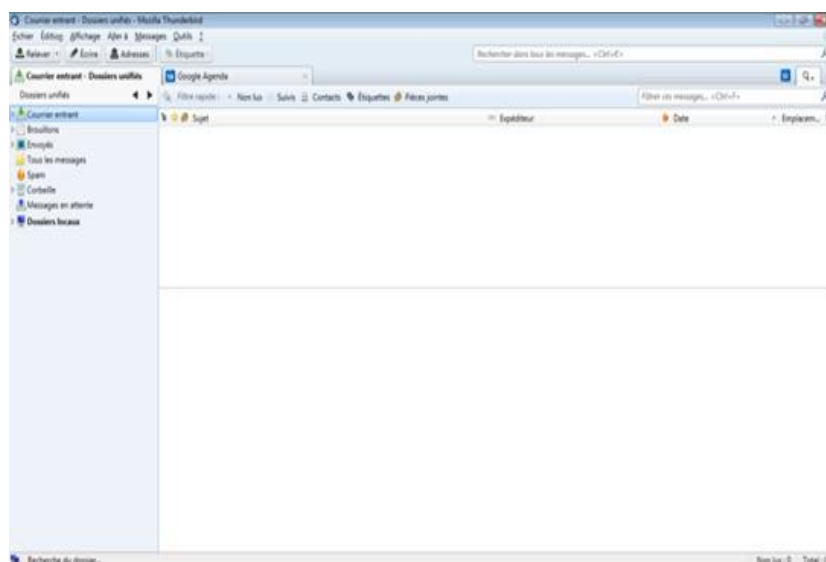


Figure 7 : Mozilla Thunderbird

Le 7 décembre 2004 fut lancé la première version de Thunderbird, la version

courante est la version 12.0.1 sortie le 30 avril 2012 elle est disponible en 52 langues.

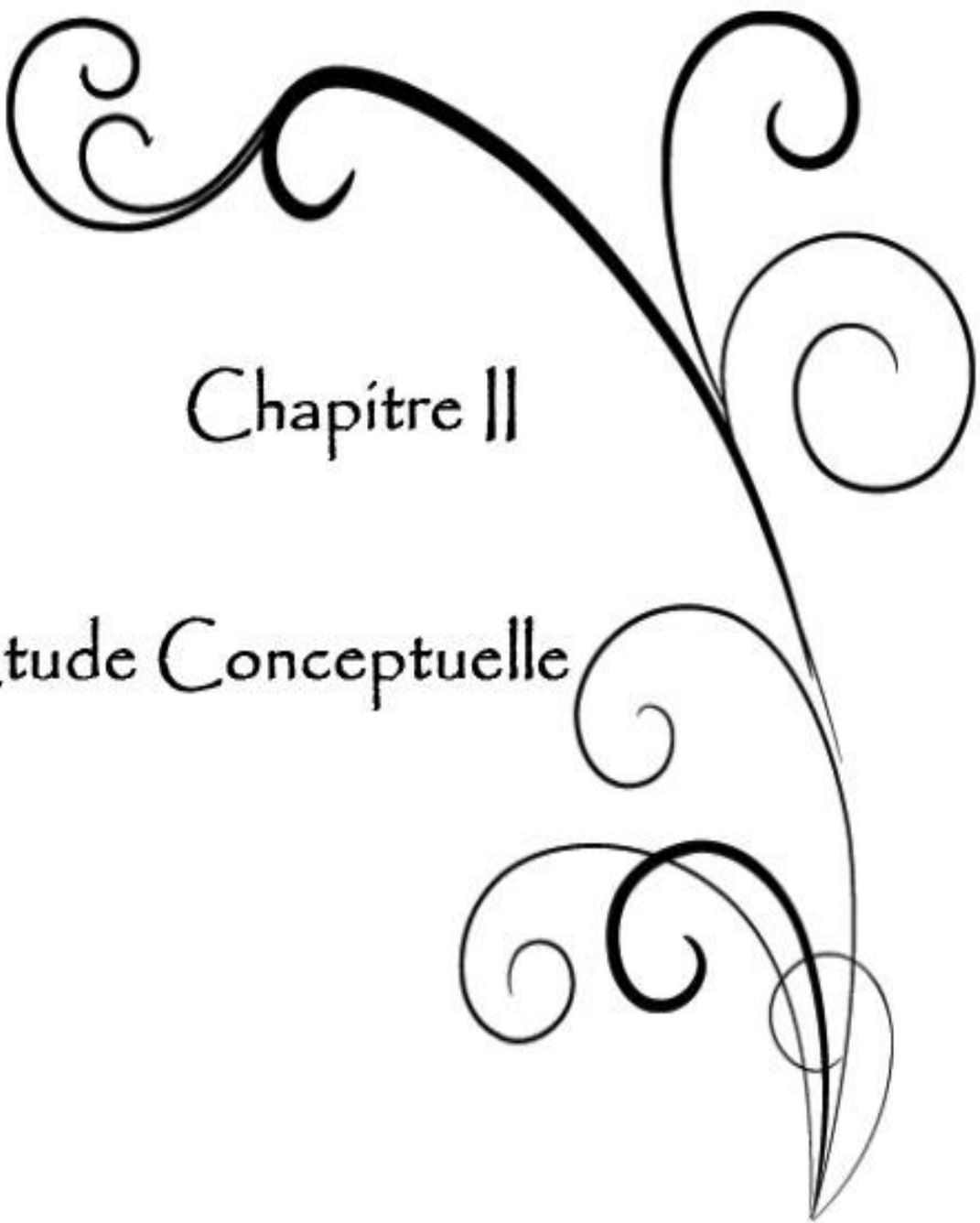
7.3 Comparaison entre Outlook et Thunderbird :

	Microsoft Outlook			Mozilla Thunderbird		
Prix	Payant			Gratuit (libre)		
Première version (Date de sortie/Version)	1997/97			2004/1.0		
Dernière version (Date de sortie/Version)	2010/14.0			2012/12.0.1		
Systèmes d'exploitation supportés	Windows	Mac Os	Linux	Windows	Mac Os	Linux
	oui	oui	Non	oui	oui	oui
Protocoles supportés	SMTP	POP3	IMAP4	SMTP	POP3	IMAP4
	Oui	oui	Oui	oui	oui	oui

Tableau 1 : Comparaison entre Outlook et Thunderbird

8. Conclusion :

Tout au long de ce chapitre nous avons fait le tour des différents éléments théoriques impliqués dans la réalisation d'un client de messagerie électronique. Nous avons commencé par présenter qu'est-ce qu'un service de messagerie électronique et les différents agents intervenant dans son fonctionnement. Nous avons ensuite étudié les différents clients de messagerie électronique, le format d'un message électronique et présenté le standard MIME qui normalise la structure et le codage des messages. Et finalement, nous avons présenté une multitude de protocoles intervenant dans les communications entre les différents agents du service de messagerie.

A large, elegant black decorative flourish or scrollwork element that curves from the top left, arches over the text, and descends towards the bottom right, framing the chapter title.

