Rapport de TPI - MailStorage

Maziero Marco – CIN4A

Lausanne - ETML

Durée : 5 semaines

Chef de projet : Melly Jonathan - [jonathan.melly@vd.educanet2.ch](mailto: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 4](#_Toc481748438)

[1.1 Titre 4](#_Toc481748439)

[1.2 Description 4](#_Toc481748440)

[1.3 Matériel et logiciels à disposition 4](#_Toc481748441)

[1.4 Prérequis 4](#_Toc481748442)

[2 Planification Initiale 5](#_Toc481748443)

[2.1 Liste des tâches 5](#_Toc481748444)

[2.2 Diagramme de Gantt 5](#_Toc481748445)

[3 Analyse 6](#_Toc481748446)

[3.1 Opportunités 6](#_Toc481748447)

[3.1.1 Utilisation du protocole IMAP 6](#_Toc481748448)

[3.1.2 Stockage de fichiers sur boite mail 7](#_Toc481748449)

[3.1.3 Approfondissement des connaissances 8](#_Toc481748450)

[3.2 Document d’analyse et conception 8](#_Toc481748451)

[3.2.1 Architecture du projet 8](#_Toc481748452)

[3.2.2 Interfaces 11](#_Toc481748453)

[3.2.3 Vérification du formulaire de connexion 13](#_Toc481748454)

[3.2.4 Connexion IMAP 15](#_Toc481748455)

[3.2.5 Interactions base de données 16](#_Toc481748456)

[3.2.6 Détection des modifications de fichiers 17](#_Toc481748457)

[3.3 Conception des tests 17](#_Toc481748458)

[3.4 Planification détaillée 17](#_Toc481748459)

[3.4.1 Semaine 1 17](#_Toc481748460)

[3.4.2 Semaine 2 17](#_Toc481748461)

[3.4.3 Semaine 3 17](#_Toc481748462)

[3.4.4 Semaine 4 17](#_Toc481748463)

[3.4.5 Semaine 5 17](#_Toc481748464)

[4 Réalisation 17](#_Toc481748465)

[4.1 Dossier de Réalisation 17](#_Toc481748466)

[4.2 Dossier des tests 17](#_Toc481748467)

[5 Conclusion 17](#_Toc481748468)

[5.1 Bilan des fonctionnalités demandées 17](#_Toc481748469)

[5.2 Bilan personnel 17](#_Toc481748470)

[6 Divers 17](#_Toc481748471)

[6.1 Journal de travail 17](#_Toc481748472)

[6.2 Webographie 17](#_Toc481748473)

[6.3 Cahier des charges 17](#_Toc481748474)

[6.3.1 INFORMATIONS GENERALES 17](#_Toc481748475)

[6.3.2 PROCÉDURE 18](#_Toc481748476)

[6.3.3 TITRE 18](#_Toc481748477)

[6.3.4 SUJET 18](#_Toc481748478)

[6.3.5 MATÉRIEL ET LOGICIEL À DISPOSITION 19](#_Toc481748479)

[6.3.6 PRÉREQUIS 19](#_Toc481748480)

[6.3.7 DESCRIPTIF DU PROJET 19](#_Toc481748481)

[6.3.8 POINTS ÉVALUÉS DURANT LE PROJET 19](#_Toc481748482)

[6.3.9 LIVRABLES 20](#_Toc481748483)

[6.3.10 POINTS TECHNIQUES ÉVALUÉS SPÉCIFIQUES AU PROJET 20](#_Toc481748484)

[6.3.11 VALIDATION 21](#_Toc481748485)

# Spécifications

## Titre

MailStorage

## Description

MailStorage est une application qui permet d’utiliser l’espace disponible sur une boite 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 boite 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.

## 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 |

## Prérequis

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

# 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.

## 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 |

## 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 est basée sur cette planification initiale.

# 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.

## Opportunités

### Utilisation du protocole IMAP[[1]](#footnote-1)

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

Envoi et réception de données avec le protocole IMAP

Programme

MailStorage

Boite mail

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 boite 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 boite mail, contrairement au protocole POP qui ne permet que de récupérer la liste des mails présents dans la boite de réception.[[2]](#footnote-2)

Crée 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 SMTP ver un serveur mail spécifié en C#. Idéal pour l’envoi ou la réception de mails.

