



République Tunisienne
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Tunis El Manar
École Nationale d'Ingénieurs de Tunis



DEPARTEMENT GENIE ELECTRIQUE

Mini Projet Programmation Orientée-Objets C++

Application de gestion de réservation du vol

Réalisé par :

Kaddoudi Louy

Sabri Ben Naceur

Classe : 2AGE1

Encadré par :

Prénom NOM

Année universitaire 2020/2021

Table des matières

Table des figures	4
Liste des tableaux	5
1 Étude préliminaire du projet	7
1.1 Introduction	8
1.2 Présentation logicielle	8
1.2.1 Qt Creator	8
1.2.1.1 SQLite	8
1.2.1.2 DB Browser for SQLite	9
1.3 Étude préliminaire du projet	9
1.3.1 Énoncé du projet	9
1.3.2 Diagramme des cas d'utilisations	10
1.3.2.1 Étude de base de données	10
1.4 Conclusion	12
2 Réalisation de l'interface graphique du Notre Projet	13
2.1 Introduction	14
2.2 Description Générale de l'interface graphique	14
2.3 Fenêtre d'accueil	14
2.4 Vérification des données	15
2.5 Fenêtre "Admin"	17
2.5.1 Onglet "Login"	17
2.5.2 Onglet "Flight management"	19
2.6 Fenêtre "User"	19
2.7 Conclusion	19
3 Validation du projet	20
3.1 Introduction	21
3.2 Interface de "Login"	21
3.3 Interface de "Admin"	21
3.3.1 Onglet "welcome"	22
3.3.2 Onglet "Login info"	22
3.3.3 Onglet "Flight management"	22
3.3.4 Onglet "About"	22
3.4 Interface de "User"	22
3.4.1 Onglet "welcome"	22

<i>TABLE DES MATIÈRES</i>	3
---------------------------	---

3.4.2 Onglet "flight reservation and consultation"	22
3.4.3 Onglet "About"	22
3.5 Conclusion	22

Table des figures

1.1	Logo du logiciel Qt Creator	8
1.2	Logo du logiciel SQLite	9
1.3	Logo du logiciel DB Browser (SQLite)	9
1.4	Diagramme des cas d'utilisation)	10
1.5	Schéma relationnel de base de données	10
1.6	Tables de base de données	11
1.7	Tables de base de données (PowerShell)	12
2.1	Fonction principale	14
2.2	Objet désigne fenêtre d'accueil	15
2.3	Fichier .txt	15
2.4	Code vérification des données -1-	16
2.5	Code vérification des données -2-	17
2.6	Objets onglet "Login"	18
2.7	Code pour la fonction de lecteur	18
2.8	Code pour la fonction d'écriture	19
3.1	Fenêtre principale	21
3.2	"Admin"	21
3.3	"Welcome Admin"	21
3.4	Fenêtre dédiée à l'Admin	22
3.5	"Onglet "Login info" Read process	22
3.6	"Onglet "Login info" Wirte process	23

Liste des tableaux

Introduction générale

Le texte du rapport peut être fourni en Français ou en Anglais. Les rapports seront imprimés sur format A4 avec des marges de 2,5 cm en haut, en bas, à gauche et à droite. Le titre du rapport, le(s) auteur(s), leur(s) encadrant(s) couvrent toute la largeur de la page de garde. Un résumé avec mots-clés est demandé. La table des matières, la liste des figures et des tableaux doivent être présentées. Une liste des abréviations peut être ajoutée selon le besoin après la liste des tableaux.

Chapitre 1

Étude préliminaire du projet

1.1 Introduction

Étant donné qu'il s'agit d'un nouvel outil de programmation C++, il est primordial de commencer par une étude générale portant sur les logiciels à utiliser pendant la réalisation de notre mini projet ainsi qu'une étude préliminaire de ce dernier. Le projet vise à réaliser d'une application graphique permettant de gérer les réservations des vols au niveau d'une agence de voyage.

