Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Travail de Bachelor 2008

Filière Informatique de gestion

Google Android

Etudiant : Joël Salamin

Professeur : Yann Bocchi

www.hevs.ch

Chapitres

Annexes

Bachelor | 08

Introduction

Chapitre 1

Google Android

Joël Salamin

Hes·so VALAIS WALLIS

Table des matières

1. Présentation d'Android

En août 2005, le géant Google rachète une entreprise spécialisée dans le développement d'applications mobiles. Deux ans plus tard, le 15 novembre 2007, Google annonce officiellement la sortie de son système d'exploitation mobile baptisé Android. C'est également à cette date qu'est fondée la Open Handset Alliance, regroupant une trentaine de partenaires tels que HTC, Samsung, Motorola et LG avec comme but de promouvoir ce nouveau système d'exploitation open source.

Le géant du référencement indique clairement ses intentions de bouleverser le monde de la téléphonie mobile où dominent actuellement Nokia avec Symbian et Microsoft avec Windows Mobile.

Android apparaît comme un concurrent à ces systèmes d'exploitation, le défi est de taille pour Google qui n'en est pas à son coup d'essai dans la conquête d'un nouveau marché. La question est de savoir ce que propose réellement le système Android ; ce qu'il a de plus que les autres et ce qui fera qu'un développeur ou un utilisateur se tournera vers le système de Google plutôt que vers un autre système ayant déjà fait ses preuves.

En plus de cela, aucun appareil équipé d'Android n'est encore commercialisé. Il y a donc de nombreuses inconnues dans l'équation et c'est avec l'objectif de cerner les enjeux et répondre à ces différentes questions que ce Travail de Bachelor est réalisé.

Avant de rentrer dans le détail de ce système, il me parait judicieux de faire un tour d'horizon du marché sur lequel il se trouve ainsi que d'observer ses différents concurrents.

Bref aperçu du monde des systèmes d'exploitation mobiles

Le marché des systèmes d'exploitation mobile est dominé par Symbian de Nokia (65%), Windows Mobile de Microsoft (12%) et RIM BlackBerry (11%). Le reste du marché se partage entre iPhone de Apple, Linux, PalmOS, etc…

Pour avoir une vision complète du marché, il est important de citer les différents projets de plateforme mobile linux qui se sont d'ailleurs pour une partie regroupés au sein de la LiMo Foundation. Il y a également OpenMoko qui est le tout premier Smartphone équipé d'un système d'exploitation complètement libre. La communauté linux mobile propose ainsi une alternative aux systèmes propriétaires actuels.

Le dernier acteur à ne pas négliger se nomme Apple avec son iPhone qui a apporté un regard complètement différent sur l'expérience que peut apporter un téléphone mobile à l'utilisateur.

En marge de ces grands noms de la téléphonie mobile, il faut également tenir compte de RIM BlackBerry qui est très présent dans le secteur professionnel et Palm produisant ses téléphones équipés de son système d'exploitation propriétaire Palm OS.

Depuis quelques mois, les annonces fracassantes animent le secteur de la téléphonie mobile, à commencer par Nokia qui rachète la totalité de Symbian et a fondé par la même occasion la Symbian Foundation regroupant de nombreux grands noms de la téléphonie.

D'ici fin 2008/début 2009, Android va faire son apparition sur ce marché. Il est évident que tous les acteurs présents appréhendent cette arrivée.

Exemple de système d'exploitation mobile

Avant de se concentrer sur le système de Google, il est intéressant de voir ce qui se fait actuellement sur le marché et de remarquer la diversité qui existe dans le monde du mobile.

Les différentes interfaces graphiques suivantes font également ressortir certaines informations sur les choix pris par les différents acteurs dans la réalisation de leur système.

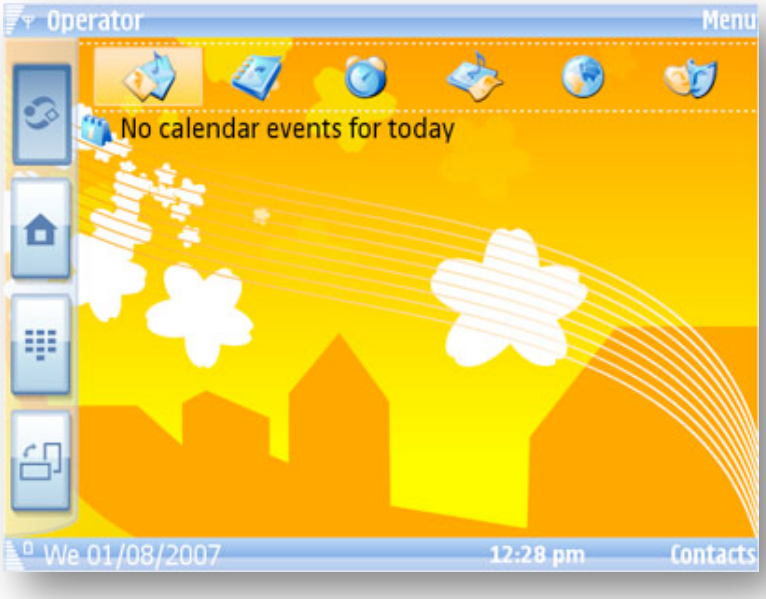

Figure 1 - Symbian S60 5th edition

Nokia proposera prochainement la nouvelle version de Symbian S60 (version la plus courante du système) prévue pour fonctionner avec le doigt ou un stylet.

Figure 2 - Windows Mobile 6.1

Microsoft propose son système Windows Mobile 6.1 qui a vu le jour en avril 2008, mais la sortie de la version 7 est agendée à début 2009 et s'annonce comme étant complètement remanié par rapport aux versions précédentes.

Figure 3 - BlackBerry OS version 4.7

Destiné principalement au monde professionnel, le système BlackBerry OS version 4.7 est le dernier né de RIM annoncé pour une commercialisation fin 2008.

Figure 4 - iPhone OS

Apple avec son iPhone équipé du système iPhone OS a déjà fait coulé beaucoup d'encre depuis son arrivée sur le marché en 2007.

2. Présentation du travail de Bachelor

Ce travail gravite autour de la nouvelle plateforme mobile proposée par Google. Les ambitions de ce travail de recherche sont :

- Fournir une vision globale de l'environnement Android afin d'en comprendre au mieux les enjeux
- Travailler avec l'environnement de développement pour déboucher sur la création d'une application fonctionnelle ayant pour but de se former sur l'outil, en mesurer le potentiel et en ressortir les forces et faiblesses

C'est pourquoi plusieurs axes sont définis :

- WP1 - Analyse et recherche
- WP2 - Développement d'une application Android
- WP3 - Analyse basée sur le vécu et prise de position sur l'outil

WP1 - Analyse et recherche

Cette première phase regroupe plusieurs points essentiels du projet. Tout d'abord, une période de découverte d'Android se fait, cette découverte se focalise dans un premier temps sur les aspects non techniques. L'objectif étant de faire connaissance avec la communauté Android, l'actualité, la vision de Google, les points forts de cette plateforme, etc…

Une fois cette première approche effectuée, il est nécessaire de faire une approche plus technique afin d'évaluer les capacités de cette plateforme et des possibilités qu'elle offre. Le délivrable de cette étape sera la proposition d'un scénario de prototype. Il est

important que le prototype qui sera développé couvre un maximum de fonctionnalités d'Android afin d'avoir la connaissance la plus large possible de cet outil.

WP2 - Développement d'une application Android

Cette phase représente le noyau du projet, elle se basera sur le cahier des charges issu de la première phase. La découverte du framework est également comprise dans la phase de développement.

Le délivrable est un prototype fonctionnel implémentant les éléments de priorité 1 du cahier des charges de l'application et les différents autres éléments selon l'état d'avancement permis par le temps à disposition.

WP3 - Analyse basée sur le vécu et prise de position sur l'outil

Cette phase s'inscrit dans le prolongement du développement du prototype, car celui-ci permettra de se confronter au framework de Google Android. Il est important de faire tout d'abord une analyse de l'outil et du déroulement d'un développement sur Android. Une fois cette analyse faite, il est intéressant d'avoir un regard critique sur cet environnement de développement en mettant l'accent sur les points forts et points faibles de celui-ci et les différentes expériences acquises tout au long du développement.

En conclusion pour cette phase, il est pertinent de prendre position sur Android, partager le vécu et l'expérience acquise.

3. Motivation personnelle

Le choix du thème pour ce travail de Bachelor s'est fait de manière assez simple étant donné l'actualité dans le monde de l'informatique dans laquelle Android est omniprésent, et mon grand intérêt pour le monde de la téléphonie mobile.

Le thème a été choisi pour les raisons suivantes :

- Recherche et analyse
- Nouvelle technologie
- Plateforme du futur
- Comparaison à un environnement connu

Recherche et analyse

La recherche représente une grande partie du projet et un grand travail d'analyse est nécessaire pour déboucher sur une phase de développement. Cette phase de recherche permet d'apprendre énormément sur un domaine que je ne connais encore que trop peu.

L'idée de découvrir quelque chose de complètement nouveau et qui est au cœur de l'actualité a fortement influencé mon choix.

Nouvelle technologie

Le monde de la téléphonie est en ébullition constante depuis quelques années et Android représente une révolution annoncée pour l'année 2008 suite à celle amenée par Apple en 2007 avec l'annonce de l'iPhone.

Ce travail permet de faire connaissance avec un environnement complètement nouveau et qui tend à s'imposer comme un leader dans les prochaines années selon certains analystes.

Le sentiment de travailler en étroite dépendance avec l'actualité est une expérience passionnante.

Plateforme du futur

Android est déjà annoncé par de nombreux professionnels comme étant une technologie et une plateforme vouée à s'imposer dans un avenir relativement proche.

Le fait que ce travail m'apporte des connaissances et des compétences qui représentent un atout non négligeable pour l'avenir est source d'une motivation forte et une envie supplémentaire de s'investir complètement.

Comparaison à un environnement connu

Le fait d'avoir une certaine expérience dans le développement sur Windows Mobile me pousse à élargir mon expérience en développement mobile et surtout mon esprit critique.

4. Structure du rapport

La suite du rapport se compose des différentes étapes qui ont marqué le déroulement du travail. Les chapitres sont organisés de manière à respecter une certaine logique dans la lecture :

- Cahier des charges de l'application
- Android SDK – Développement
- Android SDK – Retour d'expérience
- Android – Les enjeux
- Conclusion générale
- Bibliographie

A cela viennent s'ajouter des annexes permettant de profiter d'un complément d'informations relatives au prototype développé :

- Manuel de l'utilisateur
- Manuel technique « Naviboo – RoadBook »

Cahier des charges de l'application

Ce chapitre découle d'une première phase de découverte et de recherche. L'objectif de ce cahier des charges est de permettre le développement d'un prototype exploitant au mieux l'outil proposé par Google.

Android SDK – Développement

Cette partie du rapport revient sur toute la phase de développement de l'application sur la base du cahier des charges établi. Toutes les grandes étapes du développement y sont décrites, les problèmes rencontrés ainsi que les choix pris dans le cadre du projet.

La phase de développement apporte une bonne expérience autant dans la prise en main de l'outil que dans son utilisation dans tout le processus de création d'une application.

De cette expérience acquise va naître le chapitre suivant, soit le retour d'expérience.

Android – Retour d'expérience

Dans la continuité du développement, ce chapitre décrit les grands axes techniques de l'environnement Android sur la base de l'expérience acquise et des informations collectées. Le but est de fournir une vue complète du SDK et d'en comprendre les concepts, les points forts et les faiblesses.

Android – Les enjeux

Ce chapitre prend de la distance par rapport à tout l'aspect développement, le but étant d'avoir une vision plus générale sur les enjeux et la position de Google par rapport à la concurrence.

Le monde de la téléphonie mobile évolue et la question est de savoir à quelle place Google peut prétendre avec son système d'exploitation.

Conclusion générale

Ce chapitre permet de tirer un bilan du travail effectué. Il représente la synthèse du rapport avec une description du déroulement du projet ainsi que le bilan personnel sur cette expérience.

Manuel de l'utilisateur

Description fonctionnelle de l'application et modalité d'usage.

Manuel technique « Naviboo – RoadBook »

Document technique décrivant l'application réalisée dans son ensemble avec le détail de chaque composant.

5. Tableau des illustrations

Bachelor | 08

Cahier des charges de l'application

Chapitre 2

Google Android

Joël Salamin

Hes·so VALAIS WALLIS

Table des matières

1. Contexte

L'objectif de cette application est de couvrir le maximum des fonctionnalités disponibles sur Android. Suite à une phase d'analyse nécessaire à l'établissement d'un cahier des charges pertinent et réalisable, les fonctionnalités suivantes sont retenues :

- Google Maps
- Base de données
- Envoi de Mail
- Consommation de WebService
- Caméra/Photo (si possible)

Pour couvrir ces fonctionnalités, l'application se présente sous forme d'un guide permettant de créer/charger un parcours composé de Points. Ce parcours est affiché à l'aide de Google Maps afin que l'utilisateur profite d'une visite adaptée à ses désirs. En plus de ce guide, l'application permet de publier son RoadBook sur un Blog pour que tout le monde puisse suivre chaque étape importante du voyage.

Tout cela est parfaitement intégré à l'application et permet à l'utilisateur de profiter pleinement de toutes ces fonctionnalités de manière intuitive et simple.

2. Description

L'application mobile créée permet à l'utilisateur de profiter d'un outil de visite touristique dynamique et interactif. Il sera alors possible de profiter pleinement de la découverte d'une région tout en profitant d'une indépendance complète.

A ce guide touristique de poche vient s'ajouter la possibilité de tenir un carnet de route de son voyage/sa visite. Grâce à cela, l'utilisateur pourra à tout moment prendre une photo d'un lieu, d'un monument ou simplement d'un paysage et y ajouter un commentaire afin d'avoir au terme de son voyage un album souvenir composé de nombreuses informations sur celui-ci. Les informations ainsi récoltées lui permettront de revivre son voyage et de le faire vivre à ses amis grâce à la possibilité de publication du carnet sur un Blog.

3. Cas d'utilisation

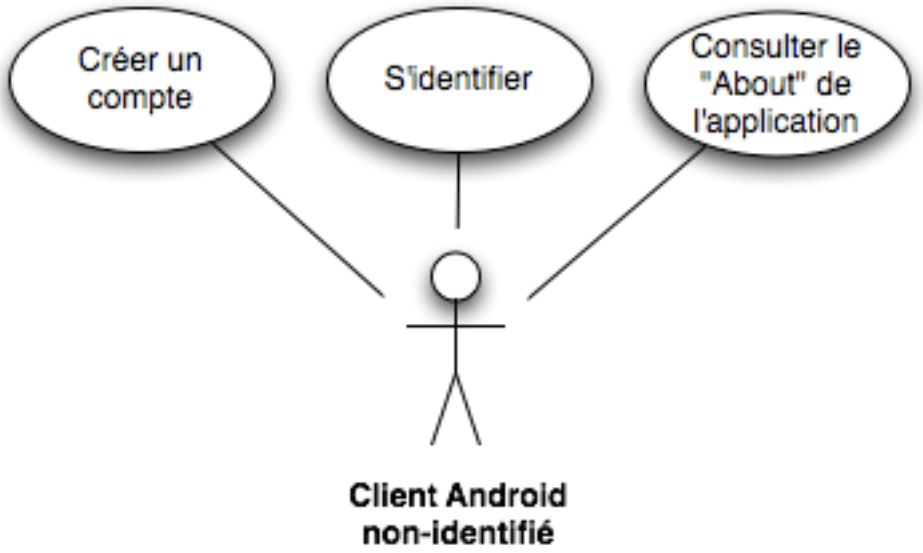

Figure 1 - Cas d'utilisation : Client Android non identifié

Client Android non-identifié

Client Android qui n'est pas encore identifié ou enregistré.

Créer un compte

Si l'utilisateur ne possède pas déjà un compte, il lui est possible de s'enregistrer afin de créer un nouveau compte qui sera utilisable dès la fin du processus d'inscription.

S'identifier

Le client possédant déjà un compte a la possibilité de s'identifier en saisissant son login et password.

Consulter le « About » de l'application

L'utilisateur peut obtenir des informations au sujet de l'application en cliquant sur le bouton "About" de celle-ci.

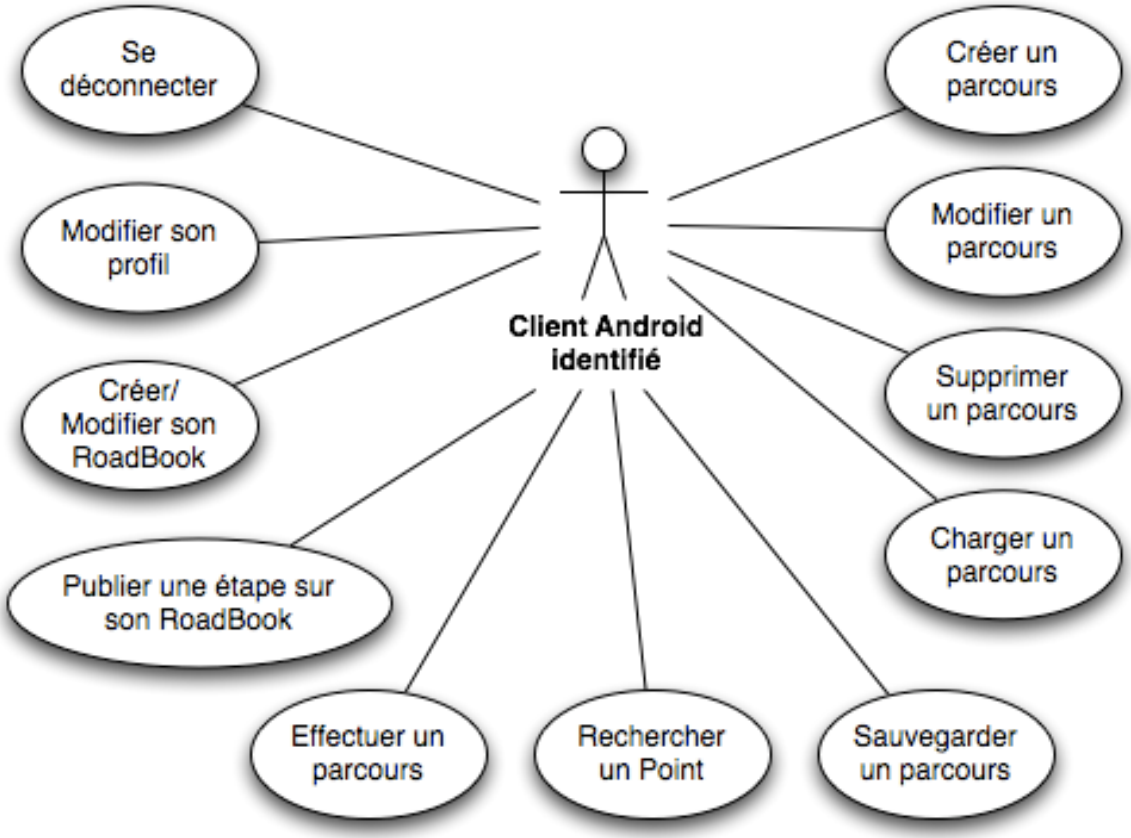

Figure 2 - Cas d'utilisation : Client Android identifié

Client Android identifié
Client Android qui s'est identifié à l'aide de son login/password.

Se déconnecter
L'utilisateur identifié peut se déconnecter de l'application à tout moment.

Modifier son profil
Le client identifié peut modifier son profil à tout moment depuis le menu principal.

Créer/Modifier son RoadBook
Le client peut créer et modifier son RoadBook. Pour que le RoadBook soit utilisable, le client doit saisir un nom, une description et un compte Mail-to-Blogger.
Le RoadBook contient les données nécessaires à la publication d'articles sur son Blog.

Créer un parcours
Le client peut créer un nouveau parcours selon ses choix.

Modifier un parcours
Le client pourra modifier un parcours avant de le charger ou lors de sa création. Il peut ajouter/supprimer des Points à ce parcours avant de le valider.

Supprimer un parcours
Le client peut supprimer un parcours de sa liste de parcours sauvegardés.

Charger un parcours

Le client peut charger un parcours de sa liste de parcours sauvegardés. Lors du chargement du parcours il lui est possible de le modifier ou de le garder dans l'état.

Sauvegarder un parcours

Lorsqu'un nouveau parcours est créé ou que le client souhaite interrompre le parcours en cours, il a la possibilité de sauvegarder celui-ci dans sa liste de parcours afin de pouvoir le charger ultérieurement.

Rechercher un Point

Il est possible de rechercher un Point selon une chaîne de caractère.

Effectuer un parcours

Une fois le parcours créé/chargé, le client peut démarrer sa visite grâce à la carte et les différents Points indiqués sur celle-ci.

Publier une étape sur son RoadBook

A tout moment lors de la visite, le client peut (s'il a configuré son RoadBook) publier une étape de son parcours sur celui-ci.
Il lui est possible de saisir un nom, une description, une photo et une note et à cette étape il peut attacher certains de ces contacts (s'il est accompagné dans son voyage).
Une fois l'étape validée, l'article sera publié sur son Blog afin que tout le monde puisse le consulter et suivre son voyage.

4. Processus

Identification

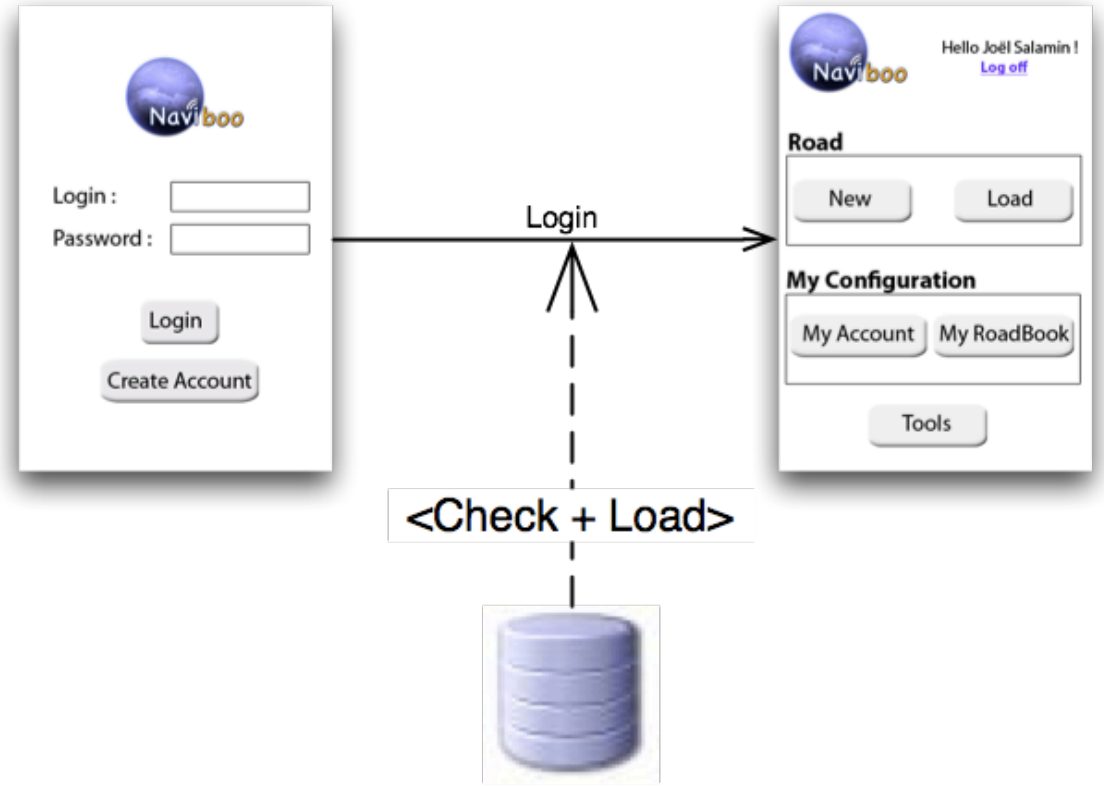

Figure 3 - Processus : Identification

Pré condition

L'utilisateur doit avoir créé un compte.

Description

L'utilisateur est invité à saisir son identifiant et mot de passe lorsque l'application est démarrée.

Une fois que l'utilisateur a fait sa saisie et clique sur « Login », une vérification est faite auprès de la base de données pour s'assurer que les données saisies sont correctes. Si la vérification est faite avec succès, l'utilisateur est alors identifié et se retrouvera sur le menu de l'application.

Post condition

L'utilisateur est identifié et peut profiter de l'application.

Inscription/Création d'un compte

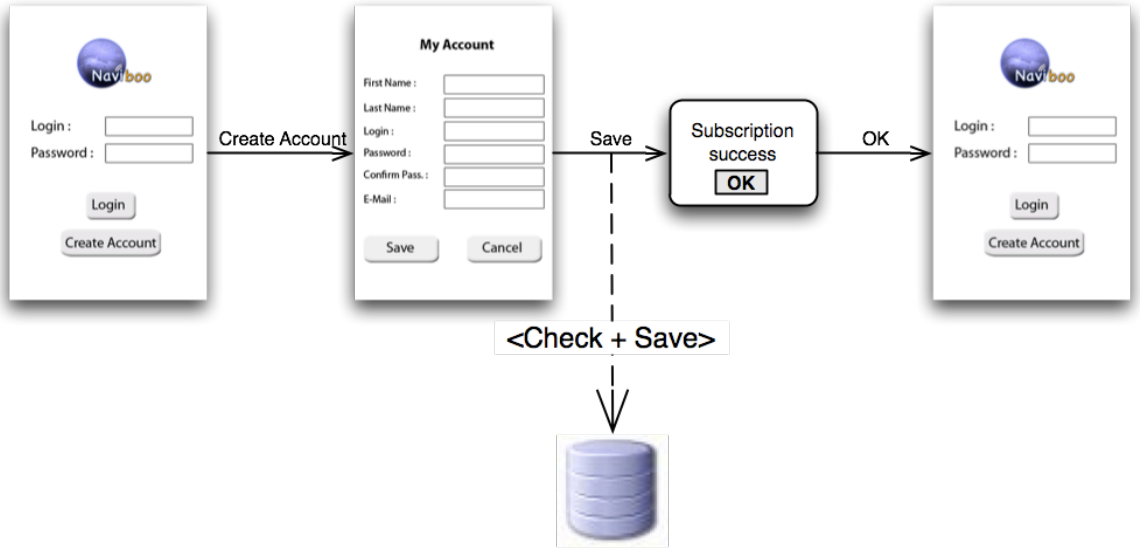

Figure 4 - Processus : Inscription / Création d'un compte

Pré condition

L'utilisateur n'a pas encore créé de compte et ne peut donc pas s'identifier.

Description

L'utilisateur ne possédant pas de compte peut en créer un simplement en cliquant sur « Create Account ». Il est alors invité à saisir les différentes informations relatives à son compte et finaliser l'inscription en cliquant sur « Save ».

Toutes les informations sont obligatoires et la saisie est vérifiée avant de stocker les informations dans la base de données.

Une fois le compte créé et les données persistées en base de données, un message d'information s'affiche pour confirmer à l'utilisateur que l'inscription s'est déroulée avec succès.

Post condition

Le compte est sauvegardé en base de données et l'utilisateur peut s'identifier sur l'application.

Modifier son compte

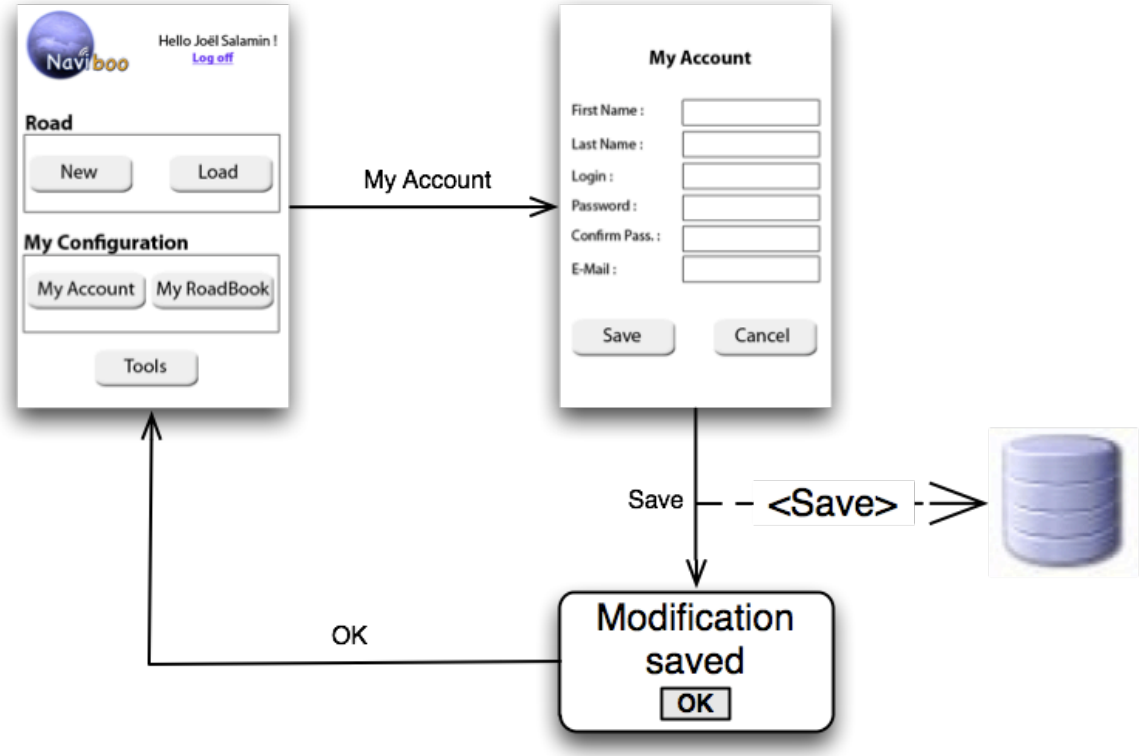

Figure 5 - Processus : Modifier son compte

Pré condition

L'utilisateur doit avoir un compte et être identifié.

Description

Depuis le menu, l'utilisateur peut à tout moment modifier son compte. Le seul élément de son compte qu'il ne peut pas modifier est son identifiant.

Une fois la modification faite, il lui suffit de cliquer sur « Save » et toutes les informations saisies sont vérifiées et sauvegardées dans la base de données.

Lorsque les modifications ont été correctement persistées en base de données, un message d'information s'affiche pour confirmer que la modification du compte s'est déroulée avec succès.

Post condition

Les modifications apportées au compte de l'utilisateur sont sauvegardées dans la base de données.

Consulter le « A propos »

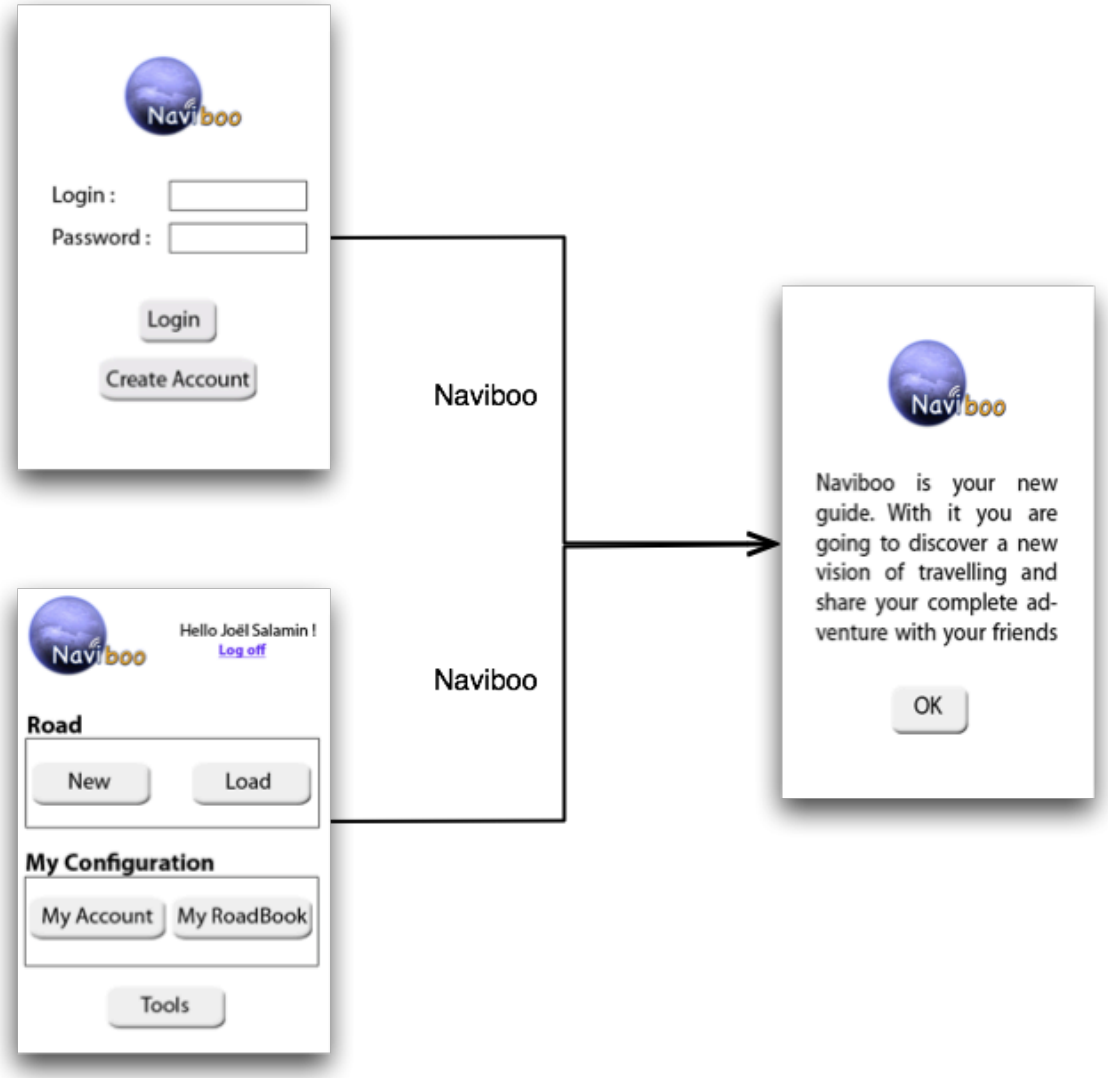

Figure 6 - Processus : Consulter le "A propos"

Pré condition

Aucune.

Description

L'utilisateur peut consulter les informations concernant l'application simplement en cliquant sur le logo Naviboo présent sur l'écran d'accueil ainsi que sur l'écran du menu.

Post condition

L'utilisateur possède toutes les informations concernant l'application.

Création d'un parcours

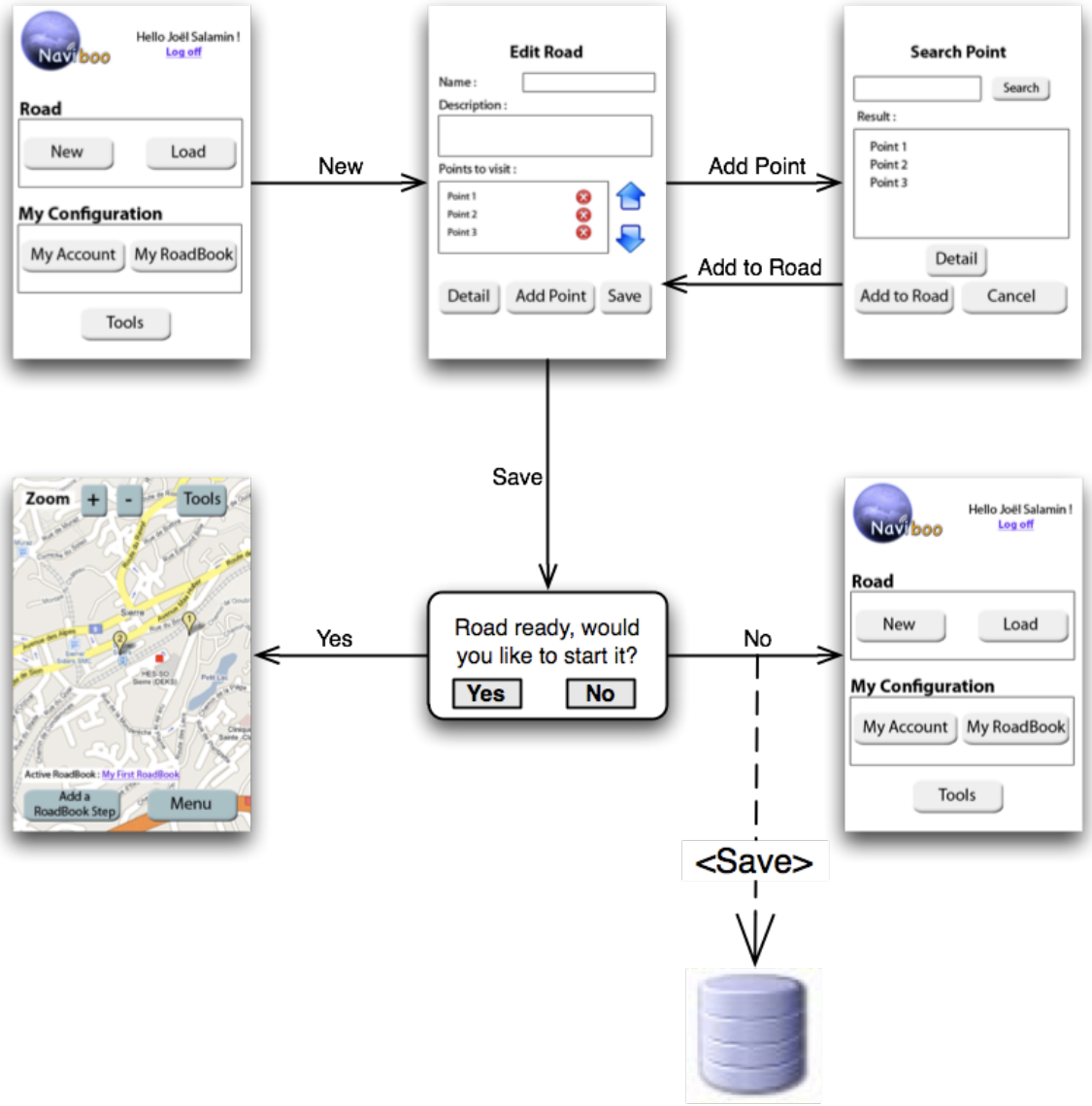

Figure 7 - Processus : Création d'un parcours

Pré condition

L'utilisateur doit avoir un compte et être identifié.

Description

L'utilisateur peut créer un nouveau parcours en cliquant sur « New » dans le menu. Il lui est alors demandé de saisir un nom pour le nouveau parcours ainsi qu'une description.

Un parcours se compose de points à visiter, l'utilisateur peut en ajouter autant qu'il le souhaite grâce au bouton « Add Point » qui le dirige vers un outil de recherche de points. L'ordre des points est important dans l'éditeur de parcours car il définit l'ordre de la visite. Une fois la liste des points complétée selon les envies de l'utilisateur, il suffit de cliquer sur « Save » pour terminer le processus.

Une fois le parcours créé, un message d'information s'affiche pour demander à l'utilisateur s'il souhaite démarrer sa visite ou non. Si oui, le parcours est chargé et la carte s'affiche ; si non, le parcours est sauvegardé en base de données afin de pouvoir être chargé ultérieurement et l'utilisateur est redirigé vers le menu principal.

Post condition

Le parcours créé est soit sauvegardé en base de données pour être utilisé ultérieurement, soit chargé pour que l'utilisateur commence sa visite.

Affichage du détail d'un point

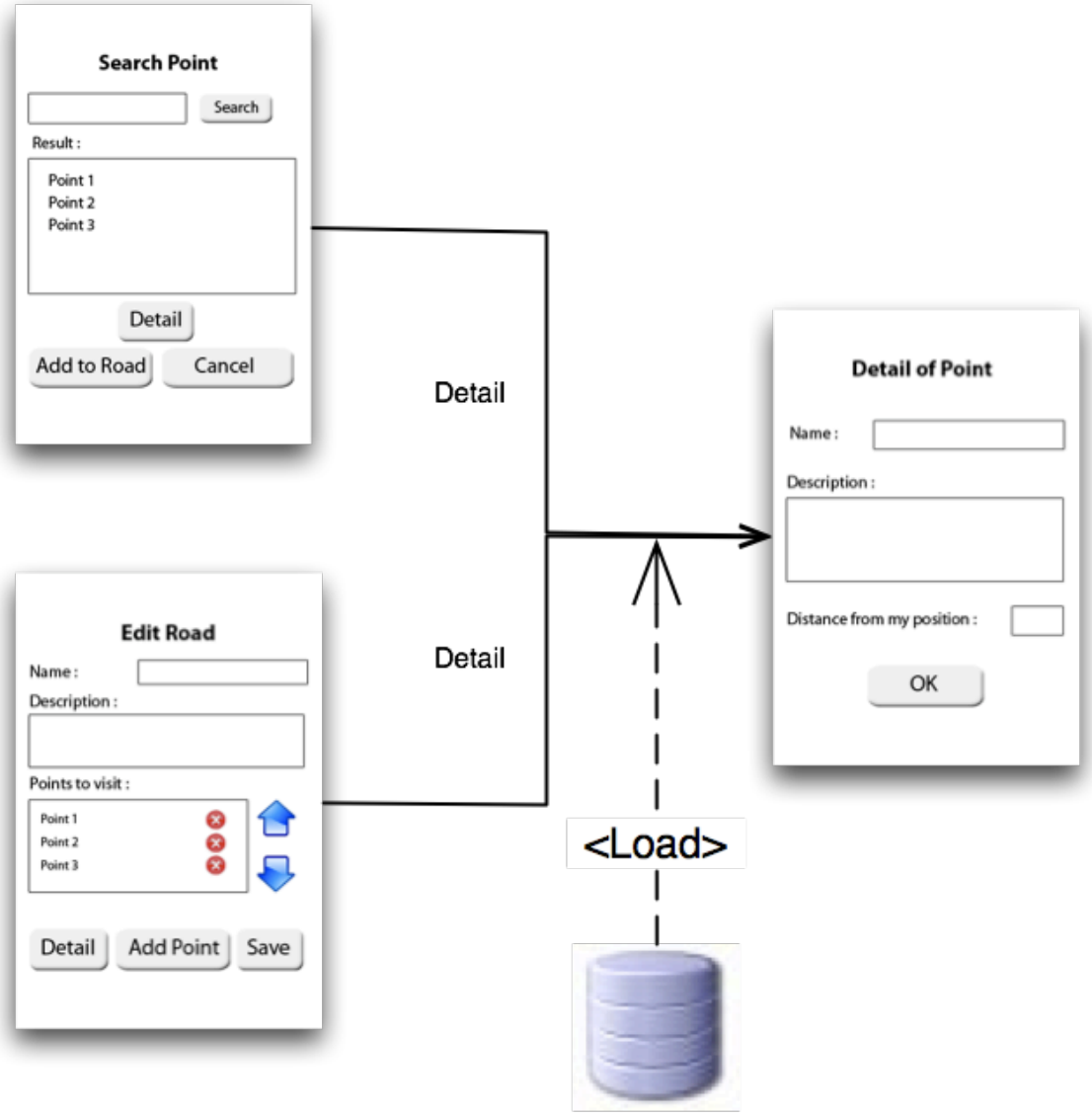

Figure 8 - Processus : Affichage du détail d'un point

Pré condition

L'utilisateur doit avoir un compte et être identifié.

Description

L'utilisateur peut à tout moment consulter le détail d'un point afin d'en connaître la description exacte. Pour ce faire, il lui suffit de sélectionner le point désiré puis de cliquer sur « Detail ».

Les informations relatives au point sélectionné sont chargées depuis la base de données afin d'être affichées sur l'écran.

Post condition

L'utilisateur a toutes les informations concernant un point.

Charger un parcours

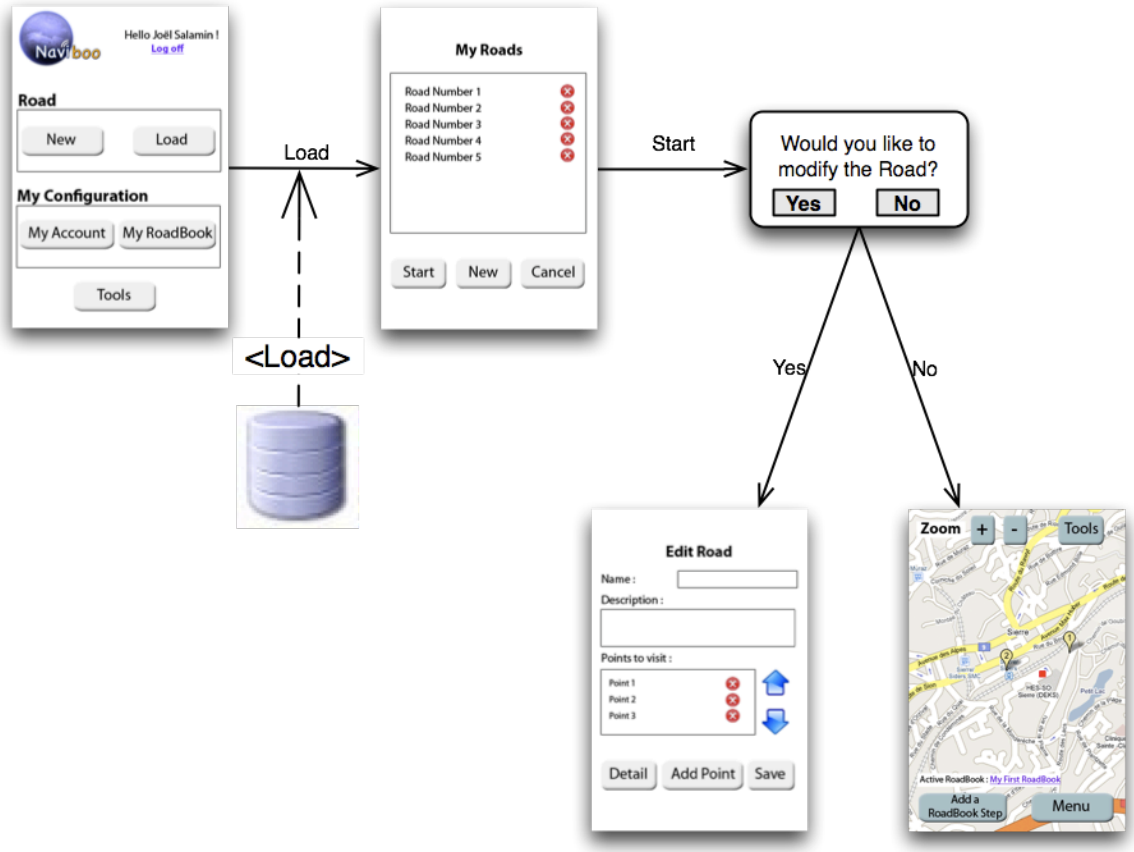

Figure 9 - Processus : Charger un parcours

Pré condition

L'utilisateur doit avoir un compte et être identifié.

