



Rapport de TPI - MailStorage

Maziero Marco – CIN4A

Lausanne – ETML - 2017

Durée : 5.5 semaines

Chef de projet : Melly Jonathan - jonathan.melly@vd.educanet2.ch

Expert N°1 : Montemayor Ernesto - ernesto@bati-technologie.ch

Expert N°2 : Moren Auxilio - auxilio.moren@orif.ch

Table des matières

1	SPÉCIFICATIONS.....	8
1.1	TITRE	8
1.2	DESCRIPTION	8
1.3	MATÉRIEL ET LOGICIELS À DISPOSITION	8
1.4	PRÉREQUIS.....	8
2	PLANIFICATION INITIALE.....	10
2.1	LISTE DES TÂCHES	10
2.2	DIAGRAMME DE GANTT	10
3	ANALYSE.....	12
3.1	OPPORTUNITÉS.....	12
3.1.1	<i>Utilisation du protocole IMAP</i>	<i>12</i>
	Solution	12
3.1.2	<i>Stockage de fichiers sur boîte mail</i>	<i>13</i>
	Solution	13
3.1.3	<i>Approfondissement des connaissances.....</i>	<i>14</i>
3.2	DOCUMENT D'ANALYSE ET CONCEPTION.....	15
3.2.1	<i>Architecture du projet</i>	<i>15</i>
	Classe LoginWindow	16
	Classe DbManager	16
	Classe MailStorage	16
	Classe statique MailManager.....	16
	Classe statique FilesManager	16
	Classe statique Globals	17
	Classe AppFile	17
3.2.2	<i>Interfaces.....</i>	<i>18</i>
3.2.3	<i>Vérification du formulaire de connexion</i>	<i>20</i>
	Vérification du nom du serveur IMAP	21
	Vérification du nom du numéro de port	21
	Vérification du mail	21
	Vérification du mot de passe	21
	Vérification du dossier racine	22
3.2.4	<i>Connexion IMAP</i>	<i>22</i>
3.2.5	<i>Interactions base de données.....</i>	<i>23</i>
3.2.6	<i>Détection des modifications de fichiers.....</i>	<i>24</i>
3.2.7	<i>Organisation des fichiers dans la boîte mail</i>	<i>25</i>
3.2.8	<i>Transformation des fichiers « dossier – boîte mail ».....</i>	<i>25</i>
3.2.9	<i>La synchronisation des fichiers.....</i>	<i>26</i>
	Les synchronisations initiales et au démarrage	27
3.2.10	<i>Diagramme de flux du fonctionnement de l'application</i>	<i>28</i>
3.3	CONCEPTION DES TESTS.....	29
3.3.1	<i>Scénario N°1 : Connexion</i>	<i>29</i>
3.3.2	<i>Scénario N°2 : Synchronisation</i>	<i>29</i>
3.3.3	<i>Scénario N°3 : Changement de compte.....</i>	<i>30</i>
3.3.4	<i>Scénario N°4 : Espace disponible.....</i>	<i>30</i>
3.3.5	<i>Scénario N°5 : Synchronisation initiale</i>	<i>30</i>
3.3.6	<i>Scénario N°6 : Création de fichiers</i>	<i>31</i>
3.3.7	<i>Scénario N°7 : Suppression de mail.....</i>	<i>31</i>
3.3.8	<i>Scénario N°8 : Suppression de fichier.....</i>	<i>31</i>
3.3.9	<i>Scénario N°9 : Édition de fichier</i>	<i>32</i>
3.3.10	<i>Scénario N°10 : Gestion des erreurs</i>	<i>32</i>

3.4	PLANIFICATION DÉTAILLÉE	33
3.4.1	Semaine 1	33
3.4.2	Semaine 2	34
3.4.3	Semaine 3	35
3.4.4	Semaine 4	36
3.4.5	Semaine 5	37
3.4.6	Semaine 6	38
4	RÉALISATION	40
4.1	DOSSIER DE RÉALISATION	40
4.1.1	Interface utilisateur	40
	Fenêtre de login	40
	Fenêtre de traitement des fichiers	44
4.1.2	Connexion au serveur IMAP et interactions boîte mail	48
4.1.3	Interactions avec la base de données	52
4.1.4	La synchronisation automatique des fichiers	54
	Mise à jour de la liste locale	59
	Mise à jour de la liste distante	60
	Envoi des nouveaux fichiers sur la boîte mail	61
	Suppression des fichiers de la boîte mail	62
4.1.5	Synchronisation initiale	64
4.1.6	Synchronisation au démarrage	66
4.1.7	Mail d'index des dossiers	67
4.1.8	Cryptage du mot de passe de l'utilisateur	68
4.2	DOSSIER DES TESTS	69
4.2.1	Scénario N°1 : Connexion	69
4.2.2	Scénario N°2 : Synchronisation	69
4.2.3	Scénario N°3 : Changement de compte	70
4.2.4	Scénario N°4 : Espace disponible	70
4.2.5	Scénario N°5 : Synchronisation initiale	71
4.2.6	Scénario N°6 : Création de fichiers	71
4.2.7	Scénario N°7 : Suppression de mail	72
4.2.8	Scénario N°8 : Suppression de fichier	72
4.2.9	Scénario N°9 : Édition de fichier	73
4.2.10	Scénario N°10 : Gestion des erreurs	73
4.2.11	Autres tests	74
5	CONCLUSION	76
5.1	BILAN DES FONCTIONNALITÉS DEMANDÉES	76
5.2	FONCTIONNALITÉS SUPPLÉMENTAIRES À IMPLÉMENTER	77
5.2.1	Gestion des fichiers de grande taille	77
5.2.2	Exécution simultanée sur plusieurs ordinateurs	78
5.3	BILAN PERSONNEL	79
6	DIVERS	82
6.1	JOURNAL DE TRAVAIL	82
6.1.1	Semaine 1	82
6.1.2	Semaine 2	83
6.1.3	Semaine 3	84
6.1.4	Semaine 4	85
6.1.5	Semaine 5	86
6.1.6	Semaine 6	87
6.2	GANTT FINAL	88
6.2.1	Semaine 1	88
6.2.2	Semaine 2	89

6.2.3	Semaine 3.....	90
6.2.4	Semaine 4.....	91
6.2.5	Semaine 5.....	92
6.2.6	Semaine 6.....	93
6.3	WEBOGRAPHIE	94
6.4	CAHIER DES CHARGES.....	95

Spécifications

1 SPÉCIFICATIONS¹

1.1 Titre

MailStorage

1.2 Description

MailStorage est une application qui permet d'utiliser l'espace disponible sur une boîte mail pour stocker des fichiers. Au démarrage un dossier racine est spécifié et l'utilisateur peut y prendre ou placer des fichiers. Le tout est ensuite synchronisé dans la boîte mail ce qui permet d'accéder aux fichiers depuis n'importe quel ordinateur ayant une connexion Internet.

L'avantage d'une telle solution permet de disposer d'une petite espace de stockage et de pouvoir y accéder depuis presque n'importe quel ordinateur. Cela offre aussi un avantage : il n'est pas nécessaire de créer un compte pour bénéficier de cet espace de stockage. En effet, une simple adresse mail suffit amplement.

1.3 Matériel et logiciels à disposition

Ci-dessous la liste du matériel à disposition pour le projet :

Ordinateur ETML

MacBook Pro personnel

Microsoft Office

Libre Office

Visual Studio Community 2017

ReSharper Ultimate

Un accès à Internet

Des modèles de fichiers pour la documentation

1.4 Prérequis

La réalisation de ce projet nécessite d'avoir acquis les compétences en programmation C# structurée, événementielle, orientée-objet et en gestion de projet. Toutes ces compétences ont été acquises lors des cours à l'ETML.

¹ Pour plus de détails, voir le cahier des charges en annexe

Planification initiale

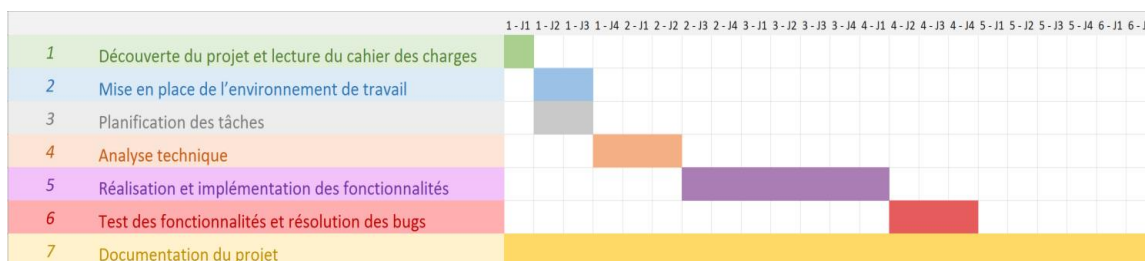
2 PLANIFICATION INITIALE

La planification initiale consiste à définir une liste de tâches représentant les grands thèmes du projet afin d'obtenir une vision d'ensemble. La planification permet ainsi d'évaluer les délais et les difficultés techniques du travail à fournir. Un diagramme de Gantt est aussi créé afin de donner un aspect visuel à la charge de travail.

2.1 Liste des tâches

N°	Description de la tâche
1	Découverte du projet et lecture du cahier des charges
2	Mise en place de l'environnement de travail
3	Planification des tâches
4	Analyse technique
5	Réalisation et implémentation des fonctionnalités
6	Test des fonctionnalités et résolution des bugs
7	Documentation du projet

2.2 Diagramme de Gantt



Les tâches sont réparties sur les 5.5 semaines de travail. Chaque semaine ne compte que 4 jours de travail consacrés au TPI. La planification détaillée qui se trouve dans l'analyse est basée sur cette planification initiale.

Analyse

3 ANALYSE

L'analyse est l'une des phases les plus importantes du projet, cette étape permet d'en éclaircir chaque partie technique, de décider de quelle façon le développement et la logique seront organisés et de trouver des solutions aux éventuelles difficultés algorithmiques ou technologiques.

3.1 Opportunités

3.1.1 Utilisation du protocole IMAP²

Une des composantes principales de ce projet est l'utilisation du protocole IMAP pour stocker et récupérer des fichiers depuis une boîte mail. Ce protocole représente un défi technique et doit être mis en place correctement afin d'éviter des erreurs de connexion ou d'authentification.



Le protocole IMAP, Internet Message Access Protocol, est l'un des protocoles utilisés par les services mail. Son utilité est de pouvoir récupérer des mails en se connectant sur une boîte mail.

Il existe deux protocoles permettant la récupération de mails : l'IMAP et le POP (Post Office Protocol). L'avantage de l'IMAP est qu'il permet la gestion des dossiers dans la boîte mail, contrairement au protocole POP qui ne permet que de récupérer la liste des mails présents dans la boîte de réception.³

Créé en 1986, le protocole IMAP n'a cessé d'évoluer et en est actuellement à sa version 4.

Solution

L'implémentation du protocole IMAP dans le projet peut se faire via la librairie MailKit. Cet API permet de créer des connexions IMAP4, POP3 et

² « Internet Message Access Protocol », https://fr.wikipedia.org/wiki/Internet_Message_Access_Protocol, 22.02.2017

³ « Post Office Protocol », https://fr.wikipedia.org/wiki/Post_Office_Protocol, 02.05.2017

SMTP vers un serveur mail spécifié en C#. Idéal pour l'envoi et la réception de mails.

3.1.2 Stockage de fichiers sur boîte mail

L'utilité primaire d'une boîte mail consiste à recevoir des messages d'autres personnes. Ces messages contiennent généralement un sujet, un contenu, l'adresse mail de l'émetteur, l'adresse du destinataire et parfois, le mail peut aussi être accompagné d'une pièce jointe.



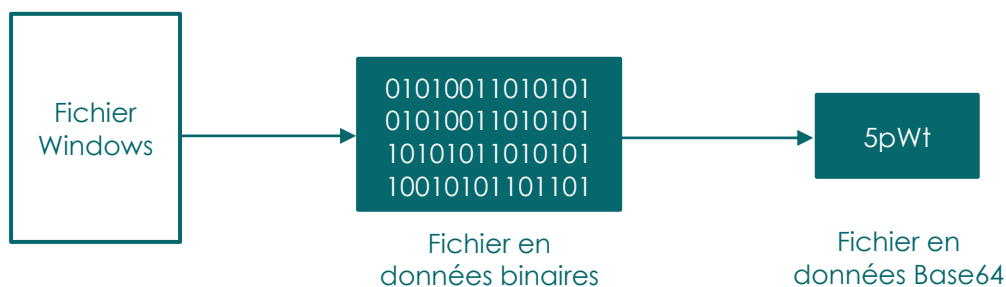
Cependant, il n'est pas impossible d'utiliser ces messages pour y joindre d'autres éléments, il n'y a pas de restrictions sur le texte présent dans le sujet et le contenu mis à part la taille du mail.

Dans ce projet, le but est de pouvoir exploiter la boîte mail de l'utilisateur afin d'y stocker des fichiers et leur arborescence. Cela peut être accompli de différentes façons.

Solution

Le but de ce programme est de pouvoir envoyer des fichiers sur la boîte mail afin de pouvoir y accéder depuis n'importe quel ordinateur.

Les fichiers une fois identifiés peuvent être convertis en binaire, puis en texte grâce à une fonction de conversion en Base64. La conversion du code binaire en Base64 transforme tous les « 1 et 0 » en caractères ASCII. Comme un caractère ASCII nécessite un groupe de 6 bits, cela permet d'avoir une chaîne de caractères sortante bien plus petite que si elle était en binaire.



Une fois le fichier transcrit en texte Base64, il peut être stocké dans le contenu d'un mail. Les autres informations du fichier telles que le nom, le chemin d'accès et les dates de création et de modification peuvent être stockées dans le sujet du message.



Pour la récupération des fichiers, il suffit simplement de télécharger le mail depuis le serveur IMAP, transcrire le contenu Base64 en fichier Windows et lui appliquer son nom ainsi que les dates de création et de modification.

3.1.3 Approfondissement des connaissances

Durant ce projet certaines compétences seront approfondies :

- Compétences en développement C#
- Développement orienté-objet
- Utilisation de la librairie SQLite
- Amélioration de la gestion de projet



3.2 Document d'analyse et conception

3.2.1 Architecture du projet

Le projet doit être capable d'accomplir certaines tâches et pour se faire, des classes spécifiques sont nécessaires :

- Une classe pour la fenêtre de connexion
- Une classe pour la fenêtre principale
- Une classe statique pour les variables globales
- Une classe statique pour communiquer en IMAP avec la boîte mail
- Une classe pour communiquer avec la base de données
- Une classe pour gérer les fichiers
- Une classe représentant un fichier

Les classes principales sont celles qui gèrent les fenêtres, c'est depuis ces éléments que les autres classes sont instanciées et appelées.

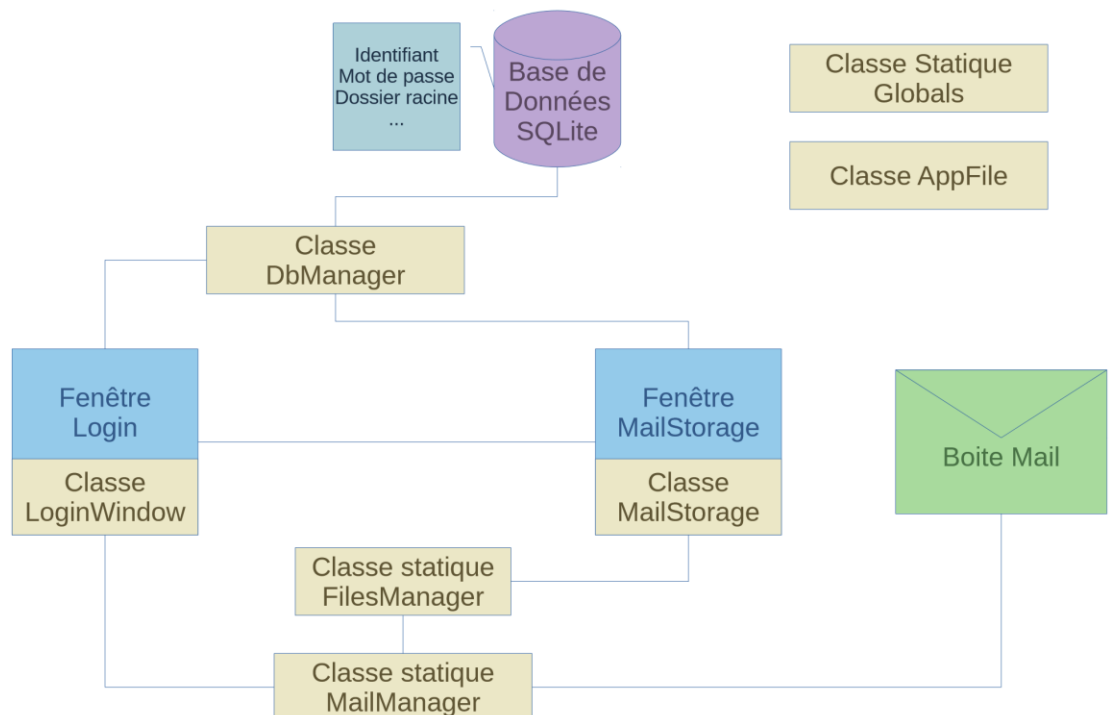


Figure 1 : Schéma des classes et des éléments de l'application

Classe LoginWindow

Cette classe représente la fenêtre de login, c'est dans ce code que se font les interactions avec la base de données. Les valeurs du formulaire de connexion présent dans la fenêtre sont ajoutées à la base de données si elle est vide. Si la base de données contient déjà des valeurs, celles-ci sont remplacées par les nouvelles.

Classe DbManager

Toutes les interactions directes avec la base de données sont faites dans cette classe. Les fonctions disponibles permettent d'ajouter et de retirer des données.

Classe MailStorage

La fenêtre principale est gérée par la classe *MailStorage*. C'est dans ce code que la vérification continue des fichiers s'exécute. Les fichiers ajoutés au dossier local sont envoyés à la boîte mail et les fichiers manquants dans le dossier local y sont rajoutés via la classe *FileManager*.

Classe statique MailManager

Pour envoyer et recevoir des mails, il faut une connexion IMAP constante. C'est le rôle de la classe *MailManager*. Les envois et réceptions de mail ainsi que leur contenu y est géré.

Au lancement du programme, une connexion IMAP est ouverte avec les données de la base de données afin de pouvoir prendre le contrôle de la boîte mail.

Classe statique FilesManager

Dans cette classe, les fichiers sont transformés en mails ou inversement. Cela permet d'obtenir les données dans la forme souhaitée.

Les fichiers sont transformés en texte Base64 qui sont ensuite placés dans le contenu du mail, tandis que les mails récupérés sont retransformés en fichiers afin d'être placés dans le dossier local.

C'est depuis cette classe que l'utilisation de la classe *MailManager* se fait. *FilesManager* se charge d'envoyer les données du fichier pour que l'envoi dans la boîte mail se fasse.

Classe statique *Globals*

Toutes les variables importantes qui doivent être accessibles partout dans le code se trouvent dans la classe statique *Globals*.

Classe *AppFile*

Cette classe représente un fichier. Elle est utilisée afin de disposer de tous les attributs des fichiers : nom, date de création, date de modification, chemin d'accès, etc.

Les instances de cette classe représentent les fichiers du dossier local et de la boîte mail. Elles sont stockées dans les listes qui répertorient tous les fichiers.

3.2.2 Interfaces

Le programme ne comporte que deux fenêtres car la majorité des manipulations se font dans l'explorateur Windows. La fenêtre principale est conçue pour tourner en arrière-plan afin que les fichiers se mettent à jour automatiquement.



La fenêtre de connexion IMAP est présentée avec un titre "Connexion IMAP". Elle contient quatre champs de saisie : "Serveur IMAP", "Port", "Adresse mail" et "Mot de passe". En dessous, il y a un champ "Dossier racine" et un bouton "Parcourir". Une barre verte en bas de la fenêtre contient le bouton "Suivant".

Figure 2 : Fenêtre de connexion

La première fenêtre s'affichant au démarrage de l'application est celle de connexion.