Chapitre II

Etude Conceptuelle

1. Introduction :

Afin de garantir le bon fonctionnement et l'efficacité de notre application nous allons aborder dans ce chapitre une étude conceptuelle englobant les différentes fonctionnalités qu'on voudrait intégrer dans notre application.

2. Architecture du système :

Dans ce qui suit nous retrouvons une présentation de la conception du prototype proposé de notre application et cela sous forme de schéma.

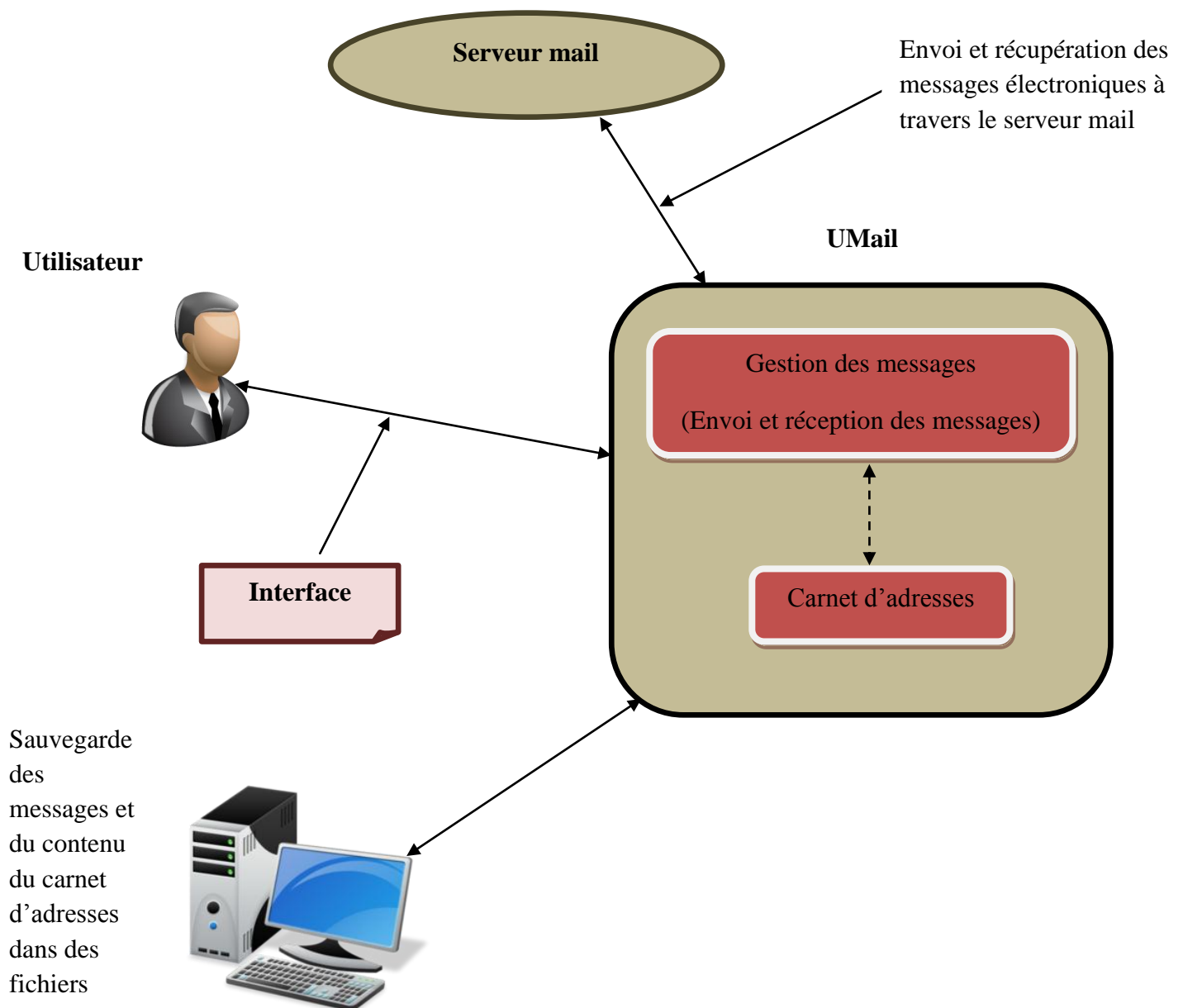


Figure 8 : Schéma général du système

Notre prototype représenté dans la figure précédente, se compose principalement de deux modules dont les objectifs sont :

- La gestion des messages électronique i.e. l'envoi et la réception des messages ainsi que des pièces jointes.
- La gestion du carnet d'adresse (ajout, suppression et modification des contacts).

Toutes les fonctionnalités des deux principales fonctions seront mise à la portée de l'utilisateur grâce à une interface graphique qui permettra en plus de ça à l'utilisateur de communiquer avec l'application en entrant des données et en exécutant des tâches.

3. Diagrammes de cas d'utilisation :

Dans le but de réaliser une meilleure application nous devons déterminer d'une façon précise ce que doit accomplir cette dernière. Ce n'est autre que les diagrammes de cas d'utilisation qui nous indiquerons cela.

Les diagrammes des cas d'utilisation représentent toutes les tâches que peuvent demander les différents utilisateurs de l'application, et qui seront réalisées par cette dernière.

Nous retrouvons ci-dessous les différents diagrammes de cas d'utilisation de notre application « UMail ».