Description

L'utilisateur peut charger un parcours qu'il a sauvegardé. Pour se faire, il lui suffit de cliquer sur « Load » du menu principal. Les différents parcours sauvegardés par l'utilisateur sont chargés depuis la base de données afin d'être affichés.

Il est alors possible de supprimer, créer ou démarrer un parcours. Pour cela il faut le sélectionner dans la liste et cliquer sur « Start », un message d'information s'affiche pour demander à l'utilisateur s'il souhaite modifier ou non le parcours avant de commencer la visite.

Post condition

Le parcours choisi est chargé et la visite peut commencer.

Configurer le RoadBook

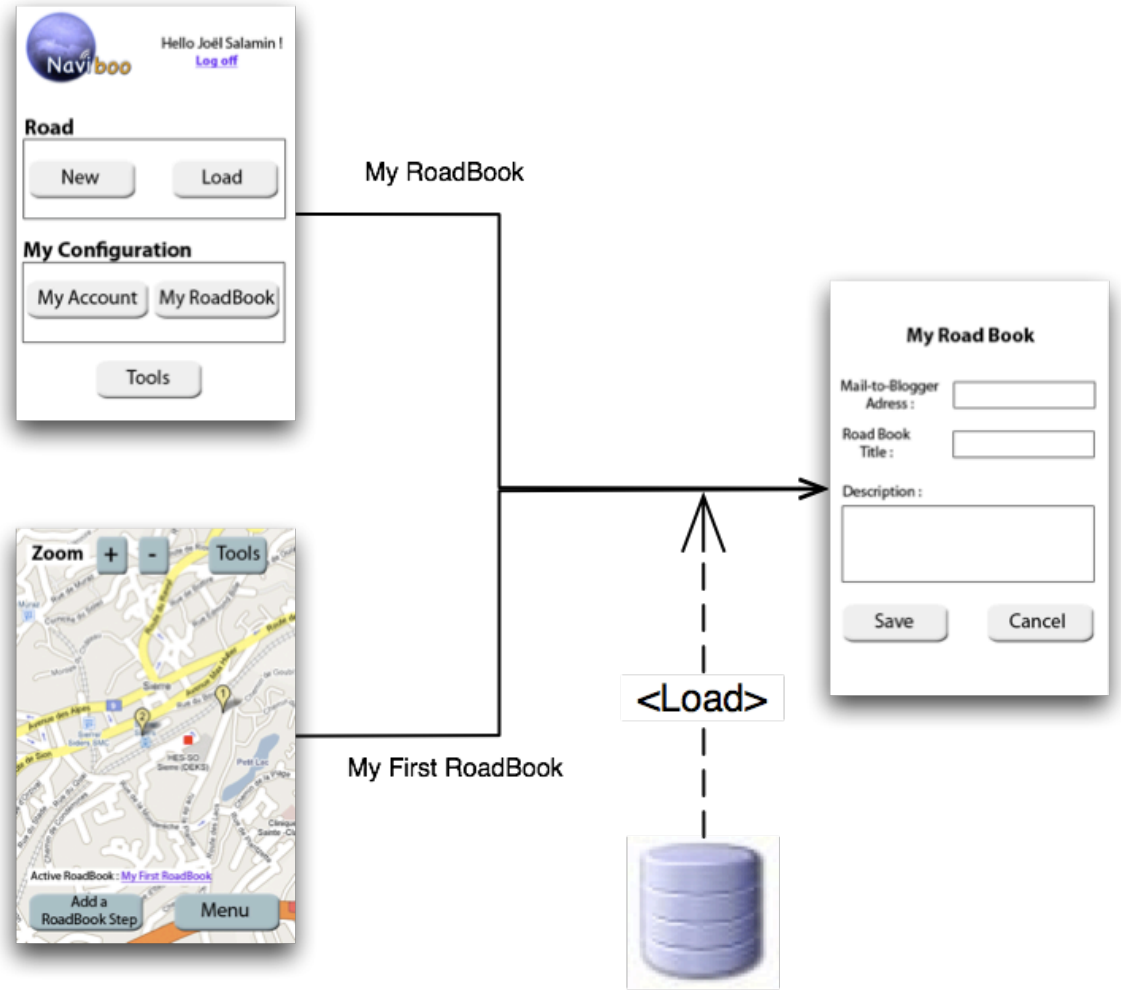

Figure 10 - Processus : Configurer le RoadBook

Pré condition

L'utilisateur doit avoir un compte, être identifié et posséder un compte Mail-to-Blogger.

Description

L'utilisateur doit configurer son RoadBook afin de pouvoir l'utiliser lors de la visite. Pour cela il a deux possibilités, soit depuis le menu en cliquant sur « My RoadBook », soit directement depuis la visite en cours en cliquant sur « Active RoadBook : … ».

Les données sont récupérées depuis la base de données et une fois la création/modification effectuée, les informations sont sauvegardées dans la base de données.

Post condition

Le RoadBook est sauvegardé en base de données et est utilisable lors du parcours.

Ajouter une étape au RoadBook

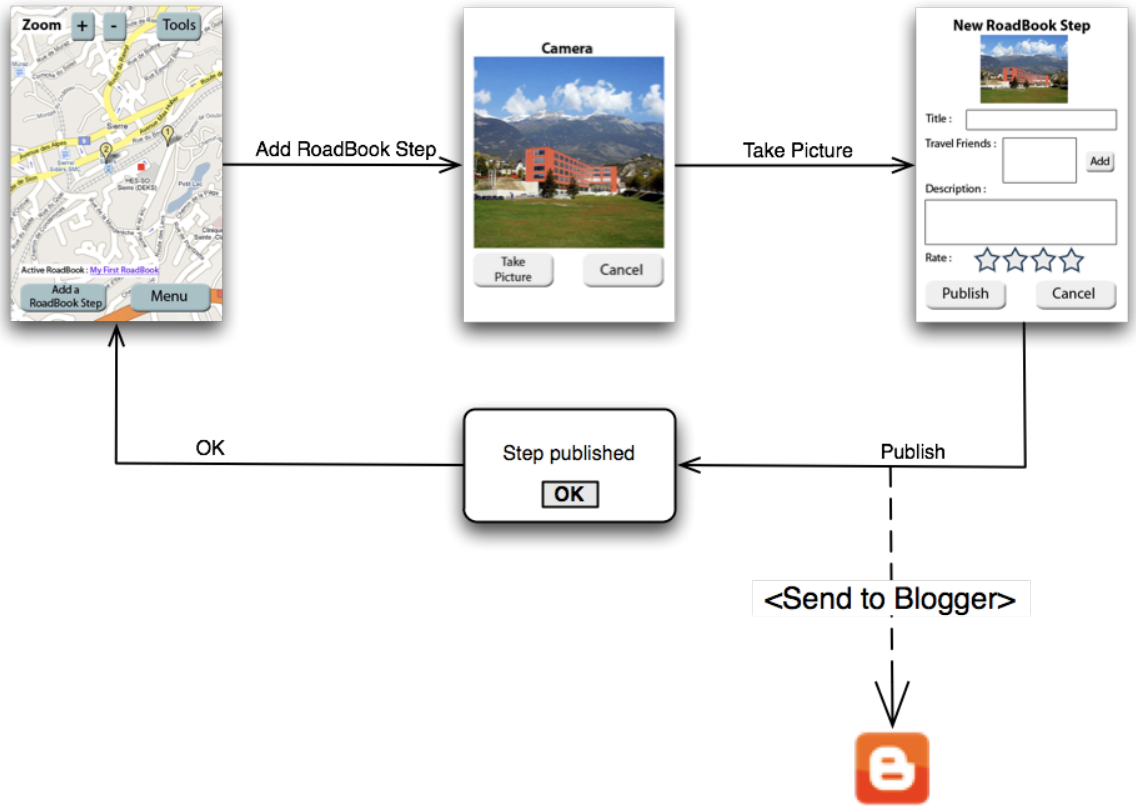

Figure 11 - Processus : Ajouter une étape au RoadBook

Pré condition

L'utilisateur doit avoir un compte, être identifié et avoir configuré correctement son RoadBook.

Description

L'utilisateur peut à tout moment, lors de sa visite, ajouter une étape à son RoadBook actif en cliquant sur « Add RoadBook Step ». Il peut alors prendre une photo et saisir les informations relatives à cette nouvelle étape.

Une fois toutes les informations saisies, il lui suffit de cliquer sur « Publish » pour publier l'étape sur Blogger.

Un message d'information s'affiche pour indiquer à l'utilisateur que l'étape est correctement publiée.

Post condition

L'article est publié sur le Blog de l'utilisateur.

Utiliser l'utilitaire de voyage

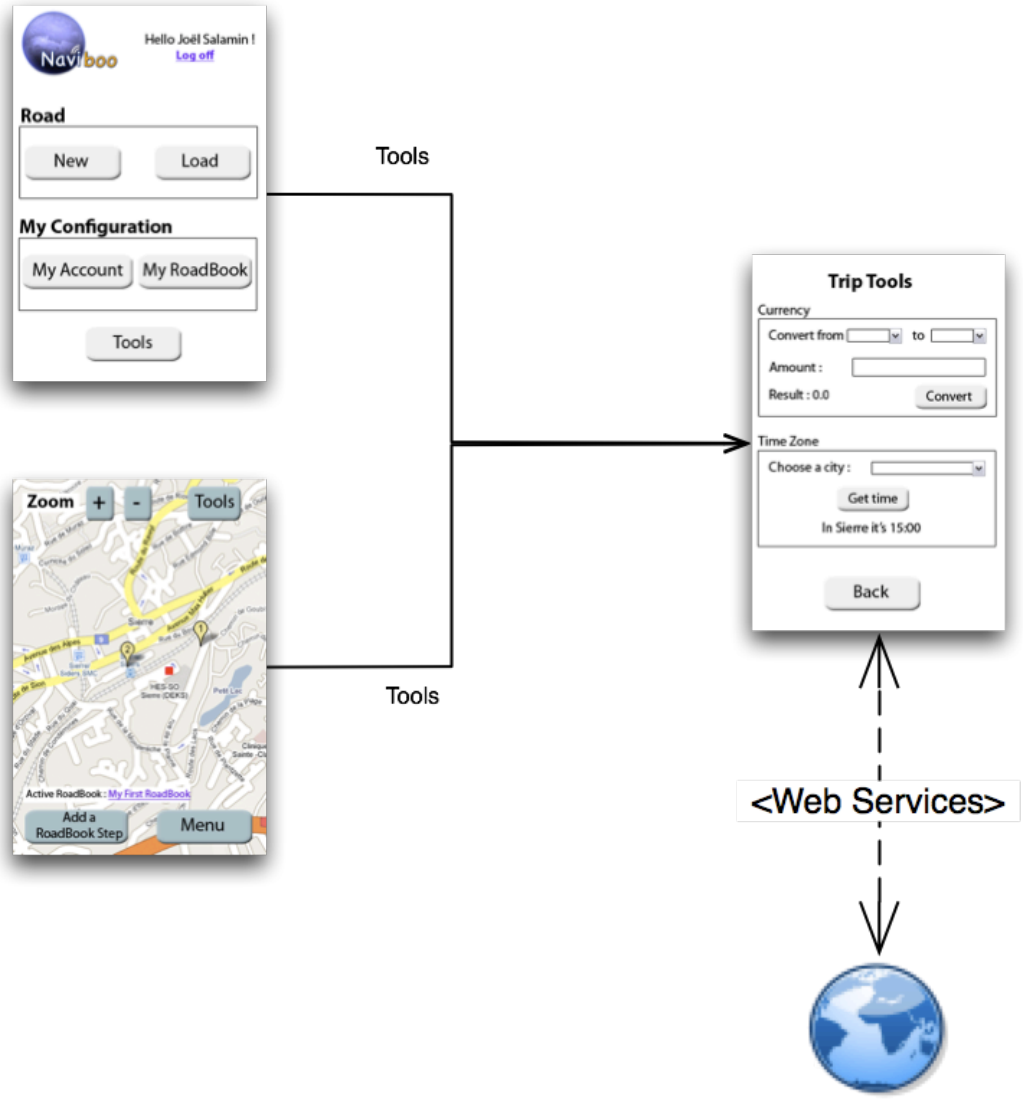

Figure 12 - Processus : Utiliser l'utilitaire de voyage

Pré condition

L'utilisateur doit avoir un compte et être identifié.

Description

L'utilisateur peut à tout moment, lors de sa visite ou directement depuis le menu, se servir de l'utilitaire de voyage mis à disposition par l'application.

L'utilitaire fournit les réponses souhaitées en consommant les Web Services adéquats. Cela permet d'avoir des résultats pertinents (principalement pour la conversion de monnaies dont les taux varient régulièrement).

Post condition

Les informations souhaitées sont récupérées depuis le service web et affichées à l'utilisateur.

Structure globale de l'application

L'application est composée de plusieurs écrans et l'utilisateur se retrouvera à naviguer à travers cette interface selon ses besoins et ses désirs.

Ci-dessous un schéma décrivant la navigation et la communication entre chacun des écrans :

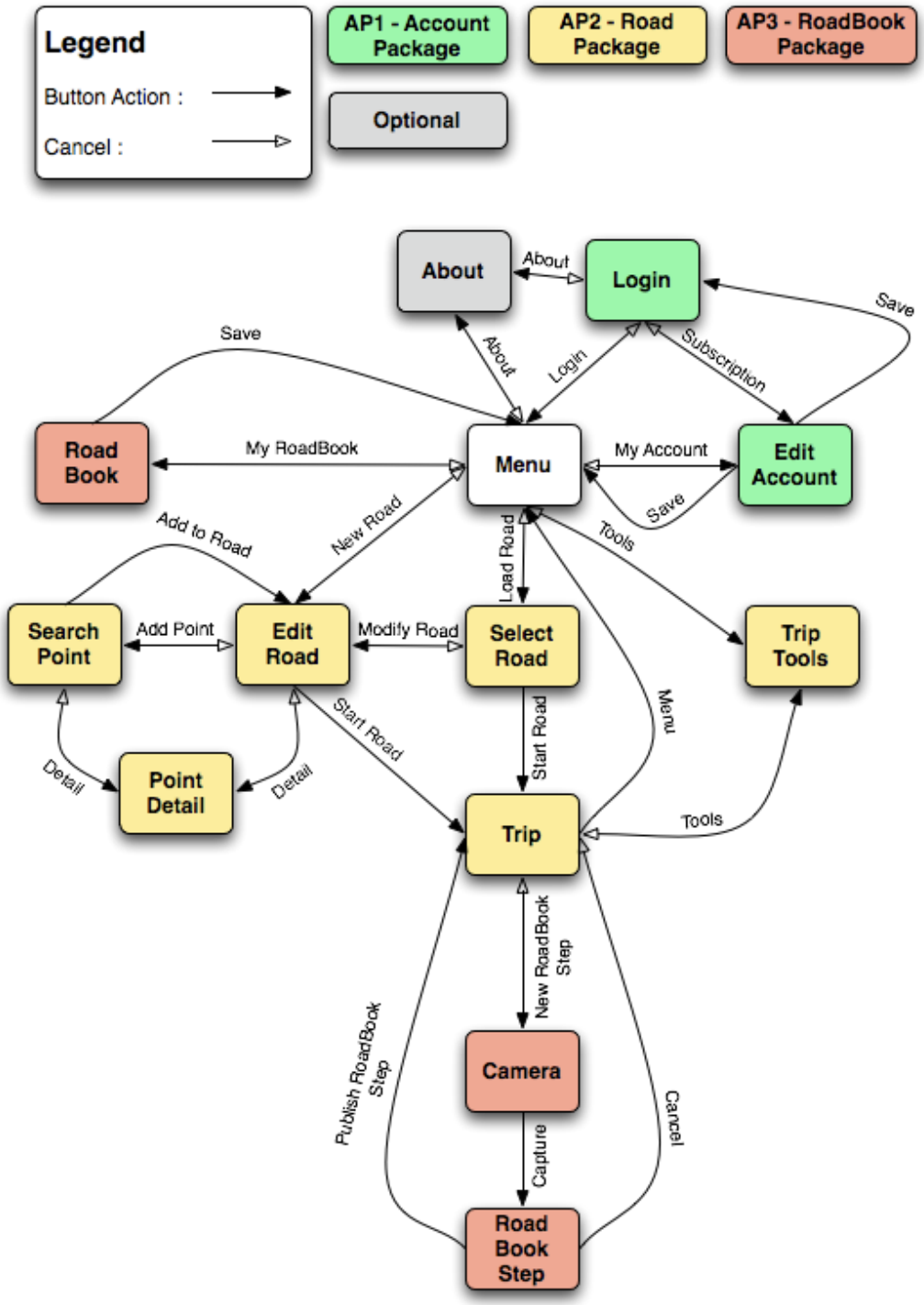

Figure 13 - Structure globale de l'application

Chaque flèche correspond à l'action d'un bouton de l'écran concerné. Ce schéma permet d'avoir une vision globale de toute la navigation entre les différents écrans. Il est également à remarquer que les différents « package » de l'application sont différenciés par des couleurs afin d'identifier facilement ces différents éléments.

5. Détail des différents packages

AP1 - Account Package

Ce package couvre toute la partie gestion du compte utilisateur. Les processus « Identification », « Inscription/Création d'un compte » et « Modifier son compte » font référence à ce package.

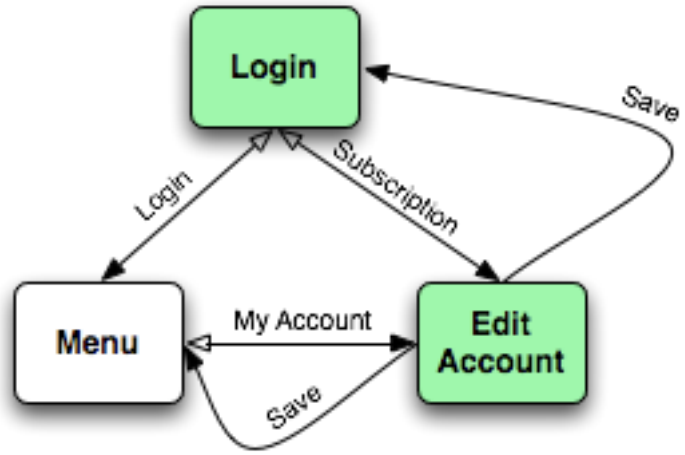

Figure 14 - Structure du Account Package

1.1 Itération 1 (Must Have)

- Création de chaque écran
- Implémentation de la navigation entre chaque écran
- Implémentation des méthodes, sans interaction avec la base de données

1.2 Itération 2 (Must Have)

- Implémentation des interactions avec la base de données
- Gestion des erreurs

1.3 Autres (Nice to Have)

- Implémentation de tests unitaires
- Remplacer les interactions avec la base de données par des interactions avec des services web

AP2 – Road Package

Ce package couvre toute la partie gestion des parcours de l'utilisateur. Les processus « Créer un parcours », « Affichage du détail d'un point » et « Charger un parcours » font référence à ce package.

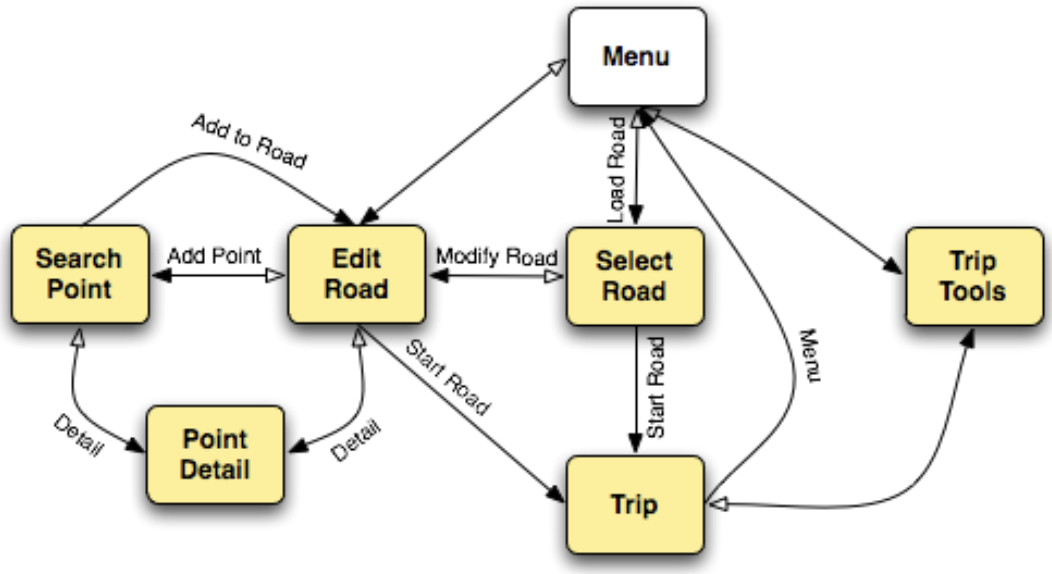

Figure 15 - Structure du Road Package

2.1 Itération 1 (Must Have)

- Création de chaque écran
- Implémentation de la navigation entre chaque écran
- Implémentation des méthodes, sans interaction avec la base de données

2.2 Itération 2 (Must Have)

- Implémentation de l'outil de voyage (Trip Tools) en utilisant un service web

2.3 Itération 3 (Must Have)

- Implémentation des interactions avec la base de données
- Gestion des erreurs

2.4 Itération 4 (Must Have)

- Implémentation du module GoogleMaps (affichage de ma position seulement) avec simulation d'entrées GPS

2.5 Itération 5 (Nice to Have)

- Implémentation de l'affichage du parcours/des points sur la carte

2.6 Autres (Nice to Have)

- Utilisation d'une base de données externes pour la recherche de points
- Utilisation d'un GPS externe pour la géo localisation
- Implémentation de tests unitaires
- Remplacer les interactions avec la base de données par des interactions avec des services web
- Génération de parcours sur la carte (pas seulement les points mais le parcours complet)

AP3 - RoadBook Package

Ce package couvre toute la partie gestion du RoadBook de l'utilisateur. Les processus « Configurer le RoadBook » et « Ajouter une étape au RoadBook » font référence à ce package.

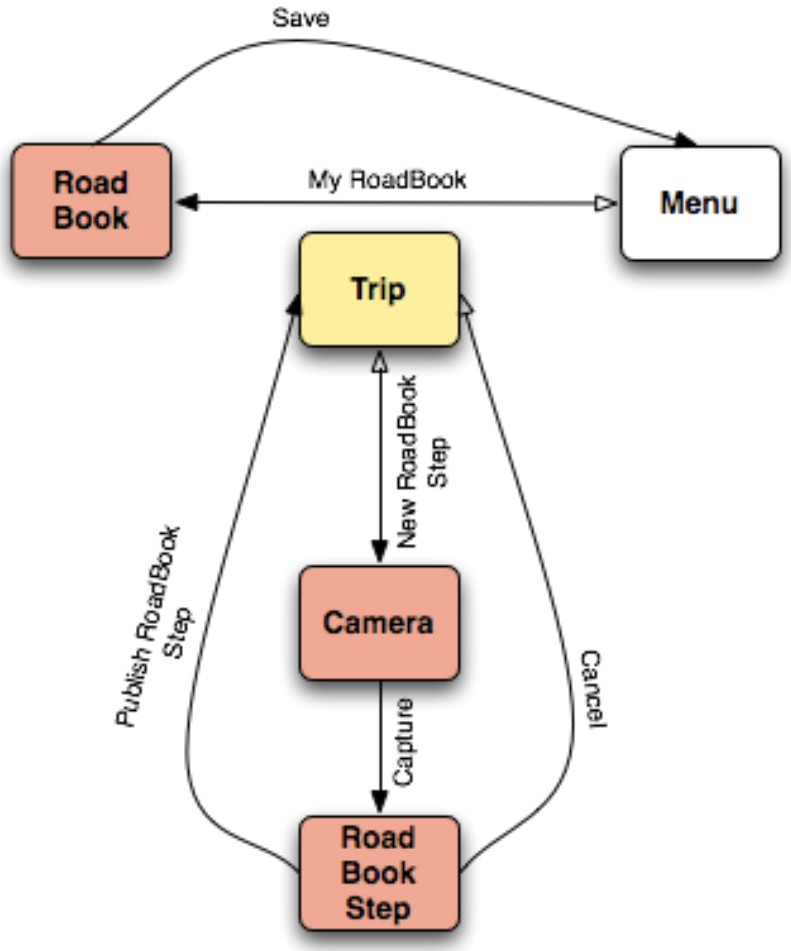

Figure 16 - Structure du RoadBook Package

3.1 Itération 1 (Must Have)

- Création de chaque écran (sans la fonction caméra)
- Implémentation de la navigation entre chaque écran
- Implémentation de chaque méthode, sans interaction avec la base de données

3.2 Itération 2 (Must Have)

- Implémentation des interactions avec Blogger
- Gestion des exceptions

3.3 Itération 3 (Nice to Have)

- Implémentation de la simulation d'une caméra grâce à l'iSight ou un explorateur de fichier en locale

3.4 Autres (Nice to Have)

- Implémentation de tests unitaires
- Remplacer les interactions avec la base de données par des interactions avec des services web

6. Délivrables souhaités

Il y a 2 principaux délivrables souhaités au terme du développement de l'application :

- Application fonctionnelle
- Documentation technique

Application fonctionnelle

Au terme du développement, ainsi qu'après chaque itération, il doit être possible de tester une application stable et fonctionnelle.

L'application devra, dans la mesure du réalisable, être le plus intuitive possible et offrir à l'utilisateur un produit facile à utiliser et simple à comprendre. Pour cela il est important d'utiliser et exploiter le mieux possible les choix pris par Google pour son environnement de développement (utilisation d'éléments graphiques orientés « touch », etc…).

Les itérations indiquées comme « Must Have » doivent être réalisées dans le temps mis à disposition. En revanche les itérations indiquées comme « Nice to Have » ne sont pas obligatoires mais sont à implémenter si le temps le permet.

Documentation technique

Au terme du développement, une documentation technique doit également être délivrée.

Cette documentation a pour but de fournir un support à l'application, sous forme d'un manuel d'utilisateur ou autres documents pertinents.

7. Table des illustrations

Bachelor | 08

Android SDK - Développement

Chapitre 3

Google Android

Joël Salamin

Hes·so VALAIS WALLIS

Table des matières

1. Introduction

Ce chapitre présente le projet réalisé dans son ensemble. Chaque étape de la réalisation du projet est illustrée par des exemples tirés du développement afin d'en comprendre le fonctionnement. Tout d'abord une description de la structure de l'application, les normes de programmation adoptées ainsi qu'un descriptif détaillé des éléments importants qui composent le prototype.

L'application a été réalisée sur la base du cahier des charges établi[1]. Pour résumer, l'objectif de ce prototype est de couvrir le plus largement possible les possibilités mises à disposition par le Framework Android créé par Google dans le but de s'imposer comme leader dans le monde des systèmes d'exploitation mobiles.

Présentation du projet

Le projet est baptisé « Naviboo – RoadBook » en raison de son lien de parenté avec l'application mobile Naviboo développée pour Windows Mobile dans le cadre d'une entreprise école lancée par l'association Business Experience.

Figure 1 - Logo Naviboo

Naviboo est un guide touristique virtuel offrant une nouvelle dimension à la visite touristique d'un lieu ou d'une région. Le prototype Naviboo – RoadBook reprend ce concept en y ajoutant une dimension supplémentaire grâce à l'apport d'un carnet de route. En effet, la dénomination RoadBook décrit le fait que le visiteur pourra tenir un carnet de route tout au long de sa visite et ainsi garder un souvenir et une trace de sa visite. Le prototype utilise toutes les technologies les plus récentes afin d'offrir un outil complet et dynamique à l'utilisateur.

Le RoadBook est directement publié sur le Blog de l'utilisateur, ce qui permet aux personnes de son entourage de suivre son voyage étape par étape et ce système lui permet également de garder un historique et un souvenir numérique de chacune de ses visites.

Un outil de voyage est également à la disposition de l'utilisateur, celui-ci lui permettra de convertir une somme d'argent d'une monnaie à une autre et également de convertir une distance d'une unité de mesure à l'autre. Un outil qui s'avèrera très utile lors d'un voyage à l'étranger où la monnaie est différente et parfois même les unités de mesures. Pour que cet outil soit dynamique et à jour, il utilise des WebServices qui permettent d'avoir à tout moment les cours monétaires actuels.

Enfin, Naviboo – RoadBook s'appuie sur GoogleMaps pour fournir à l'utilisateur une application simple d'utilisation avec des cartes actuelles et très complètes. La possibilité de changer l'affichage de la carte (carte routière/vue satellite) est également un point très intéressant pour répondre aux préférences de l'utilisateur.

---

[1] Se référer au chapitre 2 : « Cahier des charges de l'application »

L'application se veut la plus intuitive et la plus simple d'utilisation possible. Pour se faire, Naviboo – RoadBook est une application que l'on peut définir de « touch » car elle est entièrement utilisable avec le doigt et aucun stylet ni autres manipulations particulières ne sont nécessaires pour vivre pleinement l'expérience Naviboo.

2. Arborescence de l'application Naviboo – RoadBook

La structure de l'application est propre à Android et respecte les normes définies par Google. Il est bien entendu donné libre choix au développeur d'utiliser ou non cette structure. L'arborescence se veut relativement « logique », mais peut paraître complexe au début. Une fois le premier contact établi, on remarque rapidement que la structure est très simple à comprendre et qu'il est aisé de s'y retrouver et surtout de trouver les éléments que l'on cherche.

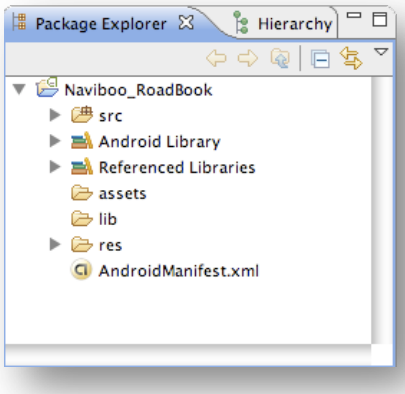

Figure 2 - Arborescence : Globale

Le fichier de configuration principal de chaque application se trouve à la racine du projet et se nomme « AndroidManifest.xml ». Il contient toutes les informations concernant l'application, les éléments qui la composent, la gestion des droits etc…

Arborescence des sources

Les sources de l'application se trouvent toutes dans le dossier « src » regroupées dans un ou plusieurs packages selon le projet.

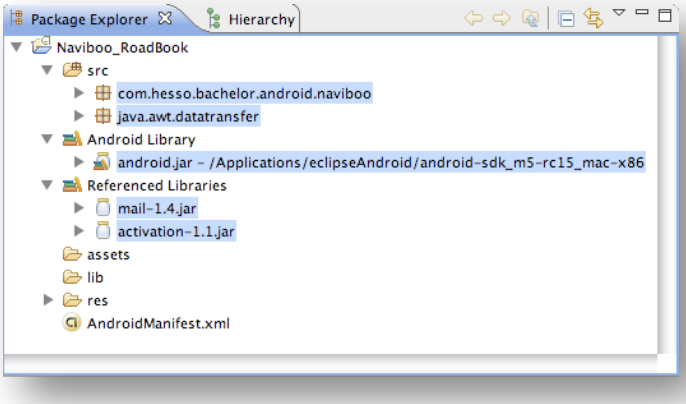

Figure 3 - Arborescence : Packages et librairies

Pour le projet, deux packages sont nécessaires :

- com.hesso.bachelor.android.naviboo
- java.awt.datatransfer

L'ensemble des Class développées pour l'application se situent dans le premier package.

Le SDK Android se trouve dans le dossier « Android Library », cette librairie contient tous les éléments nécessaires au développement d'une application Android.

Malheureusement tout n'est pas présent dans le SDK Android. Dans le cadre du projet, pour implémenter l'envoi de mail, certains éléments externes sont nécessaires. C'est pourquoi l'application est composée d'un second package permettant de gérer le transfert de données et de deux librairies Java qui sont rattachées au projet :

- mail-1.4.jar
- activation-1.1.jar

Ces librairies sont utilisées pour l'authentification lors de l'envoi d'un mail ainsi que la création du mail à envoyer.

Arborescence des ressources « drawable »

Toutes les ressources graphiques sont regroupées dans le dossier « res/drawable ».

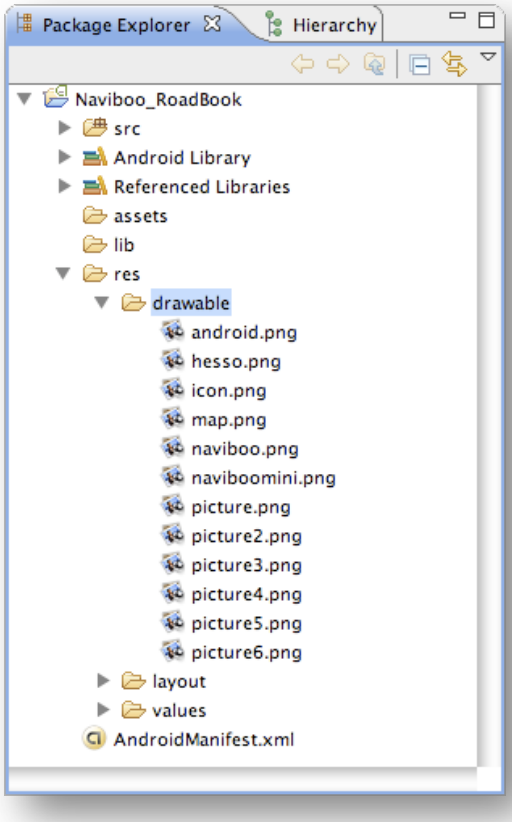

Figure 4 - Arborescence : Ressources "drawable"

Pour le projet, toutes les images sont stockées dans ce dossier. Le format des images est le PNG, choisi pour sa compression et la possibilité de gérer la transparence des images.

Arborescence des ressources « layout »

Toutes les ressources relatives à la structure visuelle de chaque écran de l'application sont regroupées dans le dossier « res/layout ».

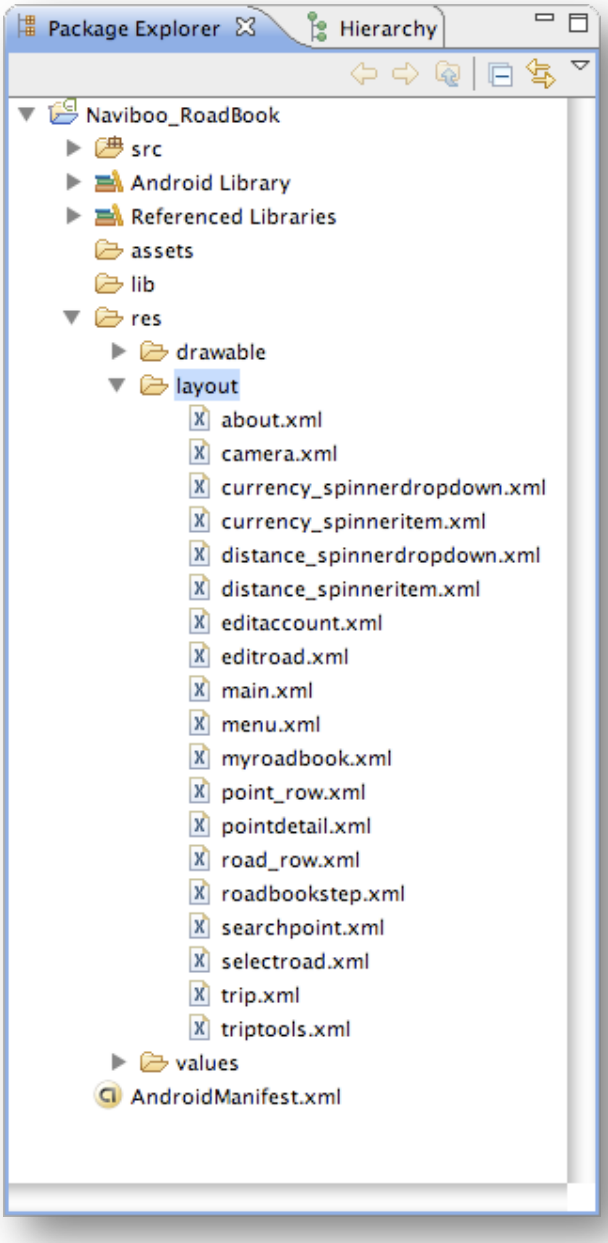

Figure 5 - Arborescence : Ressources "layout"

Pour le projet, il y a un fichier de layout par écran ainsi que certains fichiers spécifiques à un élément graphique tel que par exemple un élément d'une liste. Le fichier de layout est en fait un simple fichier XML contenant les différents éléments visuels[2] qui composent l'écran concerné avec différents paramètres pour déterminer son comportement. Le détail complet de ces éléments est présenté dans la partie « Interface graphique » de ce chapitre.

---

[2] Appelés Widget

Arborescence des ressources « values »

Toutes les ressources textuelles de l'application sont regroupées dans le fichier strings.xml du dossier « res/values ».

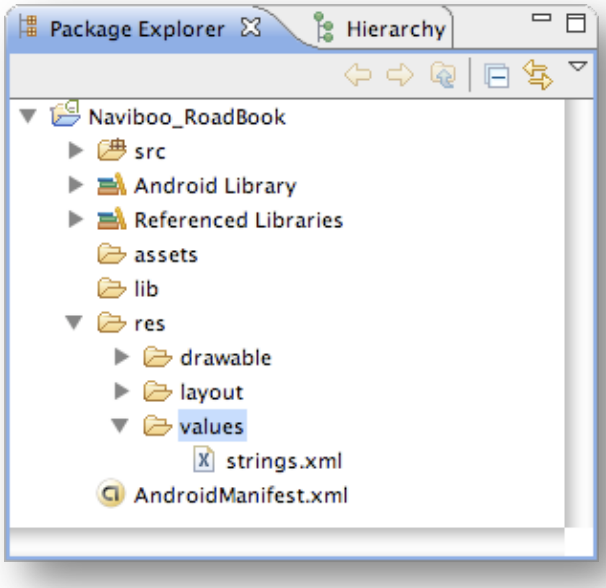

Figure 6 - Arborescence : Ressources "values"

Pour le projet, tous les textes composant les menus ainsi que le nom de l'application sont stockés dans ce fichier de ressources. Cela permet par exemple de pouvoir traduire l'application assez simplement et relativement rapidement sans avoir à manipuler la couche métier.

Fichiers indispensables pour une application Android

Certains fichiers doivent être présents pour qu'un projet Android puisse fonctionner. Lors de la création d'un nouveau projet à l'aide du plugin Eclipse, toute l'arborescence est automatiquement créée ainsi que les fichiers décrits ci-dessous.

AndroidManifest.xml

Ce fichier contient toutes les informations nécessaires à l'application pour fonctionner, chaque écran composant le projet doit y être déclaré avec les paramètres nécessaires.

Un écran qui n'est pas déclaré dans ce fichier n'est tout simplement pas reconnu par l'émulateur lors du déploiement de l'application, il est donc important de ne pas négliger ce composant du projet.

C'est également dans ce fichier que sont déclarées les permissions accordées à l'application au niveau de l'accès aux différentes ressources de l'appareil mobile.

Voici un extrait du fichier relatif au projet réalisé :

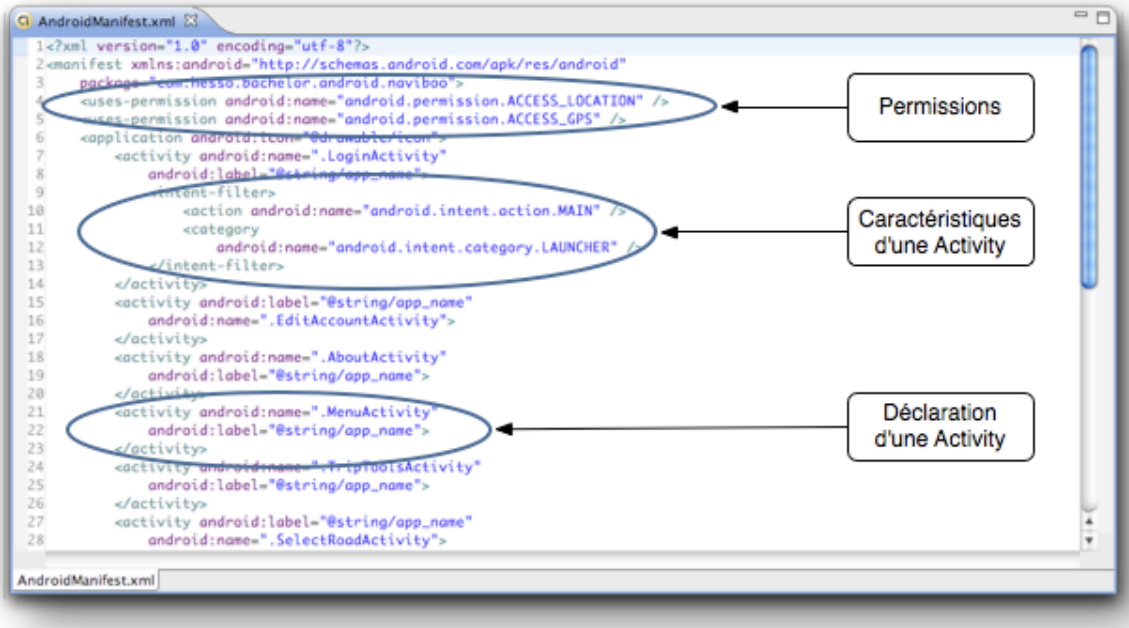

Figure 7 - AndroidManifest.xml

src/R.java

Ce fichier est généré automatiquement, il est conseillé de ne pas le modifier étant donné qu'il fait le lien entre toutes les ressources externes et la couche métier de l'application.

Chaque élément externe est identifié grâce à la catégorie de ressources à laquelle il appartient et à l'identifiant qui lui est attribué. Pour illustrer cela, voici tout d'abord un Widget qui fait partie d'un fichier de layout :

Et voici un extrait du fichier de ressources avec la référence au Widget choisit :

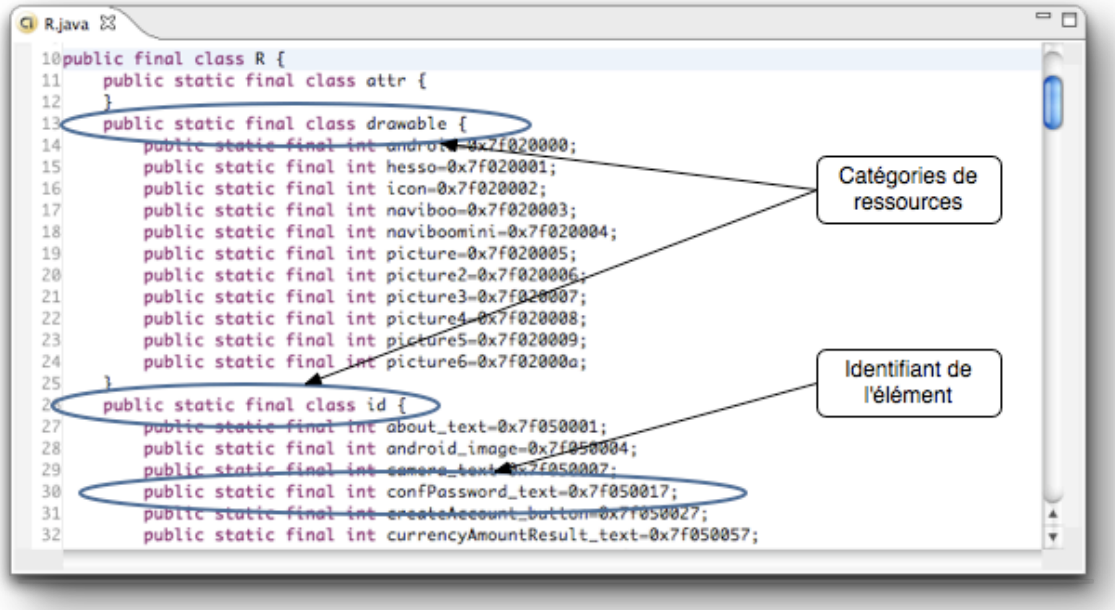

Figure 8 - R.java

Le fichier de ressources est structuré de manière à ce que chaque groupe de ressources soit distinct. On peut voir dans l'extrait ci-dessus 2 de ces groupes, le premier regroupant les ressources images et le deuxièmes regroupant les ressources Widgets.

