République Tunisienne
Ministère de l'Enseignement Supérieur et
de la recherche scientifique

Université de Gabès
Ecole Nationale d'Ingénieurs de Gabès
Département de Génie des
Communications et des Réseaux

ENIG

Projet de Fin d'Etudes
N° d'ordre: 2013

Mémoire de Projet de Fin d'Etudes

Développement d'une application Android pour médecins.

Présenté à

L'Ecole Nationale d'Ingénieurs de Gabès

En vue de l'obtention du

Diplôme National d'Ingénieur en Communications et Réseaux

Réalisé par : Riadh Habbachi

Encadré par : Mr. Ikbel Azaiez
Mr. Aymen Elj

Soutenu le --/--/2013, devant la commission d'examen:

Mr.	*Président*
Mr.	*Membre*
Mr.	*Encadrant*
Mr.	*Invité*

AU : 2012/2013

Table des matières

Liste des tableaux

Table des figures

Liste d'Abréviations

API Application Programing Interface. 5, 9, 14, 15

E-OTD Enhanced Observed Time Difference. 5, 17

GPS Global Positioning System. 5, 16, 17
GPX GPS eXchange Format. 5
GSM Global System for Mobile Communications. 5, 17

KML Keyhole Markup Language. 5

LBS Location Based Services. 5, 16

MIAA Medical Information, Anytime, Anywhere. 5, 9, 10

SDK Software Development Kit. 5, 14, 15

TDoA Time Difference of Arrival. 5, 17
TTM Time-to-Market. 5, 14

Chapitre 1

Introduction Général

Ces temps ci, le mobile s'est imposé et devient la norme pour les consommateurs. les statistiques ne le cache pas, c'était prévisible, mais tout les analystes le soulignes : "Le marché des PC s'effondre face aux smartphones et aux tablettes" [3]. Et un des secteur qui pourrait bénéficier par l'avalanche des systèmes mobile est le secteur médicale.

les applications mobiles offre un potentiel énorme pour supporté et activé des nouvelles opportunité pour les services médicaux sont impressionnante. Localisation, instantanéité, efficacité, personnalisation et une très grande accommodation vont offrir plusieurs moyen nouveau pour amélioré l'expérience des services médicaux, du côté du patient serte, mais tend aussi à rendre l'établissement plus conviviale pour les médecins et en général du staff médicale.

Investir dans une application mobile de représente pour les hôpitaux ,et les institutions qui les implémentes, un autre moyen pour étendre les outils numériques déjà en place. En offrent des fonctionnalités qui sont auparavant cloué aux ordinateurs des administrations, ce qui facilite le processus de traitement des malades.

Ce pendant, l'usage des smartphone dans les établissement est sujet aux questions notamment sur le plan technique. les technique d'accès et de sécurisation des données des patients et divers technologies utilisé et surtout le manque de standardisation pose un sérieux challenge pour les entreprises voulant offrir des solutions pour les établissements médicaux.

Dans ce même thème se présente ce projet de fin l'étude sur la conception et développement d'une application mobile sur plate-forme Android^{TM}destiné aux médecins dans le but de facilité l'accès au dossiers médicaux des patients en intégrants les techniques de localisation. Ce rapport est subdivisé en trois sous parties : La premier parti expose le cadre général de projet en présentant l'entreprise hôte ainsi que les objectifs de l'application. La deuxième partie évoque les solutions similaires déjà présente dans le marché ainsi que une

présentation de la plate-forme pour la quelle l'application est développer. La troisième et dernière parti décortique le travail effectué pour accomplir les objectifs.

Chapitre 2

Cadre Général du Projet

2.1 Introduction

Ce chapitre est subdivisé en deux partis : la première parti est consacré pour à la présentation de l'organisme d'accueil *Tunav*. La deuxième parti est destiné à la présentation du projet en soit et les différents facteurs qui on pesés l'or du passage vers la réalisation.

2.2 Présentation de l'organisme d'accueil

Tunav se situ à la Cité Technologique des Communications Parc Technologique -BP-55 LA GAZELLE 2088 ARIANA, et a était fondé par sont Président Directeur Général Mohamed Anis Kallel.

En guise de présentation, rien de mieux que de l'avoir directement du boss lui même [4] :

"Tunav est une société technologique, créée au mois d'août 2004, implantée à la technopole El Gazala et spécialisée dans la technologie GPS et ses diverses applications dans les domaines de navigation et de gestion de flotte."

"Tunav est connue en Tunisie par son système « LaTrace » de gestion de flotte par GPS, lequel a été commercialisée pour la première fois en Octobre 2005. Il s'agit d'un système articulé autour d'une application très évoluée de gestion de flotte, d'une gamme d'appareils GPS/GPRS et d'une base de données géographique richement renseignée."

Tunav possède un savoir faire reconnu dans le domaine de la localisation qui peux être exploité dans le domaine médical.

2.3 Présentation du projet

2.3.1 Utilisateurs Cibles

Cette application vise principalement les médecins. Et malgré que, suite à des choix conceptuels, rien n'empêche qu'avec des modifications minime une audience plus large dans le corps médical pourrai être ciblées, ce n'ai pas -pour le moment- le but de l'application. Les médecins, malgré leur formation prolongé dans le domaine médicale, représente une cible sans une vrais profondeur technique, ce que requière de l'application d'être le plus simple possible.