3.1. Diagramme général de cas d'utilisation :

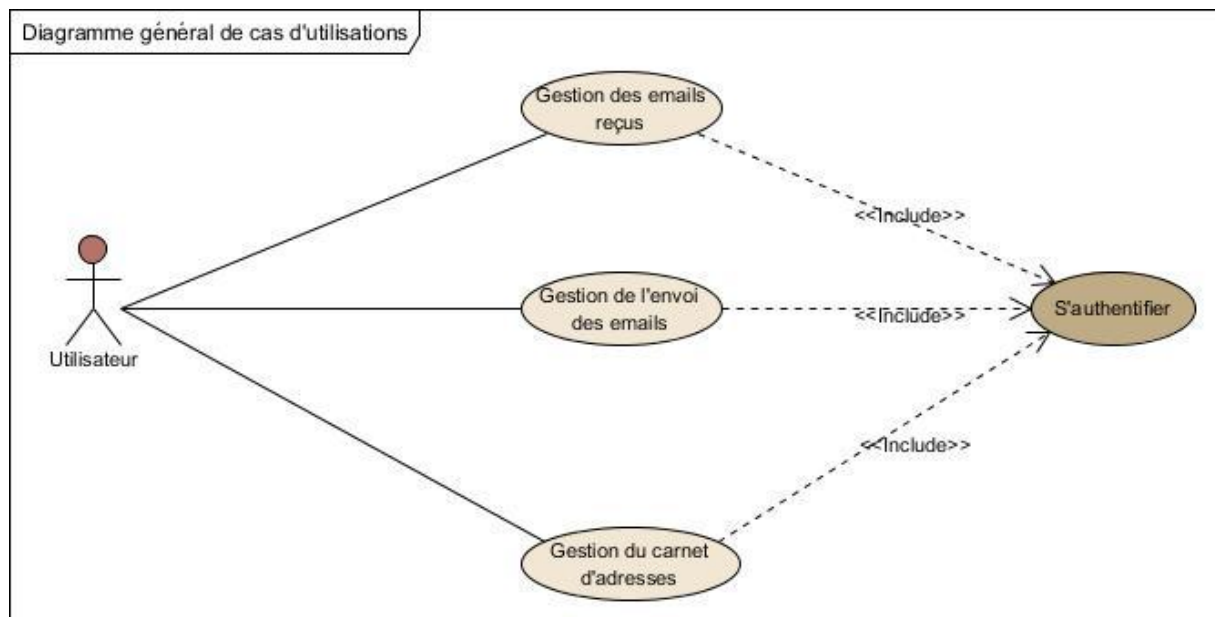


Figure 9 : Diagramme général de cas d'utilisation

3.2. Diagrammes détaillés de cas d'utilisation :

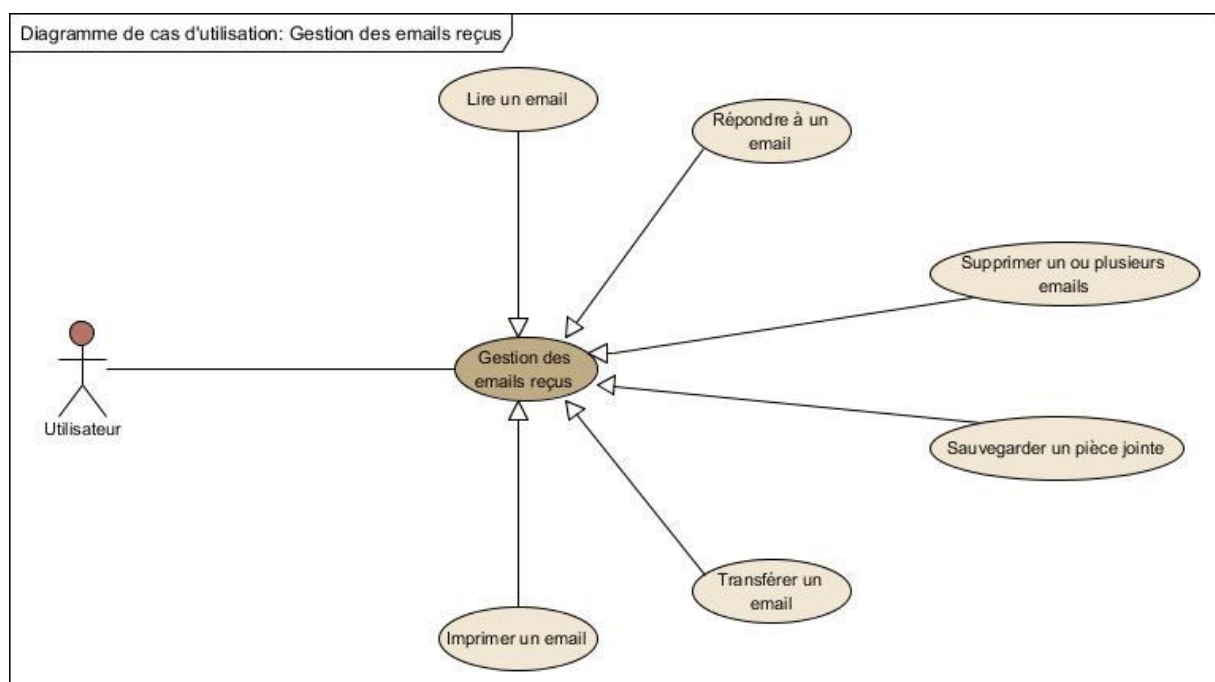


Figure 10 : Gestion des emails reçus

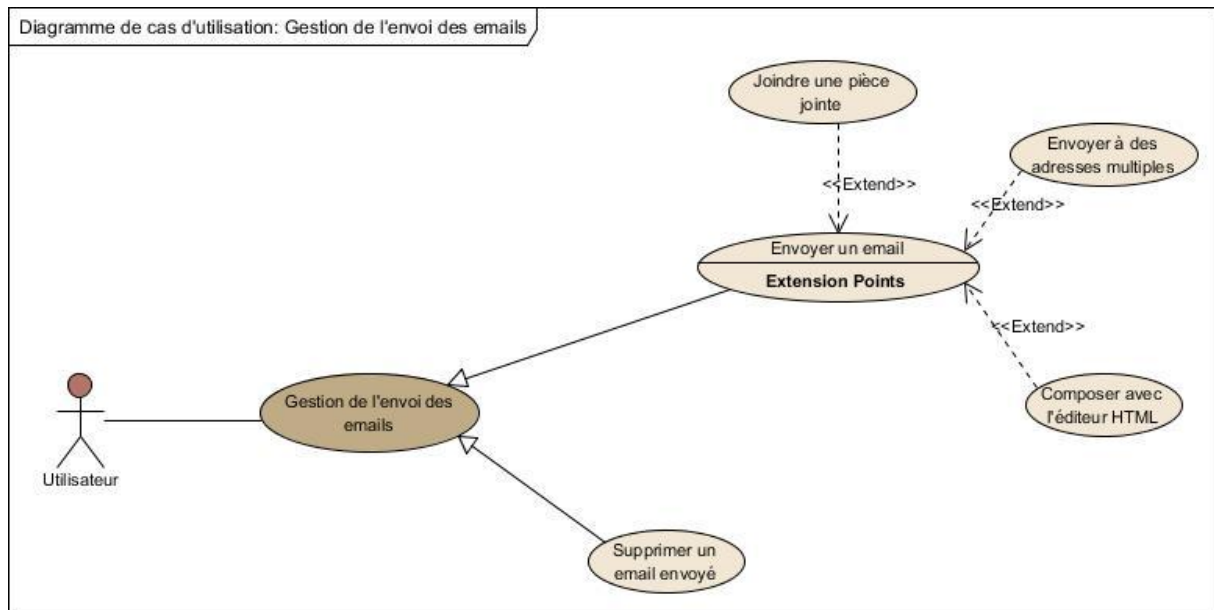


Figure 11 : Gestion de l'envoi des emails

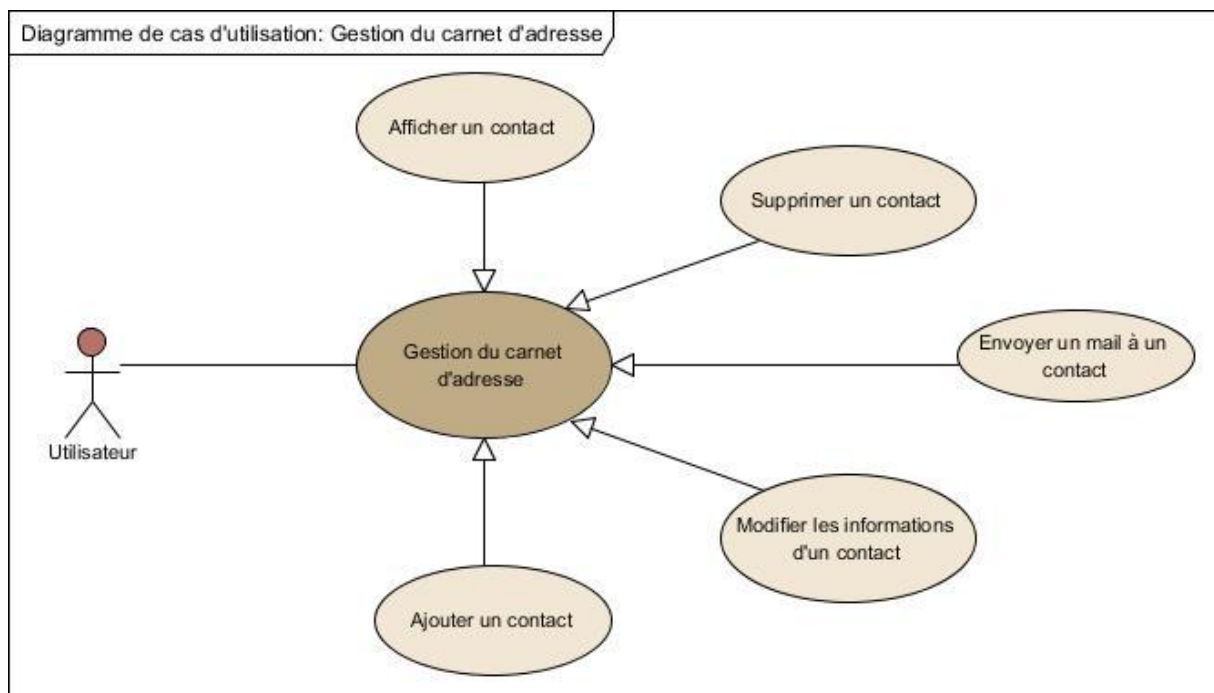


Figure 12 : Gestion du carnet d'adresses

4. Identification des classes :

4.1. Présentation des classes :

- **SMTPAuthenticator :**

La plupart des serveurs SMTP ont besoin d'authentifier les utilisateurs distants qui tentent d'envoyer des e-mails. Pour cette raison, nous avons besoin de créer une classe SMTPAuthenticator qui hérite de javax.mail.Authenticator et fournit la méthode getPasswordAuthentication.

Authenticator représente un objet qui sait comment obtenir l'authentification pour une connexion réseau, dans la sous-classe qu'on a créé, on a redéfini la méthode getPasswordAuthentication qui retourne Le PasswordAuthentication grâce aux données recueillies auprès de l'utilisateur, ou null si aucune n'est fournie.

La classe PasswordAuthentication est un support de données qui est utilisé par l'authentificateur. Il est tout simplement un dépôt pour un nom d'utilisateur et un mot de passe.

On créera une instance de la classe SMTPAuthenticator lors de la création de la session, Lorsque l'authentification est requise, le système invoque la méthode getPasswordAuthentication.

- **EmailSender :**

Remarque : les filles, la méthode prepareproperties n'est pas nécessaire dans la classe EmailSender => elle a été supprimée car l'objet props est configuré dans la classe accueil dans la méthode connect.

La classe EmailSender comprend deux méthodes : prepareMessage et SendEmail.

1. **PreparMessage** : cette fonction modélise un message, elle retourne un **MimeMessage** (qui signifie Internet Multipurpose Internet Mail Extensions Message expliqué en détail dans le chapitre précédent).
2. **SendEmail** : cette méthode se charge d'envoyer le message Crée un objet **Transport** qui peut transporter le message aux adresses de pacifiées.

- **LireMessage :**

Cette classe est l'interface qui apparait lors d'un clic sur un message précis. Elle affiche toutes les informations d'un email (Expéditeur, objet, date, texte, pièces jointes). Trois opérations sont possibles, voir quatre si le message contient une ou plusieurs pièces jointe. On peut éventuellement répondre à l'expéditeur ; une interface **SendEmail** s'ouvre et transcrit automatiquement l'adresse de l'expéditeur dans le champ du destinataire, ou alors transférer le message ; de la même manière le texte du message est transcrit dans l'éditeur HTML de la fenêtre **SendEmail**. L'utilisateur peut aussi imprimer son message.

