RAPPORT PROJET : APPLICATION MOBILE

4A ESIEA

BANZOUZI Christanis Meril

2018-2019

I. DESCRIPTION DU PROJET :

Introduction :

Ce projet a pour principale objectif de mettre en évidence les notions et les outils de développement d'application mobile appris en cours notamment : les architectures : MVVM, Clean Architecture, l'utilisation des outils Kotlin et Android : avec le logiciel Android-Studio. Pour ce projet, j'ai choisi de suivre un tutoriel permettant de développer une application étape par étape et mettant en place les architecture demander pour ce projet.

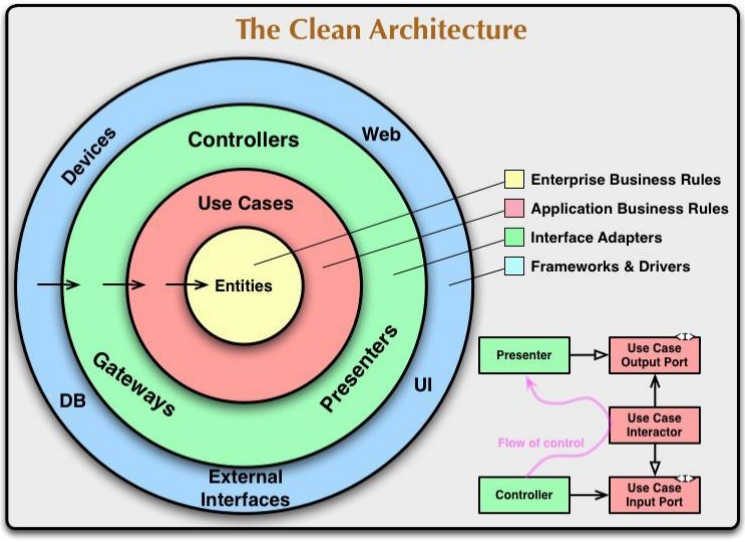
Principe de l'application mobile :

« Forecast » est une application mobile qui permet à l'utilisateur de consulter en temps réel l'état actuel de la météo et aussi sur sept jours (Une semaine) s'il le désir. L'utilisateur a la possibilité de choisir l'unité de mesure (degré Celsius ou Fahrenheit) et la localisation.

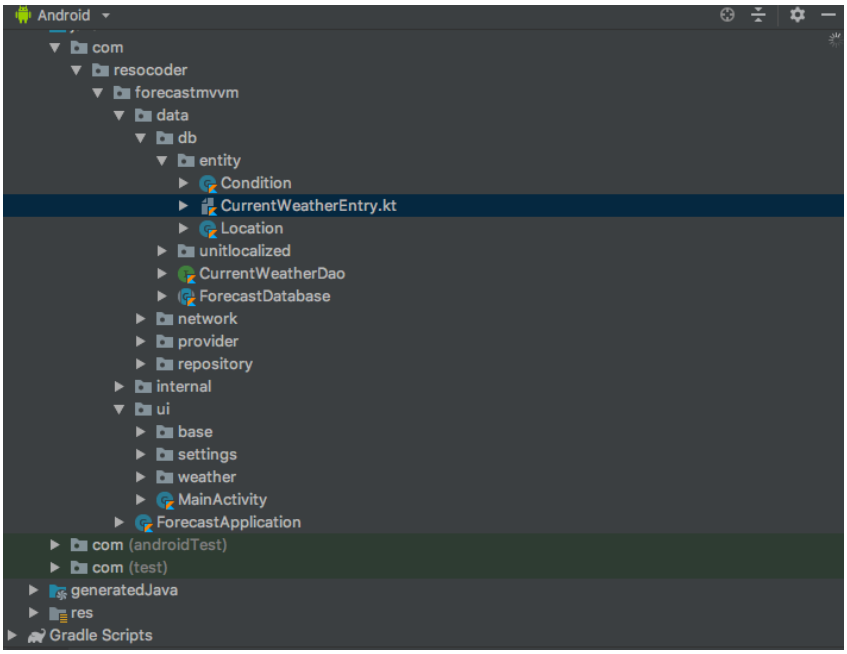
II. ETAPE DE CONCEPTION :

Architecture :

Clean Architecture : le principe est de séparé le code en couches en forme d'oignon avec une règle de dépendance : les couches internes ne doivent rien savoir des couches externes. Cela signifie que les dépendances doivent être dirigées vers l'intérieur, comme nous le montre l'image ci-dessous :



Implémentation de clean l'architecture dans le projet :



Donc cela implique que le code soit : indépendant des cadres, testable, indépendant de l'interface utilisateur, indépendant de la base de données, indépendant de tout organisme externe. Celà signifie pour Android que mon application est développée sur trois couches :