### Stockage de fichiers sur boite mail

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

Mail

**De :** [personne@mail.com](mailto:personne@mail.com)

**A :** [moi@mail.com](mailto:moi@mail.com)

**Sujet :** Titre du message

**Contenu :** Texte du message

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.

Dans ce projet, le but est de pouvoir exploiter la boite 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 boite 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 chaine de caractères sortante bien plus petite que si elle était en binaire.

Fichier en données Base64

Fichier en données binaires

01010011010101010100110101011010101101010110010101101101

1010101101010110010101101101

Fichier

Windows

5pWt

Une fois le fichier transcrit en texte Base64, il peut être stocké dans le contenu d’un mail. Le chemin d’accès au fichier peut être stocké dans le sujet du message, ainsi il reste toujours une trace du nom et de l’extension du fichier.

**De :** [mailstorage@mail.com](mailto:mailstorage@mail.com)

**A :** [moi@mail.com](mailto:moi@mail.com)

**Sujet :** C:\\Users\name\file.txt

**Contenu :** 5pWt

Mail

5pWt

Fichier en données Base64

Pour la récupération des fichiers, il suffit simplement de récupérer

### 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





## Document d’analyse et conception

### Architecture du projet

Le projet doit être capable de 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 pour communiquer en IMAP avec la boite mail
* Une classe pour communiquer avec la base de données
* Une classe pour gérer les fichiers

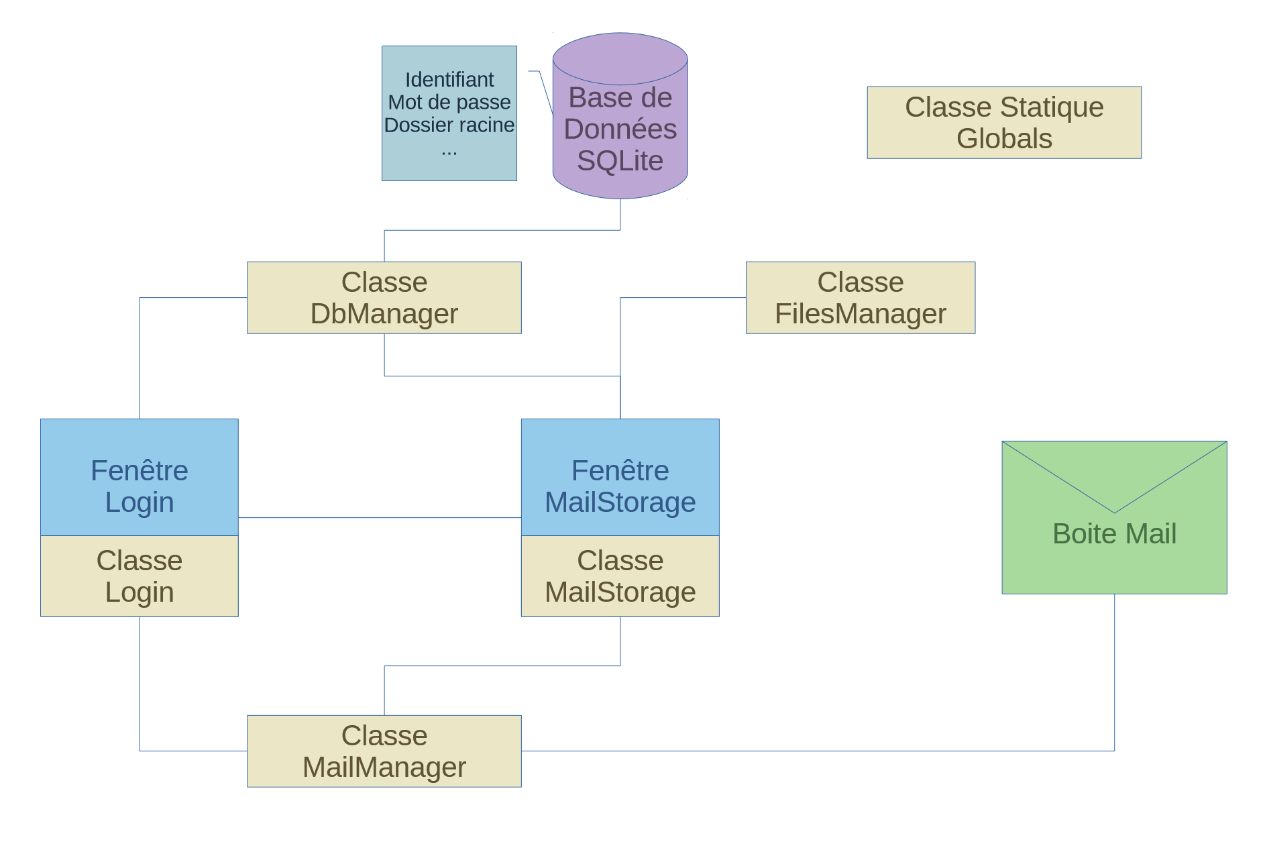
Les classes principales sont celles qui gèrent les fenêtres, c’est depuis ces éléments que les autres classes sont instanciées et appelées.