Utilisation du Manifest Android Editor

Pour configurer correctement le fichier AndroidManifest.xml, le SDK Android met un assistant très pratique à disposition du développeur. Pour l'ouvrir, il faut faire un clic-droit sur le fichier AndroidManifest.xml puis sélectionner « open with Manifest Android Editor ».

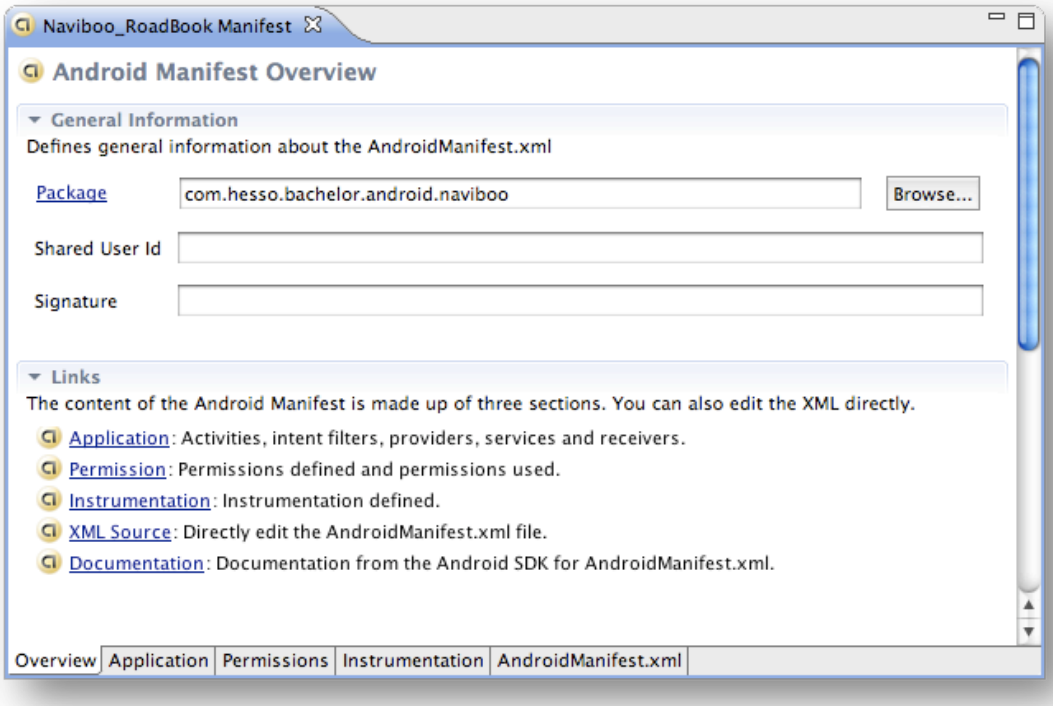

Figure 9 - Manifest Android Editor : Overview

Grâce à cet éditeur il est possible de configurer complètement le fichier pour qu'il permette à l'application de fonctionner et de communiquer correctement avec l'appareil.

Voici la description des 2 parties les plus importantes de cet assistant afin de comprendre l'utilité de celui-ci lors du développement.

Application

Cette partie est consacrée à l'organisation de l'application et aux différents éléments qui la composent.

A chaque nouvel écran créé il doit être ajouté à la liste des composants du projet sinon il ne sera pas reconnu et provoquera des erreurs lors du déploiement sur l'appareil mobile. Pour ajouter un élément il suffit de cliquer sur le bouton « Add… », choisir le type d'élément puis le sélectionner dans la liste proposée.

Voici un extrait de la partie Application de l'assistant avec les différentes Activity composant le projet et pour chacune d'elle, les différents attributs définis :

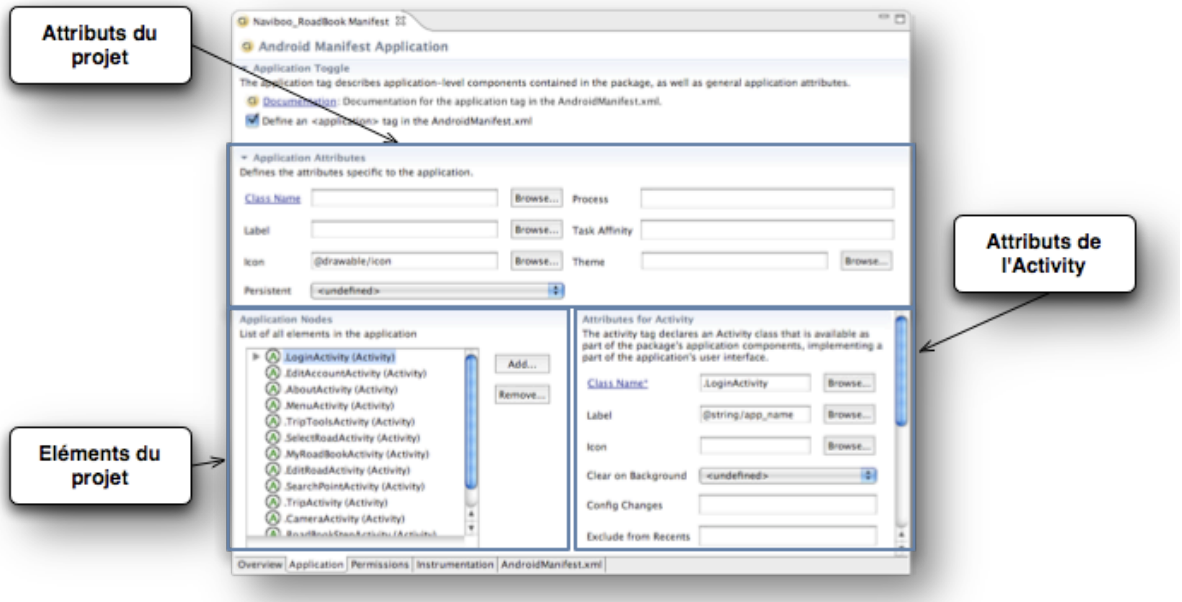

Figure 10 – Manifest Android Editor : Application

Permission

Par défaut, une application développée avec le SDK d'Android est complètement dépourvue de permissions, en d'autres termes elle ne peut manipuler aucune information ni aucune données provenant de l'appareil mobile : Carnet d'adresse, message, appel, GPS, etc…

Cette partie de l'assistant de configuration permet de définir les permissions dont pourra bénéficier notre application et ainsi lui permettre d'accéder à certaines données provenant de l'appareil mobile.

Pour le projet « Naviboo – RoadBook », les deux permissions nécessaires sont pour la première l'accès aux données GPS et pour la seconde, l'accès au simulateur de déplacement.

3. Normes de programmations adoptées

Ce point décrit les différentes décisions prises dans l'organisation du code et la structure adoptée pour le développement.

Introduction

En me basant sur les tutoriels diffusés par Google et sur les différentes sources trouvées sur des forums de discussion, j'ai établi une norme de programmation pour mon projet avec pour objectif d'avoir un code homogène et structuré.

Convention d'écriture

Pour le développement de l'application j'ai adopté comme convention d'écriture le CamelCase. Cette convention définit les règles suivantes :

- La première lettre de chaque mot composant le nom de la variable est en majuscule (à l'exception de la première lettre selon le cas de figure)
- La première lettre de la variable est une majuscule s'il s'agit du nom d'une Class (UpperCamelCase)
- La première lettre de la variable est une minuscule s'il s'agit d'une variable de Class, nom d'un champ, nom d'une méthode, etc… (lowerCamelCase)

Cette convention est beaucoup utilisée en programmation, elle est d'ailleurs la convention officielle pour le nommage de fichier Java, Microsoft .NET ainsi que Python recommande l'utilisation de cette convention[3].

Nommage des différents éléments du code

Etant donné qu'il n'y a aucune restriction au niveau du nommage des éléments, j'ai défini mes propres règles afin d'avoir des composants facilement identifiables. Il me parait nécessaire de faire cette démarche car sur la base d'une certaine expérience en programmation, j'ai remarqué qu'il est très rapidement compliqué de s'y retrouver si les éléments ne sont pas correctement nommés.

Ressources

Les différents éléments de ressources se doivent d'être convenablement nommés étant donné qu'ils sont régulièrement manipulés dans le code source. Les composants graphiques propres à Android sont appelés Widget et sont décrits au point 4 : « Interface graphique ».

Dans la partie layout, les éléments respectent les règles suivantes (les règles couvrent tous les éléments utilisés lors du développement) :

- TextView ➔ [lCC]_text
- EditText ➔ txt_[UCC]
- Button ➔ [lCC]_button
- ImageView ➔ [lCC]_image
- Spinner ➔ [lCC]_spinner
- Gallery ➔ [lCC]_gallery
- RelativeLayout, LinearLayout,… ➔ [lCC]_layout

lCC indique que le nom de l'élément respecte le lowerCamelCase, par exemple « nomDeMonElement_text » et UCC indique que le nom respecte le UpperCamelCase, par exemple « txt_NomDeMonElement ».

A noter qu'il y a un Widget spécial qui est utilisé dans les layouts, il s'agit de ListView qui représente une liste d'éléments. Son utilisation est décrite dans la partie « Interface graphique » de ce chapitre, il est juste à relever que l'identifiant d'un ListView est sous cette forme : « android :list » et qu'il est couplé avec un élément s'affichant seulement en cas de liste vide et qui possèdera l'identifiant « android :empty ».

---

[3] Source : www. wikipedia.org

Code sources

Les différents éléments utilisés dans le code source doivent être nommés de manière à être facilement identifiables et manipulables.

Pour cette raison, les éléments du code respectent les règles suivantes :

- Les variables de Class commencent par « m » suivit du nom de la variable en respectant le UpperCamelCase. Exemple d'une variable de Class :

```
// User Id
private Long mUserId;
```

- Les variables *final* s'écrivent en majuscule avec une dénomination selon le type de la variable. Un identifiant de menu s'écrit « ACTIONNAME_ID », où ACTIONNAME correspond à l'action effectué par cet élément du menu. Un identifiant de requête s'écrit « ACTIVITY_ACTIVITYNAME », où ACTIVITYNAME correspond au nom de la Class de type Activity effectuant la requête. Exemple d'un identifiant de menu et d'un identifiant de requête :

```
// Menu ID
private final static int SAVE_ID = Menu.FIRST;
private final static int CANCEL_ID = Menu.FIRST + 1;
```

```
// RequestCode
private static final int ACTIVITY_ABOUT = 0;
private static final int ACTIVITY_EDITROAD = 1;
```

- Les variables locales commencent par les initiales en minuscule indiquant de quel type est la variable, suivit du nom de la variable. Exemple de variables locales :

```
ImageView ivLogo = (ImageView) findViewById(R.id.navibooMenu_image);
```

```
ArrayList<String> alCurrency = new ArrayList<String>();
```

- Les OnClickListeners étant des variables de Class commencent par « m » et se terminent par « OnClick » pour indiquer que la variable est de type événementiel. Exemple d'un OnClickListener :

```
// This is the action listener for the Button bConvert
private OnClickListener mCurrencyConvertOnClick = new OnClickListener()
```

Ordre des composants dans les Class

Etant donné que les Activity sont composées de nombreux éléments, il est également important de garder une certaine rigueur dans la structure de celles-ci. Pour définir l'ordre le plus pertinent, je me suis une nouvelle fois inspiré de la structure des différents tutoriels de Google et autres exemples rencontrés sur les forums traitant du SDK Android.

Les différentes parties d'une Activity sont indiquées clairement par des balises de commentaires distinctives. Ca rend le développement et surtout l'ajout d'éléments bien plus simples.

Les différentes parties mises en évidence sont les suivantes :

```
/*****************************************************************************************/
/************************************** CLASS VARIABLES **************************************/
/*****************************************************************************************/
```

Cette partie regroupe toutes les variables de Class, les variables *static* et *final* sont également situées dans cette partie.

```
/*****************************************************************************************/
/************************************** HANDLER & LISTENERS **************************************/
/*****************************************************************************************/
```

Cette partie regroupe les éléments plus complexes de l'Activity tels que les Listeners qui sont attachés à des événements comme un clic sur un bouton. Ces éléments sont placés à la suite des variables de Class car ils sont utilisés comme des éléments faisant partie intégrante de l'Activity.

```
/*****************************************************************************************/
/************************************** EXTRA CLASSES **************************************/
/*****************************************************************************************/
```

Cette partie n'est pas présente dans toutes les Activity, elle regroupe toutes les Extra-Class nécessaires à l'Activity pour qu'elle puisse fonctionner correctement. Par exemple une Class héritant de IntentReceiver qui est nécessaire pour la réception des événements de l'Activity.

```
/*****************************************************************************************/
/************************************** OVERRIDE METHODS **************************************/
/*****************************************************************************************/
```

Cette partie regroupe toutes les méthodes qu'on override dans le but d'adapter les comportements de l'Activity à nos besoins. Parmi ces méthodes, il y a celle qui représente le point d'entrée de chaque Activity :

```
@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
```

Il est à noter que toutes les méthodes sur définies sont annotées « @Override » pour les différencier des autres méthodes.

```
/*****************************************************************************************/
/************************************** METHODS **************************************/
/*****************************************************************************************/
```

Cette dernière partie regroupe toutes les méthodes créées selon les besoins de l'application.

Cette structure se retrouve dans chacune des Activity et offre des repères visuels très utiles pour le développeur, cela permet également l'ajout relativement simple de nouvelles méthodes, nouvelles variables, etc…

Documentation du code

Toujours dans le but de rendre le projet plus lisible, évolutif et simple à la compréhension, chaque Class est complètement commentée à l'aide des balises et attributs spécifiques à la JavaDoc. Cette manière de commenter permet à la fin du projet de générer une JavaDoc complète sur tout le projet grâce aux outils intégrés à Eclipse. Malheureusement pour le moment cette fonction n'est pas utilisable car le SDK Android n'est pas complètement ouvert et la génération ne peut pas se faire.

Chaque Activity possède un commentaire indiquant l'auteur, le descriptif et la version de l'application. Chaque méthode est également commentée ainsi que différents éléments ou parties de code afin de les rendre facilement compréhensibles par quelqu'un n'ayant pas participé au projet.

4. Interface graphique

Ce point détaille la totalité de l'interface de l'application avec un descriptif de certains écrans permettant la compréhension du fonctionnement ainsi que la structure de chacun.

Introduction

Chaque écran est une Class d'un type spécifique à Android appelé Activity. Lors de la création de l'Activity, un schéma XML lui est attribué pour définir les éléments graphiques qui la composent : le layout. Chaque layout porte le nom de l'Activity **(choix personnel)** pour laquelle il définit la couche visuelle à l'exception de l'Activity « LoginActivity » qui représente le point d'entrée de l'application et se voit donc attribuer le layout « main.xml ».

Un fichier de layout est composé de Widget simple (View) ou de Widget composés appelé conteneur (GroupView) avec différents paramètres permettant de définir entre autres la taille, la position sur l'écran, certains éléments du comportement et bien d'autres. Le détail complet de ces éléments et de leur utilisation se trouve dans la documentation en ligne de Google : code.google.com/android.

Voici un exemple de fichier de layout pour mieux comprendre leur structure, cet extrait est tiré de l'écran principal : LoginActivity :

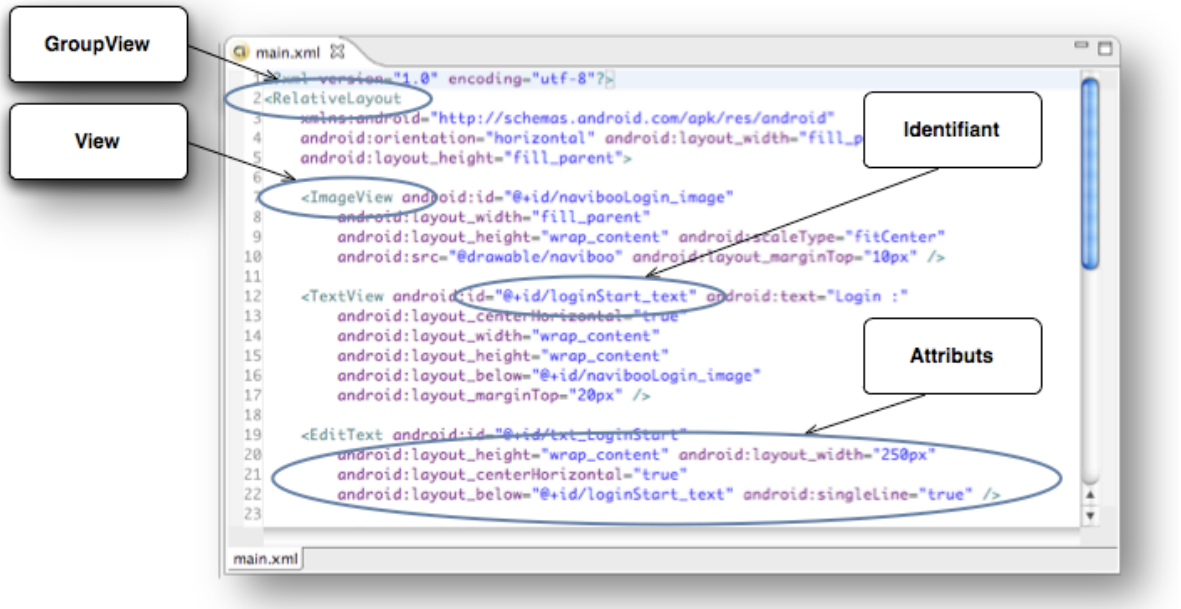

Figure 11 - Interface graphique : Exemple de fichier layout

Choix pris dans le cadre du projet

Google a choisi une orientation très clairement « touch » dans la composition des éléments visuels. Pour mon projet, je souhaite exploiter au maximum tous ses éléments afin de les mettre au service de l'utilisateur. L'objectif visé est d'avoir une application entièrement utilisable à l'aide du doigt sans que cela nécessite des manipulations complexes et peu intuitives. Il est tout de même à relever que l'application est entièrement utilisable en se servant uniquement du pavé directionnel, ce qui permet à celle-ci d'être compatible avec des appareils qui ne sont pas équipés d'un écran tactile.

Etant donné que le système d'exploitation Android met à disposition un menu accessible par la touche « menu » de l'appareil, il me paraît pertinent de l'exploiter au maximum afin d'alléger le plus possible l'écran et éviter que l'utilisateur croule sous un trop grand nombre d'éléments visuels. Quelques écrans sont détaillés ci-dessous pour passer en revue les composants utilisés lors du développement.

Exemple pratique - EditAccountActivity

Cet écran remplit deux fonctions, soit il est utilisé lors de la création d'un nouveau compte, soit lors de la modification d'un compte déjà créé. EditAccountActivity est lié au fichier layout : « editaccount.xml ».

Figure 12 - Interface graphique : EditAccountActivity

Cet écran va permettre d'illustrer l'utilisation de différents Widgets et du menu. On remarque que les éléments ne sont pas tous affichés, Android fournit un objet très utile pour permettre à l'utilisateur de naviguer simplement entre les différents composants. Sur l'illustration de droite, on peut voir les différentes fonctions accessibles depuis le menu, cela permet de soulager l'affichage en évitant de l'encombrer avec des boutons fixes.

Les deux types de Widgets simples utilisés pour cet écran sont :

- TextView : Label de texte
- EditText : Champ de saisie

Etant donné qu'il y a trop d'éléments pour que tout soit visible sur l'appareil, on va englober tous ces éléments dans un composant spécial : le « ScrollView » qui va utiliser un système d'ascenseur lorsque tous cela est nécessaire. Avec cette solution le layout est complètement indépendant de la taille de l'écran de l'appareil mobile.

Voici la structure du layout final :

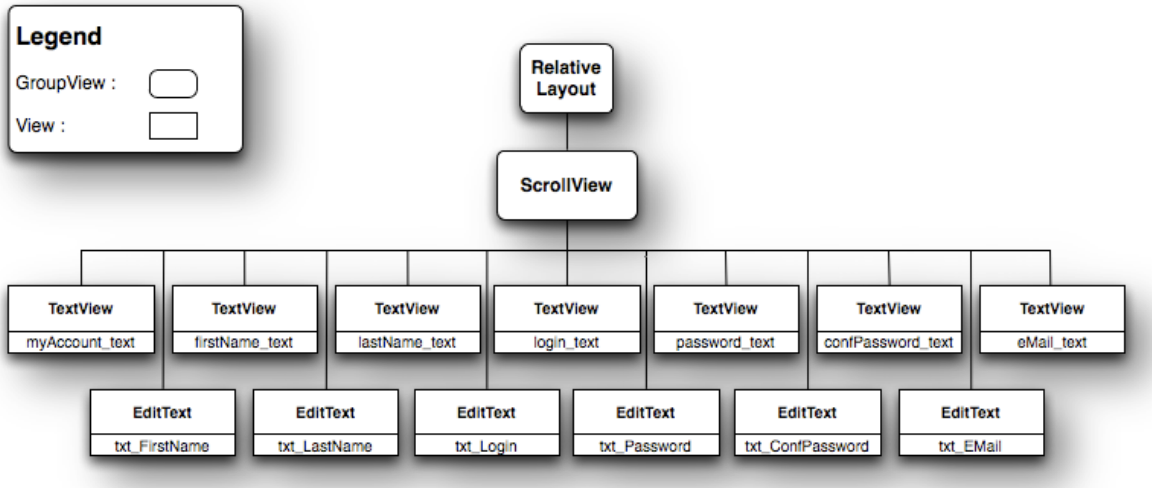

Figure 13 - Structure d'écran : EditAccountActivity

La structure peut paraître complexe mais elle est relativement simple en réalité. Tout layout possède un Widget de type GroupView à la racine, c'est lui qui va définir la manière dont sont répartis les éléments à l'écran.

Sur le schéma on comprend bien le rôle du ScrollView qui vient entre le conteneur et les éléments à afficher, cela permet d'adapter le layout à l'appareil qui sera utilisé en utilisant un système d'ascenseur si nécessaire.

Pour faire défiler les éléments, l'utilisateur peut soit utiliser le doigt en le gardant appuyer tout en dirigeant les éléments vers le haut ou vers le bas, soit en utilisant le pavé directionnel de l'appareil qui permet de passer d'un élément de saisie à un autre.

Exemple pratique - SelectRoadActivity

Cet écran est différent du précédent car il contient une ListView, un Widget qui ne se manipule pas de la même manière que les autres.

Avant tout, la Class qui est composée d'un ListView doit être de type ListActivity, cela permet d'accéder à des méthodes bien spécifiques au niveau du code pour la manipulation de la liste.

Pour qu'une ListActivity fonctionne correctement, il faut lui attribuer une source de données qui sera utilisée par l'objet ListView :

```
// Get all Roads created by the user
Cursor cRoads = mRoadDBAdapter.fetchAllRoads(mUserId);
startManagingCursor(cRoads);

// Create an array to specify the fields to display in the list
String[] sDatabaseFields = new String[]{RoadDBAdapter.KEY_ROAD_NAME};

// Create an array of the target fields in the list
int[] iLayoutFields = new int[]{R.id.roadTitle_text};

// Create the CursorAdapter needed to map the list with data
SimpleCursorAdapter scaRoads = new SimpleCursorAdapter(this, R.layout.road_row, cRoads, sDatabaseFields, iLayoutFields);

// Set the list Adapter
setListAdapter(scaRoads);
```

Une foit que la source est définie, la ListView va automatiquement être remplie par les informations contenues fournies par le SimpleCursorAdapter. C'est pour cette raison qu'au niveau du layout, le Widget de liste est identifié par « android :list ».

Au cas où la liste est vide, c'est le Widget identifié par « android :empty » qui est affiché à la place de la liste.

Figure 14 - Interface graphique : SelectRoadActivity

Pour cet écran, le menu est également utilisé dans le but de simplifier l'interface et regrouper toutes les actions que l'utilisateur peut effectuer dans le cadre de la gestion de ses parcours.

Voici la structure du layout final :

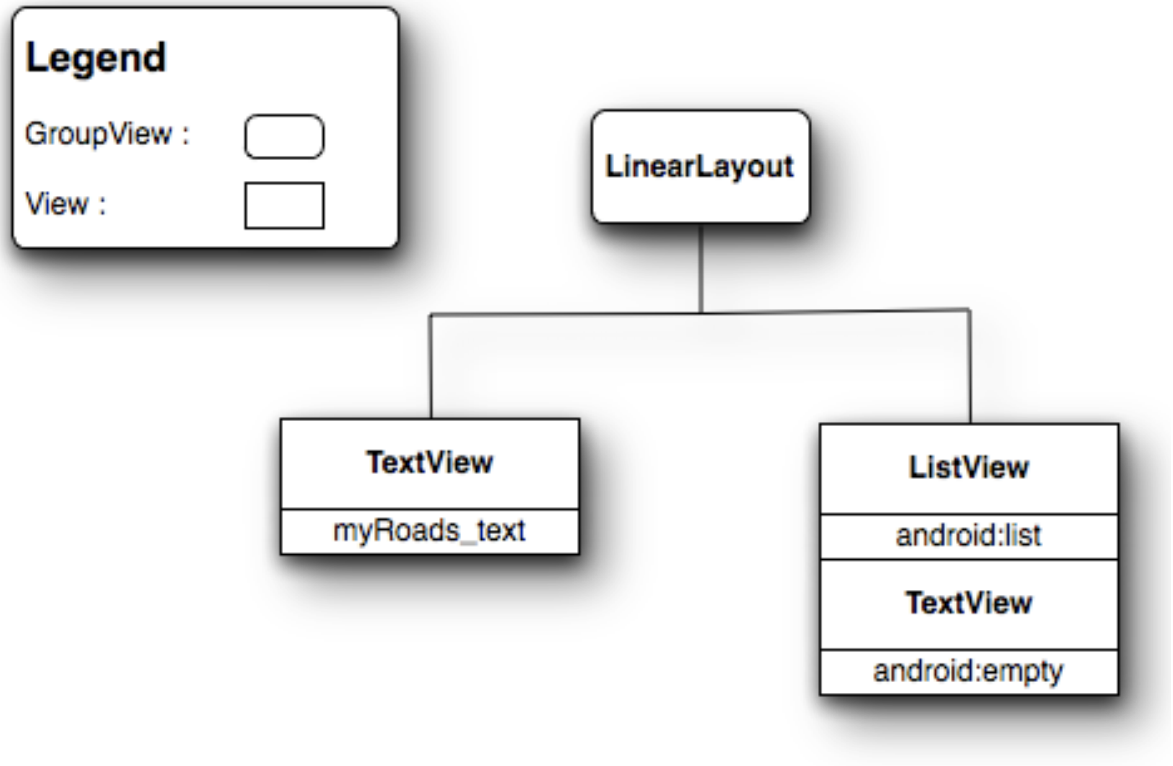

Figure 15 - Structure d'écran : SelectRoadActivity

On remarque la particularité d'une ListActivity car elle possède deux éléments différents pour une même position sur l'écran. Suivant l'état de la liste, elle peut être affichée ou non. La gestion de la source de données se fait automatiquement, il n'y a rien à faire au niveau du code d'une fois que le ListAdapter est défini.

Exemple pratique - SearchPointActivity

Le dernier exemple est intéressant pour comprendre la différence entre les deux types de conteneurs principalement utilisés dans le projet.

Figure 16 - Interface graphique : SearchPointActivity

L'interface est simple, on peut remarquer qu'elle possède une liste de points d'intérêt qui correspondent à la recherche. Dans ce cas précis, aucun point n'a été trouvé et c'est donc l'élément identifié par « android :empty » qui est affiché, en l'occurrence c'est un TextView qui affiche le texte « No Points corresponding ! ».

Voici la structure du layout qui va nous permettre de différencier les conteneurs utilisés :

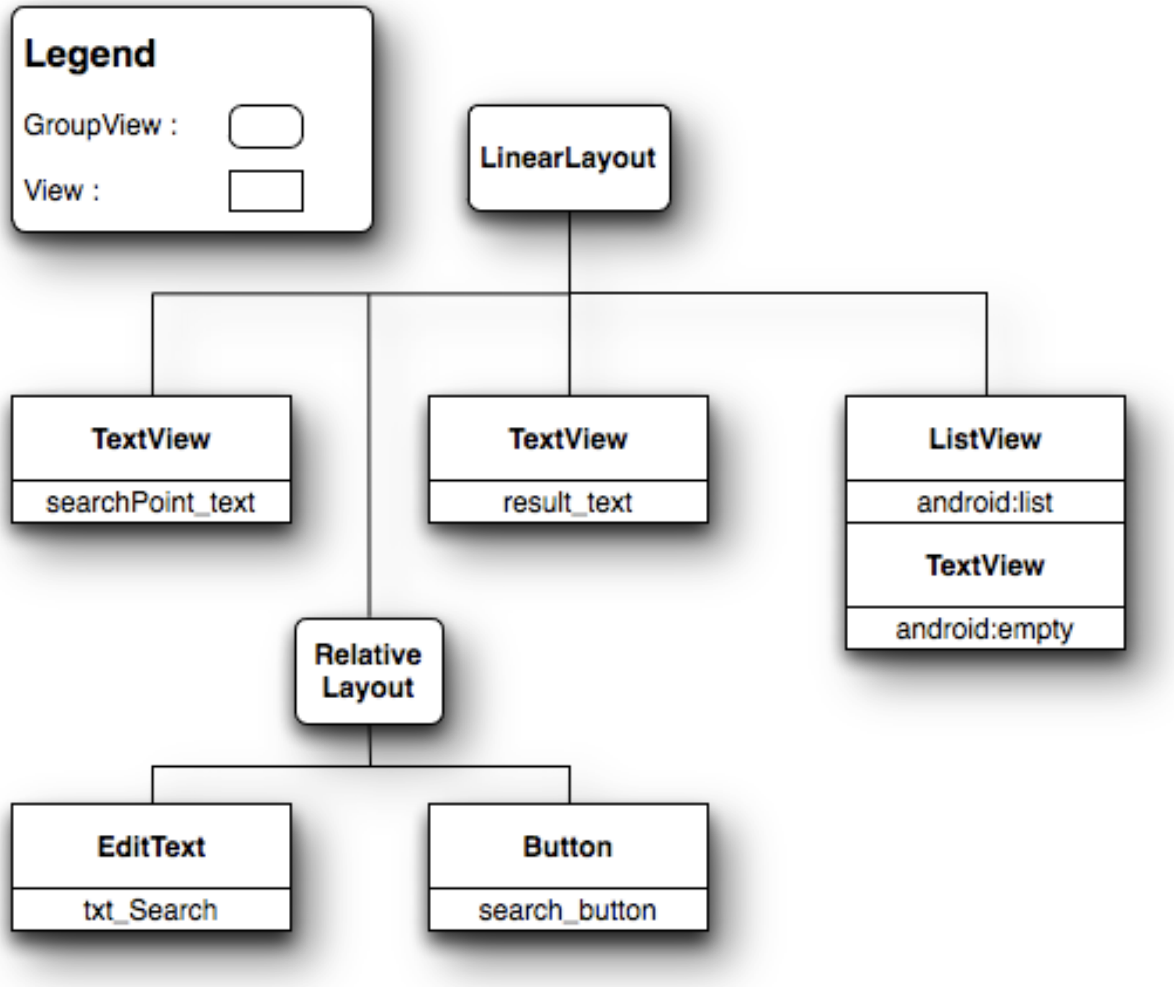

Figure 17 - Structure d'écran : SearchPointActivity

Le LinearLayout est, comme son nom l'indique, un conteneur qui affiche les éléments les uns à la suite des autres de manière verticale ou horizontale suivant la configuration. Il ne permet pas de positionner les éléments différemment et cela peut poser problème suivant les interfaces que l'on veut créer.

Le RelativeLayout est quant à lui un conteneur qui permet de positionner les éléments les uns par rapport aux autres. Il est alors possible par exemple d'avoir plusieurs éléments sur la même ligne.

Dans le cas de notre écran, une combinaison de ces deux conteneurs est faite pour profiter des avantages que propose chacun.

Le conteneur linéaire permet de se décharger de la contrainte du positionnement de l'élément mais est en revanche très rigide. Le conteneur relatif est par contre très flexible au point de vue du positionnement, mais oblige le développeur à définir la position de chacun des éléments et cette tâche devient très contraignante lorsque l'interface est complexe.

Manipulation des éléments graphiques au niveau du code

Une fois les différentes interfaces créées, la couche métier peut les utiliser et les manipuler très simplement. Avant tout, chaque Activity nécessite qu'un layout lui soit attribué pour déterminer sa structure visuelle

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
        // Load up the layout
        setContentView(R.layout.main);
```

Cet extrait tiré de LoginActivity illustre parfaitement l'attribution du fichier de layout « main.xml » lors de la création de l'Activity. Cette étape est indispensable car une Activity est un écran de l'application et il a besoin d'une structure graphique pour exister.

A présent que le layout est chargé, il est possible de manipuler chaque éléments de l'interface graphique simplement en le récupérant à l'aide de son identifiant, d'où l'importance d'avoir une certaine rigueur dans le nommage.

```
EditText mPasswordEditText = (EditText) findViewById(R.id.txt_PasswordStart);
```

Cet exemple illustre la récupération d'un Widget de type EditText à l'aide de la méthode findViewById(). Le paramètre fournit à cette méthode est obtenu à l'aide du fichier de ressources R.java.

Une fois l'objet récupéré, il n'y a plus qu'à l'utiliser selon les besoins et il sera automatiquement mis à jour au niveau visuel.

Problèmes rencontrés

Lors de la création de l'interface graphique de l'application, certains problèmes sont apparus :

Absence d'assistant graphique

Le SDK Android ne fournit aucun assistant pour la création de l'interface graphique d'une application. Etant donné le nombre relativement important d'éléments disponibles et la quantité de paramètres qui peuvent être définis, la création d'interface est relativement compliquée à réaliser.

La documentation Google est très complète et permet de comprendre tous les principes importants de la réalisation d'une interface mais lorsqu'il s'agit de faire une interface personnalisée selon nos besoins, l'absence d'assistant ralenti considérablement la tâche.

Mises à jour du SDK

Google s'investit considérablement dans l'évolution de son environnement de développement Android et c'est un point très positif. Malheureusement ces mises à jour ont des répercutions relativement négatives sur les développements en cours car

certains changement nécessite une adaptation de l'application pour répondre aux nouvelles règles fixées par Google.

Au niveau de l'interface graphique, de nombreux éléments et attributs ont changé complètement de nom. Cela ralenti le développement si la mise à jour intervient durant la période de projet étant donné qu'il est nécessaire de remanier une partie du code. Par chance je n'ai pas eu ce problème-là étant donné la durée relativement courte du projet. En revanche ce problème de mise à jour a eu un impact sur mon projet lors de la phase de formation et lorsqu'il fallait trouver la solution à un problème de développement car sur internet les réponses varient selon la version du SDK.

Conclusion

Google fournit avec le plugin Android un outil très complet et très simple d'utilisation pour un développeur ayant déjà travaillé avec Java et Eclipse. Les nombreux choix pris par Google s'avèrent pour la plupart entièrement adaptés au développement d'une application mobile, malgré certains éléments comme l'absence d'assistant graphique qui ralentissent le développement.

L'interface créée est très propre, les éléments sont très agréables à utiliser et l'orientation « touch » apporte un réel avantage à l'application réalisée.

Le niveau design des éléments proposés par le SDK Android est très réussi et chacun des composants est complètement paramétrable pour modeler l'interface selon les envies du développeur.

Le menu est un élément qui permet de simplifier de manière conséquente la réalisation d'une interface graphique peu encombrée. Son utilisation s'avère presque indispensable si on souhaite rendre l'application agréable à utiliser.

Les premiers écrans sont compliqués à réaliser car l'utilisation d'un fichier XML pour parvenir à créer l'affichage souhaité s'avère être un exercice relativement périlleux lorsque l'on ne possède aucune expérience. Mais les craintes des premières heures se dissipent rapidement car une fois la technique acquise, il devient assez simple de manipuler ces éléments et de réaliser des écrans répondant aux attentes fixées.

5. Base de données – SQLite

Ce point détaille toute la couche base de données de l'application avec une description de la structure de base ainsi que les éléments importants de la manipulation de celle-ci.

Introduction

Pour le projet Naviboo – RoadBook, l'utilisation d'une base de données est indispensable pour stocker les différentes données relatives aux utilisateurs et aux parcours. Grâce à la base de données, il est possible de sauvegarder les informations concernant les parcours créés, le carnet de route et les données concernant le compte de l'utilisateur.

Choix pris dans le cadre du projet

Au niveau de la base de données, l'environnement Android propose un package spécifique à SQLite. Google a choisit d'intégrer SQLite ainsi que l'outil de gestion de base de données directement à son SDK.

Puisque l'outil fait partie intégrante du SDK, son utilisation est plutôt simple. Toute la manipulation de la base de données peut se faire dans le code source en utilisant le package « android.database.sqlite » ou en utilisant l'outil « sqlite3 » en mode console lorsqu'on est connecté à l'appareil mobile.

Le choix pour le projet est tout naturel car SQLite est fournit par l'environnement mais surtout parce que ce choix est le mieux adapté à la situation étant donné la légèreté de ce système au niveau des ressources. Vu que le développement se fait pour un appareil mobile, un aspect très important est d'avoir une application légère et la base de données se prêtant le mieux à ce cas de figure est SQLite.

Pour le projet, toutes les manipulations sont faites depuis la couche métier, l'utilisation de l'outil sqlite3 ne sera donc pas détaillée.

Description

La base de données utilisée dans le cadre du projet est très simple, elle est composée de 5 tables :

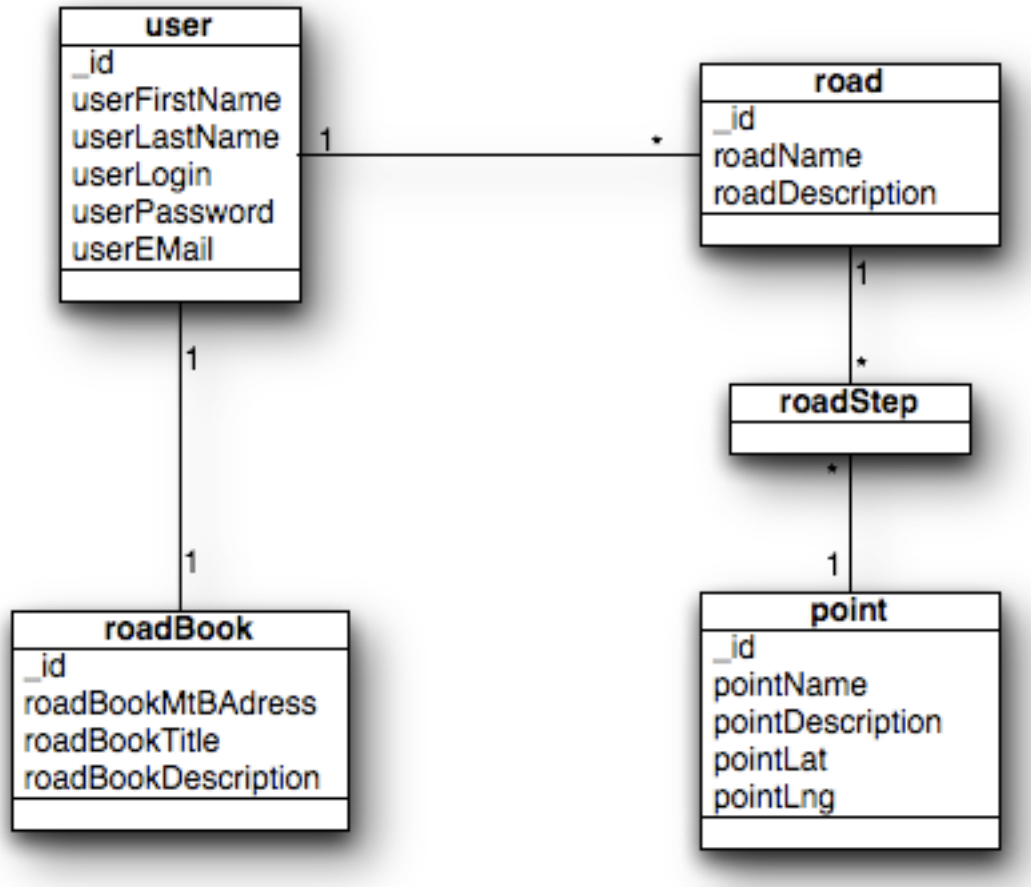

Figure 18 - Diagramme de base de données

La structure de la base de données a nécessité une certaine réflexion car l'application doit être évolutive et par conséquent la base de données également. C'est pour cette raison que la table utilisateur et la table carnet de route sont distinctes malgré la relation un-à-un entre elles.

Cela permet par exemple dans une version prochaine de l'application d'offrir la possibilité à l'utilisateur d'avoir plusieurs carnets de route sauvegardés.

Hormis ce choix qui mérite d'être justifié, le reste de la structure est très classique et permet de couvrir tous les besoins de l'application.

Concernant les étapes du carnet de route, je pense qu'il est préférable de ne pas en garder une trace au niveau de la base car il faut garder à l'esprit que le développement se fait pour un appareil avec des ressources relativement limitées.

DomainLogic utilisé

Pour toutes les interactions avec la base de données, deux Class regroupent la totalité des transactions nécessaires. Je me suis basé sur le DomainLogic « TransactionScript » en groupant toutes les requêtes en un même endroit :

- UserDBAdapter : Responsable de toute la partie relative à l'utilisateur et à son carnet de route
- RoadDBAdapter : Responsable de la partie relative aux parcours et aux points d'intérêt

Ces Class gèrent la connexion avec la base de données et effectuent les différentes requêtes demandées en gérant les éventuelles erreurs qui peuvent survenir lors de la transaction.

Exemple pratique – Récupération des données de l'utilisateur

La première étape consiste à créer une instance de la Class UserDBAdapter en lui donnant le contexte actif comme paramètre. C'est ce contexte qui sera utilisé pour établir la connexion à la base. Etant donné que l'Activity doit pouvoir manipuler la base de données à tout moment, l'instanciation se fait lors de la création de celle-ci :

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    setContentView(R.layout.editaccount);

    // Creation of the UserDBAdapter needed for user manipulation
    mUserDBAdapter = new UserDBAdapter(this);
    mUserDBAdapter.open();
```

Une fois la connexion ouverte, il suffit d'utiliser la méthode souhaitée. En l'occurrence je souhaite récupérer toutes les données du compte de l'utilisateur actif, j'appelle donc la méthode correspondante avec comme paramètre l'identifiant de l'utilisateur :

```
// Get the user from the Database
Cursor cUser = mUserDBAdapter.fetchUser(mUserId);
startManagingCursor(cUser);
```

La méthode retourne un curseur avec le pointeur situé sur l'enregistrement qui correspond à la recherche, si aucun utilisateur correspond à l'identifiant saisit, le curseur est null.

La méthode startManagingCursor() permet de se décharger des responsabilités de la gestion du curseur. Cela évite la surcharge de la mémoire de l'appareil et tout est géré de manière autonome.

A présent il est possible de manipuler les données contenues dans le curseur obtenu simplement en faisant référence au nom du champ dont on souhaite la valeur en indiquant le type de données que l'on attend :

```
String sFirstName = cUser.getString(cUser.getColumnIndex(UserDBAdapter.KEY_USER_FIRSTNAME));

mFirstNameEditText.setText(sFirstName);
```

Pour rendre l'utilisation des curseurs plus simple, le nom de chaque champ des tables manipulées est stocké en variable statique dans les deux Class DBAdapter. Cela évite de commettre des erreurs lors de la saisie et permet de modifier la base de données sans que cela nécessite des changements trop contraignants :

```
// Keys related to the user table
public static final String KEY_USER_ID = "_id";
public static final String KEY_USER_FIRSTNAME = "userFirstName";
public static final String KEY_USER_LASTNAME = "userLastName";
public static final String KEY_USER_LOGIN = "userLogin";
public static final String KEY_USER_PASSWORD = "userPassword";
public static final String KEY_USER_EMAIL = "userEMail";
```

Enfin il ne faut pas oublier de clore la connexion lorsque cela est nécessaire, si l'Activity se ferme ou rencontre un problème par exemple :

```
@Override
protected void onStop() {
    // Close the database connection
    mUserDBAdapter.close();
    super.onStop();
}

@Override
public void onFreeze(Bundle icicle){
    // Close the database connection
    mUserDBAdapter.close();
    super.onFreeze(icicle);
}
```

Problèmes rencontrés

Lors de la création et l'utilisation de la base de données SQLite, certains problèmes sont apparus :

Utilisation de l'outil « sqlite3 »

SQLite étant une base de données intégrée au système d'exploitation d'Android, pour la manipuler il est nécessaire d'utiliser un Terminal en ligne de commande avec l'outil intégré « sqlite3 ».

Cette utilisation est délicate à prendre en main lorsqu'on ne possède pas d'expérience dans le domaine, mais on se retrouve rapidement en terrain connu étant donné qu'une fois connecté à la base de données, c'est un environnement SQL standard en ce qui concerne les commandes.

Conclusion

Google fournit un package et un outil complètement intégré à Android et cela simplifie grandement la tâche. Quelques éléments nécessitent un certain apprentissage avant de pouvoir les exploiter correctement mais à ce niveau-là Google fournit de très bons tutoriels et la communauté Android est très riche en informations.

Tous les points forts de SQLite sont mis en évidence dans un environnement tel qu'Android étant donné qu'il est obligatoire d'avoir un système de base de données léger et nécessitant le minimum de ressources possible.

6. Interaction Web

Cette partie détaille toutes les interactions web permettant à l'application d'offrir un outil complet à l'utilisateur.

Introduction

Les interactions Web représentent un élément indispensable pour avoir une application dynamique. Il y a de nombreuses manières d'interagir avec le web : L'utilisation de Web Service, l'envoi de mail, la navigation internet, l'affichage de cartes géographiques, etc…

Utilisation dans le cadre du projet

Dans le cadre du projet, l'application nécessite certaines interactions web afin de proposer un outil dynamique à l'utilisateur.