2.3.2 Spécification des Besoins

Besoins fonctionnels

- Le médecin doit être capable à partir de son terminal d'avoir des informations sur les patients qui lui sont assigné en fonction de leur position géographique.
- L'application doit être capable de détecté la proximité d'un patient en fonction de la position du terminal.
- Le médecin peut télé-consulter le dossier médical du patient.

Besoins non fonctionnels

- Une bonne ergonomie qui vise à faciliter l'obtention de l'information, avec un minimum d'efforts pour l'utilisateur cible et avec le moindre risque d'erreur. Les choix graphiques et conceptuels sont des considération à tenir en compte.

Besoins techniques

- L'application mobile vise à utilisé les systèmes déjà en place des établissements clients. Vu l'absence de standardisation et les différentes implémentation possible, une certaine abstraction est requise pour pouvoir déployer l'application dans des environnements possible avec le minimum de modification.

2.4 Conclusion

La présentation de l'entreprise nous à permit de mieux cerné les points forts qu'on pourrait compté sur pendant le développement de notre solution.

Et une connaissance exhaustive des objectifs de ce projet offre une base solide nécessaire pour éviter de s'engager dans des fausses pistes.

Chapitre 3

État de l'art

3.1 Introduction

Dans ce chapitre on présente une étude de marché en énumérant les applications dont les fonctionnalités sont équivalentes à la notre tout en soulignent les différences qui subsistent. Ensuite on présente la plate-forme Android™ et en passe en revu l'architecture d'une application android.

3.2 Étude de marché

Plusieurs sociétés offre des solution en relation avec celle proposé par ce présent rapport. Malheureusement, la plus par d'entre elle sont des solutions commerciales et, faute de documentation disponible, on n'a pas pu les étudier d'une point de vu techno-technique et on c'est contenté de relayé leurs caractéristiques tel que présenté dans les sources cités.

NB : Les solutions présenté ici sont le fruit des sociétés bien établis avec des ressources considérable et des salariés professionnels. Les comparé avec le travail incubé dans ce rapport serai abusive, l'indulgence est de mise.

3.2.1 MIAA - Palomar Pomerado Health

MIAA (figure 3.1) est une application mobile issu d'un projet R&D chez *Palomar Pomerado Health*, l'institution public la plus large dans l'état de Californie (USA). Elle permet au médecins d'accédé rapidement au dossier médical complet du patient depuis une variété de source différentes qui sáffranchie des frontières des organisation [5]. Elle vise les terminales équiper

FIGURE 3.1 – MIAA sur un émulateur Cisco Cius

avec le systéme d'exploitation Android™ comme les smartphones et les tablettes. *Palomar Pomerado Health* a choisie de déployer cette application dans le *Palomar Medical Center* à *Escondido* (319 lits) et le *Pomerado Hospital* à *Poway* (107 lits) sur des tablette Cisco Cius [6], ce choix s'est basé sur le support qu'offre Cisco pour ces divers équipements. Les avantages de MIAA sont : [7]

- Application mobile facile à utilisé conçu spécifiquement aux médecins, tournant sur la plate-forme Android™.
- Un service *Cloud* qui fournit un accès permanent à l'historique médicale des patients à partir de divers sources de données qui sáffranchie des frontières des organisations.
- Interopérabilité avec les pionniers des systèmes électroniques de l'historique médicales tel que Cerner - **Millennnium**™, **NextGen**™, et *Veterans Administration* - **VistA**™.
- Intégration en temps-réel des technologies de surveillance des signes vitaux sans fils comme l'ECG, SPO_2, rythme cardiaque, température, respiration, et pression du sang à partir des équipements sans-fils.
- Affichage des information génétique personnel.
- Application dynamique qui s'ajuste automatiquement à lh́ôpital, clinique, au à la maison.
- Simple, facile à utilisé, avec une User Interface (UI) tactile de nouvelle génération.
- Intégration d'une messagerie inter-médecins sécurisé tout en maintenant le contexte du patient.
- Des plan future pour intégré NHIN *Connect* et les services *Direct*.

3.2.2 PowerChart Touch™ - Cerner

PowerChart Touch™ est une solution mobile conçu par le laboratoire Cerner qui fait parti de l'ensemble de solutions **Millennium+**™ et qui permet de facilité le travail des médecins. Elle offre une expérience native sur iPad pour géré les visites médicales et permet aux médecins d'effectuer tout une visite typique qui inclue : [8]

- Consultation des emplois du temps et les chartes des patients.
- Satisfaire les demandes récurrentes comme les commandes simple et les recharges des médicaments.
- Consultation des diagnostiques et résultats cliniques.

– Documenté les allergies, les problèmes de santé et l'historique du patient.
– Crée et signé les notes de progressions.

Dé la fin du flux de travail du médecin ambulant. Cerner étend ces même fonctions et les adaptes aux établissements hospitalier, les urgences et les divers spécialistes. Les avantages clés du PowerChart Touch™ sont : [8]

– Des réponses instantanés avec un flux de travail aisée.
– Pas besoin de configuré l'application.
– Adapter pour les visite médicales, aux patients et aux conditions de la consultation.
– Transmission sécurisé des données.
– Des capacités de reconnaissance vocale.