1.2 Présentation logicielle

1.2.1 Qt Creator

Qt Creator est un environnement de développement intégré multiplateforme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++. Il intègre directement dans l'interface un débogueur, un outil de création d'interfaces graphiques, des outils pour la publication de code sur Git et Mercurial ainsi que la documentation Qt. L'éditeur de texte intégré permet l'autocomplétion ainsi que la coloration syntaxique. Qt Creator utilise sous Linux le compilateur gcc. Il peut utiliser MinGW ou le compilateur de Visual Studio sous Windows. Qt Creator a été traduit en français par l'équipe Qt de Developpez.com.



FIGURE 1.1 – Logo du logiciel Qt Creator

1.2.1.1 SQLite

SQLite est un système de base de données ou une bibliothèque proposant un moteur de base de données relationnelles. Il repose sur une écriture en C, un langage de programmation impératif, et sur une accessibilité via le langage SQL (Structured Query Language). SQLite présente la particularité d'être directement intégré aux programmes et dans l'application utilisant sa bibliothèque logicielle. Avec SQLite, la base de données est intégralement stockée dans un fichier indépendant du logiciel. Créé au début des années 2000 par D. Richard Hipp, SQLite propose un accès plus rapide aux données, mais aussi plus structuré et avec davantage de sécurité.



FIGURE 1.2 – Logo du logiciel SQLite

1.2.1.2 DB Browser for SQLite

DB Browser for SQLite est un outil open source graphique de haute qualité pour créer, concevoir et modifier des fichiers de base de données compatibles avec SQLite. L'interface utilisée est de type tableur familière.

- Les tables créées à l'aide de DB Browser à l'aide du système de base de données

SQLite, en plus de l'interface graphique offerte par Qt Creator, nous permettront de créer une application accessible à toute catégorie d'utilisateurs vu sa facilité d'utilisation. En plus, cette combinaison garantit une meilleure optimisation pour l'administrateur qui sera capable de contrôler et de manipuler facilement les places dans un vol.



FIGURE 1.3 – Logo du logiciel DB Browser (SQLite)

1.3 Étude préliminaire du projet

1.3.1 Énoncé du projet

L'objectif est de développer une application graphique permettant de gérer les réservations des vols au niveau d'une agence de voyage. L'utilisateur de cette application peut :

- Ajouter un vol.
- Rechercher et consulter des vols existant par date départ, destination etc.
- Ajouter un voyageur et lui réserver un vol
- Il peut aussi modifier ou annuler un vol.

1.3.2 Diagramme des cas d'utilisations

Le diagramme des cas d'utilisations nous permettons de mieux comprendre actions réalisées par le système en interaction avec les acteurs en vue d'une finalité. L'ensemble des cas d'utilisation permet ainsi de décrire les exigences fonctionnelles d'un système en adoptant le point de vue et le langage de l'utilisateur final.