Pour l'outil de voyage, l'utilisation de Web Service est nécessaire pour fournir des taux de conversions actuels car ces informations ne peuvent pas être stockées de manière statique sous peine de ne plus être pertinentes lors de conversions effectuées.

Pour la publication d'une étape du carnet de route, il est nécessaire d'implémenter un système d'envoi de mail sur un compte Blogger.

L'application utilise également le web pour utiliser le service GoogleMaps, mais cet élément est détaillé dans le point suivant.

Description des services web utilisés

L'application consomme deux Web Service pour fournir un outil de voyage optimal.

Taux monétaire

Le premier service fournit le taux de conversion entre deux monnaies, et l'application convertit le montant saisi à l'aide du taux adéquat.

Le service est mis a disposition librement par : www.webservicex.net

Les monnaies disponibles représentent la totalité des monnaies existantes, mais dans le cadre du projet je n'ai retenu que six d'entre elles afin de montrer que le système fonctionne :

- AUD ➔ Dollar Australien
- EUR ➔ Euro
- HKD ➔ Honk Kong Dollar
- XPF ➔ Franc Pacifique
- USD ➔ U.S. Dollar
- CHF ➔ Franc Suisse

Il est évident que l'ajout de monnaies supplémentaires peut se faire très simplement.

Conversion de mesure

Le second service permet de convertir une distance entre deux unités de mesures différentes.

Le service est mis à disposition également de manière libre par le même site web : www.webservicex.net

Les unités de distance disponibles pour ce service englobent la totalité des unités de mesures existantes. Dans le cadre du projet j'ai retenu simplement les unités les plus courantes :

- Kilometers
- Meters
- Centimeters
- Miles
- Yards
- Feet
- Inches

L'ajout de nouvelles unités peut se faire de manière très simple étant donné que le service web les propose.

Exemple pratique – Consommation d'un service web

L'environnement de développement Android ne fournissant aucun outil pour manipuler des Web Service, je me suis tourné vers les outils standard à Java qui permettent de le faire.

Avant toute chose, il faut identifier le service à utiliser ainsi que les paramètres qu'il peut manipuler :

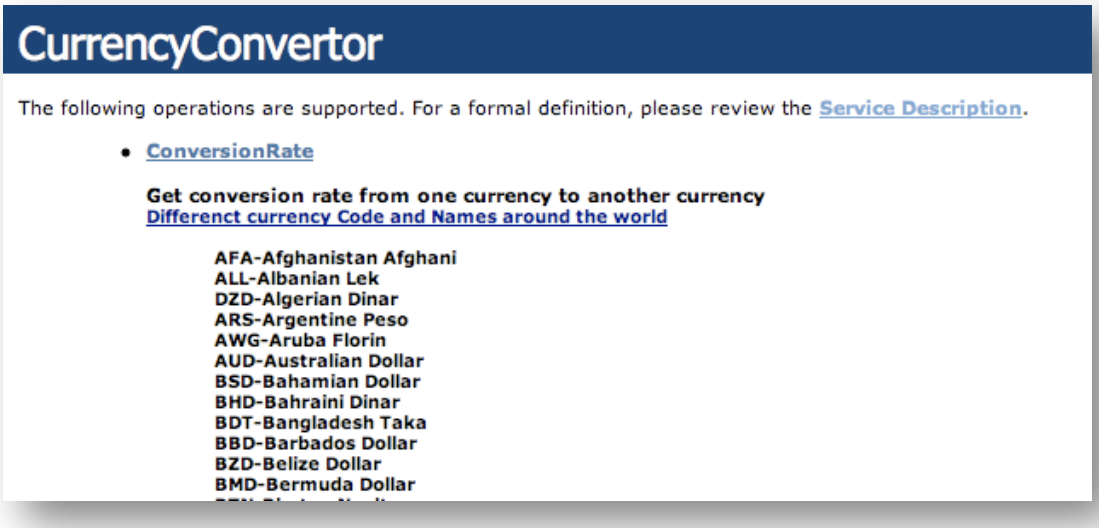

Figure 19 - Interaction Web : Web Service – Détail de méthode

La description relative à la méthode GET est la suivante :

Figure 20 - Interaction Web : Web Service - HTTP GET

Nous allons donc utiliser la méthode GET pour obtenir le taux de conversion souhaité, la forme du fichier XML de réponse est également détaillées.

La création de l'adresse en fonction des paramètres saisis par l'utilisateur se fait simplement par concaténation des différents éléments. Cette adresse est utilisée lors de la création de l'objet permettant la consommation du service via la méthode GET :

```
// Create the GetMethod that will be executed
GetMethod getMethod = new GetMethod(
          "http://www.webservicex.net/CurrencyConvertor.asmx/ConversionRate?FromCurrency="
            + sFrom + "&ToCurrency="
            + sTo);
```

Une fois l'objet GetMethod créé, la consommation du service ainsi que la récupération du résultat se font de la manière suivante :

```
// Creation of HttpClient to be able to execute the given GetMethod
HttpClient client = new HttpClient();

// Execute the GetMethod and get the returned status to know if the execution is OK
int status = client.executeMethod(getMethod);
String res = "";

// If the result is OK, we get the ResponseBody
if (status == HttpStatus.SC_OK)
     res = getMethod.getResponseBodyAsString();
```

Une fois le service consommé, il ne faut pas oublier de fermer la connexion :

```
// After consummation, the connection is closed
getMethod.releaseConnection();
getMethod = null;
```

Voilà les principales étapes nécessaires à la consommation d'un service web. Une fois le résultat récupéré, il suffit de convertir le montant saisi par l'utilisateur et de l'afficher.

Description de l'envoi de mail

L'application utilise un système d'envoi de mail pour publier un article sur un compte Blogger.

Le système s'appuie sur la plateforme de Blog fournie par Google nommée Blogger. Cette plateforme permet d'activer une option de publication sécurisée afin que l'utilisateur de puisse poster un nouvel article sur son Blog simplement par l'envoie d'un mail à une adresse protégée. L'adresse mail est sous cette forme :

identifiantGmail.phraseSecrete@blogger.com

L'utilisateur doit configurer son carnet de route afin d'indiquer quelle est l'adresse Blogger à utiliser pour la publication d'articles sur le Blog.

Pour implémenter l'envoi d'un mail il est nécessaire d'utiliser des librairies Java car rien ne permet de réaliser cette action avec le SDK Android. Les librairies importées pour parvenir à envoyer un mail avec une authentification auprès du serveur SMTP Gmail sont les suivantes :

- mail-1.4.jar
- activation-1.1.jar

Exercice pratique – Envoi d'un mail

Le serveur SMTP Gmail nécessite une authentification avant de pouvoir être utilisé, j'ai donc créé une Class MailSender qui hérite de Authenticator pour avoir une authentification et un outil adapté aux besoins de l'application.

Le constructeur de la Class effectue la configuration de la session utilisée pour envoyer le message que l'on va créer :

```
public MailSender() {
    Properties props = new Properties();
    props.setProperty("mail.transport.protocol", "smtp");
    props.setProperty("mail.host", "smtp.gmail.com");
    props.put("mail.smtp.auth", "true");
    props.put("mail.smtp.port", "465");
    props.put("mail.smtp.socketFactory.port", "465");
    props.put("mail.smtp.socketFactory.class",
            "javax.net.ssl.SSLSocketFactory");
    props.put("mail.smtp.socketFactory.fallback", "false");
    props.setProperty("mail.smtp.quitwait", "false");

    mSession = Session.getDefaultInstance(props, this);
}
```

Parmi ces paramètres, notons la présence du protocole à utiliser, le serveur, le port et d'autres informations nécessaires à la création de la session.

Maintenant que la session est créée, passons à la création du message :

```
// Creation of the message to send
MimeMessage message = new MimeMessage(mSession);
// Create the DataHandler needed for the message, it represents the message content
DataHandler handler = new DataHandler(new ByteArrayDataSource(body.getBytes(), "text/plain"));
// Set the internet address of the sender
message.setSender(new InternetAddress(sender));
// Set the subject of the message
message.setSubject(subject);
// Set the Datahandler
message.setDataHandler(handler);
// Set the internet address of the recipient
message.setRecipient(Message.RecipientType.TO, new InternetAddress(recipient));
```

Le message est composé de l'expéditeur, du sujet, du destinataire ainsi que du DataHandler qui représente le contenu du message.

Une fois le message correctement créé et paramétré, il ne reste plus qu'à l'envoyer :

```
Transport.send(message);
```

Les extraits de code décrivent uniquement les points importants dans le processus de création et d'envoi d'un mail et non la totalité de celui-ci.

Problèmes rencontrés

Lors de l'implémentation des interactions avec le web, certains problèmes sont apparus :

Aucune API intégrée

L'environnement Android ne met aucun outil à disposition pour effectuer ces interactions, il n'y a donc pas d'autre choix que d'utiliser des librairies Java standard pour le moment.

Il est possible que des API supplémentaires soient intégrées au SDK Android dans les prochaines versions diffusées par Google, mais à l'heure actuelle il n'y a rien de disponible.

Implémentation de Thread

Etant donné que la consommation de services web tout comme l'envoi de mail nécessitent un certain temps d'attente, il est nécessaire d'utiliser des Thread pour l'affichage d'un message d'attente indiquant à l'utilisateur que l'action est en cours.

Android fournit tous les éléments nécessaires à l'utilisation de Thread, mais le manque de documentation à ce sujet complique considérablement l'implémentation. Il y a également le fait que le fonctionnement des Threads ait subit certains changement entre la version actuelle du SDK et la version précédente, ce qui rend une grande partie des tutoriels obsolète.

Conclusion

Je suis relativement surpris de remarquer que Google n'ait pas jugé utile d'intégrer une API permettant la consommation de services web vu l'accent mis sur tout ce qui concerne le web.

Cette partie du développement a fait ressortir une faiblesse du SDK mais il est intéressant de noter que l'utilisation de packages Java standard se fait sans aucun problème au sein d'un projet Android.

Le fait de se tourner vers le langage Java apporte des avantages conséquents car même dans une situation telle que celle-ci où le SDK Android ne propose rien, il est possible d'y remédier rapidement en se tournant vers la communauté Java qui met à disposition tous les outils nécessaires.

7. GoogleMaps

Cette partie décrit le module GoogleMaps utilisé dans le cadre du projet afin d'offrir une expérience de visite interactive et agréable à l'utilisateur.

Présentation

GoogleMaps est un service de carte géographique en ligne gratuit et donne également la possibilité de générer des trajets routiers.

Un module GoogleMaps est intégré au SDK Android afin de mettre toute la puissance de cette application au service du système d'exploitation mobile dont Google est à l'origine.

Description

L'API GoogleMaps permet de proposer à l'utilisateur un guide touristique dynamique et très complet pour effectuer une visite en toute simplicité.

Pour l'application Naviboo – RoadBook, de nombreux éléments de cette API sont utilisés :

- Affichage d'une carte
- Géo localisation
- Affichage de POI (Point Of Interest)

La carte est utilisée pour indiquer à l'utilisateur sa position actuelle ainsi que la position géographique des différents points composant le parcours chargé.

L'utilisateur peut modifier le niveau de zoom à sa guise ainsi que changer le mode d'affichage (Carte routière/vue satellite) grâce au bouton « S » du clavier et il peut également centrer la carte sur sa position à l'aide du bouton « C ».

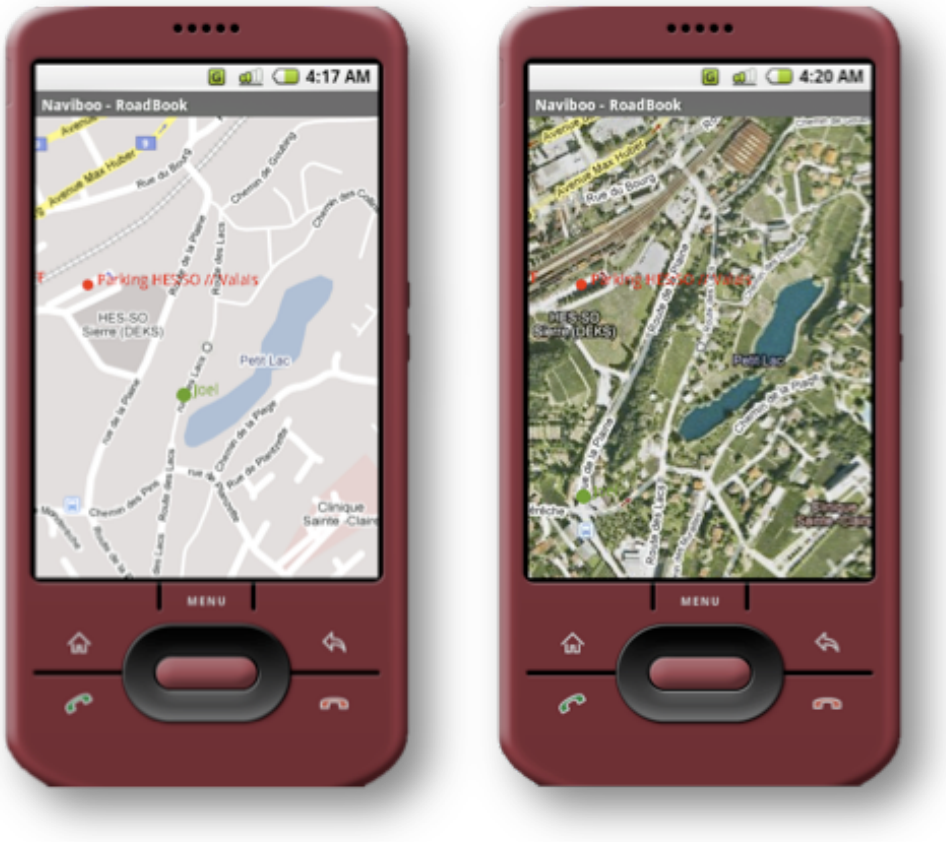

Figure 21 - GoogleMaps Display

La totalité de ces fonctionnalités sont intégrées au package « com.google.android.maps ». Ce qui fournit une interface simple à utiliser avec de nombreux éléments proposés par défaut tel que la possibilité de zoomer/dézoomer simplement en maintenant le doigt pressé sur l'écran afin d'afficher l'outil de zoom.

Il est possible de paramétrer entièrement l'affichage de la carte, l'actualisation de la position géographique et de nombreux autres éléments très simple d'accès.

Exemple pratique – Module GoogleMaps

Pour afficher une carte, il faut utiliser un type de Class spécifiquement destiné à cet effet faisant parti du SDK Android : MapActivity.

Ce type d'Activity, contrairement à celles utilisées jusqu'à présent, n'a pas besoin de fichier layout pour déterminer les éléments graphiques qui la composent. Lors de sa création, un élément de type MapView lui est attribué et c'est cet élément qui est chargé d'afficher la carte :

```
@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    mMapView = new MapView(this);
    setContentView(mMapView);
```

Il est possible de changer certains paramètres au niveau de l'affichage de la carte comme par exemple le mode utilisé (vue satellite, carte routière, etc…) :

```
// Change to Satellite display
mMapView.toggleSatellite();

// Change to Traffic display
mMapView.toggleTraffic();
```

Pour manipuler la carte, il faut récupérer l'élément MapController lié à la carte. Grâce à ce contrôleur, il est possible d'agir sur la carte, par exemple changer le niveau de zoom et centrer la carte sur un point donné :

```
// Get the MapController needed to manage the map
mMapController = mMapView.getController();

// Change zoom level
mMapController.zoomTo(20);

// Creation of a point centered on "Gare CFF, Sierre"
Point p = new Point((int)(46.292438*1E6), (int)(7.532527*1E6));

// Center the map on the desired point
mMapController.animateTo(p);
```

La dernière étape consiste à afficher des éléments personnalisés sur la carte, pour cela il faut ajouter une couche appelée Overlay qui s'affiche au dessus de la carte. Cet objet contient les différents éléments que l'on souhaite afficher comme par exemple la position de la HES :SO // Valais.

Avant tout il faut créer notre Overlay qui affichera un point là où se situe l'école :

```
// Overlay that display the position of HES:SO // Valais
protected class HesLocationOverlay extends Overlay {
    @Override
    public void draw(Canvas canvas, PixelCalculator calculator, boolean shadow){
        super.draw(canvas, calculator, shadow);

        // Paint is the tool used to draw everything on the screen
        Paint pPaint = new Paint();

        // Position of HES:SO // Valais
        Point p = new Point((int)(46.293178*1E6), (int)(7.536632*1E6));

        // Setup the paint style to draw a filled circle
        pPaint.setStyle(Style.FILL);

        // Convert the point into a coordinates on the screen
        int[] iCoords = new int[2];
        calculator.getPointXY(p, iCoords);

        // Draw the HES:SO position on the map
        canvas.drawCircle(iCoords[0], iCoords[1], 7, pPaint);
    }
}
```

A présent la couche affichant nos éléments est créée, il suffit de l'ajouter à la carte. Pour cela il faut récupérer le contrôleur chargé des différentes couches de la carte et y ajouter la nôtre.

```
// Get the OverlayController needed to manage the overlays of the map
mOverlayController = mMapView.createOverlayController();

// Create an instance of our specific Overlay
HesLocationOverlay hlo = new HesLocationOverlay();

// Add the Overlay to the controller of the map
mOverlayController.add(hlo, true);
```

Nous avons maintenant une carte centrée sur la gare de Sierre avec un niveau de zoom très proche et la HES :SO // Valais indiquée par un point.

Problèmes rencontrés

Lors de l'implémentation du module GoogleMaps à l'application, certains problèmes sont survenus :

Rafraîchissement de la carte

Afin d'avoir un outil de géo localisation pertinent il est important de gérer le rafraîchissement de la carte dans le but d'avoir la position de l'utilisateur en temps réel. Pour parvenir à une carte dynamique il est nécessaire d'utiliser un élément fournit par le SDK qui permet à l'écran d'être à l'écoute et de pouvoir actualiser la carte dès que cela est utile.

L'outil est complètement intégré au SDK mais son utilisation n'est pas très simple car de nombreux éléments sont à prendre en compte afin que l'application fonctionne correctement et une nouvelle fois la mise à jour du SDK a posé certains problèmes lors de la recherches d'informations pour résoudre les soucis relatifs au module de carte.

Gestion des ressources

Un point très important pour une application mobile consiste à ne pas saturer l'appareil inutilement et dans le cadre du projet la gestion des ressources est indispensable.

Une fois de plus le SDK Android est très complet et assez simple à utiliser étant donné la documentation relative à ce sujet.

Conclusion

J'ai pris un grand plaisir à utiliser cette API qui s'avère être très puissante et il est possible de réaliser de grandes choses assez rapidement.

On remarque l'accent mis par Google sur ces éléments de géo localisation et il faut reconnaître qu'ils sont parfaitement intégrés et bien pensés pour un système mobile.

A noter quand même que si on souhaite créer une application évoluée avec des fonctionnalités plus élaborées, il y a un certain manque au niveau des outils de formation tels que des tutoriels et autres exemples mis à disposition par Google. Mais pour palier à se problème une communauté très riche s'est formée autour d'Android.

8. Diagramme de Class

Ce point passe brièvement en revue les différentes parties qui composent l'application.

Introduction

Le diagramme de Class permet d'avoir une vue complète de l'application développée tout au long du projet ainsi que le détail des différents modules la composant.

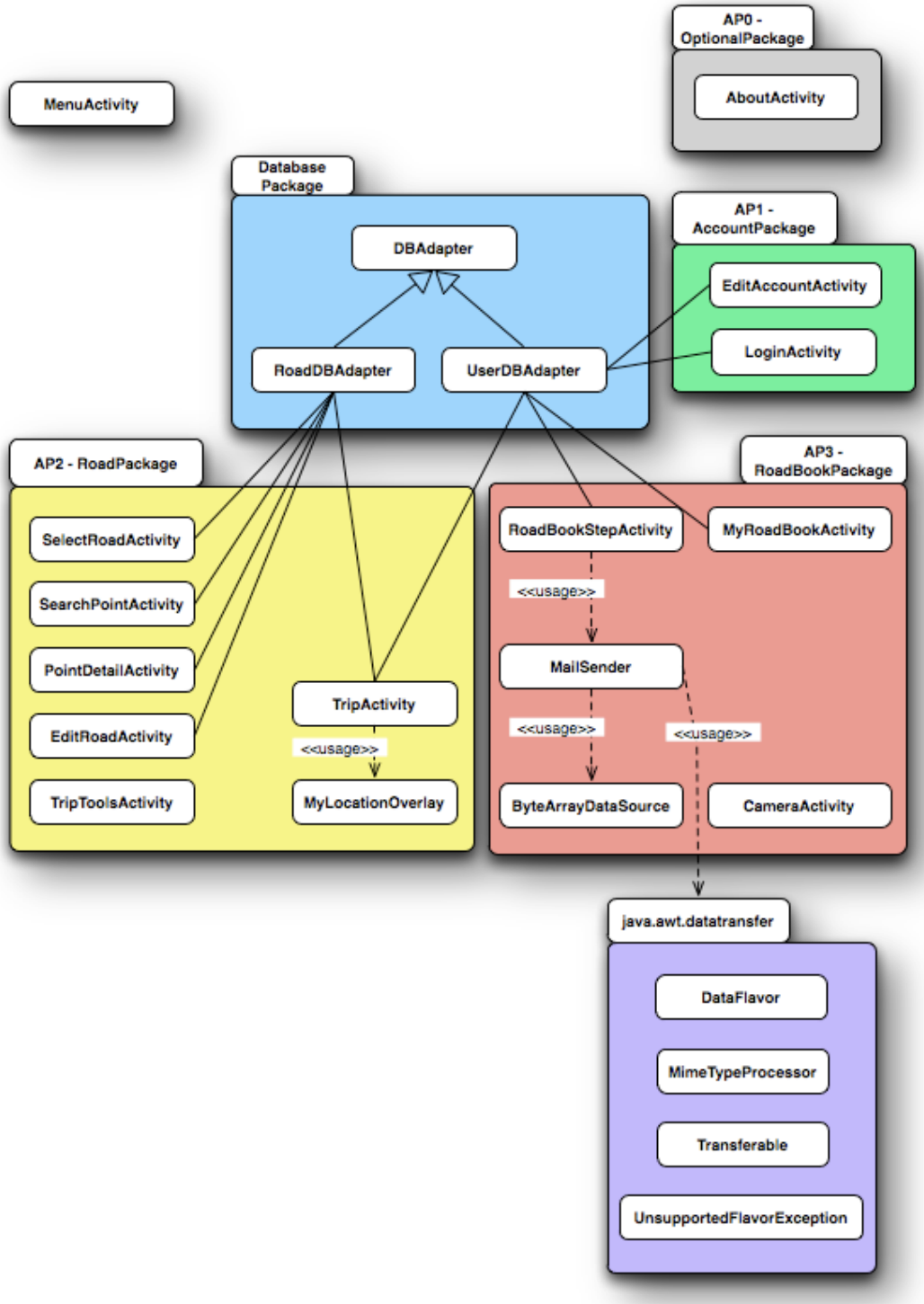

Figure 22 - Diagramme de Class

La structure de l'application est complexe car elle regroupe de nombreuses fonctionnalités différentes.

Les Class de type Activity sont nommées de manière à pouvoir les distinguer des Class normales. Chaque partie de l'application est représentée par un package qui fait référence au différents ApplicationPackage définis lors de la rédaction du cahier des charges :

- AP0 – OptionalPackage
- AP1 – AccountPackage
- AP2 – RoadPackage
- AP3 – RoadBookPackage

Ainsi que certains packages nécessaires à l'application pour les interactions avec la base de données et pour l'envoi de mail :

- Database Package
- Java.awt.datatransfer

AP0 – OptionalPackage

Ce package est définit comme optionnel car il ne fait pas parti des éléments exploitant les capacités mises à disposition par l'environnement d'Android. Mais au vue de la planification du projet, j'ai pu me permettre de l'intégrer au projet.

Description

OptionalPackage ne se compose que d'une seule Class de type Activity qui permet l'affichage d'un écran d'informations définissant le cadre dans lequel est réalisée l'application Naviboo – RoadBook.

AP1 – AccountPackage

Ce package englobe toute la partie relative à l'utilisateur.

Description

AccountPackage est composé de deux Class de type Activity, l'une permettant l'identification à l'aide de l'identifiant et du mot de passe, et la seconde permettant de créer ou modifier le compte utilisateur.

AP2 – RoadPackage

Ce package englobe toute la partie relative aux parcours.

Description

RoadPackage est composé de sept Class dont six sont de type Activity et la dernière est une Class de type Overlay. Les différentes Activity permettent à l'utilisateur de créer, modifier et gérer ses parcours ainsi que d'effectuer sa visite et profiter de l'outil de voyage. La Class de type Overlay quant à elle permet de gérer la couche d'affichage au dessus de la carte géographique afin d'avoir la position de l'utilisateur et les différents points d'intérêt du parcours chargé.

AP3 – RoadBookPackage

Ce package englobe toute la partie relative au carnet de route.

Description
RoadBookPackage est composé de cinq Class dont trois sont de type Activity, une de type Authenticator et la dernière de type DataSource. Les différentes Activity permettent à l'utilisateur de gérer son carnet de route, de choisir une photo à attribuer à une étape du carnet de route et de publier chaque étape sur un Blog. La Class de type Authenticator permet la création et l'envoi d'un mail et la Class de type DataSource est utilisée pour la manipulation des données nécessaires à l'objet MimeMessage représentant le mail à envoyer.

MenuActivity
Cette Activity n'est intégrée à aucun des packages étant donné qu'elle fait le lien entre chacun d'eux.

Description
Cette Activity est le menu principal de l'application, c'est de cet endroit que l'utilisateur peut effectuer toutes les actions qu'il souhaite.

Package java.awt.datatransfer
Ce package est importé du SDK Java afin de permettre l'envoi de mail depuis l'application Naviboo – RoadBook. Il est indépendant au niveau de l'arborescence du projet, contrairement à tous les autres éléments qui, eux, font parti du package « com.hesso.bachelor.android.naviboo ».

Database Package
Ce package englobe toute la partie concernant les interactions avec la base de données.

Description
Database Package est composé de trois Class qui permettent à l'application de manipuler la base de données SQLite embarquée sur le mobile Android.

Problèmes rencontrés
Certains problèmes qui sont survenus lors de la création et le développement n'ont pas été mentionnés dans les différents points de ce chapitre :

Choix du DomainLogic pour les interactions avec la base de données
Le choix de la structure de l'application est important en vue d'implémenter les interactions avec la base de données.

Etant donné que je n'ai aucune expérience dans le développement Android, j'ai choisi de calquer ma structure sur celle présentée dans les différents tutoriels diffusés par Google et par la communauté Android.

J'ai donc créé deux Class responsables des manipulations avec la base de données, ces deux Class représentent les deux parties principales de l'application : Partie utilisateur et partie parcours.

Choix de la gestion des erreurs
Il est primordial de faire un choix en ce qui concerne la gestion des erreurs afin de fournir une application dépourvue de bugs inattendus.

Pour ce qui concerne cette gestion dans le cadre du prototype développé, tous les problèmes pouvant survenir de manière attendue ou non, sont canalisés par des

méthodes de récupération. L'utilisateur est informé qu'une erreur est survenue grâce à un pop-up qui s'affiche sans que l'application se ferme brusquement.

Chaque erreur est signalée dans un journal de logs afin de pouvoir identifier rapidement la source du problème et ainsi y remédier au plus vite. Toute cette partie de journal de logs est intégrée au plugin Eclipse et je l'utilise au mieux pour avoir un développement sain et pour pouvoir être le plus efficace possible lorsqu'une erreur survient.

Conclusion

La structure utilisée permet à l'application d'être évolutive et l'ajout de nouvelles fonctionnalités est relativement simple. Le fait que chaque Class respecte la convention de nommage établie au début du développement permet de retrouver très rapidement les éléments recherchés et cela rend également le développement plus agréable et plus simple.

9. Gestion des erreurs et outil de debug

Le plugin Android propose un outil très intéressant et très complet qui, entre autres, permet de gérer un journal de logs pour l'application créée.

Pour accéder à cet outil depuis Eclipse, il suffit de sélectionner la perspective DDMS dans le menu des perspectives :

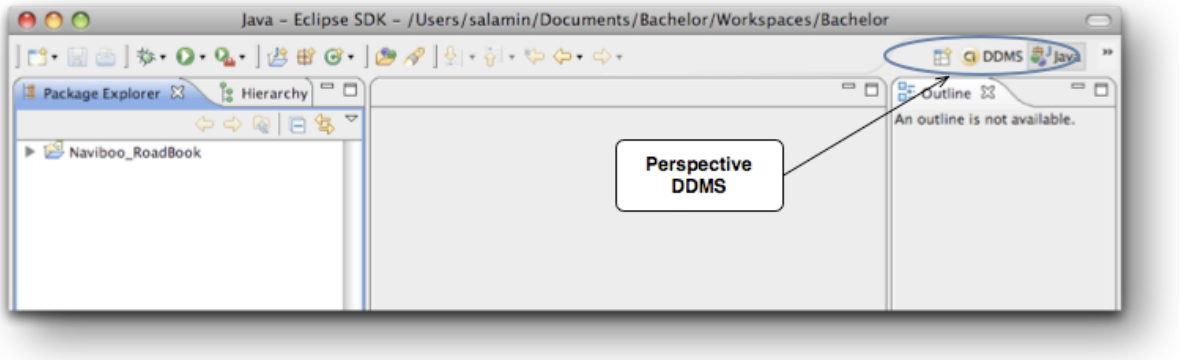

Figure 23 - DDMS : Choix de la perspective

Les possibilités offertes sont nombreuses, mais dans le cadre du projet l'élément qui s'est avéré indispensable est le journal LogCat qui se situe dans la partie inférieure de la perspective DDMS :

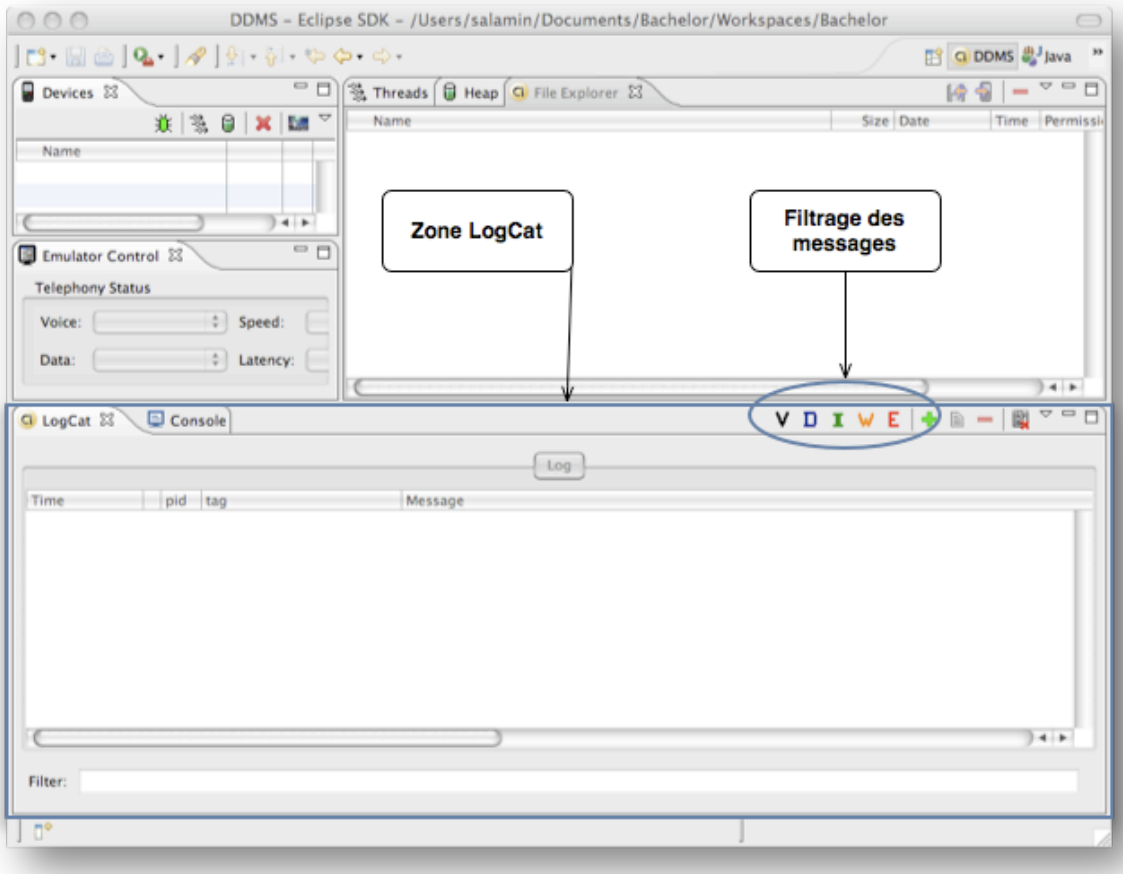

Figure 24 - DDMS : LogCat

Ce journal affiche des messages de différentes catégories :

- V : Verbeux
- D : Debug
- I : Information
- W : Avertissement
- E : Erreur

Ces informations permettent au développeur de suivre les différentes étapes effectuées lorsque l'application est déployée et également lors de son utilisation.

De nombreux messages sont générés automatiquement, mais la grande force de cet outil est de pouvoir créer des messages personnalisés permettant par exemple d'identifier la cause d'une erreur ou de faire un debug de l'application.

Au niveau du code, rien de plus simple. Il suffit d'utiliser l'objet Log qui permet d'afficher le message du type que l'on désire dans LogCat.

Un exemple de message d'erreur lorsqu'une exception est générée :

```
} catch(Exception e){
    Log.e("NaviError", "In LoginActivity.mLoginOnClick() : " + e.getMessage());
    displayErrorMessage();
}
```

Un exemple de message de debug pour connaître la valeur d'une variable à un moment choisi, en l'occurrence il s'agit de l'identifiant de l'utilisateur :

```
Log.d("NaviDebug", "In TripActivity.onCreate(), UserId : " + mUserId);
```

L'affichage dans LogCat peut être filtré pour ne voir que les messages de debug par exemple. Pour cela il suffit de cliquer sur le filtre « D » et la liste de messages sera automatiquement filtrée :

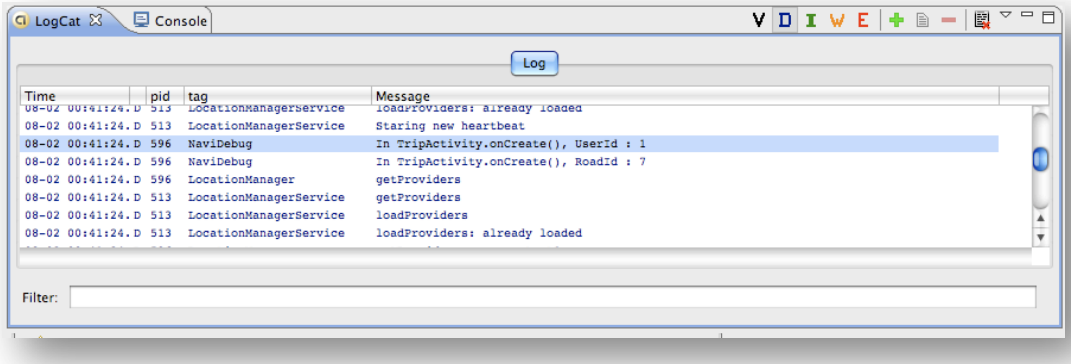

Figure 25 - DDMS : Filtrage des messages

Pour résumer, le plugin Android est vraiment très riche et certains de ses composants favorisent largement le développement.

10. Conclusion du développement

Le temps pour le développement n'a peut-être pas été assez long pour maîtriser complètement la philosophie et le fonctionnement d'Android, mais il m'a permis d'avoir un premier contact très riche en découvertes.

Un certain temps est nécessaire pour commencer à prendre en main cet environnement, mais d'une fois que cette étape est passée, l'expérience devient passionnante. La communauté qui existe autour d'Android y est pour beaucoup car il y a une sincère volonté de s'entraider et de partager les expériences vécues par chacun afin d'exploiter au mieux le SDK.

Au final je me trouve enrichi d'une expérience conséquente dans le développement pour cette plateforme et j'ai pu vivre cet apprentissage en étant au cœur de l'actualité.

Fort de cette nouvelle expérience, je peux prendre un certain recule afin de faire une analyse globale du SDK Android. Ce retour d'expérience est l'objet du chapitre suivant.

11. Table des illustrations

Bachelor | 08

Android SDK – Retour d'expérience

Chapitre 4

Google Android

Joël Salamin

Hes·so VALAIS WALLIS

Table des matières

1. Introduction

Ce chapitre s'inscrit dans le prolongement du développement, il passe en revue les éléments principaux du SDK Android proposé par Google en se basant sur l'expérience acquise lors de cette phase de développement.

Chaque partie du kit Android est analysée afin de mettre en évidence les points forts et les points faibles rencontrés. Dans la mesure du possible une comparaison avec la concurrence est faite sur la base de mon expérience. Le parallèle avec les autres environnements de développement permet de mettre en évidence les décisions prises par Google dans l'élaboration de son SDK.

L'analyse se fait selon une logique respectant au mieux le processus de développement. Cela permet de passer en revue chaque étape avec une certaine cohérence dans le déroulement. Avant même de décrire l'outil Android, il est à mon sens très intéressant d'aborder la phase d'installation du kit de développement, puis de passer brièvement en revue l'émulateur fournit.

L'objectif de ce chapitre est d'apporter un regard critique sur l'outil utilisé tout au long de cette phase de développement.

2. Kit de développement Android

Ce point décrit le kit de développement dans son ensemble. Cela comprend avant tout le choix au niveau du composant à installer, puis une brève description des éléments importants de l'outil.

Présentation

Google propose aux développeurs le choix entre deux possibilités d'installation. La première se présente sous la forme d'un SDK indépendant et la deuxième proposition consiste en un plugin à installer sur l'environnement Eclipse.

Dans le cadre du projet, je me suis orienté vers la seconde solution car je suis déjà familier avec Eclipse et surtout parce que le plugin Android fournit un outil plus complet que le SDK indépendant. La raison de cette différence est simple : l'environnement Eclipse regroupe de nombreuses fonctionnalités qui sont mises à contribution par le plugin, par exemple l'environnement Eclipse permet de disposer d'un outil de debug intégré.

Les choix pris par Google

Eclipse

Proposer un plugin pour un environnement aussi populaire qu'Eclipse permet de toucher un large public déjà familier avec cet outil. Le module Android est très simple à installer et permet de se décharger d'une installation trop contraignante. De plus le tutoriel d'installation mis à disposition par Google offre une assistance complète pour réaliser cette tâche.

Un point que je tiens à relever concerne la structure du tutoriel qui est, à mon avis, pas optimale. En effet tous les éléments nécessaires à la réussite de l'installation sont

présents, mais la manière de naviguer au travers ce document n'est pas des plus intuitive.

Hormis cet élément, le support de Google est pleinement satisfaisant pour parvenir à mettre en place son environnement avec succès.

Multiplateforme

En se tournant vers Eclipse, Google a pris une orientation très clairement multiplateforme et c'est une des grandes forces de cet environnement de développement.

Plusieurs concurrents imposent une certaine configuration pour que le développeur puisse travailler sur leur plateforme mobile. Prenons l'exemple d'Apple qui restreint la diffusion de son SDK aux possesseurs d'ordinateur Mac. Un deuxième exemple pour illustrer cette dépendance qui est imposée aux développeurs d'applications mobiles : Microsoft. La firme de Redmond, qui représente la deuxième plus grande part de marché des systèmes d'exploitation mobiles, impose également une configuration spécifique.

Il est évident que chacune de ces orientations a des avantages et des inconvénients. En se limitant à une configuration spécifique on peut offrir un outil certainement plus optimisé, mais on limite la communauté cible. Alors qu'en offrant une possibilité multiplateforme on cible une communauté bien plus large, au risque de proposer un environnement moins optimal.

Langage

Le choix du langage de programmation peut également avoir un impact sur la popularité que peut rencontrer un environnement de développement. En l'occurrence, Google s'est tourné vers une communauté de développeurs conséquente étant donné la renommée et le succès de Java.

Une fois de plus il est intéressant de faire le parallèle avec l'environnement Windows Mobile qui lui, impose d'utiliser un des langages spécifiques à Microsoft. Certes ces langages sont relativement simples à prendre en main si on possède de bonnes notions de programmation, mais il n'en est pas moins nécessaire d'adopter un des langages proposés.

Le plus restrictif des environnements au niveau du langage de programmation reste Apple avec son iPhone car tout le développement se fait en Objective-C qui est un langage propre à Apple.

Les divergences sont nombreuses entre les stratégies adoptées par les différents acteurs du marché, et chacune peut se justifier. Google fait le choix d'un des langage les plus répandu au monde, Microsoft propose un panel de langages qui offrent une certaine diversité au développeur et Apple a son propre langage, mais qui a l'avantage d'être déjà connu et maitrisé par une majorité des développeurs cibles étant donné que c'est un environnement exclusivement disponible sur Mac.

Symbian est en pleine refonte de son système et je n'ai aucune expérience sur cet environnement, il m'est donc difficile de le citer en comparaison. Il en est de même pour les autres systèmes disponibles sur le marché.

L'objectif recherché par ce document n'est pas de comparer Android à chacune des solutions concurrentes, mais de décrire les choix pris par Google avec, lorsque cela est possible, un parallèle avec mon expérience sur d'autres environnements.

3. Description de l'outil

La description de l'outil ne se porte que sur le plugin Eclipse étant donné que c'est la solution utilisée pour toute la phase de développement et c'est également la solution conseillée par Google. D'ailleurs tous les tutoriels proposés sur le site officiel se basent sur l'utilisation de l'interface Eclipse.

La base du plugin est identique à n'importe quel autre plugin disponible sur Eclipse, je ne vais donc pas m'attarder sur ce qui est commun à tous, mais me focaliser sur l'élément indispensable qui est propre à Android : DDMS[1].

Eclipse – DDMS perspective

DDMS est une perspective spécifique à Android et apporte tout ce dont un développeur a besoin pour créer une application. Pour ouvrir la perspective, il suffit de cliquer sur « DDMS » dans le menu de choix de perspective qui se trouve en haut à droite de la fenêtre Eclipse.

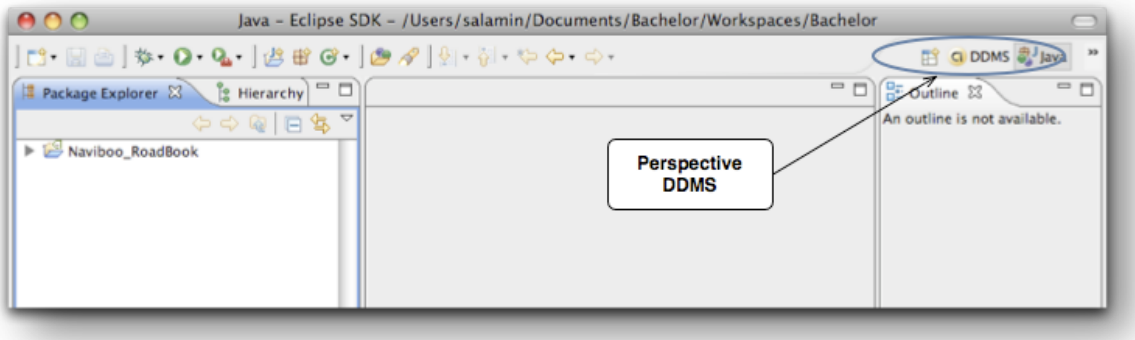

Figure 1 – Eclipse : Choix de la perspective

Chaque composant de DDMS est décrit ci-dessous.

Devices

Cet élément permet de gérer les différents appareils sur lesquels est effectué le développement. Pour chaque appareil il est possible d'afficher les processus lancés, et pour chacun d'eux il est possible de recueillir un nombre conséquent d'informations comme par exemple les Threads qui sont utilisés ou la mémoire de l'appareil qui est occupée par le processus.

Pour accéder à ces informations, il faut sélectionner le processus dont on souhaite le détail, puis activer les options que l'on désire. Une fois que tout est configuré, les données provenant de l'appareil sont affichées sur l'écran de droite.

[1] Dalvik Debug Monitor Service

Prenons un cas concret pour illustrer l'utilisation de cette partie de l'outil. Commençons par sélectionner le processus du projet « Naviboo – RoadBook » qui est lancé sur l'émulateur :

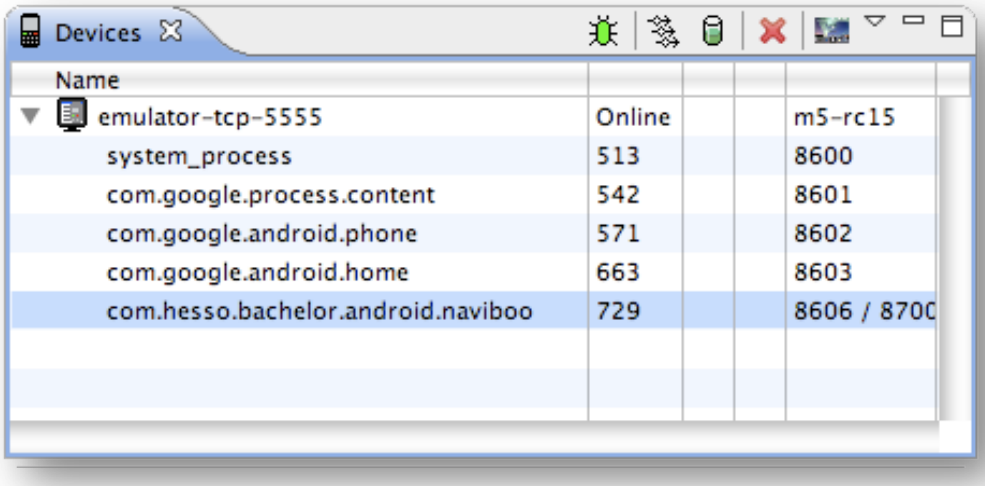

Figure 2 - DDMS perspective : Devices

On active alors la récupération des données au niveau des Threads et de la mémoire de l'appareil en cliquant sur les boutons correspondants :

Figure 3- DDMS perspective : Options de récupération de données

Sur la partie de droite de la fenêtre Eclipse, il est à présent possible de consulter toutes les informations récoltées en sélectionnant l'onglet correspondant.