3.3 Le système d'exploitation Android™

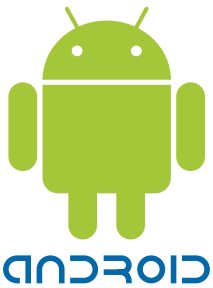

FIGURE 3.2 – Logo et sigle d'Android™

Android™ est un système d'exploitation basé sur *Linux* conçu pour les équipements mobile avec d'un écran tactile comme les *smartphones* et les tablettes. Développer à l'origine par *Android™, Inc.* que *Google* a supporté financièrement et plus-tard acquérir en 2005. Android™ a été dévoilé en 2007 parallèlement a la fondation de l'*Open Handset Alliance* : un consortium composé de sociétés dévoué a l'avancement des standards ouverts pour les équipements mobile. Le première téléphone sous Android™ est vendu en Octobre 2008.

La dernière version stable d'Android™ en date (Mai 2013) est 4.2.2 *Jelly Bean* sortie le 11 Février 2013.

Android™ est basé sur le Kernel Linux et utilise pleinement ses capacités de support matériels exhaustif. Mais la comparaison avec les distribution Linux, embarqué ou même destiné aux bureaux, s'arrête a ce niveaux. [9]

FIGURE 3.3 – *Google* Nexus 7, un terminal Android™

3.3.1 Parts du marché

L'adoption du système d'exploitation Android™suit une courbe exponentiel depuis quelque temps et la tendance n'est pas prête de s'inverser, selon les dernière rapport du cabiné d'analyse *Strategy Analytics*, Android™a réussi à capturé environ 68.4% du marché globale [10].

Système d'exploitation	Volume de production 3Q2012[13]	Parts du Marché 3Q2012[1]	Volume de production 3Q2011[23]	Parts du Marché 3Q2011[2]	Différence
Android™	136.0	75.0%	71.0	57.5%	91.5%
iOS	26.9	14.9%	17.1	13.8%	57.3%
BlackBerry	7.7	4.3%	11.8	9.5%	-34.7%
Symbian	4.1	2.3%	18.1	14.6%	-77.3%
Windows Phone 7/ Windows Mobile	3.6	2.0%	1.5	1.2%	140.0%
Linux	2.8	1.5%	4.1	3.3%	-31.7%
Autres	0.0	0.0%	0.1	0.1%	-100.0%
Totales	181.1	100.0%	123.7	100.0%	46.4%

TABLE 3.1 – Les six major systèmes d'exploitation mobile en terme de Volume et du parts de marché en 3etrimestre 2012 [1]

	2008	2009	2010	2011	2012[4]
Unités Android™ produites	0.7	7.0	71.1	243.4	333.6
Parts de marché Android™	0.5%	4.0%	23.3%	49.2%	68.2%

TABLE 3.2 – Production et parts de marché entre 2008 et 2012 [1]

3.3.2 Versions Android™ en circulation

Le tableaux 3.3 représente les différentes versions d'Android™ et leurs taux d'utilisation respective. On remarque que la plupart des terminaux mobiles Android™ sont sous la version 2.3 *Gingerbread* sortie le 6 Décembre 2010, Ceci est dú aux fait que plusieurs téléphones bas de gamme équiper de cette version sont encore en production.

1. 3etrimestre 2012
2. 3etrimestre 2011
3. En million d'unité
4. Estimation

Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	2.2%
2.2	Froyo	8	8.1%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	45.4%
3.1	Honeycomb	12	0.3%
3.2		13	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.0%
4.1	Jelly Bean	16	12.2%
4.2		17	1.4%

TABLE 3.3 – Distribution des versions Android™ en circulation qui on accéder au *Google Play*[5]

3.3.3 Les raisons du succès d'Android™ [9]

Les raisons pour le succès Android™ peuvent être dénombrés comme suit :

Un *Framework* d'Application Riche. Android™ fourni un excellent Software Development Kit (SDK) avec des Application Programing Interface (API) stable dans le long-terme, ce qui assure au partenaires tiers un écosystème standardisé. Alors que le système en lui même est en constante évolution, la stabilité des API pour la plupart est préserve, ce qui permet dínvestir dans le long-terme sur la plate-forme. Concevoir et construire des application pour les distribués sur différent plate-formes permet des réductions drastique en terme des coûts et effort pour les entreprises.

Un Time-to-Market (TTM) Agressif. Concevoir des appareil avec Android™ peut réduire le TTM d'une manière significative. Il suffit de se procuré les sources, les adapter pour le matériel en question et le vendre. Et dans le cas ou les schémas et usages de référence sont appliqué, la sorti d'un nouveau produit est possible au cour de quelque mois. Seulement voilà, c'est pas aussi facile et une certaine expertise et connaissances dans Android™ sont requise. Et même si sortir un système basé sur Android™ peut être plus rapide comparé à d'autre solutions, le suivit des évolutions du système ainsi que maintenir le code à long terme est une autre histoire.

5. Données récolté pendant une période de teste de 14 jours arrêter le 4 Février 2013.

Concentrer sur « Ce qui compte réellement ». En fournissant un *Framework* pratique, Android™ permet aux développeur de se concentrer sur les aspects à valeur commercial. L'assemblage d'un appareil et une activité qui consomme énormément du temps et de ressources et ne pas avoir à réinventer un - encore - autre système d exploitation permet d'éviter un autre gaspillage de temps.

Open Source. Malgré qu'il n'est pas développer d'une manière communautaire, Android™ reste 100% modifiable et diffuse un sentiment de sécurité parmi les entreprises contre les menaces légales.