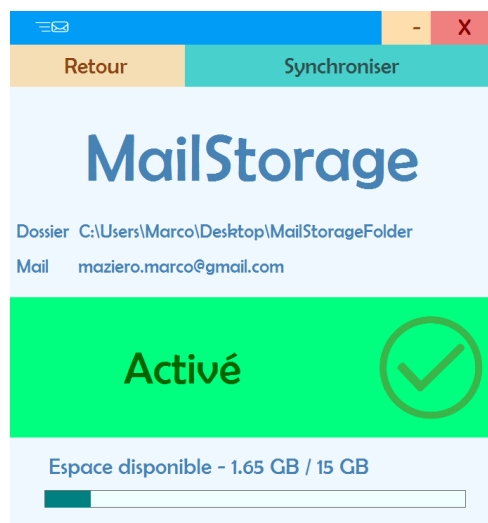
C'est dans cette fenêtre que les champs de configuration du serveur IMAP, d'adresse mail, de mot de passe et de dossier racine sont spécifiés.

Une action sur le bouton suivant lance la tentative de connexion IMAP avec la boîte mail.

La deuxième fenêtre représente la connexion active de l'application.

Les seules actions possibles sont le retour au login et la fonction de synchronisation manuelle.

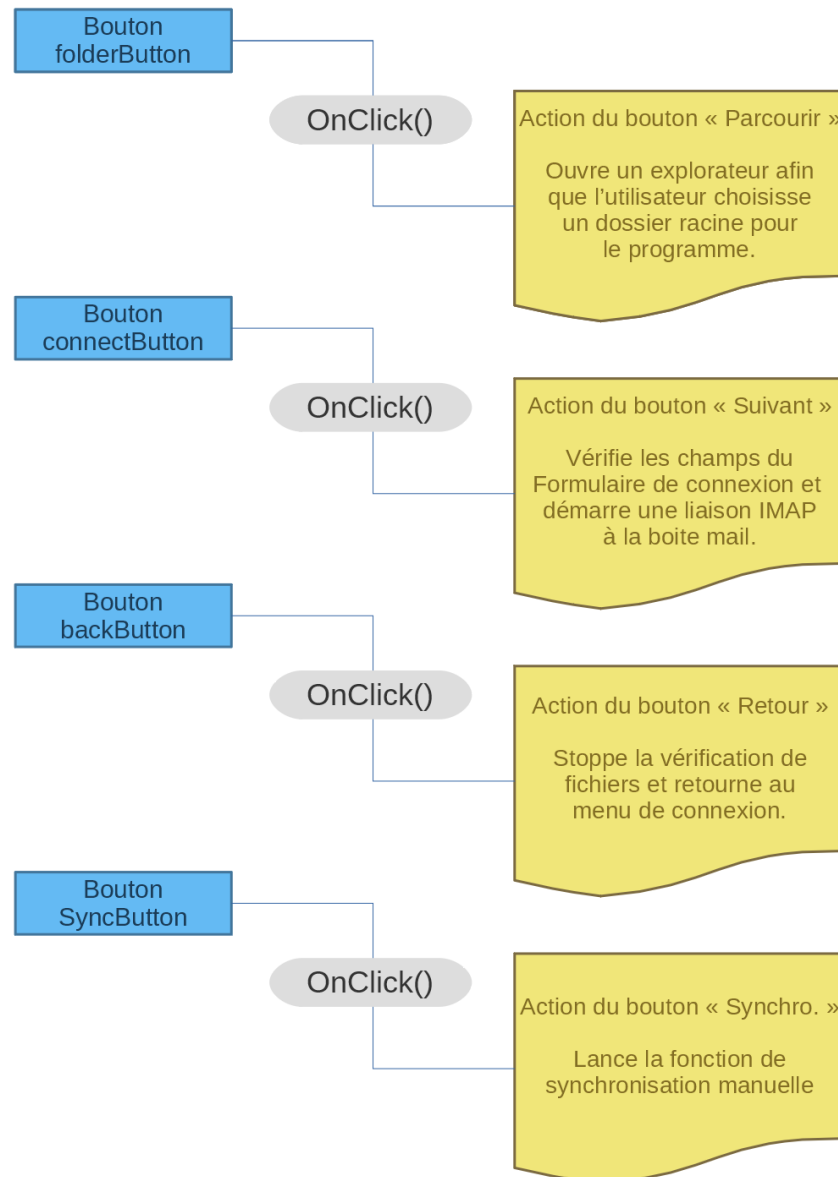
Le statut de la connexion est affiché afin de toujours avoir une confirmation du fonctionnement du programme.



La fenêtre de traitement des fichiers est présentée avec un titre "MailStorage". Elle contient deux boutons : "Retour" et "Synchroniser". En dessous, il y a un champ "Dossier" et un champ "Mail". Une barre verte en bas de la fenêtre contient le statut "Activé" et un bouton "Synchroniser".

Figure 3 : Fenêtre de traitement des fichiers

Les interactions avec l'interface ne sont pas très nombreuses. Ce ne sont que des boutons qui réagissent au clic. Le schéma événementiel ci-dessous décrit les interactions possibles avec l'interface et les actions que cela produit.



Sur la première fenêtre de connexion, deux boutons sont présents : le bouton de sélection du dossier racine et le bouton de connexion à la boîte mail.

Sur la deuxième fenêtre, les boutons disponibles sont : le bouton de retour à la fenêtre de connexion et le bouton servant à lancer une synchronisation IMAP manuelle.

3.2.3 Vérification du formulaire de connexion

La première étape lors de l'utilisation du programme est la connexion en IMAP au serveur mail. Pour ce faire, l'utilisateur doit entrer plusieurs données dans un petit formulaire afin que le programme ait les valeurs nécessaires à la connexion.

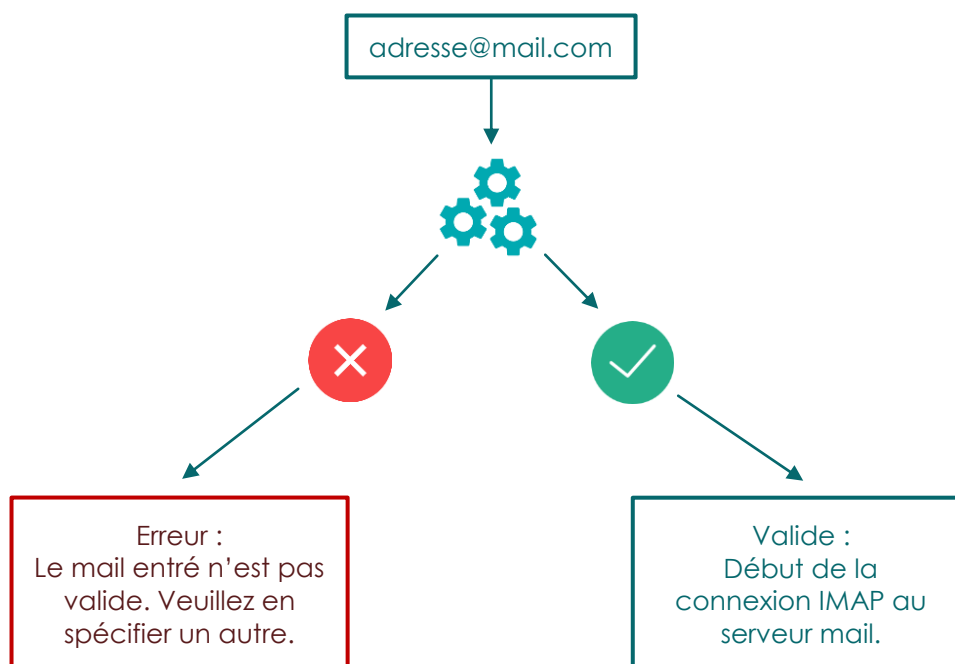
Pour commencer, l'utilisateur entre le nom du serveur IMAP ainsi que le port correspondant.

Ensuite, il entre son adresse mail, le mot de passe de son compte mail et sélectionne le dossier racine ou il veut gérer ses fichiers.

Enfin, la confirmation se fait en cliquant sur le bouton « Suivant ».



Les valeurs sont ensuite vérifiées dans le programme afin de s'assurer que l'utilisateur a bien entré des données valides. Dans le cas où la validation retourne une erreur, un message d'avertissement s'affiche à l'écran pour avertir l'utilisateur.



Vérification du nom du serveur IMAP

Le nom du serveur IMAP ne possède pas beaucoup de vérifications. Le programme vérifie juste que le champ de formulaire n'est pas vide. Dans le cas contraire, un message d'erreur est affiché.

Généralement un nom de serveur IMAP est composé de trois ou quatre parties. Par exemple, le nom du serveur IMAP de Gmail est « imap.gmail.com ».

Vérification du nom du numéro de port

Comme pour la vérification du nom du serveur IMAP, celle pour le numéro de port ne vérifie que si le champ de formulaire n'est pas vide et affiche un message d'erreur si tel est le cas.

Vérification du mail

Pour être valide, une adresse mail doit respecter plusieurs conditions :

- Posséder un symbole « @ »
- Au moins 2 caractères avant le symbole « @ »
- Au moins 2 caractères après le symbole « @ »
- Un caractère « . » après le symbole « @ »
- Une extension après le caractère « . »

adresse@mail.com

Vérification du mot de passe

Aucune vérification particulière n'est effectuée sur le mot de passe si ce n'est que le champ de formulaire ne doit pas être vide.

Le mot de passe spécifié doit correspondre au compte mail afin que le programme puisse ensuite s'authentifier auprès du serveur IMAP et puisse lancer la connexion qui permettra de transférer les fichiers.

Vérification du dossier racine

La première vérification est s'assurer simplement que la valeur n'est pas vide et qu'elle contient du texte.

Dans ce champ, l'utilisateur ne peut rentrer des données manuellement. Pour remplir le champ il faut cliquer sur le bouton « Parcourir » afin qu'une fenêtre d'explorateur Windows s'ouvre pour que l'utilisateur puisse choisir le dossier racine qu'il souhaite.

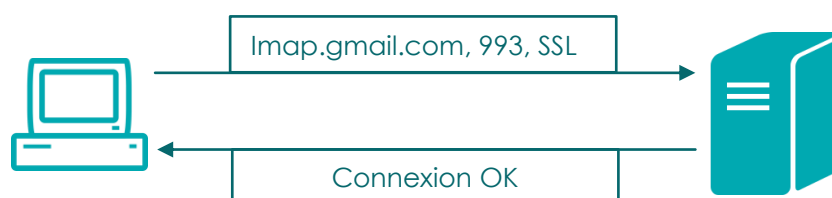
L'ouverture d'un explorateur Windows est effectuée grâce à la classe `FolderBrowserDialog` disponible dans le namespace `System.Windows.Forms`.⁴

Le chemin du dossier est ensuite récupéré puis stocké dans la base de données.

3.2.4 Connexion IMAP

Une fois les données validées, le programme doit tenter de se connecter au serveur mail. La classe `MailManager` est instanciée afin de disposer des moyens nécessaires à la connexion. La librairie utilisée est MailKit. Elle permet de communiquer en SMTP, POP3 ou IMAP4 avec les serveurs mail.

Tout d'abord le programme commence par établir une connexion sur le serveur mail, il s'annonce en spécifiant le nom du serveur, le port à utiliser et si la connexion est du type SSL. Dans la plupart des serveurs, le port IMAP utilisé est le 993 pour une connexion SSL.



Ensuite, une demande d'authentification est lancée avec les identifiants de l'utilisateur (adresse mail et mot de passe). Le serveur répond soit avec un OK soit avec un refus.



⁴ « Browse for a directory in C# », <http://stackoverflow.com/questions/11767/browse-for-a-directory-in-c-sharp>, 08.08.2014

Lorsque le programme est connecté et authentifié, il peut accéder, créer ou supprimer des mails dans la boîte mail de l'utilisateur. La connexion IMAP est établie.

3.2.5 Interactions base de données

La base de données utilisée dans ce programme est très petite, elle ne comporte qu'une seule table de données.

	appData	
1	servername	varchar
2	serverport	varchar
3	usermail	varchar
4	userpassword	varchar
5	rootpath	varchar

La table appData est composée de 5 champs. Tout d'abord, les deux champs concernant le nom et le port du serveur IMAP.

Ensuite viennent les champs de l'utilisateur qui stockent son adresse mail et son mot de passe crypté ainsi que le dossier racine

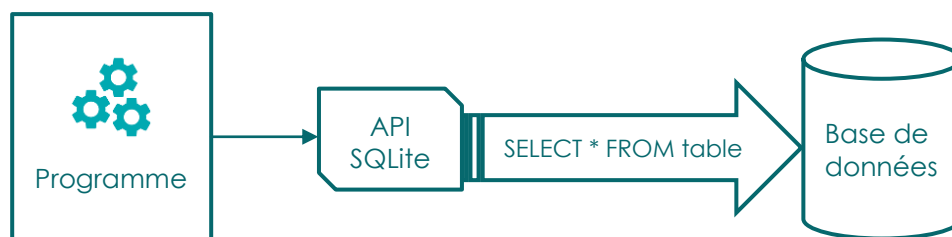
choisi.

L'utilisation d'une base de données pour si peu de valeurs se justifie par le fait que cela permet l'ajout d'autres fonctionnalités si le temps le permet (la gestion de plusieurs utilisateurs par exemple) ou si le projet est ensuite réutilisé par une autre personne.

En effet, un simple fichier texte aurait suffi pour réaliser le travail demandé. Cependant, n'ayant jamais utilisé de base de données SQLite, ce projet est un bon moyen de découvrir le fonctionnement de cette librairie.

Les interactions avec la base de données se font via la classe *DbManager*. C'est le seul endroit du code qui va directement interagir avec les données stockées dans la base de données pour les retransmettre ou les insérer.

L'API SQLite est une librairie qui sert d'intermédiaire entre le code C# et la base de données. Les requêtes sur la base de données sont écrites en SQL, ce qui permet de posséder toutes les fonctionnalités d'une base de données intégrée à un programme C#.

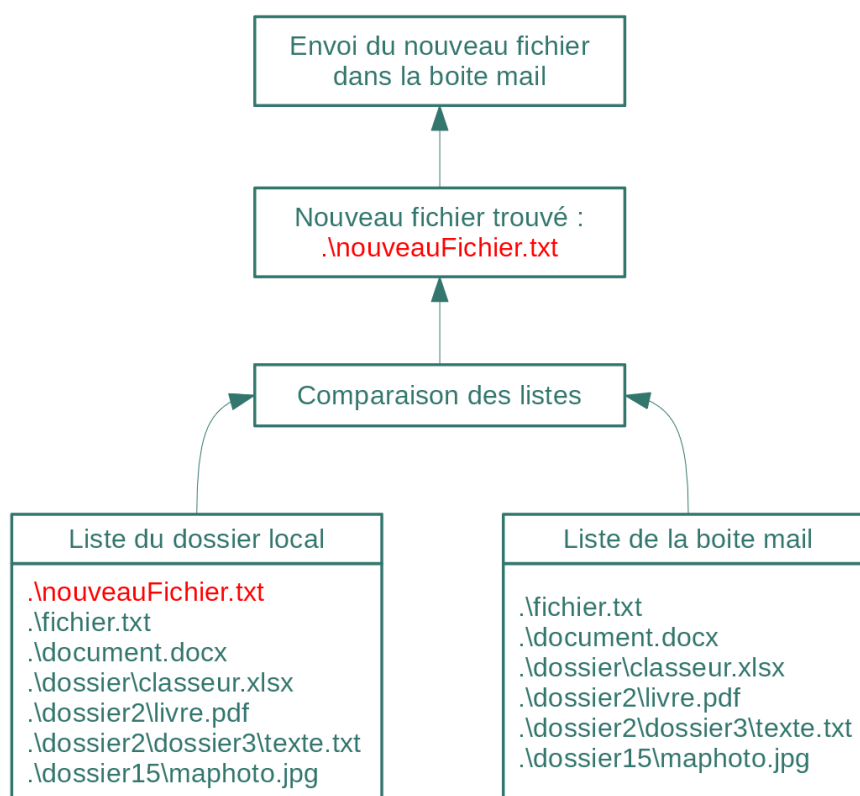


3.2.6 Détection des modifications de fichiers

La détection des changements dans les fichiers de l'arborescence est l'une des fonctionnalités principales du programme. C'est grâce à cette détection que les fichiers sont placés et mis à jour en même temps dans le dossier local et la boîte mail de l'utilisateur.

La détection des fichiers fonctionne à base de listes de fichiers. Lorsqu'un fichier est ajouté, soit dans le dossier local, soit dans la boîte mail, les listes sont comparées et l'élément manquant est placé dans le dossier ou la boîte mail.

Lors de l'exécution du programme, deux listes sont actives en permanence : la liste des fichiers du dossier local et la liste des fichiers présents dans la boîte mail. Ces listes contiennent les chemins d'accès aux fichiers et leur date de modification, ce qui permet de connaître l'arborescence totale du dossier racine ainsi que le moment où un fichier est modifié, supprimé, renommé ou ajouté.

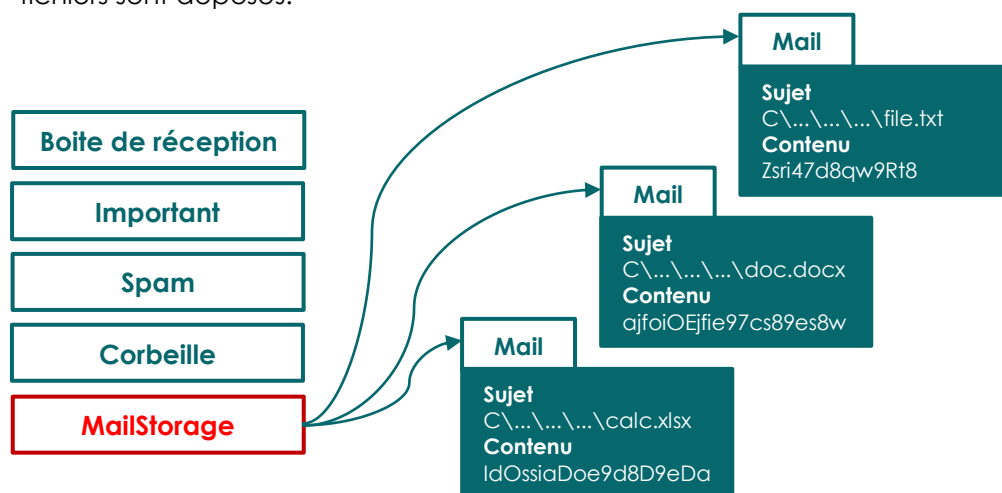


Lorsqu'un fichier est ajouté dans le dossier racine (voir illustration ci-dessus), la liste va récupérer son chemin et lorsque la comparaison des deux listes

sera effectuée, le fichier sera indiqué comme nouveau et sera envoyé sur la boîte mail afin que les deux listes s'égalisent et continuent à posséder les mêmes valeurs.

3.2.7 Organisation des fichiers dans la boîte mail

Afin de ne pas désorganiser complètement la boîte mail de l'utilisateur, l'application crée un dossier « MailStorage » dans la boîte mail où tous les fichiers sont déposés.

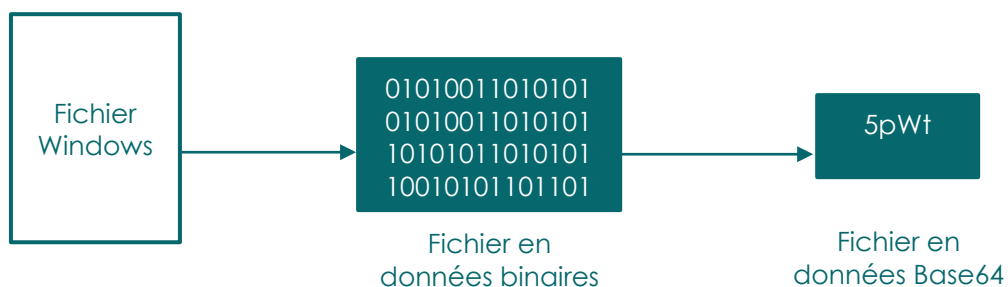


Les fichiers déposés dans le dossier mail « MailStorage » sont tous composés de la même façon : le sujet du mail contient les informations du fichier dans le dossier local (nom, chemin d'accès, date de création et de modification) et le contenu contient le fichier complet sous forme de texte.

3.2.8 Transformation des fichiers « dossier – boîte mail »

Les fichiers qui se trouvent dans le dossier local et qui doivent être placés dans la boîte mail sont récupérés via la comparaison des listes (voir section sur la détection des modifications de fichiers).

