International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Development of Native Android Mobile application using MVVM Architecture

AR ROHAN[1], Sowmya Nag K[2]

[1]*UG Student, Department of ECE, RV College of Engineering, Bengaluru, India*
[2]*Assistant Professor, Department of ECE, RV College of Engineering, Bengaluru, India*

Abstract - *In recent times android has become one of the most popularly used mobile operating system due to its architecture based on Linux kernel which gives it seamless compatibility with almost all hardware platforms starting from a smart watch to a television. Development of applications using native approach can be very difficult to manage for example when the orientation of the screen is changed. There are lot other difficulties that are faced in native approach due to tight coupling between the layout files and business logic files. To enable loose coupling between components , this paper MVVM architecture to achieve loose coupling and also describes a sports equipment retail app which is built using the native approach and kotlin with MVVM architecture , live data ,room database. All the screen UI are made according to material design.*

***Key Words*: MVVM Architecture, Loose coupling, Android Architecture, native development, material design, Android Studio.**

1. INTRODUCTION

Android is an open source OS based on Linux kernel .Since Android is a mobile operating system it has constraints such as storage, power, heat dissipation.[1]Application development on android is very different from that on a PC. In android application development is basically coordinating the components and managing their state of the components. Since android devices are of various configurations and android has a base from Linux kernel, it is easy to install applications on different android devices. Most android devices have different screen sizes and it is important to make the UI scalable and the orientation. Screen orientation of the device can be changed at any time during the use of the application. Hence it is very important to loosely couple the UI layout with the business logic components in the native development of the application, for this MVVM architecture is very reliable for loose coupling.

This paper introduces the MVVM architecture and android application components with the help of a sports retail E-commerce app which is developed using android native approach, kotlin with MVVM architecture.

2. INTRODUCTION TO ANDROID PLATFORM ARCHITECTURE

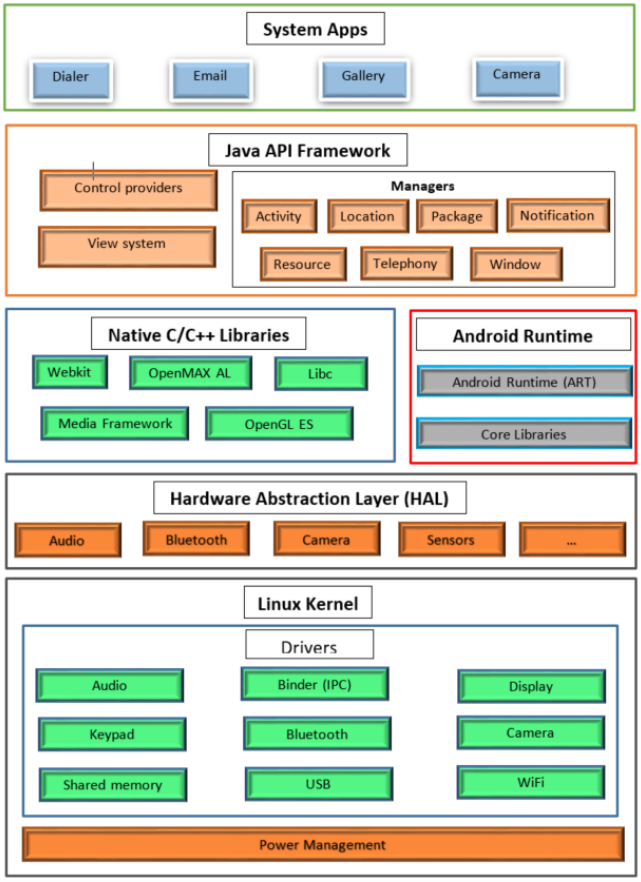

Fig -1: Android Platform Architecture

Android is a Linux-based open source layered operating system [2].

2.1 Linux Kernel

Bottom most layer is the Linux kernel which provides a level of abstraction between the device hardware, it contains one of the most important drivers like USB, camera and networking drivers as well which will solve the problem of interfacing to the peripheral hardware. It also provides core system services such as memory management, security, process management.

2.2 Android Runtime

It is the second layer from the bottom, this has a few core libraries and a main component called Dalvik virtual machine which is a java virtual machine which is optimized by google for android. Features of Dalvik Virtual Machine:

- Android application will run as an instance of a DVM. DVM can have multiple instances and each instance runs in a separate process.[1]
- DVM relies on Linux kernel for memory management, thread scheduling.[1]

2.3 Application Framework

This provides high level services to applications in the form of kotlin/java classes. All android apps use application framework. The android framework services include:

- App life cycle, activity stack are controlled by the activity manager.
- Content providers which help application share data with other applications.
- Resource manager provides access to embedded resources such as localized strings, localized color settings and user interface layout files.

2.4 Application Layer

layer in which all Android applications gallery, email, contacts, dialer. All applications are written using kotlin/java.

3. MVVM ARCHITECTURE IN ANDROID

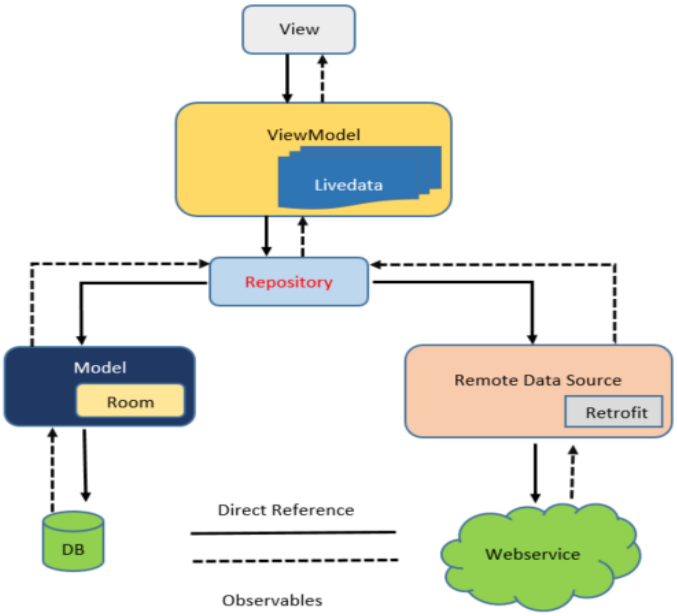

Fig -2: MVVM flow in android native development

MVVM is a Model-View-View Model architecture that removes the tight coupling between components. Here in this architecture, the children component will not have the direct reference to the parent, have the reference by observables.

