REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE.
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.
Université Mouloud Mammeri de Tizi-Ouzou.
Faculté de Génie Electrique et Informatique.
Département Informatique.

Mémoire de Fin d'Etudes

En vue de l'obtention du diplôme de Master en Informatique

Option : Conduite de Projet Informatique « CPI »

Thème

Conception et réalisation d'une application mobile sous « Android » pour la réservation de vols.

Dirigé par :
Mme: CHAMEK L.

Réalisé par :
Melle: KHELFANE Kamilia.
Melle: LAGAB Chahinas.

2014/2015

Remerciements

Nous tenons à exprimer nos vifs remerciements, à tous ceux qui ont contribué de près ou de loin, à la réalisation de ce projet :

- A Mme CHAMEK L. , pour avoir accepté de diriger ce travail, pour tout le temps qu'elle a consacré à son élaboration et pour ses précieux conseils ;
- Aux membres du jury, pour l'honneur qu'ils nous font d'accepter d'examiner notre travail.

Nos remerciements vont également à tous les enseignants qui ont contribué à notre formation.

Dédicaces

A mes très chers parents en témoignage de reconnaissance et d'affection,

A mes deux frères Kamal et Salem,

A mes deux cousines Lynda M. et Lynda S.,

A tous les membres de ma famille grands et petits,

A tous mes amis(es),

Je dédie ce travail.

Chahinas.

Dédicaces

A mes très chers parents en témoignage de reconnaissance et d'affection,

A tous les membres de ma famille grands et petits,

A tous mes amis(es),

Je dédie ce travail.

Kamilia.

Table des matières

Introduction générale 1

Chapitre 1 : Généralités sur Android

Introduction 2
I. Présentation générale d'Android 2
I.1 Définition 2
I.2 Plateforme Android 2
I.3 Historique d'Android 3
I.4 Versions d'Android 4
I.4.1 Chronologie des versions 5
II. Caractéristiques d'Android 6
II.1 Architecture d'Android 6
II.1.1 Les applications 7
II.1.2 Le Framework applicatif 8
II.1.3 Les bibliothèques 8
II.1.4 Android Runtime 9
II.1.5 Le noyau Linux 10
II.2 SDK Android 10
II.2.1 Les API(s) 11
II.2.2 Documentation du SDK 11
II.2.3 Les exemples 11
II.2.4 Les outils de développement du SDK 12
1. **SQLite** 12
2. **Émulateur** 12
3. **ADB** 13

4. DDMS 14
III. Concepts d'Android 15
1. Bureau virtuel 15
2. Widgets 16
3. Android Market 17
Conclusion 18

Chapitre 2 : Analyse et Conception

Introduction 19
I. Analyse 19
I.1 Présentation du langage UML 19
I.1.1 Définition 19
I.2 Objectif de l'application 20
I.3 Les acteurs du système 20
1. Identification des acteurs 20
2. Extraction des acteurs 20
I.4 Cas d'utilisations 21
I.4.1 Définition d'un cas d'utilisation 21
I.4.2. Relations entre cas d'utilisations 21
I.4.3 Diagramme de cas d'utilisation général 21
I.4.4 Quelques diagrammes de cas d'utilisations 23
II. Conception 27
II.1 Diagrammes de séquence 27
II.2 Diagrammes de classes 32
II.3 Diagramme de classes général 35
III. Conception de la base de données 36
III.1 Les différentes tables avec la codification utilisée 36
Conclusion 40

Chapitre 3 : Réalisation

Introduction **41**

I. Environnement de travail **41**

I.1 Environnement matériel **41**

I.2 Environnement logiciel **41**

II. Technologies utilisées **41**

1. **IDE Eclipse** **41**
2. **Le SDK Android** **42**
3. **Le plugin ADT** **42**
4. **SQLite** **43**

III. Langages de programmation utilisés **44**

1. **Langage JAVA** **44**
2. **Langage XML** **44**
3. **Langage SQL** **45**

IV. Interfaces Homme/Machine **45**

IV.1 Interface d'accueil de l'application **45**

IV.2 Interface « Rechercher » **46**

IV.2.1 Interface « Offres Promotionnelles » **47**

IV.3 Interface « Créer Compte » **51**

IV.4 Interface « S'authentifier » **52**

Conclusion **58**

Conclusion Générale **59**

Bibliographie

Liste de figures

Figure 1.1 : Évolution des versions d'Android 5
Figure 1.2 : Architecture d'Android 7
Figure 1.3 : Émulateur Android 13
Figure 1.4 : Composants du DDMS 15
Figure 1.5 : Bureau virtuel Android 16
Figure 1.6 : Site web d'Android Market 17

Figure 2.1 : Diagramme de cas d'utilisation général 22
Figure 2.2 : Diagramme de cas d'utilisation « Effectuer une recherche rapide ». 23
Figure 2.3: Diagramme de cas d'utilisation « Créer compte » 24
Figure 2.4: Diagramme de cas d'utilisation « Effectuer une réservation » 25
Figure 2.5 : Diagramme de cas d'utilisation « Annuler une réservation » 26
Figure 2.6 : Diagramme de séquence pour le cas d'utilisation « Authentification du client » 28
Figure 2.7 : Diagramme de séquence pour le cas d'utilisation « Rechercher vol, acteur : utilisateur simple » 29
Figure 2.8 : Diagramme de séquence pour le cas d'utilisation « Création nouveau compte utilisateur, acteur : utilisateur simple » 30
Figure 2.9 : Diagramme de séquence pour le cas d'utilisation « Annulation d'une réservation, acteur : client » 31
Figure 2.10 : Diagramme de classes pour le cas d'utilisation « Authentification d'un client » 33
Figure 2.11 : Diagramme de classes pour le cas d'utilisation « Annulation d'une réservation, acteur : client » 33
Figure 2.12 : Diagramme de classes pour le cas d'utilisation « Création d'un compte, acteur : utilisateur simple » 34
Figure 2.13 : Diagramme de classes pour le cas d'utilisation « Rechercher vol, acteur : client »... 34
Figure 2.14 : Diagramme de classes global 35

Figure 3.1 : Capture d'écran d'une fenêtre Eclipse 42
Figure 3.2 : Logo SQLite 44
Figure 3.3 : Interface d'accueil de l'application 46
Figure 3.4 : Interface de Recherche 47
Figure 3.5 : Liste offres 48
Figure 3.6 : Formulaire de recherche de vols pour une offre 48
Figure 3.7 : Liste vols offres 49
Figure 3.8 : Vérification des détails du voyage 49
Figure 3.9 : Détails vol 50
Figure 3.10 : Formulaire de réservation 51
Figure 3.11 : Liste réservations 51
Figure 3.12 : Formulaire de création de compte 52
Figure 3.13 : Espace client 52
Figure 3.14 : Formulaire d'authentification 53
Figure 3.15 : Echec de connexion 53
Figure 3.16 : Formulaire de recherche 54
Figure 3.17 : Liste réservations 55
Figure 3.18 : Demande de confirmation d'annulation de réservation 55
Figure 3.19 : Liste personnes 56
Figure 3.20 : Demande de confirmation d'annulation de vol pour une personne choisie 56
Figure 3.21 : Liste vols (échanger) 57
Figure 3.22 : Demande de confirmation de changement de réservation. 57

Introduction Générale

Il ne fait désormais plus aucun doute que les réseaux informatiques représentent la révolution la plus importante et la plus innovante qui a marqué la vie de l'humanité en ce siècle passé. En effet, depuis la création du réseau Internet et son développement, notre manière de vivre a radicalement changé, à l'aide d'un ordinateur connecté à ce réseau, le monde entier nous est tout simplement accessible en l'espace de quelques clics.

De plus, l'avènement de la nouvelle technologie mobile nous permet de réaliser ce que hier n'était que pure chimère : communiquer, travailler, s'instruire, se divertir, prendre connaissance de l'actualité n'importe où et à n'importe quel moment.

Étant une combinaison de la technologie informatique et de la technologie mobile, la nouvelle génération de téléphones portables appelés Smartphones, a révolutionné le marché du mobile et a séduit un public de plus en plus large qui se voit tout faire avec, d'où la nécessité d'innover en ce sens.

Par son ouverture et ses possibilités de déploiement, la plateforme Google Android basée sur Linux offre un socle et un environnement de développement puissant pour créer des applications mobiles robustes et ergonomiques. Elle met à la portée des professionnels et des particuliers la réalisation d'applications à la fois riches en fonctionnalités et adaptées aux contraintes de l'utilisation mobile.

Une compagnie aérienne, comme toute autre entreprise, peut faire appel à cette nouvelle technologie à savoir : les applications mobiles pour créer de nouveaux services, élargir leurs supports de communication, augmenter le nombre de clients fidèles ou encore pour mieux se positionner sur le marché et être plus compétitive.

En effet, même avec l'existence de plusieurs agences de voyage à l'échelle nationale et internationale qui effectuent la réservation, le problème de déplacement reste inévitable, car les différentes agences commerciales, en général, se situent dans les grandes villes ce qui n'arrange pas tout le monde, notamment ceux qui habitent les coins isolés du pays.

Dans ce cadre intervient notre projet de fin d'étude dont l'objectif est de développer une application mobile sous Android permettant aux utilisateurs de consulter et/ou de réserver des vols proposés par une compagnie aérienne.

Notre travail est scindé en plusieurs chapitres :

Le premier chapitre présente quelques généralités sur la plateforme Android, ses différentes caractéristiques et le kit de développement logiciel Android « SDK ».

L'analyse et la conception de l'application fait l'objet du deuxième chapitre où le langage de modélisation UML est utilisé pour présenter quelques diagrammes de cas d'utilisation, de séquence et de classes ;

Le troisième et dernier chapitre est consacré à la présentation de l'environnement de travail ainsi qu'à l'illustration de quelques interfaces de notre application.

CHAPITRE 1

« Généralités sur Android »

Introduction

Le marché de la téléphonie mobile connaît actuellement une véritable révolution. Google, ayant réalisé le potentiel de ce marché, a décidé de s'y introduire en rachetant une startup travaillant sur un système d'exploitation ouvert pour terminal mobile : Android.

Ces dernières années, l'utilisation des applications « intelligentes » sur les Smartphones et tablettes devient de plus en plus fréquente et cela grâce aux différents systèmes d'exploitation mobiles tel que « Android », « IOS1» ou bien « *Windows Phone* » qui ne cessent de se développer.

Ce présent chapitre sera donc consacré à la présentation générale de la plateforme Android, à ses différentes caractéristiques et enfin à la présentation du kit de développement logiciel Android « SDK ».

I. Présentation générale d'Android

Afin de s'immerger pleinement dans l'univers d'Android, une présentation de la plateforme s'impose. Nous allons donc découvrir les origines et l'historique de la plateforme Android, ses différentes versions et enfin l'architecture de cette plateforme sur laquelle s'exécutent les différentes applications.

I.1 Définition [1]

Android est un système d'exploitation pour téléphones mobiles et tablettes tactiles, développé par une petite startup qui fut rachetée en 2007 par la société mondialement connue : Google qui poursuit activement son développement avec *l'Open Handset Alliance* « OHA », celle-ci comprend plus de 35 constructeurs, fournisseurs de logiciel, et opérateurs.

Android concurrence des plateformes telles que l'*IphoneOS* d'*Apple*, *Windows Mobile* de *Microsoft*, *RIM OS* intégré dans les *BlackBerry* de *Research In Motion, WebOS d'HPBada* de Samsung, ou encore *Symbian* et *MeeGo* de Nokia.

I.2 Plateforme Android [2]

Selon Google qui est un majeur distributeur, Android est une plateforme puissante, moderne, sécurisée, ouverte, basée sur le noyau Linux et utilisant la plateforme Java pour ses applications. Android est entièrement gratuite et sa plateforme est très flexible ce qui permet aux développeurs d'intégrer, d'agrandir et de replacer les composants existants et d'adapter les applications aux besoins du client ou les remplacer entièrement, l'utilisateur peut donc personnaliser facilement son appareil.

De plus, il n'ya pas de distinction entre les applications natives et les applications qui sont développées par les développeurs, toutes sont disponibles sur l'*Android Market*, renommé *Google Play Store* en mars 2012, et qui permet le téléchargement d'applications gratuites ou payantes. Il est aussi possible de les noter et de les commenter.

La progression du nombre d'applications sur *Google Play Store* est exponentielle. Cette progression s'explique par le développement totalement ouvert d'Android, et les applications peuvent d'ailleurs être distribuées autrement que par ce biais. Pour simplifier le développement d'applications, Google a développé une interface web : *App Inventor* permettant de développer facilement une application qui pourra ensuite être mise à disposition sur le marché.

En termes d'applications, Android a intégré plusieurs services de Google pour accéder rapidement aux services d'internet comme *Gmail, YouTube, Google Talk, Google Calendar* et *Google Maps.*

La plateforme Android propose notamment : **[7]**

- Un *Framework* permettant la réutilisation et le remplacement de composants ;
- Une machine virtuelle optimisée pour les appareils mobiles ;
- Un navigateur intégré basé sur le moteur open source *WebKit* ;
- Un moteur graphique optimisé, propulsé par une librairie 2D dédiée et un moteur 3D basé sur les spécifications OpenGL ES 1.0 ;
- Le système de gestion de base de données SQLite pour le stockage de données ;
- Un support média pour les principaux formats audio, vidéo et images ;
- La téléphonie GSM, les communications *Bluetooth*, EDGE, 3G, et *WiFi* ;
- Un accès à la caméra, au GPS, à la boussole et aux accéléromètres ;
- Un environnement de développement riche : émulateur, outils de débogage, etc.

I.3 Historique d'Android [1]

Rachetée par Google en 2005, Android était initialement une startup (jeune entreprise) qui développa un système d'exploitation pour appareil mobile.

Dès lors, nombre de rumeurs annonçaient la sortie d'un téléphone Google nommé *Gphone*. Mais Google préparait, en fait, bien plus que cela.

Le 5 novembre 2007 fut annoncée la création de l'OHA « *Open Handset Alliance* » : un consortium créé à l'initiative de Google réunissant une trentaine d'entreprises à ses débuts (en

Mi-2011, l'OHA regroupe plus de quatre-vingts sociétés membres). La plus grande partie de ces entreprises étaient des opérateurs mobiles, des constructeurs, des industriels et des éditeurs logiciels. Le rôle de l'OHA est de favoriser l'innovation sur les appareils mobiles en fournissant une plateforme véritablement ouverte et complète.

Le même jour, l'OHA annonça officiellement Android ; la première plateforme complète et ouverte pour les appareils mobiles. Cette plateforme inclut un système d'exploitation le middleware (les logiciels intermédiaires), une interface utilisateur et des applications phares.

Quelques jours plus tard, en novembre 2007, l'OHA annonça la sortie du premier SDK « *Software Develepment Kit* » ou kit de développement logiciel Android permettant aux développeurs de pouvoir créer des applications pour la plateforme Android.

Octobre 2008, Google et l'OHA annoncèrent la mise à disposition du code source de la plateforme Android en open source, sous licence Apache **2.0**

Il est dès lors possible de télécharger le code source du système Android, le compiler, l'installer et l'exécuter. Il est également possible de contribuer à son évolution.