Si le message contient des pièces jointes, l'utilisateur peut choisir la ou les pièces à sauvegarder et où les sauvegarder.

- **Aperçu :**

Cette classe génère un aperçu du message à imprimer avant impression.

- **SauvegarderPieces :**

Une interface qui affiche la liste des pièces jointes à un message précis, et offre la possibilité de sélectionner au choix les pièces à sauvegarder ainsi que le choix du dossier de sauvegarde.

- **SendEmail :**

Cette interface doit apparaître lorsque l'utilisateur souhaite envoyer un message (répondre ou écrire à un contact du carnet d'adresse). Elle contient éventuellement les champs essentiels à l'envoi d'un message : destinataire, objet, ou pourquoi pas une pièce jointe, ainsi que la zone du texte à rédiger, qui est un éditeur HTML offrant de nombreuses options sur l'esthétique du texte.

- **Message :**

Une interface qui affiche une ligne des informations primaires d'un message (expéditeur, objet, date). Si ce message contient des pièces jointes, une icône sera visible sur la ligne d'informations de ce dernier.

Les Classes du carnet d'adresses :

Le carnet d'adresse a pour fonction de gérer les contacts de l'utilisateur de UMail et d'offrir toutes les opérations qu'il peut effectuer sur un contact (supprimer, écrire).

Carnet_View est la classe principale du carnet, donc fait appel aux classes suivantes :

- **Contact :** elle regroupe toute les informations concernant un contact dans un tableau, elle est identifiée par un entier.
- **Infos :** elle contient toutes les informations concernant un contact précis. Ce qui la diffère de la classe précédente est le fait qu'elle soit une interface plus détaillée que celle de la classe contact.
- **Fichier :** le stockage des contacts (Informations évidemment comprises) se fait dans un fichier où chaque ligne représente un contact. Cette classe a pour fonction de lire et d'écrire dans ce fichier grâce aux classes BufferedWriter et BufferedReader.
- **Saisie :** cette classe permet à l'utilisateur d'introduire les informations d'un contact précis.
- Enfin La classe principale Carnet_View représente l'interface du Carnet. Elle affiche une liste de contacts (de type Contact) ainsi que l'intégralité des informations détaillées d'un contact sélectionné au click. Elle offre la possibilité d'ajouter un contact, en remplissant les champs de la classe Saisie par les informations de ce dernier, qui seront enregistrés par la suite dans un fichier (type Fichier). En parallèle si l'on veut envoyer un email à un contact déjà existant dans la liste affichée, on le sélectionne puis on clique sur Ecrire, son adresse sera écrite

automatiquement dans le champ destinataire, on fait de même pour supprimer un contact.

