



Haute École de Namur-Liège-Luxembourg  
Implantation IESN

# DÉVELOPPEMENT D'UNE APPLICATION ANDROID D'ASSISTANCE POUR AGENTS DE TERRAIN

par

**Andrew KEYMOLEN**

Namur

Année académique 2015-2016

*Promoteur*

Chantal BERTRAND

*Maître de stage*

Thomas NYSSEN

Travail de fin d'étude présenté en vue de l'obtention du  
diplôme de BACHELIER EN INFORMATIQUE DE GESTION





Haute École de Namur–Liège–Luxembourg  
Implantation IESN

# DÉVELOPPEMENT D'UNE APPLICATION ANDROID D'ASSISTANCE POUR AGENTS DE TERRAIN

par

**Andrew KEYMOLEN**

Namur

Année académique 2015-2016

*Promoteur*

Chantal BERTRAND

*Maître de stage*

Thomas NYSSEN

Travail de fin d'étude présenté en vue de l'obtention du  
diplôme de BACHELIER EN INFORMATIQUE DE GESTION

La rédaction de ce rapport n'aurait pas été possible sans l'intervention d'un certain nombre de personnes, c'est pourquoi je tiens ici à les remercier comme il se doit. J'adresse donc en premier mes remerciements à Vincent MERICHE, développeur chez Djm digital ô combien patient, qui a su me former à Android et me partager sa passion pour le développement sans jamais manquer de répondre à mes bien trop nombreuses questions. Mes remerciements se tournent également vers tous les autres membres du personnel de Djm digital qui ont su m'accueillir au sein de leur équipe durant ces quatre mois des plus riches en expériences. Parmi eux, Dominique MAES pour m'avoir permis d'effectuer ce stage, Aurélie LESAGE pour l'avoir organisé et Thomas NYSSEN pour avoir accepté d'endosser le rôle de mon maître de stage. Je tiens également à remercier Chantal BERTRAND, ma promotrice et professeure, pour son suivi et son soutien tout au long de ces quatorze semaines. Enfin, je tiens à remercier toutes les personnes qui m'ont aidé et soutenu de près ou de loin dans le cadre de mes études, bien trop nombreuses que pour ici les énumérer.

## TABLE DES MATIÈRES

1	Introduction.....	4
2	Présentation de Djm digital .....	5
2.1	Généralités.....	5
2.2	Situation géographique .....	5
2.3	Organisation interne.....	6
3	Présentation du stage.....	7
4	Méthodes, outils et technologie .....	8
4.1	Android Studio.....	8
4.2	Gradle .....	11
4.3	Outils divers utilisés en complément.....	13
4.4	Matériel utilisé lors des tests.....	15
5	Android.....	17
5.1	Historique .....	18
5.2	Les différentes versions d'Android et leurs APIs .....	19
5.3	Google Play.....	22
5.4	Le Software Development Kit d'Android.....	23
5.5	Architecture.....	24
5.6	Design adaptatif et le material design.....	25
5.7	Développer pour Android.....	27
5.7.1	Java.....	27
5.7.2	La classe Application.....	28
5.7.3	Context .....	28
5.7.4	Activity .....	28
5.7.4.1	Le cycle de vie d'une activity .....	29
5.7.5	Fragment.....	31
5.7.6	Service .....	32
5.7.7	Intent.....	32
5.7.8	Gestures .....	33

5.7.9	Adapter .....	35
5.7.10	Les bibliothèques de compatibilité .....	35
5.7.11	MVVM et data binding.....	36
5.7.11.1	Présentation du MVVM.....	37
5.7.11.2	Présentation du data binding.....	39
5.8	Création d'un projet fictif.....	42
5.8.1	Description de l'application fictive .....	42
5.8.2	Structure.....	43
5.8.3	AndroidManifest.xml.....	44
5.8.4	Les ressources .....	46
5.8.4.1	strings.xml.....	46
5.8.4.2	dimens.xml.....	47
5.8.4.3	styles.xml .....	47
5.8.4.4	colors.xml .....	48
5.8.4.5	Les layouts.....	48
5.8.5	R.java.....	52
5.8.6	Les fichiers sources .....	52
6	Présentation du projet .....	56
6.1	Analyse.....	57
6.1.1	Diagramme d'activité.....	57
6.1.2	Diagramme d'état.....	58
6.1.3	Diagramme des cas d'utilisation.....	58
6.1.3.1	Cas 1: Créer le rapport .....	58
6.1.3.2	Cas 2 : Modifier le rapport .....	59
6.1.3.3	Cas 3 : Poster le rapport.....	59
6.1.3.4	Cas 4 : Visualiser le rapport .....	59
6.1.3.5	Cas 5 : Finaliser le rapport .....	59
6.1.4	Diagramme des classes persistantes.....	60
6.1.5	Diagramme de navigation.....	61

6.1.6	Besoins non fonctionnels .....	61
6.2	Visite guidée d’Ethias Prevention.....	62
6.2.1	Écran de connexion.....	62
6.2.2	Écran d'accueil.....	63
6.2.3	Menu.....	65
6.2.4	Écrans de création de rapports.....	67
6.2.4.1	Première étape, les données générales .....	68
6.2.4.2	Deuxième étape, les observations et les recommandations .....	71
6.2.4.3	Troisième étape, les sections .....	72
6.2.5	Historique des rapports .....	73
6.3	Bibliothèques utilisées .....	74
6.3.1	Retrofit pour les services web.....	74
6.3.2	Realm pour la persistance des données .....	77
7	Critiques et suggestions.....	79
8	Conclusion.....	82
9	Bibliographie.....	83
10	Table des figures.....	87
11	Annexes .....	89
11.1	Un rapport en PDF .....	89

## 1 INTRODUCTION

En guise d'activité professionnelle d'intégration, il est demandé aux étudiants en dernière année d'informatique de gestion de l'Hénallux de réaliser un stage de quatorze semaines au sein d'une entreprise. Celui-ci a pour buts la préparation au domaine entrepreneurial et l'acquis d'une première expérience dans le monde professionnel mais il doit également aboutir à la rédaction d'un travail de fin d'étude, c'est-à-dire un rapport revenant en détails sur le déroulement du stage et sur les différents projets qui y ont été réalisés.

Ainsi, il m'a été possible de consulter plusieurs offres d'activité professionnelle avant de postuler pour celles qui m'intéressaient le plus. Après plusieurs entretiens fructueux, mon choix s'est finalement porté sur un stage dans un secteur extrêmement en vogue actuellement et particulièrement en accord avec mes ambitions. En effet, Djm digital proposait d'accueillir un étudiant afin de lui offrir la possibilité de s'exercer au développement sur mobiles et c'est vers eux que je choisis alors de me tourner.

Le stage a donc débuté en février et ma mission consistait en la réalisation d'une application pour le compte d'Ethias, un programme d'assistance pour agents de terrain nommé Ethias Prevention. J'eus la chance d'être l'unique développeur à travailler dessus et bien que j'étais tout de même grandement aidé, cela me permit d'en saisir l'entièreté des fonctionnalités. Celles-ci tournent autour de la création et de la centralisation de rapports de constatation de risques. Ainsi, un employé de chez Ethias en mission peut noter toutes ses observations sur l'application et retrouver le résultat plus tard sur un autre système. L'objet principal de ce travail de fin d'étude est donc cette application que je tâche plus loin d'analyser et de présenter en détail.

Après une brève présentation de l'entreprise donc, un point sera consacré à une description plus en profondeur du stage. Après quoi j'énumèrerai les différents outils et technologies utilisés tout au long de ces quatorze semaines dont, entre autres, l'indispensable Android Studio.

La suite du travail se consacrera à une étude approfondie de la plateforme à laquelle est destinée Ethias Prevention, à savoir Android. J'y expliquerai un historique de cet OS ainsi que son fonctionnement, les paradigmes qui lui sont associés et les détails d'un développement sur celui-ci en passant par une explication du Java pour Android ou encore du Model View ViewModel, le design pattern utilisé.

Enfin, et avant de conclure sur l'expérience vécue lors de ce stage en entreprise, suivront l'analyse et la présentation du projet ainsi qu'une réflexion sur les apports possibles concernant le développement d'une telle application.

## 2 PRÉSENTATION DE DJM DIGITAL

Le stage d'intégration professionnel et le développement du projet se sont déroulés comme prévu dans une entreprise à raison de trente-huit heures par semaine pendant quatorze semaines, de février à mai. Voici la présentation de cette entreprise, Djm digital.

### 2.1 GÉNÉRALITÉS



Figure 1 : logo de Djm digital

78 Porte de Lorette  
4600 Visé, Belgique

+32 (0)4 379 69 97

[info@djmweb.be](mailto:info@djmweb.be)

<http://www.djmdigital.be/>

Société à responsabilité limitée liégeoise implantée à Visé, Djm digital fut fondée en 1999 par Dominique MAES et se développe depuis pas à pas. Aujourd'hui constituée de seize personnes issues de disciplines telles que le web, le mobile, le développement et le graphisme, Djm digital a pour vocation de conseiller ses clients, en amont comme en aval de leur projet, afin d'intégrer le digital à leur stratégie globale d'entreprise.<sup>1</sup>

### 2.2 SITUATION GÉOGRAPHIQUE

Les locaux sont situés à Visé, capitale de la Basse-Meuse et centre touristique reconnu, à mi-chemin entre la ville de Liège et de Maastricht.

Ils sont facilement accessibles en voiture depuis l'autoroute en prenant la sortie vers Visé ou en transports en commun, que ce soit en train ou en bus depuis Liège ou Maastricht, Visé étant munie d'une gare.

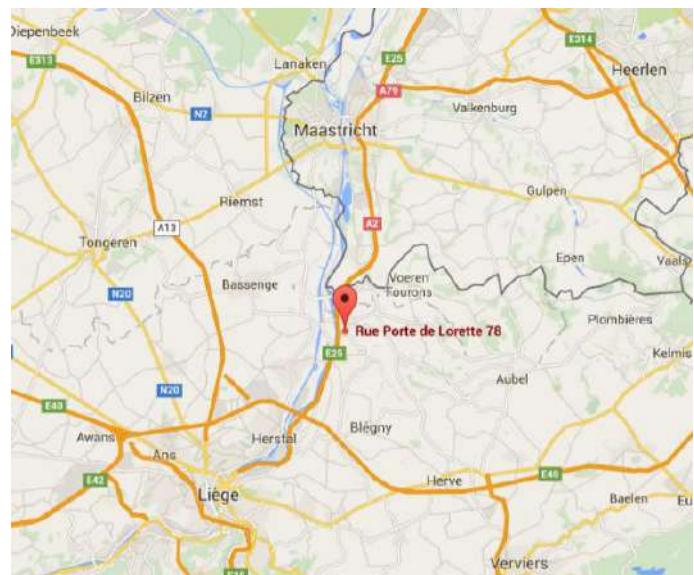


Figure 2 : localisation des locaux  
(récupéré sur : <https://www.google.be/maps/>)

<sup>1</sup> (Djm Digital : Création de site internet & application mobile, s.d.)

## 2.3 ORGANISATION INTERNE

Les types d'activité principalement pratiqués ici sont le développement mobile, le développement web et l'e-marketing ainsi que la consultation dans chacun de ces domaines. Ces différentes disciplines interagissant de façon organique, elles s'organisent autour des quatre pôles que sont le département design/graphisme, celui de développement, de gestion de projets et enfin, le secrétariat de direction.

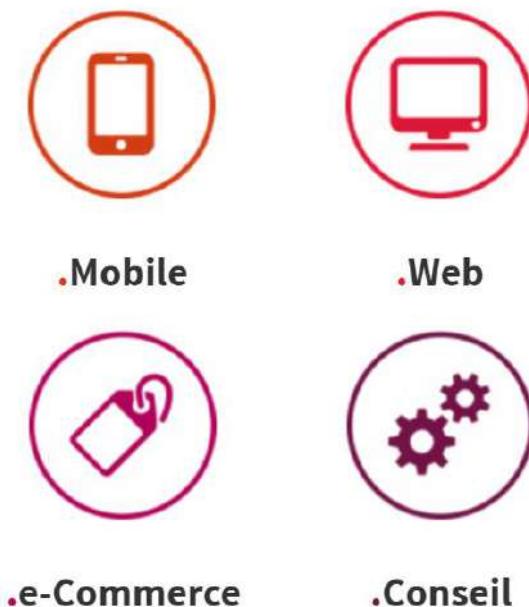


Figure 3 : les quatre domaines d'expertise de Djm digital

### 3 PRÉSENTATION DU STAGE

Débuté le lundi 8 février 2016, ce stage a pour objectif le développement d'applications mobiles pour Android. Si le premier jour est dédié à l'introduction aux différents outils et technologies utilisés chez Djm digital, les deux premières semaines sont quant à elles consacrées à une phase d'apprentissage sur une application dont la nature fut décidée arbitrairement par Vincent MERCIE, un des développeurs Android chez Djm digital.

La mission d'introduction consistait donc au développement d'une application de gestion de profil autour du service T411, un site web dédié au partage de torrents qui sont des fichiers contenant les métadonnées d'autres fichiers et dossiers rendu disponibles au partage par différents utilisateurs.<sup>2</sup>

Un tel exercice servit de parfaite formation au développement sur Android et en couvrit la plupart des aspects de base. Les treize semaines suivantes allaient être consacrées au corps du stage, une application commandée par Ethias destinée à ses agents de terrains. Une fois chez le client, celle-ci doit en effet les assister dans la rédaction de leurs rapports. Constituée principalement de formulaires à remplir, cette application permet également de joindre des photos, des vidéos ou des messages audio ainsi que le lieu d'intervention, des législations, des pictogrammes et tout ce dont l'agent aurait besoin pour décrire la situation à laquelle il assiste. Ainsi, ces rapports sont centralisés et accessibles au besoin depuis l'application par le biais d'un historique.

Les fonctionnalités de l'outil ayant été précisément décidées au préalable – une version iOS de celui-ci étant déjà disponible ainsi qu'une première tentative d'implémentation de certains designs, le stage se consacre surtout à la compléction et la mise à niveau de celui-ci en prenant en compte les nouvelles technologies Android, à savoir la librairie Data Binding et le design pattern MVVM.<sup>3</sup>

Cette tâche, et donc le sujet de ce stage, est réalisée au sein des locaux de Djm digital dans une ambiance d'entreprise, depuis un PC Windows Seven, principalement sur Android Studio<sup>4</sup> et à raison d'environ huit heures par jour.

---

<sup>2</sup> (Torrent file - Wikiwand, s.d.)

<sup>3</sup> cf. 5.7.11 MVVM et data binding

<sup>4</sup> cf. 4.1 Android Studio

## 4 MÉTHODES, OUTILS ET TECHNOLOGIE

Qu'il soit question d'organisation, de support au développement, d'aide à la centralisation ou autres, les outils qui suivent ont tous été nécessaires dans le cadre du stage et du projet associé.

### 4.1 ANDROID STUDIO



<http://developer.android.com/sdk/index.html>

Figure 4 : logo d'Android Studio

Android Studio est un IDE – Integrated Development Environment – ou environnement de développement intégré, dont l'objectif est de fournir un outil de travail facilitant la création et le développement d'applications Android. Basé sur IntelliJ IDEA, un IDE Java<sup>5</sup> commercial très complet développé par JetBrains, il est proposé par Google en une version « Early Access Preview » en mai 2013. Ceci afin de remplacer à terme une distribution spécifique d'Eclipse, un autre IDE Java, munie du SDK – Software Development Kit – ou kit de développement logiciel, d'Android. Eclipse est en effet la solution officielle jusque-là proposée par Google aux développeurs voulant s'exercer sur son système d'exploitation. C'est en décembre 2014 qu'Android Studio passe en 1.0 et que Google le conseille à toutes personnes voulant tenter l'aventure Android.<sup>6</sup>

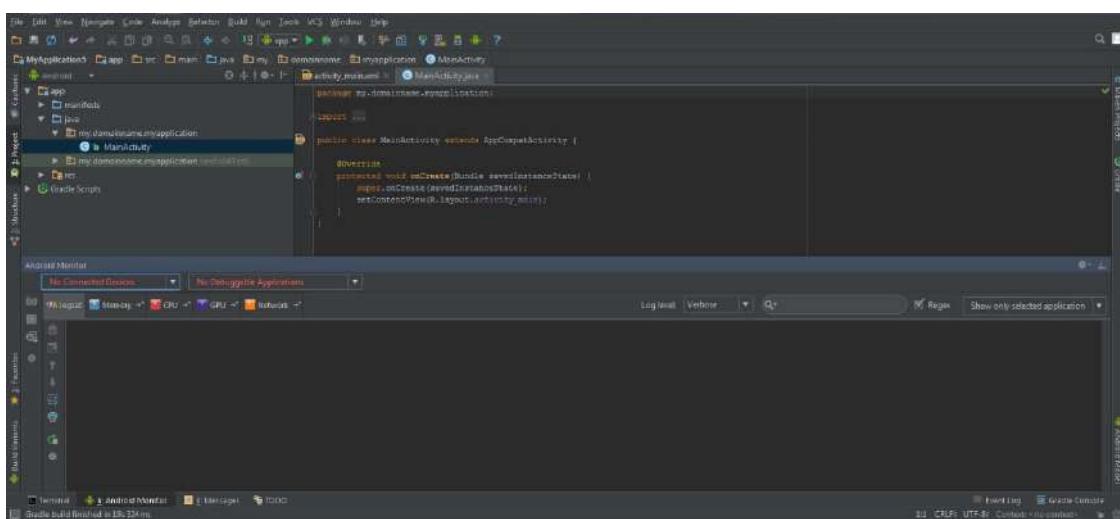


Figure 5 : interface utilisateur d'Android Studio

<sup>5</sup> cf. 5.7.1 Java

<sup>6</sup> (Download Android Studio and SDK Tools | Android Developers, s.d.)

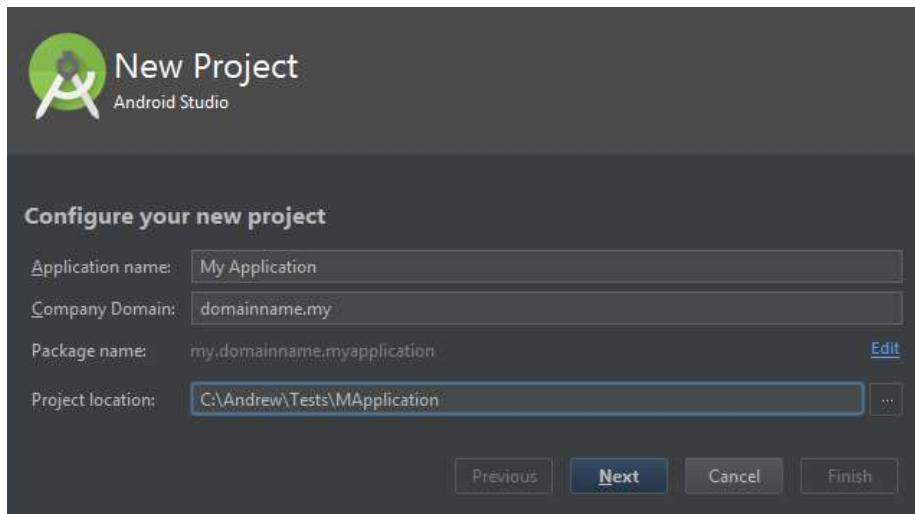


Figure 6 : écran « New Project »

Outre sa fonction d'édition de fichiers Java et des fichiers de configuration nécessaires aux applications Android, Android Studio propose également des outils pour gérer le développement

d'applications multilingues ainsi que pour visualiser directement l'apparence des pages créées sur des écrans de résolutions variées.

L'interface (cf. Figure 5 : interface utilisateur d'Android Studio), plutôt standard par rapport à d'autres IDE, permet de créer un projet de façon intuitive : aller dans « File » puis « New » et « New Project » fait apparaître un écran (cf. Figure 6 : écran « New Project ») permettant de décider du nom de l'application, du nom de la compagnie pour laquelle cette application est développée, du nom du package au sein duquel se trouve le projet et enfin, de la localisation du projet sur le disque dur.

L'écran suivant (cf. Figure 7 : écran « Target Android Devices ») permet, quant à lui, de décider du type de projet à réaliser.

Android étant utilisé sur différents supports, une décision s'impose ici quant à la compatibilité de l'application. Si la réponse la plus fréquente est le développement pour tablettes et smartphones, il est également possible de cibler les dispositifs tournant sous Android Wear (les montres connectées), Android TV (les télévisions connectées), Android Auto (les tableaux de bord des voitures compatibles) ainsi que Glass, le

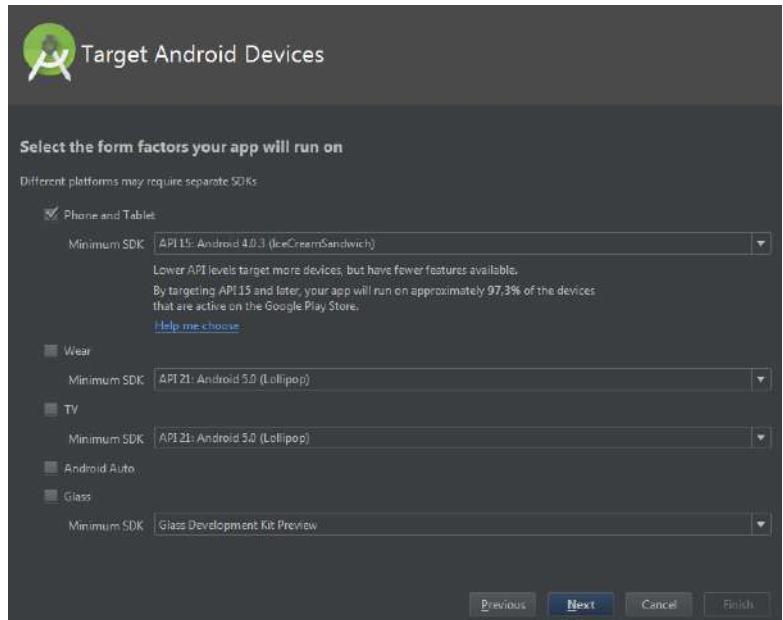


Figure 7 : écran « Target Android Devices »

système d'exploitation des Google Glass – des lunettes à réalité augmentée. Cet écran est important afin de choisir un ou plusieurs dispositifs cibles et donc son public, l'API correspondante donnant accès à plus ou moins de fonctionnalités et de rétrocompatibilité. La suite de cette présentation d'Android Studio s'occupe cependant du développement pour mobile.

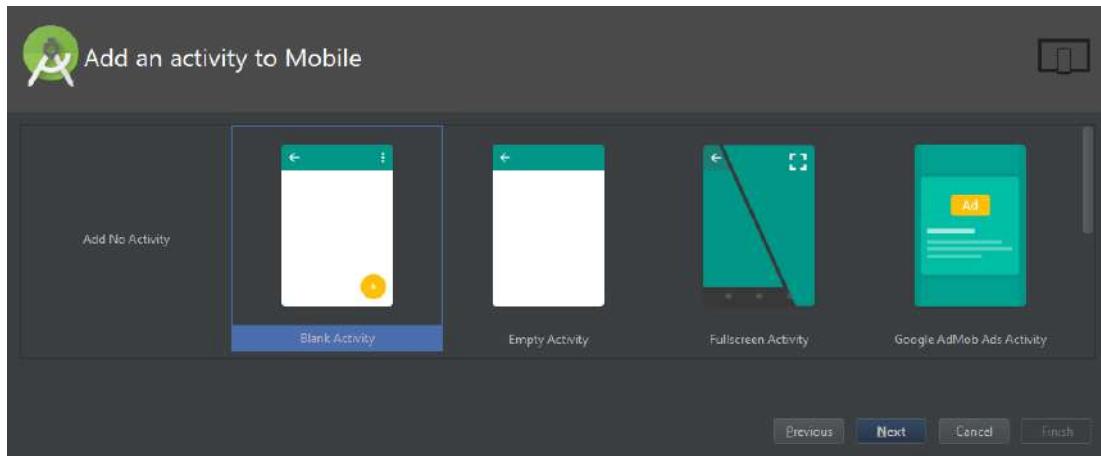


Figure 8 : écran « Add an activity to Mobile »

La prochaine étape se penche sur la nature (cf. Figure 8 : écran « Add an activity to Mobile ») et la configuration (cf. Figure 9 : écran « Customize the Activity ») de la première activity,<sup>7</sup> c'est-à-dire le premier écran qui est développé pour l'application. Un choix judicieux serait une activity vierge ou presque ou un écran de connexion par exemple. Mais bien d'autres options sont disponibles et le choix n'appartient qu'au développeur.