En novembre 2008 fut lancé l'*Android Market*, le magasin d'applications de Google permettant aux développeurs et éditeurs de logiciels Android de proposer leurs applications aux utilisateurs Android.

Le premier Smartphone Android est sorti en octobre 2008 aux États-Unis et en mars 2009 en France. Ce fut le HTC *Dream G1*.

Depuis, et jusqu'à mi-2011, est sorti sur le marché une centaine de modèles de Smartphones Android.

Mi-2009 apparurent les premières tablettes tactiles Android. Mais il faudra attendre début 2011 et la sortie de la version d'Android **3.0** spécialement conçue pour les tablettes pour démocratiser auprès du grand public.

Depuis l'apparition de la plateforme Android, chaque semaine apporte sont lot de nouveaux Smartphones et tablettes.

I.4 Versions d'Android

À l'instar de l'empire Romain, Android ne s'est pas fait en un jour. Pourtant, la progression du nombre de ses fonctionnalités est à l'image de celle de ses parts de marchés, c'est-à-dire tout simplement stupéfiante.

Ces fonctionnalités, améliorations et corrections de bogues ont fait leurs apparitions au fil du temps dans plusieurs versions.

Figure 1.1 : Évolution des versions d'Android.

I.4.1 Chronologie des versions [2] [7]

L'historique des versions d'Android a débuté avec la sortie de la version **1.0** en septembre 2008. Android a connu plusieurs mises à jour depuis sa première version. Ces mises à jour servent généralement à corriger des bugs et à ajouter de nouvelles fonctionnalités. Dans l'ensemble, chaque version est développée sous un nom de code basé sur des desserts. Ces noms de codes suivent une logique alphabétique.

- ✓ La version **1.5** « **Cupcake** » a été publiée en avril 2009, améliorant l'interface graphique d'Android et rendant le système plus utilisable. En effet, c'est à partir de cette version **1.5** qu'est apparu le premier clavier tactile et le presse-papier. Avant cela, il était impossible de faire un copier/coller d'un texte ;

- ✓ La mise à jour **1.6** « **Donut** » sortie en septembre 2009 a permis de gérer toutes les définitions allant jusqu'à la HD et a apporté également la fonction de recherche directement en local sur l'appareil Android ou encore les recherches Internet grâce à Google Search. Les deux versions **2.0** « **Éclair** » et **2.1** quant à elles ont apporté beaucoup de fonctionnalités intéressantes, comme l'amélioration du clavier tactile, l'utilisation des fonds d'écran animés, la prise en charge du Bluetooth **2.1**, etc ;

- ✓ La version **2.2** « **Froyo** » sortie en mai 2010 a fortement mis l'accent sur la synergie avec Internet. L'envoi d'applications et de liens instantanés depuis un ordinateur est désormais possible. De plus, Google a annoncé que le navigateur chrome intégré à Android **2.2** est le navigateur mobile le plus rapide au monde grâce à l'intégration du moteur JavaScript V8 (Version 8) ;

- ✓ La version **2.3 « Gingerbread »** voit le jour en 2010 apportant des mises à jour pour l'interface utilisateur et des améliorations du système, des performances de réseau sur la version 4G du Nexus S, de Gmail, de l'autonomie et enfin de la caméra. Ajouté à cela la correction du bug du Bluetooth sur le Samsung Galaxy S, ainsi que d'autres fonctionnalités ;

- ✓ Android **3.0 « Honeycomb »** est apparue en février 2011 et est spécialement étudiée pour les tablettes tactiles. Cette mise à jour comprend de nombreux changements dans l'interface utilisateur ;

- ✓ En octobre 2011 apparait la version **4.0 « Ice Cream Sandwich ».** Cette nouvelle version unifiée pour Smartphones et tablettes tactiles apporte de nombreux changements notamment l'ajout des Widgets depuis un nouveau menu similaires à celui des applications, l'amélioration de la messagerie visuelle, l'ajout d'un éditeur de photos et l'amélioration de l'application photo, l'ajout de dictionnaires dans le clavier virtuel, etc ;

- ✓ Juin 2012, Android **4.1 « Jelly Bean »** apparait ayant comme principale nouveauté l'amélioration des fonctionnalités et des performances de l'interface utilisateur rendant celle-ci plus fluide ainsi que l'amélioration de l'accessibilité et la possibilité d'ajout des Widgets depuis le tiroir d'applications depuis un écran d'accueil alternatif sans avoir à rooter l'appareil, etc ;

- ✓ Android **4.4 « KitKat »** sortie en septembre 2013 apporte des mises à jour à l'exemple du clavier Google qui ajoute des emojis, en plus d'un changement de design ainsi que des mises à jour de l'application Téléchargements permettant une meilleure gestion avec de nouvelles options de triage et une nouvelle interface;

- ✓ Android **5.0 « Lollipop »** sortie en novembre 2014 est une évolution majeure d'Android qui propose de nombreuses modifications et nouveautés, et qui étend sa disponibilité sur de nouveaux supports tels que la télévision, la voiture ou les montres connectées.

II. Caractéristiques d'Android

II.1 Architecture d'Android [3]

Android est conçue pour des appareils mobiles au sens large. Nullement restreinte aux téléphones, elle ouvre d'autres possibilités d'utilisation des tablettes, des ordinateurs portables, des bornes interactives, des baladeurs, etc.

La plateforme Android est composée de différentes couches :

- **Un noyau Linux** qui lui confère notamment des caractéristiques multitâches ;
- **Des bibliothèques** graphiques, multimédias ;
- **Une machine virtuelle Java** adaptée : la *Dalvik Virtual Machine ;*
- **Un framework applicatif** proposant des fonctionnalités de gestion de fenêtres, de téléphonie, de gestion de contenu, etc ;
- **Des applications** dont un navigateur web, une gestion des contacts, un calendrier, etc.

Les composants majeurs de la plateforme Android sont résumés sur le schéma suivant :

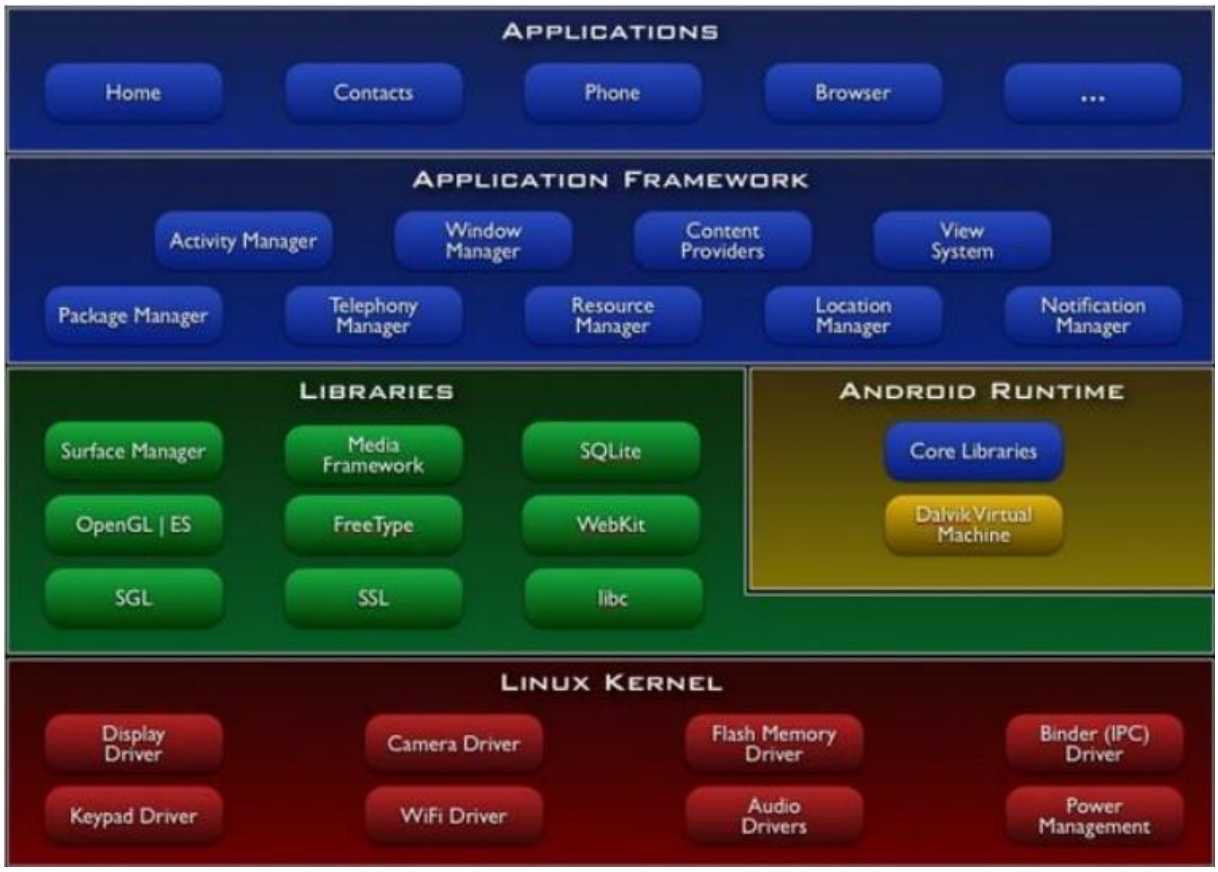


Figure 1.2 : Architecture d'Android.

II.1.1 Les applications

Android est fournie avec un ensemble d'applications dont un client email, une application SMS, un calendrier, un service de cartographie, un navigateur, etc. toutes écrites en Java.

Toutes les applications, qu'elles soient natives ou non, sont construites sur la couche applicative avec les mêmes bibliothèques d'API. Les applications sont exécutées par le

moteur d'exécution *« Runtime »* Android, elles utilisent les classes et les services rendus disponibles par le *Framework*.

II.1.2 Le Framework applicatif

En fournissant une plateforme de développement ouverte, Android offre aux développeurs la possibilité de créer des applications extrêmement riches et innovantes.

Les développeurs ont un accès complet à l'API utilisé par les applications de base. L'architecture d'application est conçue pour simplifier la réutilisation des composants, n'importe quelle application peut publier ses capacités et n'importe quelle autre application peut alors faire usage de ces capacités. Ce même mécanisme permet aux composants de base d'être remplacés par l'utilisateur.

Toutes les applications sous-jacentes forment un ensemble de services et de systèmes, y compris :

- Un jeu extensible de vues qui peuvent être utilisées pour construire une application, y compris des listes, des grilles, des zones de texte, boutons, et même un navigateur web ;
- Des fournisseurs de contenu qui permettent aux applications d'accéder aux données d'autres applications (telles que les Contacts), ou de partager leurs propres données ;
- Un gestionnaire de ressources ;
- Un gestionnaire de notification qui permet d'afficher des alertes personnalisées dans la barre d'état ;
- Un gestionnaire d'activités qui gère le cycle de vie des applications.

II.1.3 Les bibliothèques

Android dispose d'un ensemble de librairies C/C++ utilisées par les différents composants du système. Elles sont offertes aux développeurs à travers le Framework.

Exemples :

- Système de bibliothèque C, dérivé de la bibliothèque C standard du système (libc) ;
- Médiathèques, basées sur *PacketVideo* de *OpenCore* ; les librairies permettant la lecture et l'enregistrement audio et vidéo, ainsi que la gestion des fichiers image, y compris MPEG4, H.264, MP3, AAC, AMR, JPG et PNG ;
- Surface Manager qui gère l'accès au sous-système d'affichage de façon transparente ;

- LibWebCore, un moteur de navigateur web moderne qui fait tourner, à la fois le navigateur Android et une vue web intégrable ;
- SGL, le moteur graphique 2D ; et les bibliothèques 3D, implémentation basée sur OpenGL ES 1.0 ;
- *FreeType*, bitmaps et vectoriels de rendu de polices ;
- SQLite, un moteur de base de données relationnelle puissant et léger, etc.

La bibliothèque d'Android permet la création d'interfaces graphiques selon un procédé similaire aux Framework(s) de quatrième génération (XUL, JavaFX,…) ; l'interface graphique peut être construite par déclaration et peut être utilisée avec plusieurs skins.

La programmation consiste à déclarer la composition de l'interface dans des fichiers XML ; la description peut comporter des ressources (textes, images,...). Ces déclarations sont ensuite transformées en objets tels que des fenêtres et des boutons, qui peuvent être manipulés par de la programmation Java. Les écrans ou les fenêtres (nommés activités dans le jargon d'Android), sont remplis de plusieurs vues ; chaque vue étant une pièce d'interface graphique (bouton, liste, case à cocher…).

Android **3.0**, destinée aux tablettes, introduit la notion de fragments : des panneaux contenant plusieurs éléments visuels. Une tablette ayant, contrairement à un téléphone, généralement suffisamment de place à l'écran pour plusieurs panneaux.

II.1.4 Android Runtime

Android inclut un ensemble de librairies de base offrant la plupart des fonctionnalités disponibles dans les bibliothèques de base du langage de programmation Java.

Chaque application Android s'exécute dans son propre processus, avec sa propre instance de la machine virtuelle Dalvik. Dalvik a été écrite pour que le dispositif puisse faire tourner plusieurs machines virtuelles de manière efficace. La machine virtuelle Dalvik exécute des fichiers dans l'exécutable Dalvik « .DEX », un format optimisé pour ne pas encombrer la mémoire.

La machine virtuelle est la base de registres et fonctionne grâce aux classes compilées par un compilateur Java et transformées dans le format DEX.

La machine virtuelle Dalvik s'appuie sur le noyau Linux pour les fonctionnalités de base telles que le filetage et la gestion de la mémoire de bas niveau.

II.1.5 Le noyau Linux

Android est basée sur un *kernel* Linux **2.6** mais ce n'est pas Linux. Il ne possède pas de système de fenêtrage natif (X Windows System). La glibc n'étant pas supportée, Android utilise une libc customisée appelée Bionic libc.

Android utilise un *kernel* avec différents patches pour la gestion de l'alimentation, le partage mémoire, etc. permettant une meilleure gestion de ces caractéristiques pour les appareils mobiles.

Android n'est pas Linux mais elle est basée sur un *kernel* Linux car ce dernier a un système de gestion mémoire et de processus reconnu pour sa stabilité et ses performances.

Le model de sécurité utilisé par Linux, basé sur un système de permission, est connu pour être robuste et performant. Il n'a pas changé depuis les années 70.

Le *kernel* Linux fournit un système de driver permettant une abstraction avec le matériel. Il permet également le partage de librairies entre différents processus, le chargement et le déchargement de modules à chaud.

Le *kernel* Linux est entièrement open source et il y a une communauté de développeurs qui l'améliorèrent et rajoutent des drivers.

II.2 SDK Android [2]

Exploiter une nouvelle plateforme n'est jamais chose aisée. C'est pourquoi Google fournit, en plus du système d'exploitation, un kit de développement « *Software Development Toolkit* » ou SDK. Ce SDK est un ensemble d'outils qui permet aux développeurs et aux entreprises de créer des applications.