4.2. Description des classes (attributs et méthodes) :

- **Classes concernant l'envoi et la réception des messages :**

Classe	Attributs	Description des attributs	Méthodes
SMTPAuthenticator (hérite de Authenticator)	username password needAuth (booléen)	Nom d'utilisateur Mot de passe	Constructeur() getPasswordAuthentication()
Authentication (Interface)	nomUser motPasse portSMTP portPOP serveurSMTP serveurPOP serveur	Nom d'utilisateur Mot de passe Numéro du port SMTP Numéro du port POP Serveur SMTP Serveur POP Le nom du serveur	Connexion() + Les getters
EmailSender	smtpServer port user password auth from	Serveur SMTP Numéro du port SMTP Nom d'utilisateur Mot de passe Nom de l'expéditeur	Constructeur() prepareProperties() prepareMessage() sendEmail()
SendEmail (Interface)	destinataire htmlEditor objet pieceJointe	Adresse du destinataire Editeur de texte html Objet du message Fichier attaché	Constructeur() Envoyer() PieceJointe() + les getters
LireMessage (Interface)	émetteur dests date objet	L'émetteur Les destinataires Date de la réception Objet du message	Constructeur() Sauvegarder() Répondre() Transférer() Imprimer()
Aperçu (Interface)	page	L'aperçu du message à imprimer	Constructeur() Impression()

Message	date from object	Date de l'envoi du message Adresse de l'expéditeur Objet du message	Constructeur() afficherInfo()
SauvegarderPiece	pieces	Liste de pièces jointes	Constructeur() Sauvegarder() AjouterPiece()

- **Classes concernant le carnet d'adresses :**

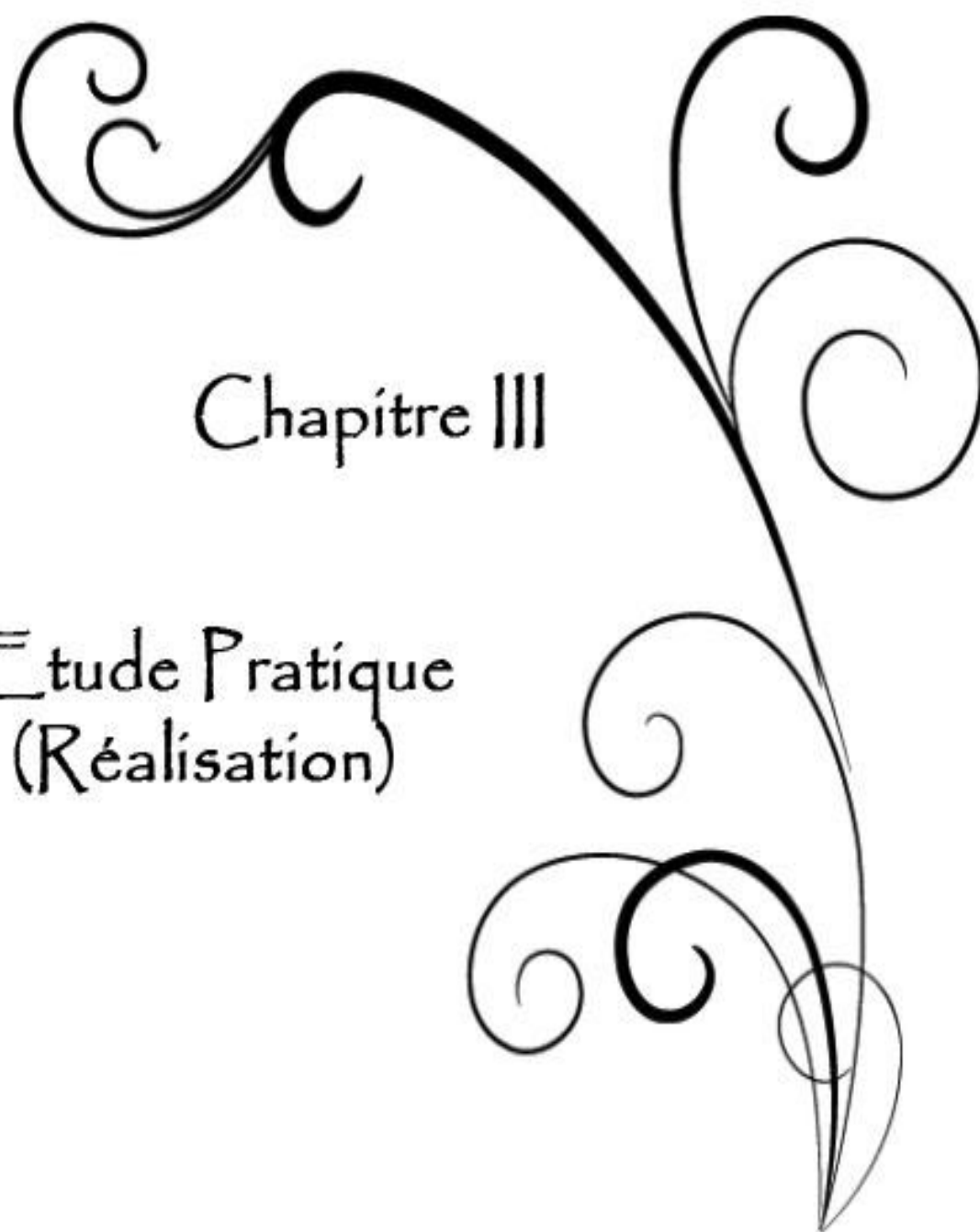
Classes	Attributs	Description des attributs	Méthodes
Saisie (Interface)	nom prénom courriel numDomicile numPortable	Nom du contact Prénom du contact Adresse email du contact Les numéros du contact	Constructeur() setDonné() Supprimer() ecrireLeContact() ecrireLaListe() + les setters et les getters
Info (Interface)	nom prénom courriel numDomicile numPortable photo	Nom du contact Prénom du contact Adresse email du contact Les numéros du contact Photo du contact	Constructeur() + les setters et les getters
Contact (Interface)	listeInfo id	La liste des informations du contact L'identifiant du contact	Constructeur() afficheInfo() affichePhoto() + les setters et les getters
Fichier	urlFichier	Le chemin vers le fichier où les contacts sont sauvegardés	ouvrirEnLecture() lire() ouvrirEnEcriture() ecrire() fermer()
Carnet_View (Interface)	listeContacts	La liste des contacts	Constructeur() initContacts() Ajouter() Supprimer() Ecrire() Contact()

- **Description de la classe accueil :**

Nom	Attributs	Description des attributs	Méthodes
Accueil (Interface contenant la classe main)	messages folder session props fichierElementsEnvoyes fichierMessagesReçus listeMessageReçus listeMessageEnvoyés		MaiRen() Connect() afficherMessagesReçus() afficherMessagesEnvoyés() supprimerMessagesReçus() supprimerMessagesEnvoyés() message() carnet() nouveau() répondre() transférer() supprimer() rafraîchir() sauvegarder() déconnexion()

5. Conclusion :

Nous avons tout d'abord vu dans ce chapitre un schéma global de notre système. Ensuite nous avons approfondi notre étude conceptuelle en réalisant les différents diagrammes de cas d'utilisation. Enfin, nous avons étudié en détail les différentes classes qui composent notre application.



Chapitre III

Etude Pratique (Réalisation)

1. Introduction:

Ce chapitre constitue l'âme du processus de développement du logiciel et a pour objectif la mise en œuvre de chacun des modules décrits dans le chapitre précédent. Nous consacrons la première partie à la présentation de l'environnement de l'application. Par la suite, nous présenterons les codes sources des deux fonctions principales de notre application. Enfin, nous exposerons quelques interfaces de cette dernière.