3.3.4 La pile logiciel d'Android™ [11]

D'une manière simple. La pile logiciel d'Android™ est un Kernel Linux et une collection de bibliothèques C/C++ exposé à travers un framework d'application qui fournit des services pour l'environnement d'exécution et les applications. On peut énumérer les éléments composant la pile logiciel comme suit :

Kernel Linux Services de base qui inclue les pilotes matériels, gestion des processus et de la mémoire, sécurité, réseaux et gestion d'autonomie. Fourni aussi une couche d'abstraction entre le matériel et le reste de la pile.

Bibliothèque Se situ au dessus du Kernel, Android™ inclue divers bibliothèques C/C++ de base comme *libc* et *SSL* ainsi q
- Une bibliothèque multimédia pour la lecture des fichier audio et video.
- Un *Surface manager* pour la gestion de l'affichage.
- Des bibliothèques graphiques qui inclue le *SGL* et *OpenGL* pour les graphiques 2D et 3D.
- Un support native de base de donnée à travert *SQLite*.
- *SSL* et *WebKit* pour le navigateur web intégrer et la sécurité internet.

Environnement d'Execution (runtime) Android™ L'environnement d'exécution et le facteur qui sépare un terminal Android™ d'une implémentation Linux mobile. En cohérence avec les bibliothèques de base et la machine virtuel *Dalvik*, l'environnement d'exécution Android™ est le moteur qui fait fonctionner les applications et, avec les bibliothèques, forme les bases du framework application.

Bibliothèque de Base Même si la plus part des applications Android™ sont écrits avec du langage *Java*, *Dalvik* n'est pas une machine virtuel java. Les bibliothèques Android™ de base fournit la plus part

des fonctionnalités qu'on retrouve dans les bibliothèques de base *Java*, en plus de quelque bibliothèques spécifiques à Android™.

La Machine Virtuel devDalvik *Dalvik* est une machine virtuel qui a était optimisé pour s'assurer que chaque terminal peut faire fonctionner plusieurs instance d'une maniéré efficace. Il s'appuie sur le Kernel Linux pour le threading et la gestion bas niveaux de la mémoire.

Le *Framework* Application Le *Framework* application fournit les classes utilisés pour crée les application Android™. Il fournit une abstraction générique pour l'accès matériel et gère l'UI et les ressources de l'application.

Couche Application Toutes les applications, quelle soit native ou produite par un tiers, est construites sur la couche applications via les même API. La couche application opère à l'intérieur de l'environnement d'exécution Android™, utilisant les classes et les services mise à disposition par le *framework* application.

3.3.5 Architecture des applications Android™ [11]

L'architecture d'Android™ encourage la réutilisation des composants, ce qui nous permet de publier et de partager des *Activities*, services, et données avec d'autre application. Avec une gestion d'accès gérer par les restriction de sécurité que nous définissons.

Le même mécanisme qui nous permet de produire un gestionnaire de contact alternative ou un compositeur de numéros nous permet aussi d'exposé les composons de notre application pour permettre à d'autre développeurs de les réutiliser en créant des nouveaux UI ou d'étendre des fonctionnalités.

Les services application suivants représente les bases architectural de tout application Android™, fournissant le *Framework* qu'on va utilisé pour notre application.

l'*Activity Manager* et le *Fragment Manager* Contrôle le cycle de vie de nos *Activities* et nos *Fragments* respectivement, y inclue la gestion de la pile des *Activities*.

Views Utilisé pour construire l'UI de notre *Activities* et *Fragments*.

Notification Manager Fournit un mecanisme consistent et non-intrusive de signalisation pour l'utilisateur.

Content Providers Permeté à notre application le partage des données.

Resource Manager Offre un moyen d'externalisé les ressources (comme par exemple les chaînes de charactéres et les images.)

Intents Présente un mécanisme pour transférer les données entre les applications et leurs composants.

Une des fonctionalité les plus intéressante pour l'aboutissement de notre projet offerte par Android™ sont ces capacité de localisation, étudier dans la parti suivante.

3.3.6 Location Based Services

Concept

Pour positionner un terminal, en spécifie ces coordonnées géographique en utilisant le géo-codage.

Géo-codage [12] Le géo-codage est le processus de retrouvé les coordonnée géographiques associé (exprimé souvent en terme de *latitude* et *longitude*) d'après d'autre données géographique comme l'adresse de la rue, code postale. Ces coordonnées géographique peuvent être inséré dans un système d'information géographiques ou intégré dans des médias comme les photos numériques par le biais de géo-marquage. Cette opération est communément appelé le *Forward Geocoding*.

Le *Reverse Geocoding* est la procédure inverse : retrouvé les lieux textuel comme l'adresse de la rue d'après les coordonnés géographiques. Car même si l'usage des paramètres comme la longitude et la l'attitude fourni un moyen pratique pour localisé l'individu d'une maniéré relativement précise. Les utilisateurs penche à pensés en terme de rues et adresses.

A fin de déterminer la position du terminal, plusieurs technologie de localisation sont à notre disposition.

Localisation par GSM On peut retrouvé la position de terminale mobile par le biais de sa cellule Global System for Mobile Communications (GSM). Cette technique fait intervenir divers moyens de triangulation des signales parvenant depuis les cellules qui dissérve un téléphone mobile. La position géographique du terminal est déterminé par une multitude de méthodes comme la Time Difference of Arrival (TDoA) ou l'Enhanced Observed Time Difference (E-OTD).