Le SDK Android est composé de plusieurs éléments pour aider les développeurs à créer et à maintenir des applications :

- Des API « *Application Programming Interface* » ou « Interfaces de Programmation » ;
- Des exemples de code ;
- De la documentation ;
- Des outils, parmi lesquels un émulateur permettant de couvrir quasiment toutes les étapes du cycle de développement d'une application.

II.2.1 Les API(s)

Une API « *Application Programming Interface* » est un ensemble de classes regroupant des fonctions mises à la disposition des développeurs. Ces fonctions ou méthodes peuvent être regroupées dans des bibliothèques logicielles ou des services. Le plus souvent, elles effectuent des traitements de bas niveau et proposent au développeur une interface de plus haut niveau pour qu'il puisse accéder à des fonctionnalités plus facilement et surtout plus immédiatement.

Par exemple, la plupart des systèmes proposent une API graphique permettant d'afficher des éléments graphiques à l'écran (fenêtres, boutons, etc.) sans avoir à gérer le périphérique dans son intégralité et ce, pixel par pixel.

II.2.2 Documentation du SDK

La documentation du SDK Android est scindée en deux parties bien distinctes :

- Le guide du développeur, disponible en HTML dans le répertoire du SDK ;
- La documentation des API au format javadoc, qui est également située dans le répertoire docset accessible grâce au chemin toujours depuis le répertoire d'installation.

À propos de la javadoc

La documentation des classes et des différentes méthodes d'une bibliothèque (ou de l'API Java elle-même) est proposée au format HTML. Pour un développeur Java, cette documentation est incontournable pour trouver une classe ou une méthode mise à disposition par une classe et apprendre à s'en servir.

Cette bible est générée automatiquement à partir des commentaires du code source formatés avec une syntaxe précise. Par exemple on utilise le tag @param pour décrire un des paramètres d'une méthode.

L'outil permettant de produire cette documentation est « javadoc.exe », disponible dans le répertoire « bin » des kits de développement Sun.

Une liste complète des outils accompagnés de leur description est disponible sur le site officiel d'Android.

II.2.3 Les exemples

Le kit de développement est accompagné d'un certain nombre d'exemples illustrant les possibilités du SDK Android. Parmi ces exemples :

- Un jeu du serpent (répertoire Snake) ;

- Un projet qui couvre l'utilisation de plusieurs pans de l'API Android comme les alarmes, les notifications, les menus, etc. (répertoire APIDemos) ;
- Une application pour sauvegarder des notes (répertoire NotePad).

Et bien d'autres exemples.

II.2.4 Les outils de développement du SDK

Le SDK est livré avec un certain nombre d'outils couvrant différents aspects du cycle de développement d'une application Android. En effet, le kit de développement propose une boîte à outils complète pour les tâches de compilation, débogage, génération de code AIDL, signature des applications, etc. **[2]**

Le répertoire Tools du kit de développement contient ainsi un ensemble de programmes dont voici quelques exemples :

1. SQLite

Android ne fournit aucune base de données de son propre chef. C'est donc au développeur que revient la tâche d'en créer de type SQLite.

SQLite est une bibliothèque écrite en C, et comme son nom l'indique utilise un dialecte de SQL pour effectuer des requêtes, des manipulations de données, et des définitions de données.

Il est si efficace en termes de mémoire que le moteur d'exécution d'Android peut l'inclure dans son intégralité

Contrairement aux serveurs de base de données classiques, comme MySQL, il ne reproduit pas le schéma habituel client-serveur mais est directement intégré aux programmes.

L'intégralité de la base de données (déclaration, tables, index et données) est stockée dans un fichier indépendant de la plateforme.

2. Émulateur [4]

L'émulateur est le parfait outil de test et de débogage des applications. C'est une implémentation de la machine virtuelle Dalvik, faisant de celle-ci une plateforme exécutant les applications Android comme le ferait n'importe quel téléphone Android.

Étant découplé de tout matériel, c'est une excellente base de test des applications. Il fournit une connectivité réseau complète ainsi que la possibilité d'ajuster finement vitesse de connexion et latence au cours du débogage des applications. Il est également possible de simuler l'envoi et la réception d'appels et de SMS.

Le plugin ADT intègre l'émulateur à Eclipse, qui le lance automatiquement avec l'AVD sélectionné lorsqu'un projet est exécuté ou débogué. Si le plugin n'est pas utilisé ou l'émulateur est utilisé hors d'Eclipse, il devient possible de s'y connecter via Telnet et de le contrôler depuis une console.

Avant d'exécuter l'émulateur, un appareil virtuel doit être créé, celui-ci sera lancé par l'émulateur, qui y exécutera une instance Dalvik.

Figure 1.3 : Émulateur Android.

3. ADB [4]

L'ADB « *Android Debug Bridge* » est à la fois un client et un service permettant de se connecter à un émulateur Android ou un appareil. Il est composé de trois parties :

- Un démon exécuté par l'émulateur ;
- Un service exécuté par la machine de développement ;
- Des applications clientes (comme le DDMS) qui communiquent avec le démon via le service.

Jouant le rôle d'un conduit de communications entre la machine de développement et l'émulateur ou le dispositif Android, ADB permet d'installer des applications, de copier ou lire des fichiers et d'exécuter des commandes Shell sur le dispositif cible.

Grâce au Shell du dispositif, il est possible de modifier les réglages des journaux et d'interroger ou modifier les bases de données SQL qu'il contient.

L'ADT automatise et simplifie grandement les interactions avec l'ADB, notamment l'installation et la mise à jour d'applications, la journalisation et le transfert de fichiers (via la perspective DDMS).

4. DDMS [4]

Contrairement à l'émulateur qui permet de voir à quoi ressemble l'application et la façon dont elle se comporte et interagit, le DDMS permet de voir ce qui se passe en profondeur. En effet, c'est un puissant outil de débogage avec lequel il est possible d'interroger les processus actifs, d'examiner la pile et le tas, de surveiller et de mettre en pause les threads actifs et enfin d'explorer le système de fichier de n'importe quel matériel Android connecté.

La perspective DDMS sous Eclipse fournit également un accès simplifié aux captures d'écran de l'émulateur et aux journaux générés par LogCat. Le DDMS « *Dalvik Debug Monitor Service* » est complètement intégré à Eclipse via le plugin ADT et disponible dans une perspective. Il est aussi possible de lancer le DDMS depuis la ligne de commande : il se connectera automatiquement à tout appareil ou émulateur actif.

Les composants de DDMS

- Debug Process Icon : il montre l'état de connexion du débogueur.
- Update Heap : en cliquant sur ce tas d'informations, il permet de vider le processus de sorte que la mémoire ne soit pas libérée manuellement.
- Dump HPROF : génère un vidage de tas, utile pour le suivi des fuites de mémoire dans l'application.
- Cause : appelle le garbage collector de collecter des données heap.
- Update thread : affiche les informations sur le fil conducteur pour le processus sélectionné
- Start Method Profiling : assure le suivi des mesures liées à une méthode. Il collecte des informations comme la quantité de temps nécessaire pour exécuter une méthode, le nombre d'appels.
- Stop Process : arrête le processus en cours de sélection.
- Screen Capture : prend une capture d'écran de ce qui est affiché sur l'écran.
- Reset ADB : comme son nom l'indique, il réinitialise l'ADB.

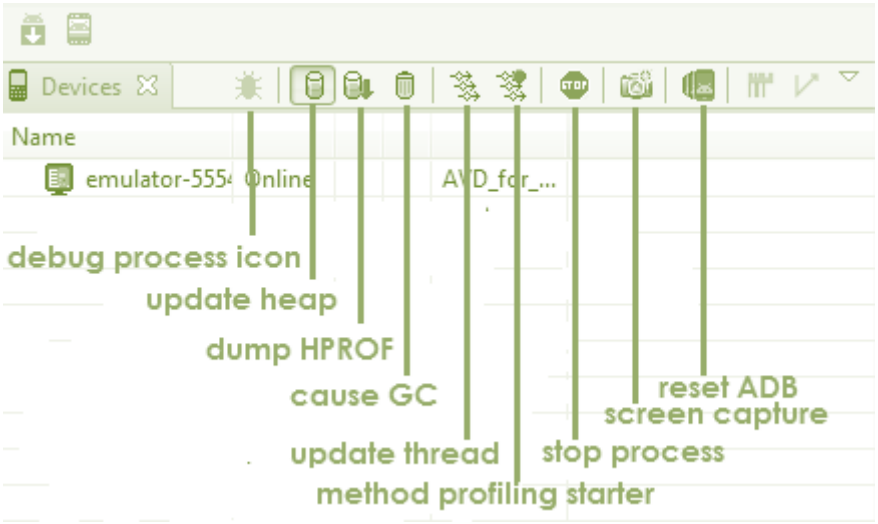


Figure 1.4 : Composants du DDMS.

III. Concepts d'Android

Android possède aussi un environnement très riche, ainsi que des fonctionnalités toutes aussi intéressantes, voici quelques unes auxquelles nous feront appel dans la réalisation de notre projet :

1. Bureau virtuel

Le bureau virtuel est l'écran d'accueil du Smartphone, il est composé de 5 parties (ou plus). Chacune est personnalisable, il est possible d'y mettre des raccourcis (vers des applications, des fichiers, des dossiers, des contacts, etc.) ou des Widgets (calendrier, horloge, notes, etc.).

L'image de fond s'étend aussi sur tout le bureau et bouge dés qu'on passe sur, d'une partie à une autre, ce qui donne l'impression que le contenu fait partie du décor. Cependant, et dans le but de faire la différence, les constructeurs ajoutent leurs touches personnelles par exemple : le constructeur HTC possédant l'interface Sence qui est dotée de 7 bureaux.

Le problème avec cette personnalisation est que lorsqu'une nouvelle version d'Android sort, chaque constructeur doit procéder aux adaptations dans sa propre version. Donc soit la mise à jour n'est jamais effectuée, soit elle sort plusieurs semaines après la version de base.

Figure 1.5 : Bureau virtuel Android.

2. Widgets [2] [6]

Les Widgets sont des éléments d'interface graphique ou des composants visuels qui peuvent être ajoutés à l'écran de démarrage ou peuvent être utilisés dans une application permettant ainsi à l'utilisateur d'interagir avec celle-ci. Chaque Widget possède un nombre important d'attributs XML et de méthodes Java.

Les Widgets peuvent être aussi définis comme étant des vues standards incluses dans la plateforme Android. Les vues héritant toutes de la classe View, chaque Widget hérite aussi de cette classe. Android tire profit des méthodes de chaque vue et de chaque Widget pour former une plateforme paramétrable et modulaire.

3. Android Market [5]

L'*Android Market* renommé *Google Play Store* en mars 2012 est le système de publication d'applications officiel d'Android. En effet, c'est une application préinstallée sur chaque téléphone fonctionnant sous Android, permettant de télécharger des « applications » développés par des sociétés ou des développeurs indépendants.

Néanmoins, contrairement à l'*App Store d'Apple*, la distribution d'applications au public n'est pas un monopole de Google et techniquement, rien n'empêche d'installer des applications depuis une autre source. D'ailleurs, des alternatives à *Android Market*, comme par exemple *SlideMe* émergent peu à peu.

Pour proposer son application sur *Android Market*, il faudra détenir un compte Google et s'acquitter des frais d'enregistrement qui s'élèvent à 15 $.

Bien sûr, l'application devra être parfaitement packagée et signée, par ailleurs *Android Market* requiert que la date de validité du certificat soit postérieure au 22 octobre 2033, qu'une icône et un intitulé soient précisés dans le manifeste et que les champs de version soient renseignés.

Figure 1.6 : Site web d'Android Market.

Conclusion

Au cours de ce chapitre, nous avons présenté la plateforme Android, ses différentes caractéristiques ainsi que le kit de développement Android SDK.

Les différents concepts traités dans ce chapitre nous aiderons à comprendre au mieux notre environnement de développement et les notions fondamentales pour mener à bien notre application.

Les étapes et détails de la conception et la réalisation de notre application feront l'objet du prochain chapitre.

CHAPITRE 2

« Analyse et Conception »

Introduction

Dans le chapitre précédent, nous avons présenté des généralités concernant la plateforme Android et son environnement de développement. Ce présent chapitre sera consacré à l'analyse et à la conception.

Avant l'implémentation et la réalisation de toute application informatique, il convient de suivre une démarche méthodologique pour mettre en place cette dernière. Pour réaliser notre application, nous commençons par une analyse profonde et bien réfléchie, suivie d'une conception en se basant sur le formalisme UML « *Unified Modeling Language* » qui permet de bien représenter les aspects statiques et dynamiques de notre projet par la série des diagrammes qu'il offre.

I. Analyse

La phase d'analyse débute par la mise en évidence des différents acteurs qui interviennent dans le système, et leurs besoins. Ensuite, modéliser les objectifs à atteindre dans la phase de conception et ce en s'appuyant sur la phase d'analyse.

Pour se faire, nous avons utilisé le formalisme de modélisation UML qui sera présenté dans le point suivant.

I.1 Présentation du langage UML

I.1.1 Définition [9]

UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue.UML unifie à la fois les notations et les concepts orientés objet . Il ne s'agit pas d'une simple notation graphique, car les concepts transmis par un diagramme ont une sémantique précise et sont porteurs de sens au même titre que les mots d'un langage.

UML unifie également les notations nécessaires aux différentes activités d'un processus de développement et offre, par ce biais, le moyen d'établir le suivi des décisions prises, depuis l'expression du besoin jusqu'au codage. Dans ce cadre, un concept appartenant aux exigences des utilisateurs projette sa réalité dans le modèle de conception et dans le codage. Le fil tendu entre les différentes étapes de construction permet alors de remonter du code aux besoins et d'en comprendre les tenants et les aboutissants. En d'autres termes, on peut retrouver la nécessité d'un bloc de code en se référant à son origine dans le modèle des besoins.

I.2 Objectif de l'application

Les applications mobiles sont actuellement en pleine croissance grâce aux succès des Smartphones et plus particulièrement ceux qui tournent sous Android OS. Cette tendance offre aux entreprises de formidables opportunités pour créer de nouveaux services ou élargir leurs supports de communication.

Même avec l'existence de plusieurs agences de compagnie aérienne à l'échelle nationale et internationale qui effectuent la réservation, le problème de déplacement reste inévitable, car les différentes agences commerciales, en général, se situent dans les grandes villes ce qui n'arrange pas tout le monde, notamment ceux qui habitent les coins isolés du pays.

Pour remédier à ce genre de problème et améliorer la qualité de son service, une compagnie aérienne, comme toute autre entreprise, peut faire appel à cette nouvelle technologie à savoir : les applications mobiles pour faire évoluer la communication avec ses clients et augmenter le nombre de clients fidèles ou encore de mieux se positionner sur le marché et être plus compétitive.

Ainsi, l'objectif de ce travail est de réaliser ce besoin en développant une application mobile sous Android permettant d'effectuer des réservations ou tout simplement de consulter les vols disponibles pour une compagnie aérienne et ce n'importe où et à n'importe quel moment disposant uniquement de l'application installée sur le Smartphone personnel.