2. Environnement de travail :

2.1. L'environnement matériel :

- ❖ Le fonctionnement de notre logiciel nécessite une connexion internet.
- ❖ Nous avons utilisé des PC portables de puissance moyenne :
 - Dell Inspiron Pentium core_2duo 3Go de mémoire vive.
 - Deux HP G62 core i3 3Go de mémoire vive.
 - Toshiba C660 3Go de mémoire vive.
 - Compaq HP.

2.2. L'environnement logiciel :

Le long de la phase de développement, nous avons utilisé l'environnement logiciel suivant :

❖ Système d'exploitation :

Windows 7 (32 bits/64 bits).

❖ Outil de développement :

- Netbeans version 7.0.1. téléchargeable sur fr.netbeans.org
- Bibliothèques utilisées au sein de Netbeans :
 - Javamail API version 1.4.4.
 - Swing et AWT + Jtattoo
 - DJNativeSwing + FCKEditor.

- Date chooser

❖ Rédaction du rapport :

Microsoft office word 2010

3. Présentation des bibliothèques utilisées :

3.1. Javamail API :

JavaMail est une API qui permet d'utiliser le courrier électronique (e-mail) dans une application écrite en java (application cliente, applet,...). Son but est d'être facile à utiliser, de fournir une souplesse qui permet de la faire évoluer et de rester le plus indépendant possible des protocoles utilisés [W5].

Pour l'utiliser, il est possible de la télécharger sur le site de SUN : <http://java.sun.com/products/javamail>.

Les classes et interfaces sont regroupées dans quatre packages : javax.mail, javax.mail.event, javax.mail.internet, javax.mail.search.

Cette API permet une abstraction assez forte de tout système de mails, ce qui lui permet d'ajouter des protocoles non gérés en standard. Pour gérer ces différents protocoles, il faut utiliser une implémentation particulière pour chacun d'eux, fournis par des fournisseurs tiers. En standard, JavaMail 1.2 fournit une implémentation pour les protocoles SMTP, POP3 et IMAP4. JavaMail 1.1.3 ne fournit une implémentation que pour les protocoles SMTP et IMAP : l'implémentation pour le protocole POP3 doit être téléchargée séparément.

La dernière version de cette API et la version 1.4.5 sortie en mai 2012.

3.2. Jtattoo :

Jtattoo consiste en différents thèmes 'Look and Feel' pour les application Swing. Elle offre au développeur la possibilité d'améliorer l'aspect de l'interface graphique[W8].

4. Réalisation des fonctions de l'envoi et de la réception :

4.1. Code source de la fonction 'SendEmail' responsable de l'envoi des messages :

```
// etablr une connexion et envoyer le message
public void sendEmail(Session mailSession,String subject,String HtmlMessage,String[] to,
String PieceJointe,String Fichier)
throws Exception
{
    Transport transport = null;
    transport = mailSession.getTransport("smtps");
    MimeMessage message = prepareMessage(mailSession,"ISO-8859-2",
    from, subject, HtmlMessage, to,PieceJointe);
    message.setSentDate(new Date());
    System.out.print("préparé");

    transport.connect(smtpServer, Integer.parseInt(port),user,password);
    System.out.print("préparé");
    Transport.send(message, message.getAllRecipients());
    //.send(message);

    System.out.print("préparé");
    //Sauvegarde du message dans un eml
    OutputStream out;
    out = new FileOutputStream(new File(Fichier));
    try {
        message.writeTo(out);
    } catch (FileNotFoundException ex) {
        System.out.print("file");
        if (out != null)
        { out.flush();
        }out.close();
    }

    finally{
        try {
            transport.close();
        } catch (MessagingException ex) {
            System.out.print("close!");
            Logger.getLogger(EmailSender.
            class.getName()).log(Level.SEVERE, null, ex);

        }
    }
}
```

4.2. Code source de la fonction 'Connect' responsable de la récupération des messages :

```
/* Cette fonction la connexion au serveur de messagerie,
   récupération des messages et affichages de leurs en-têtes*/

public void connect(boolean auth){

    // Afficher la boîte de dialogue pour entrer les paramaitres de connexions.
    if(auth)
    {
        dialog = new Authentification(this,true);
    }

    // Construire une URL de connexion à partir des paramètres de connexion de la boîte de dialogue.
    StringBuffer connectionUrl = new StringBuffer();
    connectionUrl.append("pop3s" + "://");
    connectionUrl.append(dialog.getNomUser().getText() + ":");
    connectionUrl.append(dialog.getMotPasse() + "@");
    connectionUrl.append(dialog.getServeurPOP().getText() + "/" );
    this.getProfil().setText(dialog.getNomUser().getText());

    // Etablir une session JavaMail et se connecter au serveur.
    Store store = null;
    try {
        // configurer l'objet "Properties" à partir de la boîte de dialog.
        props=new Properties();
        props.setProperty("mail.smtp.starttls.enable", "true");
        props.setProperty("mail.smtp.socketFactory.port", dialog.getPortSMTP().getText());
        props.setProperty("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
        props.setProperty("mail.smtp.host", dialog.getServeurSMTP().getText());
        props.setProperty("mail.smtp.port", dialog.getPortSMTP().getText());
        props.setProperty("mail.smtp.password", dialog.getMotPasse());
        props.setProperty("mail.smtp.auth", "true");

        // Etablir la session javamail à partir de l'objet props et SMTPAuthenticator
        session = Session.getDefaultInstance(props,new SMTPAuthenticator(dialog.getNomUser().getText(),
            dialog.getMotPasse(),true));
```

```

        // Se connecter au serveur e-mail.
        URLName urln = new URLName(connectionUrl.toString());
        store = session.getStore(urln);
        store.connect();

    } catch (Exception ex) {
        // afficher un message d'erreur.

        Erreur("connexion impossible.", true);
    }

    try {
        //Recuperer le dossier des messages reçus du serveur
        folder = store.getFolder("INBOX");
        System.out.print("reussi!\n");

        // Ouvrir le dossier "INBOX" en mode lecture ecriture.
        folder.open(Folder.READ_WRITE);

        // Recuperer la liste des messages à partir du dossier.
        messages = folder.getMessage();

        // Affichage des messages reçus et envoyés
        afficherMessagesReçus(messages);
        afficherMessagesEnvoyés(construireEnvoyes());

    }
    catch (Exception e) {
        // afficher un message d'erreur.
        Erreur("Impossible De Telecharger Les Messages.", true);
    }

}

```

5. Présentation de l'interface graphique :

Au lancement d'UMail la fenêtre d'authentification suivante s'affiche, l'utilisateur doit introduire les différentes informations concernant sa boîte de messagerie qu'il veut gérer.