Localisation parGPS [13] Global Positioning System (GPS) est un système de navigation par satellites qui fourni la localisation et le temps dans toute condition météorologique et partout sur terre s'il existe un accès non bloquant à 4 ou plus satellites GPS. Ce Système fourni des services essentiel

dans le domaine militaire, civile et commercial partout dans le monde. Il est maintenu par les États Unis d'Amérique et accessible à quiconque possédant un récepteur GPS.

Point de vu Android™ [14]

L'accès aux Location Based Services (LBS) se fait essentiellement via deux objets :

Location Manager Permet d'exploiter les services basés sur la localisation.

Location Providers Chaque *providers* représente une technologie de localisation utilisé afin de déterminer la localisation actuel du terminale.

On utilise ces deux Classes pour les fins suivantes :

- Obtenir la position actuel.
- Suivre les mouvement.
- Alerte de proximité dans le cas ou l'on approche ou s'éloigne d'une zone spécifique.
- Retrouvé les fournisseurs de localisation disponible.
- Observé le status du récepteur GPS.

Généralement deux techniques de détection de localisation sont disponible dans le terminal : détection par le réseau *Network Provider* et la détection par GPS *GPS Provider*. Le choix de la technologie a utilisé est soit explicite ou automatique suivant des critères prédéfinie par le développeur de l'application. Avant de pouvoir exploité un service de localisation, un niveau de précision doit figuré dans le manifeste de l'application via les *uses-permission tags*.

Listing 3.1– permission pour la localisation par le réseau.

```
<uses-permission android:name="android.permission.
    ACCESS_COARSE_LOCATION"/>
```

Listing 3.2– permission pour la localisation par GPS.

```
<uses-permission android:name="android.permission.
    ACCESS_FINE_LOCATION"/>
```

A noté qu'une application ayant la permission *FINE* possède implicitement la permission *COARSE*.

3.4 Conclusion

Dans ce chapitre, on s'est permit d'inspecté les solutions similaires à la notre dans le but de s'éclaircir les idées sur les problèmes qu'on pourrait rencontré et pour mieux cerné les difficultés que nous allons rencontré. Vient en suite la présentation de la plate-forme ciblé en plus d'une application type, des connaissance critique pour le chapitre suivant qui porte sur le travail effectué.

Chapitre 4

Travail Accompli

4.1 Introduction du chapitre

Dans ce chapitre on procéde à la presentation des cas utilisateur de notre systéme ainsi que l'identification des acteurs impliqué dans ces cas. Puis on décortique l'implementation que nous proposant en citant eventuellement nos motifs et intentions.

4.2 Vue Général

Identification des acteurs

Notre systéme interagie essentiellement avec trois acteurs différents :

Le medecin C'est l'acteur principale de notre systéme.

Le service web Source des données à achiminé vers le medecin (Tache et)

Systéme d'exploitation Communique à notre systéme les information receuille des divers composants qui nous interresse (localisation GPS/-Network, etat de la connectivité, etat de la battery).

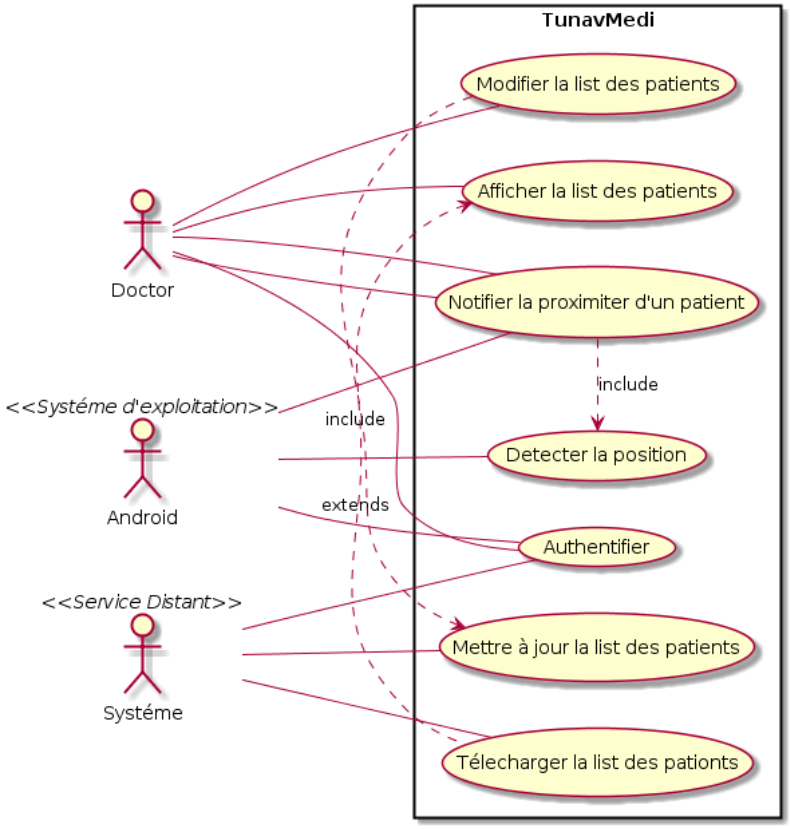

FIGURE 4.1 – Diagramme UML des cas d'utilisation.