I.3 Les acteurs du système

1. Identification des acteurs

Un acteur est un utilisateur type qui a toujours le même comportement vis-à-vis d'un cas d'utilisation. Ainsi les utilisateurs d'un système appartiennent à une ou plusieurs classes d'acteurs selon les rôles qu'ils tiennent par rapport au système. **[10]**

2. Extraction des acteurs

- **Administrateur (en background) :** est le créateur de l'application ou encore la personne chargée du maintien et de la mise à jour de la BDD et de l'application ;

- **Utilisateur simple :** toute personne pouvant utiliser l'application pour visualiser les offres promotionnelles, les vols disponibles ou tout simplement effectuer une recherche rapide ;

- **Client :** toute personne pouvant accéder à son compte personnel et effectuer une recherche d'offres promotionnelles, de vols disponibles ou tout simplement

effectuer une recherche rapide. Un client peut aussi faire une réservation ou encore accéder aux réservations personnelles pour effectuer d'éventuels changements (modifier / annuler une réservation).

I.4 Cas d'utilisations

I.4.1 Définition d'un cas d'utilisation

Un cas d'utilisation « *use case* » modélise une interaction entre le système informatique à développer et un utilisateur ou acteur interagissant avec le système. Plus précisément, un cas d'utilisation décrit une séquence d'actions réalisées par le système qui produit un résultat observable pour un acteur. **[12]**

Le format de représentation d'un cas d'utilisation est complètement libre, cependant UML propose un formalisme et des concepts issus de bonnes pratiques.

I.4.2 Relations entre cas d'utilisations [10]

- **Relation « *Include* »**

Une relation d'inclusion d'un cas d'utilisation **A** à un cas d'utilisation **B** signifie qu'une instance de **A** contient le comportement décrit dans **B.** Le cas d'utilisation **A** ne peut être utilisé seul.

- **Relation « *Extend* »**

Une relation d'extension d'un cas d'utilisation **A** par rapport à un cas d'utilisation **B** signifie qu'une instance de **A** peut être étendue par le comportement décrit dans **B.**

- **Relation « *Use* »**

Lorsqu'une ou plusieurs tâches sont utilisées régulièrement on peut les factoriser dans un même use case et faire de telle sorte que d'autres uses cases l'utilisent en pointant par une flèche.

I.4.3 Diagramme de cas d'utilisation général

Un diagramme de cas d'utilisation est une vue graphique de tous les acteurs d'un système, de ses cas d'utilisation et de leur interactions. Il permet de montrer quels acteurs interagissent avec chaque cas d'utilisation.

La figure suivante représente le diagramme de cas d'utilisation général de notre application.

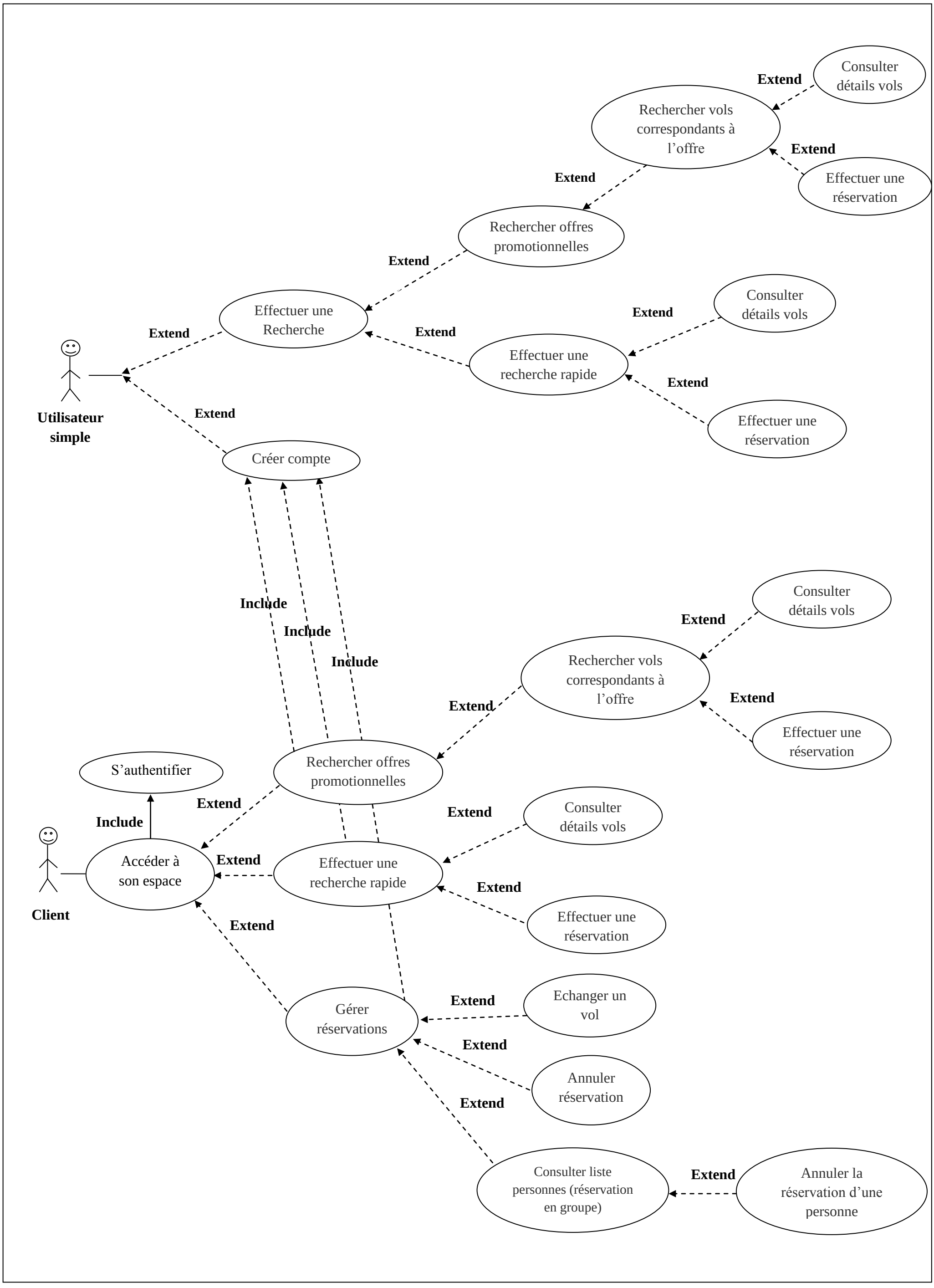
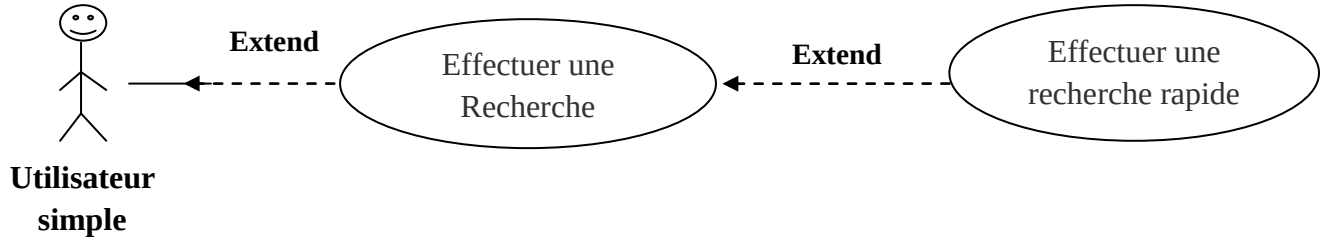


Figure 2.1 : Diagramme de cas d'utilisation général.

I.4.4 Quelques diagrammes de cas d'utilisations

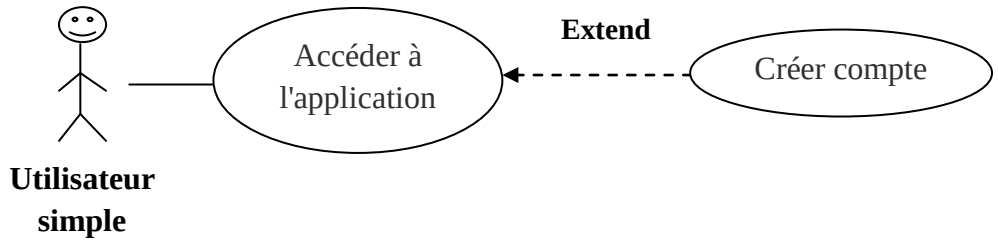
Dans ce qui suit, nous allons présenter quelques diagrammes correspondant à quelques cas d'utilisation de notre application :



Scénario nominal :

1. L'utilisateur accède à l'application ;
2. L'utilisateur clique sur le bouton « Rechercher » ;
3. Le système affiche l'interface de recherche ;
4. L'utilisateur clique sur le bouton « Recherche Rapide» ;
5. L'utilisateur remplit le formulaire de recherche contenant le départ, la date de départ, l'arrivée, la date d'arrivée, etc ;
6. Le système affiche la liste des vols correspondants à la requête.

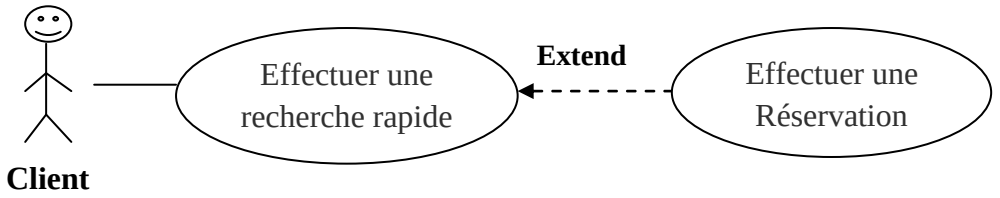
Figure 2.2 : Diagramme de cas d'utilisation « Effectuer une recherche rapide ».



Scénario nominal :

1. L'utilisateur accède à l'application ;
2. L'utilisateur clique sur le bouton « Créer Compte » ;
3. L'utilisateur remplit le formulaire d'inscription contenant le nom d'utilisateur, le mot de passe, le nom, le prénom etc ;
4. Le système affiche un message de succès (Compte crée) ou d'erreur dans le cas où l'un des champs saisis est erroné ou dans le cas où le nom d'utilisateur existe déjà.

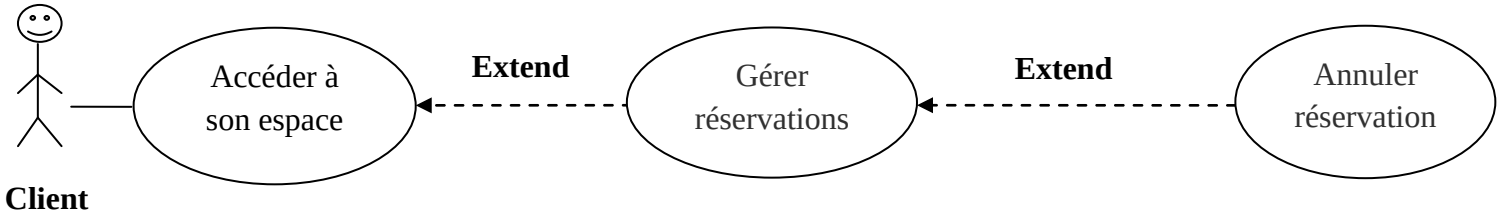
Figure 2.3: Diagramme de cas d'utilisation « Créer compte ».



Scénario nominal :

1. Le client remplit le formulaire de recherche contenant le départ, la date de départ, l'arrivée, la date d'arrivée, etc ;
2. Le système affiche la liste des vols correspondants à la requête.
3. le client coche un vol et clique sur le bouton « Sélectionner » ;
4. Le système affiche les détails et les tarifs de la réservation ;
5. le client clique sur le bouton « Continuer » ;
6. le système enregistre la demande ;
7. Le client est redirigé vers son espace où une liste de réservations lui est affichée ;

Figure 2.4: Diagramme de cas d'utilisation « Effectuer une réservation ».



Scénario nominal :

1. Le client accède à son espace après son authentification ;
2. Le client clique sur le bouton « Mes réservations » ;
3. Le système affiche la liste des réservations ;
4. Le client sélectionne une réservation à annuler et clique sur le bouton « Annuler » ;
5. Le système affiche une boite de dialogue pour confirmer l'annulation ;
6. Le client clique sur le bouton « Oui » pour confirmer l'annulation ou le bouton « Non » pour annuler son choix ;
7. Le système supprime la réservation de la BDD si le client appuie sur le bouton « Oui » et affiche la nouvelle liste.

Figure 2.5: Diagramme de cas d'utilisation « Annuler une réservation ».

II. Conception

La conception est en grande partie un processus d'affinement du modèle d'analyse, cependant elle consiste à apporter des solutions techniques aux descriptions définies lors de l'analyse : architecture techniques, performances et optimisation, stratégies de programmation.

Pour développer notre projet nous sommes appelés à concevoir l'application et la base de données avec laquelle interagie le système. Dans la conception de notre application, nous allons construire les diagrammes de séquence suivis des diagrammes de classes et enfin du diagramme de classe général.

II.1 Diagrammes de séquence [10]

Les diagrammes de séquences permettent de décrire les scenarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets.

Ce type de diagramme insiste sur l'aspect temporel, ils sont formés avec des classes traduisant la dynamique du système et qui seront utilisées dans l'activité de conception.

Les digrammes de séquence correspondants à notre application sont :

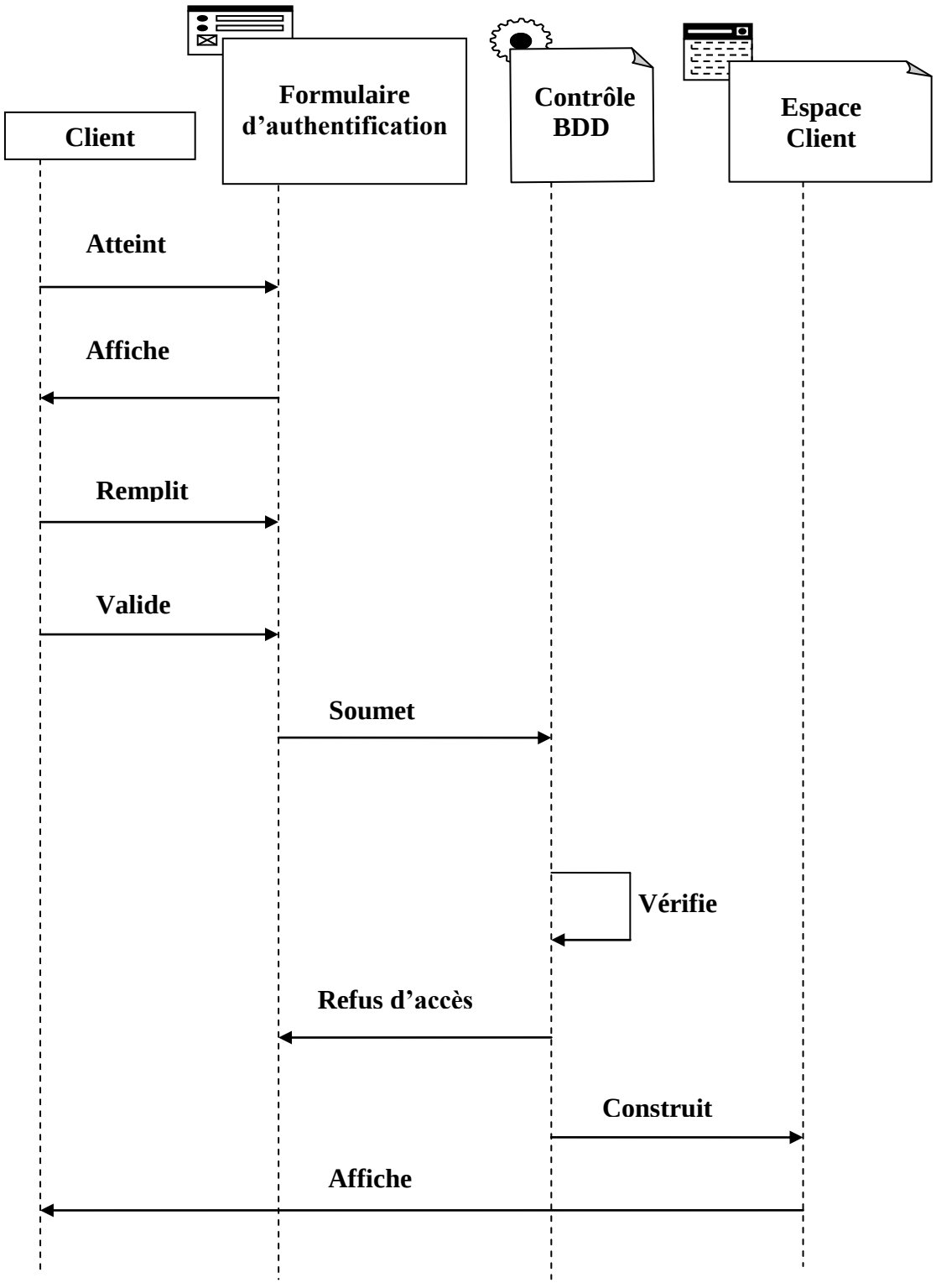


Figure 2.6 : Diagramme de séquence pour le cas d'utilisation « Authentification du client ».