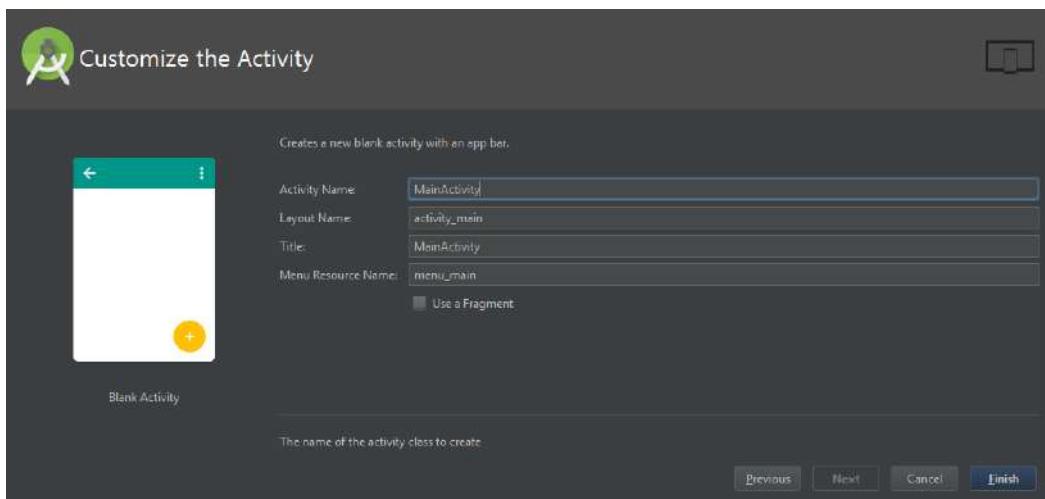


Figure 9 : écran « Customize the Activity »

Valider la première activity conduit donc à l'espace de travail (cf. Figure 5 : interface utilisateur d'Android Studio) découpé principalement en trois parties. En

<sup>7</sup> cf. 5.7.4 Activity

haut à gauche, la zone de navigation où se présente la structure du projet,<sup>8</sup> à sa droite la zone d'édition où est rédigé le code de l'application et en dessous, la console permettant l'affichage des messages d'erreur, un terminal Windows classique et surtout LogCat, une fenêtre au sein de laquelle il est possible d'afficher des messages customisés grâce à la classe « `android.util.Log` ». Cette fonction extrêmement prisée des développeurs permet de déboguer le code très efficacement, LogCat permettant également d'appliquer des filtres sur ce qui est affiché afin de se concentrer sur un problème à la fois.<sup>9</sup>

Enfin, la barre d'outils située en haut de cet espace de travail donne accès à toutes les fonctionnalités indispensables comme la compilation, l'exécution ou encore le mode debug.

## 4.2 GRADLE



Figure 10 : logo de Gradle

Le système de build, ou construction – le procédé qui permet de convertir du code en un fichier exécutable – d'Android consiste en un plugin Android pour Gradle. Ce dernier est un module de production avancé qui permet de construire des projets en Java, Scala, Groovy et C++. Gradle gère les dépendances – à des librairies, par exemple – et aide à définir une logique de construction personnalisée. Il est possible d'utiliser le plugin Android de Gradle de façon indépendante à Android Studio, permettant la conversion en fichiers exécutables de projets par lignes de commande sur des machines où Android Studio ne serait pas installé, tel un serveur.<sup>10</sup>

Rédigés en Groovy, les fichiers de build que permet de réaliser Gradle rendent alors possible l'importation de tâches standards, permettant parfois de construire des programmes utilisant un ou plusieurs langages, parfois d'importer des librairies dans un fichier jar, parfois d'exécuter des tests unitaires, etc., le tout en restant simple et compact.<sup>11</sup>

Le fichier `build.gradle` permet donc une configuration aisée des aspects suivants de l'application lors de son développement :

---

<sup>8</sup> cf. 5.8.2 Structure

<sup>9</sup> (DUBISY, 2015)

<sup>10</sup> (Android Pugin for Gradle | Android Developers, 2016)

<sup>11</sup> (MAZELIER, 2011)

- **Les variantes de construction**

En effet, il est envisageable de générer différents .apk – les packages de fichiers compressés permettant de lancer l’application sur le mobile – depuis le même projet.

- **Les dépendances**

Il est ici question de la gestion des librairies. Plus besoin de télécharger de .jar, Gradle s’en charge.

- **Les entrées du manifeste**

Il est possible, grâce au fichier build.gradle, de faire varier les valeurs du manifeste<sup>12</sup> – le fichier servant à définir les propriétés de l’application – comme le nom du projet, par exemple.

- **La signature**

Gradle se charge de la signature du fichier .apk en vue de la publication.

- **ProGuard**

Ce dernier est un outil qui permet de réduire et d’optimiser le code. Il est facilement configurable dans le fichier build.gradle.

- **Les tests**

Gradle permet de créer un .apk spécifiquement dédié au test de l’application.

Bien sûr, lors de la création d’un projet, ces différents aspects sont déjà prévus par défaut et il ne faut les déterminer que si nécessaire. Ci-dessous un exemple de fichier build.gradle :

```
apply plugin: 'com.android.application'  
android {  
    compileSdkVersion 23
```

---

<sup>12</sup> cf. 5.8.3 AndroidManifest.xml

```

buildToolsVersion "23.0.3"

dataBinding {
    enabled = true
}

defaultConfig {
    applicationId "com.andrew.fictional_login"
    minSdkVersion 15
    targetSdkVersion 23
    versionCode 1
    versionName "1.0"
}
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
    }
}
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.3.0'
    compile 'com.android.support:design:23.3.0'
}

```

### 4.3 OUTILS DIVERS UTILISÉS EN COMPLÉMENT

Plutôt secondaires concernant le développement du projet principal, les outils brièvement présentés ici sont surtout propres au mode de fonctionnement des développeurs au sein de Djm digital. Leurs fonctions respectives, centrées sur la gestion de projet, la centralisation et la décentralisation du travail effectué, le test de ce travail, et autres, en font néanmoins des outils puissants et il paraît nécessaire de les énumérer.

- git



<https://git-scm.com/>

Figure 11 : logo de Git

Un logiciel de gestion de versions de projet décentralisé.

- **SourceTree**



<https://www.sourcetreeapp.com/>

*Figure 12 : logo de SourceTree*

Un client Git mais aussi Mercurial pour bureau à destination des développeurs permettant une prise en main facilitée de Git.

- **SoapUI**



<https://www.soapui.org/>

*Figure 13 : logo de SoapUI*

Une application open source facilitant le test de services web au sein d'une architecture orientée service.

- **Node.js**



<https://nodejs.org/>

*Figure 14 : logo de Node.js*

Une plateforme logicielle libre en JavaScript tournée vers les applications réseau et permettant par exemple de faire tourner un serveur web sans nécessiter un logiciel externe.

- **toggl**



<https://toggl.com/>

*Figure 15 : logo de toggl*

Un service facilitant la quantification du temps passé à développer un projet à des fins de facturation ou d'aide à l'estimation du temps nécessaire à la réalisation d'un nouveau projet.

- **Jira**



Figure 16 : logo de Jira

<https://fr.atlassian.com/software/jira>

Un système de suivi de bogues, de gestion des incidents et plus généralement de gestion de projets.

- **Bitbucket**



Figure 17 : logo de Bitbucket

<https://bitbucket.org/>

Un service web d'hébergement et de gestion de développement logiciel semblable à SourceTree, utilisant également Git et Mercurial.

#### 4.4 MATÉRIEL UTILISÉ LORS DES TESTS

Le projet étant destiné aux supports mobiles, il devient important de le tester sur plusieurs dispositifs dotés de différentes versions d'Android afin de s'assurer du comportement de l'application sur ceux-ci. Aussi à considérer, les multiples tailles d'écran, la couche logicielle ajoutée par le constructeur ou encore les performances de l'appareil. Voici la liste des différents appareils utilisés en phase de test accompagnés de leur description.

- **Le Sony Xperia M2**

- 4,8 pouces 960x540 px
- 1 GB
- 1,2 GHz
- Android 5.1.1



Figure 18 : Xperia M2

- **La Nexus 7**

- 7,02 pouces 1920x1200 px
- 2 GB
- 1,5 GHz
- Android 6.0.1



*Figure 19 : Nexus 7*

## 5 ANDROID



Figure 20 : Bugdroid, la mascotte représentant Android

S'il y a principalement trois systèmes d'exploitation qui se partagent le marché dans le monde du mobile, à savoir iOS, Windows Phone et Android, c'est bien le dernier des trois le seul à être open source et donc à permettre librement à la communauté des développeurs de s'attaquer à son code source afin d'en tirer le meilleur. Peut-être une des plus importantes raisons de son succès, la politique de Google – les développeurs en titre d'Android – vis-à-vis de sa plateforme a indéniablement transformé les

applications mobiles en ce qu'elles sont aujourd'hui et cette transformation n'en est pas encore à sa phase finale.

Basé sur le noyau Linux, Android peut être défini comme étant une pile de logiciels organisée en cinq couches :

- **Kernel**

Le kernel, proche du hardware, se charge de la gestion des pilotes.

- **Bibliothèques logicielles**

Les bibliothèques logicielles toutes prêtes, utiles aux différents softwares qui sont installés en aval, permettent de gérer plus facilement des outils nécessaires tels une base de données ou des rendus graphiques.

- **Environnement d'exécution**

L'environnement d'exécution permet d'exécuter des programmes en Java<sup>13</sup> – le langage utilisé pour développer des applications Android.

- **Framework**

Le framework, qui consiste en un kit aidant les développeurs à créer des applications.

---

<sup>13</sup> cf. 5.7.1 Java

### • Applications standards

Les applications standards comme le navigateur web, la fonction d'appel, etc.

Bien entendu, ça ne s'arrête pas là car il est aussi bon de noter qu'Android Studio ainsi que Google Play,<sup>14</sup> de par leur utilité indéniable, font partie intégrante de ce qui est reconnu comme les fonctionnalités d'Android.

Initialement destiné aux appareils photo, ce système peut désormais faire tourner des dispositifs en tout genre allant de la machine à laver à la dernière montre connectée sans oublier la voiture et la télévision.<sup>15</sup> En 2015, c'est l'OS de Google qui est en tête avec 80% de parts de marché dans les smartphones.<sup>16</sup> Il semble donc qu'Android n'en est pas à sa dernière version et pendant que ses développeurs réfléchissent déjà à un nom de friandise commençant par x, y ou z,<sup>17</sup> d'autres sont encore en train d'étudier ses fonctionnalités dont les actuelles sont détaillées plus loin.

## 5.1 HISTORIQUE

Une plateforme avec une si grande influence aujourd'hui mérite un passage sur ce qui en a fait une des priorités de tout développeur mobile. Ainsi l'histoire d'Android commence en 2003 lorsqu'une startup au nom éponyme est fondée. Si peu de détails transpirent au sujet de cette startup, il est tout de même de notoriété publique que c'est un ancien employé de chez Apple, Andy RUBIN, qui l'a fondée et son intention aurait bel et bien été, dès le commencement, de développer un système d'exploitation pour mobiles. Mais en 2005, RUBIN accepte de céder Android Inc. à Google pour une somme estimée à cinquante millions de dollars américains et de devenir vice-président de l'ingénierie chez Google afin d'y superviser le développement de sa création.<sup>18</sup>

Si quelqu'un d'autre est aujourd'hui assis à cette place de VP, il n'empêche que la plateforme évolue beaucoup sous l'aile de RUBIN. C'est en 2006 que la décision est prise d'adapter OS Linux en quelque chose de moins archaïque et de plus apte à servir un utilisateur de mobile. Mais c'est en 2007 que le vrai changement se produit. En novembre de cette année est officiellement annoncée l'OHA – Open Handset Alliance – un regroupement d'entreprises dont le but est d'instaurer des standards open source aux mobiles afin de donner un coup de fouet au secteur. Il en résulte qu'en 2008 sort le premier mobile tournant sous Android, démarrant ainsi une longue course aux

---

<sup>14</sup> cf. 5.3 Google Play

<sup>15</sup> (DANON, 2014)

<sup>16</sup> (ZDNet.fr, 2015)

<sup>17</sup> cf. 5.2 Les différentes versions d'Android et leurs APIs

<sup>18</sup> (LEGGETT, 2011)

nouvelles versions, aux fonctionnalités innovantes et aux performances toujours meilleures. La toute dernière version en date, annoncée en mai 2015, est Android 6.0 – Marshmallow – mais Android N, la prochaine version dont un aperçu et les outils de développement sont déjà disponibles, devrait voir sa date de sortie officielle révélée prochainement.<sup>19</sup>

Google officie cependant un changement de stratégie depuis peu. Si jusque-là Android est principalement destiné aux mobiles et, par extension, aux tablettes, l'avènement d'Android au sein d'autres dispositifs peut désormais être observé. Une annonce en mars 2014 révèle Android Wear,<sup>20</sup> un système d'exploitation destiné aux montres connectées. Si cette dernière annonce n'est pas surprenante pour la presse, les deux suivantes qui ont lieu en juin de la même année le sont un peu plus. En effet, c'est à son rendez-vous annuel avec le public, le Google I/O, que Google annonce Android TV,<sup>21</sup> un système d'exploitation destiné à envahir les salons, ainsi qu'Android Auto,<sup>22</sup> un système « projeté » permettant aux conducteurs de contrôler leur smartphone Android en utilisant uniquement les différents contrôles de la voiture.

Soit, Android a encore une longue vie devant lui quand bien même elle commence à devenir multidisciplinaire. Les développeurs ne sauraient trop espérer que le support que Google leur offre perdure au fil des nouveautés amenées par cet OS si polyvalent et accessible.

## 5.2 LES DIFFÉRENTES VERSIONS D'ANDROID ET LEURS APIs

Comme expliqué plus haut, Android est passé par de nombreuses versions, chacune amenant son lot de nouveautés. Si Marshmallow, soit Android 6.0.1, est la dernière version en date, c'est bien avec Android 1.1, alors nommé Petit Four semble-t-il,<sup>23</sup> que tout commence chez le consommateur. Il est intéressant de noter qu'à partir de la version 1.5, ou Cupcake, chaque version se voit attribuer le nom d'une sucrerie selon l'ordre alphabétique, les précédentes subissant le même traitement plus tard pour correspondre à cette logique. Un certain nombre d'entre elles étant toujours en circulation, il est bon d'y penser lors du développement d'une application afin de décider du public cible et donc de choisir l'API à utiliser. Par exemple, KitKat, ou Android 4.4, est au moment de la rédaction de ce paragraphe, la version d'Android la

---

<sup>19</sup> (LARDINOIS, 2015)

<sup>20</sup> (D'ORAZIO, 2014)

<sup>21</sup> (OPAM, 2014)

<sup>22</sup> (BANERJEE, 2014)

<sup>23</sup> (AMADEO, 2012)

plus présente sur le marché.<sup>24</sup> Il est donc de bon ton actuellement, lors du développement d'une application, de la rendre compatible pour cette version.

Ci-après la liste des principales versions d'Android ainsi que la version de l'API correspondante :

Version	Nom de code	API	Année de distribution	Caractéristiques notables
1.0	<i>Apple pie</i>	1	2008	Application Youtube Google Maps Synchronisation Gmail, contacts et Google Agenda Support de l'appareil photo Navigateur web Téléchargements et mises à jour via l'Android Market <sup>25</sup>
1.1	<i>Petit Four puis Banana Bread</i>	2	2009	Sauvegarde des pièces jointes des MMS
1.5	<i>Cupcake</i>	3	2009	Support du bluetooth Clavier tactile Support des vidéos
1.6	<i>Donut</i>	4	2009	Framework pour le tactile
2.0	<i>Eclair</i>	5 à 7	2009	Amélioration de l'interface
2.2.x	<i>Froyo</i>	8	2010	Installation des applications sur la mémoire externe Support de l'upload de fichier dans le navigateur

<sup>24</sup> (Dashboards | Android Developers, 2016)

<sup>25</sup> cf. 5.3 Google Play

2.3.x	<i>Gingerbread</i>	9 et 10	2010	Amélioration de l'interface et de l'ergonomie Ajout des fonctionnalités sociales Support de l'appel vidéo
3.x.x	<i>Honeycomb</i>	11 à 13	2011	Support du multicœur Meilleure stabilité sur tablette Introduction de l'Action Bar et des fragments
4.0.x	<i>Ice Cream Sandwich</i>	14 et 15	2011	Nouvel écran de verrouillage Amélioration du navigateur et de la reconnaissance vocale Reconnaissance faciale
4.1.x	<i>Jellybeans</i>	16	2012	Google Now, un assistant personnel intelligent Recherche vocale Options d'accessibilité
4.2.x		17	2012	Widgets sur l'écran de verrouillage Plusieurs sessions possibles sur tablettes
4.3.x		18	2013	Support du 4K
4.4.x	<i>KitKat</i>	19	2013	Interface translucide Meilleures notifications
5.0.x	<i>Lollipop</i>	21	2014	Material design <sup>26</sup>

<sup>26</sup> cf. 5.6 Design adaptatif et le material design

5.1.x		22	2015	Appels haute définition
6.0	<i>Marshmallow</i>	23	2015	Android Pay, un système de paiement sans carte depuis son mobile Meilleure gestion des permissions Support de l'authentification par empreintes digitales

Figure 21 : tableau des différentes versions d'Android<sup>27</sup>

Voici la répartition des différentes versions d'Android sur le marché en date du 1er février 2016.<sup>28</sup>

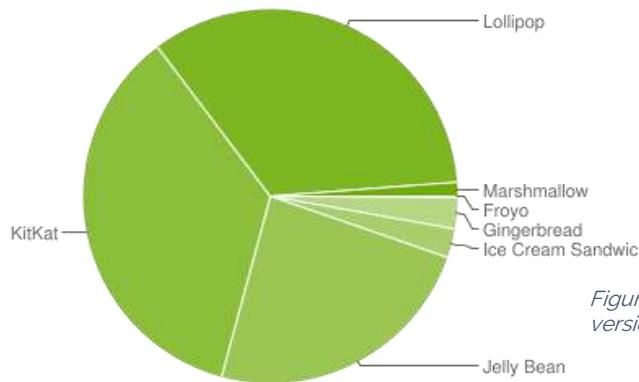


Figure 22 : répartition des différentes versions d'Android sur le marché

Lollipop, KitKat et Jellybeans sont sans surprise les trois versions les plus populaires d'Android et donc celles à cibler lors du développement d'une application. Il ne faut cependant pas oublier Marshmallow, encore très récente mais très prometteuse, les dispositifs compatibles Android 6.0 ne faisant qu'arriver.

### 5.3 GOOGLE PLAY



<https://play.google.com/>

Figure 23 : logo Google Play

<sup>27</sup> (Android versions comparison | Comparison tables - SocialCompare, 2016)

<sup>28</sup> (Dashboards | Android Developers, 2016)

Immense bibliothèque d'applications, musiques, films et livres payants ou non en ligne, Google Play, anciennement Android Market, est le magasin virtuel où tout utilisateur Android peut rechercher et télécharger ce qu'il désire ainsi qu'offrir un feedback pour chacun de ses achats. L'acheteur peut revenir sur la page d'une application pour y laisser une note, un commentaire ou encore mettre à jour cette application.

Ce service est aussi utile aux développeurs, ceux-ci pouvant y publier une application avec un minimum d'effort. En effet, il leur suffit pour ce faire d'exporter l'application sur Google Play, la faire certifier et enfin l'envoyer sur le Store.

## 5.4 LE SOFTWARE DEVELOPMENT KIT D'ANDROID

Le kit de développement d'Android représente un ensemble d'outils destinés à faciliter la création sur cette plateforme. Car si les applications Android sont développées en Java,<sup>29</sup> c'est en fait une variante de ce langage que parle un appareil muni de l'OS mobile de Google. C'est là qu'intervient le SDK, permettant d'écrire du Java orienté Android en s'assurant qu'Android Studio, l'IDE officiel pour le SDK Android, puisse assister au mieux les développeurs.

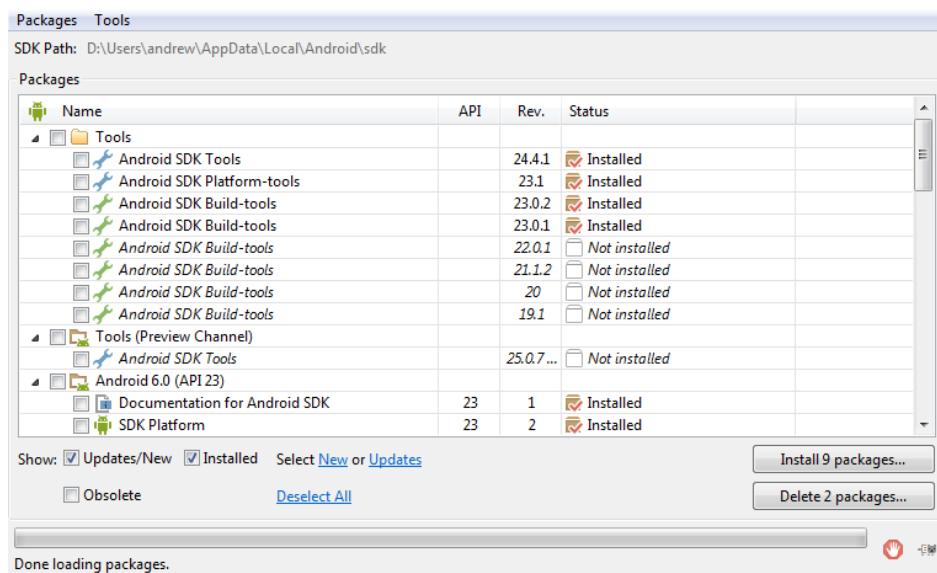


Figure 24 : interface utilisateur du SDK Android

L'altruisme de Google envers les développeurs n'est cependant pas suffisant. Le Java étant le langage de programmation utilisé ici, il faut également installer en amont l'environnement Java, le JDK – Java Development Kit.

<sup>29</sup> cf. 5.7.1 Java

Le SDK fournit donc une quantité d'outils pratiques tels un débogueur, des bibliothèques logicielles, un émulateur qui permet de simuler les différentes versions d'Android, de la documentation ou encore des exemples de code et des tutoriaux. Enfin, chaque version du SDK Android contient également l'intégralité du code source d'Android, avis aux amateurs.

Disponible à tous sur le site officiel du guide des développeurs Android,<sup>30</sup> son interface (cf. Figure 24 : interface utilisateur du SDK Android) est intuitive et permet de choisir quelles API installer en amont du développement.