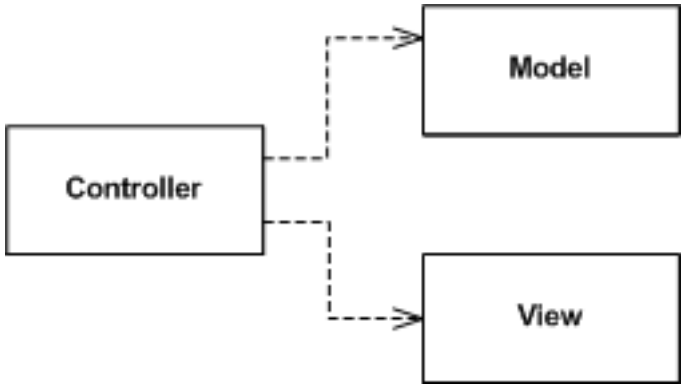

FIGURE 4.2 – Diagramme UML du patron *Passive Viev* [**?**]

4.2.1 Cas d'utilisations

Cas : Authentifier

Cas : Notifier la proximiter d'un patient

Cas : Detecter la position du terminal

Cas : Afficher la list des patients

Cas : Télecharger la list des patients

Cas : Modifier la list des patients

Cas : Mettre à jour la list des patients

4.2.2 Environnement de développement

4.3 Architecture Général

L'architecture globale de l'application est calqué sur Le patron "Vue Passive" (Passive View). Le patron *Passive View* (fig 4.2) est une variation des patrons model–view–controller (MVC) et model–view–presenter (MVP), à ce qui ce passe dans ces patrons l'interface utilisateur est divisé entre une vu qui s'occupe de l'affichage des données et un controlleur qui repont aux interactions de l'utilisateur. La différance majeur avec le *Passive View* est que la vue est completemement passive et n'ai pas responsable de sa mise à jour depuis le model. Dans ce cas tout la logique de la vue est dans le controlleur et aucune dependances ni dans un sens au dans un autre entre le vue et le model [**?**].

Ce patron est idéal dans notre cas pour deux raisons majeurs :

- Dans notre projet le enview n'est pas une partie très important dans la mesure ou l'objectif est d'intégré un système éventuellement préconçu, donc avec une autre logique de présentation. Déporter les interactions avec le modèle dans le contrôleur permet d'intégrer d'autre implémentation d'affichage plus facilement.
- La nature même de cette procédure d'accés - a savoir l'aspect abstrait, donc plus fragile - nous pousse à réduire les composants en relations pour réduire la marge d'erreur possible et facilité les tests.

Dans la suite de ce chapitre, on procède à l'explication détaillés de chaque composant de cette architecture.

4.4 Le Modèle

Un des objectifs de ce projet étant de fournir une solution d'accès au donnée flexible à fin de couvrir les besoins de chaque client de manière individuel. On a opter donc pour un modèle basé sur l'implémentation de deux interfaces (figure 4.3) :

- Interface d'authentification.
- Interface d'accès à la liste des patients.

L'idée est simple : pour chaque client, une implémentation spécifique à son infrastructure sera développez soit par son propre effectif, soit par une des équipes de Tunav, ou dans le cas idéale par une alliance formé par des agents des deux camps qui garantie une collaboration plus poussée pour des résultats meilleurs. Ces ensemble d'interfaces nous permet de construire notre application

4.4.1 Implémentation de test

Une implémentation de la couche d'accès abstraite est réalisé dans le cadre de ce projet pour pouvoir testé la solution. Cette implémentation est de caractère locale à l'application à travers les API de la base de données *SQLite* qui fait parti de l'SDK Android™. En fait une implémentation locale nous affranchies des problèmes qui peuvent se produire dont la corrélation avec l'application est faible. Cette même idée a influencé la mise en place même de cette implémentation qui à su rester la plus simple possible en restant très proches des objets de base de notre application.

Se trouvant dans le package *com.tunav.tunavmedi.dal.sqlite*, cette implémentation peut être subdivisé en trois éléments :

Les Contrats représente les contrats relative aux tables dans notre implémentation de test. chaque contrat implémente l'interface *android.provider.BaseColumns*