The screenshot shows a window titled "Connexion à UMail". On the left is a decorative graphic with the text "UMail". On the right is a panel titled "Authentification" containing the following fields and buttons:

- Serveur SMTP :
- Port :
- Serveur POP :
- Port :
- Serveur :
- Nom d'utilisateur :
- Mot de passe :
- Buttons:

Figure 13 : Fenêtre d'authentification

Après authentification la fenêtre de d'accueil s'affiche qui est l'interface principale de notre application, L'utilisateur se trouve devant une liste de choix :

- Composer un nouveau message et l'envoyer.
- Consulter les messages reçus.
- Consulter son carnet d'adresses et le gérer.
- Utiliser le calendrier.



Figure 14 : Fenêtre principale (l'accueil)

Lors de l'envoi d'un nouveau message la fenêtre suivante s'affiche :

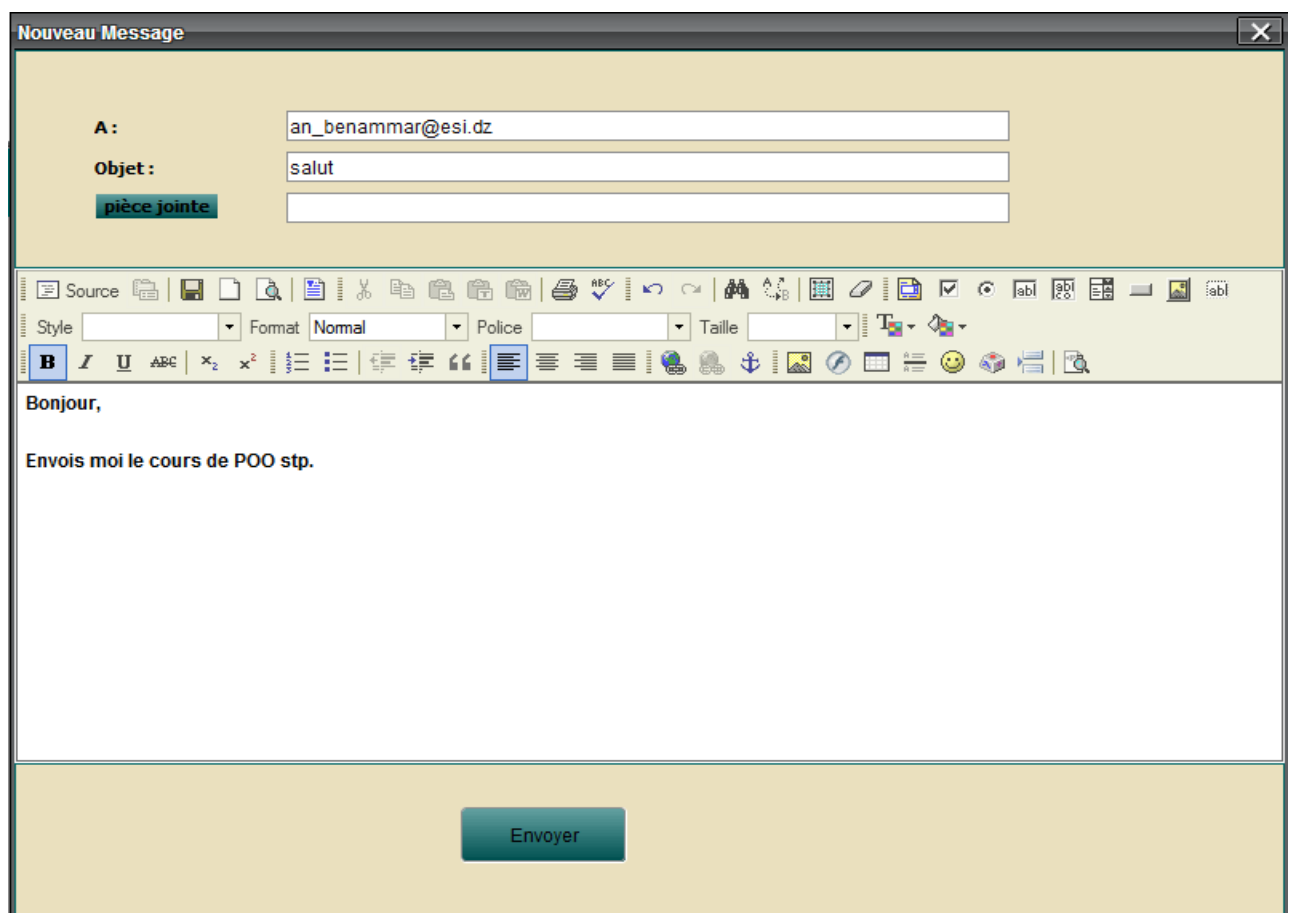


Figure 15 : Fenêtre de l'envoi d'un message (avec un éditeur HTML)

La fenêtre suivante représente l'accueil du carnet d'adresse l'utilisateur peut ajouter, visualiser, supprimer des contacts ou même encore leurs écrire.