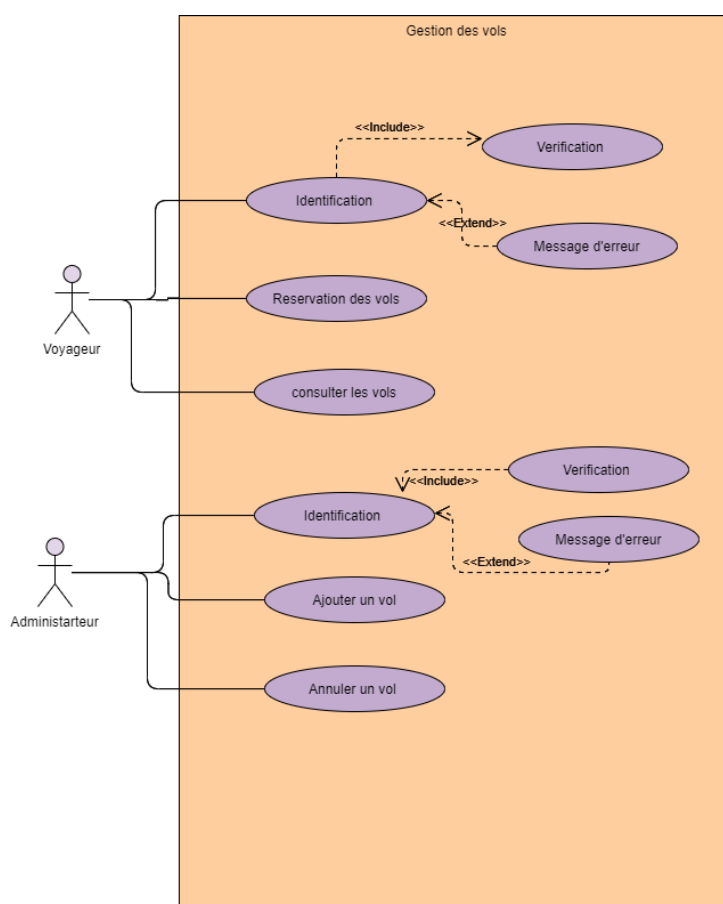


FIGURE 1.4 – Diagramme des cas d'utilisation)

1.3.2.1 Étude de base de données

Pour faire une bonne conception de base de données nous commençons par la réalisation de schéma relationnel suivant :

FIGURE 1.5 – Schéma relationnel de base de données

Une fois le modèle choisi, il faut implémenter la base de données. Pour la création de notre base de données, nous avons utilisé le logiciel DB Browser (SQLite) et avons suivi les étapes suivantes :

1. création d'une nouvelle base de données Il est assez simple de créer une nouvelle base de données dans DB Browser (SQLite).

Notre base de données est créée sous le nom "GestionVol". Quand cette étape est terminée, la création des tables peut commencer.

2. création des tables Nous avons créé les tables suivantes :