Une fois ces fichiers récupérés, ils sont transformés en Base64. La transformation en Base64 permet d'obtenir une chaîne de caractères correspondant aux données binaires du fichier.



Ce texte est ensuite placé dans la boîte mail, dans le contenu d'un nouveau mail situé dans le dossier « MailStorage ». Les informations du fichier sont quant à elles placées dans le sujet du mail sous cette forme :

Sujet du mail					
Fichier.txt	::	.\dossier\fichier.txt	::	13.06.17	:: 15.06.17

3.2.9 La synchronisation des fichiers

Le rôle de la fonction de synchronisation consiste à reprendre tous les éléments du processus de transformation et de détection des fichiers.

Premièrement, la détection des fichiers manquants dans le dossier local et la boîte mail s'effectue. Cela permet d'identifier quels fichiers il faut déplacer.

Ensuite, s'il manque des fichiers dans la boîte mail, ces fichiers sont récupérés dans le dossier local, puis un par un, ils sont transformés en texte Base64 puis placés dans un mail dans la boîte mail.

Dans le cas où un fichier a été supprimé du dossier local, sa suppression est répliquée dans la boîte mail. Le mail correspondant au fichier est donc supprimé.

Lors de l'exécution du programme, ce sont les manipulations du dossier local qui ont la priorité sur la boîte mail.

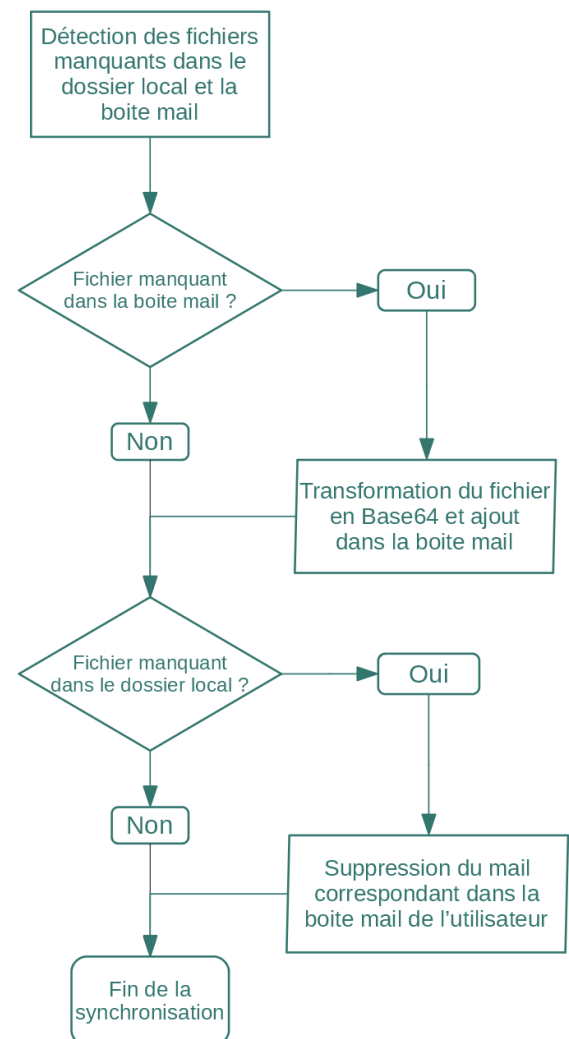


Figure 4 : Diagramme de flux du fonctionnement de la synchronisation

Les synchronisations initiales et au démarrage

Les synchronisations dites « initiales » et « au démarrage » sont deux types de synchronisations qui diffèrent de celle par défaut. En effet, ces deux processus vont donner la priorité à la boîte mail plutôt qu'au dossier local. Cela permet de mettre à jour l'état du dossier racine lorsque celui-ci vient d'être spécifié ou que son état n'est plus à jour.

Les deux listes de fichiers sont donc comparées mais les fichiers qui sont dans la boîte mail et absents du dossier local sont téléchargés tandis que les fichiers absents de la boîte mail et présents dans le dossier local sont supprimés.

Cela permet à l'application d'être utilisée sur différents ordinateurs en ayant toujours les fichiers ajoutés sur d'autres ordinateurs à disposition.

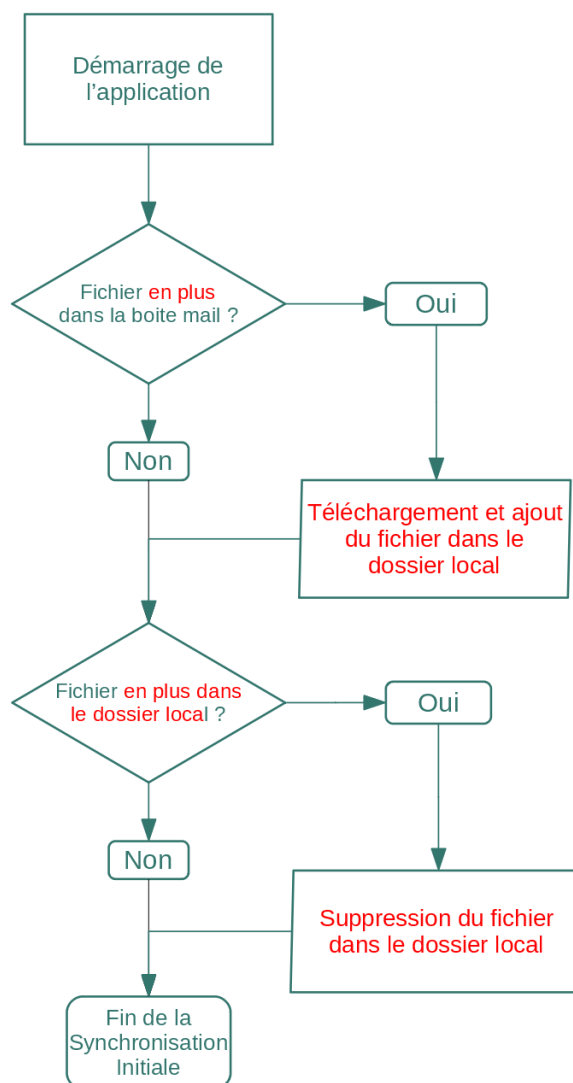
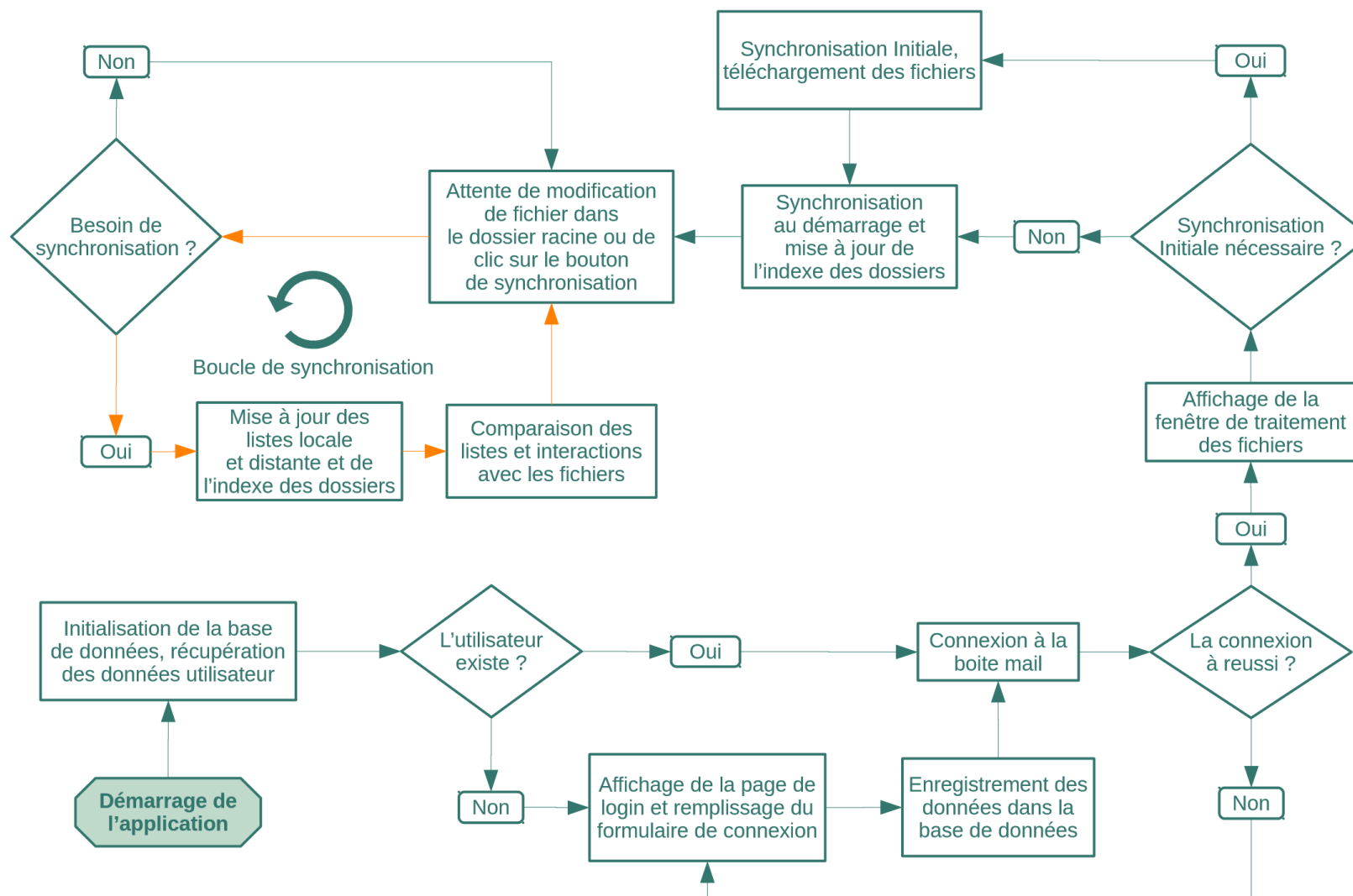


Figure 5 : Diagramme de flux montrant le fonctionnement de la synchronisation initiale

3.2.10 Diagramme de flux du fonctionnement de l'application



3.3 Conception des tests

3.3.1 Scénario N°1 : Connexion

Ce test consiste à fournir une adresse mail, un mot de passe et un dossier racine à l'application afin qu'elle établisse une connexion à la boîte mail de l'utilisateur. La connexion doit se faire sans erreur et la fenêtre de traitement des fichiers doit s'afficher correctement.

Étape	Description
Entrée	Fenêtre de connexion à l'application
Quand	Saisie du compte mail, du mot de passe et du dossier racine
Résultat	Connexion au compte mail et affichage de la fenêtre de traitement des fichiers

3.3.2 Scénario N°2 : Synchronisation

Le test de vérification du bouton synchroniser s'effectue afin de vérifier que la synchronisation « dossier local – boîte mail » se fait sans erreur. Le résultat doit être visible lorsque la boîte mail et le dossier racine possèdent les mêmes fichiers.

Étape	Description
Entrée	Bouton de synchronisation manuelle des fichiers
Quand	Clic sur le bouton de synchronisation
Résultat	Mise à jour et synchronisation des fichiers « dossier - boîte mail »

3.3.3 Scénario N°3 : Changement de compte

L'utilité de ce test est la vérification du bon fonctionnement de la base de données. Une fois un utilisateur enregistré, une autre adresse mail et mot de passe doivent être insérés dans le programme et la connexion doit se faire avec la nouvelle boîte mail. Lors du prochain lancement de l'application, les nouveaux identifiants doivent se placer automatiquement dans les champs car ils auront été enregistrés dans la base de données.

Étape	Description
Entrée	Fenêtre de connexion à l'application
Quand	Changement du compte mail, mot de passe et dossier racine
Résultat	Connexion au compte mail et affichage de la fenêtre de traitement des fichiers

3.3.4 Scénario N°4 : Espace disponible

L'affichage de l'espace disponible doit correspondre à l'espace réellement disponible dans la boîte mail de l'utilisateur.

Étape	Description
Entrée	Fenêtre de traitement des fichiers
Quand	À tout moment
Résultat	L'espace disponible affiché correspond à l'espace disponible dans la boîte mail

3.3.5 Scénario N°5 : Synchronisation initiale

Après le lancement de l'application, les fichiers absents du dossier local doivent être ajoutés depuis les mails présents dans la boîte mail de l'utilisateur. Au final, les fichiers du dossier local doivent être les mêmes que ceux de la boîte mail.

Étape	Description
Entrée	Dossier local
Quand	Après connexion à la boîte mail
Résultat	Création automatique des fichiers depuis la boîte mail

3.3.6 Scénario N°6 : Création de fichiers

Une fois l'application connectée à la boîte mail. Tout fichier ajouté dans le dossier local doit être détecté par le programme et ajouté sous forme de mail dans la boîte mail de l'utilisateur.

Étape	Description
Entrée	Dossier local
Quand	Ajout d'un fichier dans le dossier
Résultat	Ajout automatique du fichier dans la boîte mail

3.3.7 Scénario N°7 : Suppression de mail

Lorsqu'un mail de fichier est supprimé, le programme en cours d'exécution doit alors détecter la faille et recréer un nouveau mail avec le fichier concerné.

Étape	Description
Entrée	Boîte mail
Quand	Suppression d'un mail de fichier
Résultat	Recréation automatique du mail dans la boîte mail

3.3.8 Scénario N°8 : Suppression de fichier

Lorsqu'un fichier est supprimé du dossier local, le programme doit également identifier le mail correspondant à ce fichier et le supprimer afin qu'il ne reste plus aucune trace du fichier.

Étape	Description
Entrée	Dossier local
Quand	Suppression d'un fichier dans le dossier local
Résultat	Suppression automatique dans la boîte mail

3.3.9 Scénario N°9 : Édition de fichier

Si un fichier est modifié depuis le dossier local, la nouvelle version doit venir remplacer l'ancienne dans la boîte mail. L'ancienne version correspondante est identifiée et supprimée pour faire place à la nouvelle version.

Étape	Description
Entrée	Dossier local
Quand	Édition d'un fichier dans le dossier local
Résultat	Mise à jour automatique du fichier dans la boîte mail

3.3.10 Scénario N°10 : Gestion des erreurs

Si une fausse adresse mail ou un mauvais mot de passe sont entrés dans le formulaire de connexion, le programme doit être capable de gérer les exceptions et afficher un message d'erreur indiquant le problème à résoudre.

Étape	Description
Entrée	Fenêtre de connexion à l'application
Quand	Insertion d'une fausse adresse mail et mauvais mot de passe
Résultat	Affichage des messages d'erreur

3.4 Planification détaillée

3.4.1 Semaine 1

Tâches - objectifs	Nb 1/4 heure	S 1																											
Absence - Imprévus	0																												
Lancement : Lecture du cahier des charges	9																												
Lancement : Création et rédaction de la planification	12																												
Lancement : Mise en place de l'environnement	9																												
Analyse : Recherche des technologies utilisables	6																												
Analyse : Architecture du projet	6																												
Analyse : Fonctionnalité de connexion IMAP	6																												
Analyse : Détection des changements sur les fichiers	9																												
Analyse : Organisation des fichiers dans la boîte mail	6																												
Analyse : Fonctionnalité de synchronisation	12																												
Analyse : Interface utilisateur	3																												
Implémentation : Création de l'interface utilisateur	12																												
Implémentation : Connexion au serveur IMAP	12																												
Implémentation : Scan des fichiers modifiés	15																												
Implémentation : Ajout des fichiers modifiés à la boîte	18																												
Implémentation : Téléchargement des fichiers plus récents	18																												
Test : Préparation des tests	12																												
Test : Exécution des tests	18																												
Documentation : Rédaction du rapport de projet	198																												
Documentation : Rédaction du journal de travail	0																												
Autres : Résolution de bug	21																												
Autres : Déploiement de l'application	15																												
Autres : Travail en plus	21																												
Autres : Congé / Vacances / Examens	83																												
Total planifié	521																												
Total réalisé	0																												

Tâches (chronologique)	Durée [1/4h]
Lancement : Lecture du cahier des charges	9
Lancement : Création et rédaction de la planification initiale	12
Documentation : Rédaction du rapport de projet	9
Lancement : Mise en place de l'environnement	9
Analyse : Recherche des technologies utilisables	6
Analyse : Architecture du projet	6
Documentation : Rédaction du rapport de projet	6
Analyse : Fonctionnalité de connexion IMAP	6
Documentation : Rédaction du rapport de projet	6
Analyse : Détection des changements sur les fichiers	9
Documentation : Rédaction du rapport de projet	21

3.4.2 Semaine 2

Tâches - objectifs	Nb 1/4 heure	S 2																											
Absence - Imprévus	0																												
	0																												
Lancement : Lecture du cahier des charges	9																												
	0																												
Lancement : Création et rédaction de la planification	12																												
	0																												
Lancement : Mise en place de l'environnement	9																												
	0																												
Analyse : Recherche des technologies utilisables	6																												
	0																												
Analyse : Architecture du projet	6																												
	0																												
Analyse : Fonctionnalité de connexion IMAP	6																												
	0																												
Analyse : Détection des changements sur les fichiers	9																												
	0																												
Analyse : Organisation des fichiers dans la boîte mail	6																												
	0																												
Analyse : Fonctionnalité de synchronisation	12																												
	0																												
Analyse : Interface utilisateur	3																												
	0																												
Implémentation : Création de l'interface utilisateur	12																												
	0																												
Implémentation : Connexion au serveur IMAP	12																												
	0																												
Implémentation : Scan des fichiers modifiés	15																												
	0																												
Implémentation : Ajout des fichiers modifiés à la boîte mail	18																												
	0																												
Implémentation : Téléchargement des fichiers plus récents	18																												
	0																												
Test : Préparation des tests	12																												
	0																												
Test : Exécution des tests	18																												
	0																												
Documentation : Rédaction du rapport de projet	198																												
	0																												
Documentation : Rédaction du journal de travail	0																												
	0																												
Autres : Résolution de bug	21																												
	0																												

Tâches (chronologique)	Durée [1/4h]
Analyse : Organisation des fichiers dans la boîte mail	6
Documentation : Rédaction du rapport de projet	6
Analyse : Fonctionnalité de synchronisation	12
Documentation : Rédaction du rapport de projet	6
Analyse : Interface utilisateur	3
Documentation : Rédaction du rapport de projet	3
Test : Préparation des tests	12
Implémentation : Création de l'interface utilisateur	12
Implémentation : Connexion au serveur IMAP	12
Documentation : Rédaction du rapport de projet	12
Implémentation : Scan des fichiers modifiés	15

3.4.3 Semaine 3

Tâches - objectifs	Nb 1/4 heure	S 3																			
Absence - Imprévus	0																				
	0																				
Lancement : Lecture du cahier des charges	9																				
	0																				
Lancement : Création et rédaction de la planification	12																				
	0																				
Lancement : Mise en place de l'environnement	9																				
	0																				
Analyse : Recherche des technologies utilisables	6																				
	0																				
Analyse : Architecture du projet	6																				
	0																				
Analyse : Fonctionnalité de connexion IMAP	6																				
	0																				
Analyse : Détection des changements sur les fichiers	9																				
	0																				
Analyse : Organisation des fichiers dans la boîte mail	6																				
	0																				
Analyse : Fonctionnalité de synchronisation	12																				
	0																				
Analyse : Interface utilisateur	3																				
	0																				
Implémentation : Création de l'interface utilisateur	12																				
	0																				
Implémentation : Connexion au serveur IMAP	12																				
	0																				
Implémentation : Scan des fichiers modifiés	15																				
	0																				
Implémentation : Ajout des fichiers modifiés à la boîte mail	18																				
	0																				
Implémentation : Téléchargement des fichiers plus récents	18																				
	0																				
Test : Préparation des tests	12																				
	0																				
Test : Exécution des tests	18																				
	0																				
Documentation : Rédaction du rapport de projet	198																				
	0																				
Documentation : Rédaction du journal de travail	0																				
	0																				
Autres : Résolution de bug	21																				
	0																				
Autres : Déploiement de l'application	15																				
	0																				
Autres : Travail en plus	21																				
	0																				
Autres : Congé / Vacances / Examens	83																				
	0																				
Total planifié	521																				
Total réalisé	0																				