Pour l'exemple, affichons la liste et le détail des Threads démarrés par l'application :

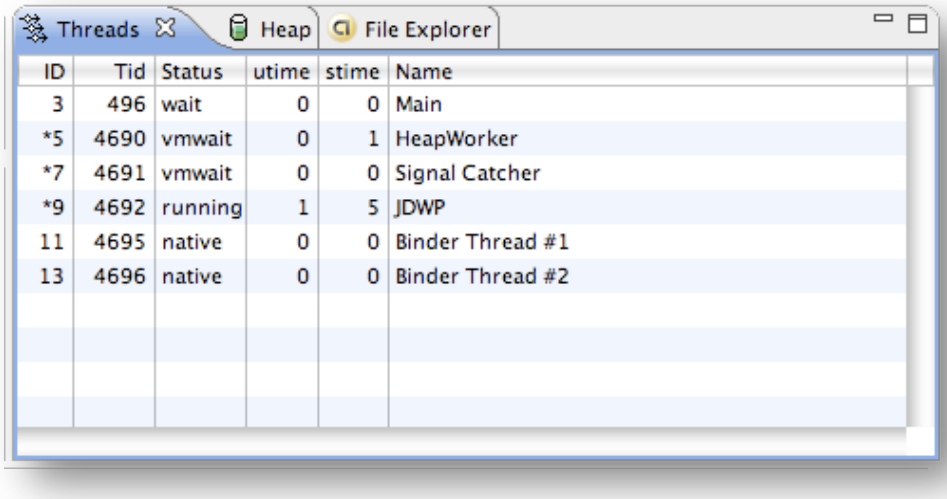

Figure 4 - DDMS perspective : Threads

Sur cette même fenêtre, il est important de mentionner l'onglet « File Explorer » qui est un explorateur de fichier dédié à l'appareil sélectionné.

Emulator Control

Cet élément permet de simuler différents comportements qui sont propres à un appareil mobile.

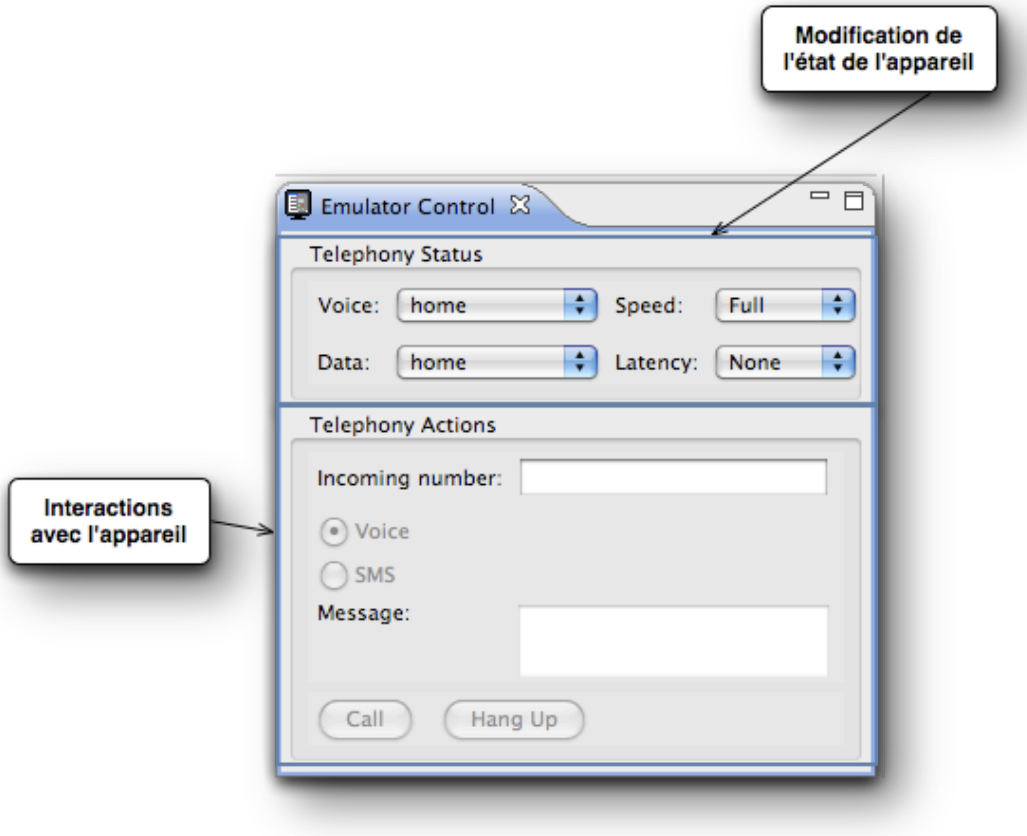

Figure 5 - DDMS perspective : Emulator Control

Cela permet par exemple d'émuler la réception d'un appel ou d'un message texte. Cet outil s'avère très utile lorsqu'on développe une application ayant des interactions et des traitements avec ce type d'événements.

LogCat/Console

Ce dernier élément offre la possibilité de suivre chaque action effectuée par l'appareil et par l'application. Le LogCat est un journal de logs permettant de consulter et personnaliser les messages générés par l'application et la Console permet d'avoir un suivi des actions relatives à l'appareil.

Un exemple d'utilisation du LogCat est disponible dans le chapitre 3.

Voici un extrait de la Console où on peut voir le détail du lancement de l'appareil ainsi que le déploiement de l'application sur celui-ci :

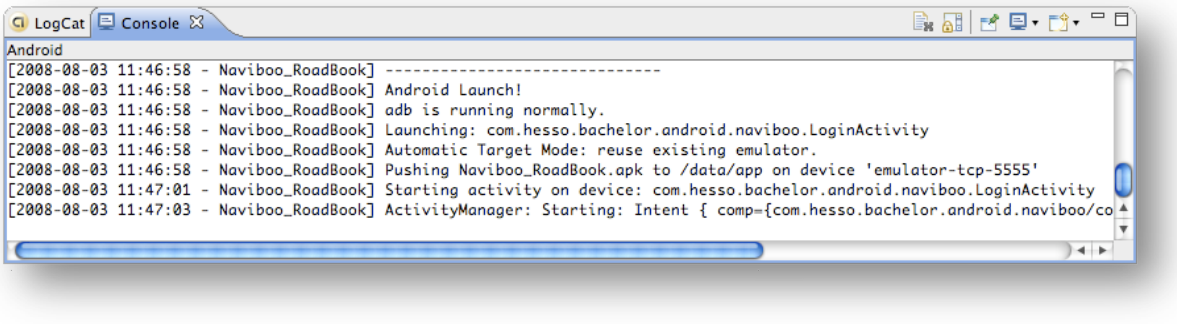

Figure 6 - DDMS perspective : Console

Conclusion

La mise en place de tous les outils nécessaires au développement d'une application Android est relativement simple et l'environnement Android propose un ensemble d'outils très complets et puissants.

Ce SDK a de quoi séduire malgré quelques lacunes sur lesquelles je reviendrai dans les points suivants.

4. L'émulateur Dalvik

L'émulateur qui accompagne le kit de développement de Google est complètement indépendant d'Eclipse. Il s'agit en fait d'une machine virtuelle hébergée sur la machine de travail et il agit comme un appareil mobile complètement indépendant.

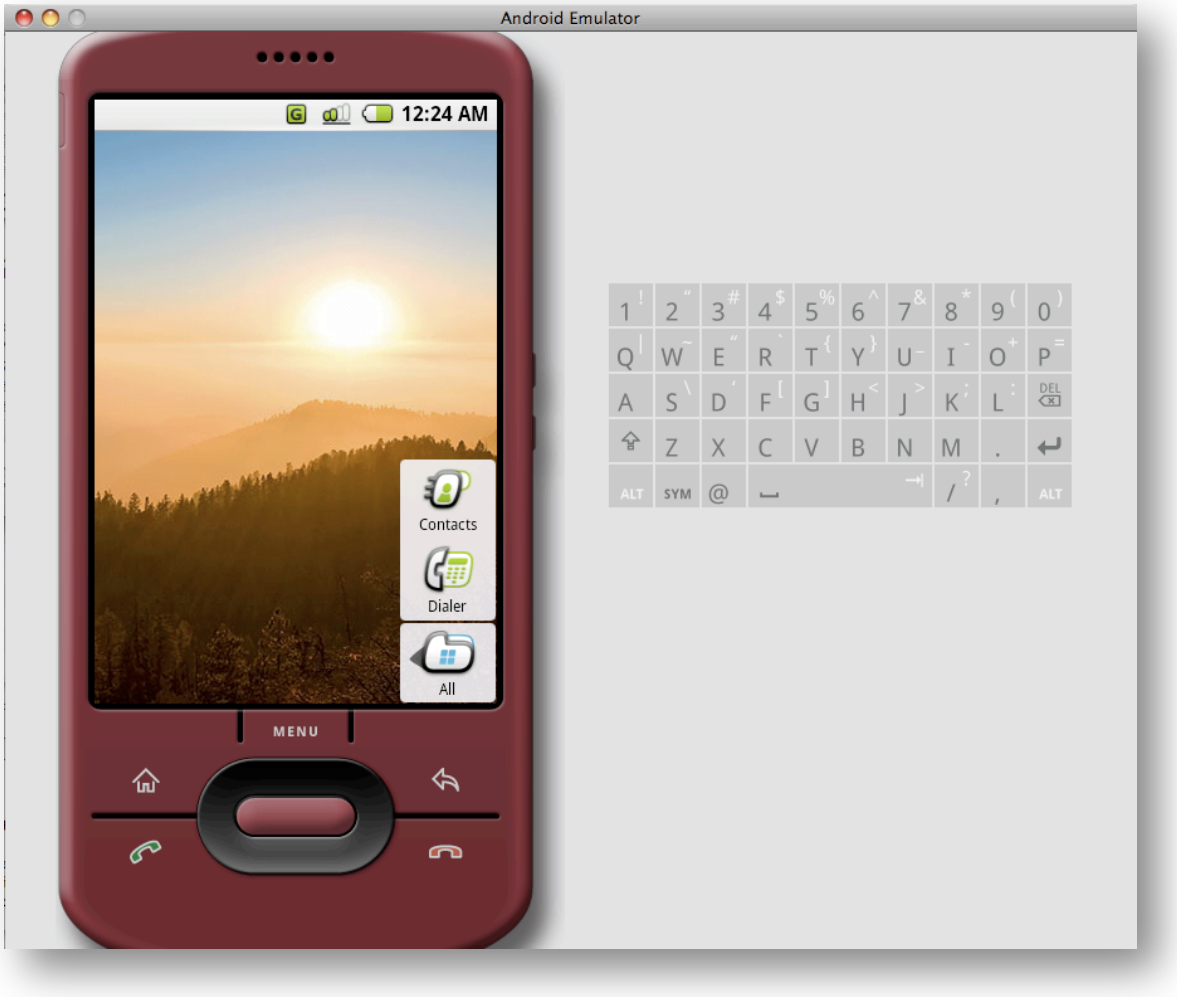

Figure 7 - Emulateur Dalvik

Il est possible de créer autant d'instance de l'appareil que l'on souhaite. Cela s'avère très pratique, par exemple lorsqu'on développe une application impliquant des interactions entre plusieurs appareils Android.

De base, certaines applications et certains composants sont intégrés au système de l'émulateur. Parmi ceux-ci se trouvent les différentes démos illustrant l'utilisation de chacun des composants proposés par le SDK Android. Cela permet de faire connaissance avec la navigation sur un appareil Android ainsi que de se familiariser avec ce système d'exploitation, son navigateur internet, son carnet d'adresse, etc…

5. Base de données SQLite

Le système de base de données SQLite est parfaitement intégré à l'environnement de développement et offre toutes les fonctionnalités nécessaires à l'élaboration d'une application mobile.

La manipulation de la base au niveau du code est simple, la syntaxe étant du SQL standard. L'utilisation de l'API sqlite est relativement aisée étant donné les tutoriels fournis par Google. Je ne vais donc pas décrire son utilisation.

Voici un exemple de requête pour faire une insertion dans la base de données :

```
// create the set of values that will be used for the insertion
ContentValues initialValues = new ContentValues();
// Put the needed values
initialValues.put(KEY_ROAD_NAME, name);
initialValues.put(KEY_ROAD_DESCRIPTION, description);
initialValues.put(KEY_ROAD_USERID, userId);

// Creation of the new road and store the roadId in a local variable
long roadId = mDatabase.insert(DATABASE_TABLE_ROAD, null, initialValues);
```

Je souhaite par contre m'attarder sur l'outil sqlite3 qui fait également parti du SDK Android et permet de manipuler la base depuis un simple Terminal ou une Console Windows.

La manière de démarrer l'outil et de le configurer sont décrites dans le document annexe « Manuel Technique – « Naviboo – RoadBook » », je ne présente ici que les éléments importants mis à disposition par cet outil.

Sqlite3

Cet utilitaire permet de manipuler la base de données sans avoir à passer par l'application. Cela s'avère très utile lorsqu'on souhaite par exemple tester la pertinence d'une requête, vérifier le contenu d'une table ou encore afficher la structure de la base.

Toutes les manipulations se font à l'aide de requêtes SQL et l'outil propose des commandes supplémentaires qui s'avèrent être très utiles pour obtenir des informations nécessaires dans le cadre du développement.

Par exemple la commande « .schema » permet d'avoir tout le script de création de la base de données, « .database » affiche le détail des fichiers composants la base et « .help » fournit le détail de chaque commande disponible :

```
Terminal — adb — 79×14
sqlite> .help
.bail ON|OFF           Stop after hitting an error.  Default OFF
.databases             List names and files of attached databases
.dump ?TABLE? ...      Dump the database in an SQL text format
.echo ON|OFF           Turn command echo on or off
.exit                  Exit this program
.explain ON|OFF        Turn output mode suitable for EXPLAIN on or off.
.header(s) ON|OFF      Turn display of headers on or off
.help                  Show this message
.import FILE TABLE     Import data from FILE into TABLE
.indices TABLE         Show names of all indices on TABLE
.load FILE ?ENTRY?     Load an extension library
.mode MODE ?TABLE?     Set output mode where MODE is one of:
                         csv      Comma-separated values
```

Figure 8 - sqlite3 : Commande ".help"

6. Structure d'une application Android

Une application Android possède une structure qui lui est propre et qui apporte un certain nombre d'avantages lors du développement.

Le fait de séparer la couche métier de la couche graphique permet d'avoir une vision plus claire des éléments à manipuler.

L'interface se crée simplement à l'aide d'un fichier XML, sans avoir à se soucier de la logique métier qui interviendra derrière. Avec ce système il est possible de remanier complètement l'interface d'une application sans que son fonctionnement en soit affecté (en gardant le nommage des éléments intact bien évidemment). Il est également possible par exemple de créer plusieurs layouts différents pour un même écran et offrir à l'utilisateur la possibilité de changer le thème de l'application selon ses goûts.

Le fait d'avoir toutes les ressources textuelles regroupées dans un même fichier permet de gérer tous les aspects linguistiques et par exemple de changer la langue de l'application simplement en modifiant ce fichier.

L'utilisation de fichiers XML pour tout ce qui concerne les ressources est également un avantage car c'est un format très léger et simple à utiliser.

Néanmoins il n'y a pas que du positif dans une telle structure pour une application. Cela nécessite avant tout une très grande rigueur dans le nommage des variables. Il faut adopter une certaine méthodologie dans la manière de travailler pour ne pas se perdre. Une fois ces points de repères assimilés, le développement s'avère être très agréable.

7. Géolocalisation

Google a mis un accent très appuyé sur tout ce qui touche à la géolocalisation et Android ne fait pas entorse à la règle. Je ne compte pas décrire l'utilisation de GoogleMaps car celle-ci est détaillée dans le chapitre 3, par contre Android fournit un fonction très intéressante qui consiste à simuler une source de données GPS.

Afin de rendre le développement et l'application complètement indépendants du matériel de géo localisation, un système de simulation d'entrées GPS est intégré à l'appareil mobile.

Avant de pouvoir définir la source de données nécessaire pour la simulation, il faut créer un simple fichier plat définissant un parcours simulé. Le contenu du fichier consiste en une succession de positions GPS qui seront interprétées à intervalles réguliers par l'objet appelé LocationProvider [2] que l'on utilisera dans le code.

Il existe déjà une simulation par défaut qui est intégrée à l'appareil, mais elle représente un parcours dans les rues de San Francisco. Il est donc intéressant de comprendre comment en créer une personnalisée.

Exemple pratique – Simulation d'entrées GPS

Pour parvenir à simuler un déplacement, il faut avant tout le fichier contenant l'itinéraire du parcours. Voici un exemple de fichier[3] qui représente un parcours dans la ville de Sierre :

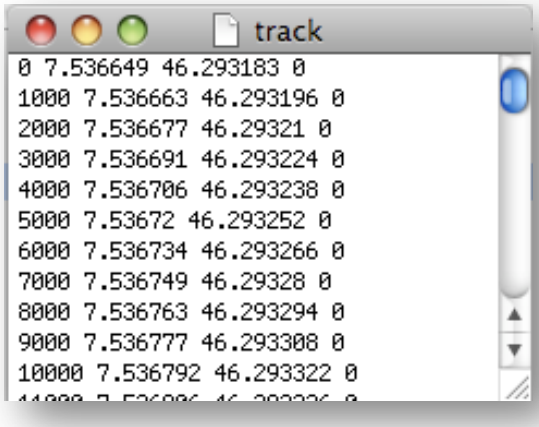

Figure 9 - Géolocalisation : Itinéraire dans la ville de Sierre

Une fois ce fichier généré, on peut créer un nouveau dossier sur l'appareil mobile afin de fabriquer un nouveau « faux » fournisseur de positionnement appelé « mock LocationProvider ».

Pour faire cela, il suffit d'utiliser l'explorateur de fichier du DDMS et de créer un nouveau dossier dans le répertoire contenant les différentes sources de données GPS (par défaut il n'y en a une seule nommée « gps »). Une fois le dossier créé, le fichier contenant l'itinéraire souhaité peut y être déposé.

---

[2] Fournisseur de positionnement
[3] Le fichier est généré par une application disponible sur Android : TrackBuilder

Pour l'exemple, un dossier nommé « sierre » est créé et le fichier généré y est déposé :

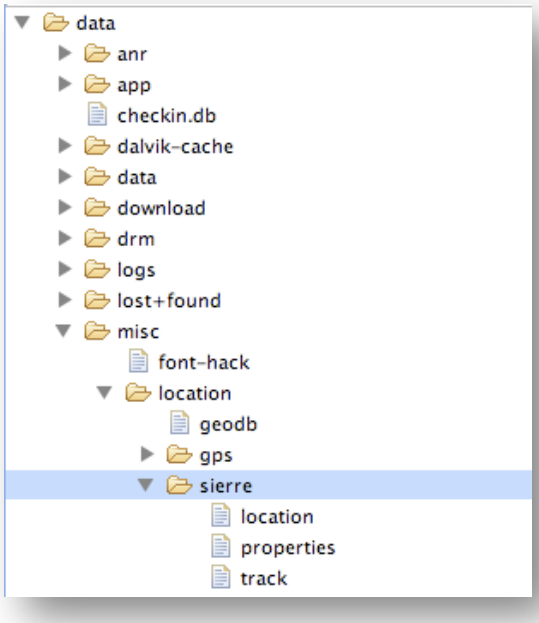

Figure 10 - Géolocalisation : Création d'un fournisseur de positionnement

Les fichiers « location » et « properties » sont simplement copiés du dossier de démonstration, ils ont pour but de décrire le comportement de l'appareil lors de l'interprétation des données.

Maintenant que le nouveau LocationProvider est créé, il est possible de l'utiliser depuis le code métier :

```
// Get the the system Location Service to manage the GPS input
mLocationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

// The selected provider is "sierre", that is a simulation of a road in Sierre/VS
LocationProvider lpProvider = mLocationManager.getProvider("sierre");
```

Nous avons à présent des données GPS qu'il suffit d'utiliser pour l'affichage de la position de l'utilisateur, centrer la carte, etc…

Cette possibilité proposée par Android est très intéressante et permet d'avoir une application complètement compatible avec un appareil muni d'un système GPS étant donné qu'il suffit de modifier la source d'entrée.

8. Conclusion

Pour conclure et faire la synthèse de cette phase de développement et de l'expérience acquise, je propose un tableau SWOT présentant un profil d'Android d'un point de vue purement technique :

Forces : - Langage Java - Fichier XML - Evolutivité - Orientation « touch » - SDK très complet - Grande communauté de développeurs	**Faiblesses :** - Absence d'assistant graphique - Mises à jour du SDK qui ralentissent les développeurs - Absence de certaines API - Structure de la documentation en ligne
Opportunités : - Séduire un grand nombre de fabricants grâce à la portabilité d'Android - Evolution du SDK pour répondre à la demande - Proposer la plateforme à tout type de téléphone mobile - Ouverture à d'autres langages de programmation	**Menaces :** - Etre boudé par les développeurs à cause de la politique utilisée - Ne pas investir le marché assez tôt - Ne pas répondre aux besoins du marchés

Le SDK Android propose de nombreux arguments de poids et les opportunités sont grandes. Mais la vision technique ne suffit pas pour évaluer les chances réelles que possède Google pour s'imposer auprès de la concurrence.

Le chapitre suivant permet de prendre du recul par rapport à la technique et de jeter un regard plus général sur la situation et l'état du marché actuel.

9. Table des illustrations

Bachelor | 08

Android – Les enjeux

Chapitre 5

Google Android

Joël Salamin

Hes·so VALAIS WALLIS

Table des matières

1. Introduction

Ce chapitre fournit une analyse complète sur la stratégie adoptée par Google au niveau de son arrivée dans le monde de la téléphonie mobile mais également une analyse sur l'évolution de ce marché. Il est également intéressant de se pencher sur la réaction de la concurrence et les mesures prises dans le but de contrer Google.

Toutes les affirmations qui suivent sont issues des différentes annonces faites soit par Google lui-même, soit par des professionnels du domaine.

2. Les enjeux pour Android

Google souhaite conquérir un marché dans lequel il n'a encore que très peu d'expérience, mais ses intentions sont claires et il n'hésite pas à annoncer un système allant complètement à contre-courant des leaders actuels que sont Nokia et Microsoft.

L'annonce du 5 novembre 2007

Google présente Android comme la première plateforme mobile vraiment ouverte et compatible avec n'importe quel matériel. L'objectif de ce système est d'innover et d'offrir à l'utilisateur une expérience surpassant largement tout ce qui est proposé sur le marché à ce jour.

Android c'est également un environnement de développement plus ouvert permettant de mettre l'accent sur la collaboration. L'objectif est d'unir la communauté de la téléphonie mobile et de palier au fractionnement qui y règne. Avec cette nouvelle plateforme les fournisseurs de matériel ainsi que les opérateurs téléphoniques pourront travailler de concert afin de créer des produits plus innovants, moins coûteux et plus rapides à mettre en place.

L'Open Handset Alliance qui est née le jour de l'annonce de Google représente le point de départ vers une unification dans le monde de la téléphonie et la plateforme Android est le premier pas vers cet objectif.

Android est publié sous une des licences open-source les plus populaires pour les développeurs[1], car elle offre une certaine liberté et flexibilité aux fournisseurs de matériel et opérateurs téléphoniques.

Android Developer Challenge

Suite à l'annonce de l'arrivée d'Android, Google a présenté un concours très astucieux. Le concept est simple : Il s'agit de créer une application mobile pour la plateforme Android en faisant preuve d'imagination et de créativité.

Parmi tous les projets reçus, Google sélectionne les 50 qui sont jugés les plus innovants et ayant le plus de potentiel. Les équipes responsables des projets retenus reçoivent une somme de 25'000$ pour faire évoluer leur application et parvenir à faire parti des 10 meilleurs projets qui seront récompensés par un montant de 100'000$ annonçant la 3ème et dernière phase de ce concours. Lors de cette dernière phase, seul les 10 meilleurs

[1] Apache Software License

sont retenus et au terme du concours, le vainqueur remporte la coquette somme de 250'000$.

Ce premier challenge (un deuxième est prévu pour fin 2008) est un coup de maître dans le sens où il provoque un engouement de la part de la communauté des développeurs et cela représente également une quantité importante d'applications qui pourront être disponibles sur la plateforme dès sa sortie.

La commercialisation

Lors de l'annonce de Novembre 2007, Google faisait mention d'un premier appareil Android avant la fin 2008. Certaines rumeurs ont émis un sérieux doute sur cette commercialisation étant donné un certain retard accumulé par Google dû à des difficultés d'organisation entre fabricants, opérateurs et système d'exploitation. Mais Google a confirmé récemment qu'il y aurait bel et bien un premier téléphone Android sur le marché avant la fin de l'année et que celui-ci sera très certainement produit par HTC étant donné que le fabricant d'appareils mobiles a confirmé sont calendrier de fin d'année.

Pour le moment, le fait de n'avoir encore aucun appareil commercialisé entretien un certain flou autour de la place que se fera Google sur le marché. C'est également un facteur important au niveau de la communauté de développeurs qui commencent à s'impatienter de travailler pour un système qui n'existe encore que sous la forme d'un émulateur.

Conclusion

De nombreux éléments sont encore inconnus et laissent planer un certain doute au sujet de la réussite ou non de Google dans sa conquête de la téléphonie mobile. Les enjeux sont impressionnants étant donné l'investissement financier fait par Google et son image est également en jeu.

La venue d'Android ne fait pas de vagues que dans le monde des médias, les acteurs du marché des plateformes mobiles sont également très attentifs car les enjeux sont aussi importants pour eux.

3. Réactions de la concurrence

Depuis quelques semaines, les annonces se succèdent et traduisent une certaine appréhension au sujet de l'arrivée du système Android sur le marché.

En me basant sur l'actualité et en évaluant la pertinence des annonces, j'ai retenu les annonces les plus importantes afin d'illustrer l'état du marché et le comportement de la concurrence.

Symbian

Le leader mondial Nokia annonce le rachat de la totalité de Symbian, il était jusqu'alors détenteur de 48% du système d'exploitation. L'objectif de cet investissement est de proposer une plateforme Symbian partiellement en Open Source pour d'une part limiter le fractionnement au niveau du système mais également dans le but d'élargir la communauté de développeurs.

Figure 1 - Symbian Foundation

Pour parvenir à proposer un système Symbian unifié et libre, Nokia a fondé la Symbian Foundation qui compte déjà de nombreux grands noms de fabricants et opérateurs.

LiMo

Le Forum LiPS (Linux Phone Standard), dont le but est de promouvoir une plateforme libre sous Linux et qui regroupe de nombreux acteurs de la téléphonie, annonce qu'il va rejoindre la LiMo Foundation.

Figure 2 - LiMo Foundation

La LiMo Foundation a comme objectif de créer un système d'exploitation mobile basé sur Linux qui pourra être intégré sur n'importe quel appareil. Cette fusion entre deux représentants de la communauté Linux traduit un désir d'accélérer les choses.

iPhone 3G

Apple propose maintenant son iPhone nouvelle génération qui palie aux nombreuses faiblesses dont souffrait la première version de son appareil. Cette arrivée sur le marché permet à Apple de prendre une longueur d'avance face à Android et la grande force de l'iPhone est de pouvoir se reposer sur une plateforme de diffusion d'application riche de plusieurs milliers de produits.

OpenMoko

Ce système d'exploitation basé sur le noyau Linux est commercialisé depuis peu de temps sur un appareil entièrement libre, le Neo Freerunner.

Figure 3 - OpenMoko

Ce téléphone ne vise clairement pas les parts de marché d'Apple ou celles que convoite Google, mais c'est un projet qui rencontre un certain succès auprès des développeurs étant donné que toute la couche matérielle ainsi que la couche logicielle sont Open Source.

Windows Mobile 7

Le nouveau système d'exploitation mobile de Microsoft est annoncé à partir du premier trimestre 2009. Pour le moment aucune image officielle n'a été diffusée mais selon les annonces de Microsoft, la plateforme a subit un remaniement complet afin de proposer une toute nouvelle expérience à l'utilisateur.

Blackberry Thunder

L'entreprise RIM connue pour son système Blacberry OS qui connaît un grand succès sur le marché des professionnels a annoncé la sortie d'un nouvel appareil surnommé « Thunder ».

La grande nouveauté réside dans le fait que pour la première fois un Blackberry sera équipé d'un écran tactile et RIM annonce un système très intuitif et très accessible. Pour le moment aucune date précise n'est donnée, mais l'appareil devrait être commercialisé d'ici la fin 2008.

Conclusion

Le marché des plateformes mobiles est en effervescence et la fin de l'année 2008 promet d'être passionnante. Tous les acteurs du marché cherchent à s'entourer de grands noms dans le but de proposer la meilleure plateforme et le système le plus innovant possible. D'ailleurs certains grands fournisseurs et opérateurs ne se cachent pas d'être membre de plusieurs de ces alliances pour prendre part à l'évolution du marché.

Une grande tendance vers l'Open Source est à signaler et la fragmentation du marché tend à diminuer suite aux réactions de la concurrence.

4. Avis des analystes

Il est intéressant d'avoir un regard et une analyse de la part des professionnels afin de mieux comprendre les enjeux pour chacun. Les sources d'où proviennent les citations suivantes sont répertoriées dans le chapitre 7.

Gartner

« Symbian devrait conserver une longueur d'avance chez les particuliers sur le marché des smartphones : "La plateforme Windows Mobile reste, quant à elle, toujours très attractive pour les entreprises, notamment en raison de son intégration avec les applications Microsoft. (...) RIM devrait maintenir sa position sur le marché Nord américain et sur le segment de l'e-mail mobile avec sa solution Blackberry." La percée d'Apple reste fulgurante : L'iPhone "devrait faire son entrée dans le top five des smartphones les plus vendus dès la fin de l'année 2008."

Toujours aussi fragmenté, Linux n'en reste pas moins un acteur avec lequel il faudra compter, mais pas avant qu'une offre cohérente ne se dessine, soit après 2010 : "Android de Google pourrait venir se substituer à LIPS ou LIMO." Gartner voit plutôt Android occuper le marché résidentiel. »

ABI Research

« Linux devrait raffermir sa position sur le marché des smartphones grâce au soutien de la LiMo Foundation et de l'Open Handset Alliance. ABI Research pense que les actuels éditeurs de systèmes d'exploitation pour smartphones devraient rapidement se heurter à la compétence technique de ces deux consortiums.

D'ici 2013, nous pensons que Linux réalisera 23 % de part de marché et s'imposera comme la solution la plus utilisée après Symbian. LiMo et Android devraient se tailler la part du lion sur le marché des solutions Linux»

J. Gold Associates

« Symbian et Android vont fusionner pour ne former qu'un seul système d'exploitation mobile. Cette prévision se base non seulement sur les difficultés de Google à créer Android, mais aussi sur les avantages d'une éventuelle fusion pour Symbian.

Symbian contrôle une énorme part du marché des Smartphones, mais pourrait utiliser ce coup de pouce pour cimenter sa position avec la communauté Open Source et ne pas seulement apparaître comme un coup de pub de Nokia.

Cette fusion n'aurait pas lieu avant 18 à 24 mois mais lorsqu'elle aura lieu, le nouvel OS sera une référence dans son domaine et portera un coup dur à ses concurrents notamment Windows Mobile et RIM »

Conclusion

Les analystes se rejoignent sur le fait que Linux tend à prendre de l'importance sur le marché et que l'arrivée d'Android n'y est pas anodine. Quoiqu'il en soit, le marché va subir une évolution sans précédant. La fin de l'année 2008 et principalement l'année 2009 s'annoncent mouvementées au niveau des systèmes d'exploitation pour mobiles.

5. Mon point de vue

Pour moi, il y avait le monde de la téléphonie mobile avant Apple et Google, et ce monde va changer de visage suite à leurs arrivées. Je qualifie cette période 2007-2009 de révolutionnaire au point de vue des systèmes d'exploitation mobiles et Google y joue un grand rôle.

Avant la révolution

Le marché est solidement dominé par Nokia qui partage cette domination avec Microsoft qui est très présent également. Certains challengers se disputent les miettes du marché, avec RIM qui s'en sort d'ailleurs très bien en réalisant un volume de ventes considérable dans le secteur des professionnels

De nombreux projets tentent de faire bouger les choses avec notamment des projets comme LiMo et OpenMoko qui s'efforcent de mettre en avant une plateforme Open Source. Mais malheureusement le marché du libre est tellement fragmenté qu'aucune entité ne parvient réellement à inquiéter les deux géants bien installés sur leurs parts de marché.

Les systèmes d'exploitation mobiles évoluent mais sans vraiment révolutionner quoique ce soit, ni au niveau de l'interface, ni au niveau de l'utilisation d'ailleurs.

Seuls certaines exceptions, comme le fabricant HTC, tentent d'amener quelque chose de nouveau dans l'expérience de utilisateur.

En résumé, les forces en présence sont solidement installées et n'ont pas de raison de se mettre en danger en essayant de remodeler trop profondément leurs systèmes qui connaissent un large succès auprès des consommateurs.

2007-2009, la révolution est en marche

L'année 2007 marque l'arrivée d'Apple dans le monde des plateformes mobiles avec un appareil aux antipodes de ce qui se fait chez Nokia ou Microsoft. L'iPhone renvoie le stylet au placard et offre un nouveau regard sur le mobile. L'utilisateur se trouve devant une interface simple et intuitive, la possibilité de naviguer sur internet avec un outil agréable à utiliser et un appareil jouissant d'un design très soigné. Apple a, à mon sens, lancé le premier pavé dans la marre si tranquille que représentait le monde du mobile.

Puis fin 2007, Google annonce l'arrivée de sa plateforme Android, la création d'une alliance autour de ce projet et le choix de proposer un système sous le signe de l'Open Source.

Le marché de la téléphonie mobile entre alors en ébullitions et pas un jour ne passe sans une nouvelle annonce ou rumeur. A mon avis, la brèche ouverte par l'arrivée d'Apple et l'annonce du géant Google a bouleversé la sérénité des forces en présence. D'ailleurs l'annonce de Nokia qui rachète l'intégralité de Symbian, fonde une alliance et passe son système en Open Source, c'est un signe très clair d'une appréhension concernant l'avenir de son système s'il ne prend pas rapidement une nouvelle voie pour tenir tête à Apple et Google.

C'est une grande opportunité pour que la mentalité change et que la téléphonie affiche un nouveau visage. Je pense que Google a une très grande carte à jouer et, fort de sa

grande expérience dans le domaine du marketing, le géant peut se faire une bonne place sur ce marché.

Après la révolution

Il est très compliqué de pouvoir dire comment se présentera le marché des plateformes mobiles dans les années à venir, mais je pense qu'il sera moins fragmenté au vue des différentes associations et rachats qui se font depuis quelques semaines. Un élément dont je suis certain c'est que l'utilisateur sera le premier bénéficiaire de ce que j'appelle « la révolution de la téléphonie ».

Faisant parti de la communauté des développeurs, je pense également que nous profiterons de ces changements car si les systèmes deviennent plus ouverts, cela induit des possibilités de développement plus grandes et la communauté se verra également enrichie.

Google Android doit encore faire l'objet de quelques retouches et il lui faut surtout arriver sur le marché pour se faire une idée du comportement des consommateurs à son égard. Je pense qu'un large public sera séduit, car le système est vraiment conçu pour l'utilisateur. La grande question qui à mon avis jouera un rôle primordial dans le succès de Google, c'est son prix.

6. Analyse SWOT

Ce point fait une synthèse de tous les points forts et les points faibles de Google, ce qui permet d'avoir une vision globale sur la position qu'il occupe face à la concurrence :

Forces : - Grande renommée - Alliés puissants - Compétences marketing redoutables - Plateforme innovante - Kit de développement puissant	**Faiblesses :** - Aucun appareil commercialisé - Certaines lacunes dans le SDK - Aucune expérience dans la téléphonie - Manque de communication auprès de la communauté de développeurs
Opportunités : - Prendre des parts de marché - Apporter un nouveau regard sur les systèmes mobiles - Envahir le monde de l'internet mobile	**Menaces :** - Réactions de la concurrence - Désintérêt des développeurs si les promesses ne sont pas tenues

7. Conclusion

Le monde des systèmes d'exploitation mobiles est passionnant et dynamique, Android tient le rôle d'épouvantail depuis l'annonce de son arrivée. Les réactions du marché sont nombreuses et les enjeux sont considérables pour chacun des acteurs déjà présents mais également pour Google qui investit beaucoup dans son système.

Chacun des acteurs met tout en œuvre pour sauver sa place et les risques pris par Google ne laissent personne indifférent. Si Android parvient à se faire une place au soleil, il aura une position idéale dans sa quête vers l'internet mobile.

8. Table des illustrations

Bachelor | 08

Conclusion

Chapitre 6

Google Android

Joël Salamin

Hes·so VALAIS WALLIS

Table des matières

1. Respect du cahier des charges de l'application

Au point de vue du développement, je suis entièrement satisfait de la manière dont se sont déroulées les différentes étapes.

Les efforts fournis lors de la rédaction du cahier des charges de l'application ont considérablement participé à la réussite du développement. Le fait de pouvoir m'appuyer sur un document aussi complet tout au long de la phase de développement m'a évité un grand nombre de réflexions relatives à l'application et cela m'a permis de me concentrer pleinement sur les aspects techniques.

Au niveau du respect du cahier des charges, dans l'ensemble tout a pu se dérouler comme selon la planification à l'exception de certains éléments qui ont dû être modifiés ou remplacés.

Modification du cahier des charges

Les différentes itérations à réaliser m'ont amené à explorer un très large éventail de fonctionnalités d'Android, je considère donc que les objectifs fixés sont pleinement atteints à ce niveau là.

Il est malgré tout nécessaire de signaler certains écarts par rapport au document original :

Must Have

La totalité de ces éléments a pu être implémentée selon le cahier des charges, malgré certains problèmes rencontrés lors du développement et certaines adaptations aux composants fournis par Google Android.

Nice to Have

Etant donné le peu de temps à disposition pour le développement de l'application, la majorité de ces éléments n'a malheureusement pas pu être implémentée.

Le temps qu'il me restait au terme de toutes les itérations prioritaires m'a donné la possibilité d'explorer plus en profondeur l'API GoogleMaps. Je suis donc parvenu à implémenter l'itération 2.5.

Un autre élément qu'il m'a été possible de réaliser concerne l'itération 3.3. Android fournit un élément visuel appelé Gallery qui permet de faire défiler et sélectionner une image dans une galerie. Le choix de l'image à joindre lors de la création d'une étape se fait à présent par l'utilisation de ce composant.

2. Déroulement du projet de Bachelor

Grâce à un encadrement très professionnel et à des objectifs clairement définis, le projet s'est déroulé de manière idéale. Je savais exactement ce qui était à faire et j'ai pu m'investir pleinement lors de chaque phase du travail.

J'ai pu profiter entièrement de chaque étape de ce projet de Bachelor et chacune m'a apporté une expérience considérable.

WP1 – Analyse et recherche

La phase de recherche a été dynamique et riche en informations car Android est au cœur de l'actualité. Il n'a pas toujours été évident de filtrer la masse importante de données afin d'en retirer l'essentiel. Une fois les premiers jours de recherche passés et la veille technologique mise en place, c'était un pur plaisir de vivre au rythme de l'actualité.

J'ai profité de mon grand intérêt envers cette technologie pour recueillir le maximum d'informations dans le but d'établir un cahier des charges le plus complet possible.

WP2 – Développement d'une application Android

La découverte de l'environnement de développement a été pour moi un des meilleurs moment de ce travail. J'ai découvert une plateforme intuitive et agréable.

Elaborer et créer une application pour ce système d'exploitation m'a procuré un grand plaisir et de nouvelles connaissances qui pourront sans aucun doute me servir pour mon avenir.

WP3 – Analyse basée sur le vécu et prise de position sur l'outil

L'expérience récoltée tout au long de la phase de développement et de recherche me permet d'analyser Android avec un regard différent.

Il y a certes tout l'aspect marketing qui entre en compte, mais le fait d'avoir utilisé l'outil et exploré la communauté Android apporte une autre dimension qu'il est intéressant de connaître.

Conclusion

Je pense que tous les objectifs fixés ont été atteints avec succès. Je me suis enrichi d'une expérience considérable dans le développement sur la plateforme Android. En plus de ça j'ai découvert le monde des systèmes d'exploitation mobiles au delà de ma simple expérience de l'environnement de développement Windows Mobile.

J'ai profité de ce travail pour apprendre à travailler de manière complètement autonome et le résultat obtenu me remplit de fierté.

3. Bilan personnel

Je tire un bilan plus que positif de ce travail de Bachelor. J'ai pu mettre en pratique une partie des connaissances acquises lors de ma formation HES et me servir de mon expérience dans le but d'acquérir des nouvelles compétences dans différents domaines.

Je me suis formé sur un outil qui m'était complètement inconnu et je suis parvenu en moins de 8 semaines[1] à développer une application Android pleinement fonctionnelle.

L'esprit d'initiative est indispensable à la réussite d'un projet aussi dépendant de l'actualité que celui-ci l'a été et je pense que je suis parvenu à l'exploiter pleinement pour atteindre les objectifs qui étaient fixés.

Pour résumé je retire de ce travail une énorme satisfaction et un sentiment de réussite à la vue des documents réalisés et du projet créé. Je tire des leçons importantes de cette expérience et je suis persuadé qu'elle me servira pour mon avenir professionnel.

Remerciements

Je tiens à remercier toutes les personnes qui m'ont soutenu tout au long de ce travail et tout particulièrement Yann Bocchi qui m'a fournit un encadrement et une aide importante dans la réalisation de ce travail de Bachelor.

Sans oublier un grand remerciement pour ma famille et mon amie pour leur présence à mes cotés.

[1] Semaine à plein temps, soit 45h

4. Déclaration sur l'honneur

Je déclare, par ce document, que j'ai effectué le travail de diplôme ci-annexé seul, sans autre aide que celle dûment signalée dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de diplôme, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail.