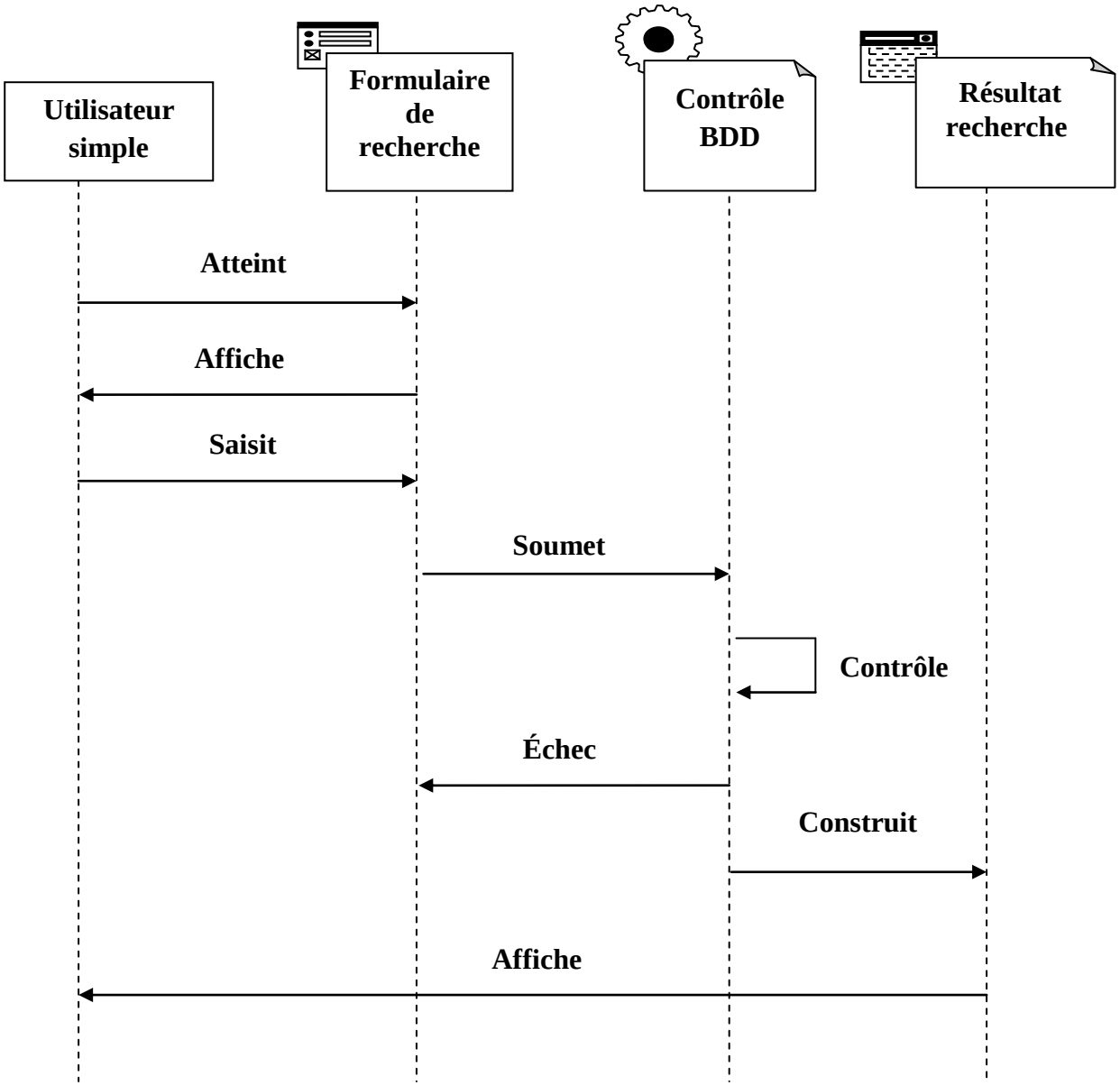


Figure 2.7 : Diagramme de séquence pour le cas d'utilisation « Rechercher vol, acteur : utilisateur simple ».

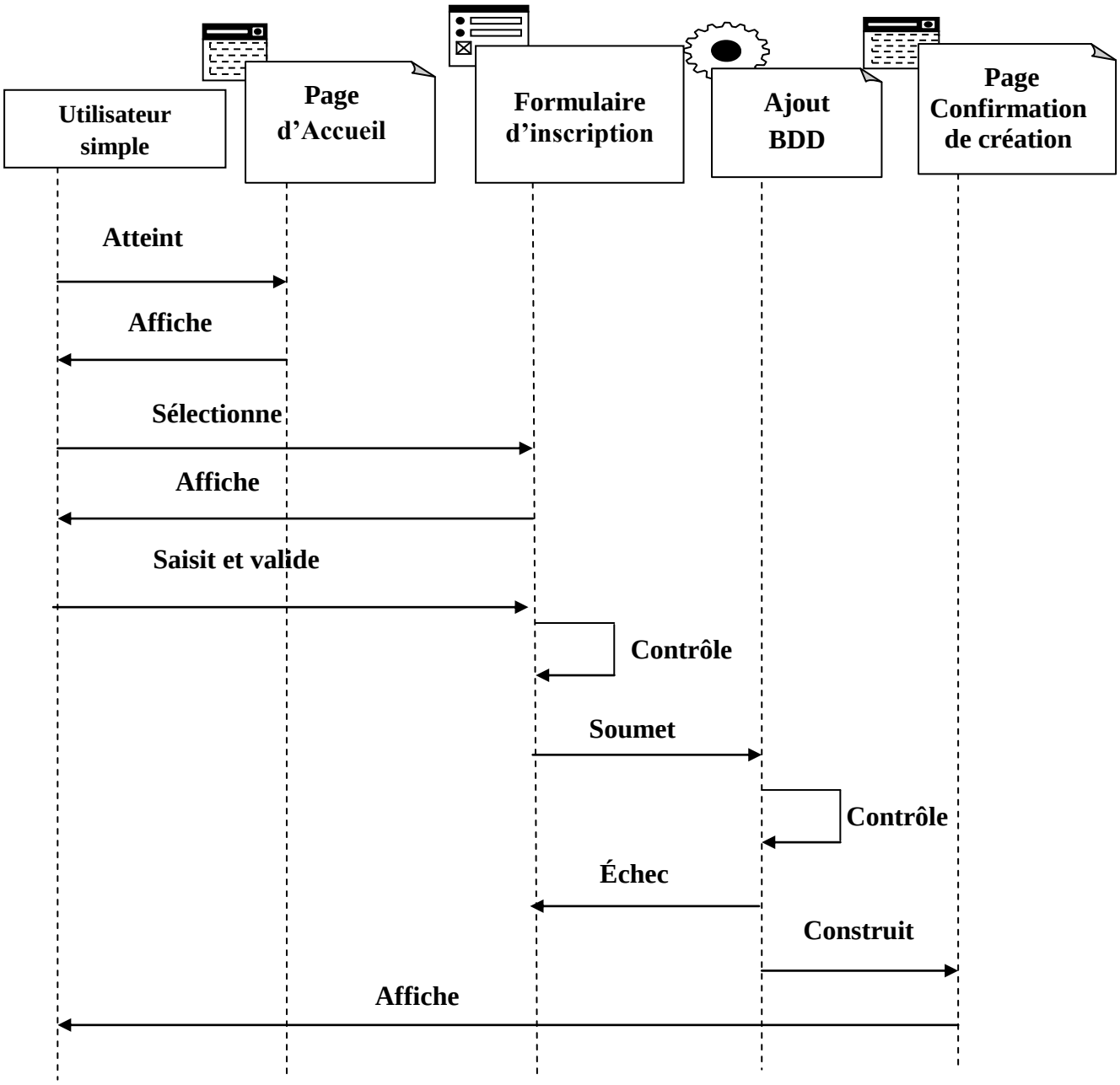


Figure 2.8 : Diagramme de séquence pour le cas d'utilisation « Création nouveau compte utilisateur, acteur : utilisateur simple ».

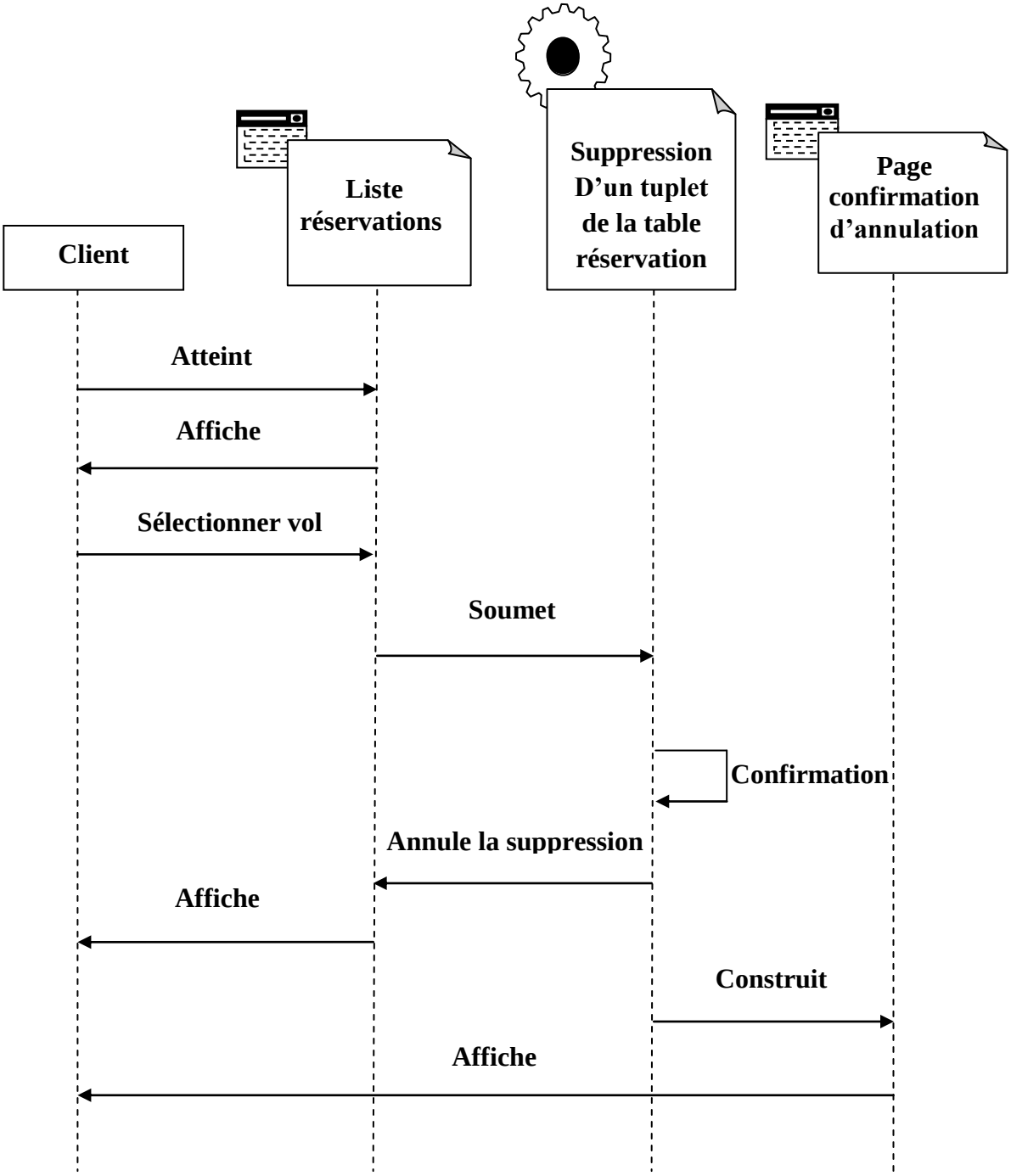


Figure 2.9 : Diagramme de séquence pour le cas d'utilisation « Annulation d'une réservation, acteur : client ».

II.2 Diagrammes de classes

Le diagramme de classes représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboitage conceptuel (héritage, marqué par une flèche terminée par un triangle) ou une relation organique (agrégation, marqué par une flèche, terminée par un diamant), Il est l'un des principaux diagrammes de l'UML. **[13]**

La réalisation des diagrammes de classes se fait à l'aide des associations, nous distinguons quatre types d'association :

- **Lien « *Link* »** : c'est une association entre une page client et une autre page client ou serveur dans le diagramme de classe. Elle représente un pointeur entre ces pages. A chaque lien correspond une balise ancre HTML.

- **Soumet « *Submit* »** : une association de soumission se trouve toujours entre un formulaire et une page serveur. Les valeurs des champs du formulaire sont soumises au serveur qui les traite, par l'intermédiaire de pages serveur. Le serveur traite la page serveur qui accepte et utilise les informations de formulaire.

- **Construit « *Build* »** : c'est une association particulière orientée entre les pages clients et les pages serveurs. Elle indique quelle page serveur est responsable de la création de la page client. Une page serveur peut construire plusieurs pages clients, mais une page client n'est construite que par une et une seule page serveur.

- **Rediriger « *Redirect* »** : c'est une association unidirectionnelle qui relie deux pages client ou serveur.