Tâches (chronologique)

Durée

[1/4h]

Documentation : Rédaction du rapport de projet

12

Implémentation : Ajout des fichiers modifiés à la boîte mail

18

Documentation : Rédaction du rapport de projet

9

Implémentation : Téléchargement des fichiers plus récents en local

18

Documentation : Rédaction du rapport de projet

9

Test : Exécution des tests

18

Documentation : Rédaction du rapport de projet

15

3.4.4 Semaine 4

Tâches - objectifs	Nb 1/4 heure	S 4																							
Absence - Imprévus	0																								
	0																								
Lancement : Lecture du cahier des charges	9																								
	0																								
Lancement : Création et rédaction de la planification	12																								
	0																								
Lancement : Mise en place de l'environnement	9																								
	0																								
Analyse : Recherche des technologies utilisables	6																								
	0																								
Analyse : Architecture du projet	6																								
	0																								
Analyse : Fonctionnalité de connexion IMAP	6																								
	0																								
Analyse : Détection des changements sur les fichiers	9																								
	0																								
Analyse : Organisation des fichiers dans la boîte mail	6																								
	0																								
Analyse : Fonctionnalité de synchronisation	12																								
	0																								
Analyse : Interface utilisateur	3																								
	0																								
Implémentation : Création de l'interface utilisateur	12																								
	0																								
Implémentation : Connexion au serveur IMAP	12																								
	0																								
Implémentation : Scan des fichiers modifiés	15																								
	0																								
Implémentation : Ajout des fichiers modifiés à la boîte mail	18																								
	0																								
Implémentation : Téléchargement des fichiers plus récents	18																								
	0																								
Test : Préparation des tests	12																								
	0																								
Test : Exécution des tests	18																								
	0																								
Documentation : Rédaction du rapport de projet	198																								
	0																								
Documentation : Rédaction du journal de travail	0																								
	0																								
Autres : Résolution de bug	21																								
	0																								
Autres : Déploiement de l'application	15																								
	0																								
Autres : Travail en plus	21																								
	0																								
Autres : Congé / Vacances / Examens	83																								
	0																								
Total planifié	521																								
Total réalisé	0																								

Tâches (chronologique)

Durée

[1/4h]

Autres : Résolution de bug

21

Documentation : Rédaction du rapport de projet

21

Autres : Congé / Vacances / Examens

57

3.4.5 Semaine 5

Tâches - objectifs	Nb 1/4 heure	S 5																			
Absence - Imprévu	0																				
Lancement : Lecture du cahier des charges	0																				
	9																				
Lancement : Création et rédaction de la planification	0																				
	12																				
Lancement : Mise en place de l'environnement	0																				
	9																				
Analyse : Recherche des technologies utilisables	0																				
	6																				
Analyse : Architecture du projet	0																				
	6																				
Analyse : Fonctionnalité de connexion IMAP	0																				
	6																				
Analyse : Détection des changements sur les fichiers	0																				
	9																				
Analyse : Organisation des fichiers dans la boîte mail	0																				
	6																				
Analyse : Fonctionnalité de synchronisation	0																				
	12																				
Analyse : Interface utilisateur	0																				
	3																				
Implémentation : Création de l'interface utilisateur	0																				
	12																				
Implémentation : Connexion au serveur IMAP	0																				
	12																				
Implémentation : Scan des fichiers modifiés	0																				
	15																				
Implémentation : Ajout des fichiers modifiés à la boîte mail	0																				
	18																				
Implémentation : Téléchargement des fichiers plus récents	0																				
	18																				
Test : Préparation des tests	0																				
	12																				
Test : Exécution des tests	0																				
	18																				
Documentation : Rédaction du rapport de projet	0																				
	198																				
Documentation : Rédaction du journal de travail	0																				
	0																				
Autres : Résolution de bug	0																				
	21																				
Autres : Déploiement de l'application	0																				
	15																				
Autres : Travail en plus	0																				
	21																				
Autres : Congé / Vacances / Examens	0																				
	83																				
	0																				
Total planifié	521																				
Total réalisé	0																				

Tâches (chronologique)

Durée

[1/4h]

Autres : Congé / Vacances / Examens

13

Autres : Déploiement de l'application

15

Autres : Travail en plus

21

Documentation : Rédaction du rapport de projet

50

3.4.6 Semaine 6

Tâches - objectifs	Nb 1/4 heure	S 6																											
Absence - Imprévus	0																												
	0																												
Lancement : Lecture du cahier des charges	9																												
	0																												
Lancement : Création et rédaction de la planification	12																												
	0																												
Lancement : Mise en place de l'environnement	9																												
	0																												
Analyse : Recherche des technologies utilisables	6																												
	0																												
Analyse : Architecture du projet	6																												
	0																												
Analyse : Fonctionnalité de connexion IMAP	6																												
	0																												
Analyse : Détection des changements sur les fichiers	9																												
	0																												
Analyse : Organisation des fichiers dans la boîte mail	6																												
	0																												
Analyse : Fonctionnalité de synchronisation	12																												
	0																												
Analyse : Interface utilisateur	3																												
	0																												
Implémentation : Création de l'interface utilisateur	12																												
	0																												
Implémentation : Connexion au serveur IMAP	12																												
	0																												
Implémentation : Scan des fichiers modifiés	15																												
	0																												
Implémentation : Ajout des fichiers modifiés à la boîte mail	18																												
	0																												
Implémentation : Téléchargement des fichiers plus récents	18																												
	0																												
Test : Préparation des tests	12																												
	0																												
Test : Exécution des tests	18																												
	0																												
Documentation : Rédaction du rapport de projet	198																												
	0																												
Documentation : Rédaction du journal de travail	0																												
	0																												
Autres : Résolution de bug	21																												
	0																												

Tâches (chronologique)

Durée

[1/4h]

Autres : Congé / Vacances / Examens

13

Documentation : Rédaction du rapport de projet

13

Réalisation

4 RÉALISATION

Dans cette section, toutes les étapes de la réalisation sont reprises et expliquées le plus techniquement possible. Chaque fonction est reprise et son fonctionnement expliqué, chaque difficulté technique rencontrée est détaillée dans le dossier de réalisation.

C'est aussi dans cette section que tous les tests sont effectués et retranscrits. Le dossier de tests contient tous les tests créés lors de l'analyse et réalisés sur le projet.

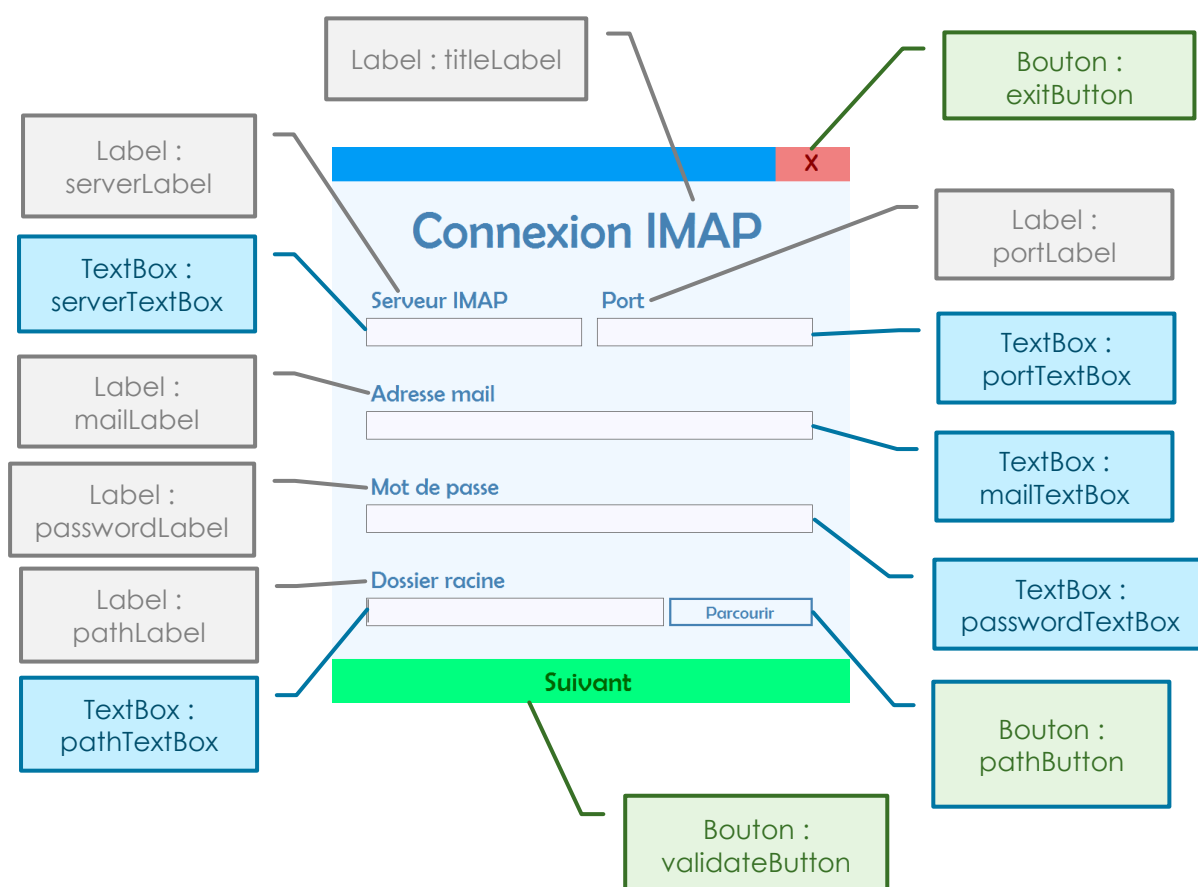
4.1 Dossier de Réalisation

4.1.1 Interface utilisateur

L'interface utilisateur possède deux fenêtres au total. La première représente le formulaire de connexion au serveur IMAP tandis que la seconde est la fenêtre de traitement automatique des fichiers.

Fenêtre de login

C'est la fenêtre qui possède le plus d'éléments à cause du formulaire de connexion.



Lors du lancement de l'application, la première fenêtre instanciée est la fenêtre de login. Cela permet à l'utilisateur d'entrer les données dont le programme a besoin pour fonctionner correctement.

Une fois les valeurs entrées, une tentative de connexion à la boîte mail est lancée. Dans le cas où la connexion réussit, les données fournies par l'utilisateur dans le formulaire de connexion sont stockées dans la base de données SQLite afin de connecter automatiquement l'utilisateur lors de la prochaine ouverture du programme.

Ci-dessous la liste complète de toutes les fonctions présentes dans la classe *LoginWindow*.

Fonction	LoginWindow() - Constructeur
Paramètres	-
Retour	-
Description détaillée	LoginWindow() est le constructeur de la classe du même nom. Dans cette fonction, les éléments de la fenêtre sont initialisés et affichés à l'écran. La nouvelle instance s'ajoute ensuite à la classe Globals dans sa variable dédiée. Cela permet d'accéder à cette instance de fenêtre de login depuis n'importe quel endroit du code.

Fonction	OnShown() - Événement
Paramètres	Object sender EventArgs e
Retour	-
Description détaillée	Cette fonction est appelée lorsque tous les objets de la fenêtre sont tous affichés à l'écran. Son but n'est que d'appeler la fonction FillFormWithExistingUser().

Fonction	ValidateConnection() - Événement
Paramètres	Object sender EventArgs e
Retour	-
Description détaillée	<p>Ce code est appelé lorsque le bouton « Suivant » situé en dessous du formulaire est cliqué.</p> <p>Premièrement le code commence par prendre chaque champ de formulaire et le valide. Si l'un des champs n'est pas valide, le booléen de vérification passe à « false ».</p> <p>Ensuite, si les champs sont valides, une connexion à la boîte mail via la classe « MailManager » est lancée.</p> <p>Si la connexion réussit, une instance de « DbManager » est créée et les données fournies par l'utilisateur.</p> <p>Enfin, la fenêtre de login est cachée et celle de traitement des fichiers est instanciée.</p>

Fonction	FillFormWithExistingUser()
Paramètres	-
Retour	-
Description détaillée	<p>Cette fonction est systématiquement appelée lorsque le programme est lancé. C'est depuis la fonction « OnShow() » que l'appel est fait.</p> <p>Dans ce code, une instance de connexion à la base de données est effectuée.</p> <p>Une vérification est ensuite faite dans la base de données afin de récupérer des données existantes s'il y en a.</p> <p>Dans le cas où il existe des données dans la base de données, elles sont récupérées et placées dans les champs de formulaire. La fonction « ValidateConnection() » est ensuite appelée.</p> <p>Cela permet la connexion automatique de l'utilisateur si ses données sont enregistrées dans la base de données.</p>

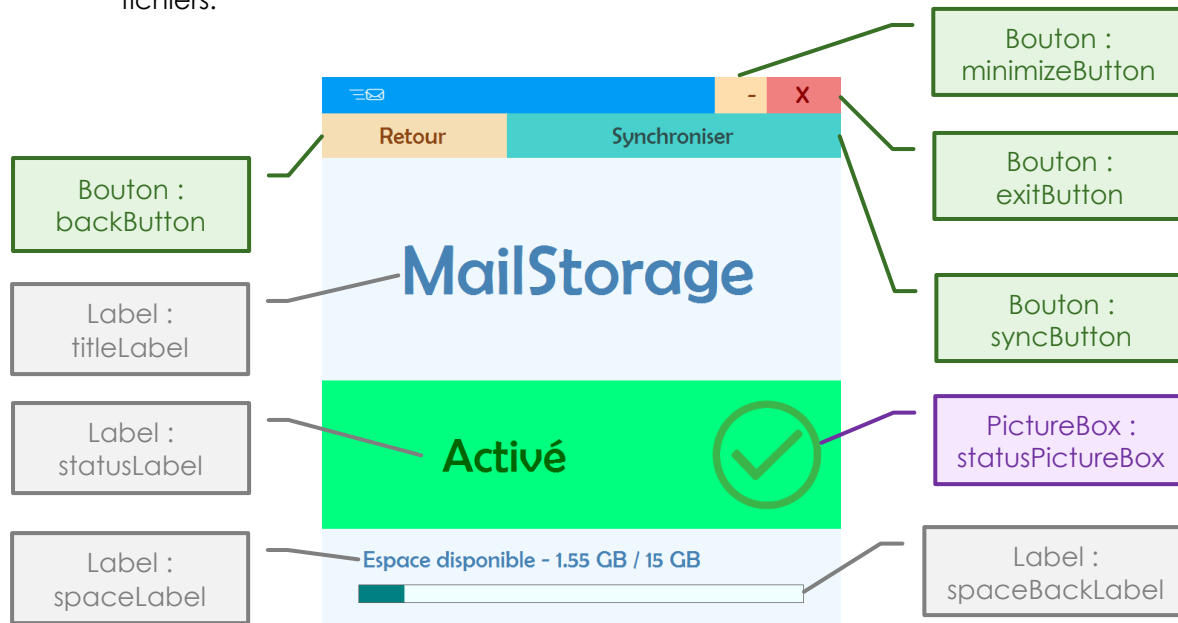
Fonction SelectDirectory() - Événement	
Paramètres	Object sender EventArgs e
Retour	-
Description détaillée	Appelée quand le bouton « Parcourir » est cliqué. Cette fonction affiche simplement une fenêtre d'explorateur afin que l'utilisateur choisisse son dossier racine.

Fonction MoveWindow() - Événement	
Paramètres	Object sender EventArgs e
Retour	-
Description détaillée	Fonction appelée lorsque l'utilisateur déplace la fenêtre.

Fonction ExitApplication() - Événement	
Paramètres	Object sender EventArgs e
Retour	-
Description détaillée	Fonction appelée lorsque l'utilisateur clique sur le bouton pour quitter l'application.

Fenêtre de traitement des fichiers

La fenêtre de traitement des fichiers possède moins d'éléments que celle de login par contre la plupart des exécutions de code se font en arrière-plan. Ci-dessous la description des éléments de la fenêtre de traitement de fichiers.



Cette fenêtre se charge de surveiller constamment le dossier racine choisi par l'utilisateur. Dès qu'un fichier est créé, déplacé, modifié, renommé ou supprimé, l'appel de la fonction de synchronisation est effectué. Les informations sur l'état de l'application sont affichées dans les labels de statuts ainsi que dans le label d'espace disponible qui, comme son nom l'indique, affiche l'espace disponible restant sur la boîte mail de l'utilisateur.

La fenêtre de gestion des fichiers correspond à la classe [MailStorage](#) décrite ci-dessous.

Fonction	MailStorage() - Constructeur
Paramètres	-
Retour	-
Description détaillée	Constructeur de la classe. Initialise tous les éléments de la fenêtre et démarre la surveillance du dossier racine.

Fonction	InitializeWindow()
Paramètres	-
Retour	-
Description détaillée	Vérifie si l'utilisateur a choisi un nouveau dossier et si tel est le cas, démarre la synchronisation initiale avec la boîte mail. Redémarre ensuite le monitoring du dossier si nécessaire.

Fonction	BackButtonClick() - Événement
Paramètres	Object sender EventArgs e
Retour	-
Description détaillée	Déconnecte l'application du serveur IMAP et retourne sur la fenêtre de login dans le cas où aucune synchronisation avec la boîte mail n'est en cours. Stoppe la surveillance du dossier racine.

Fonction	SyncButton() - Événement
Paramètres	Object sender EventArgs e
Retour	-
Description détaillée	Lance une synchronisation manuelle des fichiers en appelant la fonction « SynchroniseFiles() ».

Fonction	MoveWindow() - Événement
Paramètres	Object sender MouseEventArgs e
Retour	-
Description détaillée	En cas d'appui prolongé sur la barre de déplacement de la fenêtre, effectue un déplacement de la fenêtre par rapport à la position de la souris.

Fonction	ExitApplication() - Événement
Paramètres	Object sender MouseEventArgs e
Retour	-
Description détaillée	Appelé lors de l'appui sur le bouton de fermeture de l'application. Un message d'avertissement est affiché si l'utilisateur essaye de quitter l'application pendant une synchronisation.

Fonction	MinimizeWindow() - Événement
Paramètres	Object sender MouseEventArgs e
Retour	-
Description détaillée	Appelée lors de l'appui sur le bouton de réduction de la fenêtre. Au clic, l'application se réduit dans la barre des programmes en arrière-plan et un message d'information avertissant de l'exécution de l'application en arrière-plan avertit l'utilisateur.

Fonction	MaximizeWindow() - Événement
Paramètres	Object sender EventArgs e
Retour	-
Description détaillée	Affiche la fenêtre si elle est réduite. Appelée lors du clic sur l'icône de l'application dans la barre des programmes.

Fonction	UpdateMailboxSpace()
Paramètres	-
Retour	-
Description détaillée	Récupère les informations de la boîte mail de l'utilisateur et calcule l'espace restant pour l'afficher à l'écran.

Fonction	UpdateCurrentFile()
Paramètres	String strText
Retour	-
Description détaillée	Met à jour le label d'information sur le fichier en cours de traitement avec le texte passé en paramètres.

Fonction	SynchronizeFiles() - Événement
Paramètres	Object source FileSystemEventArgs e
Retour	-
Description détaillée	Cette fonction est appelée lorsque la surveillance du dossier racine rapporte une modification. La mise à jour des listes de fichiers est lancée puis, une fois les liste à jour, c'est la comparaison de ces listes qui est exécutée.

Fonction	RefreshWindowElements() - Événement
Paramètres	Object sender EventArgs e
Retour	-
Description détaillée	Timer qui vérifie que le dossier racine existe toujours et qui met constamment à jour le statut de l'

4.1.2 Connexion au serveur IMAP et interactions boîte mail

Pour que des communications avec la boîte mail de l'utilisateur puissent se faire, il faut commencer par établir une connexion continue avec le serveur IMAP puis authentifier l'utilisateur. Toutes les interactions avec le serveur IMAP et la boîte mail se font depuis la classe *MailManager*.

Connexion au serveur IMAP

```
// Connects to the IMAP4 server
imapCli.Connect(strServer, Int32.Parse(strPort), true);
imapCli.AuthenticationMechanisms.Remove("XOAUTH2");
```

Authentification utilisateur

```
// Authenticates the user
imapCli.Authenticate(strUser, strPassword);
```

Dans cette classe plusieurs fonctions sont disponibles pour interagir avec la boîte mail. Il est possible d'établir une connexion, de se déconnecter, d'envoyer ou encore de recevoir des mails.