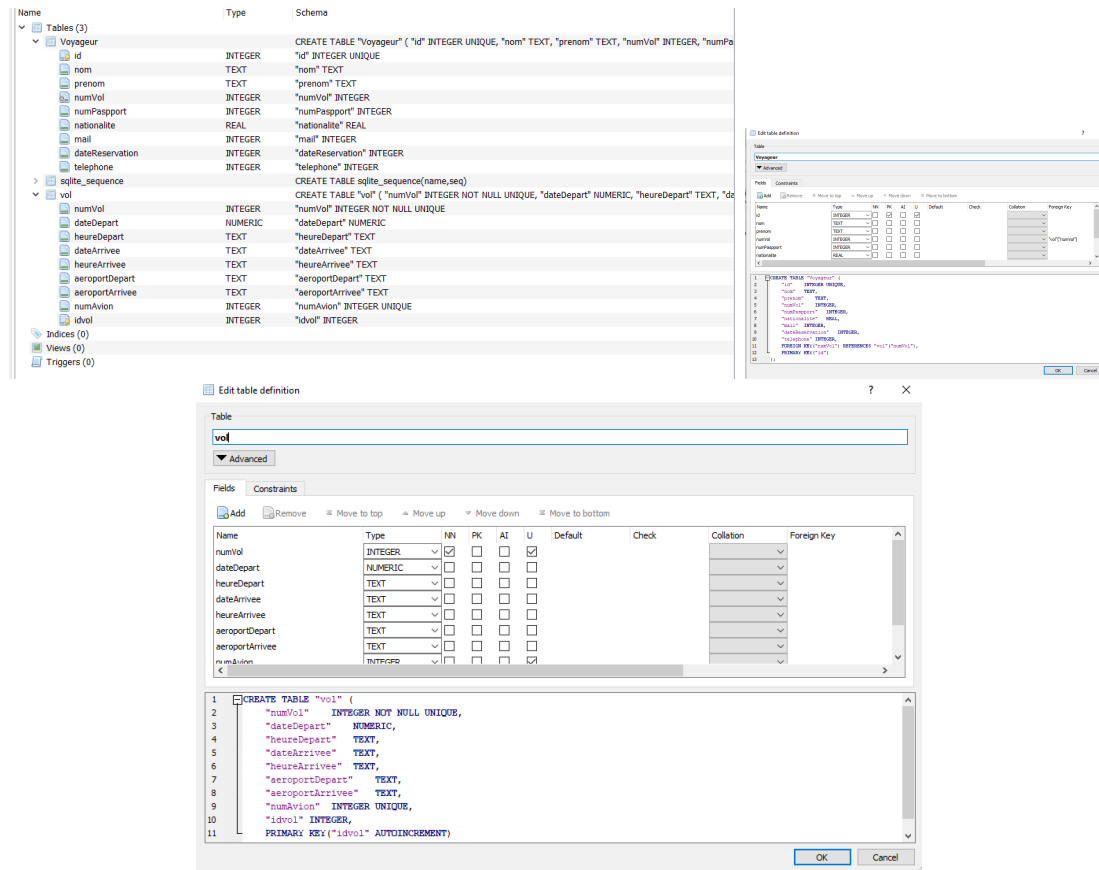


FIGURE 1.6 – Tables de base de données

```

sqlite> .schema
CREATE TABLE IF NOT EXISTS "Voyageur" (
  "id"      INTEGER UNIQUE,
  "nom"     TEXT,
  "prenom"  TEXT,
  "numVol"   INTEGER,
  "numPaspport" INTEGER,
  "nationalite" REAL,
  "mail"    INTEGER,
  "dateReservation" INTEGER,
  "telephone" INTEGER,
  PRIMARY KEY("id"),
  FOREIGN KEY("numVol") REFERENCES "vol"("numVol")
);
CREATE TABLE IF NOT EXISTS "vol" (
  "numVol"      INTEGER NOT NULL UNIQUE,
  "dateDepart"  NUMERIC,
  "heureDepart" TEXT,
  "dateArrivee" TEXT,
  "heureArrivee" TEXT,
  "aeroportDepart" TEXT,
  "aeroportArrivee" TEXT,
  "numAvion"    INTEGER UNIQUE,
  "idvol"       INTEGER,
  PRIMARY KEY("idvol" AUTOINCREMENT)
);

```

FIGURE 1.7 – Tables de base de données (PowerShell)

1.4 Conclusion

Ce chapitre nous a permis de se placer dans le cadre du projet, et ce, en connaissant les logiciels de travail ainsi que les tables et les interfaces graphiques à développer. Dans le chapitre suivant nous allons détailler la solution qu'on a proposé.

Chapitre 2

Réalisation de l'interface graphique du Notre Projet

2.1 Introduction

2.2 Description Générale de l'interface graphique

Notre application peut être divisée en deux fenêtres :

- Fenêtre accueil.
- Fenêtre secondaire.

La première fenêtre se charge de l'étape de validation des coordonnées. La deuxième fenêtre est divisée en plusieurs onglets et porte des différences liées aux natures de l'utilisateur.

A noté que notre application distingue deux types d'utilisateur :

- Voyageur.
- Administrateur.

2.3 Fenêtre d'accueil

Une fenêtre d'accueil dans la quelle l'utilisateur est demandé de remplir deux champs :

- Le nom d'utilisateur.
- Le mot de passe.

L'utilisateur de l'application quelque soit un voyageur ou un administrateur doit remplir ses coordonnées afin de valider son accès. après l'étape de saisie l'utilisateur spécifie la nature de son compte : compte "user" ou compte "Admin" puis il valide ses coordonnées par appuis sur un bouton de validation.

Pour notre fonction principale "main" elle est le suivante

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

FIGURE 2.1 – Fonction principale

La fonction main construit un objet du type "MainWindow" qui correspond à notre fenêtre principale. À l'aide de la fonction "show()" cette dernière est affichée sur l'écran.