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

Classe Login

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 continuelle des fichiers s’exécute. Les fichiers ajoutés au dossier local sont envoyés à la boite mail et les fichiers manquants dans le dossier local y sont rajoutés.

Classe MailManager

Pour envoyer et recevoir des mails il faut une connexion IMAP constante. C’est le rôle de la classe *MailManager*. Les envois et réception 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 boite mail.

Classe 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 est ensuite placé dans le contenu du mail, tandis que les mails récupérés sont retransformés en fichiers afin d’être placés dans le dossier local.

Classe statique Globals

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

### Interfaces

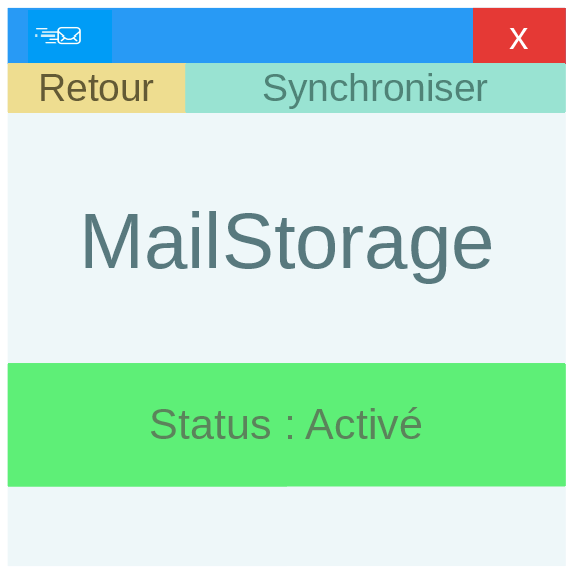
Le programme ne comporte que deux fenêtres car la majorité des manipulations se font dans l’explorateur Windows. La fenêtre principale est conçue pour tourner en arrière-plan afin que les fichiers se mettent à jour automatiquement.

Figure : 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 d’adresse mail, de mot de passe et de dossier racine sont spécifiés.