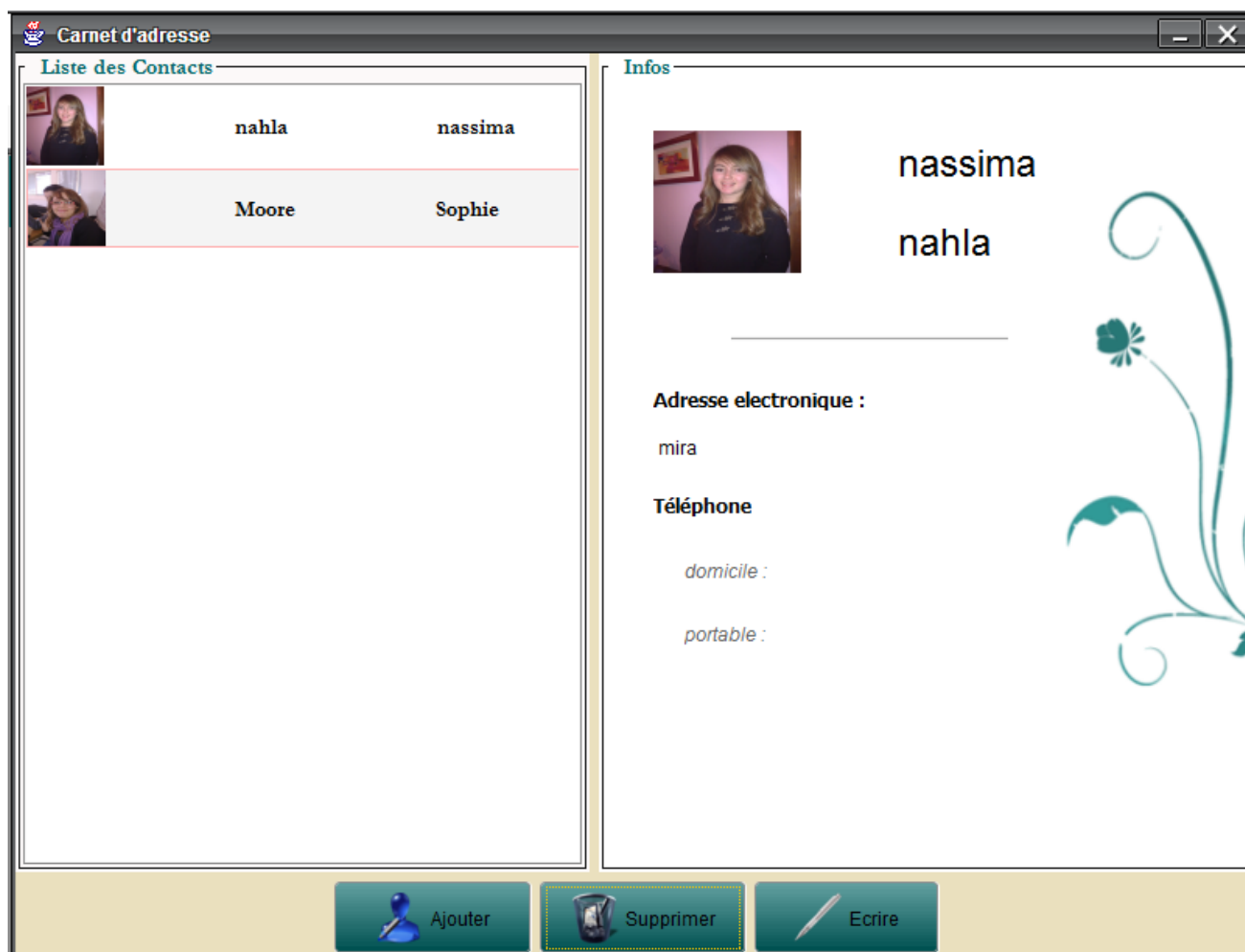
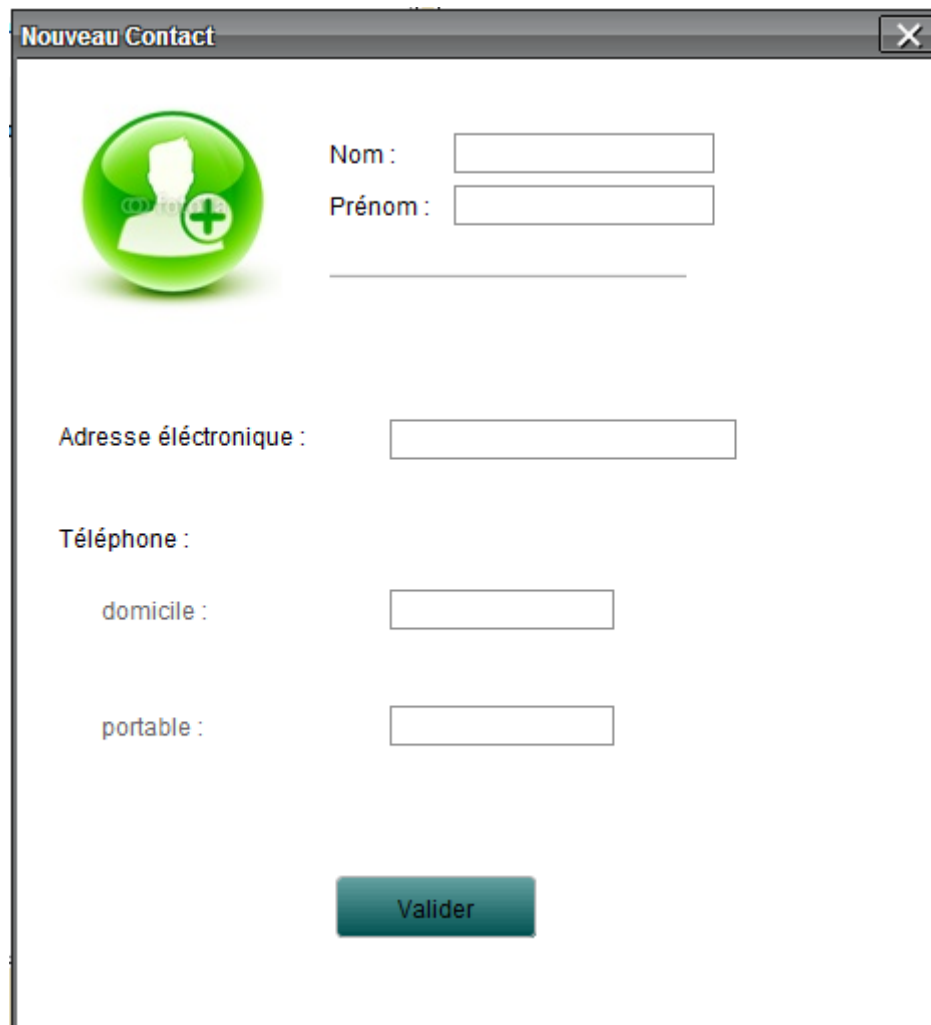


Figure 16 : Fenêtre du carnet d'adresses

La fenêtre suivante représente l'interface de l'ajout d'un nouveau contact



The screenshot shows a window titled "Nouveau Contact" with a close button (X) in the top right corner. On the left side, there is a green circular icon containing a white silhouette of a person and a plus sign. To the right of the icon, the form contains the following fields:

- Nom :** A text input field.
- Prénom :** A text input field.
- Adresse électronique :** A text input field.
- Téléphone :** A label followed by two text input fields:
 - domicile :** A text input field.
 - portable :** A text input field.

At the bottom center of the window is a green button labeled "Valider".

Figure 16 : Fenêtre d'ajout d'un contact

6. Conclusion :

Dans ce chapitre, nous avons présenté l'environnement de développement matériel et l'environnement logiciel avec lesquels ce projet a été réalisé. Nous avons présenté aussi une vue de l'application finale en utilisant quelques imprimés d'écrans.

Conclusion Générale

Ce projet était une bonne occasion pour sortir du cadre théorique et appliquer les connaissances acquises lors des deux années de classes préparatoires au sein de l'ESI dans un environnement réel de travail qui nous a permis de nous initier au domaine professionnel et d'apprendre plusieurs attitudes et habitudes sociales telles que le travail en groupe.

Sur le plan technique, ce travail nous a permis de nous approfondir dans l'apprentissage du langage Java, il nous a aussi fait découvrir de nouveaux concepts liés à la technologie de la messagerie électronique.

La contrainte de temps ainsi évoquée nous a empêchés d'ajouter d'autres fonctionnalités à notre application qu'on a estimée faisables techniquement, parmi lesquels nous citons la possibilité d'intégrer une section Recherche afin de rechercher les messages selon l'expéditeur ou bien la date par exemple.

Dans ce rapport nous avons développé trois chapitres. Dans le premier, nous avons étudié les différents protocoles et techniques utilisés dans la construction d'un client de messagerie électronique ainsi que la structure générale d'un message électronique. Ensuite, une analyse de l'application allant de l'étude de ses cas d'utilisation à la conception des différentes classes qui la composent a été élaborée dans le deuxième chapitre. Nous avons clôturé le rapport par la partie réalisation renfermant les principales interfaces de l'application.

Webographie :

- [W1] www.wikipedia.fr
- [W2] www.commentcamarche.fr
- [W3] www.siteduzero.com
- [W4] www.tech-tice.net/spip.php?article17
- [W5] www.developpez.net
- [W6] www.oracle.com
- [W7] fr.netbeans.org
- [W8] www.jtattoo.net