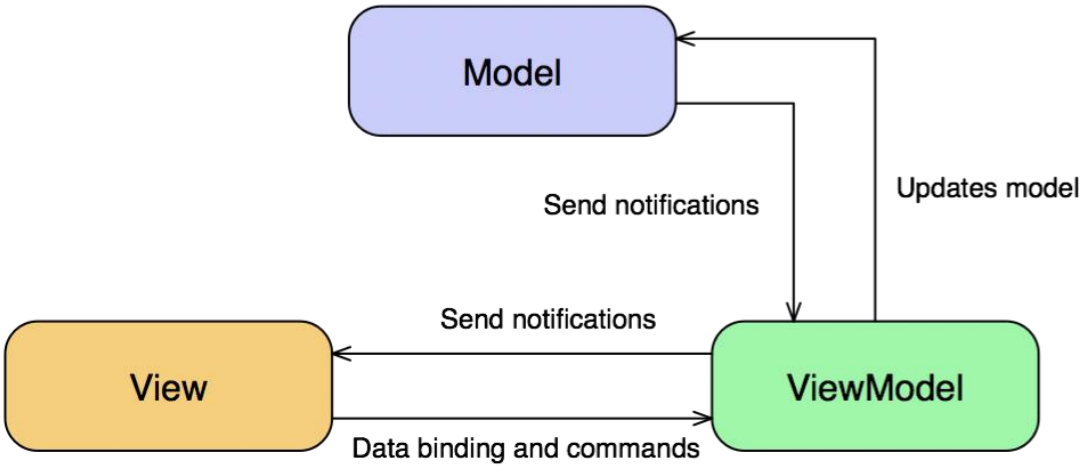
• extérieur : couche d'implémentation, est l'endroit où tout se passe dans un cadre spécifique. Le code spécifique au Framework inclut chaque ligne, cela inclut tous les outils Android tels que la création d'activités et de fragments, les intentions d'envoi et autres codes de Framework tels que le code de réseau et les bases de données.

• milieu : couche d'adaptateur d'interface, son rôle est d'agir en tant que connecteur entre la logique d'affaires et du code spécifique au Framework.

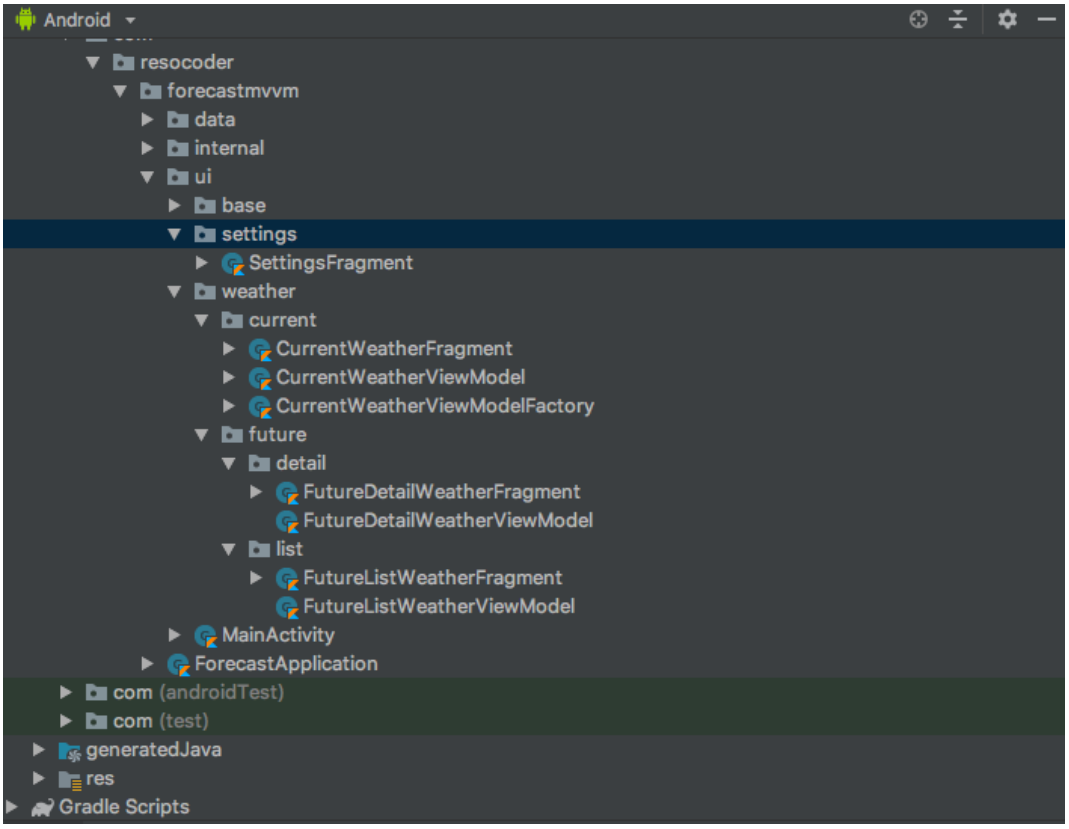
• Intérieur : couche de logique métier, C'est la couche la plus importante est la couche de logique métier. C'est elle qui résout réellement le problème de la création de votre application. Cette couche ne contient aucun code spécifique au Framework. De cette façon, le code de logique métier est facile à tester et c'est le principal avantage de l'architecture propre.