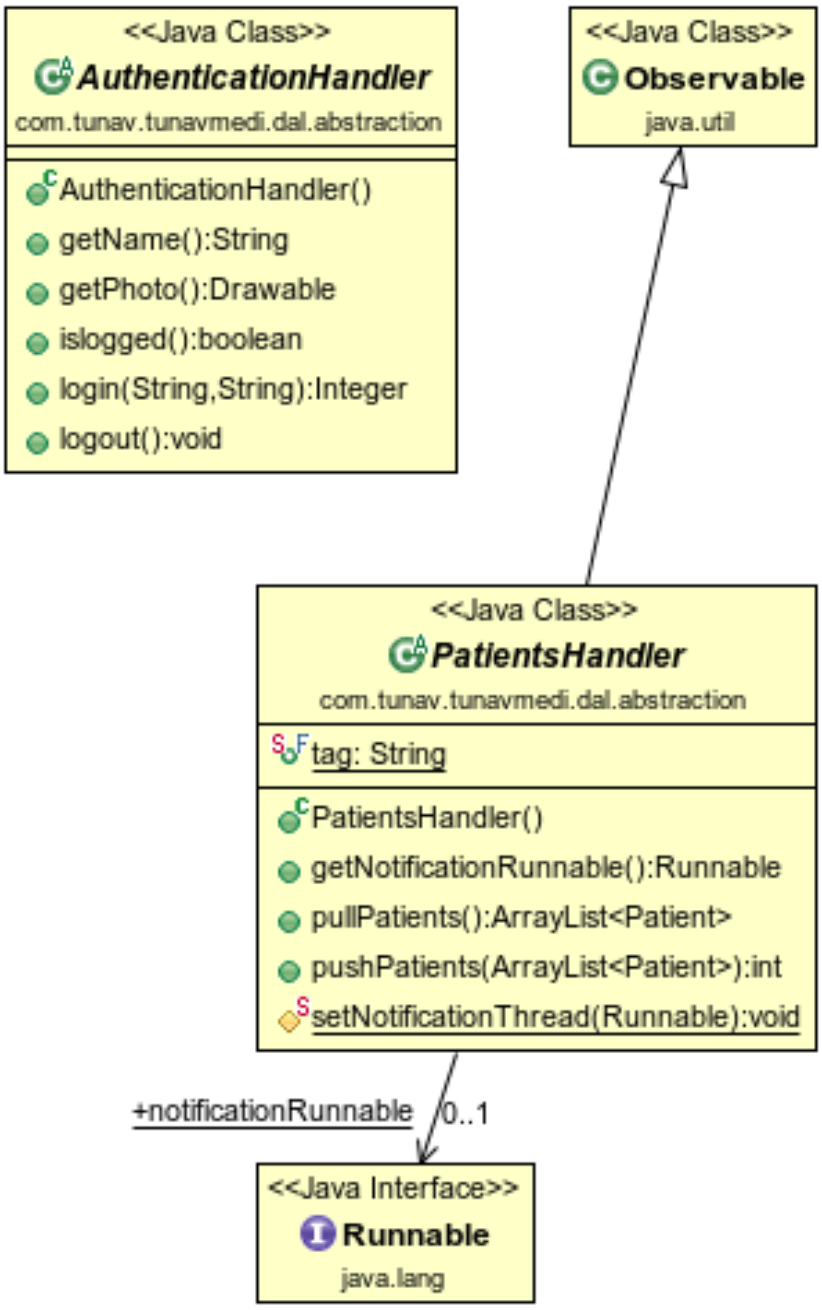

FIGURE 4.3 – Diagramme de classe de la couche d'accées.

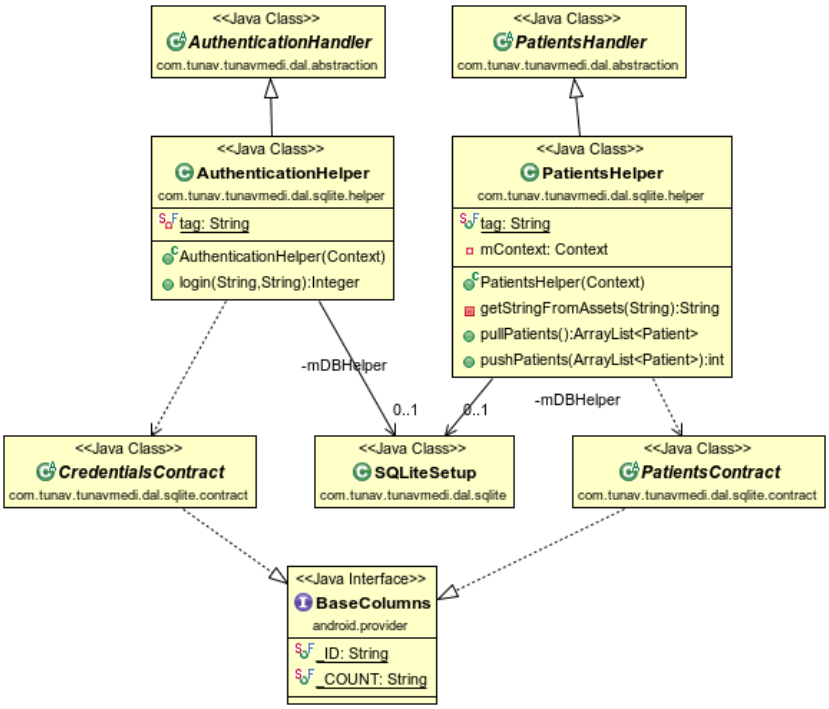

FIGURE 4.4 – Diagramme de classe de l'implémentation de la couche d'accées de test à base de SQLite.

et contient - entre autre - les commandes SQL de création et de suppression de la dite table, des éventuel index, et les commande d'insertion des données de test.

Les *Helpers* ce sont les implémentations des classes abstraites qui définisse la couche d'accès et présente les procédure d'extraction des données préinséré dans nos table fictives en faisant appel à la classe *SQLiteSetup* .

la classe *SQLiteSetup* Elle hérite de la classe *SQLiteOpenHelper* et destiner à contrôler la création et l'accès à notre base de données de teste.