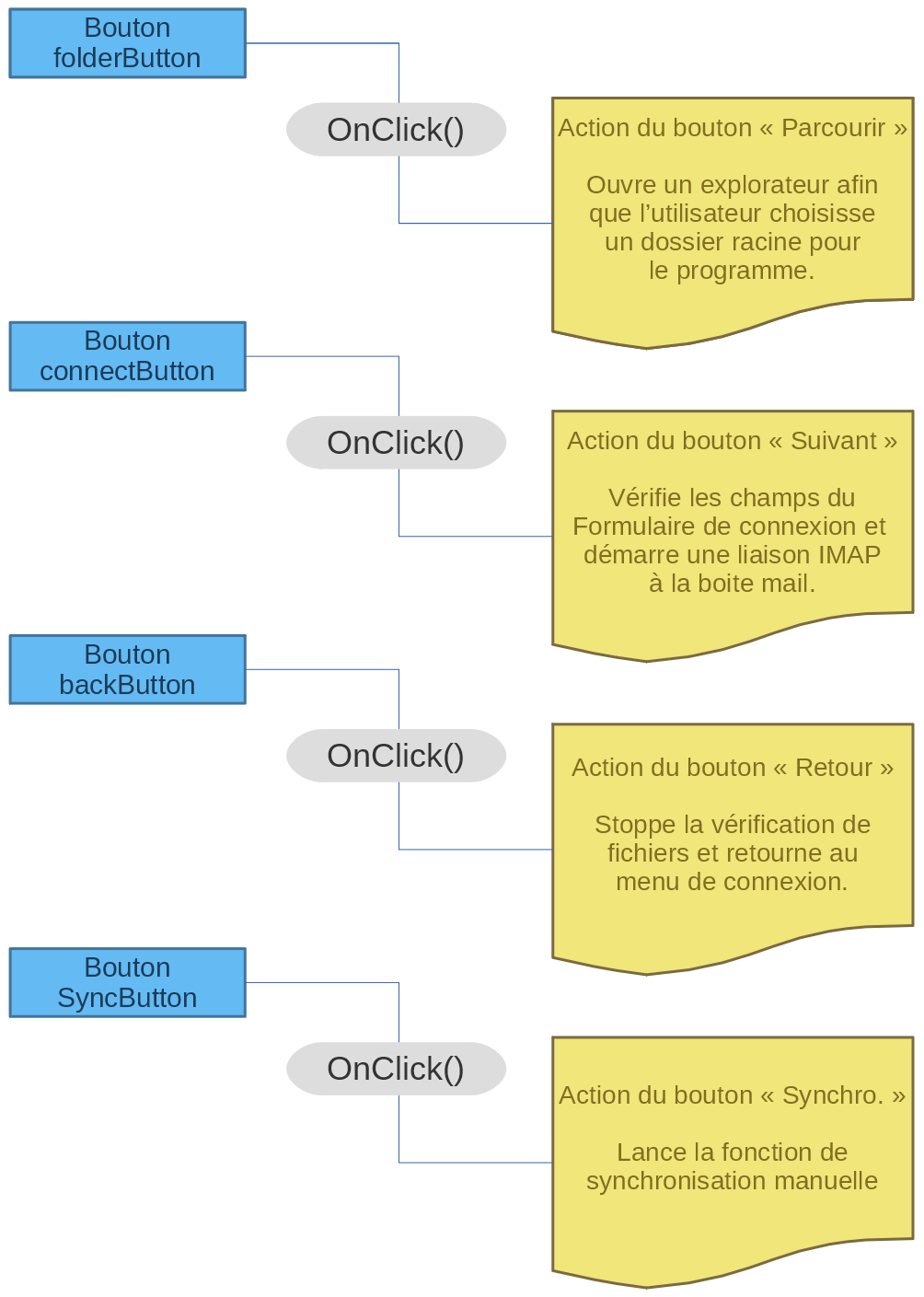
Une action sur le bouton suivant lance la tentative de connexion IMAP avec la boite mail.



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

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

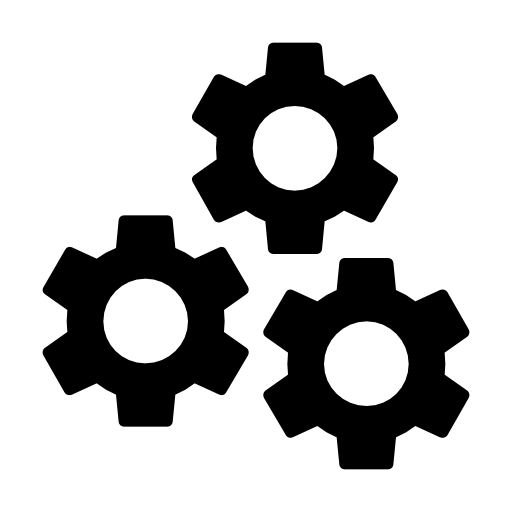
Sur la première fenêtre de connexion, deux boutons sont présents : Le bouton de sélection du dossier racine et le bouton de connexion à la boite mail.

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.