QT creator nous offre un outil de design des interfaces graphiques que nous avons utilisés pour concevoir une Fenêtre simple et facile à comprendre. les éléments mis dans la fenêtre d'accueil sont les suivants :

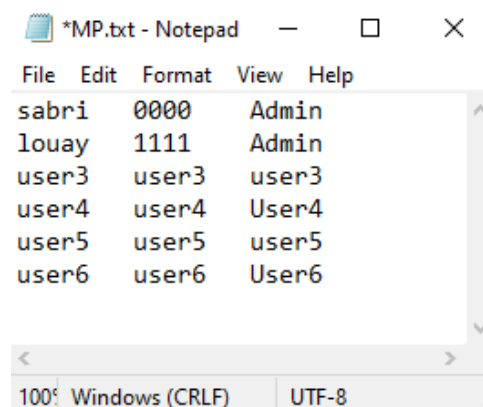
Object	Class
MainWindow	QMainWindow
centralWidget	QWidget
groupBox	QGroupBox
verticalLayout	QVBoxLayout
horizontalLayout_4	QHBoxLayout
label_4	QLabel
lineEdit_user	QLineEdit
horizontalLayout_3	QHBoxLayout
label_3	QLabel
lineEdit_password	QLineEdit
horizontalLayout	QHBoxLayout
radioButtonAdmin	QRadioButton
radioButtonUser	QRadioButton
pushButtonLogIn	QPushButton
lablepic	QLabel
menuBar	QMenuBar
mainToolBar	QToolBar
statusBar	QStatusBar

FIGURE 2.2 – Objet désigne fenêtre d'accueil

Les éléments Utilisés sont des objets labels pour marquer des informations aux utilisateurs, des objets "Line Edit" pour les champs à remplir et des objets du type "radio Button" pour la sélection de le type du compte. Nous avons utilisé aussi des objets du type "Vertical Layout" et "Horizontal Layout" pour bien positionner nos éléments.

2.4 Vérification des données

L'étape de saisie des données est suivie d'une étape de validation. c'est une étape qui se fait en arrière plan. Les données saisie sont comparés par d'autre mise dans un fichier texte "MP.txt".



File	Edit	Format	View	Help
sabri	0000	Admin		
louay	1111	Admin		
user3	user3	user3		
user4	user4	User4		
user5	user5	user5		
user6	user6	User6		

100% Windows (CRLF) UTF-8

FIGURE 2.3 – Fichier .txt

Le fichier contienne l'identificateur de l'utilisateur, sont mot de passe et son type de compte séparés par une tabulation ou des espaces. La vérification des données se fait donc par recherche des données saisie dans ce fichier.

la programmation événementielle est fondé sur les événements. Elle s'oppose à la programmation séquentielle. Le programme sera principalement défini par ses réactions aux différents événements qui peuvent se produire, c'est-à-dire des changements d'état de variable. Dans notre cas l'appui sur le bouton de validation déclenche le processus de vérification des données.

La première chose à faire est de récupérer les deux champs saisie dans deux variables. Puis on ouvre le fichier contenant les informations et On vérifie l'existence de ce dernier. L'utilisateur sera informé par un message du type "warning" si l'opération d'ouverture est mal démarré.

l'étape suivante est la décomposition du fichier en plusieurs lignes chaque ligne est décomposée à son tour en plusieurs champs qui seront mis dans une liste de la classe user conçu pour contenir les données d'utilisateurs. Cette décomposition se fait à l'aide d'une expression régulière qui détecte les tabulations ou les espaces. À noter que ces deux caractères sont utilisés pour séparer les champs dans notre fichier MP.txt contenant les données des utilisateurs.

```
void MainWindow::on_pushButtonLogIn_clicked()
{
    QString username = ui->lineEdit_user->text();
    QString password = ui->lineEdit_password->text();

    QFile file("./MP.txt");
    if(!file.open(QFile::ReadOnly | QFile::Text))
    {
        QMessageBox::warning(this, "title", "file not open");
    }
    else
    {
        QList<User> liste_user;
        QTextStream in(&file);

        while(!in.atEnd())
        {
            QString line = in.readLine();
            QRegExp rx("(\\s|\\t)");
            QStringList list = line.split(rx);
            User u;
            u.set_username(list.at(0));
            u.set_password(list.at(1));
            u.set_role(list.at(2));
            liste_user.push_back(u);
        }
    }
}
```