## 5.5 ARCHITECTURE

L'architecture d'Android est présentée ci-après par ses développeurs comme s'organisant en cinq couches, de la plus haute à la plus basse<sup>31</sup> :

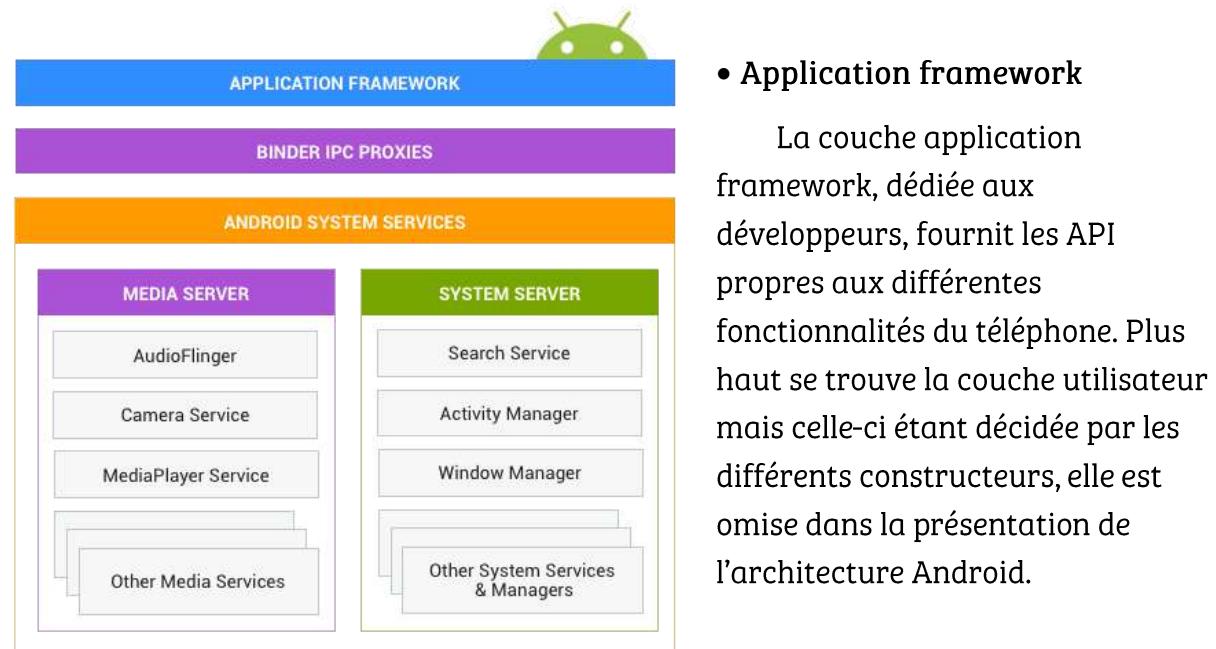


Figure 25 : architecture Android selon l'Android Open Source Project, partie 1

### • Binder IPC proxies

La couche Binder IPC – Inter-Process Communication, ou communication interprocessus – permet au framework de passer outre les limites du processus afin de faire appel au code des services système Android. Ce procédé est invisible pour les développeurs mais permet néanmoins la communication entre les applications et les différentes fonctionnalités du téléphone.

<sup>30</sup> (Download Android Studio and SDK Tools | Android Developers, s.d.)

<sup>31</sup> (Android Interfaces and Architecture | Android Open Source Project, 2016)

### • Android system services

La couche system services fait donc le lien entre le framework et le hardware du téléphone. Il est toujours question d'accès aux fonctionnalités qui s'organisent ici de façon modulaire au sein de deux grandes catégories : la partie « system », qui gère les notifications, les fenêtres et autres ; et la partie « media » dédiée à l'enregistrement ou à la lecture de différents médias.

### • HAL

La couche HAL – Hardware Abstraction Layer, ou couche d'abstraction du matériel – définit une interface hardware standard que les constructeurs peuvent implémenter et permet ainsi à Android de ne pas trop se préoccuper du hardware sur lequel il tourne. C'est la couche destinée aux différents constructeurs de mobile.



Figure 26 : architecture Android selon l'Android Open Source Project, partie 2

### • Linux kernel

La couche Linux kernel contient tous les pilotes de l'appareil. C'est ici que se situe le dernier pont entre le matériel et le logiciel. Son nom vient du fait qu'Android se base sur un Kernel Linux adapté aux mobiles.

## 5.6 DESIGN ADAPTATIF ET LE MATERIAL DESIGN

Si Android est l'OS mobile le plus populaire, c'est très certainement en grande partie parce qu'il est compatible avec le plus grand nombre de dispositifs. Ainsi, ce qui fait sa force, c'est bien cette volonté d'adaptation et donc sa flexibilité au regard des différentes dimensions d'affichage. Un tel exploit est possible car Android catégorise les différents écrans selon deux propriétés : la taille et la densité. Chacune de ces deux propriétés peut prendre quatre valeurs : small, normal, large et xlarge pour la taille et low (ldpi), medium (mdpi), high (hdpi) et extra high (xhdpi) pour la densité.

Dès lors, en se basant sur ces standards, les développeurs peuvent considérer une unité de mesure relative, le dp ou dip pour Density-independant Pixels, permettant

ainsi de conserver les proportions des ressources quelles que soient les situations lors de l'agencement de celles-ci à l'écran. Une autre unité existe également, le sp pour Scale-independant Pixel, tout aussi relative mais dédiée cette fois-ci aux polices d'écriture, ce afin de s'adapter aux préférences de l'utilisateur.<sup>32</sup>

Il n'y a ensuite plus qu'à créer différentes ressources en fonction des catégories d'écran et de les disposer grâce aux outils de visualisation offerts par Android Studio. Cette remarquable permissivité donne alors lieu à un design adaptatif, soit un design qui réagit au mieux en présence de différentes tailles d'écran et de leur orientation.

Pour aller plus loin et assurer l'ergonomie des projets sur Android, Google a décidé de publier un guide sur le material design.<sup>33</sup> Design adaptatif et intuitif de leur création, le material design est une collection de règles, un standard, sur lesquelles peuvent se baser tous les développeurs afin d'obtenir une application pratique qui ne nécessite pas d'explications quant à la navigation en son sein.

L'objectif du material design est d'incarner un langage visuel qui unit les bonnes pratiques de design en général avec l'innovation et les possibilités aujourd'hui proposées par la technologie et les sciences. Le résultat donne une présentation cohérente respectant la physique des espaces en trois dimensions et offrant une expérience utilisateur riche. L'utilisateur se voit alors expérimenter de façon intuitive avec l'espace virtuel qu'incarne l'interface tactile de l'application, guidé par des règles centrées sur la profondeur, la lumière et les ombres, le mouvement et les animations des différents éléments ainsi que leurs interactions au sein de cet espace.

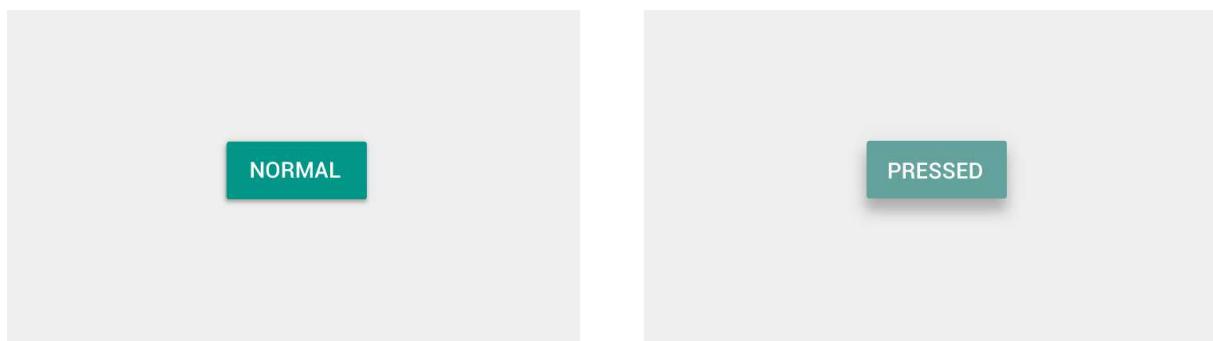


Figure 27 : les deux états d'un bouton selon le material design

Dès lors, et à titre d'illustration, une tuile portant une ombre sur un élément sous-jacent semble détachée de son support, indiquant une possible interaction. Appuyer sur cette tuile voit son ombre s'agrandir, la distance virtuelle entre elle et son support augmentant. Ce bouton est désormais activé, son ombre faisant foi, et le reste jusqu'à

<sup>32</sup> (Supporting Multiple Screens | Android Developers, 2016)

<sup>33</sup> (Introduction - Material Design - Google design guidelines, s.d.)

une nouvelle interaction avec l'utilisateur (cf. Figure 27 : les deux états d'un bouton selon le material design).

Bien entendu, ceci n'était qu'un exemple parmi beaucoup de règles. L'application de celles-ci n'est pas des plus complexes et donne un résultat très satisfaisant. Aussi, certains éléments prévus dans Android Studio comme l'Action Bar ou le Navigation Drawer prennent déjà en compte le material design. Un rappel de toutes ces lignes directrices peut être trouvé sur le guide de Google disponible en ligne.<sup>34</sup>

## 5.7 DÉVELOPPER POUR ANDROID

Savoir ce qu'est Android et ce qu'il offre est certes intéressant mais du point de vue du développeur ça ne suffit pas. C'est pourquoi ce point se concentre sur ce qui constitue une application Android en termes d'implémentation. Sans encore se lancer dans du code concret, c'est à l'explication des termes, concepts et paradigmes du Java orienté Android que s'intéresse cette partie.

### 5.7.1 JAVA

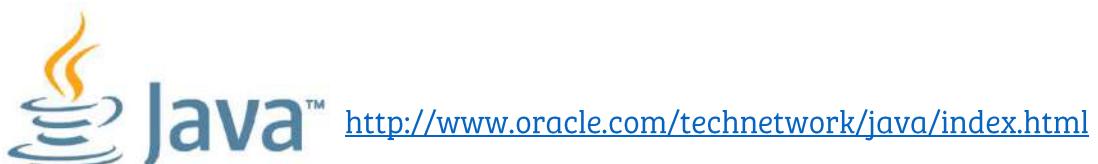


Figure 28 : logo de Java

S'il est quelque chose sans quoi Android ne serait pas, c'est bien son langage d'implémentation. Le Java donc est un langage aujourd'hui particulièrement populaire qui fut introduit en 1995 par John CAGE, alors directeur du bureau des sciences chez Sun Microsystems, et Marc ANDREESSEN, cofondateur et vice-président exécutif de Netscape.<sup>35</sup> En 2010, Oracle rachète Sun et devient dès lors détenteur et développeur en titre de Java.<sup>36</sup>

Le Java est un langage de programmation orienté objet dont la particularité et l'objectif sont la portabilité des programmes écrits dans ce langage. Ainsi, ces logiciels peuvent être portés vers divers systèmes d'exploitation tels que Windows, UNIX, Mac OS ou GNU/Linux avec peu ou pas de modifications. Qu'il soit orienté objet signifie que les types de données d'une application doivent être associés aux différentes opérations pouvant être effectuées sur ces données. Cette association se fait au sein d'une entité

<sup>34</sup> (Introduction - Material Design - Google design guidelines, s.d.)

<sup>35</sup> (BYOUS, 2003)

<sup>36</sup> (Oracle and Sun Microsystems | Strategic Acquisitions | Oracle, s.d.)

nommée « classe ». Dans cette classe se trouvent donc les objets, ces structures qui définissent les données lorsque varient leurs propriétés, ainsi que leurs méthodes associées, constituant les multiples opérations implémentées autour de ces objets.

Voilà pour le paradigme principal du langage, celui-ci introduisant d'autres concepts directement liés tels l'héritage et le polymorphisme, le premier permettant la réutilisabilité et l'adaptabilité d'objets grâce au deuxième qui, lui, consiste à fournir une interface unique à des entités de types différents. Pour le reste, tel tout autre langage de programmation, Java possède également ses propres mots réservés et ses types ainsi que sa propre approche des différentes structures de contrôle.

### 5.7.2 LA CLASSE APPLICATION

La classe Application est la classe de base d'une application servant à maintenir l'état global de celle-ci.<sup>37</sup> Chaque application en possède une par défaut et l'instancie en premier dès son lancement. Il est possible d'implémenter cette classe, de la surcharger ou encore d'implémenter une classe qui en hérite afin, par exemple, de déclarer des constantes globales ou de faire persister des informations comme certains objets ou certaines données et préférences utilisateurs. Cette classe permet également d'accéder au context<sup>38</sup> global de l'application.

### 5.7.3 CONTEXT

Le context d'une application est une interface donnant accès aux informations globales liées à son environnement.<sup>39</sup> Dans la pratique, ce context est accessible depuis n'importe quelle activity grâce à la méthode « getContext() » ou encore « getApplicationContext() ». Il est aussi accessible grâce à un appel à la classe Application. Permettant de faire un lien avec le système, une fois appelé, le context peut servir à démarrer une activity ou à émettre et recevoir un intent.<sup>40</sup>

### 5.7.4 ACTIVITY

Une activity, élément essentiel du développement sur Android, représente une des vues de l'application ainsi que ses interactions avec les autres activities et les fragments<sup>41</sup> qui la constituent s'il y en a. Elle concerne généralement un des cas

---

<sup>37</sup> (Application | Android Developers, 2016)

<sup>38</sup> cf. 5.7.3 Context

<sup>39</sup> (Context | Android Developers, 2016)

<sup>40</sup> cf. 5.7.7 Intent

<sup>41</sup> cf. 5.7.5 Fragment

d'utilisation<sup>42</sup> de l'application et se charge donc de faire évoluer l'interface graphique et de répondre aux différentes entrées de l'utilisateur.

#### 5.7.4.1 LE CYCLE DE VIE D'UNE ACTIVITY

Un des aspects importants des activities ainsi que du développement sur Android en général est celui du cycle de vie d'une activity. En effet, au fur et à mesure de son utilisation, une activity passe en fait d'un état à un autre. À chaque changement d'état, le système fait appel à une méthode que le développeur peut redéfinir afin d'implémenter le comportement que l'application doit alors adopter.

Le schéma fournit par le site Android Developers<sup>43</sup> (cf. Figure 29 : cycle de vie d'une activité selon le site Android Developers) décrit on ne peut mieux ce cycle de vie.

##### • **onCreate()**

La méthode `onCreate()` est appelée dès la création de l'activity et est destinée à l'initialisation des variables statiques, à la création des vues et à la liaison<sup>44</sup> des différentes données. Cette méthode fournit également la sauvegarde du précédent état de l'activity et est toujours directement suivie d'un appel à la méthode `onStart()`.

##### • **onRestart()**

`onRestart()` est la méthode appelée lors de la reprise de l'activity après qu'elle ait été stoppée lors d'un appel à la méthode `onStop()`.

##### • **onStart()**

Directement appelée après `onCreate()`, `onStart()` marque le moment où l'utilisateur peut voir l'activity. Elle est suivie d'un appel à `onResume()` ou `onStop()` en fonction de si l'activity passe au premier plan de l'application ou pas.

##### • **onResume()**

`onResume()` marque le moment à partir duquel l'utilisateur peut interagir avec l'activity et ne peut qu'être suivie d'un appel à `onPause()`.

---

<sup>42</sup> cf. 6.1.3 Diagramme des cas d'utilisation

<sup>43</sup> (Activity | Android Developers, 2016)

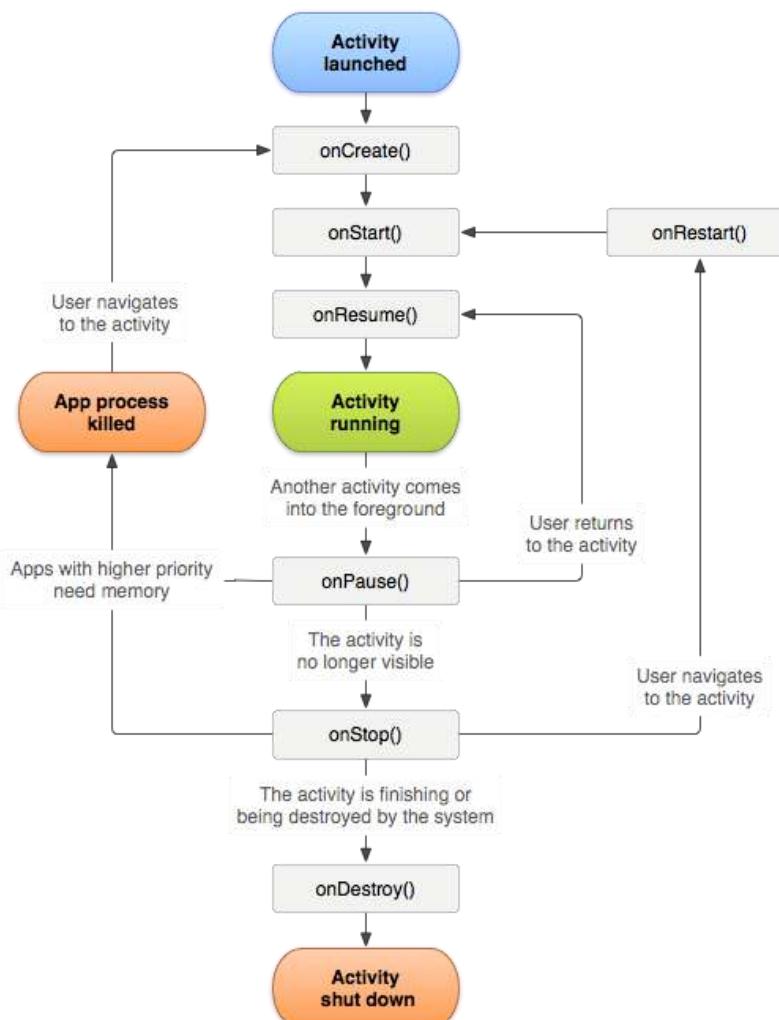
<sup>44</sup> cf. 5.7.11 MVVM et data binding

### • onPause()

Utilisée pour sauver les données que les développeurs veulent persistantes, pour arrêter les animations ou pour mettre en pause les différents processus utilisant les ressources de l'appareil, la méthode onPause() se veut une méthode légère au possible étant donné que l'activity ne peut reprendre avant qu'onPause() ne se termine.

### • onStop()

Appelée lorsque l'activity devient invisible à l'utilisateur dû à un passage à une autre activity ou à la destruction de celle en cours. onStop() est suivie de onDestroy() ou de onRestart() selon que l'activity courante est détruite ou qu'elle redevient active.



### • onDestory()

Signifiant la fin de l'activity, onDestroy() est appelée pour libérer de la ressource avant que l'application ne passe à autre chose.

Il est intéressant de noter qu'un changement de configuration, c'est-à-dire un changement d'orientation de l'appareil, cause également la destruction de l'activity avant sa recréation. C'est ainsi que la sauvegarde des données visant à être conservées lors de l'appel à onPause() prend toute son importance.

Figure 29 : cycle de vie d'une activité selon le site Android Developers

## 5.7.5 FRAGMENT

Présents depuis Android Honeycomb, ou Android 3.0,<sup>45</sup> et rendus rétrocompatibles par les bibliothèques de compatibilité,<sup>46</sup> les fragments sont sensiblement semblables aux activités, bien qu'un fragment ne puisse exister qu'au sein d'une activity. Les fragments servent à créer des interfaces plus complexes, à moduler le code ou encore à faciliter le travail d'adaptation des interfaces aux différentes tailles d'écran,<sup>47</sup> très pratique pour le développement d'applications destinées aussi bien aux smartphones qu'aux tablettes.

Il est intéressant de noter les méthodes suivantes comme faisant partie du cycle de vie (cf. Figure 30 : cycle de vie d'un fragment selon le site Android Developers) d'un fragment :

- **onAttach()**

onAttach() est appelée lorsque le fragment est ajouté à l'activity et donc avant onCreate().

- **onDetach()**

onDetach() est appelée lorsque le fragment est détaché de l'activity et donc après onDestroy().

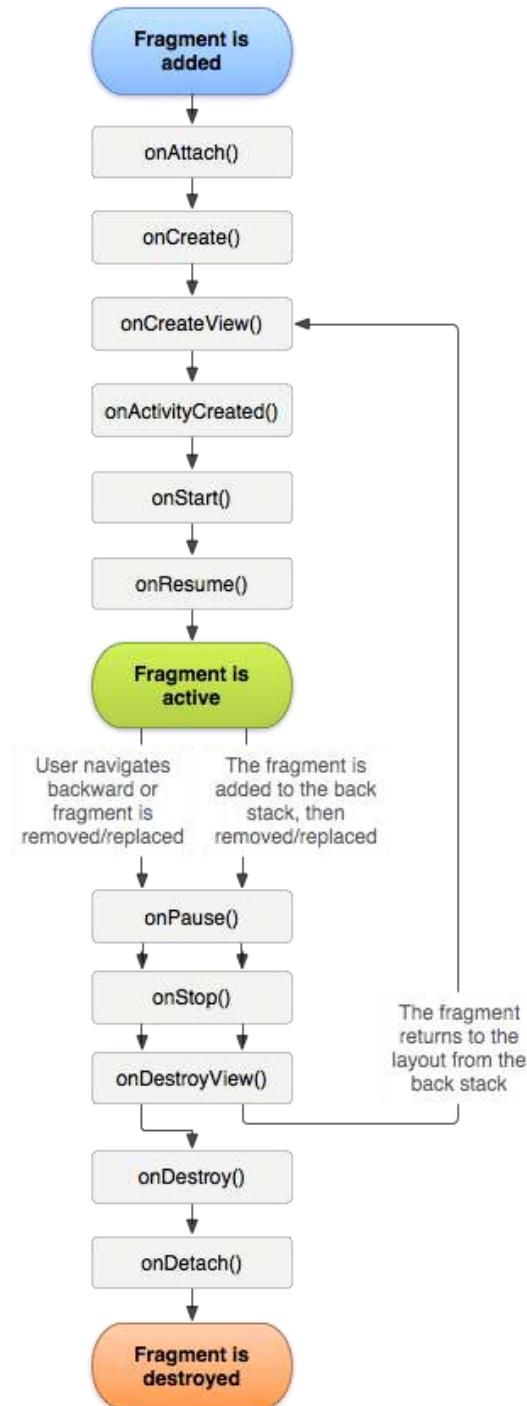


Figure 30 : cycle de vie d'un fragment selon le site Android Developers

<sup>45</sup> cf. 5.2 Les différentes versions d'Android et leurs APIs

<sup>46</sup> cf. 5.7.10 Les bibliothèques de compatibilité

<sup>47</sup> (Activity | Android Developers, 2016)

- **onActivityCreated()**

Cette méthode est appelée dès que l'activity est créée et que les vues du fragment sont instanciées. C'est au sein cette méthode qu'il est conseillé de restaurer l'état du fragment lorsque nécessaire.

- **onActivityDestroyed()**

`onActivityDestroyed()` est appelée juste avant `onDestroy()`.

## 5.7.6 SERVICE

Proches des activités et normalement utilisés pour de longues tâches en arrière plan tels le téléchargement ou le traitement lourd de données ou encore la lecture de fichiers audio, les services sont indépendants de l'interface utilisateur et continuent donc leurs opérations même lorsque l'application passe à autre chose.<sup>48</sup> Ils possèdent néanmoins leur propre cycle de vie et ont accès au context de l'application<sup>49</sup> si besoin.