Sierre, le 4 août 2008

Joël Salamin

Bachelor | 08

Bibliographie

Chapitre 7

Google Android

Joël Salamin

Hes·so VALAIS WALLIS

Table des matières

1. Historique et Description

Android mobile device Platform) :

http://en.wikipedia.org/wiki/Android_%28mobile_device_platform%29

Information sur le service Android :

http://logiciels.zorgloob.com/android-voir.php

Google dévoile son système d'exploitation pour mobiles :

http://www.trends.be/fr/economie/high-tech/12-1637-44034/google-devoile-son-systeme-d-exploitation-pour-mobiles.html

Driving Innovation on new Platforms : Android and the Open Handset Alliance :

http://www.exxplain.com/TrainingDetails.aspx?Ticket=de5d0d74-0f06-4a4f-be8e-ec31f80ff968

Google lève le voile sur Android :

http://www.cnetfrance.fr/news/mobilite/google-leve-le-voile-sur-android-39381399.htm

Le journal des mobiles de Laurent Redondo Sanchez :

http://lavieameilleurgout.mint.be/index.php/2008/06/04/le-journal-des-mobiles-de-laurent-redondo-sanchez/20085230

Google Apps Engine ouvert à tous :

http://www.journaldunet.com/developpeur/breve/27543/google-apps-engine-ouvert-a-tous.shtml

Google musèle les critiques sur sa plateforme Android :

http://www.informaticien.be/index.ks?page=news_item&id=5202

Avec Android, Google réinvente Java Me ? :

http://blog.developpez.com/adiguba?title=avec_android_google_reinvente_java_me

Android sera 100% open source :

http://www.silicon.fr/fr/news/2008/06/03/google_android_sera_100_open_source_

Android utilisera 3 licences :

http://phoneandroid.fr/2008/06/adroid-utilisera-3-licences.html

What Google's Android means to the tech industry :

http://wireless.itworld.com/4261/google-android-dr-080213/page_1.html

Google's Rich Miner Talks Android And Open Source :

http://www.talkandroid.com/109-google-mobile-rich-miner-android-interview/

La guerre des Smartphones :

http://cidotis.com/blog/?p=305

Quid d'Android :

http://musaraign.typepad.com/musa/2008/06/quid-dandrod.html

2. Présentation et Démo

Google I/O, la vidéo intégrale présentant la plateforme Android :

http://www.android-info.com/actualite_android/google-io-la-video-integrale-presentant-la-plateforme-android/

La menace Android arrive et compte bien rivaliser l'iPhone et Apple :

http://www.trackbusters.fr/actualites/20080530-google-android-menace-iphone-apple-1.html

Première démonstration publique du HTC Dream :

http://www.businessmobile.fr/actualites/technologies/0,3800003790,39381392,00.htm

3. Fonctionnalités et technique

Google dévoile son système mobile Android en images et en vidéos :

http://www.mobinaute.com/85560-sdk-google-systeme-mobile-android-images-videos.html

Android sera doté d'un navigateur Web performant :

http://www.francemobiles.com/actualites/id/200805221211357113/android_sera_dote_d_un_navigateur_web_performant.html

Google nous dévoile quelques détails de son système mobile :

http://www.businessmobile.fr/actualites/technologies/0,39044306,39381810,00.htm?xtor=RSS-10021

Aperçu de Google Android avec Andy Rubin :

http://www.android-info.com/actualite_android/aprecu-de-google-android-avec-andy-rubin/

4. Développement et implémentation

Android – An Open Handset Alliance Project :

http://code.google.com/android/

Android Development Community :

http://www.anddev.org/index.php

Dan Morrill builds a simple application on the Android Platform :

http://googleandroidblog.blogspot.com/

Sample Applications for the Android Platform :

http://code.google.com/p/apps-for-android/

Android tutorial by Geetha Ganesan :

http://geeth.ganesan.googlepages.com/android-tutorial

Undroid, plugin Android pour NetBeans :

http://www.pointgphone.com/undroid-plugin-android-netbeans-461

On pourra développer des applications Android dans d'autres langages :

http://www.frandroid.com/392/on-pourra-developper-des-appli-android-dans-dautres-langages/

Google Maps pour mobile avec Ma Position (beta) :

http://www.google.com/gmm/mylocation.html?hl=fr

Google Maps « Ma Position » remplace votre GPS :

http://www.pointgphone.com/google-maps-ma-position-remplace-votre-gps-175

Comment simuler un déplacement sur GoogleMap Android sans véritable GPS :

http://blog.molib.fr/2008/03/comment-simuler-un-dplacement-sur.html

Live Camera Previews in Android :

http://www.tomgibara.com/android/camera-source

ModMyGphone – Your premier Google Phone community :

http://modmygphone.com/forums/forumdisplay.php?f=4

Blogger on the Go :

http://www.blogger.com/mobile-start.g

iGoogle Mobile devient plus simple à utiliser sur Smartphones et iPhone :

http://www.mobinaute.com/143800-igoogle-mobile-utiliser-smartphones-iphone.html

Centre d'aide de Google Maps :

http://maps.google.fr/support/bin/topic.py?topic=13940

How to Program Google Android :

http://blogoscoped.com/archive/2007-11-19-n27.html

Le blog francophone Android :

http://androidfr.blogspot.com/

Building an Android file browser :

http://linuxdevices.com/articles/AT6247038002.html

Encoder une image au format Base64 en Java :

http://home.tele2.fr/bobremy/code_source/java/java_conversion_image_Base64.html

Développons en Java – JavaMail :

http://www.jmdoudoux.fr/java/dej/chap045.htm

Web Services Providers :

http://seekda.com/service_details?uri=http%3A%2F%2Fwww.webservicex.com%2FCurrencyConvertor.asmx%3Fwsdl

5. Concurrence

Mobilinux :

http://fr.wikipedia.org/wiki/Mobilinux

OpenMoko :

http://fr.wikipedia.org/wiki/OpenMoko

Windows Mobile :

http://fr.wikipedia.org/wiki/Windows_Mobile

Symbian OS :

http://fr.wikipedia.org/wiki/Symbian_OS

LiMo marque un point sur Google Android :

http://www.zdnet.fr/actualites/telecoms/0,39040748,39381035,00.htm

LiMo et Android vont dominer le camp Linux Mobile :

http://www.generation-nt.com/abi-research-etude-domination-limo-android-linux-mobile-actualite-101401.html

Linux ne fait pas peur à Symbian et à Microsoft :

http://www.silicon.fr/fr/news/2008/06/09/mobile_linux_ne_fait_pas_peur_a_symbian_et_a_microsoft

Nokia reste sceptique sur la plateforme Android de Google :

http://www.neteco.com/141200-nokia-sceptique-plateforme-android-google.html

Rencontre avec Andy Rubin, papa de l'anti-iPhone de Google :

http://www.trends.be/fr/economie/high-tech/12-1637-45562/android---rencontre-avec-andy-rubin--papa-de-l-anti-iphone-de-google.html

BlacBerry Partners Fund – 150 millions de dollars pour attirer les développeurs :

http://www.pointgphone.com/blackberry-partners-fund-680

Symbian against Open Source (Google Android) :

http://phonenews.blogs.sapo.pt/122680.html

L'iPhone se libère avec la seconde version de son firmware :

http://www.belgiquemobile.be/2008/06/09/liphone-se-libere-avec-la-seconde-version-de-son-firmware/

Nokia voit Linux progresser sans menacer sa domination :

http://www.capital.fr/actualite/Default.asp?source=RE&numero=298040&Cat=SOI&numpage=1

Un smartphone à l'OS libre, pour fin juin :

http://www.silicon.fr/fr/news/2008/06/16/un_smartphone_a_l_os_libre_pour_fin_juin

Photos de l'interface du Garmin Nuvifone :

http://www.mobifrance.com/news/2008-06-18/id12046/Photos-de-l-interface-du-Garmin-Nuvifone/

Java Me et .Net les plateformes mobiles les plus utilisées :

http://fr.itprofessional.be/news.cfm?id=87024&mxp=206

Symbian n'excluerait pas un rapprochement avec Google :

http://www.generation-nt.com/symbian-google-hypothese-rapprochement-actualite-122851.html

6. Parallèle avec la concurrence

Premières images d'Android :

http://techno.branchez-vous.com/actualite/2008/05/premieres_images_dandroid.html

Open Source, principal argument de Google pour contrer l'iPhone :

http://www.trackbusters.fr/actualites/20080604-android-open-source-google-argument-iphone-1.html

Interview de Florent Stroppa (Voxmobili) :

http://www.frandroid.com/403/interview-de-florent-stroppa-voxmobili/

Andy Rubin, responsable d'Android, interviewé par l'Expansion :

http://www.android-info.com/actualite_android/andy-rubin-reponsable-dandroid-interviewe-par-lexpansion/

Les Smartphones sous Android, principaux concurrents de l'iPhone 3G :

http://www.cnetfrance.fr/news/mobilite/telephones-android-principaux-concurrents-iphone-3g-39381678.htm

Samsung Omnia – Vidéo de l'interface :

http://www.blogeek.ch/index.php?2008/06/16/4362-samsung-omnia-sgh-i900-video-de-l-interface

Interview de Lucas Bonnet sur Android et OpenMoko :

http://www.frandroid.com/414/interview-de-lucas-bonnet-sur-android-et-openmoko/

Symbian vs Android, quelle sera la plateforme mobile gagnante :

http://www.businessmobile.fr/actualites/analyses/0,39044174,39382060,00.htm

Un vrai gPhone pour réussir :

http://www.macgeneration.com/unes/voir/127134/google-un-vrai-gphone-pour-reussir/1

Android :

http://sulfureetcontreculture.blogspot.com/2008/07/androd.html

Google Android peut-il faire mieux que l'App Store de l'iPhone :
http://www.businessmobile.fr/actualites/technologies/0,39044306,39382402,00.htm

7. Avenir et Evolution

Selon Gartner, Android pourrait se substituer à LiMo et LIPS :

http://www.toolinux.com/news/revue_de_presse/linux_mobile_selon_gartner_android_pourrait_se_substituer_a_limo_et_lips_ar10632.html

Android, plateforme mobile la plus prometteuse selon les lecteurs de BusinessMobile.fr :

http://www.businessmobile.fr/actualites/analyses/0,39044174,39381506,00.htm

Linux se fraie un chemin sur le marché des Smartphones :

http://www.vnunet.fr/fr/news/2008/06/04/linux_se_fraie_un_chemin_sur_le_marche_des_smartphones_

8. Rumeurs

Android Market, un seul et même endroit pour télécharger les applications :

http://www.android-info.com/actualite_android/android-market-un-seul-et-meme-endroit-pour-telecharger-les-applications/

Un terminal HTC sous Windows Mobile 7 début 2009 :

http://www.generation-nt.com/rumeur-pdaphone-htc-windows-mobile-7-2009-actualite-107331.html

Windows Mobile 7 - sortie début 2009 :

http://www.macgeneration.com/news/voir/130666/windows-mobile-7-sortie-debut-2009

Téléphone HTC sous Windows Mobile 7 en 2009 et pas d'Android HTC :

http://www.android-info.com/concurrents-de-google-android/telephone-htc-sous-windows-mobile-7-en-2009-et-pas-dandroid-htc/

Les raisons cachées du retard de Google :

http://www.zdnet.fr/blogs/2008/06/23/google-android-les-raisons-cachees-du-retard/

Vers une fusion des systmes Android et Symbian :

http://www.generation-nt.com/rumeur-speculation-analyste-fusion-systemes-android-symbian-actualite-128641.html

9. Divers

Android, panorama de la création d'applications mobiles :

http://www.internetactu.net/2008/06/03/android-panorama-de-la-creation-dapplications-mobiles/

Maps And Location Based Applications Hold The ADC Majority :

http://www.talkandroid.com/94-google-maps-location-apps-android-challeng/

Les 10 applications les plus intéressantes d'Android :

http://phoneandroid.fr/2008/05/jdn-pronostique-sur-vos-futures.html

Lamity – un clone amélioré de Second Life sur Android :

http://phoneandroid.fr/2008/05/lamity-un-clone-amlior-de-second-life.html

Les meilleures applications de géolocalisation sur Android :

http://www.renalid.com/2008/06/11/les-meilleures-applications-de-golocalisation-sur-androd/

Bachelor | 08

Manuel utilisateur

Annexe 1

Google Android

Joël Salamin

Hes·so VALAIS WALLIS

Table des matières

1. Introduction

Naviboo – RoadBook est une application pour tout téléphone fonctionnant avec la plateforme Google Android.

Cette application vous permet de profiter d'un outil de visite touristique complètement dynamique et qui s'adapte à vos désirs. Il vous est possible de découvrir une région comme vous ne l'avez encore jamais vécu auparavant. Naviboo vous guide où vous le souhaitez tout en vous laissant une indépendance complète.

Naviboo ne se résume pas à un simple guide touristique de poche, c'est également un carnet de route. En effet vous pouvez à tout moment prendre une photo d'un lieu, un monument ou simplement un paysage qui vous plait lors de votre visite et y ajouter un commentaire. Chaque étape du carnet de route sera publiée sur votre Blog, cela vous permettra de revivre vos visites et d'en garder un souvenir multimédia, mais également de faire vivre votre voyage à vos amis et votre famille.

2. Créer un compte

Avant de pouvoir profiter de Naviboo – RoadBook vous devez vous identifier et donc avoir un compte.

Pour créer un nouveau compte, appuyez sur le bouton « Create Account » à l'écran d'accueil.

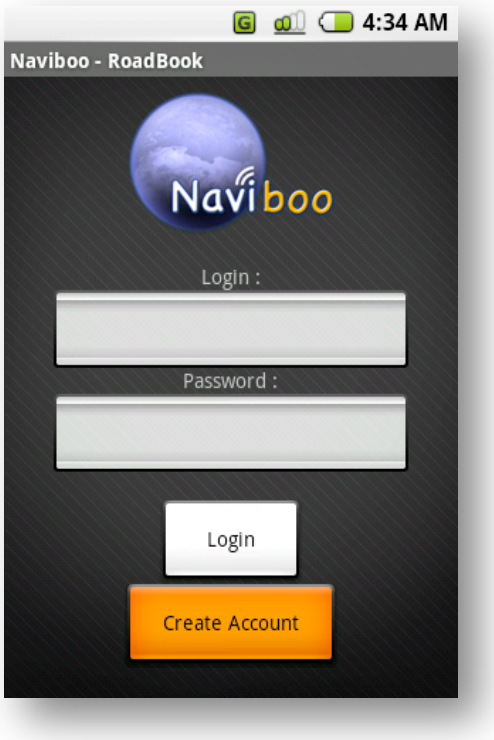

Figure 1 - Créer un compte

Vous êtes alors inviter à saisir les informations concernant votre compte. Chaque champ est obligatoire pour valider votre inscription. Les informations à saisir sont :

- Prénom
- Nom de famille
- Identifiant
- Mot de passe
- Confirmer le mot de passe (doit être identique à la saisie précédente)
- Adresse mail

Une fois toutes les informations saisies, vous devez valider votre inscription en appuyant sur le bouton menu de votre téléphone puis « Save My Account ».

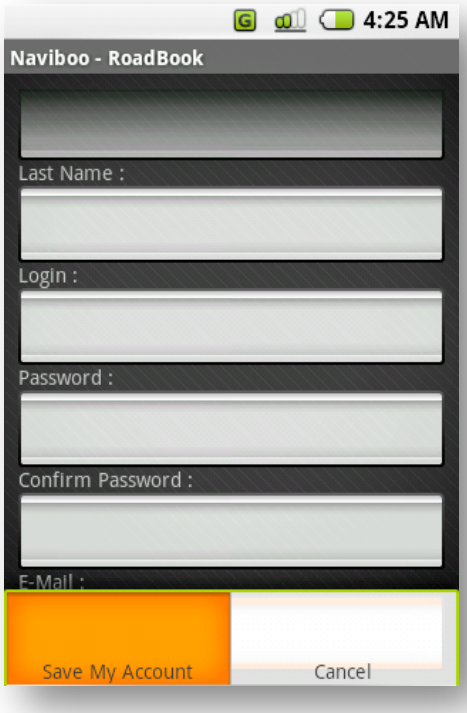

Figure 2 - Sauvegarder mon compte

Les données vont être vérifiées et si tout est correct le compte est créé et vous êtes redirigé vers l'écran d'accueil avec une fenêtre vous informant que votre compte a été correctement créé. Vous pouvez dès maintenant l'utiliser pour vous identifier.

A tout moment il vous est possible d'annuler la création du compte. Pour faire cela il vous suffit d'appuyer sur le bouton menu de votre téléphone puis « Cancel ». Vous êtes alors redirigé vers l'écran d'accueil sans que le compte soit créé.

3. S'identifier

Lorsque vous possédez un compte, vous pouvez vous identifier afin de profiter pleinement de l'application. Pour cela il suffit de saisir votre identifiant et le mot de passe correspondant, puis appuyez sur le bouton « Login ».

Une fois que l'identification est faite, vous êtes redirigé sur le menu principal de l'application d'où il vous est possible de profiter de toutes les fonctionnalités proposées par l'application.

Figure 3 - Menu principal

A tout moment il est possible de vous déconnecter de l'application, pour cela il est nécessaire d'utiliser le menu de l'appareil et d'appuyer sur le bouton « Log off » :

4. Consulter les informations relatives à l'application

Pour consulter les toutes les informations décrivant le cadre et la version de l'application, appuyez simplement sur le logo Naviboo qui est présent sur l'écran d'accueil et également sur l'écran du menu principal. Vous aurez alors accès à l'écran « à Propos » de l'application :

Figure 4 - "A propos"

5. Modifier son profil

Pour accéder à vos informations personnelles, appuyez sur le bouton « My Account » du menu principal.

Vous pouvez modifier toutes les informations de votre profil à l'exception de votre identifiant. Une fois les modifications faites, il vous suffit de les sauvegarder en appuyant sur le bouton « Save » du menu de l'appareil.

A tout moment il est possible d'annuler l'action en cours en utilisant le bouton « Cancel » du menu de l'appareil.

6. Créer et modifier son carnet de route

Lors de vos visites il vous sera possible de publier des étapes de votre carnet de route si celui-ci est correctement configuré. Pour avoir un carnet de route valide vous devez posséder un compte Blogger et y activer la fonction mail-to-blogger.

Votre adresse mail-to-blogger est indispensable pour avoir accès à la fonctionnalité du carnet de route de l'application.

La création et la modification du carnet de route se fait en appuyant sur le bouton « My RoadBook » du menu principal.

Une fois toutes les informations saisies, vous pouvez sauvegarder votre carnet de route en appuyant sur le bouton « Save » du menu de l'appareil :

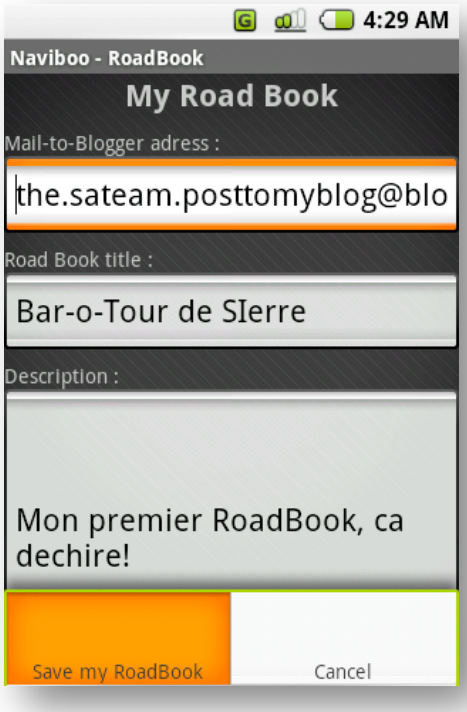

Figure 5 – Sauvegarder mon carnet de route

A tout moment il est possible d'annuler l'action en cours en utilisant le bouton « Cancel » du menu de l'appareil.

7. Créer un nouveau parcours

Pour créer un nouveau parcours il vous suffit d'appuyer sur le bouton « new Road » du menu principal.

Vous pouvez saisir le nom et la description du nouveau parcours et lui attribuer des points d'intérêt en appuyant sur le bouton « Add Point » du menu de l'appareil et vous serez alors redirigé vers un outil de recherche de points :

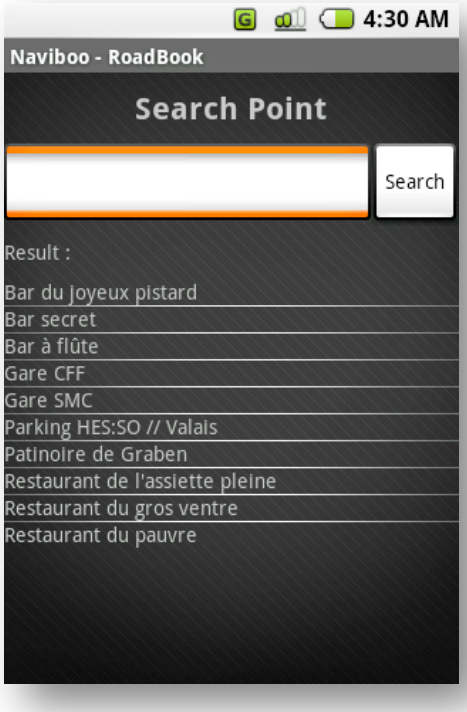

Figure 6 - Recherche de points d'intérêt

Une fois tous que tous les points souhaités sont ajoutés au parcours, vous pouvez le sauvegarder à l'aide du bouton « Save Road » du menu de l'appareil :

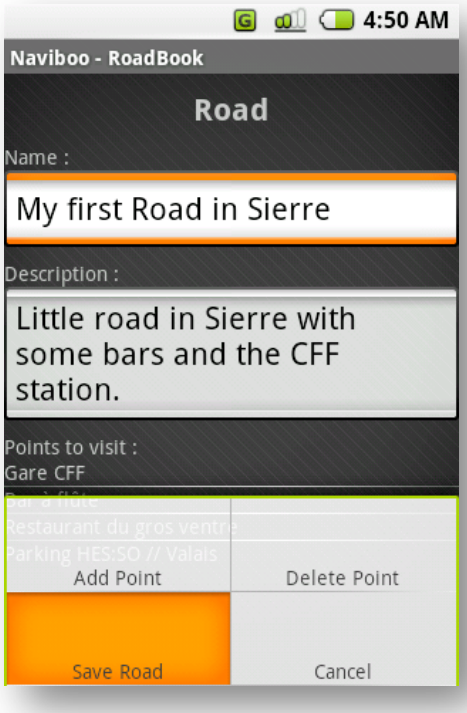

Figure 7 - Sauvegarder le parcours

A tout moment il est possible d'annuler l'action en cours en utilisant le bouton « Cancel » du menu de l'appareil.

8. Démarrer une visite

Pour démarrer une visite il faut avant tout sélectionner le parcours à charger, pour cela il vous faut appuyer sur le bouton « Select Road » du menu principal :

La liste de tous les parcours disponibles est affichée, il suffi de sélectionner celui que vous désirez à l'aide des touches directionnelles de l'appareil, puis d'appuyer sur le bouton « Start » du menu de l'appareil.

Le parcours est alors chargé et les différents points sont affichés sur la carte :

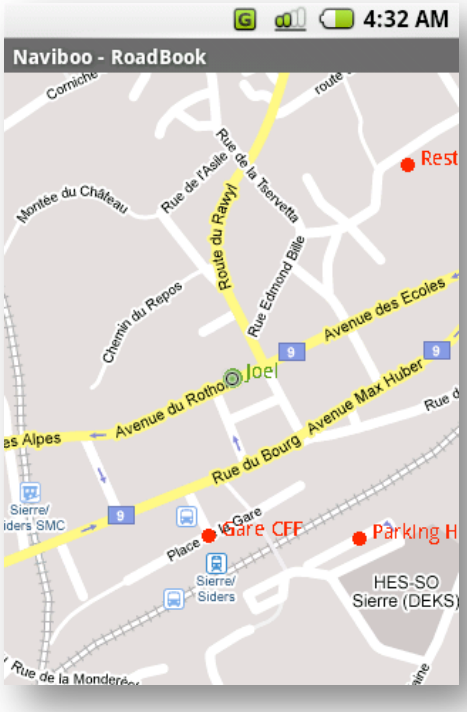

Figure 8 - Visite

A tout moment il est possible d'annuler l'action en cours en utilisant le bouton « Cancel » du menu de l'appareil.

9. Publier une étape du carnet d'adresse

A tout moment lors de votre visite, vous pouvez créer et publier une étape sur votre carnet de route. Pour cela il vous suffit d'appuyer sur le bouton « Add a RoadBook Step » sur le menu de l'appareil.

Vous devez sélectionner l'image qui illustrera votre étape puis vous pouvez appuyer sur le bouton « Take Picture» qui aura pour effet de valider votre sélection :

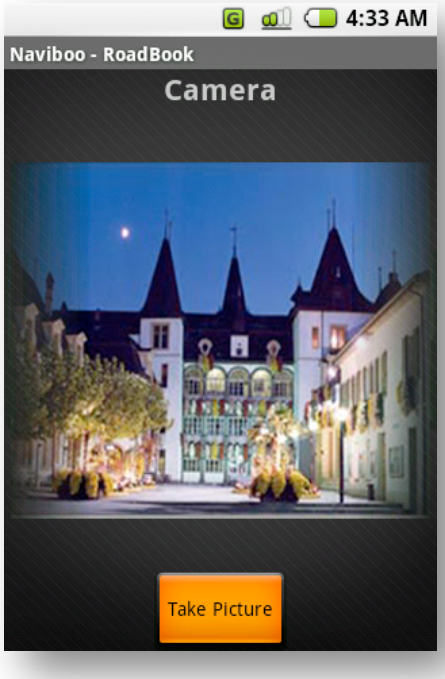

Figure 9 - Choix de l'image à publier

Finallement il vous suffit de saisir les différentes informations de votre étape, puis appuyer sur le bouton « Publish on blog» du menu de l'appareil pour que l'étape soit publiée sur votre Blog :

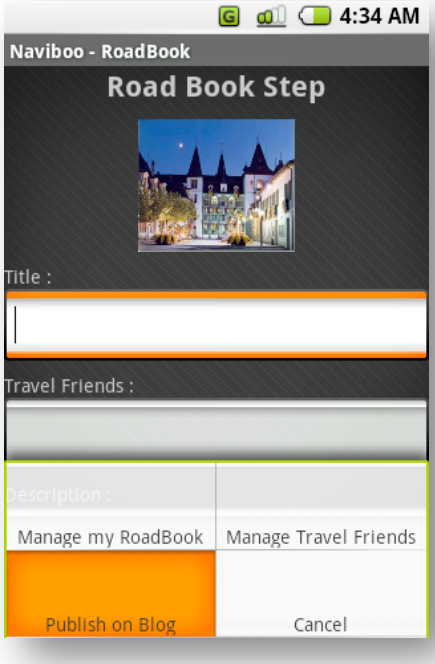

Figure 10 - Publier l'étape sur mon Blog

A tout moment il est possible d'annuler l'action en cours en appuyant sur le bouton « Cancel » du menu de l'appareil.

10. Utiliser l'outil de voyage

Vous pouvez accéder à l'outil de voyage depuis le menu principal ou en cours de visite, cet outil vous permettra de convertir un montant d'une devise monétaire à une autre. Il y a également un outil de conversion d'unités de mesure.

Lorsque vous n'avez plus besoin de l'outil de voyage, il suffit d'appuyer sur le bouton « OK » du menu de l'appareil.

11. Table des illustrations

Bachelor | 08

Manuel Technique – « Naviboo – RoadBook »

Annexe 2

Google Android

Joël Salamin

Hes·so VALAIS WALLIS

Table des matières

1. Interface graphique

Ce point détaille la totalité de l'interface de l'application avec un descriptif de chaque écran ainsi que la structure de chacun.

LoginActivity

Cet écran est le point d'entrée de l'application. LoginActivity est lié au fichier layout : « main.xml ».

Figure 1 - Interface graphique : LoginActivity

L'écran est composé de peu d'éléments visuels, l'utilisateur est invité à saisir son identifiant et mot de passe afin de profiter de l'application. Si l'utilisateur ne possède pas encore de compte pour s'identifier, il lui suffit de créer un nouveau compte. En cliquant sur le logo Naviboo il accèdera à un écran d'information au sujet du projet.

On remarque clairement que les éléments sont de grande taille dans le but de rendre l'utilisation au doigt possible. L'écran se limite à l'essentiel et le menu est utilisé pour apporter à l'utilisateur la possibilité de fermer l'application.

Les éléments affichés sont les suivantes :

- Logo Naviboo
- Champ pour la saisie de l'identifiant
- Champ pour la saisie du mot de passe
- Bouton d'identification
- Bouton de création d'un nouveau compte

Le menu permet d'ajouter la possibilité de fermer l'application sans avoir un bouton supplémentaire sur l'écran de l'appareil.

Voici la structure de l'écran final :

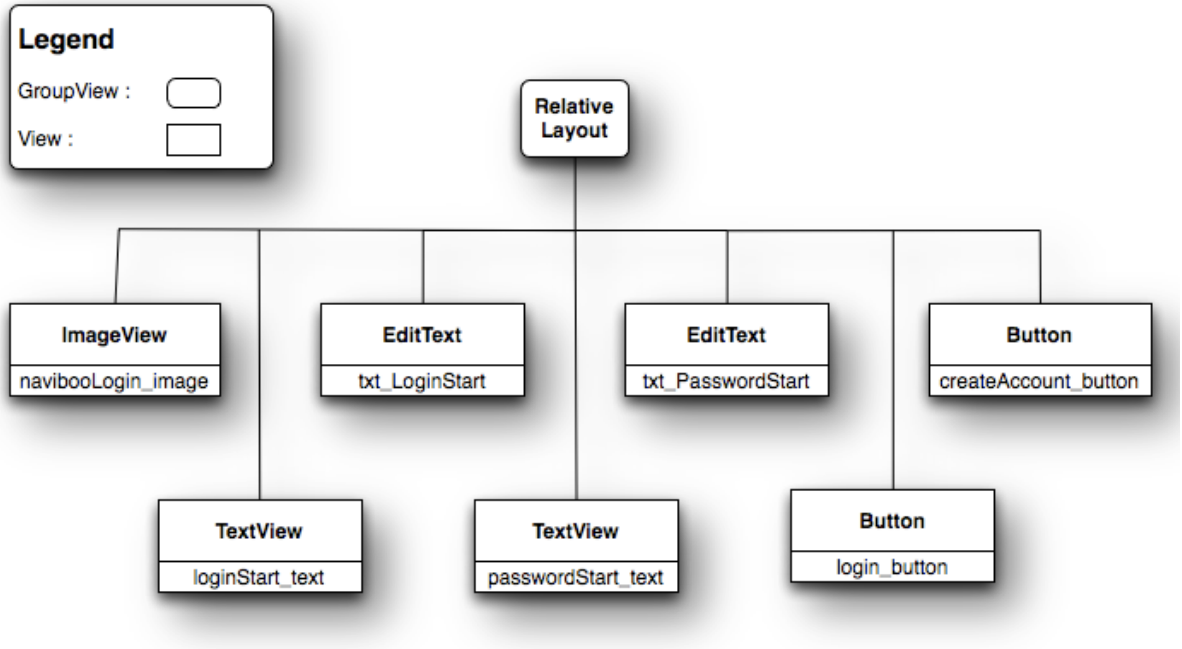

Figure 2 - Structure d'écran : LoginActivity

La structure de l'écran est basique, un conteneur regroupe tous les éléments. L'avantage apporté par un conteneur relatif par rapport à un conteneur linéaire se situe principalement sur la simplicité de répartir les éléments pour occuper au mieux la taille de l'écran. Un conteneur linéaire est bien plus rigide et n'est à mon avis pas adapté pour un écran composé d'éléments de tailles différentes et centrés. Pour de plus amples informations concernant les caractéristiques de chaque éléments, se référer au document annexe : « Manuel du développeur ».

Le schéma regroupe les informations principales sur la structure de l'écran Login :

- La catégorie de l'élément (GroupView / View)
- Le type de l'élément (RelativeLayout, EditText, Button, …)
- L'identifiant de l'élément respectant la convention de nommage

AboutActivity

Cet écran est accessible depuis LoginActivity et depuis le menu principal de l'application. AboutActivity est lié au fichier layout : « about.xml ».

Figure 3 - Interface graphique : AboutActivity

L'utilisateur peut consulter les informations relatives au projet Naviboo – RoadBook.

Les éléments affichés sont :

- Objectif fixé du travail de Bachelor
- Etudiant effectuant ce projet
- Professeur responsable
- Version de l'application
- Mention et logo de l'HES:SO // Valais
- Mention et logo de l'environnement de développement Android

L'utilisation du menu permet de présenter un écran dépourvu de boutons et contenant uniquement les informations souhaitées par l'utilisateur. Le menu permet à l'utilisateur de revenir à l'écran précédent.

Voici la structure de l'écran final :

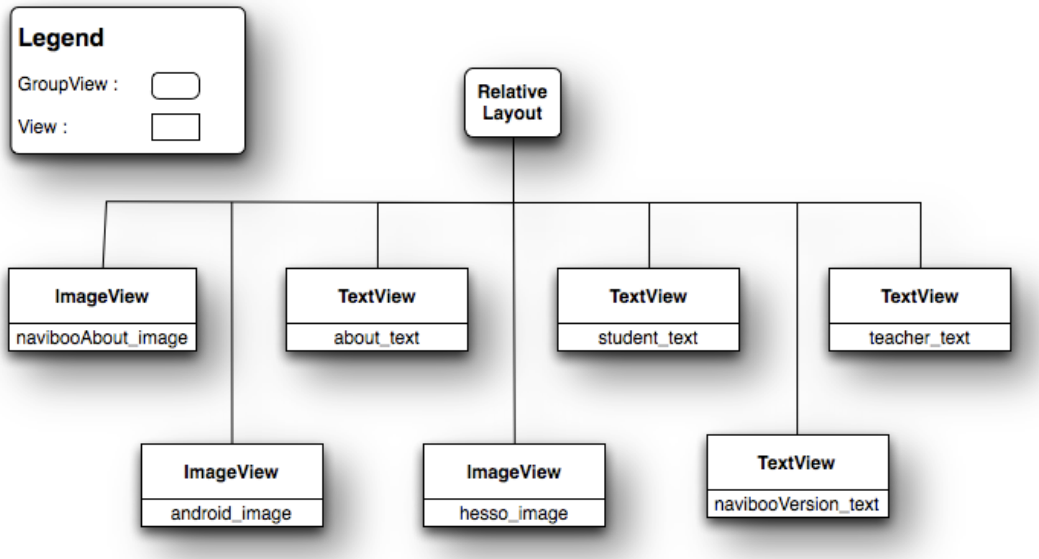

Figure 4 - Structure d'écran : AboutActivity

La structure de l'écran est relativement simple malgré le fait qu'elle soit composée d'un grand nombre d'éléments. L'utilisation d'un conteneur relatif représente une nouvelle fois la solution la mieux adaptée étant donné la répartition des éléments à l'écran.

EditAccountActivity

Cet écran remplit deux fonctions, soit il est utilisé lors de la création d'un nouveau compte, soit lors de la modification d'un compte déjà créé. EditAccountActivity est lié au fichier layout : « editaccount.xml ».

Figure 5 - Interface graphique : EditAccountActivity

L'utilisateur n'étant pas encore identifié est invité à saisir toutes les informations nécessaires à la création d'un nouveau compte. L'utilisateur étant identifié est dirigé vers cet écran lorsqu'il souhaite modifier les informations de son compte. Dans le cas d'une modification, toutes les informations seront modifiables à l'exception de l'identifiant.

Les éléments affichés sont :

- Prénom
- Nom de famille
- Identifiant
- Mot de passe
- Confirmation du mot de passe
- Adresse e-Mail

L'utilisation du menu permet de se passer de bouton sur l'écran. Etant donné la quantité d'élément à afficher, Android offre la possibilité d'utiliser un système d'ascenseur pour pouvoir naviguer sur l'écran et ainsi avoir accès à tous les champs de saisie.

Voici la structure de l'écran final :

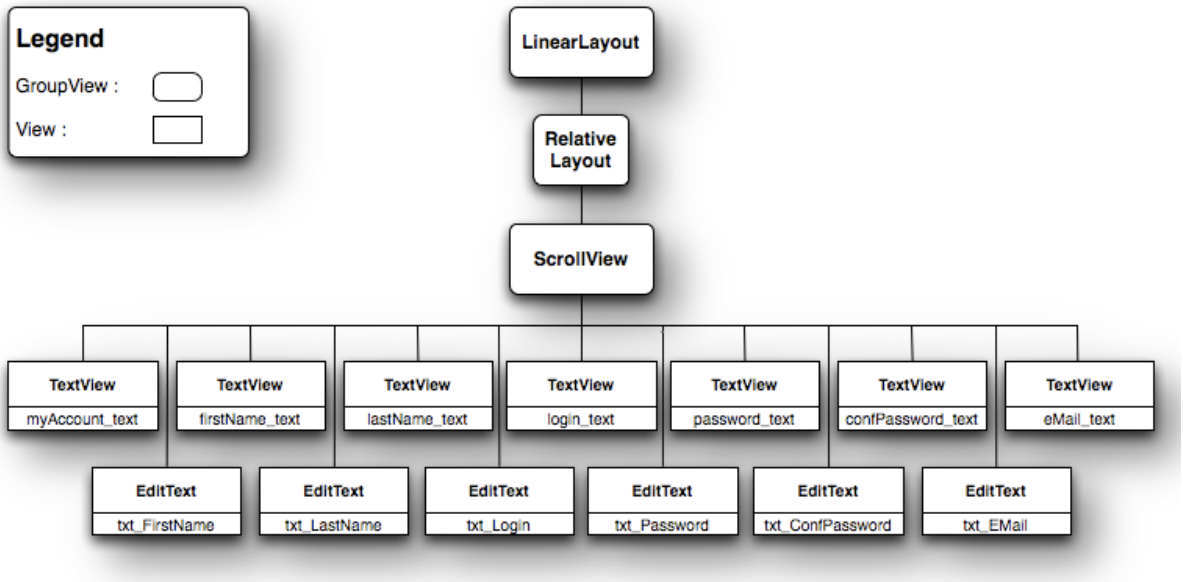

Figure 6 - Structure d'écran : EditAccountActivity

La structure est plus complexe étant donné les nombreux éléments nécessaires à la création/modification du compte. Afin de pouvoir naviguer simplement entre les différents élément, un conteneur particulier est utilisé pour englober tous les éléments : « ScrollView ». Ce conteneur permet de composer un écran avec plus d'éléments que ce qui peut être affiché sur l'appareil, et un système de défilement permet de parcourir la totalité de l'écran.

Pour le conteneur englobant tout les éléments, le choix le plus logique est d'utiliser un « LinearLayout » étant donné que tous les éléments sont de même taille et occupent toute la surface de l'écran.

MenuActivity

Cet écran représente le noyau de l'application, c'est à partir de là que l'utilisateur profitera des différentes fonctionnalités mises à sa disposition. MenuActivity est lié au fichier layout : « menu.xml ».

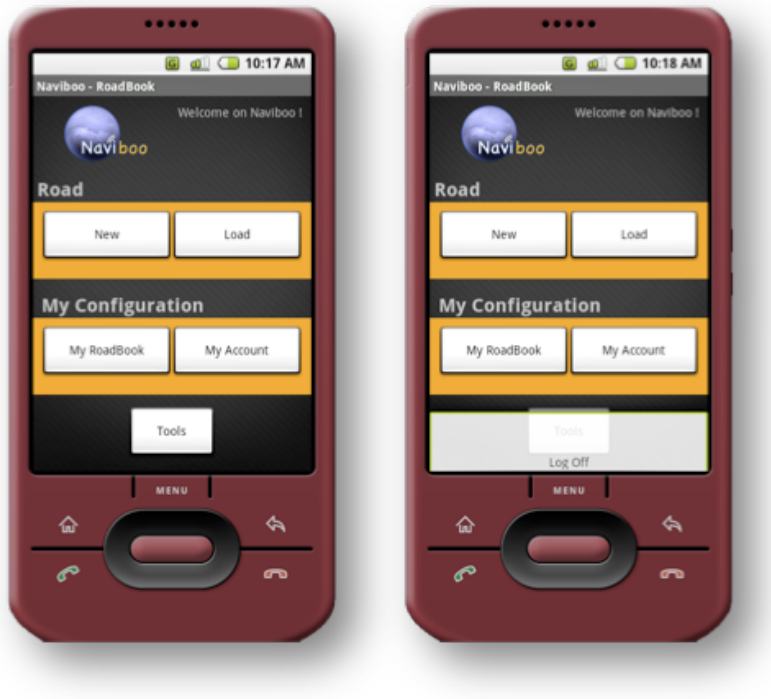

Figure 7 - Interface graphique : MenuActivity

L'écran de menu est structuré de manière à être le plus clair possible. Il y a deux parties mises en évidence et bien définies à l'aide d'un encadrement jaune. La première partie regroupe toutes les actions relatives aux parcours et la deuxième partie se concentre sur tout ce qui se rapporte à l'utilisateur. Le dernier élément est en marge de ces deux zones car il apporte un outil indépendant de l'utilisateur et des parcours.

La force des éléments « touch » se trouve accentuée par un niveau design très agréable à l'œil tout en apportant de la simplicité dans l'utilisation.

Les éléments affichés sont :

- Logo Naviboo
- Bouton de création d'un nouveau parcours
- Bouton de chargement d'un parcours existant
- Bouton de création/modification du carnet de route
- Bouton de modification du compte utilisateur
- Bouton pour l'utilisation de l'outil de voyage

L'utilisation du menu permet une nouvelle fois d'éviter de surcharger l'affichage et offre à l'utilisateur la possibilité de se déconnecter de l'application.

Voici la structure de l'écran final :

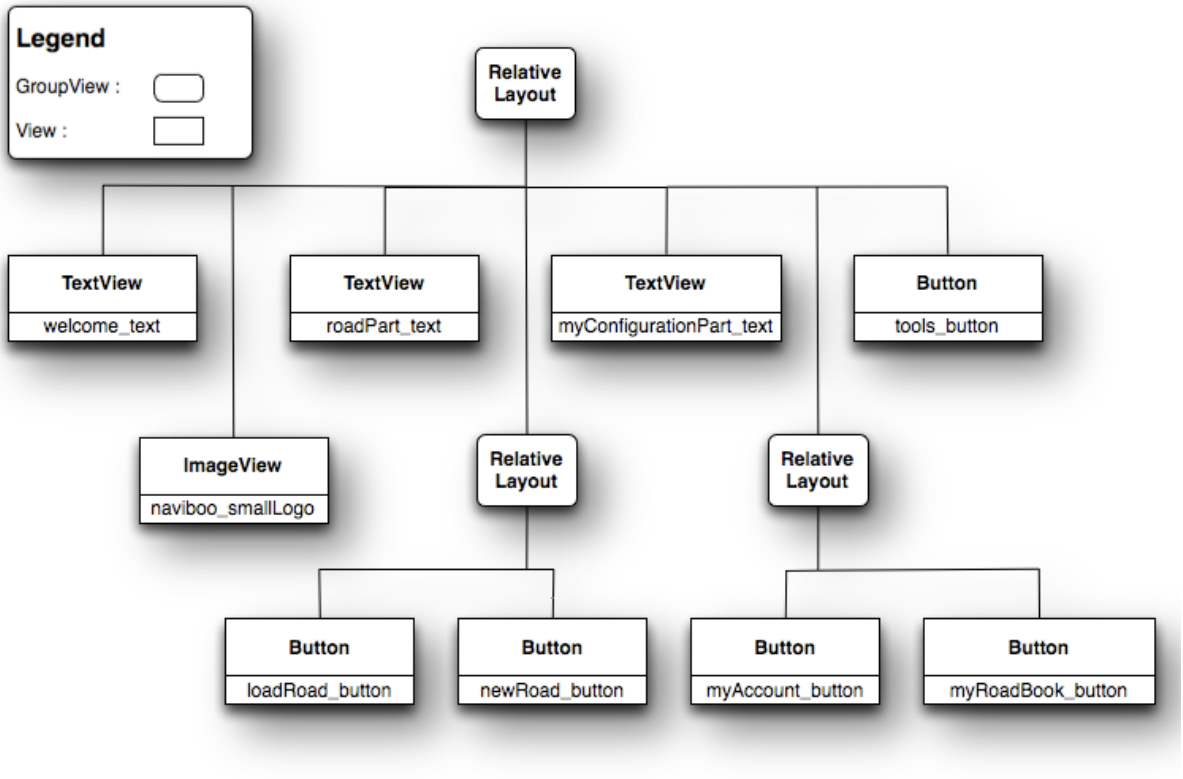

Figure 8 - Structure d'écran : MenuActivity

La structure est plus complexe que celle des écrans précédents car elle possède plusieurs niveaux de conteneurs. Le conteneur de second niveau se gère de la même manière que n'importe quel élément. Le choix d'utiliser des conteneurs à un deuxième niveau représente la meilleure des solutions pour réaliser un affichage agréable et surtout pour avoir un affichage qui restera cohérent si les dimensions de l'appareil changent.

Au niveau de la création du layout, il n'y pas de difficultés particulières en comparaison avec un écran plus simple. Du moment que la philosophie est assimilée et que les éléments sont connus, il devient relativement aisé de construire des écrans évolués et complexes.

L'utilisation de conteneurs relatifs se justifie par le fait que les éléments contenus ne sont de tailles variables et leur répartition sur l'écran ne permet pas d'utiliser un conteneur linéaire sans rencontrer de problèmes.

SelectRoadActivity

Cet écran est le point de départ du chargement d'un parcours afin de commencer une visite. SelectRoadActivity est lié au fichier layout : « selectroad.xml ».

Figure 9 - Interface graphique : SelectRoadActivity

L'écran de sélection de parcours est accessible depuis le menu principal et permet à l'utilisateur de gérer ses parcours. Les parcours sauvegardés peuvent être modifiés et il est l'ajout de parcours est également possible.

Depuis cet écran l'utilisateur peut sélectionner le parcours à charger afin de commencer sa visite.

Les éléments affichés sont :

- Parcours créés par l'utilisateur

L'utilisation du menu permet d'avoir une interface sobre et entièrement consacrée à la liste des parcours sauvegardés. Le menu permet de manipuler le parcours sélectionné, ajouter un nouveau parcours, retourner au menu ou démarrer le parcours souhaité.

La liste de parcours utilise un système d'ascenseur si celle-ci est trop longue pour être affichée complètement sur l'écran.

Voici la structure de l'écran final :

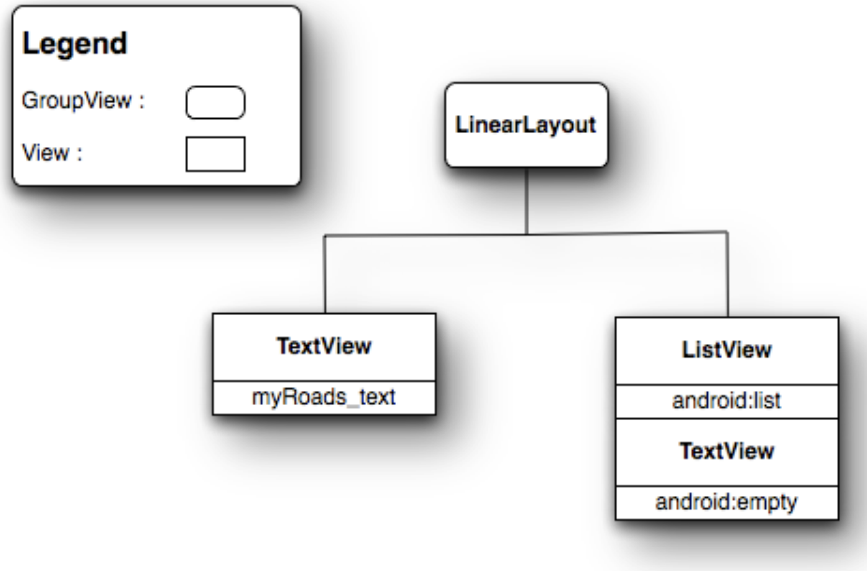

Figure 10 - Structure d'écran : SelectRoadActivity

La structure est très simple étant donné qu'elle n'est composée que de deux éléments englobés par un conteneur linéaire.

Il est important de mettre en évidence l'élément ListView qui a un comportement particulier au niveau de son utilisation. SelectRoadActivity est une ListActivity et non une Activity comme la grande majorité des écrans de l'application.

Une ListActivity doit posséder l'élément ListView auquel est attribué l'identifiant unique « android :list » et l'élément de remplacement en cas de liste vide est identifié par « android :empty ». Pour avoir le détail complet du comportement et de l'utilisation de cet élément, se référer au document annexe « Manuel du développeur ».

EditRoadActivity

Cet écran permet la création d'un nouveau parcours et permet également de modifier un parcours existant. EditRoadActivity est lié au fichier layout : « editroad.xml ».