### 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 son adresse mail, le mot de passe de son compte mail et sélectionne le dossier racine ou il veut gérer ses fichiers. Il confirme ensuite 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 ou la validation retourne une erreur, un message d’avertissement s’affiche à l’écran pour avertir l’utilisateur.

Erreur :

Le mail entré n’est pas valide. Veuillez en spécifier un autre.

Valide :

Début de la connexion IMAP au serveur mail.

adresse@mail.com

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

Une fois le mail validé, il faut déterminer le type de serveur mail. En effet, chaque serveur possède un nom différent et le programme doit pouvoir l’identifier s’il veut établir une connexion.

La partie permettant de déterminer quel serveur mail utiliser est le texte juste après le symbole « @ ». Ci-dessous la liste des serveurs mails disponibles avec l’application :

|  |  |  |
| --- | --- | --- |
| Adresse mail | Serveur IMAP | Port |
| @gmail. | imap.gmail.com | 993 |
| @yahoo. | imap.mail.yahoo.com | 993 |
| @outlook. | imap-mail.outlook.com | 993 |
| @hotmail. | imap-mail.outlook.com | 993 |

Vérification du mot de passe

Aucune vérification particulière 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 corresponde 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 et s’assure simplement que le la valeur n’est pas vide et qu’elle contient du texte.

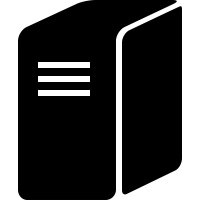
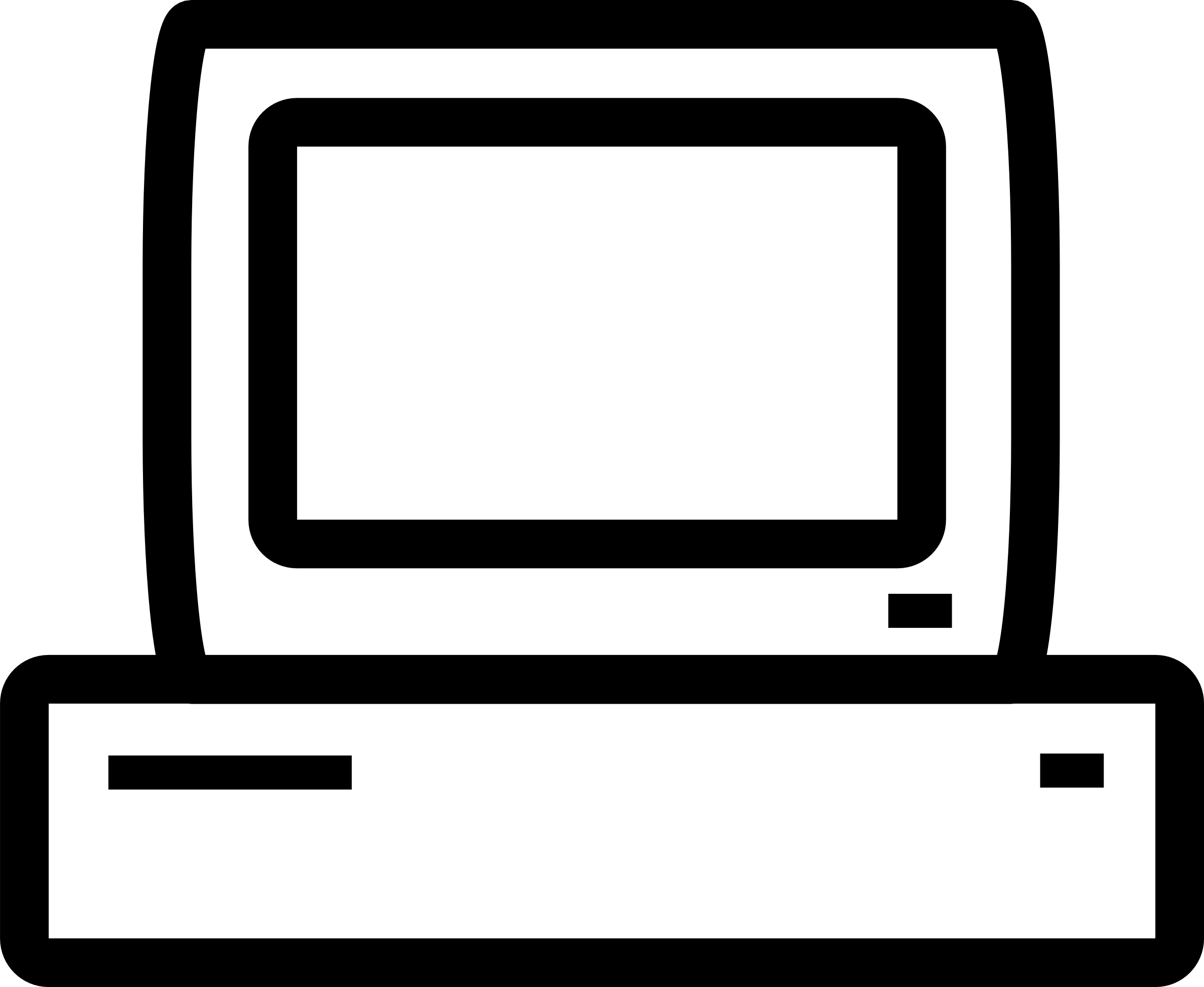
Dans ce champ, l’utilisateur ne peut rentrer des données manuellement. Pour remplir le champ il faut cliquer sur le bouton « Parcourir » afin qu’une fenêtre d’explorateur Windows s’ouvre pour que l’utilisateur puisse choisir le dossier racine qu’il souhaite.