## 5.7.7 INTENT

Un intent est une description abstraite d'une opération à effectuer et sert ainsi de message système qu'utilisent principalement les activités et les services.<sup>50</sup> Les pièces d'informations principales d'un intent sont l'action à effectuer ainsi que les données utilisées pour effectuer cette action. Des « extras » y sont également souvent adjoints, c'est-à-dire des données ou informations supplémentaires utiles à l'opération à réaliser.

Les intents se présentent sous deux formes :

- **Les intents explicites**

Les intents explicites ont des composants bien précis qui fournissent la classe exacte à exécuter. Le meilleur exemple ici est l'intent servant à démarrer une activity, la classe de l'activity à afficher étant spécifiée en son sein.

---

<sup>48</sup> (Services | Android Developers, s.d.)

<sup>49</sup> cf. 5.7.3 Context

<sup>50</sup> (Intent | Android Developers, 2016)

### • Les intents implicites

Les intents implicites, contrairement aux explicites, ne précisent pas la classe à exécuter. Ainsi, ils doivent posséder assez d'informations pour que le système puisse proposer un composant adéquat à exécuter. Vouloir envoyer un mail depuis une application nécessite ce genre d'intent, il faut alors suffisamment spécifier la nature du processus à exécuter pour que le système puisse proposer de lancer une application adéquate déjà installée.

## 5.7.8 GESTURES

L'arrivée du tactile dans la téléphonie mobile et surtout du côté des smartphones est une révolution dans le domaine de l'interaction entre l'homme et la machine que seuls le clavier et la souris peuvent se vanter d'avoir égalée. Ainsi, les gestes sont les différentes possibilités dont dispose l'utilisateur pour communiquer avec les applications par le biais des fonctionnalités tactiles de l'appareil. Si elles paraissent aujourd'hui naturelles à presque tout le monde, c'est que les développeurs ont dû penser les fonctions de chacune, quitte à instaurer un standard, pour ensuite devoir les implémenter et les utiliser au mieux afin de maximiser l'ergonomie de leurs softwares.

Désormais abouties au point que Google les détaille dans son guide sur le matériel design,<sup>51</sup><sup>52</sup> ces gestes incarnent un langage à part entière où les gestes à effectuer, appelés Touch Mechanics, seraient le vocabulaire et où leurs effets, les Touch Activities, représenteraient la grammaire.

Voici donc une liste exhaustive des Touch Mechanics actuellement existantes avec leur nom, une description du mouvement à effectuer et un exemple de résultat – une Touch Activity – généralement engendré par cette Touch Mechanic<sup>53</sup> :

Nom	Description	Exemple d'effet
Touch	Pression à un doigt, retrait du doigt	Sélectionner
Double touch	Pression à un doigt, retrait du doigt, pression à un doigt, retrait du doigt	Zoomer

<sup>51</sup> (Introduction - Material Design - Google design guidelines, s.d.)

<sup>52</sup> cf. 5.6 Design adaptatif et le material design

<sup>53</sup> (Gestures - Patterns - Google design guidelines, s.d.)

<b>Drag/Swipe/Fling<sup>54</sup></b>	Pression à un doigt, mouvement, <sup>55</sup> retrait du doigt	Faire défiler
<b>Long press</b>	Pression à un doigt, attendre, retrait du doigt	Selectionner dans une liste
<b>Long-press drag</b>	Pression à un doigt, attendre, mouvement, retrait du doigt	Prendre et déplacer
<b>Double-touch drag</b>	Pression à un doigt, retrait du doigt, pression à un doigt, mouvement, retrait du doigt	Prendre et déplacer
<b>Pinch open</b>	Pression à deux doigts, mouvement vers l'extérieur des doigts, retrait des doigts	Zoomer
<b>Pinch closed</b>	Pression à deux doigts, mouvement vers l'intérieur des doigts, retrait des doigts	Dézoomer
<b>Two-finger touch</b>	Pression à deux doigts, retrait des doigts	Dézoomer
<b>Two-finger drag/swipe/fling</b>	Pression à deux doigts, mouvement, retrait des doigts	Faire défiler
<b>Two-finger long press</b>	Pression à deux doigts, attendre, retrait des doigts	Aucun (gesture peu répandue)
<b>Two-finger long press drag</b>	Pression à deux doigts, attendre, mouvement, retrait des doigts	Prendre et déplacer
<b>Two-finger double touch</b>	Pression à deux doigts, retrait des doigts, pression à deux doigts, retrait des doigts	Dézoomer
<b>Rotate</b>	Pression à deux doigts, rotation simultanée des deux doigts autour	Faire pivoter

<sup>54</sup> Les trois se distinguent selon la vitesse à laquelle la Touch Mechanic est effectuée.

<sup>55</sup> Un mouvement consiste à faire glisser un ou plusieurs doigts dans une direction en restant en contact avec la surface tactile.

du même point central, retrait des doigts

Figure 31 : tableau des différentes gestes d'Android

Étant donnée l'importance de ces différentes fonctionnalités, Android est fourni avec les classes et interfaces nécessaires à une implémentation facilitée de celles-ci. Ainsi, un clic sur un bouton déclenche un touch event que le développeur n'a qu'à intercepter grâce à un écouteur de la classe OnClickListener précédemment lié à la vue concernée. Cet écouteur implémente la réaction que l'application doit avoir lors d'un tel évènement. Le système se charge alors de détecter l'évènement, de déterminer la nature de celui-ci et de déclencher le comportement adéquat choisi par le développeur.

### 5.7.9 ADAPTER

Un adapter en Android consiste en un concept qui regroupe deux éléments, à savoir l'AdapterView et l'objet Adapter.<sup>56</sup> Si le premier représente les vues de type ListView, GridView, RecyclerView ou encore Spinner qui sont en fait des listes de présentations différentes utiles pour présenter des informations à l'utilisateur, le deuxième est quant à lui l'objet qui permet de préparer ces informations afin de les afficher au sein du premier. Ainsi, en créant une instance d'Adapter et en lui passant les informations nécessaires, celui-ci s'occupe d'effectuer les traitements prescrits. Le résultat est alors reflété par l'AdapterView, présentant les données comme désiré et prenant en charge les potentielles interactions avec l'utilisateur sur ces dernières.

### 5.7.10 LES BIBLIOTHÈQUES DE COMPATIBILITÉ

À chaque nouveauté son lot de problèmes. En effet, si chaque mise à jour d'Android apporte sa part de nouvelles fonctionnalités, chaque application déjà existante peut potentiellement devenir obsolète. Aussi, développer un nouvel outil pour un parc de dispositifs le plus important possible tout en maximisant sa modernité et donc en implémentant les dernières fonctionnalités Android devient un challenge. C'est pourquoi Google met à disposition ce qu'il appelle la Support Library. Celle-ci a pour but de rendre compatibles les dernières nouveautés d'Android avec les plus vieilles versions du système d'exploitation afin d'éviter aux développeurs de devoir implémenter eux-mêmes ces fonctionnalités avec du code que les vieilles versions d'Android peuvent comprendre. Il existe plusieurs versions de ces bibliothèques,

<sup>56</sup> (Adapter | Android Developers, 2016)

chacune s'intéressant à des fonctionnalités spécifiques ou étant destinée à faire tourner les dernières applications sur tel ou tel système d'exploitation.

Les dates de mise à disposition de ces librairies ainsi que les spécifications de leurs différentes fonctionnalités étant communiquées de façon plutôt disparate par Google et étant donné leur nature complexe, il n'est pas réellement aisé ni même tout à fait intéressant de les expliquer en détail. Cependant, étant donné leur indéniable utilité, voici tout de même la liste exhaustive de celles qui sont aujourd'hui disponibles<sup>57</sup> :

- **v4 Support Library;**
- **Multidex Support Library;**
- **v7 Support Libraries;**
- **v8 Support Library;**
- **v13 Support Library;**
- **v14 Preference Support Library;**
- **v17 Preference Support Library for TV;**
- **v17 Leanback Library;**
- **Annotations Support Library;**
- **Design Support Library;**
- **Custom Tabs Support Library;**
- **Percent Support Library;**
- **App Recommendation Support Library for TV;**

Dans la pratique, lors de l'intégration d'une fonctionnalité récente au code d'une application, Android Studio<sup>58</sup> propose l'importation d'une de ces librairies afin de rendre la création compatible avec le plus grand nombre possible d'appareils. Une ligne de code suffit donc souvent à faire plaisir au plus grand nombre.

#### 5.7.11 MVVM ET DATA BINDING

Le Google I/O est la conférence annuelle durant laquelle Google annonce ses dernières nouveautés ainsi que ses projets à venir. C'est lors de l'édition de mai 2015 qu'est annoncée la librairie Data Binding,<sup>59</sup> un cadeau pour les développeurs Android sensé les aider à découpler l'interface utilisateur du reste du code. Les différentes vues de l'application peuvent ainsi être liées à des données sans devoir être référencées dans

---

<sup>57</sup> (Support Library Features | Android Developers, 2016)

<sup>58</sup> cf. 4.1 Android Studio

<sup>59</sup> (Getting Started with Data Binding in Android, 2015)

le code qui manipule ces données, résultant en une permissivité bien plus élevée à l'égard de potentielles modifications de l'interface en aval.

Cette librairie rend donc aisée l'implémentation du code selon le design pattern MVVM, Model View ViewModel, une alternative aux très populaires MVP, Model View Presenter, et MVC, Model View Controller, jusque-là principalement utilisés (cf. Figure 32 : MVVM, MVP et MVC).

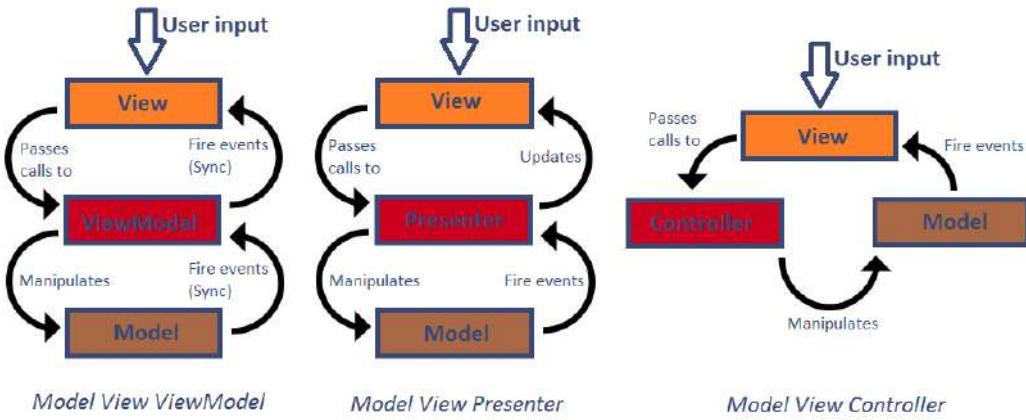


Figure 32 : MVVM, MVP et MVC

#### 5.7.11.1 PRÉSENTATION DU MVVM

Le MVVM est un design pattern visant à séparer au mieux le développement graphique du développement logique d'un programme. Celui-ci correspond donc parfaitement au monde du mobile étant donné la complexité souvent élevée de l'interface visible et la rigueur niveau développement qu'exigent les différentes interactions possibles avec l'utilisateur.<sup>60</sup>

Ce pattern vise à découper le code en trois couches bien distinctes<sup>61</sup> :

- **La couche Model**

Cette couche ne diffère pas de l'habituelle couche Model des autres design patterns. Ainsi, cette partie du code est consacrée à la représentation des données sous une forme qu'il convient de manipuler avec un langage orienté objet. Si la représentation d'un utilisateur, par exemple, est nécessaire à l'application et que cet utilisateur se distingue des autres par son adresse mail et un mot de passe, alors une classe « Utilisateur » est implémentée avec pour arguments « mail » et « motDePasse ». Dans cette classe sont également disponibles les accesseurs et les mutateurs de ces arguments qui servent à obtenir ou modifier leur valeur. Il est aussi d'usage d'y

<sup>60</sup> cf. 5.7.8 Gestures

<sup>61</sup> (BIRCH, 2015)

implémenter les méthodes très proches de ces données comme la méthode de représentation sous forme de String de l'objet, souvent nommée « `toString()` ».

Sous cette couche se trouvent également les services d'accès aux données à distance dans le cas de l'utilisation d'une base de données.

#### • La couche View

De même que la couche Model, la couche View diffère peu de l'habituelle couche View des différents design patterns. Dans le cas d'Android, les différentes vues sont implémentées en XML, Extensible Markup Language ou langage de balisage extensible – un langage de balisage tel que l'est l'HTML<sup>62</sup> – afin d'établir d'emblée les éléments auxquels l'utilisateur a accès. C'est d'ailleurs au sein du code XML que les éléments de la couche ViewModel se voient liés aux éléments de la couche View grâce au data binding.<sup>63</sup>

La logique propre aux vues – et donc n'étant pas liée aux données entrées par l'utilisateur mais plutôt aux interactions avec celui-ci ou encore à la taille de l'écran utilisé<sup>64</sup> – est quant à elle implémentée en Java<sup>65</sup> toujours dans la même couche mais dans un autre fichier.

#### • La couche ViewModel

La couche ViewModel est la couche la plus importante du MVVM et celle qui le distingue des autres design patterns. Elle est en fait une abstraction de la vue qui gère la communication entre la couche Model et la couche View sur le principe de notification. Les éléments de la vue sont donc liés à ceux du ViewModel grâce au data binding. Si une action de l'utilisateur a pour but de modifier des données, alors cette interaction avec la vue doit notifier l'élément concerné dans le ViewModel. L'élément en question est ensuite modifié et même traité si nécessaire pour refléter les changements dans l'interface avant qu'une méthode ne permette de contacter la couche Model afin de modifier et de persister la donnée.

L'utilisateur, en interagissant avec la vue, enclenche donc les fonctionnalités de la librairie Data Binding<sup>66</sup> qui agissent sur les données de la couche la plus élevée à la couche la plus basse et vice versa si nécessaire. Il n'y a ici pas besoin de référencer un des éléments d'une couche depuis une autre pour le modifier, la librairie de Google s'en

---

<sup>62</sup> (WALSH, 1998)

<sup>63</sup> cf. 5.7.11.2 Présentation du data binding

<sup>64</sup> cf. 5.6 Design adaptatif et le material design

<sup>65</sup> cf. 5.7.1 Java

<sup>66</sup> cf. 5.7.11.2 Présentation du data binding

charge automatiquement. Il est ainsi bien plus aisé de modifier une des couches ultérieurement ou encore d'effectuer des tests unitaires, consistant en une procédure permettant de tester différentes parties d'un programme individuellement.<sup>67</sup>

#### 5.7.11.2 PRÉSENTATION DU DATA BINDING

La librairie Data Binding est donc la clé de voûte concernant le MVVM sur Android. Disponible gratuitement, cette librairie s'intègre à un projet presque comme toutes les autres<sup>68</sup> à la différence près qu'il faut en plus l'activer dans le fichier build.gradle<sup>69</sup> de la façon suivante<sup>70</sup> :

```
android {  
    ...  
    dataBinding {  
        enabled = true  
    }  
}
```

Lier un élément de la vue à une variable de la couche ViewModel se fait dans le code XML :

```
<?xml version="1.0" encoding="utf-8" ?>  
<layout xmlns:android="http://schemas.android.com/apk/res/android">  
  
<data>  
    <variable name="user" type="com.example.UserViewModel"/>  
</data>  
  
<LinearLayout  
    android:orientation="vertical"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent">  
  
    <EditText android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@={user.mail}"/>  
  
    <EditText android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@={user.password}"/>  

```

<sup>67</sup> (Unit Testing, 2016)

<sup>68</sup> cf. 4.2 Gradle

<sup>69</sup> cf. 4.2 Gradle

<sup>70</sup> (Data Binding Guide | Android Developers, s.d.)

```
</LinearLayout>  
</layout>
```

La ligne suivante décrit le type du ViewModel qui est lié à la vue :

```
<variable name="user" type="com.example.UserViewModel"/>
```

Et cette ligne se charge de lier la propriété du ViewModel à l'attribut de la vue souhaité :

```
android:text="@={user.mail}"/>
```

Il faut alors se rendre dans la classe UserViewModel pour y voir la représentation des données :

```
package com.andrew.fictional_login.viewmodel;  
  
import android.databinding.BaseObservable;  
import android.databinding.Bindable;  
  
import com.andrew.fictional_login.BR;  
  
public class UserViewModel extends BaseObservable {  
    @Bindable  
    private String mail;  
    @Bindable  
    private String password;  
  
    public UserViewModel(String mail, String password) {  
        this.mail = mail;  
        this.password = password;  
    }  
  
    public String getMail() {  
        return this.mail;  
    }  
  
    public String getPassword() {  
        return this.password;  
    }  
  
    public void setMail(String mail) {
```

```

        this.mail = mail;
        notifyPropertyChanged(BR.mail);
    }

    public void setPassword(String password) {
        this.password = password;
        notifyPropertyChanged(BR.password);
    }
}

```

Bien que semblable à une classe classique d'objets en Java, elle diffère toute même de celles-ci par la présence de deux éléments. L'annotation « @Bindable » permet de générer les entrées nécessaires dans la classe BR durant la compilation tandis que « notifyPropertyChanged() » permet de notifier la couche View afin d'appliquer les modifications de l'interface en temps réel. Cette classe BR est générée par le data binding et contient les identifiants des différents éléments des classes de la couche model annotés avec un « @Bindable ». C'est grâce à ces références que fonctionne le système de notification du data binding.<sup>71</sup>

Voici à quoi ressemble la classe BR de l'exemple ci-dessus :

```

package com.andrew.fictional_login;

public class BR {
    public static final int _all = 0;
    public static final int mail = 1;
    public static final int password = 2;
    public static final int user = 3;
}

```

Il reste dès lors à déclencher le binding lors de l'exécution. Cette étape est à réaliser dans le code logique des vues, plus précisément dans la méthode `onCreate()` de la vue concernée.<sup>72</sup> Il faut alors signaler le binding lors de l'instanciation du code XML de la vue en un objet de type `View` manipulable par du code Java :

```

private ActivityLoginBinding binding =
DataBindingUtil.setContentView(this, R.layout.activity_login);
binding.setUser(new UserViewModel("mailExample@example.com",
"passwordExample"));

```

<sup>71</sup> (CAMPBELL, 2015)

<sup>72</sup> cf. 5.7.4.1 Le cycle de vie d'une activity. 5.7.5 Fragment

## 5.8 CRÉATION D'UN PROJET FICTIF

Les principaux concepts nécessaires au développement sur Android étant désormais établis, il est maintenant opportun de s'intéresser à ce que donne le développement d'une application dans la pratique. C'est pourquoi cette partie se consacre à la création d'un projet fictif, une introduction permettant de se faire une idée globale mais néanmoins représentative des tenants et aboutissants de la réalisation d'un tel projet. Sont également abordés ici les derniers éléments propres au code d'une application Android tels que la structure du projet et ses différentes ressources.

L'explication ici en détails de ces points se voit également nécessaire pour pouvoir approcher confortablement le projet clé de ce travail sans devoir rappeler chaque concept en cours de route.

### 5.8.1 DESCRIPTION DE L'APPLICATION FICTIVE

Étant donné que la plupart des applications nécessitent de l'utilisateur qu'il se connecte avant de pouvoir être utilisées, un choix judicieux est de créer un projet consistant uniquement à simuler une identification. Ainsi, l'exemple – de nom « *fictional\_login* » – consiste en une activity munie de deux champs servant à introduire une adresse mail et un mot de passe en plus d'un bouton déclenchant la demande de connexion. La requête est représentée par un faux chargement suivi de la fermeture de l'application.



Figure 33 : activity principale de *fictional\_login*

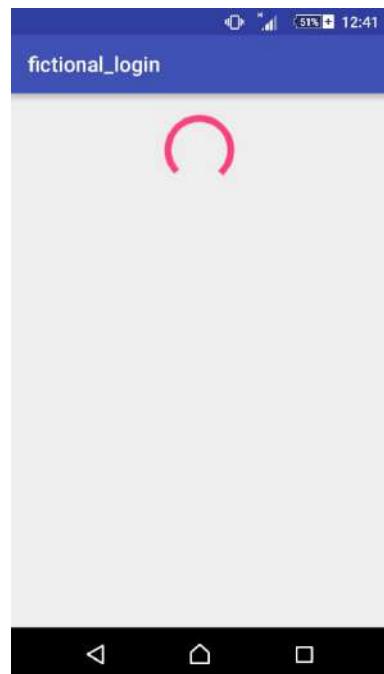


Figure 34 : chargement de *fictional\_login*

## 5.8.2 STRUCTURE

Un projet Android pouvant être de taille conséquente, il est nécessaire de maîtriser sa structure afin de pouvoir naviguer en son sein le plus efficacement possible. Aussi, une bonne structure assure une meilleure lisibilité ainsi qu'un accès facilité aux différentes ressources.

Android Studio<sup>73</sup> propose de découper le projet de la sorte (cf. Figure 35 : structure d'un projet Android) :

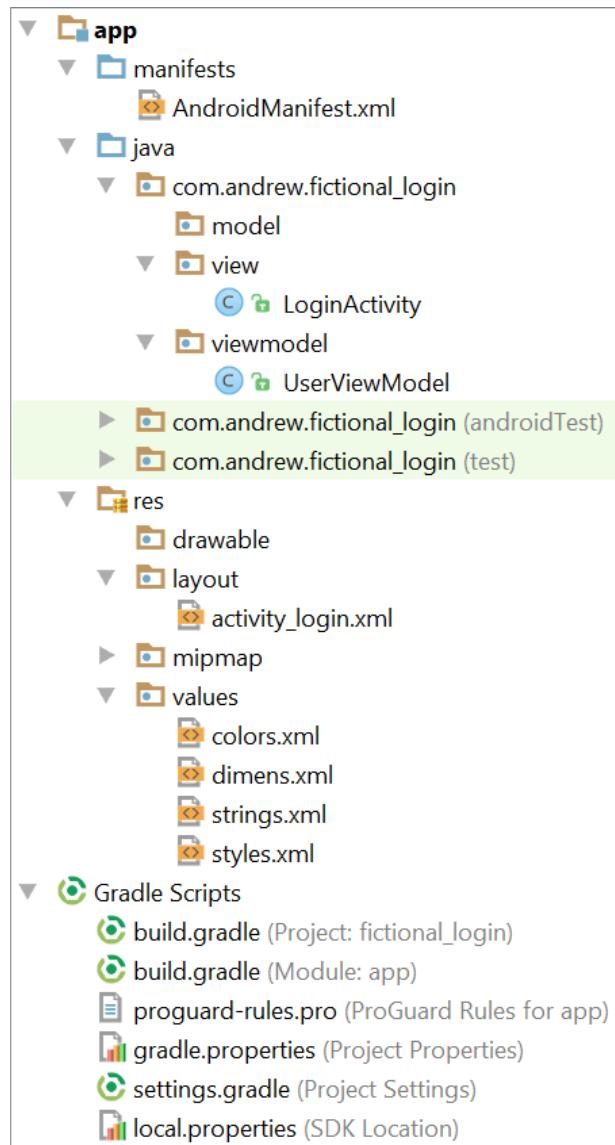


Figure 35 : structure d'un projet Android

### • manifests

Le module « manifests » contient le fichier `AndroidManifest.xml`,<sup>74</sup> un fichier de description de l'application essentiel à tous projets Android.

### • java

Le module « java » est lui consacré à stocker tous les fichiers `.java` rédigés. C'est donc ici qu'est placé le code, c'est alors au développeur de gérer au mieux la disposition des différents packages – pourquoi pas comme ici selon le design pattern MVVM<sup>75</sup> – afin de pouvoir s'y retrouver dans ses classes.