Pour la conception de notre application, nous avons réalisé les diagrammes de classes suivants :

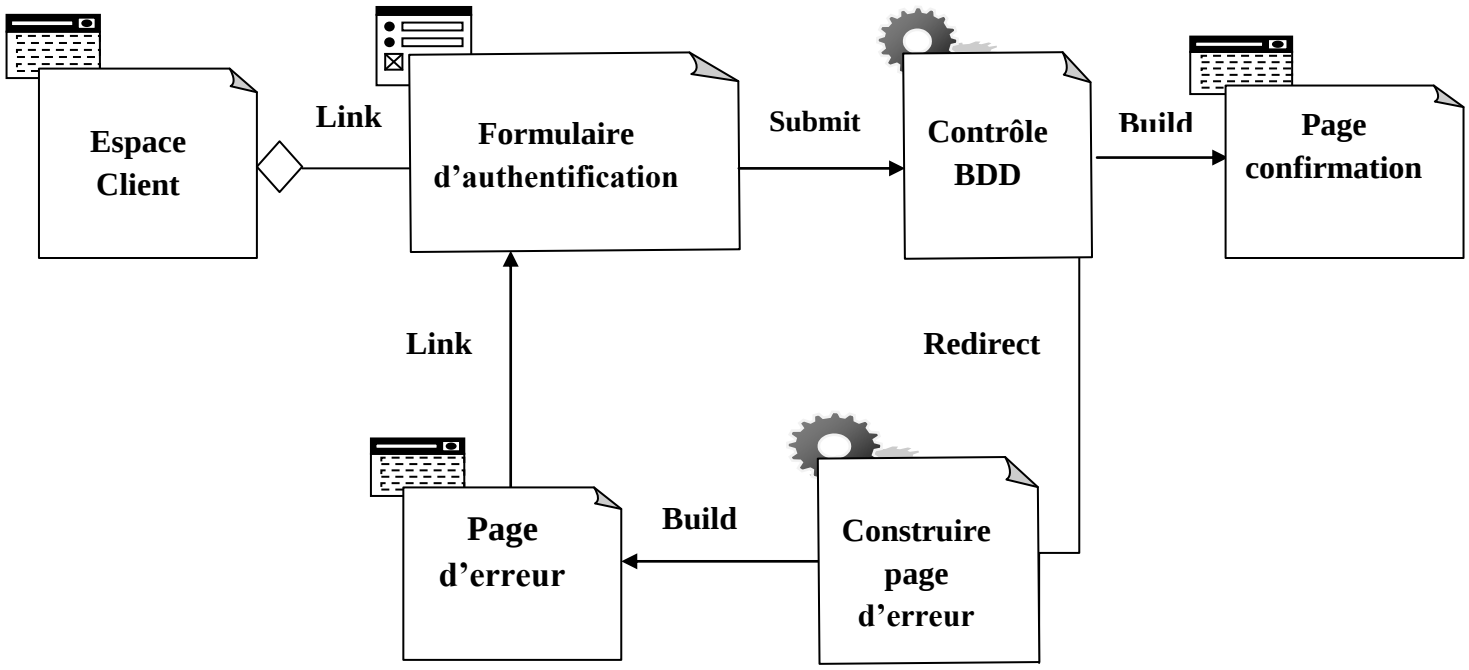


Figure 2.10 : Diagramme de classes pour le cas d'utilisation « Authentification d'un client ».

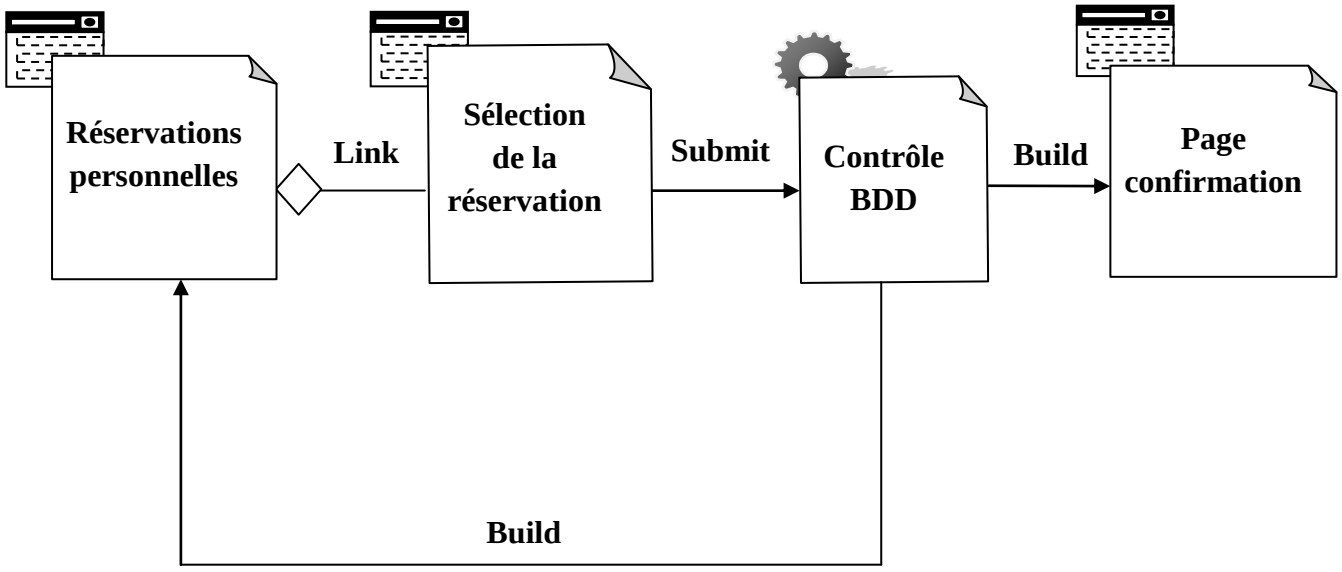


Figure 2.11 : Diagramme de classes pour le cas d'utilisation « Annulation d'une réservation, acteur : client ».

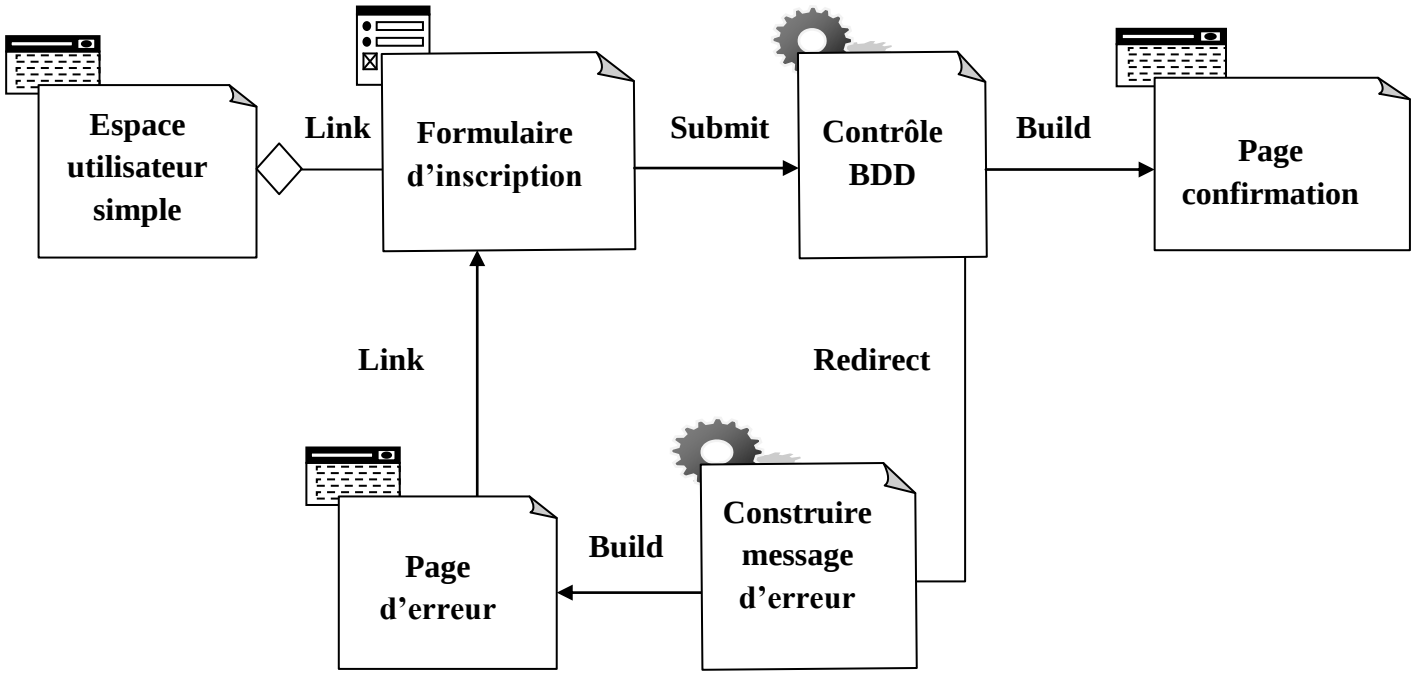


Figure 2.12 : Diagramme de classes pour le cas d'utilisation « Création d'un compte, acteur : utilisateur simple ».

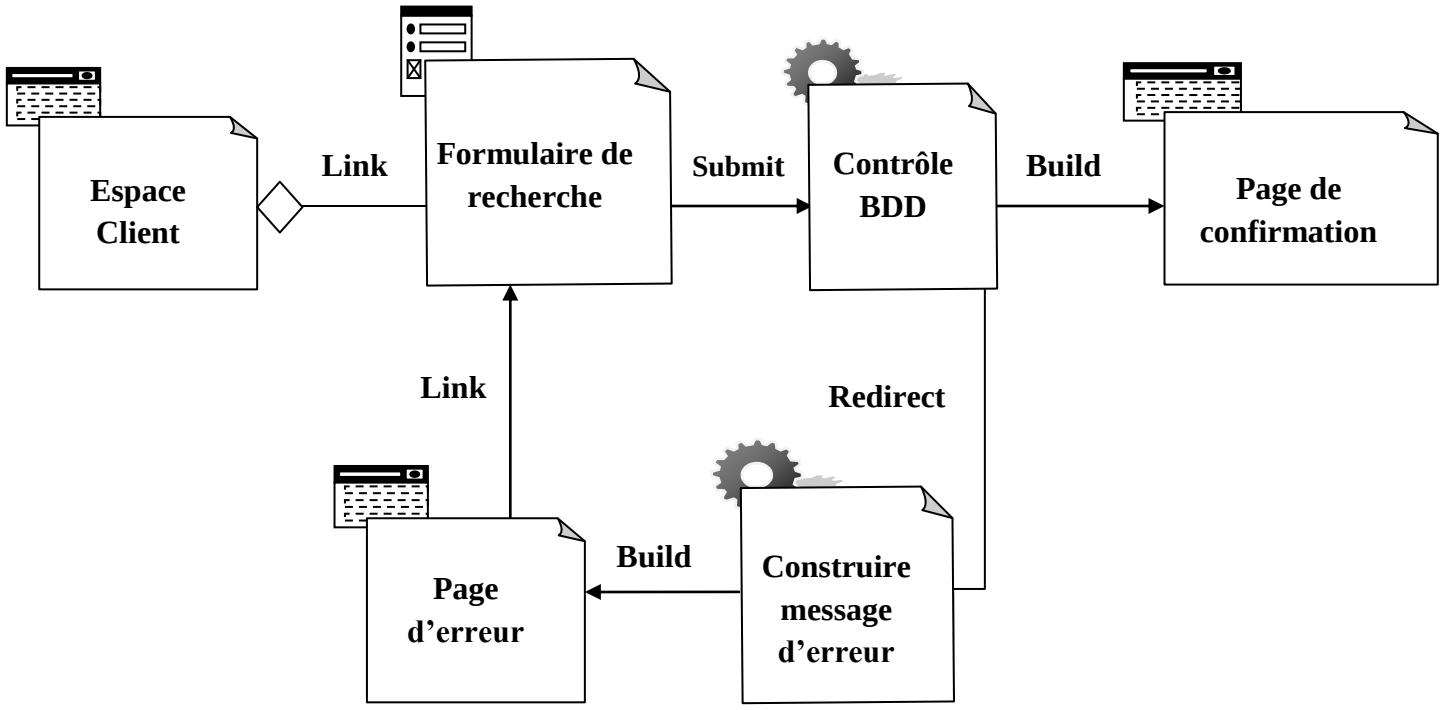


Figure 2.13 : Diagramme de classes pour le cas d'utilisation « Rechercher vol, acteur : client ».

II.3 Diagrammes de classe général

Le diagramme de classes global est représenté par le schéma suivant :

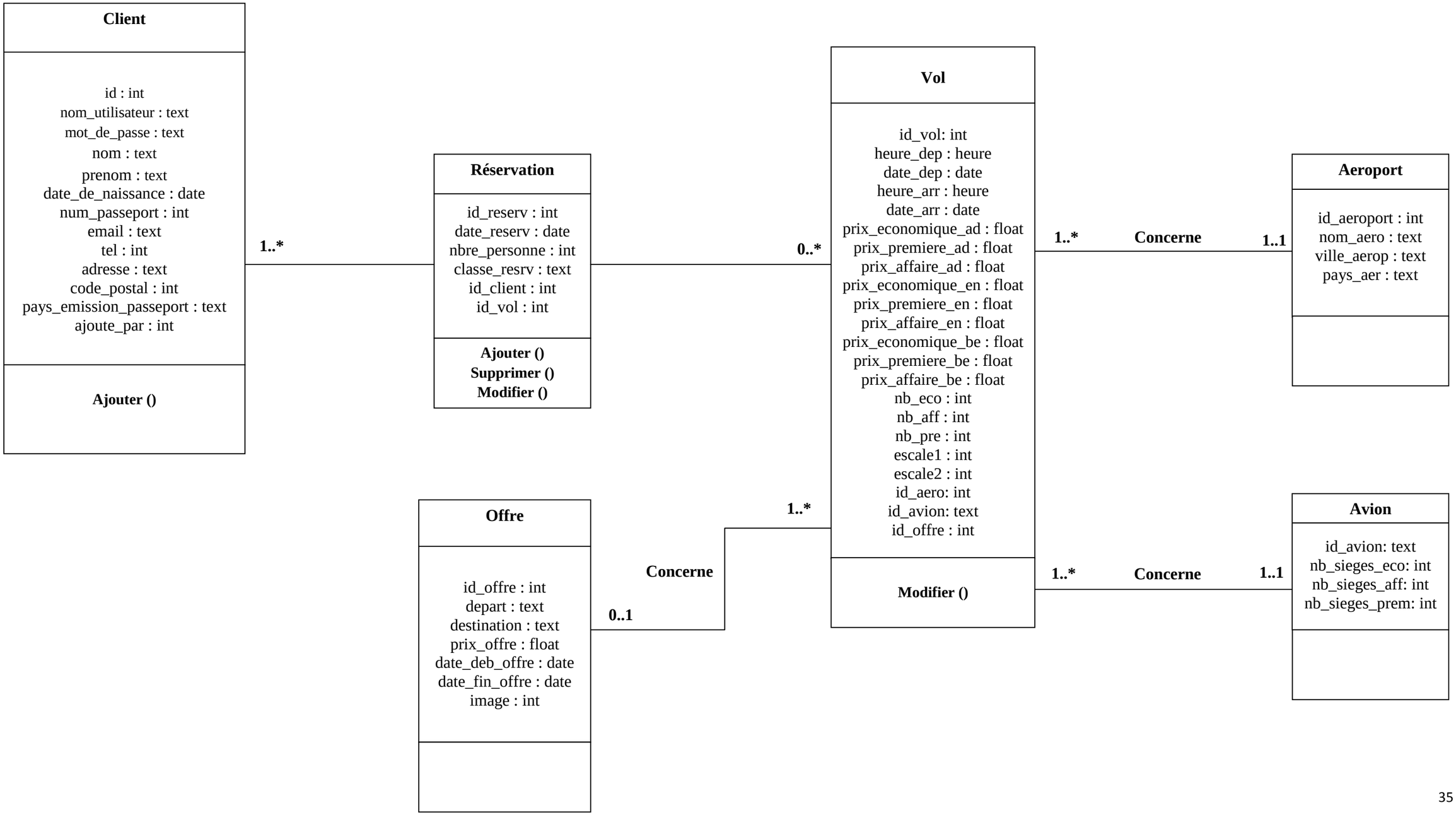


Figure 2.14 : Diagramme de classes global.