L’ouverture d’un explorateur Windows est effectuée grâce à la classe *FolderBrowserDialog* disponible dans le namespace *System.Widows.Forms*.[[3]](#footnote-3)

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

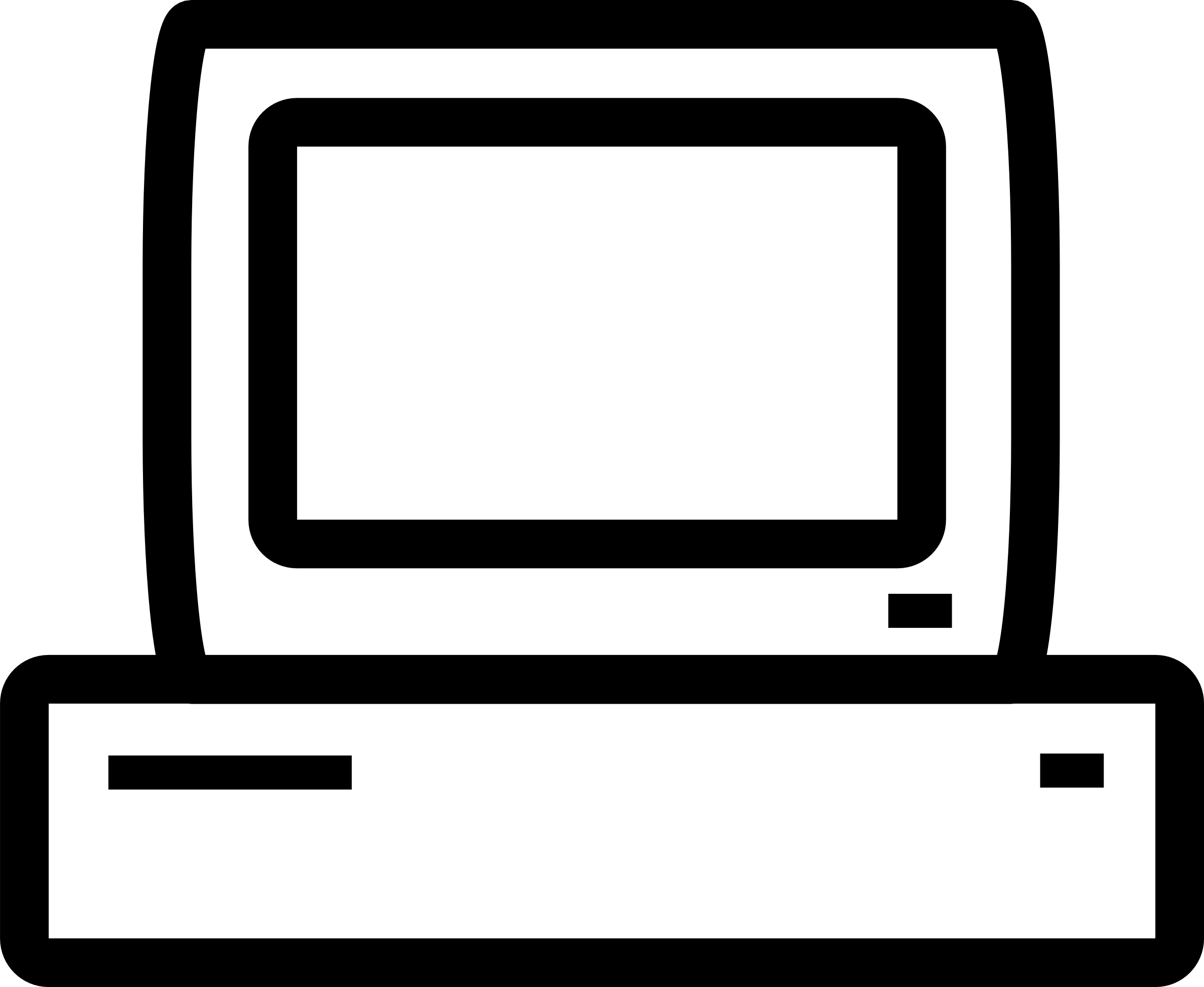
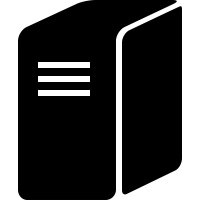
### Connexion IMAP