### • res/drawable

C'est dans ce module que sont stockées les différentes ressources visuelles (`.jpg`, `.png`, `.gif`, etc.) utilisées par l'application.

<sup>73</sup> cf. 4.1 Android Studio

<sup>74</sup> cf. 5.8.3 `AndroidManifest.xml`

<sup>75</sup> cf. 5.7.11 MVVM et data binding

- **res/layout**

Celui-ci comporte les différents fichiers layout, ceux-là même écrits en XML et destinés à construire les vues et à binder des éléments de la couche ViewModel à l'interface lors de l'utilisation du data binding.<sup>76</sup> C'est donc ici que se trouve « activity\_login.xml », le layout de la page de login.

- **res/values**

Ce dernier module est quant à lui dédié au stockage des différentes valeurs dont l'application a besoin. Que ce soit les valeurs hexadécimales des couleurs à utiliser pour le thème visuel de l'application, les dimensions des éléments de l'interface ou encore les différents textes présents dans l'application, c'est ici qu'il est conseillé de les déclarer.

C'est également grâce à ces valeurs que la gestion des langues peut se faire, les ressources String en plusieurs dialectes étant stockées ici et accédées selon le langage configuré sur l'appareil utilisé.

### 5.8.3 ANDROIDMANIFEST.XML

Le fichier AndroidManifest.xml est un fichier essentiel à toute application Android. Il sert en fait à identifier l'application et à en établir certaines propriétés.<sup>77</sup> Sa première fonction est de nommer le package de l'application. Il décrit également les différents composants de l'application, les différents services et activités<sup>78</sup> du projet devant y être déclarés. Parmi les informations fournies par ce fichier se trouve aussi la liste des différentes permissions que l'utilisateur doit accorder à l'application afin de pouvoir utiliser toutes les fonctionnalités de celle-ci.

Voici donc le manifeste du projet présenté ici :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.andrew.fictional_login"
    android:versionCode="1"
    android:versionName="1.0">

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

    <uses-sdk
        android:minSdkVersion="15"
```

<sup>76</sup> cf. 5.7.11 MVVM et data binding

<sup>77</sup> (App Manifest | Android Developers, s.d.)

<sup>78</sup> cf. 5.7.4 Activity

```

    android:targetSdkVersion="23" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".view.LoginActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
            />
            </intent-filter>
        </activity>
    </application>

</manifest>

```

Les quelques premières lignes déterminent donc le nom du package, ici « com.andrew.fictional\_login », et la version de l'application, le nom servant d'identifiant au sein de l'appareil :

```

package="com.andrew.fictional_login"
    android:versionCode="1"
    android:versionName="1.0">

```

La ligne suivante détermine les permissions nécessaires à l'application. Celle-ci nécessitant de connecter l'utilisateur, un accès à internet est nécessaire :

```

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

```

La balise « application » présente les attributs tels que le nom, l'icône de lancement à utiliser ou le thème. Les valeurs commençant par « @ » sont en fait les identifiants des constantes stockées dans le module « res ».<sup>79</sup>

---

<sup>79</sup> cf. 5.8.2 Structure

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
```

Enfin, toujours au sein de la balise « application », se situe la liste des différentes activités du projet ainsi que leur fonction. Ici, l’activity « LoginActivity » est, comme la balise « intent-filter » le précise, la première activity à être lancée ainsi que l’activity principale de l’application.

```
<activity
    android:name=".view.LoginActivity"
    android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

#### 5.8.4 LES RESSOURCES

Si le code Java d’une application Android est au cœur du développement, il est indissociable des différentes ressources utilisées. Sous ce terme se cachent en fait les différents layouts et autres valeurs nécessaires au fonctionnement du programme tels que les Strings, ou chaînes de caractères affichées à l’écran, les dimensions ou encore les valeurs hexadécimales des couleurs utiles au style qu’emprunte l’application. Les quelques explications qui suivent se consacrent donc aux ressources les plus fréquemment abordées.

##### 5.8.4.1 STRINGS.XML

Comportant les textes de l’application, ce fichier permet également la gestion d’une application multilingue. C’est en créant un répertoire « value-xx » au sein de strings.xml – xx étant le code de la langue concernée – et en y ajoutant les chaînes de caractères traduites que le développeur permet à l’application d’afficher les textes désirés lors de son lancement.

Le fichier « strings.xml » de fictional\_login se présente de la sorte et contient le titre de l'application et les messages d'indication ou de feedback destinés à l'utilisateur :

```
<resources>
    <string name="app_name">fictional_login</string>

    <string name="prompt_email">Email</string>
    <string name="prompt_password">Password (optional)</string>
    <string name="action_sign_in">Sign in or register</string>
    <string name="action_sign_in_short">Sign in</string>
    <string name="error_invalid_email">This email address is
invalid</string>
    <string name="error_invalid_password">This password is too
short</string>
    <string name="error_incorrect_password">This password is
incorrect</string>
    <string name="error_field_required">This field is required</string>
</resources>
```

#### 5.8.4.2 DIMENS.XML

C'est dans ce fichier que se trouvent les dimensions, en dp ou sp,<sup>80</sup> des différents éléments graphiques de l'application tels que les polices ou les marges internes de l'interface.

```
<resources>
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

#### 5.8.4.3 STYLES.XML

Il est d'usage de devoir respecter une charte graphique lors de la création d'une application mobile. Souvent dictée par le client qui commande cette dernière, cette charte graphique s'incarne lors du développement dans le fichier « styles.xml ». Ainsi, c'est ici que sont définis les thèmes visuels à utiliser.

L'exemple ici exposé se contente du thème par défaut fourni par Android Studio<sup>81</sup>:

---

<sup>80</sup> cf. 5.6 Design adaptatif et le material design

<sup>81</sup> cf. 4.1 Android Studio

```

<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>

```

#### 5.8.4.4 COLORS.XML

Afin de respecter une charte graphique et pour faciliter l'implémentation des thèmes utilisés, il convient de déclarer les valeurs hexadécimales des couleurs dans le fichier « colors.xml ». Il devient ainsi plus aisément de les référencer plus tard dans le code.

Pour fictional\_login, cette ressource sert en effet à la déclaration des couleurs utilisées par le thème par défaut d'Android Studio, à savoir, dans l'ordre, deux tons de bleu et un de rose :

```

<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>

```

#### 5.8.4.5 LES LAYOUTS

Les layouts sont les fichiers au sein desquels sont déclarés en XML la quasi-totalité des éléments graphiques. De façon similaire à la gestion des chaînes de caractères dans différentes langues, il est ici possible d'indiquer plusieurs layouts à sélectionner en fonction des dimensions de l'écran sur lequel s'affiche l'application. Pour ce faire, différents sous-dossiers sont créés avec pour nom « layout-xxx », où xxx représente la densité de pixel du dispositif.<sup>82</sup>

Le layout de la page de connexion de l'exemple est déclaré comme suit :

```

<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable name="user"
            type="com.andrew.fictional_login.viewmodel.UserViewModel" />

```

<sup>82</sup> cf. 5.6 Design adaptatif et le material design

```
</data>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".view.LoginActivity">

    <ProgressBar
        android:id="@+id/login_progress"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:visibility="gone" />

    <ScrollView
        android:id="@+id/login_form"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:id="@+id/email_login_form"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/email"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="@string/prompt_email"
                    android:inputType="textEmailAddress"
                    android:maxLines="1"
                    android:singleLine="true"
                    android:text="@{user.mail}" />

            </android.support.design.widget.TextInputLayout>

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/password"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="@string/prompt_password"
                    android:inputType="textPassword"
                    android:maxLines="1"
                    android:singleLine="true"
                    android:text="@{user.password}" />

            </android.support.design.widget.TextInputLayout>
        </LinearLayout>
    </ScrollView>
</LinearLayout>
```

```

</android.support.design.widget.TextInputLayout>

<Button
    android:id="@+id/email_sign_in_button"
    style="?android:textAppearanceSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="@string/action_sign_in_short"
    android:textStyle="bold" />

    </LinearLayout>
</ScrollView>
</LinearLayout>
</layout>

```

La rédaction d'un tel code étant laborieuse, il est bon de rappeler qu'Android Studio<sup>83</sup> possède un outil d'édition de layouts fonctionnant sur le principe du glisser-déposer et générant le code XML automatiquement.

Si la balise « data » est dédiée au data binding,<sup>84</sup> les autres incarnent soit des vues, les différents composants graphiques d'Android, soit des containers, qui sont des éléments non visuels héritant de la classe ViewGroup et facilitant la disposition de ces vues. Parmi les différents containers, ceux-ci sont les plus régulièrement utilisés<sup>85</sup> :

- **AbsoluteLayout**

Ce layout permet de spécifier la disposition exacte de ses composants en termes de coordonnées x/y.

- **FrameLayout**

FrameLayout est destiné à consacrer une partie de l'écran à un élément spécifique. L'élément en question se place par défaut en haut à gauche de ce layout mais changer sa position peut se faire grâce à l'attribut « android:gravity » qui se charge de gérer son alignement.

---

<sup>83</sup> cf. 4.1 Android Studio

<sup>84</sup> cf. 5.7.11.2 Présentation du data binding

<sup>85</sup> (ViewGroup | Android Developers, s.d.)

### • GridLayout

Comme son nom l'indique, GridLayout facilite la disposition de ses composants sous la forme d'une grille.

### • LinearLayout

Sûrement un des layouts les plus pratiques et donc un des plus utilisés, LinearLayout gère la disposition de ses composants sur une ligne ou une colonne. L'attribut « android:orientation » y sert à indiquer la disposition souhaitée tandis que « android:gravity » permet, comme pour le FrameLayout, d'indiquer l'alignement des éléments en son sein.

### • RelativeLayout

RelativeLayout, un autre layout des plus pratiques, permet d'indiquer la disposition des éléments selon celle des autres.

Parmi les balises restantes au sein du layout de fictional\_login se trouvent « EditText », un champ d'insertion de texte, « Button », un simple bouton, « ProgressBar », soit la vue animée indiquant le chargement, « ScrollView », qui donne la possibilité de faire défiler l'écran, et « TextInputLayout », une vue animée liée aux « EditText » et propre au material design.<sup>86</sup>

Chaque élément doit posséder au moins deux attributs, à savoir « android:layout\_width » servant à indiquer la largeur et « android:layout\_height » pour la hauteur. S'il est possible d'indiquer une valeur en dp,<sup>87</sup> il convient généralement d'utiliser les valeurs « WRAP\_CONTENT » ou « MATCH\_PARENT » servant respectivement à fixer la dimension selon la taille d'un composant ou selon la taille d'un parent.

Il existe encore toutes sortes de vues comme le « TextView » servant à l'affichage de textes fixes ou encore la « CheckBox » – une case à cocher – ainsi que toutes sortes d'attributs. C'est pourquoi les développeurs sont invités à consulter Android Developers, le site de référence de Google destiné aux développeurs Android.<sup>88</sup>

---

<sup>86</sup> cf. 5.6 Design adaptatif et le material design

<sup>87</sup> cf. 5.6 Design adaptatif et le material design

<sup>88</sup> (Android Developers, s.d.)

## 5.8.5 R.JAVA

Le fichier R.java est un fichier automatiquement généré par Android Studio<sup>89</sup> lors de la compilation. Il contient en fait des références vers tous les éléments se trouvant dans le module « res » et c'est grâce à ce fichier qu'il est possible d'accéder aux différentes ressources depuis le code source de l'application.

## 5.8.6 LES FICHIERS SOURCES

Situés dans le dossier « java »,<sup>90</sup> les fichiers sources concernent l'implémentation des services, des activités et des méthodes de l'application. Généralement agencés par classe, ces fichiers sont bien entendu ceux qui nécessitent le plus d'attention lors du développement de l'application et qui sont les plus signifiants dans son fonctionnement.

Le projet fictional\_login ne contient qu'une seule classe, LoginActivity, dont ce qui suit est l'implémentation, partiellement générée par Android Studio – qui permet de faciliter la création d'une activity dédiée à l'identification – et adaptée en vue de servir ici d'exemple :

```
package com.andrew.fictional_login.view;

import ...

public class LoginActivity extends AppCompatActivity {

    private UserLoginTask mAuthTask = null;

    private EditText mEmailView;
    private EditText mPasswordView;
    private View mProgressView;
    private View mLoginFormView;

    private ActivityLoginBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = DataBindingUtil.setContentView(this,
R.layout.activity_login);
        binding.setUser(new UserViewModel("mailExample@example.com",
"passwordExample"));

        mEmailView = binding.email;
        mPasswordView = binding.password;
```

<sup>89</sup> cf. 4.1 Android Studio

<sup>90</sup> cf. 5.8.2 Structure

```

        Button mEmailSignInButton = binding.emailSignInButton;
        mEmailSignInButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                attemptLogin();
            }
        });

        mLoginFormView = binding.loginForm;
        mProgressView = binding.loginProgress;
    }

    private void attemptLogin() {
        if (mAuthTask != null) {
            return;
        }

        mEmailView.setError(null);
        mPasswordView.setError(null);

        String email = binding.getUser().getMail();
        String password = binding.getUser().getPassword();

        boolean cancel = false;
        View focusView = null;

        if (!TextUtils.isEmpty(password) && !isValidPassword(password)) {
            mPasswordView.setError(getString(R.string.error_invalid_password));
            focusView = mPasswordView;
            cancel = true;
        }

        if (TextUtils.isEmpty(email)) {
            mEmailView.setError(getString(R.string.error_field_required));
            focusView = mEmailView;
            cancel = true;
        } else if (!isValidEmail(email)) {
            mEmailView.setError(getString(R.string.error_invalid_email));
            focusView = mEmailView;
            cancel = true;
        }

        if (cancel) {
            focusView.requestFocus();
        } else {
            showProgress(true);
            mAuthTask = new UserLoginTask(email, password);
            mAuthTask.execute((Void) null);
        }
    }

    private boolean isValidEmail(String email) {
        return email.contains("@");
    }

    private boolean isValidPassword(String password) {
        return password.length() > 4;
    }
}

```

```

    }

    @TargetApi (Build.VERSION_CODES.HONEYCOMB_MR2)
    private void showProgress(final boolean show) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB_MR2) {
            int shortAnimTime =
getResources().getInteger(android.R.integer.config_shortAnimTime);

            mLoginFormView.setVisibility(show ? View.GONE :
View.VISIBLE);
            mLoginFormView.animate().setDuration(shortAnimTime).alpha(
                show ? 0 : 1).setListener(new
AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    mLoginFormView.setVisibility(show ? View.GONE :
View.VISIBLE);
                }
            });
        }

        mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
        mProgressView.animate().setDuration(shortAnimTime).alpha(
            show ? 1 : 0).setListener(new
AnimatorListenerAdapter() {
                @Override
                public void onAnimationEnd(Animator animation) {
                    mProgressView.setVisibility(show ? View.VISIBLE :
View.GONE);
                }
            });
    } else {
        mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
        mLoginFormView.setVisibility(show ? View.GONE :
View.VISIBLE);
    }
}

public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

    private final String mEmail;
    private final String mPassword;

    UserLoginTask(String email, String password) {
        mEmail = email;
        mPassword = password;
    }

    @Override
    protected Boolean doInBackground(Void... params) {
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            return false;
        }
        return true;
    }

    @Override
    protected void onPostExecute(final Boolean success) {
        mAuthTask = null;
    }
}

```

```
        showProgress(false) ;

        if (success) {
            finish();
        } else {

mPasswordView.setError(getString(R.string.error_incorrect_password));
        mPasswordView.requestFocus();
    }
}

@Override
protected void onCancelled() {
    mAAuthTask = null;
    showProgress(false);
}
}
```

LoginActivity étend AppCompatActivity, une classe de la Support Library<sup>91</sup> étendant elle-même la classe Activity.<sup>92</sup> La méthode onCreate()<sup>93</sup> y est redéfinie afin d'y effectuer le binding<sup>94</sup> pour ensuite initialiser les composants et les variables. C'est donc dans cette méthode qu'est décrit le contenu de la vue.

Le bouton (cf. Figure 33 : activity principale de fictional\_login) est déclaré avec un écouteur d'évènements de sorte qu'un clic sur celui-ci exécute la méthode attemptLogin() qui se charge de récupérer l'adresse mail et le mot de passe rentrés par l'utilisateur grâce au data binding, de vérifier s'ils sont valides grâce à isPasswordValid() et à isEmailValid() et de lancer une tâche asynchrone simulant une tentative de connexion et affichant une animation de chargement.

La méthode `isEmailValid()` vérifie que l'adresse mail entrée par l'utilisateur contient un « @ » et `isPasswordValid()` vérifie que le mot de passe est d'une longueur de plus de quatre caractères.

La tâche asynchrone consiste à mettre l'application en pause pendant dix-mille millisecondes, le temps pour une animation de chargement (cf. Figure 34 : chargement de fictional\_login) de se produire. À la fin de ces dix-mille millisecondes, l'application s'arrête.

Le cas où l'adresse mail et le mot de passe ne correspondent pas – tel que déclaré dans `onCreate()` afin de simuler la base de données à distance – est également pris en compte mais cette application étant fictive, cela n'a aucun effet sur l'interface.

<sup>91</sup> cf. 5.7.10 Les bibliothèques de compatibilité

<sup>92</sup> cf. 5.7.4 Activity

<sup>93</sup> cf. 5.7.4.1 Le cycle de vie d'une activity

<sup>94</sup> cf. 5.7.11.2 Présentation du data binding

## 6 PRÉSENTATION DU PROJET

Les deux premières semaines de ce stage furent dédiées à un exercice de formation, à savoir la réalisation d'une application mobile fonctionnelle. Les treize semaines restantes étaient quant à elles centrées sur le développement d'une autre application, Ethias Prevention. Cette commande de la part d'Ethias est destinée à assister ses employés assureurs sur le terrain. Ainsi, un agent peut se rendre chez un client et, sans aucun besoin de carnet ou de stylo, rédiger des rapports de description des dégâts, de dépistage des risques, d'état des lieux, des rapports d'incident ou encore différents types d'audits, directement sur l'application qui se charge ensuite de la centralisation.

L'application se présente comme suit : l'utilisateur arrive au démarrage sur une page de connexion aussi pourvue d'un menu pour le choix de la langue et d'une solution en cas d'oubli du mot de passe. Cette connexion se fait selon des données déjà présentes chez Ethias et amène à la page d'accueil. Sur ce deuxième écran se trouve une liste des rapports en cours et en attente ainsi qu'une représentation visuelle – un tableau et une carte de la Belgique – de cette liste. Depuis cette page, l'agent peut directement accéder à la rédaction d'un nouveau rapport ainsi qu'à deux onglets : « rapport » et « historique ». La page « rapport » consiste en un formulaire où peuvent être renseignés le nom et la référence du dossier, le nom du client, la date de visite, le type de mission, le lieu – accessible depuis un widget Google Maps, la liste des participants et la liste des destinataires. Ces rapports peuvent être ici enrichis de sections dont l'agent décide la nature et auxquelles il peut adjoindre des pictogrammes, des législations, un niveau de risque et de priorité ainsi que des médias de type vidéo, photo ou audio. Enfin, il peut également noter des observations et des recommandations avant de soumettre le rapport. La dernière page, l'historique, consiste en une liste des rapports finis qu'il est possible d'imprimer ou d'envoyer par mail.

La mission consiste donc à reprendre le maigre existant et à le continuer, voire le terminer, en l'adaptant aux nouvelles technologies Android, à savoir la librairie Data Binding soumise il y a peu par Google ainsi que le design pattern lié, le MVVM, ou Model View ViewModel.<sup>95</sup>

---

<sup>95</sup> cf. 5.7.11 MVVM et data binding

## 6.1 ANALYSE

Les grandes lignes de l'application sont tracées. Néanmoins, un tel travail nécessite une attention élevée afin de ne pas se lancer dans un développement interminable et rempli d'embuches. C'est pourquoi cette partie est dédiée à l'analyse du fonctionnement désiré de l'application. Parmi les éléments détaillés se trouvent les différentes fonctionnalités de l'application sous forme de cas d'utilisation, l'évolution de l'état d'un rapport depuis sa création à sa finalisation, les différentes possibilités de navigation au sein de l'application et les besoins non fonctionnels de celle-ci, c'est-à-dire ce qui concerne moins ses fonctionnalités et ses objets métier que sa forme et son ergonomie.

### 6.1.1 DIAGRAMME D'ACTIVITÉ

Le diagramme d'activité consiste en une représentation concise des différentes activités et de leur déroulement ainsi que de l'évolution de l'objet métier, le rapport. Il permet également de faciliter l'analyse des différents cas d'utilisation.

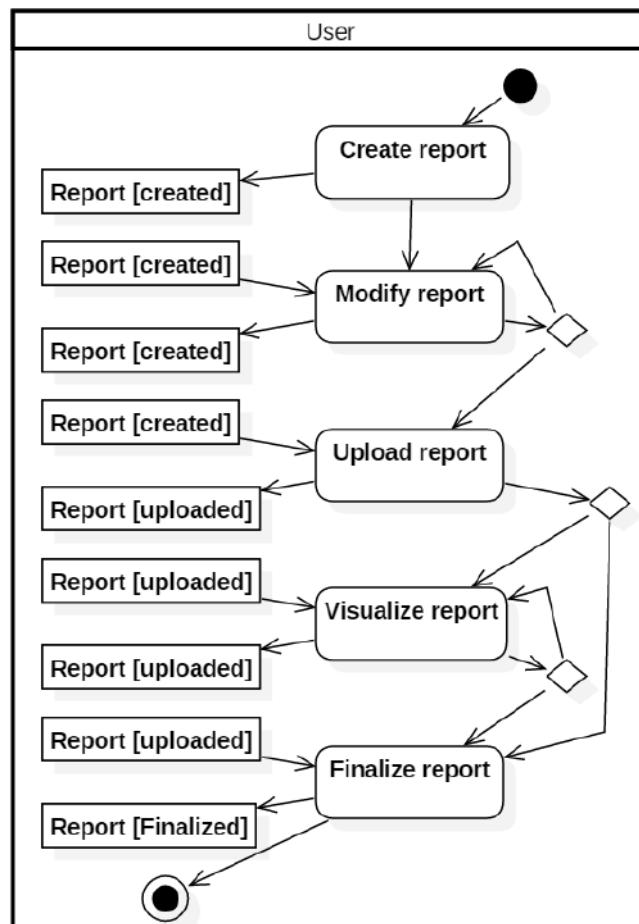


Figure 36 : diagramme d'activité

## 6.1.2 DIAGRAMME D'ÉTAT

Le diagramme d'état présenté ici aborde l'évolution de l'état d'un rapport en fonction du déroulement du processus de finalisation de ce dernier.

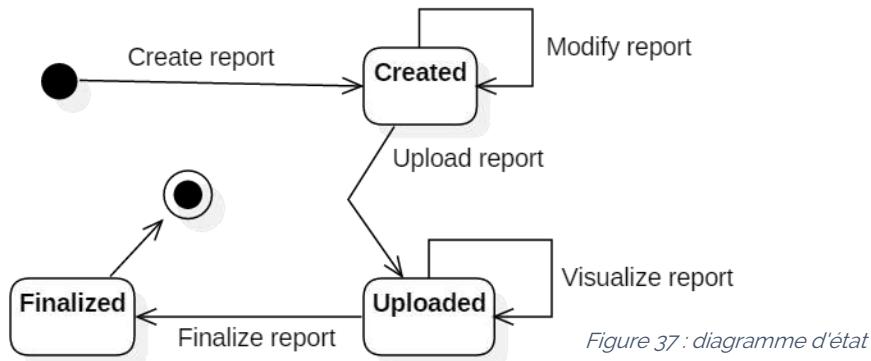


Figure 37 : diagramme d'état

## 6.1.3 DIAGRAMME DES CAS D'UTILISATION

Cette partie concerne les différents cas d'utilisation extraits du diagramme d'activité. Les cinq étapes propres à la réalisation d'un rapport, de la création à la finalisation, sont donc ici présentées en détail.