MailManager est une classe statique, c'est-à-dire qu'elle ne nécessite pas d'être instanciée pour être utilisée. Au lancement de l'application, c'est depuis la fenêtre de login qu'elle est appelée. Une connexion en IMAP au serveur mail est initiée afin que les transferts de mails puissent se faire.

Par la suite, les fonctions d'envoi et de réception de mails sont utilisées lors de la synchronisation entre le dossier racine local choisi par l'utilisateur et la boîte mail.

Ci-dessous la liste des fonctions disponibles à l'utilisation dans la classe *MailManager*.

Fonction ConnectIMAP()	
Paramètres	String strServer String strPort String strUser String strPassword
Retour	Bool - Résultat de la connexion
Description détaillée	Cette fonction est appelée depuis la fenêtre de Login une fois que le formulaire de connexion a été validé. C'est dans ce code que la connexion au serveur IMAP est initiée. Des messages d'erreur et un booléen sont envoyés en retour en cas de problème de connexion au serveur IMAP et à la boîte mail.

Fonction DisconnectIMAP()	
Paramètres	-
Retour	-
Description détaillée	Simple fonction de déconnexion du serveur IMAP.

Fonction CreateStorageFolder()	
Paramètres	-
Retour	-
Description détaillée	Ce code est appelé directement après que la connexion IMAP soit établie. Il consiste à vérifier que le dossier de mails « MailStorage » existe dans la boîte mail de l'utilisateur. Dans le cas contraire, il est créé.

Fonction	SendMailToStorage()
Paramètres	String strSubject String strBody
Retour	-
Description détaillée	C'est la fonction d'ajout de mail à la boîte mail. Un nouveau mail est créé à partir des paramètres donnés puis il est placé dans le dossier « MailStorage » de la boîte mail.

Fonction	UpdateIndexMail()
Paramètres	String indexBody
Retour	-
Description détaillée	Cette fonction va mettre à jour le mail d'index des dossiers. Ce code est appelé à chaque synchronisation.

Fonction	DeleteMailInStorage()
Paramètres	AppFile fileToDelete
Retour	-
Description détaillée	Supprime un mail spécifique. La liste des mails présents dans la boîte mail est parcourue jusqu'à trouver le mail et le supprimer.

Fonction	GetAllMailSubjects()
Paramètres	-
Retour	List<String> - Liste de tous les sujets
Description détaillée	Récupère le sujet de tous les mails présents dans le dossier MailStorage de la boîte mail. L'avantage de cette fonction est qu'elle ne parcourt que les sujets des mails, ce qui évite de télécharger le mail complet pour en retirer des informations. Le processus est donc particulièrement rapide.

Fonction	GetAllMails()
Paramètres	-
Retour	List<MimeMessage> - Liste avec tous les mails
Description détaillée	<p>Cette fonction diffère légèrement de « GetAllMailSubjects() » car celle-ci va télécharger intégralement tous les mails présents dans le dossier MailStorage de la boîte mail.</p> <p>Une liste contenant tous les messages est retournée.</p>

Fonction	GetOneMail()
Paramètres	Bool – isIndex = true AppFile – fileToGet = null
Retour	MimeMessage – Le mail téléchargé
Description détaillée	<p>Cette fonction permet de spécifier un mail à télécharger dans la boîte mail.</p> <p>Les paramètres données peuvent spécifier si le mail voulu est simplement le mail d'index des dossiers ou si c'est un mail de fichier qui est recherché.</p> <p>Le mail téléchargé est ensuite retourné sous forme de « MimeMessage »</p>

Fonction	GetMailboxQuota()
Paramètres	-
Retour	FolderQuota – Informations sur la boîte mail
Description détaillée	Récupère les informations de la boîte mail et les renvoie sous forme de « FolderQuota »

4.1.3 Interactions avec la base de données

L'utilisation d'une base de données dans le projet permet l'enregistrement des données de l'utilisateur afin que la connexion au serveur IMAP se fasse automatiquement au lancement de l'application.

La classe *DbManager* se charge d'effectuer toutes les interactions avec la base de données. Il est possible de créer la table de données, de retirer ou d'ajouter des données. L'API utilisée est SQLite.

Les seules utilisations de la base de données se font lors de l'étape de connexion au serveur IMAP, lors du login utilisateur. Les informations de l'utilisateur sont stockées dès que la connexion IMAP est établie, ce qui permet une reconnexion automatique au prochain lancement de l'application.

Lors du premier lancement de l'application, la base de données est créée puis une connexion est ouverte pour y insérer et récupérer des données.

```
// Creates the file
SQLiteConnection.CreateFile("mailstoragedb.sqlite");

// Connects to the database
dbConnection = new SQLiteConnection("Data Source=mailstoragedb.sqlite;Version=3;");

// Opens the connection
dbConnection.Open();
```

La librairie SQLite permet des interactions SQL simples avec la base de données.

Ci-dessous, les fonctions disponibles dans la classe *DbManager*.

Fonction	DbManager - Constructeur
Paramètres	-
Retour	-
Description détaillée	Dans ce constructeur, une connexion à la base de données SQLite est lancée. Si la table AppData n'existe pas, elle est immédiatement créée afin que les données puissent y être stockées.

Fonction	UpdateUserData
Paramètres	String serverName String serverPort String userMail String userPassword String dirPath
Retour	-
Description détaillée	Cette fonction va insérer tous les paramètres entrants dans la table AppData de la base de données. Les données stockées correspondent au nom et port du serveur IMAP, au mail et mot de passe de l'utilisateur et au dossier racine choisi.

Fonction	GetCurrentUserData()
Paramètres	-
Retour	List<String> - Données de l'utilisateur
Description détaillée	Permet de récupérer les données de l'utilisateur qui sont stockées dans la table AppData. Cette fonction retourne une liste avec les valeurs nécessaires pour remplir entièrement et automatiquement le formulaire de connexion.

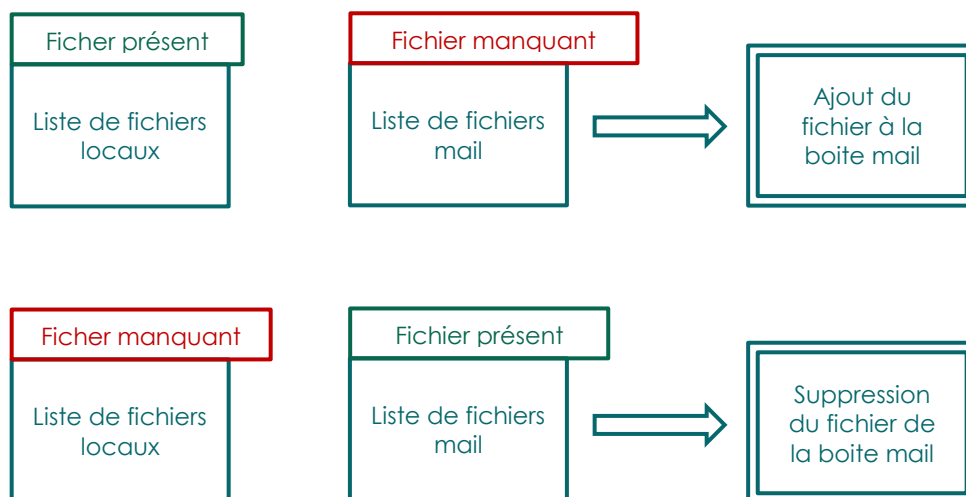
Fonction	ExecuteSQLQuery()
Paramètres	-
Retour	String – Requête SQL à exécuter
Description détaillée	Exécute une requête SQL dans la base de données. La requête peut être un INSERT ou un DELETE mais cette fonction ne retourne aucune valeur. Il n'est donc pas possible d'effectuer un SELECT.

4.1.4 La synchronisation automatique des fichiers

La synchronisation des fichiers « dossier local – boîte mail » est le cœur du programme. Afin que les fichiers présents dans le dossier racine choisi par l'utilisateur correspondent en continu aux fichiers présents dans la boîte mail, deux listes de fichiers sont constamment tenues à jour afin de répertorier tous les fichiers présents aux deux endroits.



Lorsque les deux listes sont à jour, elles sont comparées. Les fichiers présents dans la liste locale qui sont manquants dans la liste distante sont envoyés dans la boîte mail tandis que les fichiers manquants dans la liste locale mais présents dans la liste distante sont supprimés de la boîte mail.



L'application a été pensée pour être utilisée par une seule personne à la fois. Cependant, une fonctionnalité supplémentaire permettant de gérer une utilisation simultanée du programme est envisageable.

Toutes ces interactions de fichiers sont gérées par quatre fonctions principales au sein de la classe *FilesManager*.

<pre> /// <summary> /// Updates the local files list, adds the new files and removes the deleted ones /// </summary> public static void UpdateLocalFiles()...</pre>	Mise à jour de la liste locale
<pre> /// <summary> /// Updates the remote files list, adds the mail files in the list and removes the old ones /// </summary> public static void UpdateRemoteFiles()...</pre>	Mise à jour de la liste distante
<pre> /// <summary> /// Adds the new local files to the user mailbox /// </summary> public static void AddLocalFilesToMailBox()...</pre>	Ajout des fichiers à la boîte mail
<pre> /// <summary> /// Deletes the files in the mailbox that are not present in the local directory /// </summary> public static void DeleteRemotesFilesFromLocal()...</pre>	Suppressions dans la boîte mail

La classe *FilesManager* est une des classes les plus importantes du programme. C'est dans cette classe que sont gérées toutes les interactions avec les fichiers, que ce soit au niveau de l'indexation ou au niveau de la transformation des fichiers en texte.

Ci-dessous la liste de toutes les fonctions de la classe *FilesManager* et leur description.

Fonction	InitialSynchronisation()
Paramètres	-
Retour	-
Description détaillée	<p>La fonction de synchronisation initiale permet le téléchargement des fichiers de la boîte mail lorsque le dossier racine est spécifié par l'utilisateur pour la première fois.</p> <p>Cette fonction va récupérer chaque fichier de la boîte mail et le placer dans le dossier local sous forme de fichier.</p> <p>Le fichier d'indexation des dossiers va aussi être téléchargé afin de recréer l'arborescence des dossiers dans le nouveau dossier local.</p>

Fonction	StartUpdateFromMailBox()
Paramètres	-
Retour	-
Description détaillée	<p>Cette fonction représente le processus de synchronisation au démarrage.</p> <p>À chaque lancement de l'application, cette fonction est appelée et va vérifier l'état de la boîte mail.</p> <p>Si de nouveaux fichiers ou des plus récents s'y trouvent, ils sont téléchargés et placés dans le dossier local. Les fichiers présents dans le dossier local mais absents de la boîte mail sont supprimés du dossier.</p> <p>En comptant la fonction de synchronisation initiale, c'est l'un des seuls moments où la priorité est donnée à la boîte mail pour la création et suppression de fichiers.</p>

Fonction	UpdateLocalFiles()
Paramètres	-
Retour	-
Description détaillée	<p>Cette fonction représente l'étape de mise à jour de la liste des fichiers locaux.</p> <p>Pour commencer, les fichiers présents dans le dossier local sont tous listés.</p> <p>Ensuite, ceux qui ont été supprimés et qui sont encore présents dans la liste sont retirés.</p> <p>Et pour finir, nouveaux fichiers listés sont ajoutés à la liste de fichiers locaux.</p>

Fonction	UpdateRemoteFiles()
Paramètres	-
Retour	-
Description détaillée	<p>C'est dans cette fonction que les fichiers de la boîte mail sont listés dans la liste distante.</p> <p>Tout d'abord, tous les sujets des mails sont récupérés, puis chaque information est séparée.</p> <p>Ensuite, une instance de fichier « AppFile » est créée avec chaque donnée récupérée.</p> <p>Et pour finir, ces nouvelles instances sont placées dans la liste distante.</p>

Fonction	AddLocalFilesToMailBox()
Paramètres	-
Retour	-
Description détaillée	<p>Fonction faisant partie de l'étape de comparaison des listes. Cette fonction va vérifier quels fichiers sont présents dans la liste locale mais absents de la liste distante.</p> <p>Ces fichiers vont être transformés en texte Base64 puis ajoutés à la boîte mail sous forme de mail.</p>

Fonction	DeleteRemoteFilesFromLocal()
Paramètres	-
Retour	-
Description détaillée	<p>Deuxième fonction faisant partie de l'étape de comparaison des listes, c'est dans ce code que les fichiers qui ne sont plus présents dans le dossier local sont supprimés de la boîte mail.</p> <p>Les deux listes sont comparées et tous les fichiers présents uniquement dans la liste distante sont supprimés de la boîte mail afin d'égaliser les deux listes.</p>

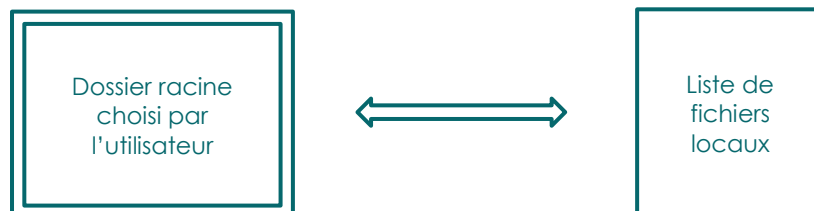
Fonction	UpdateDirectoriesIndex()
Paramètres	-
Retour	-
Description détaillée	Etablit l'arborescence complète des dossiers dans le dossier racine sous forme de texte et met à jour le mail d'indexation des dossiers.

Fonction	ConvertFileTo64()
Paramètres	AppFile fileToConvert
Retour	String – Le fichier en texte Base64
Description détaillée	Convertit le fichier donné en paramètre en chaîne de caractères Base64. Le texte est retourné à la fin de l'exécution de la fonction.

Fonction	AddLocalFileToList()
Paramètres	String strFilePath
Retour	-
Description détaillée	Cette fonction récupère les informations d'un fichier dont le chemin d'accès est donné en paramètres pour ensuite créer une nouvelle instance de « AppFile » qui sera par la suite ajoutée à la liste des fichiers du dossier local.

Mise à jour de la liste locale

La liste locale contient tous les fichiers qui sont présents dans le dossier racine choisi par l'utilisateur. À chaque synchronisation, la liste va se mettre à jour en éliminant les fichiers qui ne sont plus présents dans le dossier local et en y ajoutant les nouveaux.



La liste des fichiers du dossier local est récupérée grâce à la classe *Directory*, qui permet de chercher des fichiers récursivement à partir d'un emplacement donné. Les informations des fichiers sont ensuite retournées sous forme de *FileInfo* dans un tableau.

Le tableau de fichiers est ensuite parcouru. Dans le cas où un fichier est présent dans la liste locale mais pas dans le tableau de fichiers (donc pas dans le dossier local), ce fichier est supprimé de la liste locale.

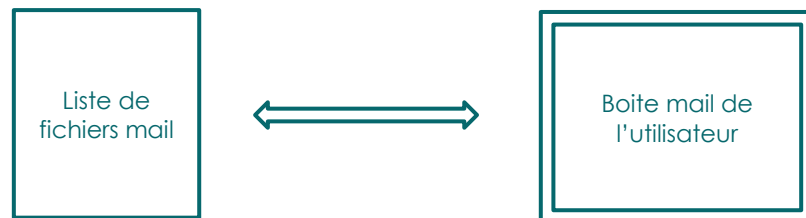
Si au contraire un fichier est présent dans le dossier local mais pas dans la liste locale, il est ajouté à cette dernière.

Le fichier	Est dans le dossier	N'est pas dans le dossier
Est dans la liste	Ne rien faire	Supprimer de la liste
N'est pas dans la liste	Ajouter à la liste	Ne rien faire

Une fois la comparaison entre la liste locale et le dossier racine effectuée, la liste locale se retrouve mise à jour avec une liste complète et correcte des fichiers présents dans le dossier racine.

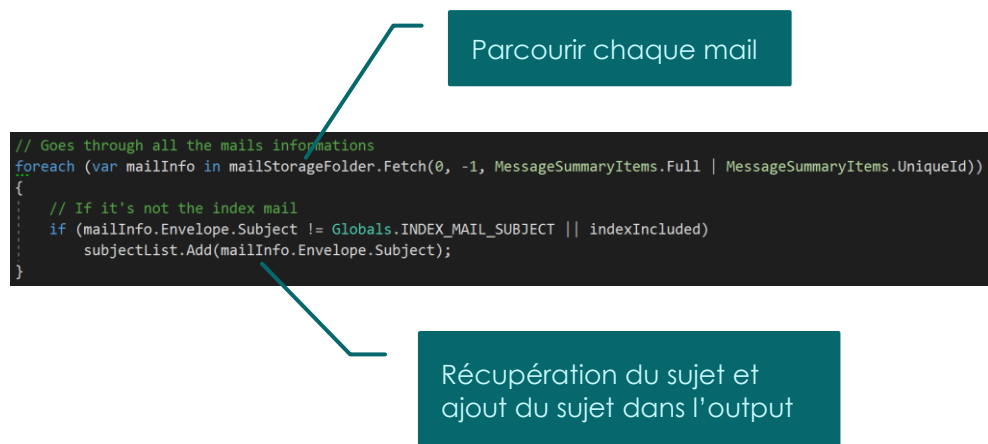
Mise à jour de la liste distante

La mise à jour de la liste distante consiste à vérifier les fichiers présents sur la boîte mail de l'utilisateur et changer les éléments de la liste afin qu'ils correspondent exactement à l'état de la boîte mail.



Le processus de mise à jour de la liste distante ressemble à celui de la liste locale. Cependant, pour pouvoir mettre la liste distante à jour, il faut pouvoir vérifier quels fichiers sont sur la boîte mail et cela implique des interactions avec le serveur mail et donc la librairie *MailKit* et la classe *MailManager*.

Pour récupérer l'état des fichiers dans la boîte mail, le programme utilise la fonction *GetAllMailSubjects()* qui va récupérer tous les sujets des mails dans la boîte mail et le renvoyer en output.



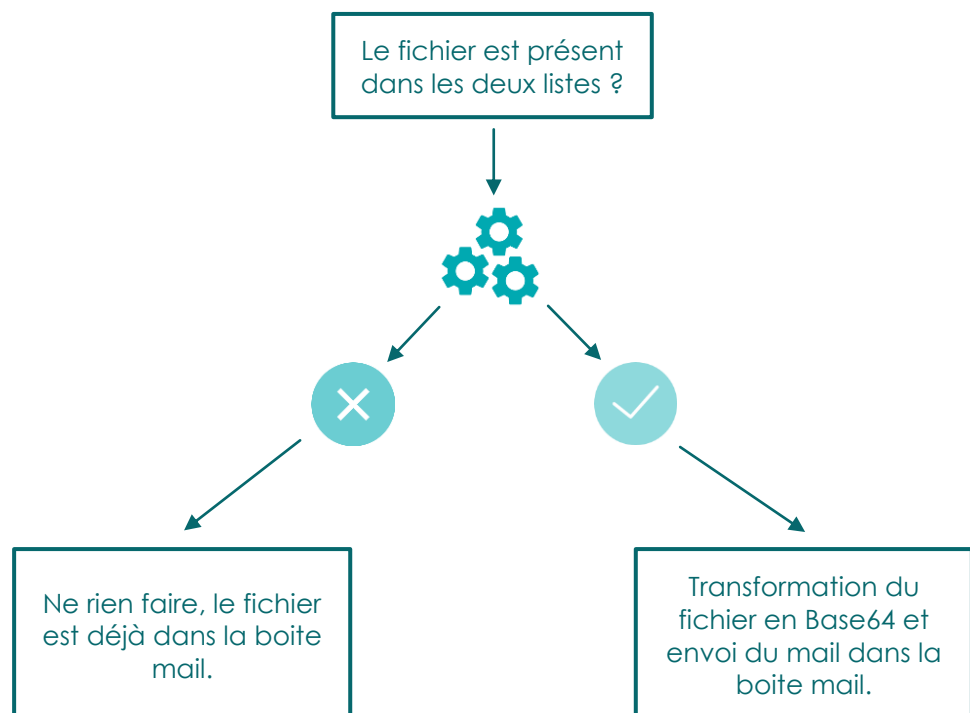
```
// Goes through all the mails informations
foreach (var mailInfo in mailStorageFolder.Fetch(0, -1, MessageSummaryItems.Full | MessageSummaryItems.UniqueId))
{
    // If it's not the index mail
    if (mailInfo.Envelope.Subject != Globals.INDEX_MAIL_SUBJECT || indexIncluded)
        subjectList.Add(mailInfo.Envelope.Subject);
}
```

Une fois les sujets récupérés, les noms des fichiers sont isolés et la comparaison de l'état de la boîte mail et de la liste distante peut se faire. Comme pour la liste locale, les fichiers présents dans la boîte mail mais pas dans la liste distante sont ajoutés à la liste tandis que les fichiers présents dans la liste distante mais pas dans la boîte mail sont supprimés de la liste.