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.

Connexion OK

Imap.gmail.com, 993, 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.

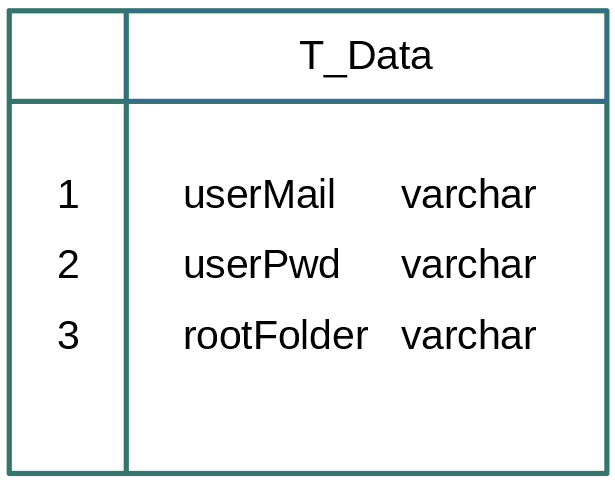
User OK, Auth OK

[user@gmail.com](mailto:user@gmail.com), \*\*\*\*\*\*\*\*\*

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

### Interactions base de données

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



Les trois champs permettent de stocker l’adresse mail de l’utilisateur, son mot de passe et le dossier racine.