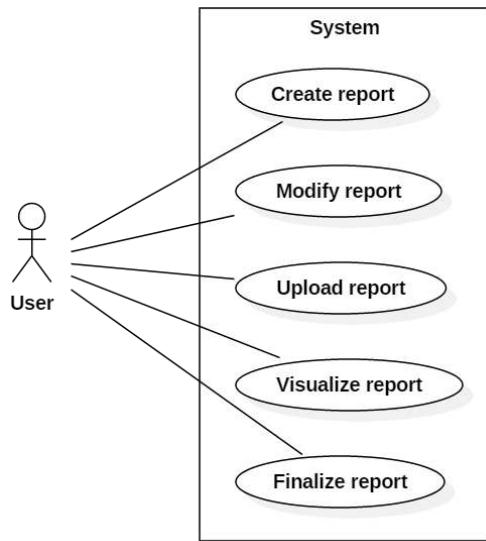


Figure 38 : diagramme des cas d'utilisation

### 6.1.3.1 CAS 1 : CRÉER LE RAPPORT

Le cas d'utilisation commence dès lors que l'utilisateur souhaite créer un nouveau rapport. Il se rend donc sur la page de création d'un rapport, complète les champs<sup>96</sup> et enregistre ensuite ce dernier. À la fin de ce cas, l'état du rapport est « créé » ou « en cours ».

<sup>96</sup> cf. 6.2.4 Écrans de création de rapports

#### 6.1.3.2 CAS 2 : MODIFIER LE RAPPORT

Accessible depuis la page d'accueil, la liste des rapports en cours – et donc sauvegardés seulement sur l'appareil pour l'instant – permet la modification de ceux-ci. Il suffit alors à l'utilisateur de cliquer sur le bouton d'édition de l'élément à modifier dans la liste afin de revenir sur l'écran de création de rapports, les champs étant déjà complétés selon la dernière sauvegarde. Il peut ainsi modifier les éléments souhaités et sauvegarder le rapport. Ce cas d'utilisation peut être répété à l'envi et ne modifie pas l'état du rapport.

#### 6.1.3.3 CAS 3 : POSTER LE RAPPORT

Une fois le rapport suffisamment complet, l'utilisateur peut le poster dans la base de données afin que celui-ci soit accessible à distance depuis d'autres dispositifs. Pour ce faire, il suffit à l'utilisateur de cliquer sur le bouton d'upload du rapport dans la liste des rapports en cours. Une fois uploadé, le compte rendu n'est plus modifiable que depuis le backend, c'est-à-dire l'outil web associé à l'application permettant d'accéder à la base de données. L'état du rapport passe alors à « en attente » ce qui est reflété par l'interface.

#### 6.1.3.4 CAS 4 : VISUALISER LE RAPPORT

Désormais dans la liste des rapports en attente, toujours accessible depuis la page d'accueil, le rapport peut être visualisé par l'utilisateur. Pour ce faire, il suffit à l'utilisateur de cliquer sur le rapport dans la liste afin de revenir sur l'écran de création de rapports mais cette fois-ci dépourvu de la possibilité de modifier les champs. La visualisation est également possible depuis l'historique des rapports finalisés<sup>97</sup> et peut être répétée à l'envi.

#### 6.1.3.5 CAS 5 : FINALISER LE RAPPORT

Toujours dans la liste des éléments en attente, le rapport peut également être finalisé. L'utilisateur n'a qu'à cliquer sur le bouton adéquat afin de modifier l'état du rapport ce qui a pour effet de le voir se déplacer dans l'historique des rapports et de voir son état passer à « finalisé ».

---

<sup>97</sup> cf. 6.2.5 Historique des rapports

## 6.1.4 DIAGRAMME DES CLASSES PERSISTANTES

Le diagramme des classes persistantes ci-dessous rappelle les différents types d'entités concernés par l'application ainsi que leurs arguments.

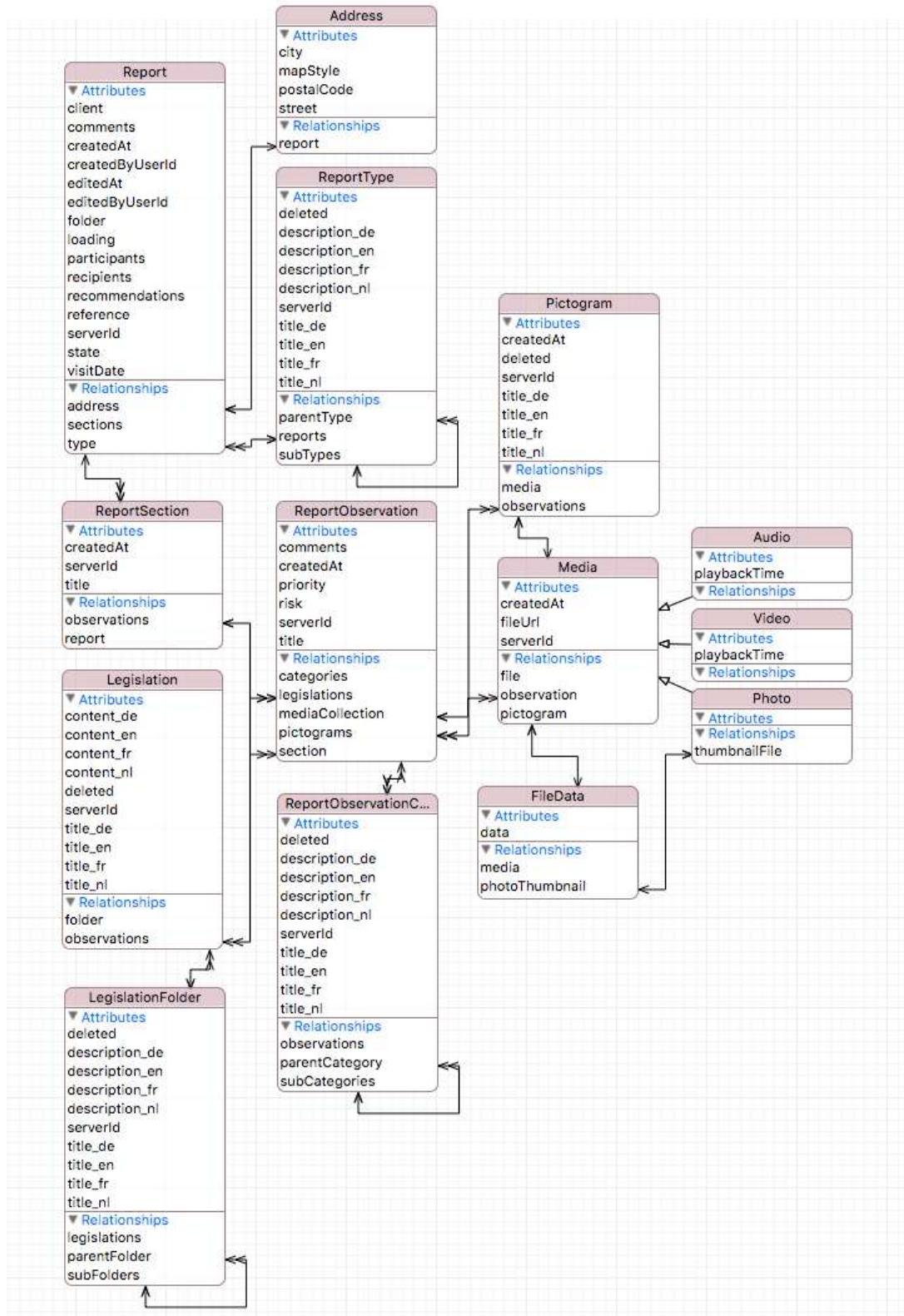


Figure 39 : diagramme des classes persistantes, généré grâce au programme Core Data pour la version iOS d'Ethias Prevention

### 6.1.5 DIAGRAMME DE NAVIGATION

Le diagramme de navigation suivant permet de représenter le fonctionnement de l'application en illustrant les différentes possibilités de navigation qui sont offertes à l'utilisateur.

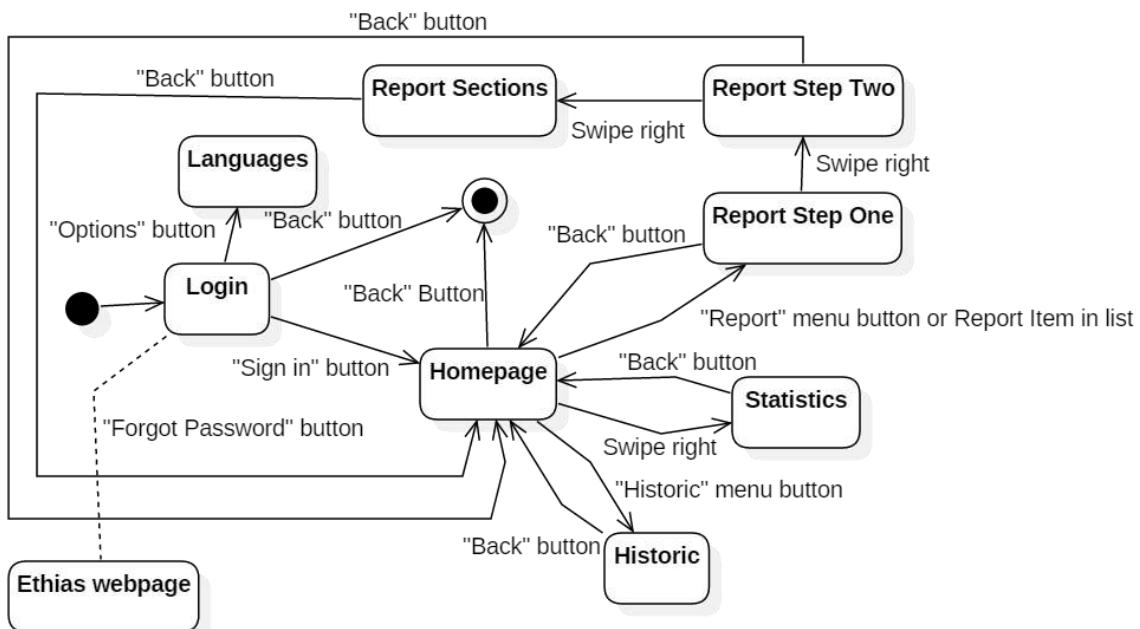


Figure 40 : diagramme de navigation

## 6.1.6 BESOINS NON FONCTIONNELS

Ethias Prevention est une application mobile. Dès lors, elle doit pouvoir s'adapter à plusieurs tailles d'écran et son utilisation doit être ergonomique et intuitive.<sup>98</sup> Ce point sous-entend également la fluidité, la lisibilité, l'efficacité et la praticité du programme.

Elle doit également être utilisable en cas de dysfonctionnement de la connexion internet, une copie de la base de données étant stockée sur le dispositif.<sup>99</sup> Développée selon le design pattern MVVM,<sup>100</sup> l'application doit être facilement modifiable et testable.

Enfin, elle doit être disponible facilement, c'est-à-dire téléchargeable via Google Play.<sup>101</sup>

<sup>98</sup> cf. 5.6 Design adaptatif et le material design

<sup>99</sup> cf. 6.3.2 Realm pour la persistance des données

<sup>100</sup> cf. 5.7.11 MVVM et data binding

<sup>101</sup> cf. 5.3 Google Play

## 6.2 VISITE GUIDÉE D'ETHIAS PREVENTION

L'analyse du projet étant effectuée, il est maintenant opportun de s'intéresser au produit fini. Cette partie se consacre donc à la visite guidée de l'application, enrichie de captures d'écran, d'explications sur le développement ainsi que de commentaires justifiant ou critiquant l'absence ou la présence de certaines fonctionnalités.

### 6.2.1 ÉCRAN DE CONNEXION



L'écran de connexion est la porte d'entrée de l'application. Chaque utilisateur doit se connecter avant de pouvoir user des différentes fonctionnalités d'Ethias Prevention. Si cette page de connexion est déjà fonctionnelle et identifie sans problèmes un utilisateur à qui Ethias aurait fourni une adresse mail et un mot de passe adéquat, elle n'est cependant pas terminée.

En effet, l'option « Mot de passe oublié ? » n'a pour l'instant aucun effet, son implémentation nécessitant l'intervention d'Ethias. Le bouton « outil » en haut à droite qui donne accès au choix des langues est quant à lui déjà effectif. Cependant, les ressources en plusieurs langues n'ont pas encore été traduites.

Aussi, une telle page devrait être capable de mémoriser les précédentes valeurs entrées par l'utilisateur et de proposer l'adresse mail liée au téléphone lorsque celui-ci clique sur le champ « E-mail », ce qui n'est pas le cas. Idéalement, cette page ne devrait même pas s'afficher à chaque démarrage de l'application car l'utilisateur ne devrait pas avoir à se reconnecter à chaque fois mais cette fonctionnalité reste à implémenter.

Figure 41 : capture d'écran de la page de connexion

Enfin, le bouton « Connexion » lance une requête d'identification grâce à un service web<sup>102</sup> et charge ensuite les informations de la base de données<sup>103</sup> afin de permettre à l'utilisateur de travailler avec une connexion internet limitée. Après ces opérations, la page d'accueil de l'application<sup>104</sup> s'affiche.

<sup>102</sup> cf. 6.3.1 Retrofit pour les services web

<sup>103</sup> cf. 6.3.2 Realm pour la persistance des données

<sup>104</sup> cf. 6.2.2 Écran d'accueil

## 6.2.2 ÉCRAN D'ACCUEIL

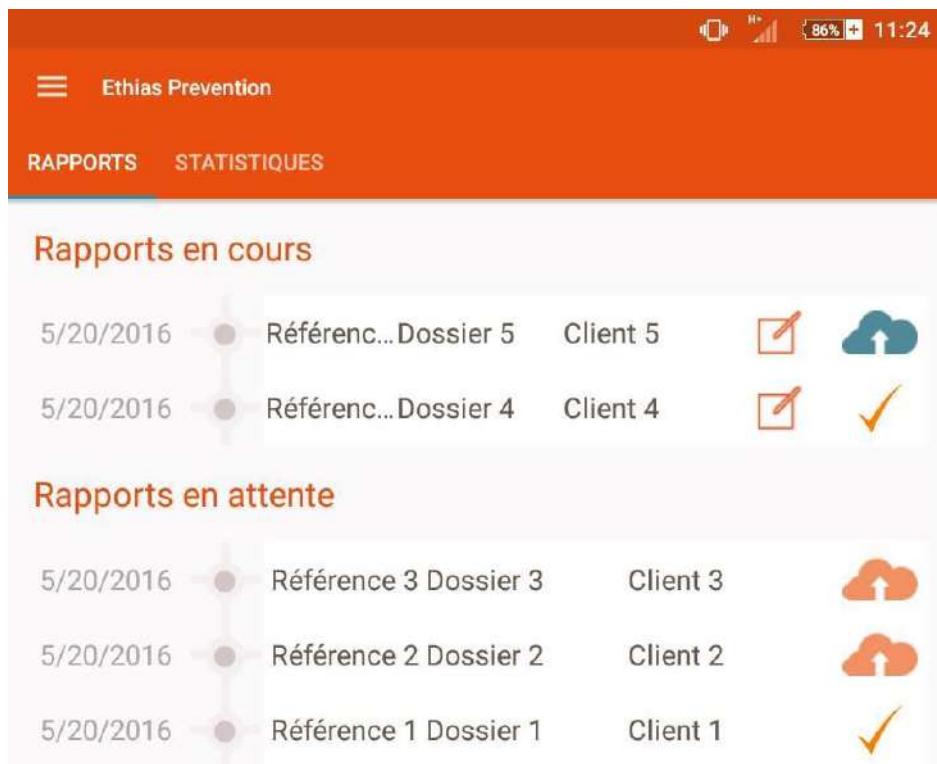


Figure 42 : capture d'écran de la page d'accueil

La page d'accueil, ici affichée dans sa disposition horizontale afin de voir un maximum d'informations, a pour but de rappeler à l'utilisateur les rapports en cours et ceux en attente. Ces rapports ont été chargés depuis la base de données locale et sont affichés dans une liste de type « RecyclerView ». Une telle liste est en fait un layout<sup>105</sup> que propose Android et qui permet aux développeurs d'agencer des éléments comme ils le souhaitent au moyen d'un adapter.<sup>106</sup> Cette liste est spécialement conçue pour optimiser l'usage de la mémoire en ne chargeant que ce qui est affiché à l'écran et en recyclant les items en de nouveaux éléments une fois que ceux-ci disparaissent de la vue de l'utilisateur.

Les chaines de caractères « Rapports en cours » et « Rapports en attente » sont les en-têtes de cette liste mais ils défilent pour l'instant avec la liste lorsque celle-ci défile par l'action de l'utilisateur. Idéalement, ceux-ci devraient rester affichés à l'écran tandis que l'utilisateur ferait dérouler la liste par dessous eux. Cependant, cette fonctionnalité, appelée « Sticky Header », reste à implémenter.

L'icône en forme de note et de crayon sert, comme son apparence le suggère, à modifier le rapport en cours. Cliquer sur ce bouton a donc pour effet d'ouvrir la page

<sup>105</sup> cf. 5.8.4.5 Les layouts

<sup>106</sup> cf. 5.7.9 Adapter

de création de rapports<sup>107</sup> garnie au préalable des différentes valeurs propres au rapport sélectionné. Du point de vue implémentation, lorsque l'utilisateur clique sur cette icône, il crée en fait un intent<sup>108</sup> dans lequel sont entreposées ces valeurs – récupérées dans la base de données locale<sup>109</sup> – ainsi que le nom de l'activité à démarrer, à savoir celle de création de rapports. Il enclenche ainsi la fin de l'activité courante et est finalement redirigé sur l'écran adéquat.

Le bouton à droite de celui-ci, en forme de nuage bleu, sert à uploader le rapport sur la base de données au moyen d'un service web.<sup>110</sup> Dans les faits, cette action ne fait que modifier un argument – l'état du rapport – et la réussite de l'opération est indiquée à l'utilisateur au moyen d'une icône de confirmation en forme de « V ». Après l'upload, au rechargeement de la page, le rapport n'est plus présent dans la liste des rapports en cours mais dans celle des rapports en attente.

Enfin, le bouton en forme de nuage orange sert à changer l'état du rapport, encore une fois au moyen d'un service web, de sorte que celui-ci soit indiqué comme finalisé. Dès lors, au rechargeement de la page, le rapport disparaît de la liste des rapports en cours et va se ranger dans l'historique des rapports.<sup>111</sup>

L'onglet « Statistiques » en haut de l'écran est censé afficher une représentation graphique des différentes statistiques des rapports. Celle-ci devrait se faire sous la forme d'une carte de la Belgique qui affiche des ronds à certains endroits dont la taille dépend du nombre de rapports effectués en ces lieux et d'un graphe représentant le nombre de rapports créés selon la date. Cependant, cette partie, plutôt technique et très graphique, reste à implémenter.

---

<sup>107</sup> cf. 6.2.4 Écrans de création de rapports

<sup>108</sup> cf. 5.7.7 Intent

<sup>109</sup> cf. 6.3.2 Realm pour la persistance des données

<sup>110</sup> cf. 6.3.1 Retrofit pour les services web

<sup>111</sup> cf. 6.2.5 Historique des rapports

### 6.2.3 MENU

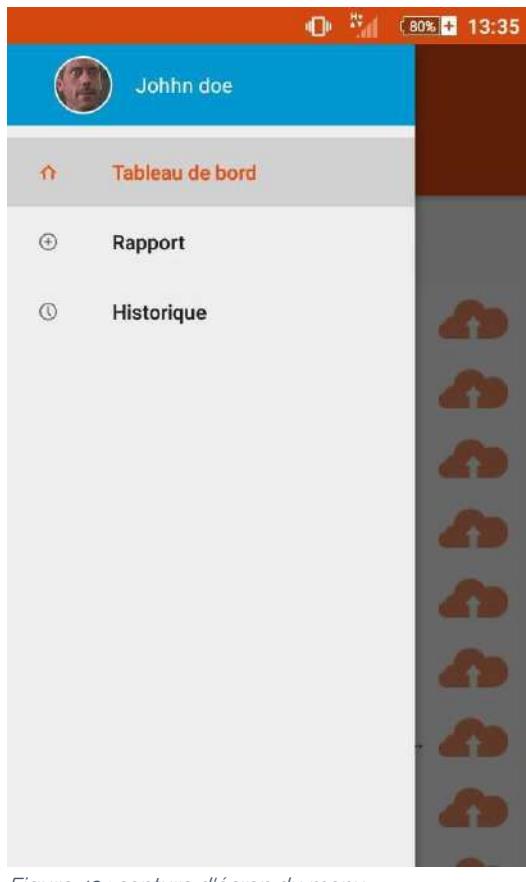


Figure 43 : capture d'écran du menu

Le menu, se présentant sous la forme d'un « Navigation Drawer », est accessible depuis l'écran d'accueil,<sup>112</sup> l'écran de création de rapports<sup>113</sup> et l'écran présentant l'historique des rapports<sup>114</sup> par le biais d'un swipe<sup>115</sup> de l'extrême gauche de l'écran vers la droite ou grâce à un clic sur le bouton « Hamburger » de l'application, une icône composée de trois lignes horizontales référencée dans le guide de Google sur le material design.<sup>116</sup>

Ce menu coulissant s'implémente grâce à une balise « DrawerLayout » en XML<sup>117</sup> qui englobe le contenu de la page et au sein duquel peut être implémentée une balise « NavigationView », représentant un menu de navigation standard. Cette dernière peut être enrichie d'une balise « menu » qui sert à agencer les éléments tel que souhaité.