Figure 11 - Interface graphique : EditRoadActivity

L'utilisateur accède à cet écran soit directement depuis le menu lors de la création d'un nouveau parcours, soit depuis l'écran de sélection de parcours lors de la modification ou l'ajout d'un parcours.

Cette interface est relativement simple et se résume au stricte minimum une nouvelle fois pour rendre l'application simple d'utilisation. Le champ de description s'adaptera à la taille du texte afin de permettre à l'utilisateur de saisir une description aussi grande que souhaitée.

Les éléments affichés sont :

- Nom du parcours
- Description du parcours
- Points composant le parcours

L'utilisation du menu permet d'avoir un écran dépouillé de boutons afin de se focaliser sur le détail du parcours. Le menu permet d'ajouter/supprimer un point au parcours, sauvegarder le parcours ou annuler l'action en cours afin de retourner à l'écran précédent.

Voici la structure de l'écran final :

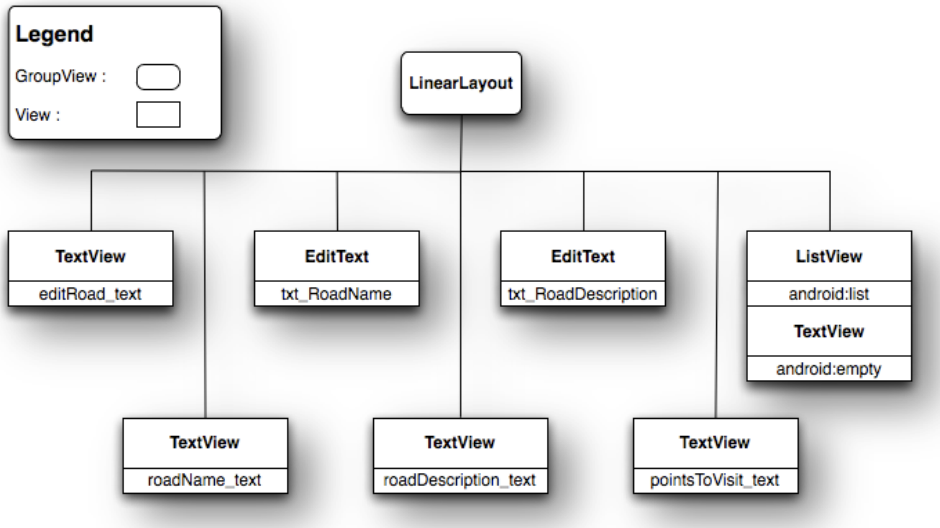

Figure 12 - Structure d'écran : EditRoadActivity

La structure est composée de nombreux éléments regroupés dans un conteneur linéaire. L'utilisation de ce type de conteneur est la solution la mieux adaptée étant donné que chaque élément occupe toute la largeur de l'écran.

EditRoadActivity est de type ListActivity et possède donc l'élément particulier regroupant un ListView et un élément de remplacement en cas de liste vide.

PointDetailActivity

Cet écran permet l'affichage du détail d'un point d'intérêt. PointDetailActivity est lié au fichier layout : « pointdetail .xml».

Figure 13 - Interface graphique : PointDetailActivity

L'utilisateur accède à cet écran depuis les différents écrans affichant une liste de points. Il suffit à l'utilisateur de cliquer sur le point dont il souhaite les détails.

L'interface se résume à deux champs permettant de consulter les différentes informations relatives au point concerné. L'utilisateur n'a pas la possibilité de modifier ces informations.

Les éléments affichés sont :

- Nom du point d'intérêt
- Description du point d'intérêt

L'utilisation du menu permet simplement d'éviter la présence d'un bouton qui ne ferait que restreindre l'espace d'affichage.

Voici la structure de l'écran final :

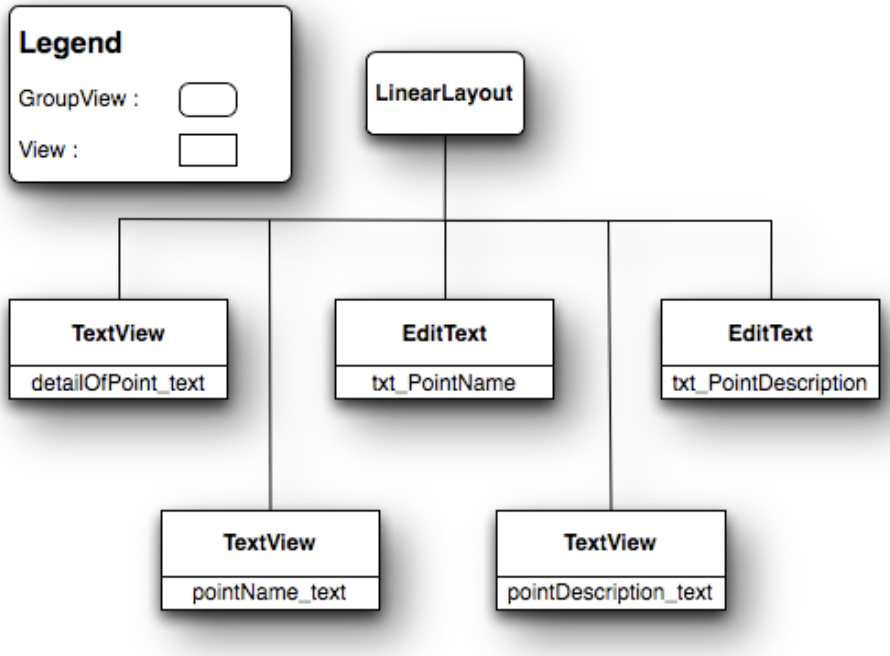

Figure 14 - Structure d'écran : PointDetailActivity

La structure est simple avec cinq éléments regroupés dans un conteneur linéaire. L'utilisation de ce type de conteneur est la mieux adaptée à l'affichage souhaité. Les éléments occupent toute la largeur de l'écran et s'adaptent à la hauteur à disposition.

SearchPointActivity

Cet écran est affiché lors de l'ajout d'un point à un parcours. SearchPointActivity est lié au fichier layout : « searchpoint.xml ».

Figure 15 - Interface graphique : SearchPointActivity

Pour ajouter un point, l'utilisateur est inviter à faire une recherche selon une chaine de caractères à saisir ou simplement en consultant la liste complète des points disponibles.

L'interface se résume à un champ de saisie et un bouton afin de faire une recherche parmi la liste des points disponible et le reste de l'espace disponible est occupé par la liste des points correspondant à la recherche.

Les éléments affichés sont :

- Champ de saisie pour la recherche
- Bouton de recherche
- Liste des points d'intérêt correspondant à la recherche

L'utilisation du menu rend l'écran agréable à utiliser car il n'est pas trop chargé en éléments affichés. Le menu sert à ajouter le point sélectionné ou simplement annuler l'action et revenir à l'écran précédent.

Voici la structure de l'écran final :

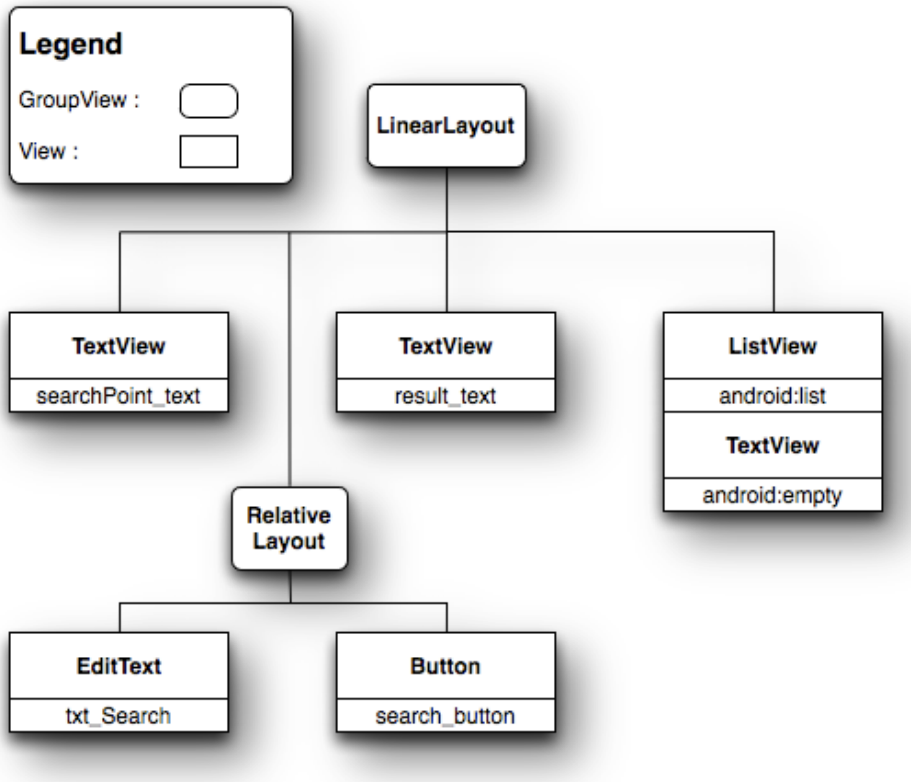

Figure 16 - Structure d'écran : SearchPointActivity

La structure est relativement complexe car elle combine des éléments simples avec des éléments complexes étant donné qu'il y a un conteneur et une ListView.

Tous les éléments sont regroupés dans un conteneur linéaire qui apporte de nombreux avantages lorsque l'interface est composée d'une suite d'éléments occupant toute la largeur de l'écran. Un conteneur relatif est utilisé pour avoir une zone de recherche sur une même ligne et ainsi ne pas occuper de l'espace inutilement et pouvoir dédier l'espace restant à l'affichage de la liste.

TripActivity

Cet écran est le guide touristique à proprement parlé, il faut avoir sélectionner un parcours avant de pouvoir commencer la visite. TripActivity est lié au fichier layout : « trip.xml ».

Figure 17 - Interface graphique : TripActivity

L'interface graphique est réduite au minimum pour offrir à l'utilisateur une carte exploitant la totalité de l'espace à disposition. La carte affiche les différents points qui composent le parcours chargé et la position GPS de l'utilisateur est signalée par un point animé en temps réel.

L'utilisateur a la possibilité de changer le mode d'affichage et d'utiliser le niveau de zoom qu'il souhaite. Pour avoir des informations complètes sur l'utilisation de l'application, se référer au document annexe « Manuel de l'utilisateur ».

Les éléments affichés sont :

- Carte géographique de la région
- Position de l'utilisateur
- Position des points d'intérêt composant le parcours chargé

L'utilisation du menu apporte un avantage considérable et permet surtout de jouir d'un affichage complètement consacré à la carte. Depuis le menu, l'utilisateur a la possibilité de zoomer/dézommer, utiliser l'outil de voyage, ajouter une étape à son carnet de route et enfin retourner au menu principal.

Voici la structure de l'écran final :

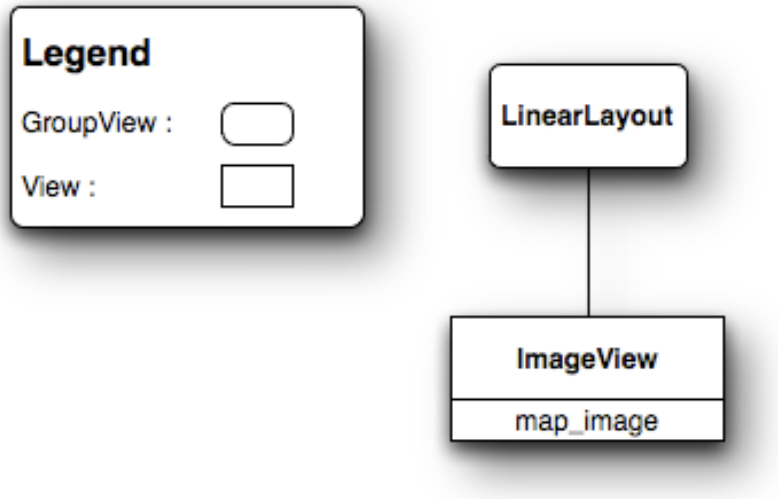

Figure 18 - Structure d'écran : TripActivity

La structure est réduite au strict minimum, elle est composée d'un conteneur et d'un seul élément permettant d'afficher la carte géographique. L'objectif étant d'avoir une application la plus simple et intuitive possible, le choix de n'afficher que la carte me paraît être le meilleur.

MyRoadBookActivity

Cet écran est accessible depuis le menu principal ou directement depuis l'écran de création d'une étape pour le carnet de route. MyRoadBookActivity est lié au fichier layout : « myroadbook.xml ».

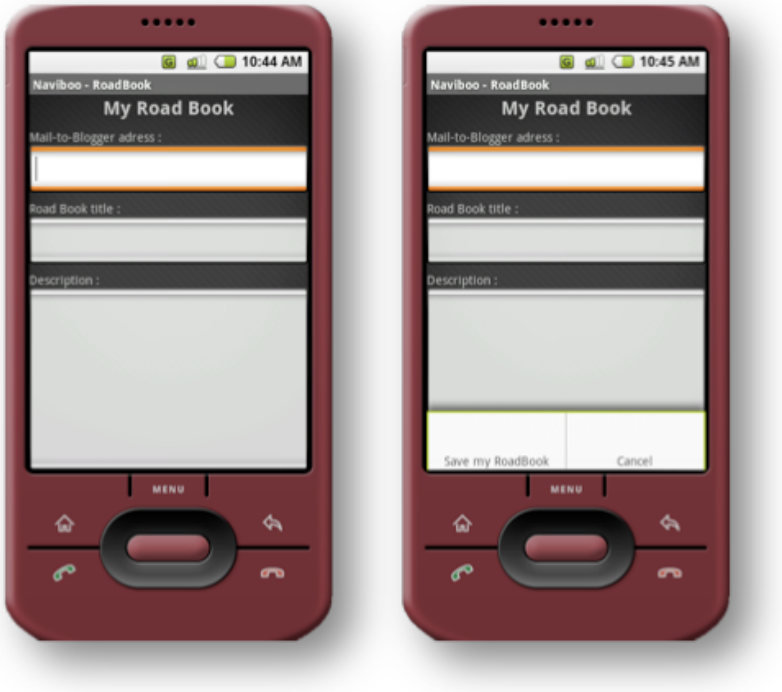

Figure 19 - Interface graphique : MyRoadBookActivity

L'utilisateur peut créer et modifier son carnet de route à tout moment, soit en passant par le menu de l'application, soit directement au moment de la publication d'une nouvelle étape.

Tout l'espace est utilisé pour les informations relatives au carnet de route et une nouvelle fois un écran dépourvu de bouton visible.

Les éléments affichés sont :

- Adresse Mail-to-Blogger
- Titre du carnet de route
- Description du carnet de route

L'utilisation du menu permet à l'utilisateur de sauvegarder les modifications effectuées ou simplement annuler l'action et retourner à l'écran précédent.

Il est important de préciser que l'utilisateur doit posséder un compte Mail-to-Blogger pour profiter de la partie carnet de route de l'application. Pour en savoir plus au sujet de l'utilisation et la configuration du carnet de route, se référer au document annexe : « Manuel de l'utilisateur ».

Voici la structure de l'écran final :

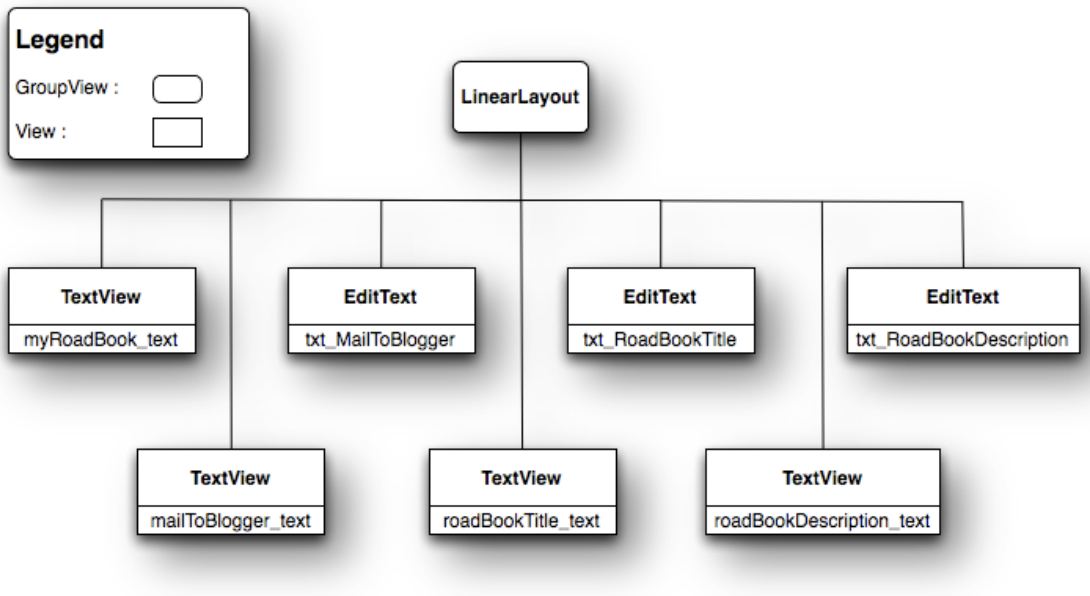

Figure 20 - Structure d'écran : MyRoadBookActivity

La structure est très simple comme l'écran de modification de compte. Un conteneur linéaire qui regroupe tous les éléments permettant de paramétrer le carnet de route.

CameraActivity

Cet écran représente la première étape lors de l'ajout d'une étape au carnet de route. CameraActivity est lié au fichier layout : « camera.xml ».

Figure 21 - Interface graphique : CameraActivity

L'utilisateur est invité à prendre une photo, ce qui représente la première étape lors de la création et la publication d'une étape du carnet de route.

L'écran est simple et le choix de mettre un bouton pour la prise de photo permet d'éviter à l'utilisateur d'avoir à manipuler le menu alors qu'il a trouvé la bonne prise à photographier. Il lui suffit de pointer la caméra vers le lieu à photographier et appuyer sur le bouton pour prendre la photo et il est automatiquement amené à l'écran suivant pour saisir les informations relatives à l'étape à publier.

Les éléments affichés sont :

- Choix de photos
- Bouton de capture d'image

L'utilisation du menu permet de concentrer l'interface uniquement à la prise de la photo. Le menu est utilisé pour annuler l'action et revenir à l'écran précédent.

Voici la structure de l'écran final :

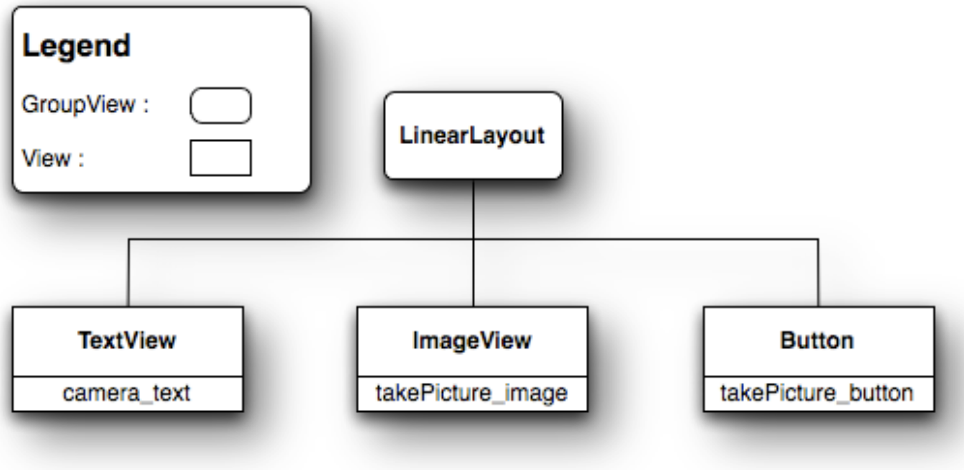

Figure 22 - Structure d'écran : CameraActivity

La structure est très simple car la prise de photo se veut la plus proche de l'utilisation d'un appareil photo avec une partie indiquant la prise de vue et un bouton permettant de valider la photo.

RoadBookStepActivity

Cet écran représente la deuxième et dernière étape lors de la publication d'une étape du carnet de route. RoadBookStepActivity est lié au fichier layout : « roadbookstep.xml ».

Figure 23 - Interface graphique : RoadBookStepActivity

L'utilisateur est dirigé vers cet écran une fois qu'il a pris la photo souhaitée pour l'étape de son carnet de route. La photo prise est affichée afin de fournir à l'utilisateur une vue complète sur l'étape avant sa publication.

Les éléments affichés se limitent aux champs relatifs à l'étape, de nombreuses informations additionnelles pourront être publiées sans que l'utilisateur ne les saisisse. Par exemple les coordonnées géographiques, le titre du carnet de route, le nombre de kilomètres parcourus depuis l'étape précédente, etc…

Les éléments affichés sont :

- Titre de l'étape du carnet de route
- Liste des compagnons de voyage (fonctionnalité non implémentée)
- Description de l'étape

L'utilisation du menu permet de regrouper toutes les actions au même endroit sans encombrer l'espace à disposition. Depuis le menu, l'utilisateur peut gérer son carnet de route, gérer ses amis de voyage (fonctionnalité qui n'est pas encore disponible), publier son étape sur internet ou bien annuler l'action afin de prendre une photo différente ou retourner au menu.

Voici la structure de l'écran final :

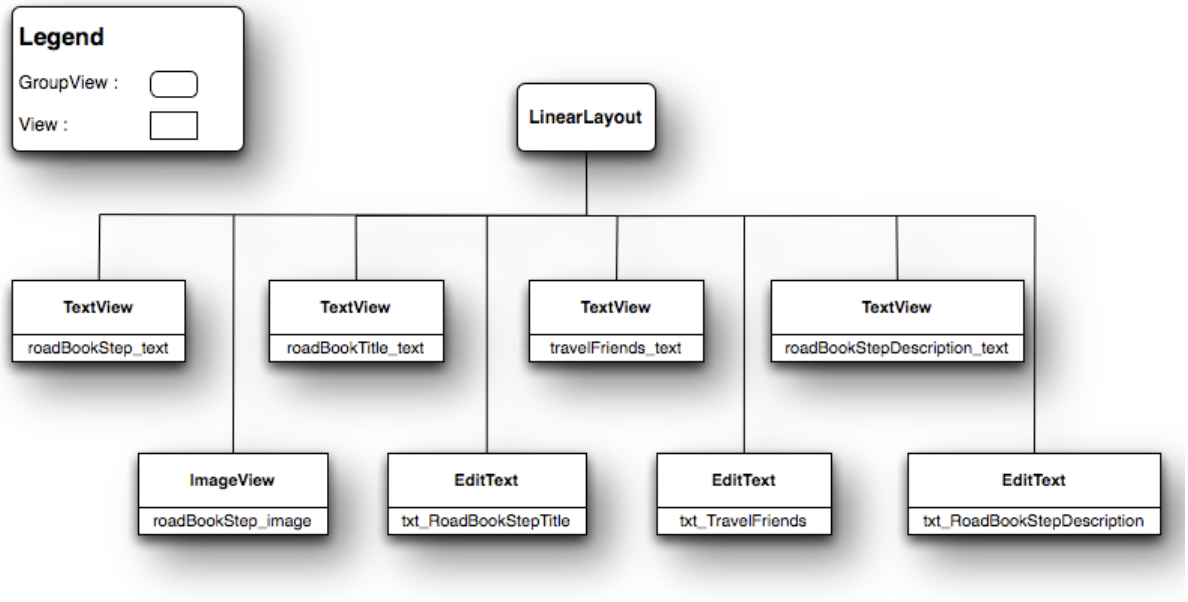

Figure 24 - Structure d'écran : RoadBookStepActivity

La structure est simple avec une succession d'élément affichés les uns au dessus des autres, c'est pourquoi le conteneur le mieux adapté est le linéaire.

TripToolsActivity

Cet écran est accessible depuis le menu ou depuis l'écran affichant la carte lors de la visite. TripToolsActivity est lié au fichier layout : « triptools.xml ».

Figure 25 - Interface graphique : TripToolsActivity

L'utilisateur peut convertir de la monnaie ainsi que des distances à tout moment grâce à l'outil de voyage mis à disposition par l'application Naviboo – RoadBook. Il lui suffit de saisir la monnaie ou la distance à convertir et de saisir l'unité souhaitée. La conversion se fait par l'utilisation de Web Service afin d'avoir des taux cohérents. L'outil se veut simple à utiliser et intuitif, c'est pourquoi les deux zones de conversions sont mises en évidence par des cadres de couleur noire.

Les éléments affichés sont :

- Choix des monnaies pour la conversion
- Montant à convertir
- Bouton de conversion de monnaies
- Résultat de la conversion
- Choix des distances pour la conversion
- Distance à convertir
- Bouton de conversion de distances
- Résultat de la conversion

L'utilisation du menu permet de soulager un peu l'interface qui est très chargée en éléments. Avec le menu, l'utilisateur peur quitter l'outil de voyage et retourner à l'écran précédent.

Voici la structure de l'écran final :

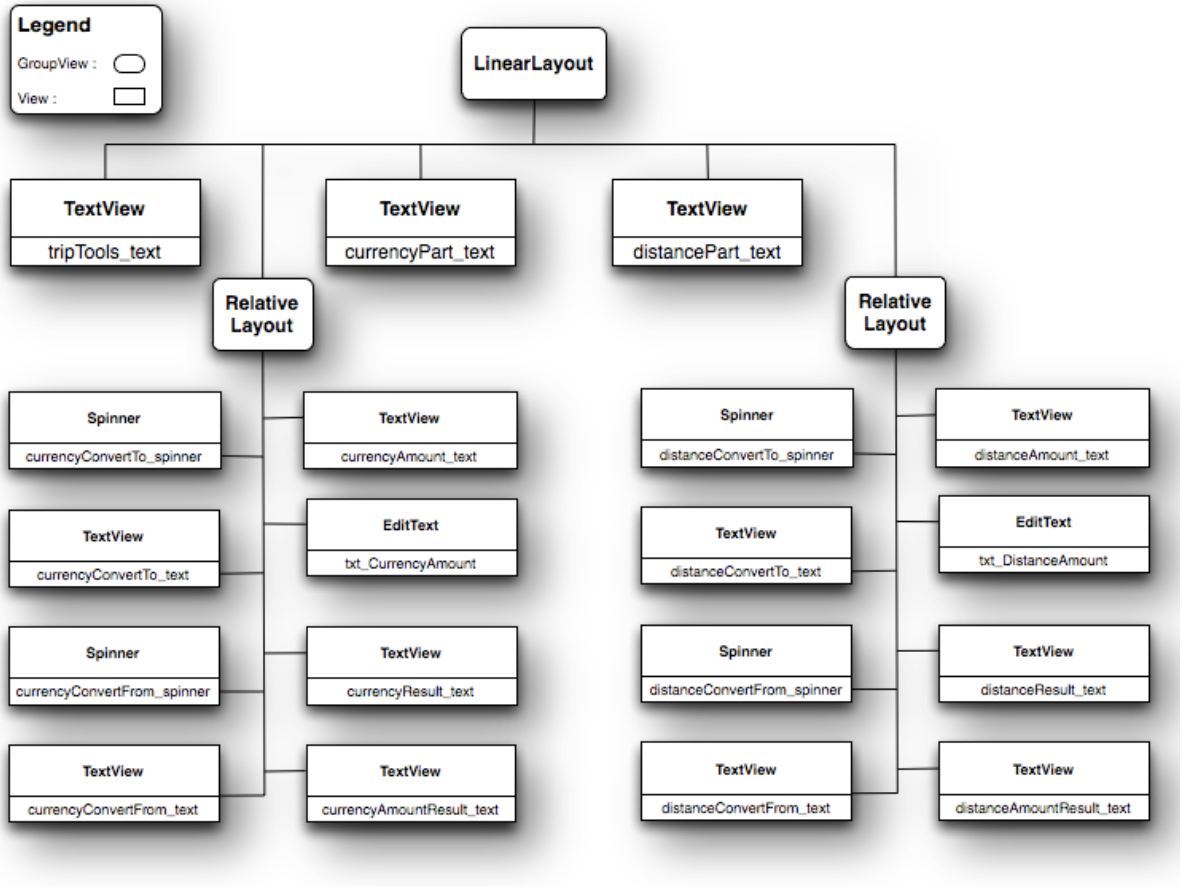

Figure 26 - Structure d'écran : TripToolsActivity

La structure est complexe car de nombreux éléments la composent. Le conteneur de base est linéaire étant donné que les éléments sont l'un à la suite de l'autre et occupent toute la largeur de l'écran.

Les deux zones de conversion sont composées d'un conteneur relatif, qui est le mieux adapté à la situation, afin de regrouper les différents éléments nécessaires à l'utilisateur.

2. Base de données – SQLite

Ce point détaille toute la couche base de données de l'application avec une description de la structure de base ainsi que le détail de chacune des tables.

Description

La base de données utilisée dans le cadre du projet est très simple, elle est composée de 5 tables.

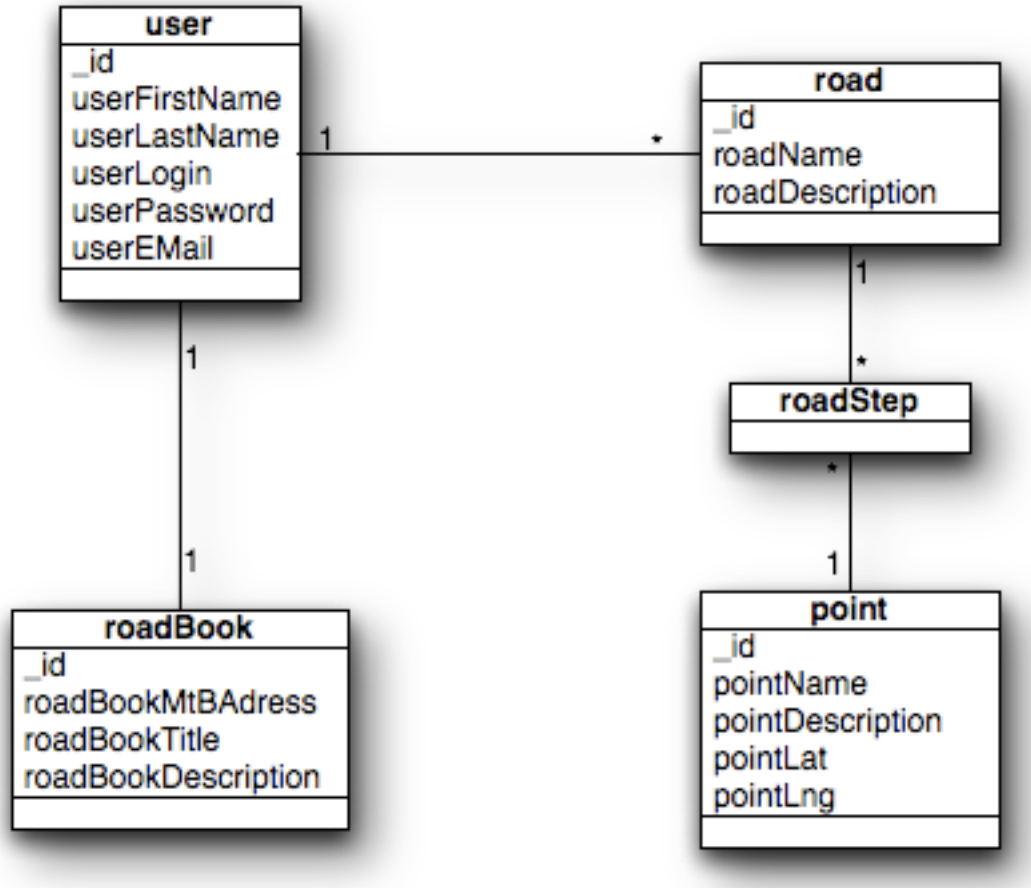

Figure 27 - Diagramme de base de données

Table user

Cette table permet de stocker toutes les informations liées au compte de l'utilisateur, chaque champ est obligatoire pour la création d'un nouvel utilisateur, excepté le champ « _id » qui est généré automatiquement lorsque le compte est créé. Le champ du prénom sera utilisé lors de l'affichage de la position de l'utilisateur sur la carte géographique et les champs identifiant/mot de passe sont utilisés lors de l'identification de l'utilisateur.

Un utilisateur possède un seul carnet de route et un nombre illimité de parcours.

Table roadBook

Cette table permet de stocker toutes les informations liées au carnet de route que possède un utilisateur. Chacun des champs de la table est obligatoires, à l'exception du champ « _id », car ils sont utilisés lors de la publication d'une nouvelle étape.

Un carnet de route n'appartient qu'à une seule personne, malgré la relation 1-1 entre l'utilisateur et le carnet de route, le choix de séparer ces deux tables a été pris dans le but de rendre la base de données plus simple à comprendre et surtout pour rendre l'application évolutive avec la possibilité d'avoir plus tard plusieurs carnets de route attribués au même utilisateur.

Table road

Cette table permet de stocker toutes les informations liées à un parcours, tous les champs sont obligatoires à l'exception du champ « _id » qui est généré automatiquement lors de la création du parcours. Les informations concernant le parcours sont utilisées lors de la publication d'une étape du carnet de route.

Un parcours est composé d'un ou plusieurs points d'intérêts, l'utilisateur est libre de créer le parcours de sont choix selon les points qui répondent à ses centres d'intérêts.

Table roadStep

Cette table est une table intermédiaire faisant le lien entre la table road et la table point. Elle permet d'attribuer des points à un parcours et un point doit pouvoir faire parti de plusieurs parcours.

Table point

Cette table permet de stocker toutes les informations liées à un point d'intérêts. Cette table est uniquement utilisée en lecture par l'application, aucune modification n'est effectuée lors de l'utilisation de Naviboo – RoadBook.

Un nouveau point doit être ajouté à l'aide de l'outil « sqlite3 » fournit par le SDK Android.

3. Diagramme de Class

Ce point passe en revue la totalité des Class qui composent le projet Naviboo – RoadBook avec le détail complet des différents choix pris lors du développement. Ce point permet de faire une synthèse de tous les éléments traités dans ce chapitre.

Introduction

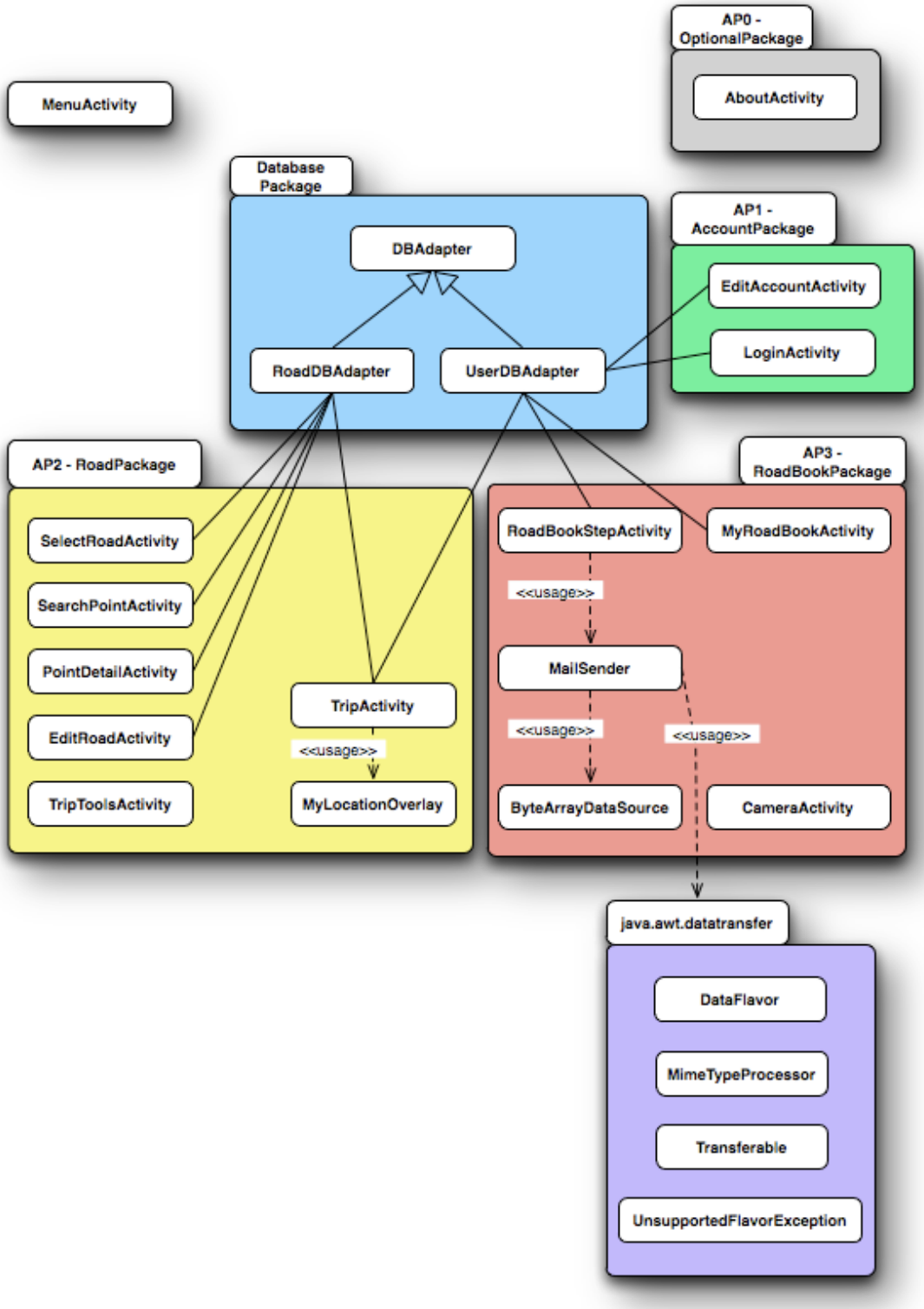

Figure 28 - Diagramme de Class : Vue globale

La structure de l'application est relativement complexe car elle regroupe de nombreuses fonctionnalités différentes.

Les Class de type Activity sont nommées de manière à pouvoir les distinguer des Class normales. Chaque partie de l'application est représentée par un package qui fait référence au différents WorkPackage définis lors de la rédaction du cahier des charges :

- AP0 – OptionalPackage
- AP1 – AccountPackage
- AP2 – RoadPackage
- AP3 – RoadBookPackage

Ainsi que certains packages nécessaires à l'application pour les interactions avec la base de données et pour l'envoi de mail :

- Database Package
- Java.awt.datatransfer

Chaque package est détaillé ci-dessous avec une brève description de chacune des Class et de leur utilité dans le cadre du projet Naviboo – RoadBook.

AP0 – OptionalPackage

Ce package est définit comme optionnel car il ne fait pas parti des éléments exploitant les capacités mises à disposition par l'environnement d'Android. Mais au vue de la planification du projet, j'ai pu me permettre de l'intégrer au projet.

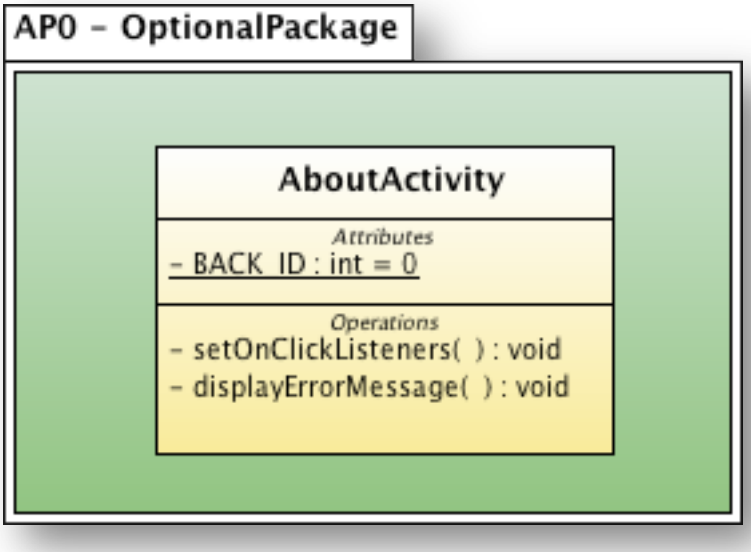

Figure 29 - Diagramme de Class : AP0 - OptionalPackage

Description

OptionalPackage ne se compose que d'une seule Class de type Activity qui permet l'affichage d'un écran d'informations définissant le cadre dans lequel est réalisée l'application Naviboo – RoadBook.

AboutActivity

Cette Activity ne possède aucune interaction avec la base de données ni aucune dépendance à une autre Class.

AboutActivity ne peut démarrer aucune sous-Activity.

Attributs

- BACK_ID : Identifiant attribué au bouton « Back » du menu

Méthodes

- setOnClickListeners : Attribution des OnClickListeners aux différents boutons de l'Activity
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- Back : Retourne à l'écran précédent

AP1 – AccountPackage

Ce package englobe toute la partie relative à l'utilisateur.

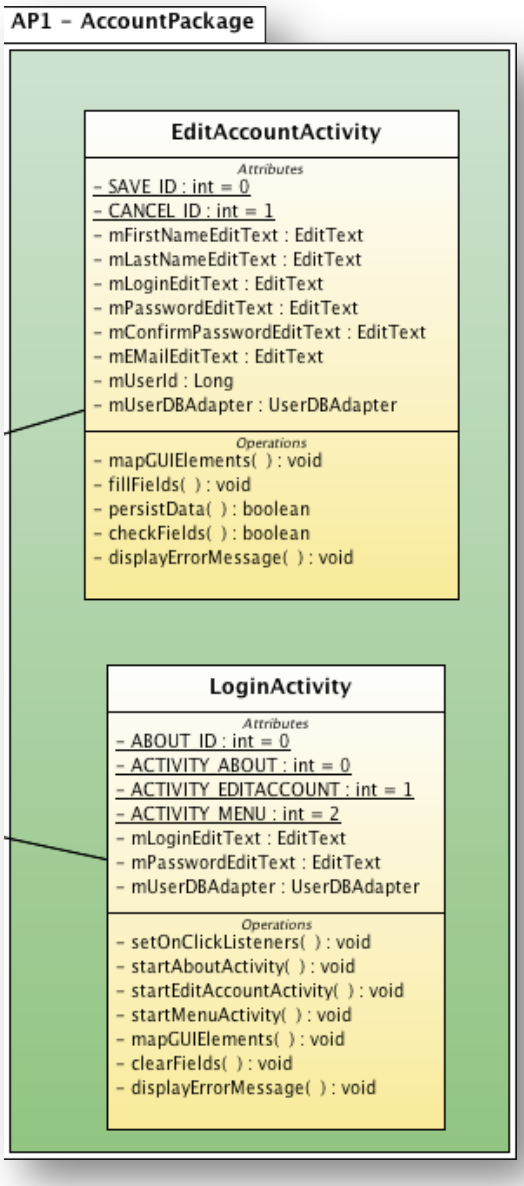

Figure 30 - Diagramme de Class : AP1 - AccountPackage

Description

AccountPackage est composé de deux Class de type Activity, l'une permettant l'identification à l'aide de l'identifiant et du mot de passe, et la seconde permettant de créer ou modifier le compte utilisateur.

LoginActivity

Cette Activity possède des interactions avec la base de données. Elle possède une instance de la Class UserDBAdapter qui lui permet d'effectuer toutes les interactions relatives à l'utilisateur dans le cadre de l'identification de celui-ci.

LoginActivity peut démarrer trois sous-Activity différentes : AboutActivity, EditAccountActivity et MenuActivity.

Attributs

- ABOUT_ID : Identifiant attribué au bouton « About » du menu
- ACTIVITY_ABOUT : Identifiant attribué à AboutActivity.class
- ACTIVITY_EDITACCOUNT : Identifiant attribué à EditAccountActivity.class
- ACTIVITY_MENU : Identifiant attribué à MenuActivity.class
- mLoginEditText : Widget de type EditText lié à l'élément « txt_LoginStart »
- mPasswordEditText : Widget de type EditText lié à l'élément « txt_PasswordStart »
- mUserDBAdapter : Instance de la Class UserDBAdapter

Méthodes

- mapGUIElements : Récupération des différents Widgets de l'interface graphique
- setOnClickListeners : Attribution des OnClickListeners aux différents boutons de l'Activity
- startAboutActivity : Instanciation de la sous-Activity de type AboutActivity.class
- startEditAccountActivity : Instanciation de la sous-Activity de type EditAccountActivity.class
- startMenuActivity : Instanciation de la sous-Activity de type MenuActivity.class
- clearFields : Remise à vide des champs de saisie
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- About : Ouvre l'écran fournissant les informations relatives à l'application

EditAccountActivity

Cette Activity possède des interactions avec la base de données. Elle possède une instance de la Class UserDBAdapter qui lui permet de créer et mettre à jour les informations relatives au compte des utilisateurs.

EditAccountActivity ne peut démarrer aucune sous-Activity.

Attributs

- SAVE_ID : Identifiant attribué au bouton « Save » du menu
- CANCEL_ID : Identifiant attribué au bouton « Cancel » du menu
- mFirstNameEditText : Widget de type EditText lié à l'élément « txt_Firstname »
- mLastNameEditText : Widget de type EditText lié à l'élément « txt_LastName »
- mLoginEditText : Widget de type EditText lié à l'élément « txt_Login »
- mPasswordEditText : Widget de type EditText lié à l'élément « txt_Password »
- mConfirmPasswordEditText : Widget de type EditText lié à l'élément « txt_ConfPassword »
- mEMailEditText : Widget de type EditText lié à l'éllément « txt_EMail »
- mUserId : Identifiant de l'utilisateur chargé
- mUserDBAdapter : Instance de la Class UserDBAdapter

Méthodes

- mapGUIElements : Récupération des différents Widgets de l'interface graphique
- fillFields : Récupération et affichage des informations relatives à l'utilisateur chargé

- persistData : Sauvegarde des données du compte utilisateur dans la base de données
- checkFields : Vérification et validation de la saisie de l'utilisateur
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- Save : Sauvegarde des informations saisies dans la base de données
- Cancel : Annule l'action en cours et retourne à l'écran précédent

AP2 – RoadPackage

Ce package englobe toute la partie relative aux parcours.