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

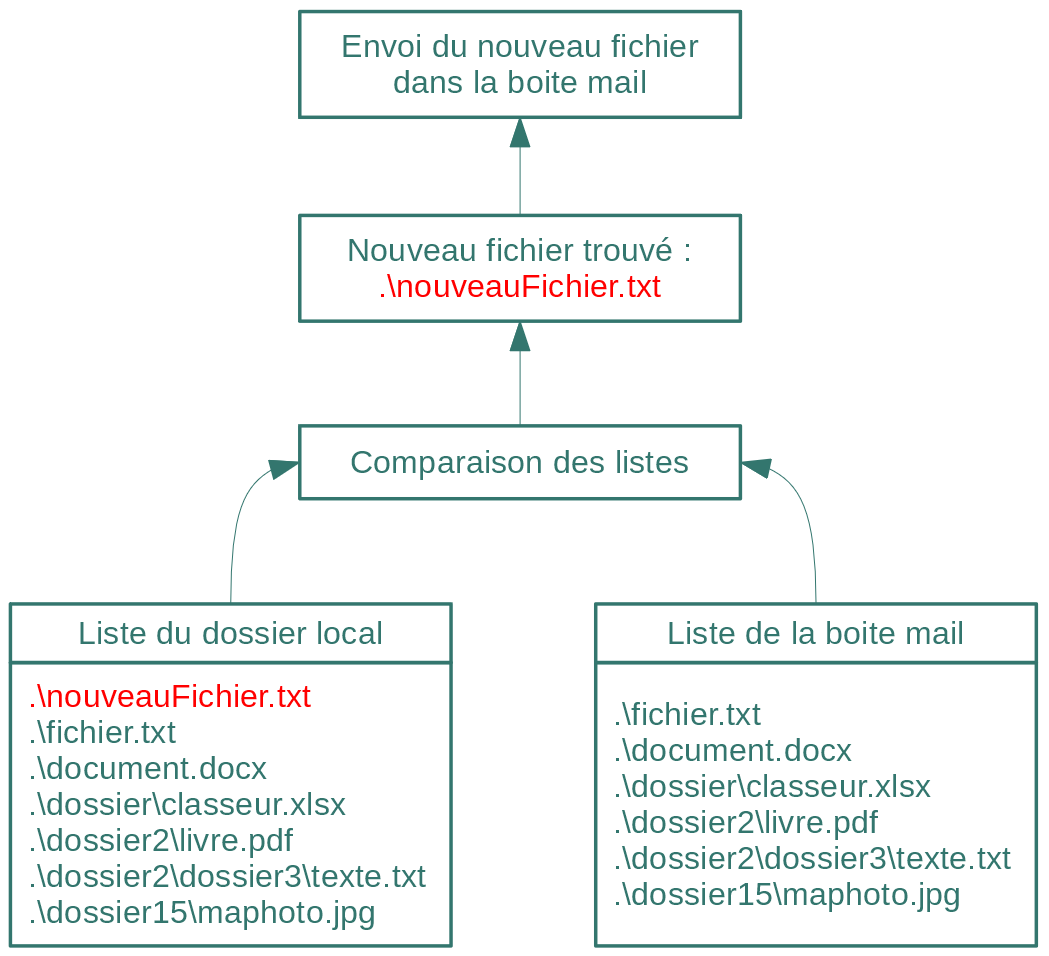
En effet, un simple fichier texte aurait suffi pour réaliser le travail demandé. Cependant, n’ayant jamais utilisé de base de données SQLite, ce projet est un bon moyen de découvrir le fonctionnement de cette librairie.

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.

### 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 boite 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 boite mail, les listes sont comparées et l’élément manquant est placé dans le dossier ou la boite 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 boite mail. Ces listes contiennent les chemins d’accès aux fichiers, ce qui permet de connaitre l’arborescence totale du dossier racine.

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 fichié sera indiqué comme nouveau et sera envoyé sur la boite mail afin que les deux listes s’égalisent et continuent à posséder les mêmes valeurs.

## Conception des tests

## Planification détaillée

### Semaine 1

### Semaine 2

### Semaine 3

### Semaine 4

### Semaine 5

# Réalisation

## Dossier de Réalisation

## Dossier des tests

# Conclusion

## Bilan des fonctionnalités demandées

## Bilan personnel

# Divers

## Journal de travail

## Webographie

## Cahier des charges

### 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.educanet2.ch](mailto:jonathan.melly@vd.educanet2.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% | |

### 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 débuter 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.

### TITRE

MailStorage

### 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).

### MATÉRIEL ET LOGICIEL À DISPOSITION

1 ordinateur ETML standard (Win7, VisualStudio Community)

### PRÉREQUIS

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

### 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.

### POINTS ÉVALUÉS DURANT LE PROJET

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

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)

### 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

### 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)

### VALIDATION

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

1. « Internet Message Access Protocol », <https://fr.wikipedia.org/wiki/Internet_Message_Access_Protocol>, 22.02.2017 [↑](#footnote-ref-1)
2. « Post Office Protocol », <https://fr.wikipedia.org/wiki/Post_Office_Protocol>, 02.05.2017 [↑](#footnote-ref-2)
3. « Browse for a directory in C# », <http://stackoverflow.com/questions/11767/browse-for-a-directory-in-c-sharp>, 08.08.2014 [↑](#footnote-ref-3)