III. Conception de la base de données

Une base de données « BDD » est un ensemble de données mémorisées sur des supports accessibles par un ordinateur pour satisfaire simultanément plusieurs utilisateurs de façon sélective et en temps très court. Elles constituent le cœur du système d'information.

Notre base de données est constituée de six (06) tables à savoir : la table « client », la table « vol », la table « offres_promotionnelles », la table « reservation », la table « aeroport», et enfin la table « avion ».

Ces six tables sont liées entre elles avec les relations suivantes :

- Un vol peut-être réservé par un ou plusieurs clients ;
- Un client peut effectuer plusieurs réservations ou n'effectuer aucune ;
- Une offre concerne un ou plusieurs vols ;
- Un vol peut-être concerné par une offre, comme peut ne pas l'être ;
- Un avion décolle/atterrit d'un seul aéroport pour un vol précis ;
- Un client peut effectuer une réservation pour une ou plusieurs personnes.

III.1 Les différentes tables avec la codification utilisée

Chaque table stockée dans notre base de données contient un ensemble d'attributs qui seront détaillés dans ce qui suit :

- **Table « client »**

Champ	Signification	Type	Clé
id	Identifiant unique pour chaque client auto-incrémenté.	INT	**Primaire (PK)**
nom_utilisateur	Nom d'utilisateur pour la connexion à l'espace personnel du client.	TEXT	
mot_de_passe	Mot de passe associé au nom d'utilisateur. La combinaison de ces deux derniers permet au client d'accéder à son propre espace.	TEXT	
nom	Le nom du client.	TEXT	

Champ	Signification	type	Clé
prenom	Le prénom du client.	TEXT	
date_de_naissance	La date de naissance du client.	DATE	
num_passeport	Le numéro passeport du client.	INT	
email	L'adresse électronique du client.	TEXT	
tel	Le numéro de téléphone du client.	INT	
adresse	L'adresse du client.	TEXT	
code_postal	Le code postal du client.	INT	
pays_emission_passeport	Le pays d'émission du passeport du client.	TEXT	
ajoute_par	Clé étrangère vers la table elle-même représentant l'identifiant du client ayant effectué une réservation pour un groupe de personne.	INT	**Etrangère (FK)**

- **Table « vol »**

Champ	Signification	Type	Clé
id_vol	Identifiant unique du vol auto - incrémenté.	INT	**Primaire (PK)**
id_avion	Référence/ identifiant de l'avion.	TEXT	**Etrangère (FK)**
id_aero	Identifiant de l'aéroport.	INT	**Etrangère (FK)**
heure_dep	Heure de départ.	HEURE	
date_dep	Date de départ.	DATE	
heure_arr	Heure d'arrivée.	HEURE	
date_arr	Date d'arrivée.	DATE	
escale1	Clé étrangère vers la table elle-même représentant l'identifiant du premier vol si celui-ci comprend une escale.	INT	**Etrangère (FK)**
escale2	Clé étrangère vers la table elle-même représentant l'identifiant du deuxième vol si celui-ci comprend une escale.	INT	**Etrangère (FK)**

Champ	Signification	Type	Clé
prix_economique_ad	Prix pour adulte dans la première classe.	FLOAT	
prix_premiere_ad	Prix pour adulte dans la classe affaire.	FLOAT	
prix_affaire_ad	Prix pour adulte dans la classe économique.	FLOAT	
prix_economique_en	Prix pour enfant dans la classe économique.	FLOAT	
prix_premiere_en	Prix pour enfant dans la première classe.	FLOAT	
prix_affaire_en	Prix pour enfant dans la classe affaires.	FLOAT	
prix_economique_be	Prix pour bébé dans la classe économique.	FLOAT	
prix_premiere_be	Prix pour bébé dans la première classe.	FLOAT	
prix_affaire_be	Prix pour bébé dans la classe affaires.	FLOAT	
nb_eco	Nombre de sièges de la classe économique.	INT	
nb_aff	Nombre de sièges de la classe affaire.	INT	
nb_pre	Nombre de sièges de la première classe.	INT	
id_offre	Identifiant de l'offre.	INT	**Etrangère (FK)**

- **Table « offres_promotionnelles »**

Champ	Signification	Type	Clé
id_offre	Identifiant unique de l'offre auto-incrémenté	INT	**Primaire (PK)**
depart	Pays de départ.	TEXT	
destination	Pays de destination.	TEXT	

Champ	Signification	Type	Clé
prix_offre	Prix de l'offre.	FLOAT	**Étrangère (FK)**
date_deb_offre	La date début de l'offre	DATE	
date_fin_offre	La date fin de l'offre	DATE	
image	Image correspondant au pays de destination	INT	

- **Table « reservation »**

Champ	Signification	Type	Clé
id_reserv	Identifiant unique de la réservation auto-incrémenté	INT	**Primaire (PK)**
date_reserv	Date de la réservation.	DATE	
id_client	Identifiant du Client.	INT	**Étrangère (FK)**
id_vol	Identifiant du Vol.	INT	**Étrangère (FK)**
nbre_personne	Nombre de personnes ayant effectué la réservation.	INT	
classe_resrv	La classe de la réservation.	TEXT	

- **Table « aeroport»**

Champ	Signification	Type	Clé
id_aeroport	Identifiant unique de l'aéroport auto-incrémenté.	INT	**Primaire (PK)**
nom_aero	Nom de l'aéroport.	TEXT	
ville_aero	Ville de l'aéroport.	TEXT	
pays_aero	Pays de l'aéroport.	TEXT	

- **Table « avion»**

Champ	Signification	Type	Clé
id_avion	Identifiant unique de l'avion auto-incrémenté.	TEXT	**Primaire (PK)**
nb_sieges_eco	Nombre de sièges de la classe économique.	INT	
nb_sieges_aff	Nombre de sièges de la classe affaires.	INT	
nb_sieges_prem	Nombre de sièges de la première classe.	INT	

Conclusion

A l'issue de ce chapitre, nous avons cerné les objectifs de notre application et pour les atteindre nous avons suivi une démarche de modélisation basée sur la méthode UML.

Pour la phase d'analyse, nous avons élaboré le diagramme de cas d'utilisation général et présenté quelques diagrammes de cas d'utilisations, en revanche, pour la partie conception, nous avons construit les diagrammes de séquence, de classes et enfin nous avons conclus avec le diagramme de classes général.

Le prochain chapitre sera consacré à la présentation de l'environnement de travail et des différents outils utilisés ainsi qu'à la présentation de quelques interfaces de notre application.

CHAPITRE 3

« Réalisation »

Introduction

Après avoir explicité dans le chapitre précédent la conception de notre application, nous allons présenter dans ce présent chapitre l'environnement de travail et les outils de développement et d'implémentation, ainsi que les langages de programmation qui nous ont servi d'appui pour la réalisation de notre projet. Enfin, nous terminons par une présentation des différentes fonctionnalités qu'offre notre application à travers diverses interfaces.

I. Environnement de travail

Pour pouvoir réaliser une application dans de bonnes conditions il faut tout d'abord bien choisir l'environnement de travail approprié et ce selon les besoins exprimés et le but ciblé.

I.1 Environnement matériel

Pour la réalisation de notre application nous avons utilisé :

- Un PC portable SONY VAIO, Intel, 2.30GHz, 4 GO de mémoire vive, Windows 7 ;
- Un PC portable Aspire E11, Intel, 2.16GHz, 2 GO de mémoire vive, Windows 8 ;
- Un Smartphone Samsung Galaxy Ace 2 : modèle GT-I8160P version Android 2.3.6.

I.2 Environnement logiciel

Notre projet de fin d'études consiste en la réalisation d'une application mobile sous Android, celle-ci doit être connectée à une base de données interne SQLite ce qui nous impose de travailler sous Eclipse avec le langage de programmation JAVA et le SDK Android pour son développement afin d'obtenir un fichier **.APK** qui sera par la suite installé sur les terminaux mobiles de type Smartphone fonctionnant sous système Android.

II. Technologies utilisées

1. IDE Eclipse [8]

Eclipse IDE « *Integrated Development Environment* » est un environnement de développement intégré principalement écrit en Java, libre, extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation.

La spécificité d'Eclipse IDE vient du fait que son architecture soit totalement développée autour de la notion de plugin : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plugin notamment le plugin « Android ».

Dans le cadre de notre projet, nous avons utilisé la version Eclipse Luna, avec le plugin ADT de Google.

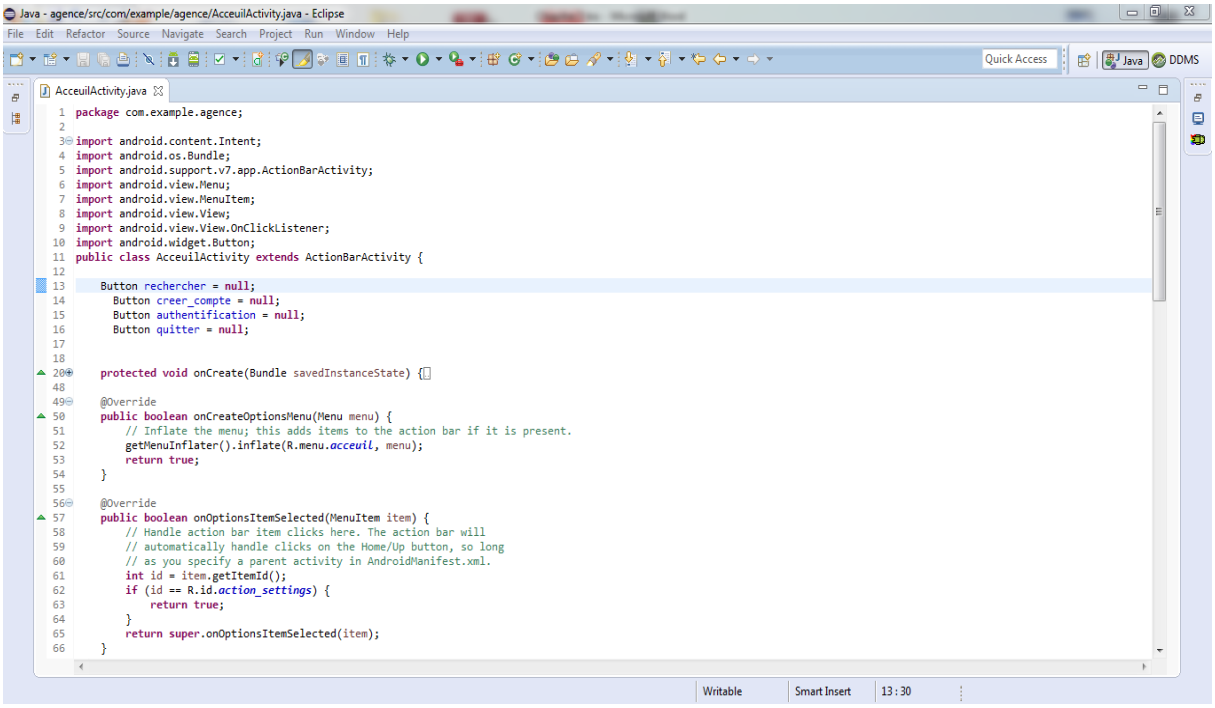


Figure 3.1 : Capture d'écran d'une fenêtre Eclipse.

2. Le SDK Android

Le kit de développement SDK Android est un ensemble complet d'outils dédiés au développement d'applications Android. Il comprend notamment un débogueur, des bibliothèques logicielles, un émulateur, de la documentation, des exemples de code et des tutoriaux. **[2]**

L'IDE officiellement supporté était Eclipse combiné au plugin d'outils de développement d'Android (ADT), cependant depuis 2015, Google officialise « Android Studio » qui devient alors l'IDE officiel pour le SDK Android. **[8]**

Les développeurs peuvent utiliser n'importe quel éditeur de texte pour modifier les fichiers Java et XML, puis utiliser les outils en ligne de commande pour créer, construire et déboguer des applications Android ainsi que contrôler des périphériques Android (pour déclencher un redémarrage, installer un logiciel à distance ou autre). **[8]**

3. Le plugin ADT [8]

Le plugin ADT « *Android Development Tools* » pour Eclipse permet de créer et de déboguer facilement et rapidement les applications, il est conçu pour donner un

environnement puissant, intégré dans lequel il est possible de construire des applications Android.

Le développement dans Eclipse avec un ADT est fortement recommandé car ce dernier étend les capacités d'Eclipse pour permettre la configuration rapide de nouveaux projets Android, créer une interface utilisateur pour l'application, ajouter des packages basés sur l'API Framework Android, déboguer les applications en utilisant les outils SDK Android, et même exporter signer (ou non signer) des fichiers.apk afin de distribuer une application.

4. SQLite [4]

SQLite est un système de gestion de bases de données relationnelles « SGBDR » bien connu. Il est open-source, conforme aux standards, léger et mono tiers. Il a été implémenté sous la forme d'une bibliothèque C compacte incluse dans Android.

Étant implémenté sous forme de bibliothèque et non exécuté dans un processus distinct, chaque base de données SQLite fait partie intégrante de l'application qui l'a créée. Cela réduit les dépendances externes, minimise la latence et simplifie le verrouillage des transactions et la synchronisation.

SQLite a une réputation de grande fiabilité et il est le SGBDR choisi par de nombreux appareils électroniques, notamment beaucoup de lecteurs MP3 et de Smartphones.

La principale différence entre SQLite et le reste des systèmes de gestion de bases de données (PostgreSQL ou MySQL) tient au fait qu'une base de données entière avec toutes ses tables est stockée dans un seul et unique fichier. Une autre particularité est que l'accès à ce fichier de données ne nécessite aucun serveur de base de données. A cela on peut rajouter qu'il est rapide (deux fois plus que PostgreSQL et MySQL pour des opérations courantes), peu gourmand en mémoire, et peut gérer des bases de données de grandes dimensions.