L'architecture Mvvm : mvvm a été introduit par Microsoft et il s'agit d'un modèle naturel pour les plates-formes XAML, mais c'est une nouvelle approche pour les plates-formes Android. Ce modèle d'architecture permet d'abstraire l'état et le comportement d'une vue, ce qui permet de développer l'interface utilisateur à partir du modèle séparément. En introduisant ViewModel en tant

que médiateur, dont la responsabilité est d'exposer les données du modèle et de gérer la logique des applications lors de l'affichage. Parfois, ce motif architectural est également appelé Model ViewBinder.



Implémentation de l'architecture MVVM dans le projet :



Comme illustré à la figure 5, le rôle de ViewModel est de mettre à jour le modèle et d'envoyer une notification à sa vue. En retour, il reçoit également une notification du modèle. De plus, la vue est connectée à ViewModel par liaison de données.

View est la partie visuelle. Il est responsable du démarrage des activités et de la gestion des menus, des autorisations, des écouteurs d'événements, des toasts, des barres de dialogue, des boîtes de dialogue, etc. La vue se lie aux variables observables exposées par viewmodel à l'aide du framework de liaison de données.

ViewModel agit en tant que médiateur pour transmettre les événements déclenchés par l'utilisateur dans le composant vue au composant de modèle. Il n'est pas lié à la vue elle-même mais sert uniquement de modèle de vue. Deuxièmement, ViewModel encapsule le modèle et prépare les données observables nécessaires à la vue. View Model reçoit ses données à partir du modèle. Le ViewModel gère les responsabilités suivantes :

- exposer les données,
- exposer l'état (en cours, hors ligne, vide, erreur, etc.),
- visibilité de la poignée,
- validation d'entrée,
- exécuter des appels sur le modèle,
- exécuter les méthodes dans la vue.

La vue doit connaître le contexte de l'application qui peut démarrer un service, lier un service, envoyer ou recevoir une diffusion et charger une valeur de ressource. Mais ViewModel ne peut pas démarrer une activité.

Outils utiliser :

- KOTLIN : Kotlin est un langage de programmation moderne, à source ouverte et à typage statique, qui permet aux développeurs de créer des applications Web et mobiles pour Android, JVM, des navigateurs modernes et des solutions natives. Il est 100% interopérable avec Java. Kotlin est également concis, simple et facile à lire et à écrire. C'est un langage très expressif et beaucoup plus concis que Java, faisant de Kotlin le substitut moderne idéal. Cet outil est est développé par JetBrains (la société derrière des outils de développement bien connus tels que IntelliJ et Android

Studio) et dispose du support officiel de Google pour le développement Android. Bien qu'il soit nouveau sur le marché, Kotlin est déjà utilisé par de grandes entreprises telles que Netflix, Uber, Amazon et Pinterest.

- API APIXU : est une puissante plate-forme d'API météo entièrement gérée offrant de nombreuses API allant de la météo et de l'astronomie au fuseau horaire et à la géolocalisation. Actuellement, nous couvrons globalement : météo en temps réel, prévisions météo jusqu'à 14 jours, prévisions météorologiques historiques, astronomie, fuseau horaire, données de localisation et jusqu'à 99,9% de disponibilité.

III. Fonctionnalités :

L'application contient trois fonctionnalités, répartis sur trois sur trois interfaces différentes :

- l'état du temps actuellement : l'application permet à l'utilisateur de consulter l'état météorologique en fonction de la position donnée,
- prévision sur 7 jours : l'utilisateur peut consulter les prévisions météo sur 7 jours,
- paramétrage : l'utilisateur peut choisir la forme des données qu'il reçoit et sa localisation.

IV. Difficultés rencontrées :

- Utilisation et apprentissage de l'outil de développement Kotlin.
- Mise en place des différentes interfaces de l'application.
- Manque de temps pour le développement de toute l'application.

V. Conclusion :

Ce projet m'a permis de mettre en pratique mes connaissances sur la programmation mobile (ANDROID) surtout d'apprendre deux nouvelle notions : Kotlin, clean architecture et l'architecture MVVM.

Aperçus de l’application :