4.4.2 Interface d'authentification

4.4.3 Interface des Données

Mécanisme de notification

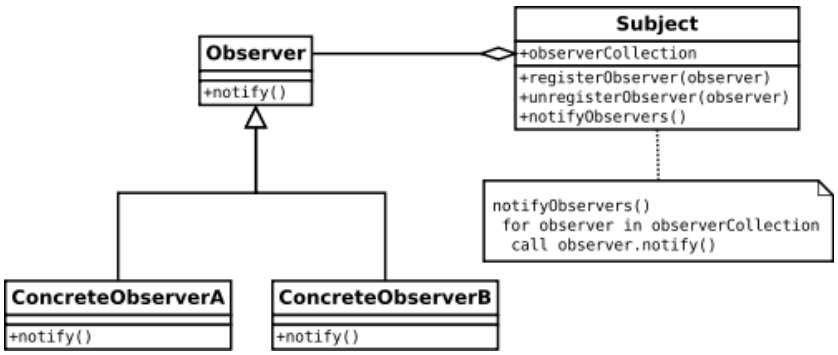

FIGURE 4.5 – diagramme UML du patron de conception Observateur [2]

Le patron **Observateur** (*observer pattern*) (fig 4.5) et un patron de conception couramment utilisé et qui nous perment d'avoir une relation 1/N entre divers objets. Le patron observateur assume que l'objet qui contient les données est séparé des l'objets qui les affiche et ces dites objets *observe* le changement de ces données [15]. Quant on implémente le patron observateur, on réfère communément à l'objet contenant les données par "Sujet" ; et chacun des consommateurs des données par "Observateur", Et chaque Observateurs implémente une interface préconnu que le Sujet invoque quant les données changes [15]. Dans le langage Java, ce patron est réaliser à travers la class *java.util.Observable* et l'interface *Java.util.Observer*. Le Sujet hérite de la classe *Observable* et les changements sont signalé par les méthodes *setChanged()* et *notifyObservers()* ou *notifyObservers(Object message)*.

4.5 Le Contrôleur

4.6 La Vue

Le système d'exploitation Android™rend facile le développement des application qui tourne sur des appareils qui possèdes des forme et des taille d'écran différents, une des amélioration

4.7 Testes

4.7.1 Pourquoi tester ? [16]

- La raison la plus évidente pour écrire les testes - étant la plus populaires aussi - est que c'est un moyen efficace pour savoir si le bout de code ajouté dans le projet marche correctement ou pas, ce qui non seulement fournit une certaine confiance dans la robustesse du logiciel mais présente un effet secondaire bénéfique en réduisant le temps nécessaire pour le débogage à la recherche d'un bug caché.
- Une autre raison pour écrire les testes est que c'est un autre forme de documentation qui aide les autres développeurs comprendre le code contenu dans le projet.
- Une raison non évidente pour écrire des teste est le fait que les tests améliore la manière dont le code est conçu en mettant en évidence les difficultés pour le maintenir.

Modifié la localisation dans l'émulateur

4.7.2 Quelque difficultées rencontrées

4.8 Conclusion du chapitre

Chapitre 5

Conclusion Général

L'intégration des technologies au sein des établissements médicaux est , malgré les divers obstacles, une tendance établie et représente un marché juteux pour les sociétés désirant le conquérir. Justifiant la judicieuse idée derrière ce projet.

Reste que l'application en elle même reste limité. En particulier, le processus de déploiement suggère un minimum d'infrastructures requise, donc pour offrir l'expérience désirer une solution alternative de support développer par Tunav est de rigueur pour soit comblé le manque dans les équipements de l' établissement client ou dans le cas extrêmes les supplanté. Une stratégie de commercialisation et un besoin évidant.

Ce projet peut être qualifié de type *proof of concept*, qui vise à explorer une idée et vérifier son applicabilité. Une aubaine pour l'application produite qui, en tout honnêteté, n'est pas encore au point et souffre de plusieurs lacunes de conceptions et d'implémentation. Si un produit sérieux dans le même thème est à offrir par Tunav, des efforts de recherche et de développement sont de mise. En particulier l'intégration de médecins pratiquants dans des hôpitaux au processus de conception et de test serai critique pour la compétitivité du produit.

Cependant, les problèmes techniques pour le développement de cette application ne sont pas les seuls à freiner sont adoption. Outre le problème de coûts et l'effort de persuasion requit, c'est un problème d'ordre psychologique qu'il faut y faire face. En effet, avec tout concepts qui change radicalement des procédures bien établie, un réticence de la par des utilisateurs ciblés , en occurrence les médecins et le staff médical dans un contexte plus large, risque de saboté les tests díntégrations. Des compagne de sensibilisation sont à prévoir.

Annexe A

Comment le projet à été realisé

A.1 Structure du project

A.2 Usage du géstionnaire de version Git

A.2.1 Git

Bibliographie

[1] IDC Worldwide Mobile Phone Tracker.

[2] Wikipedia. Observateur (patron de conception), 2013. [accessed May-2013].

[3] Elsa Bembaron. Le marché des pc s'effondre face aux smartphones et aux tablettes.

[4] fiche société.

[5] Brian T. Horowitz. Cisco cius android tablets go to work on san diego hospital private cloud.

[6] David Raths. Real-time healthcare : How one hospital uses cisco's cius to improve patient care.

[7] Palomar Pomerado Health (Press Release). Palomar pomerado health unveils wireless healthcare application for mobile devices.

[8] Cerner. The patient visit...revisited. flayer.

[9] Benjamin Zores. The growth of android in embedded systems. Technical report, The Linux Foundation, 2013.

[10] John Koetsier.

[11] Reto Meier. *Professional Android 4 Application Development*, chapter 1. Wrox, May 2012.

[12] Wikipedia. Geocoding - Wikipedia, the free encyclopedia, 2013. [accessed Feb-2013].

[13] Emna Hajlawi. Cours de communication spatial, 2012.

[14] Reto Meier. *Professional Android 4 Application Development*, chapter 13. Wrox, May 2012.

[15] James W. Cooper. *Java Design Patterns : A Tutorial.*

[16] Carl Meyer. Getting started with automated testing, March 16, 2013. [video talk].