Envoi des nouveaux fichiers sur la boîte mail

L'envoi de fichiers sur la boîte mail de l'utilisateur se fait lors de l'étape de comparaison de la liste locale avec la liste distante.

Lorsque les deux listes sont à jour, la fonction `AddLocalFilesToMailBox()` va parcourir chaque élément de la liste locale et vérifier que cet élément possède un équivalent dans la liste distante. Si ce n'est pas le cas, le fichier en question est transformé en texte Base64 puis envoyé sous forme de mail dans la boîte mail de l'utilisateur.



Les fichiers manquants dans la boîte mail y sont ainsi ajoutés et à la prochaine synchronisation, les deux listes seront équivalentes.

```
// If the file is not in the mailbox, adds id
if (!blnFileExists)
{
    // Sets the file status label
    Globals.mainWindow.UpdateCurrentFile("Envoi du fichier\n" + localFile.fileName);

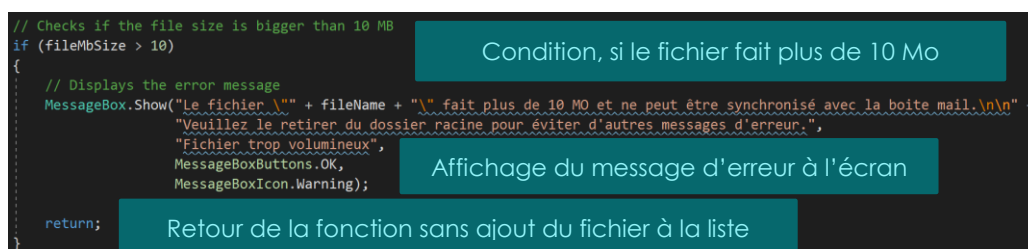
    // Sends the file to the storage
    MailManager.SendMailToStorage(localFile.fileName +
        ";;" +
        localFile.filePath +
        ";;" +
        localFile.fileCreationDate +
        ";;" +
        localFile.fileModificationDate,
        ConvertFileTo64(localFile));
}
```

Condition, si le fichier n'est pas dans la boîte mail

Envoi du mail de fichier

Il existe cependant une contrainte pour l'envoi de ces fichiers : la taille. En effet, la taille du fichier à envoyer ne doit pas dépasser 10 Mo. Cette restriction est due à des raisons de temps d'envoi et de limitation de taille de mail dans la boîte mail.

Les fichiers de plus de 10 Mo sont simplement détectés lors de la mise à jour de la liste locale et n'y sont pas ajoutés.

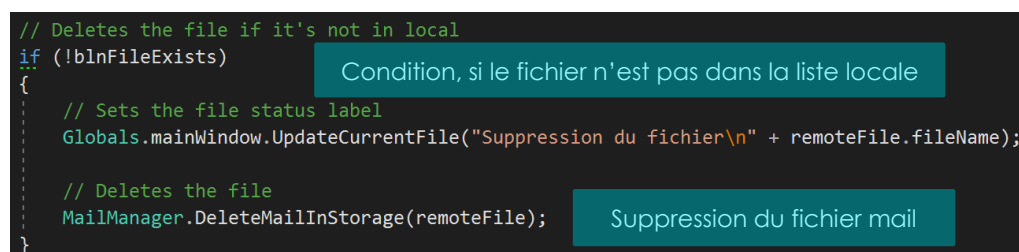


Une fonctionnalité supplémentaire a été envisagée lors de l'analyse pour permettre l'envoi de fichiers sans limitation de taille. Cependant, le manque de temps pour le projet n'a pas permis la réalisation de cette fonctionnalité. Pour plus d'informations, voir la rubrique « Fonctionnalités supplémentaires à implémenter » dans la conclusion du rapport.

Suppression des fichiers de la boîte mail

Deuxième étape de la comparaison des listes, la suppression de mails de fichiers de la boîte mail se fait lorsqu'un fichier a été supprimé, renommé ou édité.

Chaque élément de la liste distante est parcouru et pour chacun de ces éléments, le programme vérifie qu'il existe un équivalent dans la liste locale. Si ce n'est pas le cas, le fichier de la liste distante est identifié dans la boîte mail puis supprimé.



La suppression du mail de fichier se fait depuis la classe *MailManager* car cela implique une interaction avec la boîte mail.

Lorsqu'un fichier est modifié dans le dossier local, la date de modification est prise en compte dans la comparaison des listes ce qui fait que le fichier est plus récent et doit être mis à jour dans la boîte mail. C'est à ce moment que la suppression de fichier intervient. L'ancienne version du fichier est supprimée de la boîte mail afin que la nouvelle version puisse se créer.

Dans le cas d'un renommage de fichier, le principe reste le même. Le fichier ne possède plus le même nom et le programme l'identifie comme un nouveau fichier. Le fichier avec l'ancien nom est donc supprimé de la boîte mail et la nouvelle version avec le nouveau nom y est créée.

La suppression d'un fichier spécifié se fait dans la classe *MailManager*. Tout d'abord le programme parcourt tous les sujets des mails de la boîte mail et compare les noms de fichiers avec le fichier recherché.

Une fois que le fichier est trouvé, l'uuid du mail correspondant est récupéré et le tag « supprimé » est appliqué au mail. Les changements de la boîte mail sont ensuite enregistrés. Le mail du fichier spécifié est correctement supprimé de la boîte mail de l'utilisateur.

```
/// <summary> Deletes a specified file in the mailbox
public static void DeleteMailInStorage(AppFile fileToDelete)
{
    // Opens the MailStorage folder
    var mailStorageFolder = imapCli.GetFolder("MailStorage");
    mailStorageFolder.Open(FolderAccess.ReadWrite);

    // Goes through all the mails informations
    foreach (var mailInfo in mailStorageFolder.Fetch(0, -1, MessageSummaryItems.Full | MessageSummaryItems.UniqueId))
    {
        // Checks if the paths matches
        if (mailInfo.Envelope.Subject.Split(new [] { ":" }, StringSplitOptions.None)[1] == fileToDelete.filePath)
        {
            // Deletes the file
            mailStorageFolder.AddFlags(new [] { mailInfo.UniqueId }, MessageFlags.Deleted, true);

            // Applies the delete and closes the folder
            mailStorageFolder.Close(true);

            // Returns
            return;
        }
    }
}
```

Ouverture du dossier « MailStorage »

Lecture de tous les sujets de mail

Application du tag « Supprimé » au mail spécifié

Fermeture du dossier « MailStorage » et application des changements

4.1.5 Synchronisation initiale

La synchronisation initiale intervient lorsque l'utilisateur lance le programme pour la première fois, ou plus précisément, lorsqu'il définit un nouveau dossier racine au programme.

Le but de cette synchronisation est de s'informer sur l'état de l'arborescence et des fichiers dans la boîte mail pour ensuite tous les télécharger dans le nouveau dossier local.

Cette synchronisation permet de remplir le dossier racine avec les fichiers qui ne s'y trouvent pas et qui pourtant ont été ajoutés par le passé à la boîte mail.

Un exemple est l'utilisation du programme sur deux ordinateurs différents. Le premier se charge de placer de nouveaux fichiers dans la boîte mail. Lorsque l'utilisateur voudra utiliser le programme sur le deuxième ordinateur, il va spécifier un autre dossier racine. La synchronisation initiale va vérifier les fichiers présents dans la boîte mail et va tous les télécharger dans le nouveau dossier local.

La fonction s'exécute en passant par plusieurs étapes.

Premièrement, l'application vérifie si des fichiers sont déjà présents dans le dossier racine et demande à l'utilisateur s'il veut les supprimer ou les synchroniser.

```
// Checks if there're files in the root folder
if (Directory.GetFiles(Globals.ROOT_DIRECTORY, "*", SearchOption.AllDirectories).Length > 0 || Directory.GetDirectories(Globals.ROOT_DIRECTORY, "").Length > 0)
{
    // Displays the question message about the files
    var userChoice = MessageBox.Show("Vous avez changé d'adresse mail ou de dossier racine. Voulez-vous que ces fichiers du dossier soient synchronisés avec la boîte mail ou supprimés ?\n\n" +
        "(Cliquez sur OUI pour synchroniser, sur NON pour les supprimer et sur ANNULER pour quitter l'application)",
        "Fichiers dans le dossier racine",
        MessageBoxButtons.YesNoCancel, MessageBoxIcon.Warning);
}
```

Vérifie s'il y a des fichiers dans le dossier local

Message d'avertissement

Ensuite, le mail d'arborescence des dossiers est récupéré et les dossiers sont créés dans le dossier racine.

```
// Gets the directories list
var allDirectories = MailManager.GetOneMail().TextBody.Split(new[] { "::-" }, StringSplitOptions.None).ToList();

// Creates all the directories in the root folder
foreach (var folder in allDirectories)
{
    // Sets the file status label
    Globals.mainWindow.UpdateCurrentFile("Création du dossier\n" + folder);

    // Creates the directory
    Directory.CreateDirectory(Globals.ROOT_DIRECTORY + folder.Substring(1));
}
```

Récupération de l'arborescence

Création des dossiers

Et pour finir, tous les fichiers mail sont téléchargés et placés dans le dossier local.

```
// Gets all the mails from the mailbox
var allMails = MailManager.GetAllMails()

// Adds each file per mail
foreach (var mail in allMails)
{
    // If no root folder, abort
    if (!Directory.Exists(Globals.ROOT_DIRECTORY))
        return;

    // Splits the mail subject to get info
    var mailInfos = mail.Subject.Split(new[] { "::" }, StringSplitOptions.None);

    // Gets the mail full path
    var fullPath = Globals.ROOT_DIRECTORY + mailInfos[1].Substring(1);

    // Creates the file from base64
    new FileInfo(fullPath).Directory.Create();
    File.WriteAllBytes(fullPath, Convert.FromBase64String(mail.TextBody));

    // Sets the file creation and edit time
    File.SetCreationTime(fullPath, DateTime.Parse(mailInfos[2]));
    File.SetLastWriteTime(fullPath, DateTime.Parse(mailInfos[3]));
}
```

Récupère tous les mails

Parcours tous les mails

Transforme le texte Base64 en fichier

Crée le nouveau fichier

Les mails récupérés sont sous forme de texte Base64. La fonction `Convert.FromBase64String()` est utilisée pour récupérer le binaire qui est utilisé pour créer les fichiers dans le dossier local.

Les dates de création et de modification sont ensuite appliquées au nouveau fichier créé afin qu'il soit identique au fichier présent dans la boîte mail.

Les informations du fichier sont stockées dans le sujet du mail et séparées par les caractères « :: ». Une fois le sujet du mail séparé en plusieurs éléments et placé dans un tableau, cela permet d'accéder aux informations du fichier pour le recréer, lui donner le nom correct et les dates de création et de modification égales à celles de la boîte mail.

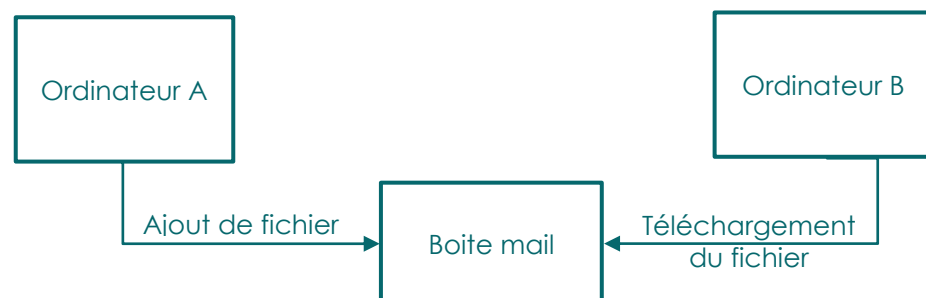
4.1.6 Synchronisation au démarrage

Cette synchronisation est automatiquement lancée à chaque lancement du programme et consiste à donner la priorité à la boîte mail.

Une synchronisation normale supprime les fichiers dans la boîte mail et les ajoute en fonction de ce que le dossier racine contient. Dans la synchronisation au démarrage, c'est en fonction de ce que la boîte mail contient que les fichiers sont ajoutés ou supprimés au dossier local. Les rôles sont inversés.

Une telle synchronisation permet de mettre à jour l'état du dossier local dans le cas où le programme est utilisé à plusieurs endroits en même temps.

Par exemple si un fichier est rajouté à la boîte mail sur l'ordinateur A et que l'utilisateur lance le programme sur l'ordinateur B, la synchronisation au démarrage va se lancer et le nouveau fichier va être téléchargé dans le dossier racine de l'ordinateur B. Sans cette synchronisation, les fichiers ne seraient jamais téléchargés ce qui rendrait l'utilisation du programme impossible.



La synchronisation au démarrage est similaire à la synchronisation initiale mais contrairement à cette dernière, elle va aussi supprimer les fichiers dans le dossier local qui ne sont pas dans la boîte mail.

Cette synchronisation initiale se fait au tout début de l'exécution du programme et ne dure généralement pas très longtemps. Aucun envoi de fichier n'y est fait. La fonction de synchronisation au démarrage est appelée *StartUpdateFromMailBox()*.

4.1.7 Mail d'index des dossiers

Dans le dossier « MailStorage » de la boîte mail, il existe un mail différent et qui n'a rien à voir avec les autres mails de fichiers : c'est le mail d'arborescence des dossiers.

Ce mail contient simplement la liste complète de tous les dossiers présents dans le dossier racine. L'utilité d'un tel mail est de pouvoir recréer l'arborescence lorsqu'un nouveau dossier racine est choisi.

Ce mail d'arborescence est mis à jour à chaque synchronisation et est appelé à chaque synchronisation initiale.

Sa structure est très simple, il possède le chemin relatif de chaque dossier présent dans le dossier racine. Chaque chemin est séparé par les caractères « :: ».

La fonction qui se charge de mettre à jour le mail d'index se trouve dans la classe *FilesManager* et se nomme *UpdateDirectoriesIndex()*. Dans ce code, l'ancien mail est supprimé et remplacé par le nouveau.

```
/// <summary> Gets all the directories in the folder and crats the updates version of the index mail
public static void UpdateDirectoriesIndex()
{
    // Sets the file status label
    Globals.mainWindow.UpdateCurrentFile("Indexation des dossiers");

    // Gets all the folders in the root
    var allDirectories = Directory.GetDirectories(Globals.ROOT_DIRECTORY, "*", SearchOption.AllDirectories);

    // Defines the index mail content
    string strMailContent = "";

    // Adds each folder in the content
    foreach (var folder in allDirectories)
    {
        strMailContent += folder.Replace(Globals.ROOT_DIRECTORY, @".") + "::";
    }

    // Deletes the last "::"
    if (strMailContent.Length > 0)
        strMailContent = strMailContent.Substring(0, strMailContent.Length - 2);

    // Sends the mail to the mailbox
    MailManager.UpdateIndexMail(strMailContent);
}
```

Récupère tous les dossiers à la racine

Ajoute le chemin de chaque dossier au nouveau mail d'index

Remplace l'ancien mail par le nouveau

4.1.8 Cryptage du mot de passe de l'utilisateur

La première question qui vient à l'esprit est « pourquoi avoir choisi le cryptage au lieu du hachage pour un mot de passe ».

La réponse à cette question est plutôt simple. Le mot de passe de l'utilisateur nécessite d'être protégé, il ne peut donc pas être stocké en clair dans la base de données. Cependant, la fonctionnalité de connexion automatique qui permet au programme de se connecter automatiquement au compte mail de l'utilisateur nécessite que le programme entre lui-même le mot de passe dans le champ de connexion au serveur IMAP.

Le problème du hachage dans cette démarche est qu'il est impossible de récupérer le mot de passe automatiquement lorsqu'il a été haché. Pour pouvoir le récupérer, il faut que l'utilisateur l'écrive dans le formulaire de connexion, ce qui casse la fonctionnalité de connexion automatique.

La deuxième solution est alors le cryptage. La solution choisie est un cryptage AES du mot de passe de l'utilisateur avec une clé de cryptage stockée en local dans le programme. La librairie utilisée se nomme *csharp-aes*.

Cette API permet le cryptage d'une chaîne de caractères très facilement.

```
// Crypts the user password
var aesPwdKey = new PasswordAes(Globals.PASSWORD_KEY);
var cryptedPwd = aesPwdKey.Encrypt(passwordTextBox.Text);
```

Le cryptage du mot de passe se fait en récupérant la clé de cryptage, puis en encryptant le mot de passe. L'output consiste en une autre chaîne de caractères qui représentent le mot de passe crypté. Cette chaîne est placée dans la base de données.

```
// Decrypts the password
var aesPwdKey = new PasswordAes(Globals.PASSWORD_KEY);
var decryptedPwd = aesPwdKey.Decrypt(liData[3]);
```

Le décryptage est très similaire au cryptage, la clé de cryptage est récupérée puis la chaîne de caractères extraite de la base de données est décryptée afin que le mot de passe puisse être récupéré en sortie.

4.2 Dossier des tests

4.2.1 Scénario N°1 : Connexion

Étape	Description
Entrée	Fenêtre de connexion à l'application
Quand	Saisie du compte mail, du mot de passe et du dossier racine
Résultat	Connexion au compte mail et affichage de la fenêtre de traitement des fichiers

Résultat OK

Remarque

La connexion fonctionne avec différents serveurs IMAP, par contre les données de l'espace disponible ne sont pas accessibles pour tous les serveurs.

Cependant, certains serveurs mail sont moins stables que d'autres. L'application semble fonctionner mieux avec le serveur IMAP de Gmail tandis que d'autres serveurs comme Outlook peuvent parfois terminer la connexion sans aucune raison apparente.

4.2.2 Scénario N°2 : Synchronisation

Étape	Description
Entrée	Bouton de synchronisation manuelle des fichiers
Quand	Clic sur le bouton de synchronisation
Résultat	Mise à jour et synchronisation des fichiers « dossier - boîte mail »

Résultat OK

Remarque

Lance la synchronisation manuelle. Les fichiers du dossier local et de la boîte mail sont synchronisés.

4.2.3 Scénario N°3 : Changement de compte

Étape	Description
Entrée	Fenêtre de connexion à l'application
Quand	Changement du compte mail, mot de passe et dossier racine
Résultat	Connexion au compte mail et affichage de la fenêtre de traitement des fichiers

Résultat OK

Remarque

La connexion au serveur mail est interrompue, la fenêtre de login est affichée et une fois la reconnexion effectuée, si l'utilisateur a décidé de changer de dossier racine, une synchronisation initiale est lancée.

4.2.4 Scénario N°4 : Espace disponible

Étape	Description
Entrée	Fenêtre de traitement des fichiers
Quand	À tout moment
Résultat	L'espace disponible affiché correspond à l'espace disponible dans la boîte mail

Résultat Fonctionnel mais limité

Remarque

L'espace disponible s'affiche lorsque l'application est connectée au serveur IMAP.

Cependant, certains serveurs IMAP (Outlook par exemple) n'offrent pas la possibilité de récupérer l'espace disponible ce qui rend impossible l'affichage de cette donnée dans le programme.

4.2.5 Scénario N°5 : Synchronisation initiale

Étape	Description
Entrée	Dossier local
Quand	Après connexion à la boîte mail
Résultat	Création automatique des fichiers depuis la boîte mail

Résultat OK

Remarque

Lors de la première connexion ou lorsque le dossier racine est changé, la synchronisation initiale est lancée et va récupérer tous les fichiers et les dossiers stockés dans la boîte mail pour les ajouter en local.

4.2.6 Scénario N°6 : Création de fichiers

Étape	Description
Entrée	Dossier local
Quand	Ajout d'un fichier dans le dossier
Résultat	Ajout automatique du fichier dans la boîte mail

Résultat OK

Remarque

Lorsqu'un fichier est créé dans le dossier local, le programme détecte automatiquement le changement et lance une synchronisation automatique.