Afin d'illustrer ces explications, voici le layout lié au « Navigation Drawer » :

```
<android.support.v4.widget.DrawerLayout
    android:id="@+id/activity_base_drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <include
        android:id="@+id/activity_base_nav_content_fragment"
        layout="@layout/activity_base_nav_content_fragment" />

    <android.support.design.widget.NavigationView
        android:id="@+id/activity_base_navigation_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/menu_toolbar"
        app:menu="@menu/menu_navigation_view" />
```

<sup>112</sup> cf. 6.2.2 Écran d'accueil

<sup>113</sup> cf. 6.2.4 Écrans de création de rapports

<sup>114</sup> cf. 6.2.5 Historique des rapports

<sup>115</sup> cf. 5.7.8 Gestures

<sup>116</sup> cf. 5.6 Design adaptatif et le material design

<sup>117</sup> cf. 5.8.4.5 Les layouts

Et voici la ressource référencée dans ce layout qui sert à agencer le menu :

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/menu_navigation_view_dashboard"
            android:icon="@drawable/ep_menu_dashboard"
            android:title="@string/menu_dashboard" />
        <item
            android:id="@+id/menu_navigation_view_report"
            android:icon="@drawable/ep_menu_rapport"
            android:title="@string/menu_rapport" />
        <item
            android:id="@+id/menu_navigation_view_historic"
            android:icon="@drawable/ep_menu_history"
            android:title="@string/menu_historique" />
    </group>
</menu>
```

Les balises précédemment citées ainsi que les trois éléments du menu y sont bien présentes.

Dès que ceci est rédigé, il suffit d'indiquer dans le code source<sup>118</sup> de la classe concernée qu'elle implémente « OnNavigationItemSelectedListener » et d'y redéfinir la méthode « onNavigationItemSelected() » afin d'obtenir une référence de l'item sélectionné par l'utilisateur. Cette référence peut alors être utilisée pour charger la page correspondante. Dans la plupart des activités<sup>119</sup> d'Ethias Prevention, ce listener est implémenté comme suit, « startActivityFromNavigationView() » étant une méthode qui se charge de lancer l'activité correspondant à la référence reçue :

```
@Override
public boolean onNavigationItemSelected(MenuItem menuItem) {
    if (!super.onNavigationItemSelected(menuItem)) {
        NavigationController.startActivityFromNavigationView(this,
menuItem.getItemId());
        return true;
    }
    return false;
}
```

<sup>118</sup> cf. 5.8.6 Les fichiers sources

<sup>119</sup> cf. 5.7.4 Activity

#### 6.2.4 ÉCRANS DE CRÉATION DE RAPPORTS

Les écrans de création de rapports sont au cœur d’Ethias Prevention étant donné que les activités<sup>120</sup> qui leur sont liées sont la raison première pour laquelle Ethias a commandé cette application. Cette création de rapports se déroule en trois parties et donc sur au moins trois écrans. Une fois ces trois pages – ou plus – visitées, l’utilisateur peut sauvegarder le rapport dans la base de données locale<sup>121</sup> en cliquant sur le bouton de confirmation en forme de « V » présent en haut à droite des écrans. Cet enregistrement a pour effet de passer l’état du rapport à « créé », ce qui peut s’assimiler à « en cours » vu que c’est dans une liste sous ce nom<sup>122</sup> que le rapport va se ranger une fois créé.

Il est bon de noter que l’utilisateur peut également modifier un rapport en cours. Dans ce cas, c’est cette page de création qui s’affiche avec les champs déjà remplis selon les informations présentes dans la base de données locale. S’il veut visualiser un rapport en attente ou finalisé, c’est encore une fois l’écran de création qui apparaît mais dépourvu de la possibilité de modifier les champs.

---

<sup>120</sup> cf. 6.1.1 Diagramme d’activité

<sup>121</sup> cf. 6.3.2 Realm pour la persistance des données

<sup>122</sup> cf. 6.2.2 Écran d’accueil

#### 6.2.4.1 PREMIÈRE ÉTAPE, LES DONNÉES GÉNÉRALES

Figure 44 : capture de l'écran "Données Générales"

La page des données générales est celle qui sert à l'utilisateur à entrer le nom du dossier et sa référence, le nom du client, la date de la visite, l'adresse où celle-ci se déroule, le type de mission effectuée ainsi que la liste des personnes participantes et des personnes à qui est destiné le rapport.

Si la plupart des informations sont introduites dans des champs classiques au sein desquels il suffit d'introduire du texte, il y a quelques informations que le système peut aider à compléter.

Cliquer sur le champ de la date, déjà garni de la date du jour par défaut, ouvre un dialogue assistant l'utilisateur quant au choix de celle-ci. Ce dialogue se nomme « *DatePickerDialog* » et consiste en un écran fourni par Android déroulant les jours, les mois et les années qu'il est possible de sélectionner. Cet écran est facilement intégrable car pris en compte par Android Studio<sup>123</sup> et il suffit, pour profiter de ses fonctionnalités, de l'appeler lorsque l'utilisateur interagit avec le champ concerné.

Le type de mission est sélectionné dans un « *Spinner* », un layout<sup>124</sup> déroulable sous forme de liste mis à disposition par Android Studio qu'il suffit de lier<sup>125</sup> aux informations de la base de données locale<sup>126</sup> au moyen d'un adapter<sup>127</sup> pour en fournir le contenu.

Enfin, s'il est possible d'insérer manuellement l'adresse d'intervention, l'utilisateur peut également laisser le widget Google Maps faire le travail. En exécutant un long click<sup>128</sup> sur un endroit de la carte – ce qui a ici pour effet d'y afficher un marqueur – ou

<sup>123</sup> cf. 4.1 Android Studio

<sup>124</sup> cf. 5.8.4.5 Les layouts

<sup>125</sup> cf. 5.7.11.2 Présentation du data binding

<sup>126</sup> cf. 6.3.2 Realm pour la persistance des données

<sup>127</sup> cf. 5.7.9 Adapter

<sup>128</sup> cf. 5.7.8 Gestures

en cliquant sur le bouton de géolocalisation disponible en haut à droite de cette dernière, le widget se charge automatiquement de compléter les champs de la rue, du code postal et de la ville avec les valeurs correspondantes à l'endroit sélectionné.

Bien qu'étant une fonctionnalité commune au sein des applications Android, l'intégration d'un widget Google Maps n'est pas des plus simples. La première étape de l'implémentation, une fois la vue destinée à accueillir la carte créée, est d'obtenir une clé d'API Google Maps.<sup>129</sup> Pour ce faire, il faut se rendre sur la page dédiée du site Google Developers<sup>130</sup> et suivre les étapes qui y sont décrites afin d'inscrire le projet dans la console des développeurs Google et de recevoir la clé. Ces étapes se résument comme suit :

- **Obtenir les informations sur le certificat de l'application**

Il faut d'abord obtenir les informations sur le certificat de l'application – consistant en une emprunte SHA-1, SHA-1 étant une méthode de hachage cryptographique – généré par l'Android SDK<sup>131</sup> que le développeur peut obtenir en entrant une ligne dans la console Windows. Pour la version debug de l'application, c'est-à-dire la version de développement, la ligne en question est la suivante :

```
keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android
```

Dans le cas de la version de production, soit la version accessible aux utilisateurs, c'est la ligne suivante qu'il faut taper :

```
keytool -list -v -keystore your_keystore_name -alias your_alias_name
```

Où « your\_keystore\_name » est le chemin du fichier keystore obtenu lors de la publication de l'application sur Google Play.<sup>132</sup>

<sup>129</sup> (Premiers pas | Google Maps Android API | Google Developers, 2016)

<sup>130</sup> (Signature et clés d'API | Google Maps Android API | Google Developers, 2016)

<sup>131</sup> cf. 5.4 Le Software Development Kit d'Android

<sup>132</sup> cf. 5.3 Google Play

- **Créer un projet d'API dans la console des développeurs Google**

Pour créer un projet d'API, il faut se rendre sur la console des développeurs Google,<sup>133</sup> créer un nouveau projet au nom de l'application et enfin activer l'API « Google Maps Android API ».

- **Obtenir la clé d'API Android**

Dans cette même console des développeurs, pour obtenir la clé d'API nécessaire au widget Google Maps, il faut d'abord la générer. Pour ce faire, la console invite les développeurs à entrer la clé de certificat de leur application obtenue précédemment ainsi que le nom de son package. Dès lors, une clé d'API est créée.

- **Ajouter la clé d'API à l'application**

Enfin, si tout s'est bien déroulé, il est maintenant nécessaire d'enregistrer cette clé dans le fichier `AndroidManifest.xml`<sup>134</sup> de l'application grâce aux lignes suivantes :

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="YOUR_API_KEY"/>
```

Voilà pour ce qui est de la partie concernant la clé d'API. L'implémentation de la carte, quant à elle, se fait comme suit dans le layout<sup>135</sup> de l'activity<sup>136</sup> concernée :

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/map"
    tools:context=".MapsActivity"
    android:name="com.google.android.gms.maps.SupportMapFragment" />
```

Et comme ceci dans la classe de cette dernière :

---

<sup>133</sup> (Bibliothèque d'API, s.d.)

<sup>134</sup> cf. 5.8.3 `AndroidManifest.xml`

<sup>135</sup> cf. 5.8.4.5 Les layouts

<sup>136</sup> cf. 5.7.4 Activity

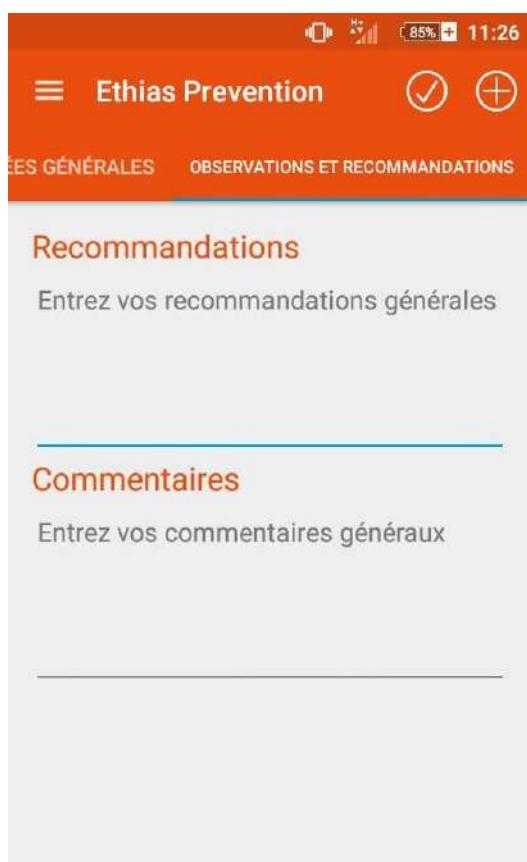
```

public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap map) {
        // Add a marker in Sydney, Australia, and move the camera.
        LatLng sydney = new LatLng(-34, 151);
        map.addMarker(new MarkerOptions().position(sydney).title("Marker
in Sydney"));
        map.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}

```

#### 6.2.4.2 DEUXIÈME ÉTAPE, LES OBSERVATIONS ET LES RECOMMANDATIONS



La page d'observations et de recommandations, on ne peut plus simple, ne se compose que de deux champs d'insertion de texte dédiés aux différents commentaires et recommandations que l'agent pourrait juger utile d'ajouter au rapport.

Figure 45 : capture de l'écran "Observations et recommandations"

### 6.2.4.3 TROISIÈME ÉTAPE, LES SECTION



Figure 46 : capture d'écran d'une nouvelle section

décrivant le risque ainsi que les législations concernées et différents médias tels que des photos, des vidéos ou des enregistrements audio pris directement depuis le smartphone mais même si le champ de certains de ces éléments est prévu, l'implémentation de ces possibilités reste à terminer.

Les observations sont ajoutées dynamiquement dans une liste de type « RecyclerView » – déjà abordée lors de la présentation de l'écran d'accueil.<sup>140</sup> Si leur sujet et leur description s'ajoutent bien entendu par le biais d'un champ d'insertion de textes, les types – tout comme ça devrait être le cas pour les pictogrammes et les

Un agent peut ajouter autant de sections qu'il le souhaite à un rapport au moyen du bouton en forme de « + » situé en haut à droite de l'écran. Ce faisant, elles s'ajoutent grâce à un adapter<sup>137</sup> en tant que nouvel onglet au sein d'un « TabLayout », c'est-à-dire un layout permettant de naviguer de page en page en swipant<sup>138</sup> ou en sélectionnant le titre de l'onglet à afficher. Celles-ci peuvent, par exemple, correspondre à une pièce du lieu d'intervention en particulier ou encore à une machine défectueuse et dangereuse.

Ces sections se composent d'un titre et d'un nombre non défini d'observations pouvant correspondre aux parties de ces pièces ou de ces machines. Une fois encore, l'agent peut en ajouter autant qu'il le souhaite au moyen d'un « FloatingActionButton », un bouton flottant décrit dans le guide sur le material design de Google<sup>139</sup> et situé ici en bas à droite de l'écran.

Ces observations consistent en un sujet, une sélection de types d'observations, une description et une estimation du niveau de risque et de priorité. Il devrait normalement être possible d'y joindre des pictogrammes

<sup>137</sup> cf. 5.7.9 Adapter

<sup>138</sup> cf. 5.7.8 Gestures

<sup>139</sup> cf. 5.6 Design adaptatif et le material design

<sup>140</sup> cf. 6.2.2 Écran d'accueil

législations – s'ajoutent au moyen d'un « CustomSpinner », c'est-à-dire un « Spinner » modifié de sorte que la sélection de plusieurs items à la fois soit possible. Ce « CustomSpinner » est lié<sup>141</sup> aux informations disponibles dans la base de données encore une fois grâce à un adapter.

Enfin, le niveau de risque et de priorité est choisi par l'utilisateur par le biais d'une « ProgressBar » – ou une barre de chargement – customisée au moyen d'une librairie nommée « philjay.valuebar ». Cette librairie fournit des barres de sélection colorées et animées. Toucher un endroit de la barre voit celle-ci s'agrandir et se rétrécir en fonction jusqu'au lieu de sélection. La barre change également de couleur selon les choix des développeurs. Celles d'Ethias Prevention supportent trois niveaux qui se colorent de vert à rouge en passant par l'orange.

#### 6.2.5 HISTORIQUE DES RAPPORTS



Figure 47 : capture de l'écran d'historique des rapports

La dernière page d'Ethias Prevention est celle de l'historique des rapports. Consistant en une « RecyclerView », cette liste abordée dans la partie consacrée à l'écran d'accueil<sup>142</sup> présente les rapports dont l'état est « finalisé ». Il est possible, en cliquant sur l'icône correspondante de chaque rapport, d'imprimer ou d'envoyer par mail une version PDF du rapport.<sup>143</sup>

La génération du rapport en .pdf est prise en charge par le backend. Ainsi, cliquer sur l'icône d'impression ou d'envoi par mail aura pour premier effet de lancer un service web<sup>144</sup> afin de télécharger le PDF correspondant. Après quoi, si l'objectif de

<sup>141</sup> cf. 5.7.11.2 Présentation du data binding

<sup>142</sup> cf. 6.2.2 Écran d'accueil

<sup>143</sup> cf. 11.1 Un rapport en PDF

<sup>144</sup> cf. 6.3.1 Retrofit pour les services web

l'utilisateur est d'envoyer ce rapport par mail, un dialogue s'affiche demandant à ce dernier d'entrer l'adresse du destinataire avant qu'un intent<sup>145</sup> soit créé avec pour informations le fichier .pdf, l'adresse du destinataire ainsi qu'une demande d'exécution d'une application de gestion de mails déjà installée sur le système. L'utilisateur peut dès lors choisir l'application à exécuter afin que celle-ci s'ouvre sur l'écran d'envoi de mail avec les champs d'adresse du destinataire et d'objet du mail déjà complétés ainsi qu'avec la version PDF du rapport en attaché.

L'impression du rapport par contre, lance directement un intent contenant le fichier PDF ainsi qu'une demande d'exécution d'une application d'impression.

## 6.3 BIBLIOTHÈQUES UTILISÉES

Réinventer la roue est la dernière chose que les développeurs souhaitent réaliser. C'est pourquoi plusieurs librairies ont été utilisées lors du développement d'Ethias Prevention. Si la plupart tiennent du détail souvent pour éviter de devoir taper du code sans intérêt ou afin de créer un élément graphique plus aisément, il y a tout de même deux librairies qu'il convient de détailler étant donné leur importance et leur popularité au sein des applications Android.

### 6.3.1 RETROFIT POUR LES SERVICES WEB

Les services web d'une application désignent les services<sup>146</sup> dédiés au traitement de données exposées sur internet.<sup>147</sup> Généralement, les données qu'ils retournent sont en JSON – un format de données textuelles générique facilement lisible par les humains et compréhensible par les machines<sup>148</sup> – mais il peut arriver qu'elles soient présentées en XML ou d'autres formats de données.

S'il est souvent laborieux d'implémenter un tel service, ceux-ci demandant une grande quantité de code « boilerplate » – ou passe-partout, il existe tout de même des alternatives facilitant grandement la tâche sous forme de bibliothèques de programmation. Celles-ci ont pour but d'abstraire une grande partie du code nécessaire aux services web afin que les développeurs n'aient plus qu'à compléter l'implémentation avec les éléments propres à l'application qu'ils développent.

---

<sup>145</sup> cf. 5.7.7 Intent

<sup>146</sup> cf. 5.7.6 Service

<sup>147</sup> (Acesyde, 2012)

<sup>148</sup> (JSON, s.d.)

Pour Ethias Prevention, c'est la bibliothèque Retrofit<sup>149</sup> qui est utilisée. Elle permet de transformer l'API HTTP en une interface Java<sup>150</sup> très pratique. Cette API, fournie avec Android Studio,<sup>151</sup> sert à la création d'un client qui se charge de gérer les requêtes HTTP vers un serveur, ces requêtes étant ce qui permet aux développeurs d'obtenir les données désirées. Ainsi, l'interface d'Ethias Prevention servant à obtenir une liste de rapports afin de l'afficher à l'écran se présente comme suit :

```
package be.ethias.ethias_prevention_android_v2.model.retrofit;

import ...

public interface EthiasService {

    @GET("/api/reports")
    Call<List<Report>> getReports(@Header("Authorization") String token);

}
```

La méthode getReports() prend un token en argument, s'assurant que l'utilisateur a bien la permission d'obtenir la liste des rapports, et retourne une liste de rapports déjà convertie du format JSON à celui des objets en Java grâce aux fonctionnalités de Retrofit. L'annotation « @GET » fait référence à la requête HTTP « GET » et l'argument « /api/reports » est en fait la suite de l'adresse URL à laquelle la requête est adressée. Elle s'ajoute ici à l'URL de base « http://ethias-test.djm.eu » déclarée dans la classe du service.

Dans le cas d'Ethias Prevention et, plus précisément, de la requête d'une liste des rapports, la classe du service est en fait ReportsService. Elle implémente IReportsService, une interface permettant de découpler le service du reste du code ainsi que Callback<List<Report>>, un callback – où retour – de la requête, fourni par Retrofit qui sert à accéder aux données reçues :

```
package be.ethias.ethias_prevention_android_v2.model.services;

import ...

public class ReportsService implements IReportsService,
    Callback<List<Report>> {
```

<sup>149</sup> (Retrofit, s.d.)

<sup>150</sup> cf. 5.7.1 Java

<sup>151</sup> cf. 4.1 Android Studio

```

@Inject
public OkHttpClient okHttpClient;

@Inject
public Gson gson;

private ReportsCallback reportsCallback;

private Retrofit retrofit;

public void getReports(@Nullable final ReportsCallback
reportsCallback, Context context) {

    EthiasApplication.getDefaultComponent(context).inject(this);

    this.reportsCallback = reportsCallback;

    retrofit = new Retrofit.Builder()
        .baseUrl("http://ethias-test.djm.eu")
        .client(okHttpClient)
        .addConverterFactory(GsonConverterFactory.create(gson))
        .build();

    EthiasService ethiasService =
retrofit.create(EthiasService.class);
    Call<List<Report>> call =
ethiasService.getReports(EthiasApplication.getAppPreference().getAuthorizationPrefValue().getValue());
    call.enqueue(this);
}

@Override
public void onResponse(Call<List<Report>> call,
Response<List<Report>> response) {
    if (response.isSuccessful()) {
        reportsCallback.onReportsCallBack(response.body());
        return;
    }

    Error myError;
    try {
        myError = (Error) retrofit.responseBodyConverter(
            Error.class, Error.class.getAnnotations())
            .convert(response.errorBody());
        Log.d("onResponse ", "onResponse: " + myError.toString());
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

Le corps de la méthode `getReport()` consiste d'abord à injecter les composants tels l'objet `Gson`, utile à la conversion des données reçues en `JSON` vers le format des objets en `Java`, et le client `HTTP` pour ensuite lancer la requête grâce à `Retrofit`. Le token assurant les permissions utilisateurs est récupéré depuis les préférences de

l'application, où il a été sauvegardé au lancement au moyen d'un autre service web, d'identification cette fois.

Enfin, la méthode `onResponse()` est appelée lorsque la requête aboutit. Si elle est réussie et que les données sont reçues, alors un appel à `onReportsCallback()` de l'interface `ReportsCallback` permet à l'application de récupérer les données et d'en faire ce que le développeur lui dicte dans la classe concernée implémentant `onReportsCallback()`. Si elle échoue, la réponse est alors convertie en un objet `Error` auquel il est aisément d'accéder afin de déterminer la nature du problème.

Parmi les fonctionnalités les plus pratiques de Retrofit sont notables la gestion asynchrone des requêtes, la gestion d'arguments, de headers et de commandes – tel le tri – au sein de l'URL accédée ou encore la présence de différents convertisseurs de formats de données.

### 6.3.2 REALM POUR LA PERSISTANCE DES DONNÉES

Assurer le bon fonctionnement d'une application connectée en toutes circonstances peut s'avérer délicat. Pour ce faire, il faut persister les données sur l'appareil, travailler sur ces dernières lorsque nécessaire et enfin mettre à jour la base de données à distance seulement quand l'état de la connexion internet le permet. Si l'accès et le traitement des données à distance tiennent du domaine de Retrofit,<sup>152</sup> la gestion de celles-ci en local concerne plutôt Realm.<sup>153</sup>

Disponible en plusieurs versions comme en Swift et en Objective-C – deux langages de programmation provenant de l'univers Apple – et surtout en Java,<sup>154</sup> Realm est une bibliothèque de programmation qui permet de persister facilement et efficacement les données de la couche model<sup>155</sup> d'une application dans la mémoire du dispositif utilisé.

Son fonctionnement est particulièrement simple comme le montre l'exemple ci-après :

```
@Override
public void onReportsCallBack(List<Report> reports) {
    RealmConfiguration realmConfig = EthiasApplication.getRealmConfig();
    Realm realm = Realm.getInstance(realmConfig);
    realm.beginTransaction();
    realm.copyToRealm(reports);
    realm.commitTransaction();
```

<sup>152</sup> cf. 6.3.1 Retrofit pour les services web

<sup>153</sup> (Java Docs - Realm is a mobile database: a replacement for SQLite & Core Data, 2016)

<sup>154</sup> cf. 5.7.1 Java

<sup>155</sup> cf. 5.7.1.1 Présentation du MVVM

```
    startDashboardActivity();  
}
```

La méthode `onReportsCallback()` est déclenchée lors de la réception de rapports provenant d'une base de données distante.<sup>156</sup> Realm permet de simplement persister ces rapports en réquisitionnant une instance de la base de données locale créée au lancement de l'application et en les y copiant, la seule condition étant que la classe `Report` de la couche model étende la classe `RealmObject` fournie par la bibliothèque.

Dès lors, accéder à un objet `Report` dans la base de données locale peut se faire de la sorte :

```
RealmConfiguration realmConfig = EthiasApplication.getRealmConfig();  
Realm realm = Realm.getInstance(realmConfig);  
Report realmReport = realm.where(Report.class).equalTo("Id",  
    intent.getStringExtra(Consts.REPORT_ID_KEY)).findFirst();
```

Ici, une référence au rapport dont l'identifiant est égal à celui récupéré dans l'intent<sup>157</sup> est obtenue et stockée dans la variable `realmReport`. Pour modifier le rapport obtenu, le développeur peut s'y prendre comme suit :

```
realm.beginTransaction();  
realmReport.setState(Consts.REPORT_STATE_FINALIZED);  
realm.commitTransaction();
```

Ainsi, l'état du rapport passe à "finalisé" et le changement persiste dans les données stockées en local. La mise à jour de la base de données distante peut dès lors se faire en aval de ces opérations en se basant sur les changements locaux.

Les fonctionnalités de Realm ne s'arrêtent pas là. Cette bibliothèque assure entre autres la gestion de clés primaires – servant à gérer les relations entre les objets, permet tout un tas de requêtes, prend en charge le JSON, facilite la migration de la base de données locale et est même livrée avec un programme permettant de consulter cette base de données au travers d'une interface pratique et ergonomique.