FIGURE 2.4 – Code vérification des données -1-

L'utilitariste peut être un voyageur ou un administrateur. La fonction "isChecked" vérifie la sélection de type du compte. Puis en utilise une boucle "while" pour parcourir la liste conçue et on vérifie l'existence des données saisies dans notre fichier.


```

if(ui->radioButtonAdmin->isChecked())
{
    QList<User>::Iterator it = liste_user.begin();
    while ( it != liste_user.end())
    {
        if(username == it->get_username() && password == it->get_password() && it->get_role()=="Admin"
        {
            QMessageBox :: information(this, "Login Info", "Pass word Admin is correct");
            hide();
            Dia = new Dialog(this);
            Dia->show();
            flagAdm=1;
            break;
        }
        it++;
    }
    if(!flagAdm)
    {
        QMessageBox :: information(this, "Login Info", "Pass word Admin is incorrect");
    }
}

if(ui->radioButtonUser->isChecked())
{
    QList<User>::Iterator it = liste_user.begin();
    while ( it != liste_user.end())
    {
        if(username == it->get_username() && password == it->get_password() && it->get_role()=="User")
        {
            QMessageBox :: information(this, "Login Info", "Pass word user is correct");
            flagUser=1;
            break;
        }
        it++;
    }
    if(!flagUser)
    {
        QMessageBox :: information(this, "Login Info", "Pass word Admin is incorrect");
    }
}

```

FIGURE 2.5 – Code vérification des données -2-

2.5 Fenêtre "Admin"

La fenêtre "Admin" est constituée de plusieurs onglets. chacun d'eux porte des fonctionnalités aux utilisateurs du type administrateur. Ces onglets sont :

- Onglet "Welcome" : l'onglet principale pour l'administrateur.
- Onglet "Login" : Permet la gestion du fichier contenant informations utilisateur.
- Onglet "Flight management" Permet la gestion des vols.
- Onglet "About" **add something here!!!!!!**
- **if we add!!!!!!**

2.5.1 Onglet "Login"

L'onglet "Login" permet la gestion du fichier contenant les informations utilisateurs. A travers cet onglet l'administrateur peut ajouter d'autre voyageur ou d'autre administrateurs.

Pour le design de cette onglet nous avons choisie les objets suivants :

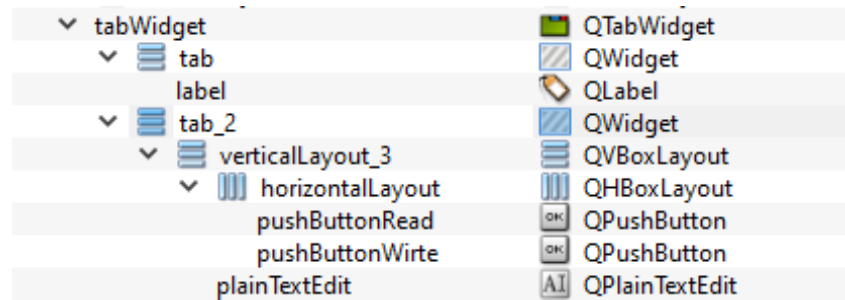


FIGURE 2.6 – Objets onglet "Login"

Nous avons utilisé deux bouton "pushBttonRead" et "pushBottonWrite" pour permettre à l'administrateur soit ouvrir le fichier en mode lecteur ou l'ouvrir en mode écriture. L'administrateur saisit ses informations dans un champ "plainTextEdit".