Léger et puissant, il diffère des moteurs de bases de données conventionnels par son typage faible des colonnes, ce qui signifie que les valeurs d'une colonne ne doivent pas forcément être d'un seul type. Chaque valeur est typée individuellement par ligne. La conséquence en est que la vérification de type n'est pas obligatoire lors de l'affectation ou de l'extraction des valeurs des colonnes d'une ligne.

Les bases de données Android sont stockées sur le terminal (ou l'émulateur) dans le dossier **/data/data/<nom_package>/databases**. Toutes les bases de données sont privées et ne sont accessibles que par l'application qui les a créées.

Figure 3.2 : Logo SQLite.

III. Langages de programmation utilisés

1. Langage JAVA [14]

Le langage Java est un langage généraliste de programmation synthétisant les principaux langages existants lors de sa création en 1995 par Gosling et Patrick Naughton, employés de Sun Microsystems. Il permet une programmation orientée objet, modulaire et reprend une syntaxe très proche de celle du langage C.

Outre son orientation objet, le langage Java a l'avantage d'être modulaire (on peut écrire des portions de code génériques, c'est-à-dire utilisables par plusieurs applications), rigoureux (la plupart des erreurs se produisent à la compilation et non à l'exécution) et portable (un même programme compilé peut s'exécuter sur différents environnements à savoir sur plusieurs systèmes d'exploitation tels qu'UNIX, Windows, Mac OS ou Linux). En contrepartie, les applications Java ont le défaut d'être plus lentes à l'exécution que des applications programmées en C par exemple.

2. Langage XML [15]

Le langage XML « *eXtended Markup Language* » est un format général de documents orienté texte. Il s'est imposé comme un standard incontournable de l'informatique. Il est aussi bien utilisé pour le stockage de documents que pour la transmission de données entre applications. Sa simplicité, sa flexibilité et ses possibilités d'extension ont permis de l'adapter à de multiples domaines.

De nombreuses technologies se sont développées autour de XML et enrichissent ainsi son environnement. Le langage XML dérive de SGML « *Standard Generalized Markup Language* » et de HTML « *HyperText Markup Language* ».

Grâce à ce langage, il devient possible de décrire les interfaces sous la plateforme Android dans un format spécial. Celles-ci seront converties automatiquement en objets Java qui seront par la suite disponibles comme tout autre objet dans le code. XML offre ainsi plus

de souplesse de développement, facilite les modifications du code et assure la séparation entre la présentation et le comportement des objets.

3. Langage SQL [16]

Etant données que les requêtes SQLITE sont basées sur le standard SQL, une petite présentation de ce dernier s'impose.

En effet, SQL « *Structured Query Language* » est un langage de manipulation de bases de données mis au point dans les années 70 par IBM. Il permet notamment :

- La manipulation des tables : création, suppression, modification de la structure des tables ;
- La manipulation des bases de données : sélection, modification et suppression d'enregistrements ;
- La gestion des droits d'accès aux tables : contrôles des données et validation des modifications.

IV. Interfaces Homme/Machine

IV.1 Interface d'accueil de l'application

En premier lieu, l'utilisateur lance l'application en appuyant sur l'icône lui correspondant dans le menu du Smartphone.

L'application débute par le lancement de la fenêtre de la **figure 3.3**, où le propriétaire du téléphone est invité à choisir entre quatre (04) boutons :

- Le premier bouton **« Rechercher »** permet d'effectuer une recherche d'offres promotionnelles ou de vols ;
- Le deuxième bouton **« Créer Compte »** permet à l'utilisateur de créer un compte ;
- Le troisième bouton **« S'authentifier »** permet à l'utilisateur d'accéder à son espace ;
- Le dernier bouton **« Quitter »** quant à lui, permet de quitter complètement l'application.

Figure 3.3 : Interface d'accueil de l'application.

IV.2 Interface « Rechercher »

En appuyant sur le bouton « **Rechercher »** figurant sur l'interface d'accueil de l'application, l'utilisateur est encore une fois invité à choisir entre les deux (02) boutons **« Offres Promotionnelles »** et **« Recherche Rapide »** de la **figure 3.4** qui lui permettent respectivement d'effectuer une recherche d'offres promotionnelles ou de vols disponibles pour une date de départ et une date de retour données.

Figure 3.4 : Interface de Recherche.

IV.2.1 Interface « Offres Promotionnelles »

Pour le cas d'un simple utilisateur nous allons présenter les interfaces obtenues en appuyant sur le bouton **« Offres Promotionnelles »** de la **figure 3.4.**

La **figure 3.5** représente donc la liste des offres promotionnelles que propose la compagnie aérienne.

La **figure 3.6** quant à elle, représente le formulaire de recherche de vols pour une offre promotionnelle choisie.

Figure 3.5 : Liste offres**.**

Figure 3.6 : Formulaire de recherche de vols pour une offre.

En remplissant les champs de la **figure 3.6** et en cliquant sur le bouton **« Valider »** la liste des vols de la **figure 3.7** s'affiche. L'utilisateur sélectionne un vol **« Aller »** et un vol **« Retour »** et appuie sur le bouton **« Sélectionner »** pour réserver.

Il obtient la **figure 3.8** qui permet de vérifier les détails du voyage avant de confirmer la réservation en cliquant sur le bouton **« Continuer »**.

Le formulaire de réservation de la **figure 3.10** s'affiche, où l'utilisateur doit remplir tous les champs obligatoires avant d'accéder directement à la liste de ses réservations représentée par la **figure 3.11**

L'utilisateur peut aussi consulter les détails de chaque vol en cliquant sur un item et obtient donc l'interface représentée par la **figure 3.9**.

Figure 3.7 : Liste vols offres**.**

Figure 3.8 : Vérification des détails du voyage**.**

Figure 3.9 : Détails vol.

Figure 3.10 : Formulaire de réservation**.**

Figure 3.11 : Liste réservations**.**

IV.3 Interface « Créer Compte »

En appuyant sur le bouton **« Créer Compte »** figurant sur l'interface d'accueil de l'application, l'utilisateur est appelé à remplir le formulaire de la **figure 3.12** et de confirmer son inscription en appuyant sur le bouton **« Valider ».** Ce dernier envoie le client vers son espace représenté par la **figure 3.13.**

Figure 3.12 : Formulaire de création de compte.

Figure 3.13 : Espace client.

IV.4 Interface « S'authentifier»

En appuyant sur le bouton **« S'authentifier »** figurant sur l'interface d'accueil de l'application, le client est appelé à remplir le formulaire d'authentification de la **figure 3.14** et de confirmer en appuyant sur le bouton de connexion**.**

La **figure 3.15** représente le cas d'échec de connexion, où un toast s'affiche si le client saisit un nom d'utilisateur/mot de passe erroné.

Figure 3.14 : Formulaire d'authentification.

Figure 3.15 : Echec de connexion.

Le client accède à son espace lorsqu'il saisit le bon nom d'utilisateur et le bon mot de passe.

Il peut dès lors choisir de cliquer sur l'un des trois boutons qui lui sont proposés dans la **figure 3.13.**

- Le bouton **« Offres Promotionnelles »** lui permet d'effectuer, tout comme un simple utilisateur, une recherche d'offres promotionnelles. Il aura donc les mêmes interfaces représentées dans la **section IV.2.1 ;**

- Le bouton **« Recherche Rapide »** quant à lui permet d'effectuer une recherche de vols en remplissant le formulaire de la **figure 3.16**.

Figure 3.16 : Formulaire de recherche.

Le client obtient une liste de vols correspondant à sa requête semblable à celle représentée par la **figure 3.7** où il pourra vérifier les détails du voyage avant de confirmer sa réservation **(figure 3.8)** et consulter les détails de chaque vol (**figure 3.9)**.

Le premier bouton **« Gérer réservations »** permet de consulter la liste des réservations effectuées **(figure 3.17)**.

Figure 3.17 : Liste réservations.

Figure 3.18 : Demande de confirmation d'annulation de réservation.

Le client peut complètement annuler sa réservation, consulter la liste des personnes dans le cas d'une réservation en groupe ou encore changer la date de sa réservation en choisissant un vol parmi une liste de vols proposés.

En sélectionnant une réservation et en cliquant sur le bouton **« Annuler »** le client obtient l'interface représentée par la **figure 3.18** où une boite de dialogue s'affiche pour confirmer son choix.

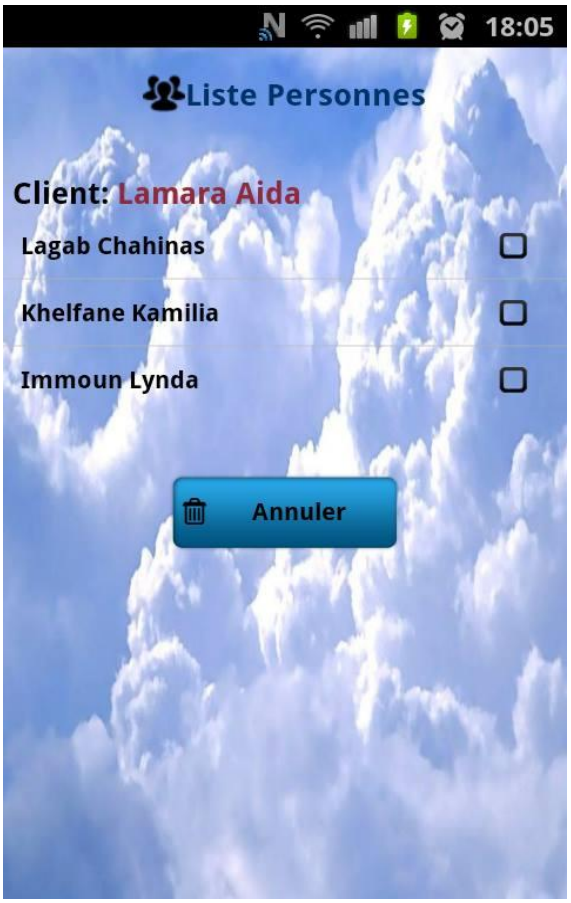


Figure 3.19 : Liste personnes

Figure 3.20 : Demande de confirmation d'annulation de vol pour une personne choisie.

En sélectionnant une réservation et en appuyant sur le bouton **« Personne »** de la **figure 3.17,** la liste de personnes de la **figure 3.19** s'affiche.

Le client peut annuler la réservation d'une personne en cliquant sur le bouton annuler où une boite de dialogue s'affiche pour confirmer son choix (**figure 3.20)**.

Figure 3.21 : Liste vols (échanger).

Figure 3.22 : Demande de confirmation de changement de réservation.

En sélectionnant une réservation et en appuyant sur le bouton **« Echanger »** la liste des vols de la **figure 3.21** s'affiche où le client choisit un vol parmi ceux qui lui sont proposés. La boite de dialogue de la **figure 3.22** lui est affichée pour confirmer son choix.

Conclusion

Ce dernier chapitre a été consacré à la présentation de l'environnement d'implémentation et de développement. Nous avons présenté les différents outils et logiciels qui nous ont permis de réaliser notre application, à savoir : les langages de programmation et les logiciels de création utilisés.

Pour conclure, nous avons présenté quelques interfaces d'utilisation de notre application.

Conclusion Générale

Au cours de ce projet, nous avons développé une application dédiée aux terminaux mobiles disposant de la plateforme « Android ». Celle-ci permet au propriétaire du Smartphone de consulter et d'effectuer une recherche de vols ou d'offres promotionnelles proposés par une agence de voyage (compagnie aérienne) elle doit donc implémenter les outils nécessaires pour couvrir les besoins des différents acteurs du système.

Pour se faire, nous avons eu recours à différentes technologies et outils d'analyse, de conception et de programmation jugés nécessaires pour aboutir à l'objectif de notre application.

Pour une réalisation réussie du travail demandé, nous avons suivi les étapes suivantes :

- ✓ Dans un premier lieu, nous avons fait une étude théorique sur les notions de base et les technologies employées, à savoir : la plateforme Android, son historique, ses versions et ses différentes caractéristiques ;

- ✓ Nous avons par la suite entamé l'analyse, la spécification des besoins et la conception en passant par une brève présentation du langage de modélisation UML qui nous a servi d'appui pour bien concevoir notre application grâce à la liste des diagrammes qu'il propose notamment le diagramme de cas d'utilisation, de séquence et de classes ;

- ✓ Enfin, nous avons décrit, depuis le chapitre de réalisation, l'implémentation de notre application tout en présentant quelques captures d'écran de certaines interfaces de celle-ci.

Ce travail nous a donné l'occasion de nous initier aux différentes étapes à suivre pour la réalisation d'un projet informatique. Il nous a aussi permis d'enrichir notre savoir et de développer nos connaissances, notamment dans le domaine de la programmation.

En effet, l'application a exigé des connaissances du langage JAVA et du langage XML pour sa partie « réalisation » et aussi du langage de modélisation UML pour sa partie « d'analyse et de conception ». La mise en œuvre de notre travail a exigé des connaissances très approfondies en la matière ainsi qu'une bonne maitrise de la configuration d'Eclipse et de l'environnement Android.

Cependant, comme toute autre application Android, et malgré les diverses fonctionnalités qu'elle offre, notre application peut être aisément améliorée pour mieux répondre aux besoins réels de l'utilisateur. En effet, grâce à son aspect ouvert, Android offre l'opportunité de créer des logiciels mobiles innovants et révolutionnaires en encourageant les développeurs à puiser dans leur imagination et à mobiliser toutes leurs compétences pour un meilleur de cette plateforme.

Bibliographie

[1] Sébastien PEROCHON, « Android Guide de développement d'applications pour Smartphones et Tablettes », édition ENI juillet 2011.

[2] Damien GUIGNARD, Julien CHABLE, Emmanuel ROBLES Avec la contribution de Nicolas SOREL et Vanessa CONCHODON, « Programmation Android De la conception au déploiement avec le SDK Google Android 2 », édition EYROLLES.

[3] Alain MENU, « Android Howto : Présentation générale », édition 2.1, septembre 2013.

[4] Reto MEIER, « Android 4 Développement d'applications avancées », édition PEARSON 2012.

[5] Florent GARIN, « Développer des applications mobiles pour les Google Phones», édition DUNOD, 2009.

[6] Mark L. MURPHY, « L'art du développement Android » édition PEARSON, 2009.

[7] Reto MEIER , « Développement d'applications professionnelles avec Android 2 », édition PEARSON, 2010.

[8] http://fr.wikipedia.org

[9] Pascal ROQUES, « Les cahiers du programmeur UML2 modéliser une application Web », 4éme édition EYROLLES.

[10] Joseph GABAY et David GABAY, « UML2 Analyse et Conception Mise en œuvre guidée avec études de cas », DUNOD 2008.

[11] Jim CONALLE, « Building Web Applications with UML», Second Edition Addison Wesley, Octobre 2002.

[12] Olivier SIGAUD, « Introduction à la modélisation orientée objets avec UML », Edition 2005-2006.

[13] Laurent AUDIBERT, « UML 2 de l'apprentissage à la pratique », 2ème édition ELLIPSES ,2014.

[14] Claude DELANNOY, « Programmer en Java », 3ème édition EYROLLES, 2006.

[15] Alain MICHARD, « XML: langage et applications », édition EYROLLES, 2000.

[16] Eric GODOC, « SQL, les fondamentaux du langage », 2ème édition ENI, 2014.