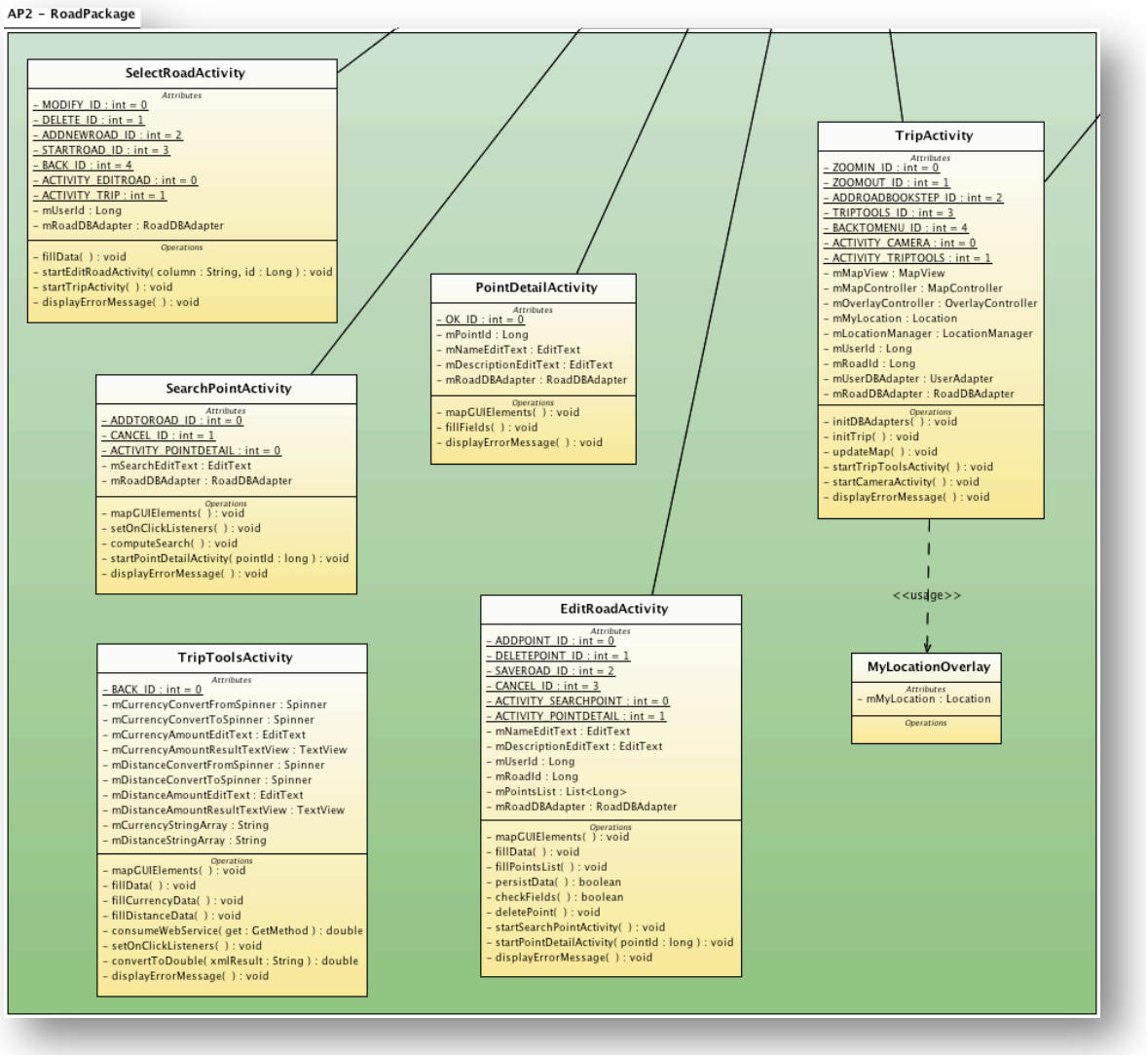

Figure 31 - Diagramme de Class : AP2 - RoadPackage

Description

RoadPackage est composé de sept Class dont six sont de type Activity et la dernière est une Class de type Overlay. Les différentes Activity permettent à l'utilisateur de créer, modifier et gérer ses parcours ainsi que d'effectuer sa visite et profiter de l'outil de voyage. La Class de type Overlay quant à elle permet de gérer la couche d'affichage au

dessus de la carte géographique afin d'avoir la position de l'utilisateur et les différents points d'intérêt du parcours chargé.

SelectRoadActivity

Cette Activity possède des interactions avec la base de données. Elle possède une instance de la Class RoadDBAdapter afin de gérer tous les parcours créés par l'utilisateur.

SelectRoadActivity peut démarrer deux sous-Activity différentes : EditRoadActivity et TripActivity.

Attributs

- MODIFY_ID : Identifiant attribué au bouton « Modify » du menu
- DELETE_ID : Identifiant attribué au bouton « Delete » du menu
- ADDNEWROAD_ID : Identifiant attribué au bouton « Add new Road » du menu
- STARTROAD_ID : Identifiant attribué au bouton « Start » du menu
- BACK_ID : Identifiant attribué au bouton « Back » du menu
- ACTIVITY_EDITROAD : Identifiant attribué à EditRoadActivity.class
- ACTIVITY_TRIP : Identifiant attribué à TripActivity.class
- mUserId : Identifiant de l'utilisateur chargé
- mRoadDBAdapter : Instance de la Class RoadDBAdapter

Méthodes

- fillData : Récupération des données relatives à l'utilisateur et attribution de la source de données créée à la ListView
- startEditRoadActivity : Instanciation de la sous-Activity de type EditRoadActivity.class
- startTripActivity : Instanciation de la sous-Activity de type TripActivity.class
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- Modify : Modification du parcours sélectionné
- Delete : Suppression du parcours sélectionné
- Add new Road : Ajout d'un nouveau parcours à la liste
- Start : Charger le parcours sélectionné afin de commencer la visite
- Back : Retour à l'écran précédent

EditRoadActivity

Cette Activity possède des interactions avec la base de données. Elle possède une instance de la Class RoadDBAdapter afin de permettre à l'utilisateur de créer et modifier un parcours et le sauvegarder.

EditRoadActivity peut démarrer deux sous-Activity différentes : SearchPointActivity et PointDetailActivity.

Attributs

- ADDPOINT_ID : Identifiant attribué au bouton « Add Point » du menu
- DELETEPOINT_ID : Identifiant attribué au bouton « Delete Point » du menu
- SAVEROAD_ID : Identifiant attribué au bouton « Save Road » du menu
- CANCEL_ID : Identifiant attribué au bouton « Cancel » du menu

- ACTIVITY_SEARCHPOINT : Identifiant attribué à SearchPointActivity.class
- ACTIVITY_POINTDETAIL : Identifiant attribué à PointDetail.class
- mNameEditText : Widget de type EditText lié à l'élément « txt_RoadName »
- mDescriptionEditText : Widget de type EditText lié à l'élément « txt_RoadDescription »
- mUserId : Identifiant de l'utilisateur chargé
- mRoadId : Identifiant du parcours chargé
- mPointsList : Liste contenant les points d'intérêt composant le parcours
- mRoadDBAdapter : Instance de la Class RoadDBAdapter

Méthodes

- mapGUIElements : Récupération des différents Widgets de l'interface graphique
- fillData : Récupération des données relatives au parcours et attribution de la source de données créée à la ListView
- fillPointsList : Stocke les différents points du parcours dans la variable de Class mPointsList
- persistData : Sauvegarde du parcours dans la base de données
- checkFields : Contrôle de la saisie de l'utilisateur
- deletePoint : Suppression du point d'intérêt sélectioné de la liste des points composant le parcours
- startSearchPointActivity : Instanciation de la sous-Activity de type SearchPointActivity.class
- startPointDetailActivity : Instanciation de la sous-Activity de type PointDetailActivity.class
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- Add Point : Ajout d'un point d'intérêt au parcours
- Delete Point : Suppression du point d'intérêt sélectionné de la liste des points composant le parcours
- Save Road : Sauvegarde du parcours dans la base de données
- Cancel : Annule l'action en cours et retour à l'écran précédent

PointDetailActivity

Cette Activity possède des interactions avec la base de données. Elle possède une instance de la Class RoadDBAdapter afin de charger toutes les informations relatives au point d'intérêt sélectionné depuis la base de données.

PointDetailActivity ne peut démarrer aucune sous-Activity.

Attributs

- OK_ID : Identifiant attribué au bouton « OK » du menu
- mNameEditText : Widget de type EditText lié à l'élément « txt_PointName »
- mDescriptionEditText : Widget de type EditText lié à l'élément « txt_PointDescription »
- mPointId : Identifiant du point d'intérêt chargé
- mRoadDBAdapter : Instance de la Class RoadDBAdapter

Méthodes

- mapGUIElements : Récupération des différents Widgets de l'interface graphique

- fillFields : Récupération et affichage des informations relatives au point d'intérêt chargé
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- OK : Retour à l'écran précédent

SearchPointActivity

Cette Activity possède des interactions avec la base de données. Elle possède une instance de la Class RoadDBAdapter afin de charger tous les points d'intérêt correspondant aux critères de recherche saisis.

SearchPointActivity peut démarrer une sous-Activity : PoinDetailActivity.

Attributs

- ADDTOROAD_ID : Identifiant attribué au bouton « Add to Road » du menu
- CANCEL_ID : Identifiant attribué au bouton « Cancel » du menu
- ACTIVITY_POINTDETAIL : Identifiant attribué à PointDetailActivity.class
- mSearchEditText : Widget de type EditText lié à l'élément « txt_Search »
- mRoadDBAdapter : Instance de la Class RoadDBAdapter

Méthodes

- mapGUIElements : Récupération des différents Widgets de l'interface graphique
- setOnClickListeners : Attribution des OnClickListeners aux différents boutons de l'Activity
- computeSearch : Recherche dans la base de données des points d'intérêt correspondant à saisie de l'utilisateur
- startPointDetailActivity : Instanciation de la sous-Activity de type PointDetailActivity
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- Add to Road : Ajout du point d'intérêt sélectionné au parcours en cours de modification
- Cancel : Annule l'action en cours et retour à l'écran précédent

TripActivity

Cette Activity possède des interactions avec la base de données. Elle possède une instance de la Class UserDBAdapter afin de charger les informations relatives à l'utilisateur identifié et une instance de la Class RoadDBAdapter afin de charger et afficher le parcours sélectionné.

TripActivity peut démarrer deux sous-Activity différentes : CameraActivity et TripToolsActivity.

Attributs

- ZOOMIN_ID : Identifiant attribué au bouton « Zoom In » du menu
- ZOOMOUT_ID : Identifiant attribué au bouton « Zoom Out » du menu
- ADDROADBOOKSTEP_ID : Identifiant attribué au bouton « Add RoadBook Step » du menu

- TRIPTOOLS_ID : Identifiant attribué au bouton « Trip Tools » du menu
- BACKTOMENU_ID : Identifiant attribué au bouton « Back to Menu » du menu
- ACTIVITY_CAMERA : Identifiant attribué à CameraActivity.class
- ACTIVITY_TRIPTOOLS : Identifiant attribué à TripToolsActivity.class
- mMapView : Element permettant l'affichage de la carte géographique
- mMapController : Element permettant de gérer la MapView
- mOverlayController : Element permettant de gérer la couche d'affichage d'éléments sur la MapView
- mMyLocation : Position GPS actuelle de l'utilisateur
- mLocationManager : Element permettant de gérer la simulation de données GPS
- mUserId : Identifiant de l'utilisateur chargé
- mRoadId : Identifiant du parcours chargé
- mUserDBAdapter : Instance de la Class UserDBAdapter
- mRoadDBAdapter : Instance de la Class RoadDBAdapter

Méthodes

- initDBAdapters : Instanciation des deux DBAdapters nécessaire aux traitements avec la base de données
- initTrip : Initialisation du parcours ainsi que de l'affichage des différents éléments de la carte géographique
- updateMap : Rafraîchissement de la carte ainsi que de l'affichage de la position de l'utilisateur
- startTripToolsActivity : Instanciation de la sous-Activity de type TripToolsActivity
- startCameraActivity : Instanciation de la sous-Activity de type CameraActivity
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- Zoom In : Augmentation du niveau de zoom de la carte
- Zoom Out : Diminution du niveau de zoom de la carte
- Add RoadBook Step : Ajout d'une nouvelle étape au carnet de route
- Trip Tools : Ouverture de l'outil de voyage
- Back to Menu : Retour au menu de l'application

MyLocationOverlay

Cette Class de type Overlay a pour fonction d'afficher des éléments sur une carte géographique.

Les éléments dessinés sont les différents points d'intérêt composant le parcours ainsi que la position de l'utilisateur.

TripToolsActivity

Cette Activity ne possède aucune interaction avec la base de données. Les outils de conversion consomment des Web Service afin de fournir à l'utilisateur les résultats souhaités.

TripToolsActivity ne peut démarrer aucune sous-Activity.

Attributs

- BACK_ID : Identifiant attribué au bouton « Back » du menu

- mCurrencyConvertFromSpinner : Widget de type Spinner lié à l'élément « currencyConvertFrom_spinner »
- mCurrencyConvertToSpinner : Widget de type Spinner lié à l'élément « currencyConvertTo_spinner »
- mCurrencyAmountEditText : Widget de type EditText lié à l'élément « txt_CurrencyAmount »
- mCurrencyAmountResultTextView : Widget de type TextView lié à l'élément « currencyAmountResult_text »
- mDistanceConvertFromSpinner : Widget de type Spinner lié à l'élément « distanceConvertFrom_spinner »
- mDistanceConvertToSpinner : Widget de type Spinner lié à l'élément « distanceConvertTo_spinner »
- mDistanceAmountEditText : Widget de type EditText lié à l'élément txt_DistanceAmount
- mDistanceAmountResultTextView : Widget de type TextView lié à l'élément « distanceAmountResult_text »
- mCurrencyStringArray : Tableau contenant les différentes monnaies gérées par l'outil
- mDistanceStringArray : Tableau contenant les différentes unités de mesure gérées par l'outil

Méthodes

- mapGUIElements : Récupération des différents Widgets de l'interface graphique
- setOnClickListeners : Attribution des OnClickListeners aux différents boutons de l'Activity
- fillData : Récupération et attribution des données pour les conversions
- fillCurrencyData : Attribution des différentes unités monétaires aux listes déroulantes correspondantes
- fillDistanceData : Attribution des différentes unités de mesure aux listes déroulantes correspondantes
- consumeWebService : Consommation du WebService désiré avec les paramètres saisis
- convertToDouble : Conversion du résultat du WebService en une variable de type Double
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- Back : Retour à l'écran précédent

AP3 – RoadBookPackage

Ce package englobe toute la partie relative au carnet de route.

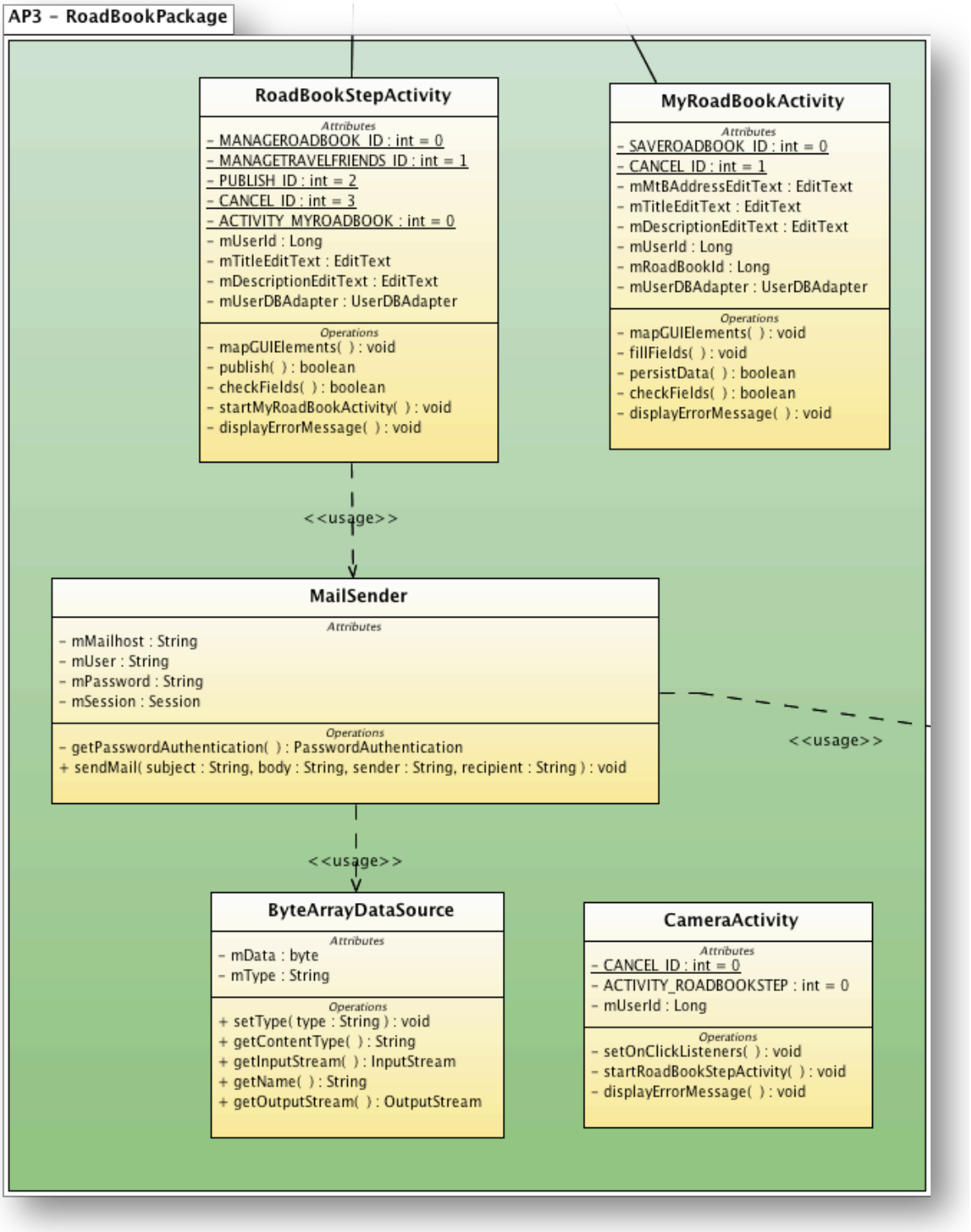

Figure 32 - Diagramme de Class : AP3 - RoadBookPackage

Description

RoadBookPackage est composé de cinq Class dont trois sont de type Activity, une de type Authenticator et la dernière de type DataSource. Les différentes Activity permette à

l'utilisateur de gérer son carnet de route, de choisir une photo à attribuer à une étape du carnet de route et de publier chaque étape sur un Blog. La Class de type Authenticator permet la création et l'envoi d'un mail et la Class de type DataSource est utilisée pour la manipulation des données nécessaire à l'objet MimeMessage représentant le mail à envoyer.

MyRoadBookActivity

Cette Activity possède des interactions avec la base de données. Elle possède une instance de la Class UserDBAdapter afin que l'utilisateur puisse gérer et configurer son carnet de route.

MyRoadBookActivity ne peut démarrer aucune sous-Activity.

Attributs

- SAVEROADBOOK_ID : Identifiant attribué au bouton « Save my RoadBook » du menu
- CANCEL_ID : Identifiant attribué au bouton « Cancel » du menu
- mMtBAddressEditText : Widget de type EditText lié à l'élément « txt_MailToBlogger »
- mTitleEditText : Widget de type EditText lié à l'élément « txt_RoadBookTitle »
- mDescriptionEditText : Widget de type EditText lié à l'élément « txt_RoadBookDescription »
- mUserId : Identifiant de l'utilisateur chargé
- mRoadBookId : Identifiant du carnet de route chargé
- mUserDBAdapter : Instance de la Class UserDBAdapter

Méthodes

- mapGUIElements : Récupération des différents Widgets de l'interface graphique
- fillFields : Récupération et affichage des informations relatives au carnet de route chargé
- persistData : Sauvegarde du carnet de route dans la base de données
- checkFields : Vérification et validation de la saisie de l'utilisateur
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- Save my RoadBook : Sauvegarde du carnet de route
- Cancel : Annulation de l'action en cours et retour à l'écran précédent

CameraActivity

Cette Activity ne possède aucune interaction avec la base de données.

CameraActivity peut démarrer une seule sous-Activity : RoadBookStepActivity.

Attributs

- CANCEL_ID : Identifiant attribué au bouton « Cancel » du menu
- ACTIVITY_ROADBOOKSTEP : Identifiant attribué à RoadBookStepActivity.class
- mUserId : Identifiant de l'utilisateur chargé

Méthodes

- setOnClickListeners : Attribution des OnClickListeners aux différents boutons de l'Activity

- startRoadBookStepActivity : Instanciation de la sous-Activity de type RoadBookStepActivity
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- Cancel : Annulation de l'action en cours et retour à l'écran précédent

RoadBookStepActivity

Cette Activity possède des interactions avec la base de données. Elle possède une instance de la Class UserDBAdapter afin de pouvoir publier une étape du carnet de route sur le Blog de l'utilisateur.

RoadBookStepActivity peut démarrer une sous-Activity : MyRoadBookActivity.

Attributs

- MANAGEROADBOOK_ID : Identifiant attribué au bouton « Manage RoadBook » du menu
- MANAGETRAVELFRIENDS_ID : Identifiant attribué au bouton « Manage Travel Friends » du menu
- PUBLISH_ID : Identifiant attribué au bouton « Publish » du menu
- CANCEL_ID : Identifiant attribué au bouton « Cancel » du menu
- ACTIVITY_MYROADBOOK : Identifiant attribué à MyRoadBookActivity.class
- mUserId : Identifiant de l'utilisateur chargé
- mTitleEditText : Widget de type EditText lié à l'élément « txt_RoadBookStepTitle »
- mDescriptionEditText : Widget de type EditText lié à l'élément « txt_RoadBookStepDescription »
- mUserDBAdapter : Instance de la Class UserDBAdapter

Méthodes

- mapGUIElements : Récupération des différents Widgets de l'interface graphique
- publish : plublication de l'étape du carnet de route sur le Blog de l'utilisateur
- checkFields : Vérification et validation de la saisie de l'utilisateur
- startMyRoadBookActivity : Instanciation de la sous-Activity de type MyRoadBookActivity
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Menu

- Manage RoadBook : Gestion du carnet de route
- Manage Travel Friends : Gestion des amis de voyage présent pour l'étape du carnet de route
- Publish : Publication de l'étape sur le Blog
- Cancel : Annulation de l'action en cours et retour à l'écran précédent

MailSender

Class standard utilisée pour créer et envoyer un mail.

Cette Class est utilisée par RoadBookStepActivity afin de permettre à l'utilisateur de publier un article du carnet de route sur un Blog à l'aide d'une adresse Mail-to-Blogger.

Attributs

- mMailHost : Hôte utilisé pour l'envoi du mail
- mUser : Identifiant utilisé pour l'authentification auprès du serveur SMTP
- mPassword : Mot de passe utilisé pour l'authentification auprès du serveur SMTP
- mSession : Session utilisé pour l'envoie du mail

Méthodes

- getPasswordAuthentication : Méthode nécessaire à l'authentification auprès du serveur SMTP
- sendMail : Envoi du mail composé des informations saisies par l'utilisateur

ByteArrayDataSource

Class de type DataSource utilisée pour la création du mail à envoyer.

Cette Class est utilisée par MailSender afin de créer l'objet MimeMessage représentant le mail à envoyer à l'adresse saisie par l'utilisateur.

Attributs

- mData : Tableau de données spécifique au DataSource créé
- mType : Type de données manipulées par le DataSource créé

Méthodes

Les méthodes définies dans cette Class sont nécessaires à la manipulation du DataSource créé dans le cadre du projet.

MenuActivity

Cette Activity n'est intégrée à aucun des packages étant donné qu'elle fait le lien entre chacun d'eux.

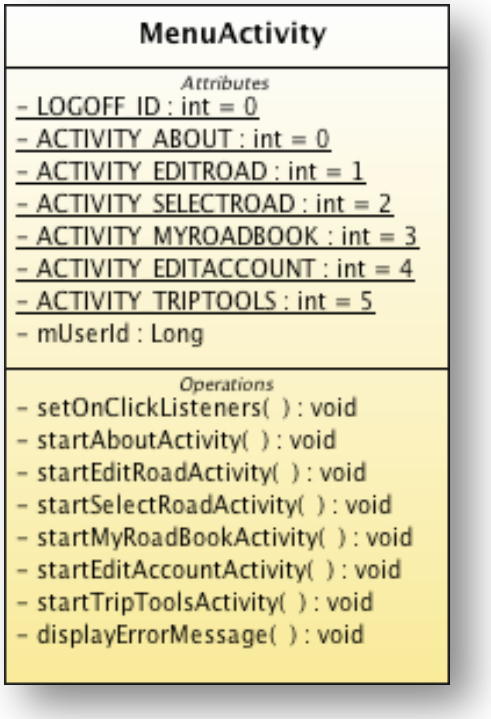

Figure 33 - Diagramme de Class : MenuActivity

Description

Cette Activity est le menu principal de l'application, c'est de cet endroit que l'utilisateur peut effectuer toutes les actions qu'il souhaite.

MenuActivity peut démarrer six sous-Activity différentes : AboutActivity, EditRoadActivity, SelectRoadActivity, MyRoadBookActivity, EditAccountActivity et TripToolsActivity.

Attributs

- LOGOFF_ID : Identifiant attribué au bouton « Log off » du menu
- ACTIVITY_ABOUT : Identifiant attribué à AboutActivity.class
- ACTIVITY_EDITROAD : Identifiant attribué à EditRoadActivity.class
- ACTIVITY_SELECTROAD : Identifiant attribué à SelectRoadActivity.class
- ACTIVITY_MYROADBOOK : Identifiant attribué à MyRoadBook.class
- ACTIVITY_EDITACCOUNT : Identifiant attribué à EditAccount.class
- ACTIVITY_TRIPTOOLS : Identifiant attribué à TripToolsActivity.class
- mUserId : Identifiant de l'utilisateur chargé

Méthodes

- setOnClickListeners : Attribution des OnClickListeners aux différents boutons de l'Activity
- startAboutActivity : Instanciation de la sous-Activity de type AboutActivity
- startEditRoadActivity : Instanciation de la sous-Activity de type EditRoadActivity
- startSelectRoadActivity : Instanciation de la sous-Activity de type SelectRoadActivity
- startMyRoadBookActivity : Instanciation de la sous-Activity de type MyRoadBookActivity
- startEditAccountActivity : Instanciation de la sous-Activity de type EditAccountActivity
- startTripToolsActivity : Instanciation de la sous-Activity de type TripToolsActivity
- displayErrorMessage : Affichage d'un pop-up indiquant à l'utilisateur qu'une erreur est survenue

Package java.awt.datatransfer

Ce package est importé du SDK Java afin de permettre l'envoi de mail depuis l'application Naviboo – RoadBook.

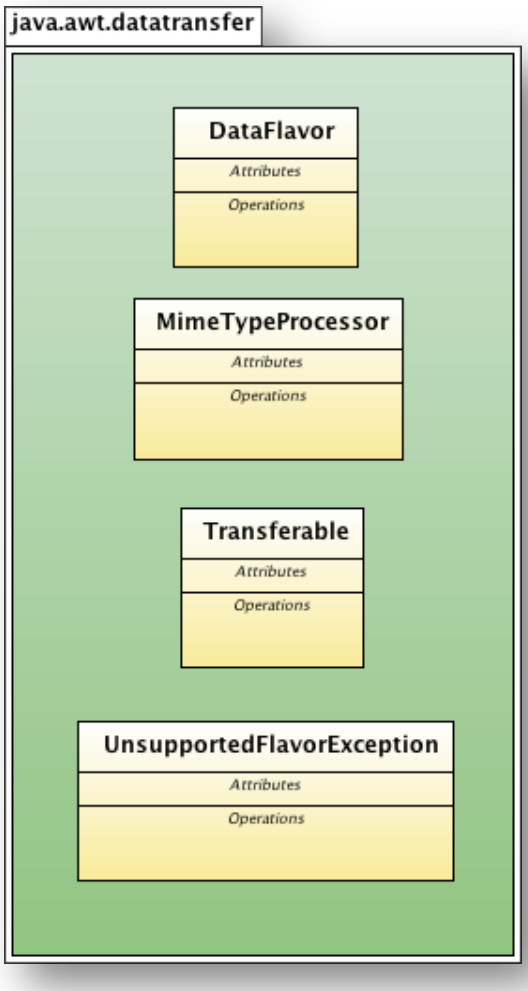

Figure 34 - Diagramme de Class : Package java.awt.datatransfer

Le détail de ce package n'est pas fournit car il est disponible dans la Javadoc s'y rapportant.

Database Package

Ce package englobe toute la partie relative aux interactions avec la base de données.

Description

Database Package est composé de trois Class qui permettent à l'application de manipuler la base de données SQLite embarquée sur le mobile Android.

DBAdapter

Cette Class possède toutes les informations nécessaires à la création de la base de données ainsi que le script de création de chacune des tables utiles pour l'application : user, roadBook, road, roadStep et point. Elle est également responsable de l'ouverture de la connexion avec la base de données ainsi que la fermeture.

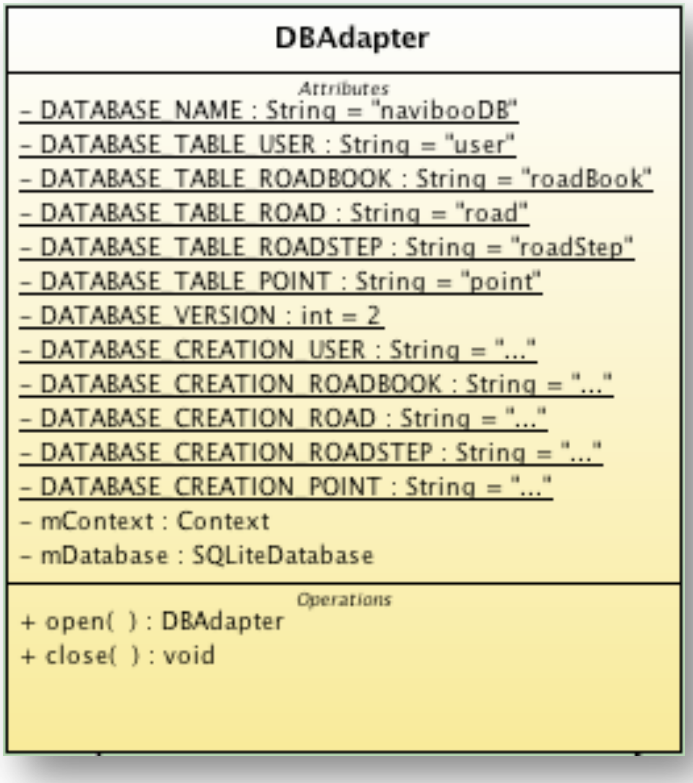

Figure 35 - Diagramme de Class : Database Package - DBAdapter

Attributs

- DATABASE_NAME : Nom de la base de données
- DATABASE_TABLE_USER : Nom de la table utilisateur
- DATABASE_TABLE_ROADBOOK : Nom de la table carnet de route
- DATABASE_TABLE_ROAD : Nom de la table parcours
- DATABASE_TABLE_ROADSTEP : Nom de la table intermédiaire étape du parcours
- DATABASE_TABLE_POINT : Nom de la table point d'intérêt
- DATABASE_VERSION : Version de la base de données
- DATABASE_CREATION_USER : Script de création de la table utilisateur
- DATABASE_CREATION_ROADBOOK : Script de création de la table carnet de route
- DATABASE_CREATION_ROAD : Script de création de la table parcours
- DATABASE_CREATION_ROADSTEP : Script de création de la table intermédiaire étape du parcours
- DATABASE_CREATION_POINT : Script de création de la table point d'intérêt
- mContext : Contexte nécessaire pour l'ouverture de la base de données
- mDatabase : Base de données manipulée

Méthodes

- open : Ouverture de la connexion avec la base de données
- close : Fermeture de la connexion avec la base de données

UserDBAdapter

Cette Class hérite de la Class DBAdapter. Elle possède toutes les informations relatives à la partie utilisateur de la base de données : Interactions avec la table utilisateur et la table carnet de route.

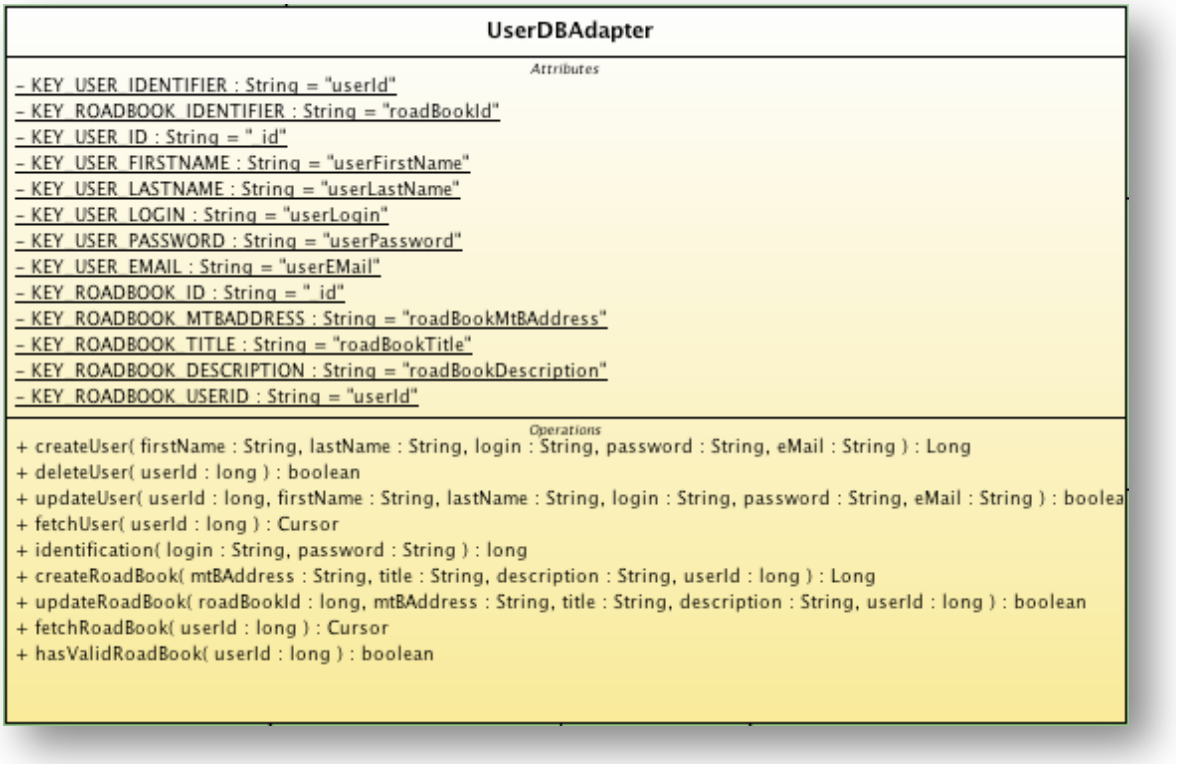

Figure 36 - Diagramme de Class : Database Package – UserDBAdapter

Attributs

- KEY_USER_IDENTIFIER : Identifiant de la variable utilisateur
- KEY_ROADBOOK_IDENTIFIER : Identifiant de la variable carnet de route
- KEY_USER_ID : Champ « clé primaire » de la table utilisateur
- KEY_USER_FIRSTNAME : Champ « prénom » de la table utilisateur
- KEY_USER_LASTNAME : Champ « nom de famille » de la table utilisateur
- KEY_USER_LOGIN : Champ « identifiant » de la table utilisateur
- KEY_USER_PASSWORD : Champ « mot de passe » de la table utilisateur
- KEY_USER_EMAIL : Champ « eMail » de la table utilisateur
- KEY_ROADBOOK_ID : Champ « clé primaire » de la table carnet de route
- KEY_ROADBOOK_MTBADDRESS : Champ « adresse Mail-to-Blogger » de la table carnet de route
- KEY_ROADBOOK_TITLE : Champ « titre » de la table carnet de route
- KEY_ROADBOOK_DESCRIPTION : Champ « description de la table carnet de route
- KEY_ROADBOOK_USERID : Champ « identifiant utilisateur » de la table carnet de route

Méthodes

- createUser : Création d'un nouvel utilisateur
- deleteUser : Suppression d'un utilisateur
- updateUser : Mise à jour d'un utilisateur existant
- fetchUser : Recherche d'un utilisateur
- identification : Vérification de l'identifiant et du mot de passe de l'utilisateur
- createRoadBook : Création d'un nouveau carnet de route
- updateRoadBook : Mise à jour d'un carnet de route existant
- fetchRoadBook : Recherche d'un carnet de route
- hasValidRoadBook : Vérifie si l'utilisateur a un carnet de route valide

RoadDBAdapter

Cette Class hérite de la Class DBAdapter. Elle possède toutes les informations relatives à la partie parcours de la base de données : Interactions avec la table parcours, la table étape du parcours et la table point d'intérêt.

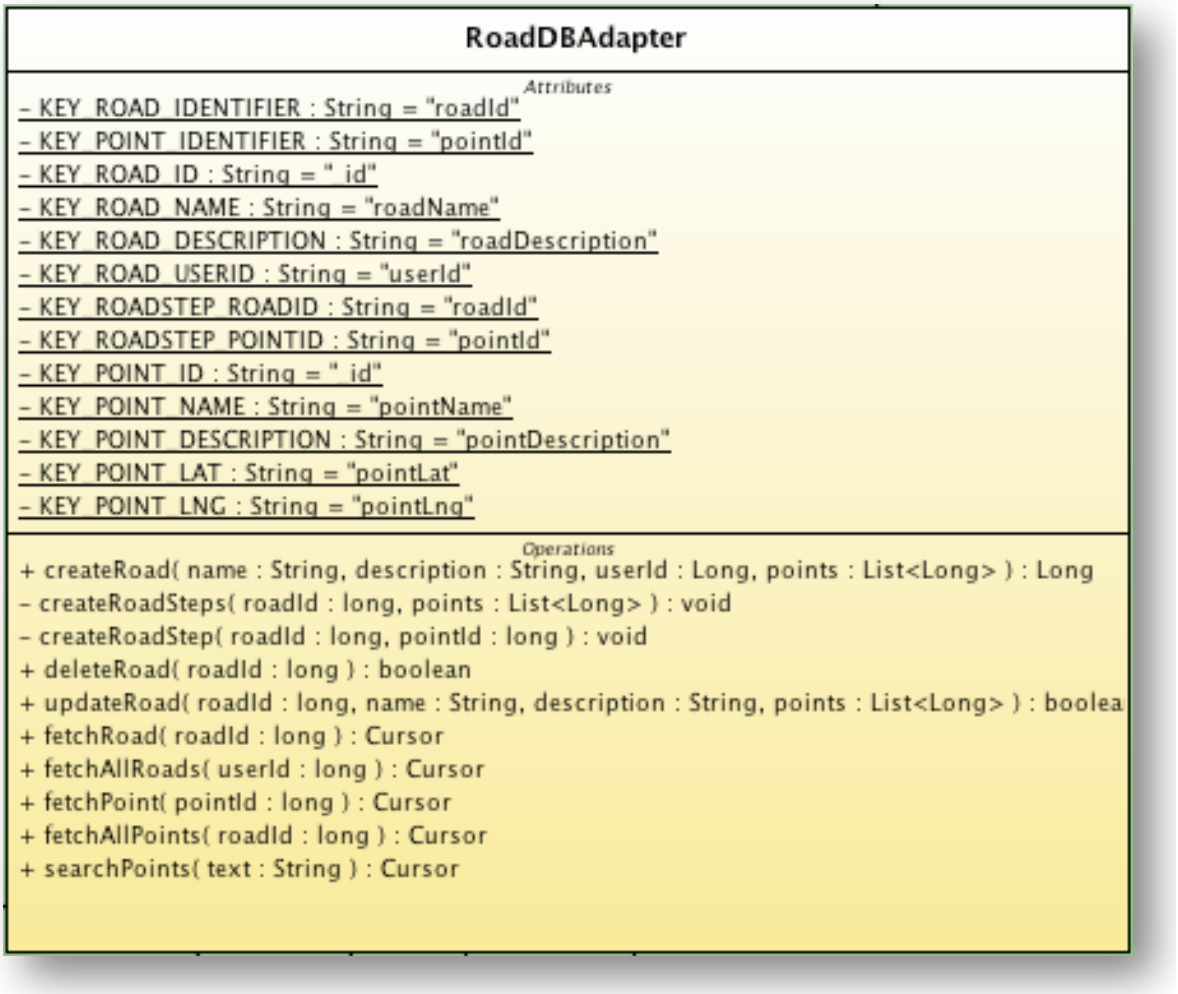

Figure 37 - Diagramme de Class : Database Package - RoadDBAdapter

Attributs

- KEY_ROAD_IDENTIFIER : Identifiant de la variable parcours
- KEY_POINT_IDENTIFIER : Identifiant de la variable point d'intérêt
- KEY_ROAD_ID : Champ « clé primaire » de la table parcours
- KEY_ROAD_NAME : Champ « nom » de la table parcours
- KEY_ROAD_DESCRIPTION : Champ « description » de la table parcours
- KEY_ROAD_USERID : Champ « identifiant utilisateur » de la table parcours
- KEY_ROADSTEP_ROADID: Champ « identifiant parcours » de la table étape du parcours
- KEY_ROADSTEP_POINTID : Champ « identifiant point d'intérêt » de la table étape du parcours
- KEY_POINT_ID : Champ « clé primaire » de la table point d'intérêt
- KEY_POINT_NAME : Champ « nom » de la table point d'intérêt
- KEY_POINT_DESCRIPTION : Champ « description de la table point d'intérêt
- KEY_POINT_LAT : Champ « latitude » de la table point d'intérêt

- KEY_POINT_LNG : Champ « longitude » de la table point d'intérêt

Méthodes

- createRoad : Création d'un nouveau parcours
- createRoadSteps : Création des étapes du parcours
- createRoadStep : Création d'une étape du parcours
- deleteRoad : Suppression d'un parcours
- updateRoad : Mise à jour d'un parcours existant
- fetchRoad : Recherche d'un parcours
- fetchAllRoads : Recherche de tous les parcours créés par un utilisateur
- fetchPoint : Recherche d'un point d'intérêt
- fetchAllPoints : Recherche de tous les points d'intérêts d'un parcours
- searchPoints : Recherche des points d'intérêt correspondant à une chaine de caractères

4. Table des illustrations

Tâches	Durée est.	Sem 01 (20h)	Sem 02 (20h)	Sem 03 (20h)	Sem 04 (20h)	Sem 05 (45h)	Sem 06 (45h)	Durée eff.	RàF	Ecart
Administratif, premier contact	2							2	0	-
WP1 - Analyse/Etude de la plateforme Android										
Découverte et recherche possibilités Android	14							14	0	-
Préparation des documents de suivis	4							4	0	-
Recherche de documentation/informations	12							12	0	-
Analyse des possibilités offertes	12							12	0	-
Etude de faisabilité des différentes possibilités	22							22	0	-
Découverte du framework Android	8							8	0	-
WP2 - Développement du prototype										
Rédaction d'un scénario de prototype	6							6	0	-
Rédaction du cahier des charges de l'application	4							4	0	-
Formation à l'aide de tutoriels, exemples, etc..	26							26	0	-
AP1 - Account package 1.1 (Iteration 1)	9							9	0	-
AP2 - Road package 2.1 (Iteration 1)	18							18	0	-
AP3 - RoadBook package 3.1 (Iteration 1)	18							18	0	-
AP3 - RoadBook package 3.2 (Iteration 2)	5							5	0	-
Divers										
Documents, correspondance, etc..	20							20	0	-
										-
										-
										-
										-
										-
										-
										-
Total (180h à disposition)	180							180		

Nbre d'heures effectuées : 360 Ecart avec la planification : -
Nbre d'heures à dispositior 0 Nbre d'heures restant à faire : 0

Tâches	Durée est.	Sem 07 (45h)	Sem 08 (45h)	Sem 09 (45h)	Sem 10 (45h)	Durée eff.	RàF	Ecart
WP2 - Développement du prototype (2)								
AP3 - RoadBook package 3.2 (Iteration 2)	15					15	0	-
AP2 - Road package 2.2 (Iteration 2)	20					20	0	-
AP1 - Account package 1.2 (Iteration 2)	15					15	0	-
AP2 - Road package 2.3 (Iteration 3)	20					20	0	-
AP2 - Road package 2.4 (Iteration 4)	22					22	0	-
Tests et finalisation du prototype	10					10	0	-
WP3 - Analyse, Critique, Prise de position								
Analyse Android sur base du vécu	20					20	0	-
Critique de l'outil	10					10	0	-
Prise de position, regard critique	6					6	0	-
WP4 - Rédaction de la documentation								
Documentation du prototype	10					10	0	-
Rapport Final	16					16	0	-
Autre documentation(tuto,etc..)	8					8	0	-
Divers								
Documents, correspondance, etc..	8					8	0	-
								-
								-
								-
								-
								-
								-
								-
Total (180h à disposition)	180					180		

Nbre d'heures effectuées : 360 Ecart avec la planification : -
Nbre d'heures à disposition 0 Nbre d'heures restant à faire : 0