<sup>156</sup> cf. 6.3.1 Retrofit pour les services web

<sup>157</sup> cf. 5.7.7 Intent

## 7 CRITIQUES ET SUGGESTIONS

Ethias Prevention est au final une application modeste. Très utile aux agents de terrain de chez Ethias qui sont le public cible, elle passera en production sous peu, c'est-à-dire qu'elle sera accessible aux utilisateurs, et rejoindra sa version iOS parmi les applications à usage professionnel. Elle est ergonomique et globalement bien pensée, sans parler du design efficace et explicite quant à son fonctionnement, permettant de ne pas avoir à former les agents de terrain sur son utilisation. Cependant, son implémentation n'était pas de tout repos, pour cause, entre autres, la possibilité d'ajouter une infinité de sections à un rapport et une infinité d'observations au sein de chacune de ces sections qui était à implémenter en respectant le MVVM.<sup>158</sup> C'est dans cette perspective globale que sont ici pensées des améliorations possibles quant au développement de l'application, que ce soit du point de vue des fonctionnalités ou du déroulement du stage.

La première et la plus facile des critiques à faire à ce projet est qu'il n'existe toujours pas de version Windows Phone de l'application. Vouloir toucher le plus large public qui soit revient à devoir développer sur le plus grand nombre de plateformes possibles ce qui signifie sur mobile qui ne faut pas oublier les utilisateurs munis de smartphones Windows, troisième sur le podium des principaux systèmes d'exploitation mobile.

Un mot sur le développement de la version Android se voit également nécessaire. En effet, si la librairie Data Binding<sup>159</sup> de Google est pleine de bonnes intentions aux applications indéniablement pratiques, il se trouve cependant qu'elle est encore très jeune et qu'il persiste toujours un nombre plutôt conséquent de cas où son utilisation n'aide que peu le respect du design pattern MVVM. Ainsi il convient en ces moments d'improviser au mieux en écrivant du code qui ne fait souvent que contourner le problème ou qui est d'une taille risible au regard de ce qu'il effectue, engendrant des pertes de temps conséquentes. Peut-être le MVC et le MVP<sup>160</sup> ont-ils encore quelques bonnes années devant eux, le temps que la librairie de Google se bonifie par l'intervention de la communauté.

Pour ce qui est des fonctionnalités, il convient de dire qu'Ethias Prevention est plutôt complète au regard de son champ d'application. Il est en effet difficile de faire mieux étant donné qu'elle prend en charge tout, de la création d'un rapport à sa finalisation, jusqu'à la génération, l'envoi par mail et l'impression d'une version PDF de ce dernier et qu'elle permet même d'y adjoindre des médias et d'assister l'utilisateur

<sup>158</sup> cf. 5.7.11.1 Présentation du MVVM

<sup>159</sup> cf. 5.7.11.2 Présentation du data binding

<sup>160</sup> cf. 5.7.11.1 Présentation du MVVM

avec des propositions préenregistrées quant à la nature des missions et des observations. Elle permet également la géolocalisation pour éviter la peine de devoir manuellement introduire le lieu d'intervention et prévoit aussi la date de visite au jour de la création du rapport. Il est même possible d'introduire des pictogrammes illustrant les risques si besoin.

Malgré ça, il est aisément d'imaginer plus encore de fonctionnalités une fois le champ d'application plutôt limité d'Ethias Prevention mis de côté. L'identification au lancement de l'application invite, par exemple, à la considération d'une partie consacrée à la gestion du profil utilisateur voire d'un bureau embarqué. Un tel espace dédié permettrait à l'agent de consulter ses mails, par exemple, ou son programme de la journée voire de la semaine comprenant les différentes missions d'intervention et de prévention qu'il doit effectuer. Toute la gestion administrative de son compte pourrait s'y voir effectuée pour une présence réduite au sein des bureaux d'Ethias et donc une présence accrue sur le terrain, pour un rendement au final plus élevé. Impliquant un développement plus conséquent mais aux avantages certains, peut aussi être considéré l'ajout d'une porte d'entrée vers l'intranet d'Ethias et toutes ses fonctionnalités.

En ce qui concerne le stage, le déroulement s'est produit sans embûche. Les deux premières semaines dédiées à la création d'une application fictive furent une parfaite formation aux tenants et aboutissants du développement sur Android et il est difficile d'imaginer meilleure introduction. Après quoi, il a suffi d'effectuer quelques recherches complémentaires en parallèle de la production afin d'aboutir à une application complète et professionnelle correspondant aux standards des applications actuellement disponibles sur la plateforme. Les quelques outils<sup>161</sup> utilisés chez Djm digital furent des plus utiles au confort de développement avec mention pour SourceTree, cet outil de centralisation et de révision du code. La qualité du matériel fourni, cependant, fut un obstacle notable, la rédaction et le test du code ayant principalement eu lieu sur un ordinateur vieillissant muni d'une version presque obsolète de Windows. Ceci eut pour conséquences des builds longs durant lesquels la seule occupation possible était l'attente, entraînant une augmentation de la durée totale nécessaire au déploiement du programme. Aussi, le nombre limité des appareils disponibles sur lesquels tester l'application aurait pu se voir accru afin de s'assurer l'absence de tout comportement inattendu.

La communication, quant à elle, est aussi candidate aux améliorations. En effet, la version iOS d'Ethias Prevention étant déjà produite, elle servit de support indispensable à l'élaboration de son équivalent Android mais l'apport d'une analyse effectuée en

---

<sup>161</sup> cf. 4 Méthodes, outils et technologie

amont afin d'éclaircir certains détails aurait été le bienvenu. Aussi, le manque d'expérience demeurant, un certain nombre de questions durent être posées. Ceci se fit donc souvent spontanément engendrant peut-être quelquefois une perte de temps pour le locuteur ou l'interlocuteur. Organiser des réunions ou se tourner vers une application de communication dédiée aurait pu optimiser ces interactions.

Enfin, pour conclure ces critiques et suggestions, il est bon de se pencher brièvement sur la plateforme qui fait tourner Ethias Prevention, à savoir Android.<sup>162</sup> Bien que bénéficiant d'une communauté de développeurs et d'utilisateurs conséquente et dévouée, le système d'exploitation mobile de Google est libre, ce qui a tendance à entraîner des problèmes de cohérence interne. Aussi, Google est connu pour ne pas être l'entreprise la mieux organisée qui soit au regard de ses produits. Ainsi, intégrer des fonctionnalités tels que la gestion de médias, la géolocalisation, l'impression de PDF et autres est dès lors plus complexe que sur une plateforme globalement plus stable comme iOS ou Windows Phone. La question de la rétrocompatibilité est aussi plus problématique malgré le soutien de Google<sup>163</sup> et chaque annonce d'une nouvelle fonctionnalité est généralement précédée d'une suée de la part des développeurs. Cependant, Android reste une plateforme multitâche très complète qu'il convient de considérer à chaque développement d'application.

---

<sup>162</sup> cf. 5 Android

<sup>163</sup> cf. 5.7.10 Les bibliothèques de compatibilité

## 8 CONCLUSION

Douze semaines ont déjà été consacrées à la réalisation d’Ethias Prevention et il reste encore quelques détails à régler avant sa mise en disponibilité aux utilisateurs. Gérer les différentes langues, implémenter une solution du côté d’Ethias pour ce qui est de l’oubli du mot de passe ou encore effectuer une légère repasse sur le design sont autant de tâches qu’il reste à exécuter avant l’export de l’application sur Google Play. Mais dans les faits, Ethias Prevention est désormais utilisable et le développement de ses fonctionnalités principales est terminé. Il est donc de bon ton, en guise de mot final de ce rapport, de revenir sur l’expérience vécue lors de ces quelques derniers mois.

Ainsi, c'est dans une ambiance professionnelle mais agréable que se sera déroulé ce stage. Les deux premières semaines étaient consacrées à ma formation en Android, plutôt efficace grâce à une immersion directe dans le développement d'une application mobile, et m'ont permis de commencer ma familiarisation avec le monde entrepreneurial. Les douze semaines qui s'en suivirent étaient quant à elles entièrement consacrées à Ethias Prevention mais n'en étaient pas moins enrichissantes et passionnantes pour autant. Le travail s'est déroulé à merveille et m'a permis d'apprendre énormément, je pourrais même presque aller jusqu'à avancer que le développement sur Android n'a plus de secrets pour moi.

Cette immersion en entreprise m'a également appris beaucoup de l'organisation du travail qui y est effectué. J'ai par exemple pu assister au déroulement d'un projet ambitieux concernant la mise en circulation de baby-foot connectés aussi bien du point de vue développement que marketing, m'ouvrant les yeux sur la complexité du fonctionnement d'une telle opération. J'ai également eu un aperçu de la rigueur exigée dans le milieu ce qui m'a permis de revenir sur mes propres méthodes de travail afin de les améliorer. Si certains points sont candidats à l'amélioration tels la communication interne de l'entreprise et le matériel qui m'a été fourni, l'expérience était tout de même, dans l'ensemble, des plus riches et le fait d'avoir participé au développement d'une application qui sera utile à nombre de personnes a quelque chose de magique dont je ne peux qu'être fier.

Une chose est sûre après cette phase d'immersion en entreprise, c'est que l'informatique en est encore à son adolescence et qu'elle jouit d'un magnifique avenir. Je ne regrette en rien mon choix de me spécialiser dans ce domaine et je continue d'envisager la voie du développement de programmes en ce qui concerne le début de ma carrière.

## 9 BIBLIOGRAPHIE

- Acesyde. (2012, Novembre 23). *Accéder aux services web via Android*. Consulté le Mai 18, 2016, sur Developpez.com: <http://acesyde.developpez.com/tutoriels/android/web-services-android/>
- Activity | Android Developers. (2016, Mai 04). Consulté le Mai 09, 2016, sur Android Developers: <http://developer.android.com/reference/android/app/Activity.html>
- ADAM, L. (2015). *Analyse et développement d'applications mobiles dédiées aux appareils sous iOS*. Travail de fin d'étude, Haute Ecole de Namur-Liège-Luxembourg - Implantation IESN, Namur. Consulté le Mai 18, 2016
- Adapter | Android Developers. (2016, Mai 04). Consulté le Mai 09, 2016, sur Android Developers: <http://developer.android.com/reference/android/widget/Adapter.html>
- AMADEO, R. (2012, Septembre 17). *A History of Pre-Cupcake Android Codenames*. Consulté le Février 23, 2016, sur Android Police: <http://www.androidpolice.com/2012/09/17/a-history-of-pre-cupcake-android-codenames/>
- Android. (s.d.). Consulté le Février 23, 2016, sur Android: <http://www.android.com/>
- Android Developers. (s.d.). Consulté le Mai 17, 2016, sur Android Developers: <https://developer.android.com/index.html>
- Android Interfaces and Architecture | Android Open Source Project. (2016, Février 25). Consulté le Février 23, 2016, sur Android Open Source Project: <http://source.android.com/devices/>
- Android Pugin for Gradle | Android Developers. (2016, Février 19). Consulté le Février 23, 2016, sur Android Developers: <http://developer.android.com/tools/building/plugin-for-gradle.html>
- Android versions comparison | Comparison tables - SocialCompare. (2016, Février 16). Consulté le Février 23, 2016, sur SocialCompare: <http://socialcompare.com/en/comparison/android-versions-comparison>
- App Manifest | Android Developers. (s.d.). Consulté le Mai 12, 2016, sur Android Developers: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>
- Application | Android Developers. (2016, Mai 04). Consulté le Mai 09, 2016, sur Android Developers: <http://developer.android.com/reference/android/app/Application.html>
- BANERJEE, A. (2014, Juin 25). *Android Auto unveiled at Google I/O: what you need to know | AndroidAuthority*. Consulté le Février 23, 2016, sur AndroidAuthority: <http://www.androidauthority.com/android-auto-google-io-397118/>
- Bibliothèque d'API. (s.d.). Consulté le Mai 22, 2016, sur Console Developers: <https://console.developers.google.com/?pli=1>
- BIRCH, J. (2015, Septembre 21). *Approaching Android with MVVM -- ribot labs*. Consulté le Mai 10, 2016, sur ribot labs: <https://labs.ribot.co.uk/approaching-android-with-mvvm-8ceec02d5442#.2vbsqode9>
- BYOUS, J. (2003, Avril). *Java Technology: The Early Years*. Consulté le Février 25, 2016, sur Sun microsystems:

<http://web.archive.org/web/20100105045840/http://java.sun.com/features/1998/05/birthday.html>

CAMPBELL, A. (2015, Novembre 15). *Android Data Binding Tutorial*. Consulté le Mai 17, 2016, sur CapTech: <https://www.captechconsulting.com/blogs/android-data-binding-tutorial>

*Context | Android Developers*. (2016, Mai 04). Consulté le 04 09, 2016, sur Android Developers: <http://developer.android.com/reference/android/content/Context.html>

DANON, G. (2014, Juin 13). *Les machines à laver Samsung et leurs application*. Consulté le Février 23, 2016, sur Android-France: <http://android-france.fr/2014/06/les-machines-laver-samsung-application/>

*Dashboards | Android Developers*. (2016). Consulté le Février 23, 2016, sur Android Developers: <http://developer.android.com/about/dashboards/index.html>

*Data Binding Guide | Android Developers*. (s.d.). Consulté le Mai 10, 2016, sur Android Developers: [http://developer.android.com/tools/data-binding/guide.html#build\\_environment](http://developer.android.com/tools/data-binding/guide.html#build_environment)

*Djm Digital : Création de site internet & application mobile*. (s.d.). Consulté le Février 18, 2016, sur djmdigital: <http://www.djmdigital.be/>

D'ORAZIO, D. (2014, Mars 18). *Google reveals Android Wear, an operating system for smartwatches | TheVerge*. Consulté le Février 23, 2016, sur TheVerge: <http://www.theverge.com/2014/3/18/5522226/google-reveals-android-wear-an-operating-system-designed-for>

*Download Android Studio and SDK Tools | Android Developers*. (s.d.). Consulté le Février 18, 2016, sur Android Developers: <http://developer.android.com/sdk/index.html>

DUBISY, F. (2015). *Programmation mobile*. Namur: Haute École de Namur - Liège - Luxembourg Implantation IESN.

*Gestures - Patterns - Google design guidelines*. (s.d.). Consulté le Mai 09, 2016, sur Google: <https://www.google.com/design/spec/patterns/gestures.html#gestures-touch-mechanics>

*Getting Started with Data Binding in Android*. (2015, Septembre 15). Consulté le Mai 10, 2016, sur opgenorth.net: <http://www.opgenorth.net/blog/2015/09/15/android-data-binding-intro/>

*Intent | Android Developers*. (2016, Mai 04). Consulté le Mai 09, 2016, sur Android Developers: <http://developer.android.com/reference/android/content/Intent.html>

*Introduction - Material Design - Google design guidelines*. (s.d.). Consulté le Février 25, 2016, sur Material Design: <https://www.google.com/design/spec/material-design/introduction.html>

*Java Docs - Realm is a mobile database: a replacement for SQLite & Core Data*. (2016). Consulté le Mai 18, 2016, sur Realm: <https://realm.io/docs/java/latest/>

*JSON*. (s.d.). Consulté le Mai 18, 2016, sur json.org: <http://www.json.org/>

LARDINOIS, F. (2015, Mai 28). *Google Launches Android M Preview With Fingerprint Scanner Support, Android Pay, Improved Permissions And Battery Life | TechCrunch*. Consulté le Février 23, 2016, sur TechCrunch: <http://techcrunch.com/2015/05/28/google-announces-android-m-with-fingerprint-scanner-support-android-pay-improved-permissions-battery-and-performance-tweaks/>

LEGGETT, J. (2011, Avril 8). *Android timeline 2003-2011*. Consulté le Février 23, 2016, sur USwitch:  
<http://www.uswitch.com/mobiles/news/2011/04/android-timeline-2003-2011/>

*Main Page - Android Wiki*. (2013, Août 5). Consulté le Février 18, 2016, sur Android Wiki:  
[http://android-dls.com/wiki/index.php?title=Main\\_Page](http://android-dls.com/wiki/index.php?title=Main_Page)

MAZELIER, G. (2011, Septembre). *Build automatisé : à la découverte de Gradle / GLMF-141 / Linux Magazine / GNU / Connect - Edition Diamond*. Consulté le Février 19, 2016, sur Connect - Edition Diamond: <http://connect.ed-diamond.com/GNU-Linux-Magazine/GLMF-141/Build-automatise-a-la-decouverte-de-Gradle>

MERCHIE, V. (2013). *Mise à jour d'une application mobile Android : "Belfius Travel"*. Travail de fin d'étude, Haute École de la Province de Liège, Liège. Consulté le Février 27, 2016

OPAM, K. (2014, Juin 25). *Google officially unveils Android TV | TheVerge*. Consulté le Février 23, 2016, sur TheVerge: <http://www.theverge.com/2014/6/25/5840424/google-announces-android-tv>

*Oracle and Sun Microsystems | Strategic Acquisitions | Oracle*. (s.d.). Consulté le Février 25, 2016, sur Oracle: <https://www.oracle.com/sun/index.html>

*Premiers pas | Google Maps Android API | Google Developers*. (2016, Mars 24). Consulté le Mai 22, 2016, sur Google Developers: [https://developers.google.com/maps/documentation/android-api/start#etape\\_4\\_obtenir\\_une\\_cle\\_dapi\\_google\\_maps](https://developers.google.com/maps/documentation/android-api/start#etape_4_obtenir_une_cle_dapi_google_maps)

*Retrofit*. (s.d.). Consulté le Mai 18, 2016, sur square.github.io: <http://square.github.io/retrofit/>

*Services | Android Developers*. (s.d.). Consulté le Mai 09, 2016, sur Android Developers: <http://developer.android.com/guide/components/services.html>

*Signature et clés d'API | Google Maps Android API | Google Developers*. (2016, Mars 24). Consulté le Mai 22, 2016, sur Google Developers: <https://developers.google.com/maps/documentation/android-api/signup#release-cert>

*Support Library Features | Android Developers*. (2016, Mai 09). Récupéré sur Android Developers: <http://developer.android.com/tools/support-library/features.html#v4>

*Supporting Multiple Screens | Android Developers*. (2016, Mai 09). Récupéré sur Android Developers: [http://developer.android.com/guide/practices/screens\\_support.html](http://developer.android.com/guide/practices/screens_support.html)

*Torrent file - Wikiwand*. (s.d.). Consulté le Mars 13, 2016, sur Wikiwand: [https://www.wikiwand.com/en/Torrent\\_file](https://www.wikiwand.com/en/Torrent_file)

*Unit Testing*. (2016). Consulté le Mai 10, 2016, sur Microsoft | Developer: <https://msdn.microsoft.com/en-us/library/aa292197%28v=vs.71%29.aspx?f=255&MSPPError=-2147217396>

*ViewGroup | Android Developers*. (s.d.). Consulté le Mai 17, 2016, sur Android Developers: <https://developer.android.com/reference/android/view/ViewGroup.html>

WALSH, N. (1998, Octobre 03). *A Technical Introduction to XML*. Consulté le Mai 10, 2016, sur XML.com: <http://www.xml.com/pub/a/98/10/guide0.html>

ZDNet.fr, L. r. (2015, Décembre 7). *Chiffres clés : les OS pour smartphones*. Consulté le Février 23, 2016, sur ZDNet.fr: <http://www.zdnet.fr/actualites/chiffres-cles-les-os-pour-smartphones-39790245.htm>

## 10 TABLE DES FIGURES

Figure 1 : logo de Djm digital.....	5
Figure 2 : localisation des locaux.....	5
Figure 3 : les quatre domaines d'expertise de Djm digital .....	6
Figure 4 : logo d'Android Studio.....	8
Figure 5 : interface utilisateur d'Android Studio.....	8
Figure 6 : écran « New Project ».....	9
Figure 7 : écran « Target Android Devices » .....	9
Figure 8 : écran « Add an activity to Mobile » .....	10
Figure 9 : écran « Customize the Activity ».....	10
Figure 10 : logo de Gradle.....	11
Figure 11 : logo de Git.....	13
Figure 12 : logo de SourceTree.....	14
Figure 13 : logo de SoapUI.....	14
Figure 14 : logo de Node.js.....	14
Figure 15 : logo de toggl .....	14
Figure 16 : logo de Jira .....	15
Figure 17 : logo de Bitbucket.....	15
Figure 18 : Xperia M2 .....	15
Figure 19 : Nexus 7 .....	16
Figure 20 : Bugdroid, la mascotte représentant Android .....	17
Figure 21: tableau des différentes versions d'Android .....	22
Figure 22 : répartition des différentes versions d'Android sur le marché.....	22
Figure 23 : logo Google Play .....	22
Figure 24 : interface utilisateur du SDK Android.....	23
Figure 25 : architecture Android selon l'Android Open Source Project, partie 1.....	24
Figure 26 : architecture Android selon l'Android Open Source Project, partie 2 .....	25
Figure 27 : les deux états d'un bouton selon le material design.....	26

Figure 28 : logo de Java .....	27
Figure 29 : cycle de vie d'une activité selon le site Android Developers .....	30
Figure 30 : cycle de vie d'un fragment selon le site Android Developers .....	31
Figure 31 : tableau des différentes gestes d'Android.....	35
Figure 32 : MVVM, MVP et MVC .....	37
Figure 33 : activity principale de fictional_login.....	42
Figure 34 : chargement de fictional_login.....	42
Figure 35 : structure d'un projet Android .....	43
Figure 36 : diagramme d'activité .....	57
Figure 37 : diagramme d'état.....	58
Figure 38 : diagramme des cas d'utilisation .....	58
Figure 39 : diagramme des classes persistantes, généré grâce au programme Core Data pour la version iOS d'Ethias Prevention.....	60
Figure 40 : diagramme de navigation.....	61
Figure 41 : capture d'écran de la page de connexion .....	62
Figure 42 : capture d'écran de la page d'accueil.....	63
Figure 43 : capture d'écran du menu.....	65
Figure 44 : capture de l'écran "Données Générales".....	68
Figure 45 : capture de l'écran "Observations et recommandations" .....	71
Figure 46 : capture d'écran d'une nouvelle section.....	72
Figure 47 : capture de l'écran d'historique des rapports.....	73

## 11 ANNEXES

### 11.1 UN RAPPORT EN PDF

**ethias**  
PREVENTION

# RAPPORT DE VISITE

Dossier	Folder name
Référence	123/456
Client	Client Name
Date de visite	18-03-16
Agent	Michele Obama
Type	Dépistage de risques - Vol ou dégradations
Lieu	Rue Haute, 4600 Visé
Participants	Participants
Destinataires	Distribution

**Préliminaire**

Le présent rapport est un compte-rendu des observations concrètes faites sur les lieux de travail et formule des recommandations dans le cadre d'une démarche d'amélioration de la sécurité et du bien-être des travailleurs dans l'entreprise.

Ce rapport constitue un instantané et ne doit donc pas être considéré comme exhaustif. La mise en œuvre des démarches d'améliorations relève de la responsabilité de l'employeur et des membres de la ligne hiérarchique (article 5 de la loi du 4/08/1996 relative au bien-être au travail et article 13 de l'AR du 27 mars 1998 portant sur les principes généraux relatifs à la politique du bien-être au travail).



Ethias SA - Rue des Croisiers 24 - 4000 Liège - [www.ethias.be](http://www.ethias.be)

1



#### Observations générales

Observations

#### Recommandations générales

Recomendations





## Section 1

### Section 1 – Topic 1

#### Risques



#### Priorité



#### Catégories

#### Législation(s)

#### Pictogrammes

#### Commentaires

#### Section 1 – Topic 1 – Observations