Le fichier est transformé en texte Base64 et est ensuite visible dans la boîte mail dans le dossier MailStorage.

4.2.7 Scénario N°7 : Suppression de mail

Étape	Description
Entrée	Boite mail
Quand	Suppression d'un mail de fichier
Résultat	Recréation automatique du mail dans la boite mail

Résultat OK

Remarque

Dans le cas où un fichier est supprimé de la boite mail manuellement, il sera automatiquement recréé lors de la prochaine synchronisation.

4.2.8 Scénario N°8 : Suppression de fichier

Étape	Description
Entrée	Dossier local
Quand	Suppression d'un fichier dans le dossier local
Résultat	Suppression automatique dans la boite mail

Résultat OK

Remarque

De la même façon que pour la création d'un fichier, lorsqu'un fichier est supprimé, le changement est détecté automatiquement et une synchronisation est lancée.

Quelques instants après, le fichier apparaît dans le dossier MailStorage de la boite mail.

4.2.9 Scénario N°9 : Édition de fichier

Étape	Description
Entrée	Dossier local
Quand	Édition d'un fichier dans le dossier local
Résultat	Mise à jour automatique du fichier dans la boîte mail

Résultat OK

Remarque

Lorsqu'un fichier est modifié, le changement est détecté automatiquement par le programme et une synchronisation est lancée. L'ancien fichier est d'abord supprimé de la boîte mail, puis la nouvelle version est ajoutée.

4.2.10 Scénario N°10 : Gestion des erreurs

Étape	Description
Entrée	Fenêtre de connexion à l'application
Quand	Insertion d'une fausse adresse mail et mauvais mot de passe
Résultat	Affichage des messages d'erreur

Résultat**OK***Remarque**Des messages d'avertissement s'affichent lorsque :*

- Le serveur IMAP interdit la connexion
- Lorsque l'authentification échoue
- Le dossier MailStorage n'est pas accessible
- Si des fichiers sont présents localement lors d'une synchronisation initiale
- Les fichiers sont trop volumineux pour une synchronisation (10 Mo)
- Si le dossier racine est supprimé
- Si la connexion au serveur IMAP est perdue
- Si l'utilisateur quitte l'application en cours de synchronisation
- Si l'utilisateur retourne au login en cours de synchronisation
- Si un fichier est impossible à supprimer
- Message de synchronisation avant la fermeture de l'application

4.2.11 Autres tests

Description	Résultat
Vérification des champs de formulaire de connexion	OK
Explorateur pour sélectionner un dossier racine	OK
Le statut change lors d'une synchronisation	OK
La barre de chargement correspond à la valeur d'espace disponible	OK
Les dossiers sont recréés lors d'une synchronisation initiale	OK
Les fenêtres peuvent être déplacées et fermées	OK
La fenêtre principale peut être réduite et tourner en arrière-plan	OK
Le mot de passe de l'utilisateur est crypté	OK
Les fichiers sont lus en lecture seule pour éviter des conflits	OK
La synchronisation au démarrage télécharge les fichiers plus récents	OK
Les threads actifs n'entrent pas en conflit	OK
Les fichiers plus anciens sont supprimés au profit des plus récents	OK
Il n'y a pas de doublons de fichiers dans la boîte mail	OK
L'état de traitement des fichiers en cours est affiché à l'écran	OK

Conclusion

5 CONCLUSION

5.1 Bilan des fonctionnalités demandées

Ci-dessous se trouvent toutes les fonctionnalités demandées dans le cahier des charges. L'état de la fonctionnalité indique si celle-ci a été implémentée ou abandonnée.

<i>Description de la fonctionnalité selon le cahier des charges</i>	<i>Etat</i>
<i>Configuration du serveur IMAP et du port</i>	OK
<i>Configuration du login et mot de passe utilisateur</i>	OK
<i>Affichage de l'espace disponible</i>	OK
<i>Créer un fichier</i>	OK
<i>Renommer un fichier</i>	OK
<i>Supprimer un fichier</i>	OK
<i>Ouvrir et éditer un fichier</i>	OK
<i>Conservation des métadonnées des fichiers</i>	OK
<i>Utilisation de l'explorateur du système d'exploitation</i>	OK
<i>Synchronisation des fichiers entre la boîte mail et le dossier local</i>	OK
<i>Utilisation de l'espace de stockage de la boîte mail</i>	OK
<i>Utilisation du protocole IMAP pour interagir avec la boîte mail</i>	OK
<i>Fonctionnement sans gestion des droits utilisateur</i>	OK

L'intégralité des fonctionnalités demandées dans le cahier des charges ont été implémentées et testées.

D'autres fonctionnalités supplémentaires étaient prévues, cependant, le manque de temps disponible n'a pas permis leur implémentation. Pour plus d'informations sur ces fonctionnalités supplémentaires, voir la rubrique suivante.

5.2 Fonctionnalités supplémentaires à implémenter

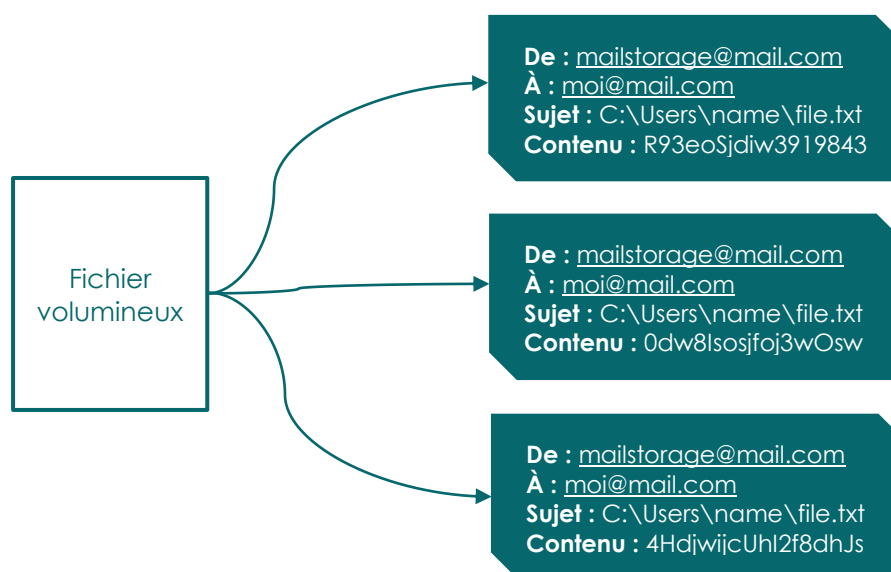
5.2.1 Gestion des fichiers de grande taille

L'une des contraintes de l'utilisation de l'espace de stockage de la boîte mail de l'utilisateur est la limitation de la taille d'un mail. Le problème qui survient avec l'application est que les fichiers envoyés ne peuvent dépasser 10 Mo au risque de recevoir un message de refus de la part de la boîte mail.

Ce problème reste pour l'instant géré par un message d'avertissement indiquant que tout fichier dépassant la limite autorisée de 10 Mo ne sera pas synchronisé.

Cette erreur est causée par le fait d'utiliser un système basé sur la solution « 1 mail = 1 fichier ». Le programme n'attribue qu'un seul mail par fichier ce qui le rend parfois trop volumineux.

La solution pour résoudre ce souci de taille consiste à attribuer plusieurs mails lorsque le fichier est trop volumineux.



Chaque mail du même fichier possède un identifiant et lorsque le fichier doit être téléchargé, tous les mails correspondant à ce fichier sont téléchargés puis leur contenu en Base64 est mis bout à bout afin d'obtenir le fichier complet.

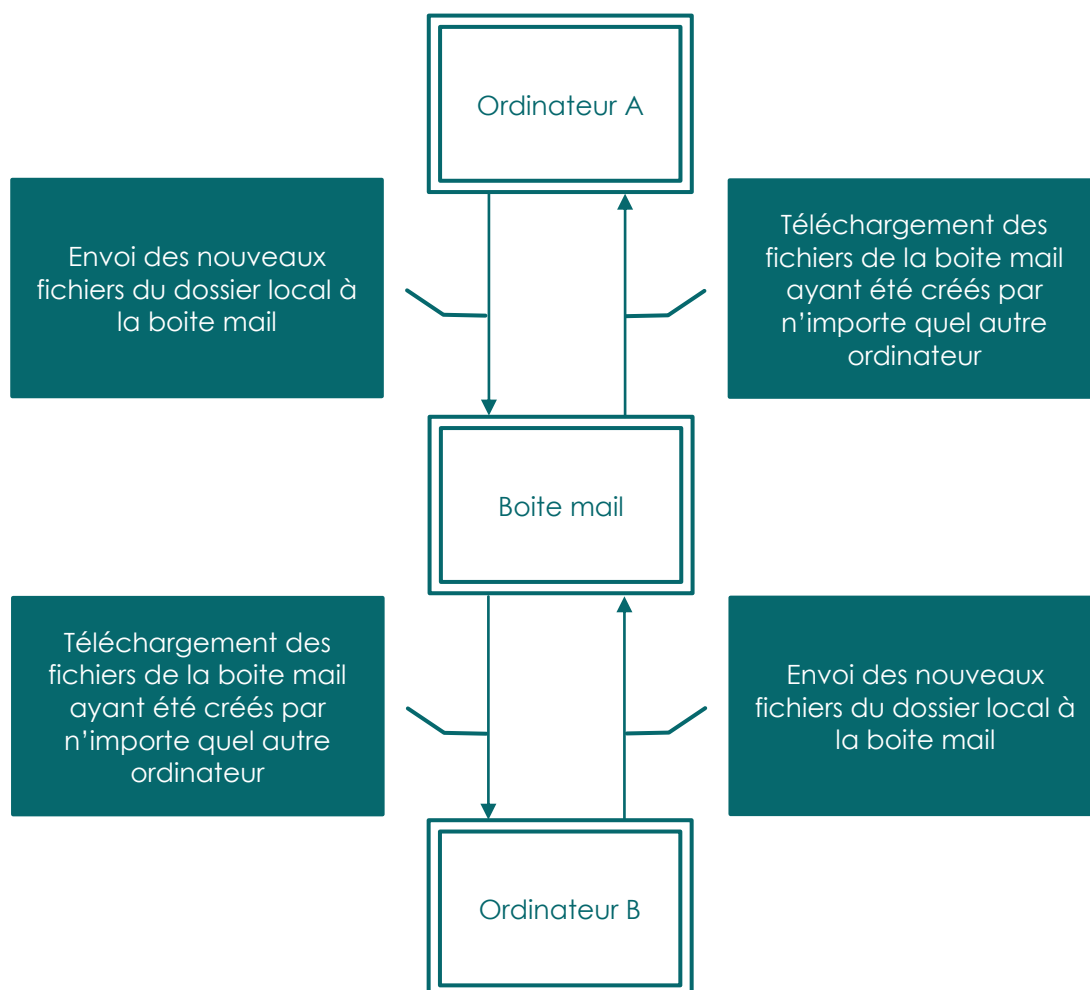
5.2.2 Exécution simultanée sur plusieurs ordinateurs

Le programme a été conçu pour ne fonctionner qu'avec une seule instance à la fois. C'est-à-dire que la personne qui veut utiliser le programme sur deux ordinateurs de doit pas le lancer deux fois en même temps, au risque d'avoir des conflits de traitement de fichiers.

Une solution pour résoudre ce problème et gérer automatiquement l'utilisation simultanée du programme serait de donner à chaque fichier dans la boîte mail une trace de quel ordinateur l'a créé, l'adresse mac par exemple.

Cela permettrait au programme en cours d'exécution de savoir quel fichier télécharger et quel fichier supprimer.

Sans cette trace de création, le programme verrait un fichier dans la boîte mail qu'il ne possède pas (ayant été rajouté par un autre ordinateur) et le supprimerait.



5.3 Bilan personnel

La conception de ce projet s'est faite en plusieurs étapes dans lesquelles j'ai dû organiser, planifier et gérer les tâches à accomplir.

Lors de la planification, j'ai immédiatement dû retenir quelles fonctionnalités qui allaient me prendre du temps pour implémenter. Ma vision ambitieuse de ce projet m'a amené à la réflexion lorsque j'ai dû faire face à mes premiers soucis de délais. Bien que les fonctionnalités demandées dans le cahier des charges aient été toutes réalisées, j'ai dû abandonner l'implémentation de quelques options supplémentaires comme la gestion des fichiers volumineux ou l'utilisation simultanée du programme pour laisser la place à des résolutions de bugs et l'amélioration de la stabilité générale de l'application.

Grâce à l'étape d'analyse, j'ai pu visualiser globalement à quoi ressemblerait l'application une fois terminée. J'ai élaboré des solutions pour toutes les implémentations dont la création semblait superflue et préparé au mieux l'environnement de travail avec lequel toute l'application serait façonnée.

C'est lors de l'étape de réalisation que j'ai pu mieux approfondir mes connaissances au niveau de l'utilisation du protocole IMAP, de la gestion d'une boîte mail ou encore de la manipulation de fichiers dans un dossier Windows, le tout en langage C#. L'exploitation de bibliothèques a changé la façon dont je visualise et conçois mon code. Ces « extensions » permettent une implémentation plus rapide de certaines fonctionnalités.

Pendant la phase de test, j'ai fait face à de nombreux problèmes et bugs qui ont parfois mis en danger le fonctionnement général de l'application. C'est à cause de ces problèmes rencontrés que certaines fonctionnalités supplémentaires ont été abandonnées au profit de la résolution de bugs et l'amélioration du programme.

Pour résumer, je dirais que ce projet m'a permis de mieux visualiser la charge de travail à accomplir par rapport à un cahier des charges donné et à mieux gérer mon temps pour la réalisation d'une application.

Divers

6 DIVERS

6.1 Journal de travail

6.1.1 Semaine 1

Semaine 1		Date: lundi, 1 mai 2017	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Lancement : Lecture du cahier des charges	9	Lundi : Réception et lecture du cahier des charges Lundi : Recherches sur le sujet Lundi : Questions au chef de projet à propos du cahier des charges Lundi : Rencontre avec M. Montemayor, un des experts du TPI	D-mazieroma-TPI-cahierDesCharges.docx
Lancement : Création et rédaction de la planification initiale	15	Lundi : Création du fichier de planification Lundi : Création du fichier de journal de travail Mercredi : Création des tâches Mercredi : Ajout des tâches dans la planification Mercredi : Modification des heures, ajout du temps de congé et d'examen	D-mazieroma-TPI-journalDeTravail.xlsm
Documentation : Rédaction du rapport de projet	9	Mercredi : Création du fichier de rapport de projet Mercredi : Ajout des titres et de la table des matières Mercredi : Rédaction des contenus de base (Introduction, description, etc ...)	D-mazieroma-TPI-rapportDeProjet.docx
Lancement : Mise en place de l'environnement	6	Mercredi : Installation de Visual Studio Community 2017 Mercredi : Installation de ReSharper Ultimate Mercredi : Création du projet "MailStorage" Mercredi : Création du projet sur GitHub Mercredi : Premier commit	
Analyse : Recherche des technologies utilisables	3	Mercredi : Ajout du framework MailKit Mercredi : Ajout du framework SQLite	https://github.com/jstedfast/MailKit https://system.data.sqlite.org/index.html/doc/trunk/www/index.wiki
Analyse : Architecture du projet	12	Jeudi : Création d'un schéma représentant les différents éléments du programme ainsi que le placement des classes	
Documentation : Rédaction du rapport de projet	12	Jeudi : Ajout des spécifications du projet Jeudi : Ajout de la description du projet Jeudi : Rédaction de la section "opportunités" Jeudi : Ajout de l'opportunité "Utilisation du protocole IMAP" Jeudi : Ajout de l'opportunité "Stockage des fichiers sur boîte mail" Jeudi : Ajout de l'opportunité "Approfondissement des connaissances" Jeudi : Documentation sur l'architecture du projet Jeudi : Documentation sur l'interface du projet	
Analyse : Fonctionnalité de connexion IMAP	6	Vendredi : Analyse de la validation du formulaire de connexion et l'identification du serveur IMAP Vendredi : Analyse de la création de la connexion IMAP dans l'application	
Documentation : Rédaction du rapport de projet	12	Vendredi : Rédaction de l'analyse de la vérification du formulaire de connexion Vendredi : Rédaction de l'analyse de la fonctionnalité de connexion IMAP Vendredi : Rédaction de l'analyse des interactions avec la base de données	
Analyse : Détection des changements sur les fichiers	6	Vendredi : Analyse sur la détection des fichiers modifiés ou ajoutés	
Documentation : Rédaction du rapport de projet	9	Vendredi : Documentation sur l'analyse des fichiers modifiés ou ajoutés	

6.1.2 Semaine 2

Semaine 2		Date: lundi, 8 mai 2017	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Documentation : Rédaction du rapport de projet	12	Lundi : Documentation sur l'analyse de la détection des fichiers modifiés ou ajoutés Lundi : Ajout de la planification initiale dans le rapport de projet Lundi : Modification de l'apparence du cahier des charges dans le rapport de projet	
Analyse : Organisation des fichiers dans la boîte mail	3	Mercredi : Analyse de l'organisation des fichiers dans la boîte mail	
Analyse : Fonctionnalité de synchronisation	3	Mercredi : Analyse de la fonctionnalité de synchronisation des fichiers "dossier - boîte mail"	http://snipplr.com/view/6034/base64-encode-or-decode-a-file/
Documentation : Rédaction du rapport de projet	7	Mercredi : Documentation sur la façon d'organiser les fichiers dans la boîte mail Mercredi : Documentation de la fonction de synchronisation des fichiers "dossier - boîte mail"	
Test : Préparation des tests	13	Mercredi : Création des 10 scénarios de test comme demandé dans le cahier des charges Mercredi : Documentation des 10 scénarios dans le rapport de projet	
Implémentation : Création de l'interface utilisateur	16	Mercredi : Début de création de l'interface utilisateur Jeudi : Ajout de la fenêtre de login Jeudi : Ajout de la fenêtre de traitement des fichiers	
Implémentation : Connexion au serveur IMAP	16	Jeudi : Création de la classe de gestion de la base de données Jeudi : Création de la classe de gestion des connexions IMAP Jeudi : Création de la fonction de vérification du formulaire de connexion Jeudi : Création de l'enregistrement des données de l'utilisateur dans la base de données Jeudi : Création de la fonction de connexion et authentification au serveur IMAP	
Documentation : Rédaction du rapport de projet	16	Vendredi : Modification de l'analyse, changement des images de l'interface Vendredi : Modification de l'analyse, changement de la partie de vérification du formulaire Vendredi : Documentation de l'implémentation de l'interface, description des éléments de la fenêtre de login Vendredi : Documentation de l'implémentation de l'interface, description des fonctions de la fenêtre de login	
Implémentation : Scan des fichiers modifiés	13	Vendredi : Implémentation de la fonctionnalité de détection des fichiers modifiés Vendredi : Création de la classe de gestion des fichiers Vendredi : Création de la classe de fichiers Vendredi : Lecture d'un document sur les identifiants uniques des fichiers Windows Vendredi : Création de la fonctionnalité de mise à jour des listes de fichiers locaux	http://haishibai.blogspot.ch/2010/08/track-file-when-its-modified-moved-or.html

6.1.3 Semaine 3

Semaine 3		Date: lundi, 15 mai 2017	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Implémentation : Scan des fichiers modifiés	3	Lundi : Modification dans la détection des fichiers supprimés	
Documentation : Rédaction du rapport de projet	10	Lundi : Documentation sur les interactions avec le serveur IMAP Lundi : Documentation sur les interactions avec la base de données SQLite	
Implémentation : Ajout des fichiers modifiés à la boîte mail	28	Mercredi : Ajout de fonctions pour envoyer et supprimer des mails dans le dossier MailStorage de la boîte mail Mercredi : Ajout de la fonction de mise à jour de la liste des fichiers locaux Mercredi : Ajout de la fonction de mise à jour de la liste de fichiers distants Mercredi : Ajout de la fonction d'upload des fichiers locaux vers la boîte mail Mercredi : Ajout de la fonction de suppression des fichiers de la boîte mail	
Implémentation : Téléchargement des fichiers plus récents en local	30	Jeudi : Ajout de la fonction de synchronisation initiale Jeudi : Ajout de la fonction de téléchargement de fichiers depuis la boîte mail Jeudi : Possibilité de changement de dossier ajoutée Jeudi : Messages d'avertissement lors du changement de dossier ajoutés Jeudi : Ajout de la fonction qui gère la barre de status Jeudi : Modification du bouton "retour" Jeudi : Abandon du système d'identifiant unique pour chaque fichier car impossible sur FAT32 Jeudi : Modification de la synchronisation automatique, abandon du timer	
Documentation : Rédaction du rapport de projet	28	Vendredi : Modifications dans la partie Analyse Vendredi : Rencontre avec M. Moren, un des experts du TPI Vendredi : Documentation sur la réalisation de l'interface de fenêtre de traitement des fichiers Vendredi : Documentation de la réalisation des interactions avec la boîte mail Vendredi : Ajout de nouvelles sections dans la partie "réalisation" de la documentation Vendredi : Documentation des fonctions de la classe MailStorage Vendredi : Documentation des fonctions de la classe MailManager	