L'ouverture du fichier en mode lecteur se déclenche suite à un appui sur le bouton "Read". Le code exécute suite à cette action est le suivant :

```
void Dialog::on_pushButtonRead_clicked()
{
    QString filter = "Text File (*.txt)";
    QString file_name = QFileDialog::getOpenFileName(this, "open a file", "./", filter);
    QMessageBox::information(this, "..", file_name);
    QFile file(defaultFile);
    if(!file.open(QFile::ReadOnly | QFile::Text))
    {
        QMessageBox::warning(this, "title", "file not open");
    }
    QTextStream in(&file);
    QString text = in.readAll();
    ui->plainTextEdit->setPlainText(text);
    file.close();
}
```

FIGURE 2.7 – Code pour la fonction de lecteur

Avec La fonction "gestOpenFileName" qui permet d'initialiser le chemin à ouvrir ainsi que d'appliquer un filtre sur les types de fichier (dans notre cas en applique un filtre sur les fichiers de type ".txt"). On vérifie ensuite que le processus d'ouverture du fichier en mode lecteur est bien terminé. A noter qu'en cas d'erreur on affiche un message de type "warning". Le contenu du fichier est affiché dans le champ "plainTextEdit".

L'ouverture du fichier en mode écriture se déclenche suite à un appui sur le bouton "Wirte". Le code exécute suite à cette action est le suivant :

```
void Dialog::on_pushButtonWirte_clicked()
{
    QFile file(defaultFile);
    if(!file.open(QFile::WriteOnly | QFile::Text))
    {
        QMessageBox::warning(this,"title","file not open");
    }
    QTextStream out(&file);
    QString text = ui->plainTextEdit->toPlainText();
    out << text;
    file.flush();
    file.close();
}
```

FIGURE 2.8 – Code pour la fonction d'écriture

Le fichier à ouvrir par défaut est "MP.txt". On vérifie ensuite que le processus d'ouverture du fichier en mode écriture est bien terminé. A noter qu'en cas d'erreur on affiche un message de type "warning". Le texte saisi dans le champ "plainTextEdit" est transféré au fichier suite à l'appui sur le bouton "wirte".

2.5.2 Onglet "Flight management"

2.6 Fenêtre "User"

2.7 Conclusion

Chapitre 3

Validation du projet

3.1 Introduction

3.2 Interface de "Login"

En lançant l'application, la fenêtre principale "Login" sera affichée sur votre écran. Cette interface, contient deux champs à saisir : "User name" et "Pass word". La section du pute de compte se fait avec les deux "radio boutons" : "Admin", "User". La validation des données est assurées par appuis sur le bouton "LogIn".

FIGURE 3.1 – Fenêtre principale

3.3 Interface de "Admin"

FIGURE 3.2 – "Admin"

Pour accéder a l'interface "Admin" il suffit de taper le User ID, le mot de passe et sélectionner le bouton "Admin" puis "Login"

FIGURE 3.3 – "Welcome Admin"

Lors du lancement de l'application, si on choisit dans la fenêtre d'accueil, le bouton radio "Admin " la fenêtre si dessous sera affiché :

L'onglet "Login info" est la partie qui nous permet de modifier nos données de connexion, si on choisit le bouton "Read" sa nous mène à ouvrir le fichier ou on a sauvegardé nos données :

si on choisit le bouton "Wirte" sa nous mène enregistrer nos données dans le fichier où on a sauvegardé nos données :

FIGURE 3.4 – Fenêtre dédiée à l'Admin

FIGURE 3.5 – "Onglet "Login info" Read process

3.3.1 Onglet "welcome"**3.3.2 Onglet "Login info"****3.3.3 Onglet "Flight management"****3.3.4 Onglet "About"****3.4 Interface de "User"****3.4.1 Onglet "welcome"****3.4.2 Onglet "flight reservation and consultation"****3.4.3 Onglet "About"****3.5 Conclusion**

FIGURE 3.6 – "Onglet "Login info" Wirte process