6.1.4 Semaine 4

Semaine 4		Date: lundi, 22 mai 2017	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Test : Exécution des tests	15	Lundi : Exécution des tests et documentation des résultats Lundi : Création de petits tests supplémentaires	
Autres : Résolution de bug	27	Mercredi : Résolution du problème de déconnexion du serveur mail au clic sur le bouton "retour" Mercredi : Restriction d'écriture dans le champ de dossier racine dans la page de login Mercredi : Suppression des dossiers lors de la synchronisation initiale Mercredi : Résolution de l'affichage de la barre d'espace disponible Mercredi : Cryptage du mot de passe de l'utilisateur dans la base de données Mercredi : Résolution d'un problème de threads avec la mise à jour de l'espace de stockage et la récupération des mails Mercredi : Ajout d'une animation de chargement lors de la connexion au serveur IMAP Mercredi : Ajout de l'affichage du traitement des fichiers. Possibilité de voir quel fichier est envoyé ou téléchargé Mercredi : Résolution d'un problème où la surveillance du dossier racine ne se mettait pas à jour Mercredi : Résolution du problème de synchronisation automatique de plusieurs fichiers groupés Mercredi : Ajouts d'option de synchronisation finale avant la fermeture de l'application Mercredi : Limitation du nombre de caractères dans les champs du formulaire de login	https://gitlab.com/czubehead/csharp-aes https://dotnetcodr.com/2014/01/01/5-ways-to-start-a-task-in-net-c/ https://stackoverflow.com/questions/9112305/how-to-asynchronously-wait-for-x-seconds-and-execute-something-then
Autres : Congé / Vacances / Examens	57	Congés de l'ascension (JEU + VEN)	

6.1.5 Semaine 5

Semaine 5		Date: lundi, 29 mai 2017	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Autres : Congé / Vacances / Examens	13	Examen de culture générale	
Autres : Résolution de bug	6	Lundi : Résolution d'un problème de synchronisation des fichiers modifiés	
Autres : Travail en plus	9	Lundi : Ajout de la fonctionnalité d'indexation des dossiers	
Autres : Résolution de bug	6	Mercredi : Résolution d'un bug de synchronisation lorsqu'un fichier est ouvert	
Documentation : Rédaction du rapport de projet	9	Mercredi : Mise à jour de la partie réalisation du rapport avec les nouvelles fonctionnalités ajoutées Mercredi : Rédaction du début de la partie de synchronisation des fichiers dans la réalisation	
Documentation : Rédaction du rapport de projet	12	Jeudi : Documentation de la réalisation de la gestion des fichiers Jeudi : Documentation sur la réalisation de la mise à jour de la liste locale	
Autres : Résolution de bug	12	Jeudi : Résolution d'un bug de téléchargement des fichiers lors de la synchronisation au démarrage	
Documentation : Rédaction du journal de travail	5	Jeudi : Modification de l'analyse des différents types de synchronisation de fichiers	
Autres : Résolution de bug	12	Vendredi : Résolution d'un bug avec la création du dossier "MailStorage" dans la boîte mail	
Test : Exécution des tests	3	Vendredi : Derniers tests de l'application, vérification des fonctionnalités	
Documentation : Rédaction du rapport de projet	12	Vendredi : Création du diagramme de flux du fonctionnement de l'application Vendredi : Mise à jour de la partie réalisation Vendredi : Modifications de cohérence dans l'analyse Vendredi : Début de création des pages de titre	
Documentation : Rédaction du rapport de projet	0	Week-End : Documentation des différentes synchronisations Week-End : Documentation des fonctions de la classe FileManager Week-End : Documentation de la conclusion Week-End : Documentation des fonctionnalités supplémentaires Week-End : Rédaction de l'avis personnel Week-End : Rédaction de la webographie	

6.1.6 Semaine 6

Semaine 6		Date: lundi, 5 juin 2017	
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
Autres : Congé / Vacances / Examens	13	Lundi de Pentecôte	
Documentation : Rédaction du rapport de projet	13	Mercredi : Correction des fautes d'orthographe Mercredi : Ajustements de la mise en page Mercredi : Ajout du journal de travail au rapport Mercredi : Relecture du rapport	

6.2 Gantt final

6.2.1 Semaine 1

Tâches - objectifs	Nb 1/4 heure	S 1
Absence - Imprévu	0	
	0	
Lancement : Lecture du cahier des charges	9	
	9	
Lancement : Création et rédaction de la planification	12	
	15	
Lancement : Mise en place de l'environnement	9	
	6	
Analyse : Recherche des technologies utilisables	6	
	3	
Analyse : Architecture du projet	6	
	12	
Analyse : Fonctionnalité de connexion IMAP	6	
	6	
Analyse : Détection des changements sur les fichiers	9	
	6	
Analyse : Organisation des fichiers dans la boîte mail	6	
	3	
Analyse : Fonctionnalité de synchronisation	12	
	3	
Analyse : Interface utilisateur	3	
	0	
Implémentation : Création de l'interface utilisateur	12	
	16	
Implémentation : Connexion au serveur IMAP	12	
	16	
Implémentation : Scan des fichiers modifiés	15	
	16	
Implémentation : Ajout des fichiers modifiés à la boîte mail	18	
	28	
Implémentation : Téléchargement des fichiers plus récents	18	
	30	
Test : Préparation des tests	12	
	13	
Test : Exécution des tests	18	
	18	
Documentation : Rédaction du rapport de projet	198	
	161	
Documentation : Rédaction du journal de travail	0	
	5	
Autres : Résolution de bug	21	
	63	
Autres : Déploiement de l'application	15	
	0	
Autres : Travail en plus	21	
	9	
Autres : Congé / Vacances / Examens	83	
	83	
Total planifié	521	
Total réalisé	521	

6.2.2 Semaine 2

Tâches - objectifs	Nb 1/4 heure	S 2
Absence - Imprévu	0	
	0	
Lancement : Lecture du cahier des charges	9	
	9	
Lancement : Création et rédaction de la planification	12	
	15	
Lancement : Mise en place de l'environnement	9	
	6	
Analyse : Recherche des technologies utilisables	6	
	3	
Analyse : Architecture du projet	6	
	12	
Analyse : Fonctionnalité de connexion IMAP	6	
	6	
Analyse : Détection des changements sur les fichiers	9	
	6	
Analyse : Organisation des fichiers dans la boîte mail	6	
	3	
Analyse : Fonctionnalité de synchronisation	12	
	3	
Analyse : Interface utilisateur	3	
	0	
Implémentation : Création de l'interface utilisateur	12	
	16	
Implémentation : Connexion au serveur IMAP	12	
	16	
Implémentation : Scan des fichiers modifiés	15	
	16	
Implémentation : Ajout des fichiers modifiés à la boîte mail	18	
	28	
Implémentation : Téléchargement des fichiers plus récents	18	
	30	
Test : Préparation des tests	12	
	13	
Test : Exécution des tests	18	
	18	
Documentation : Rédaction du rapport de projet	198	
	161	
Documentation : Rédaction du journal de travail	0	
	5	
Autres : Résolution de bug	21	
	63	
Autres : Déploiement de l'application	15	
	0	
Autres : Travail en plus	21	
	9	
Autres : Congé / Vacances / Examens	83	
	83	
Total planifié	521	
Total réalisé	521	

6.2.3 Semaine 3

Tâches - objectifs	Nb 1/4 heure	S 3
Absence - Imprévu	0	
	0	
Lancement : Lecture du cahier des charges	9	
	9	
Lancement : Création et rédaction de la planification	12	
	15	
Lancement : Mise en place de l'environnement	9	
	6	
Analyse : Recherche des technologies utilisables	6	
	3	
Analyse : Architecture du projet	6	
	12	
Analyse : Fonctionnalité de connexion IMAP	6	
	6	
Analyse : Détection des changements sur les fichiers	9	
	6	
Analyse : Organisation des fichiers dans la boîte mail	6	
	3	
Analyse : Fonctionnalité de synchronisation	12	
	3	
Analyse : Interface utilisateur	3	
	0	
Implémentation : Création de l'interface utilisateur	12	
	16	
Implémentation : Connexion au serveur IMAP	12	
	16	
Implémentation : Scan des fichiers modifiés	15	
	16	
Implémentation : Ajout des fichiers modifiés à la boîte	18	
	28	
Implémentation : Téléchargement des fichiers plus récents	18	
	30	
Test : Préparation des tests	12	
	13	
Test : Exécution des tests	18	
	18	
Documentation : Rédaction du rapport de projet	198	
	161	
Documentation : Rédaction du journal de travail	0	
	5	
Autres : Résolution de bug	21	
	63	
Autres : Déploiement de l'application	15	
	0	
Autres : Travail en plus	21	
	9	
Autres : Congé / Vacances / Examens	83	
	83	
Total planifié	521	
Total réalisé	521	

6.2.4 Semaine 4

Tâches - objectifs	Nb 1/4 heure	S 4
Absence - Imprévus	0	
	0	
Lancement : Lecture du cahier des charges	9	
	9	
Lancement : Création et rédaction de la planification	12	
	15	
Lancement : Mise en place de l'environnement	9	
	6	
Analyse : Recherche des technologies utilisables	6	
	3	
Analyse : Architecture du projet	6	
	12	
Analyse : Fonctionnalité de connexion IMAP	6	
	6	
Analyse : Détection des changements sur les fichiers	9	
	6	
Analyse : Organisation des fichiers dans la boîte mail	6	
	3	
Analyse : Fonctionnalité de synchronisation	12	
	3	
Analyse : Interface utilisateur	3	
	0	
Implémentation : Création de l'interface utilisateur	12	
	16	
Implémentation : Connexion au serveur IMAP	12	
	16	
Implémentation : Scan des fichiers modifiés	15	
	16	
Implémentation : Ajout des fichiers modifiés à la boîte mail	18	
	28	
Implémentation : Téléchargement des fichiers plus récents	18	
	30	
Test : Préparation des tests	12	
	13	
Test : Exécution des tests	18	
	18	
Documentation : Rédaction du rapport de projet	198	
	161	
Documentation : Rédaction du journal de travail	0	
	5	
Autres : Résolution de bug	21	
	63	
Autres : Déploiement de l'application	15	
	0	
Autres : Travail en plus	21	
	9	
Autres : Congé / Vacances / Examens	83	
	83	
Total planifié	521	
Total réalisé	521	

6.2.5 Semaine 5

Tâches - objectifs	Nb 1/4 heure	S 5
Absence - Imprévus	0	
	0	
Lancement : Lecture du cahier des charges	9	
	9	
Lancement : Création et rédaction de la planification	12	
	15	
Lancement : Mise en place de l'environnement	9	
	6	
Analyse : Recherche des technologies utilisables	6	
	3	
Analyse : Architecture du projet	6	
	12	
Analyse : Fonctionnalité de connexion IMAP	6	
	6	
Analyse : Détection des changements sur les fichiers	9	
	6	
Analyse : Organisation des fichiers dans la boîte mail	6	
	3	
Analyse : Fonctionnalité de synchronisation	12	
	3	
Analyse : Interface utilisateur	3	
	0	
Implémentation : Création de l'interface utilisateur	12	
	16	
Implémentation : Connexion au serveur IMAP	12	
	16	
Implémentation : Scan des fichiers modifiés	15	
	16	
Implémentation : Ajout des fichiers modifiés à la boîte mail	18	
	28	
Implémentation : Téléchargement des fichiers plus récents	18	
	30	
Test : Préparation des tests	12	
	13	
Test : Exécution des tests	18	
	18	
Documentation : Rédaction du rapport de projet	198	
	161	
Documentation : Rédaction du journal de travail	0	
	5	
Autres : Résolution de bug	21	
	63	
Autres : Déploiement de l'application	15	
	0	
Autres : Travail en plus	21	
	9	
Autres : Congé / Vacances / Examens	83	
	83	
Total planifié	521	
Total réalisé	521	

6.2.6 Semaine 6

Tâches - objectifs	Nb 1/4 heure	S 6
Absence - Imprévu	0	
	0	
Lancement : Lecture du cahier des charges	9	
	9	
Lancement : Création et rédaction de la planification	12	
	15	
Lancement : Mise en place de l'environnement	9	
	6	
Analyse : Recherche des technologies utilisables	6	
	3	
Analyse : Architecture du projet	6	
	12	
Analyse : Fonctionnalité de connexion IMAP	6	
	6	
Analyse : Détection des changements sur les fichiers	9	
	6	
Analyse : Organisation des fichiers dans la boîte mail	6	
	3	
Analyse : Fonctionnalité de synchronisation	12	
	3	
Analyse : Interface utilisateur	3	
	0	
Implémentation : Création de l'interface utilisateur	12	
	16	
Implémentation : Connexion au serveur IMAP	12	
	16	
Implémentation : Scan des fichiers modifiés	15	
	16	
Implémentation : Ajout des fichiers modifiés à la boîte mail	18	
	28	
Implémentation : Téléchargement des fichiers plus récents	18	
	30	
Test : Préparation des tests	12	
	13	
Test : Exécution des tests	18	
	18	
Documentation : Rédaction du rapport de projet	198	
	161	
Documentation : Rédaction du journal de travail	0	
	5	
Autres : Résolution de bug	21	
	63	
Autres : Déploiement de l'application	15	
	0	
Autres : Travail en plus	21	
	9	
Autres : Congé / Vacances / Examens	83	
	83	
Total planifié	521	
Total réalisé	521	

6.3 Webographie

GitHub

« MailKit »

Jstedfast

<https://github.com/jstedfast/MailKit>

System.Data.SQLite

« SQLite »

<https://system.data.sqlite.org/index.html/doc/trunk/www/index.wiki>

Snipplr

« Base64 Encode or Decode a File »

Rengber

<http://snipplr.com/view/6034/base64-encode-or-decode-a-file/>

Haishi's Blog

« Track a file when it's modified, moved, or renamed under NTFS using C# »

Haishi

<http://haishibai.blogspot.ch/2010/08/track-file-when-its-modified-moved-or.html>

GitLab

« csharp-aes »

Petr Čech

<https://gitlab.com/czubehead/csharp-aes>

Stackoverflow

« How to asynchronously wait for x seconds and execute something then? »

Dennis G

<https://stackoverflow.com/questions/9112305/how-to-asynchronously-wait-for-x-seconds-and-execute-something-then>

6.4 Cahier des charges

1 INFORMATIONS GENERALES

Candidat :	Nom : MAZIERO	Prénom : MARCO
	✉ : mazieroma@etml.educanet2.ch	☎ : 0794395962
Lieu de travail :	ETML, Sébeillon 12 1004 Lausanne	
Chef de projet :	Nom : Melly	Prénom : Jonathan
	✉ : jonathan.melly@vd.educa.net2.ch	☎ : 021 316 78 14
Expert 1 :	Nom : Montemayor	Prénom : Ernesto
	✉ : ernesto@bati-technologie.ch	☎ : 079 606 33 28
Expert 2 :	Nom : Moren	Prénom : Auxilio
	✉ : auxilio.moren@orif.ch	☎ : 024 468 67 07
Dates de réalisation :	Du lundi 1 mai au mercredi 7 juin à 11H25	
Horaire de travail : (Basé sur l'horaire officiel)	Lundi	08h00-11h25 - Pentecôte. 5 juin 2017
	Mardi	- -
	Mercredi	08h00-12H15 13h10-16h35 Exa CG 31 mai 2017 après-midi
	Jeudi	08h00-11H25 12h20-16h35 Pont de l'Asc. 25 mai 2017
	Vendredi	08h00-12H15 13h10-16h35 Pont de l'Asc. 26 mai 2017
Présentation :	Entre le mercredi 14 et jeudi 15 juin 2017	
Nombre d'heures :	Environ 110 heures	
Planning (en H ou %)	Analyse : 20%, Implémentation 35%, Tests 10%, Doc. 35%	

2 PROCÉDURE

Le candidat réalise un travail personnel sur la base d'un cahier des charges reçu le 1er jour.

Le cahier des charges est approuvé par la i-CQ VD. Il est en outre présenté, commenté et discuté avec le candidat. Par sa signature, le candidat accepte le travail proposé.

Le candidat a connaissance de la feuille d'appréciation avant de débiter le travail.

Le candidat est entièrement responsable de la sécurité de ses données.

En cas de problèmes graves, le candidat avertit au plus vite les deux experts et son chef de projet.

Le candidat a la possibilité d'obtenir de l'aide, mais doit le mentionner dans son dossier de projet.

A la fin du délai imparti pour la réalisation du TPI, le candidat doit transmettre par courrier électronique le dossier de projet aux deux experts et au chef de projet. En parallèle, une copie papier du rapport doit être fournie sans délai en trois exemplaires. Cette dernière doit être en tout point identique à la version électronique.

3 TITRE

MailStorage

4 SUJET

MailStorage est une application qui permet d'utiliser une boîte mail accessible avec le protocole IMAP comme système de fichier (similaire au projet Imapfs).

5 MATÉRIEL ET LOGICIEL À DISPOSITION

1 ordinateur ETML standard (Win7, VisualStudio Community)

6 PRÉREQUIS

Avoir suivi les modules ICT relatifs à la programmation en C#.

7 DESCRIPTIF DU PROJET

L'objectif est de fournir une application qui permette d'utiliser l'espace de stockage d'une boîte mail et les spécifications du protocole IMAP afin de stocker des fichiers (dans une arborescence) comme on le fait sur un système de fichier comme FAT32.

Cette application doit donc permettre à l'utilisateur d'utiliser un compte IMAP comme support de stockage distant pour offrir une fonctionnalité ressemblante à GoogleDrive ou DropBox.

8 POINTS ÉVALUÉS DURANT LE PROJET

8.1 Documentation

- 1 diagramme de flux montrant le fonctionnement de l'application
- 1 diagramme expliquant le protocole utilisé pour convertir les opérations fichier <--> IMAP
- 10 scénarios de test reprenant les fonctionnalités décrites ci-dessous avec leur résultat

8.2 Fonctionnalités

- Compte IMAP
 - Configuration du serveur, port, login et mot de passe
 - Affichage de l'espace disponible
- Système de fichier
 - Créer, renommer, supprimer un dossier / fichier
 - Ouvrir / Éditer un fichier
 - Métadonnées
 - Fichiers
 - Nom
 - Date de création
 - Taille
 - Dossier
 - Nom
 - Date de création

- Droits
 - Non gérés (comme pour FAT32)
- GUI
 - Utiliser l'explorateur du système d'exploitation et une fonction de synchronisation (comme le client Dropbox par exemple)

9 LIVRABLES

Le candidat est responsable de livrer à son chef de projet et aux deux experts :

- Une planification initiale
- Un rapport de projet
- Un journal de travail

10 POINTS TECHNIQUES ÉVALUÉS SPÉCIFIQUES AU PROJET

La grille d'évaluation définit les critères généraux selon lesquels le travail du candidat sera évalué (documentation, journal de travail, respect des normes, qualité, ...).

En plus de cela, le travail sera évalué sur les trois points spécifiques suivants :

1. *La solution sera validée sur l'environnement de travail du chef de projet*
2. *Les sources du projet seront stockées quotidiennement (selon horaire) sur Github*
3. *L'application sera packagée avec un assistant d'installation (par exemple InnoSetup)*

11 VALIDATION

	Lu et approuvé le :	Signature :
Candidat :		
Expert n°1 :		
Expert n° 2 :		
Chef de projet :		