3.1. MVVM File Structure for Sports Retail App

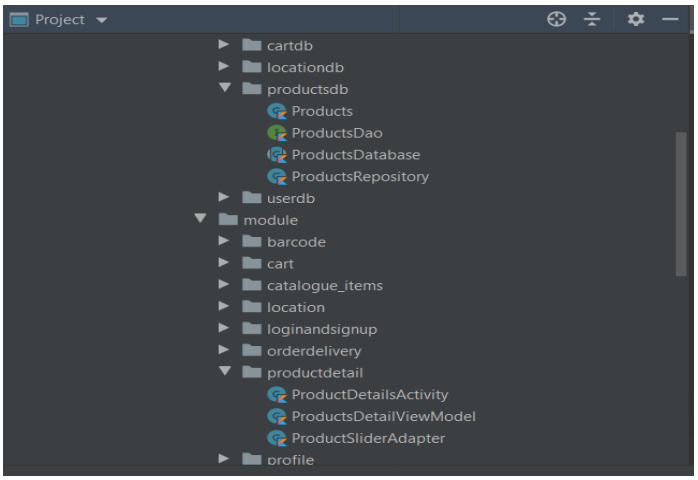

Fig -3: Sports retail app MVVM file structure

Product catalogue screen is the home screen of the app and all the products available in a particular store is displayed here. The products catalogue screen is horizontally and vertically scrollable and is created using recycler view following the MVVM architecture.

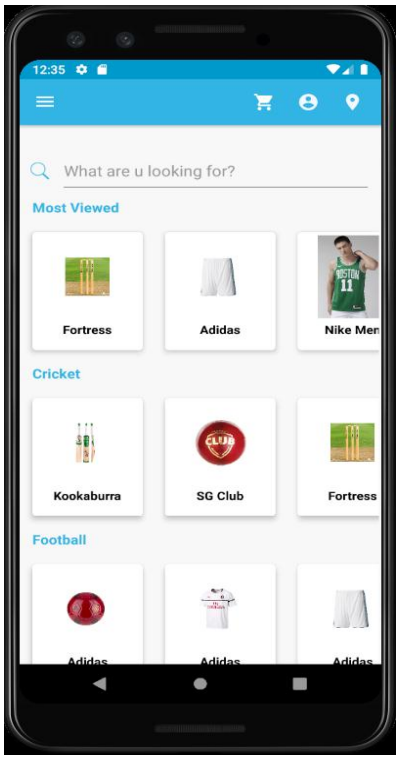

Fig -4: Products catalogue screen

3.2 View

Is a part of your app which handles what the user sees and touches on the screen A View does all the things an Activity or a Fragment can do. View does not contain any business logic like communicating with the DB. They can only display UI on the screen which they get from their view model and dispatch user interaction events to their respective view models

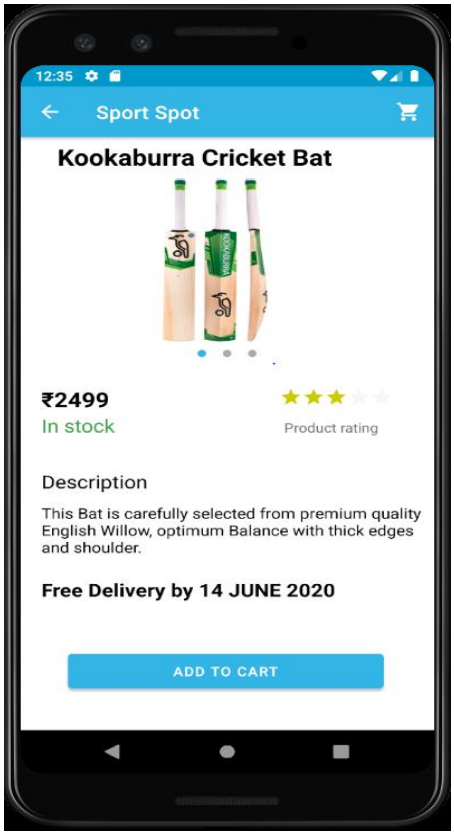

Fig -5: Product details screen

Product details screen of the sports app is implemented in the MVVM structure as shown in fig 5.

Here the view is ProductDetailsActivity.kt and the output screen for the same is shown in fig 5.Whenever add to cart button is clicked, this action is sent to ProductDetailsViewModel.kt and whenever the view model observes count of the product in the database as 0 then In stock changes to Out of stock dynamically.

3.3 View Model

VM acts as a glue between view and business logic, provides data to the view by fetching it from the repository. From fig 2 we can observe that there is no direct reference from view model to view, data in the view model is made observable, instead of updating the view each time the data changes, View already has a reference to its View Model, so it can simply observe some data which the View Model exposes. This is done through Live Data, advantage is that it automatically doesn't notify the observer if its activity or fragment is already destroyed, which gives the user the freedom to manage the lifecycle. Fig 6 shows the products cart screen in which whenever products are added or removed from the cart, the data from the repository is changed and this is observable to the view from the view model through live data. View Model also observes from the repository.

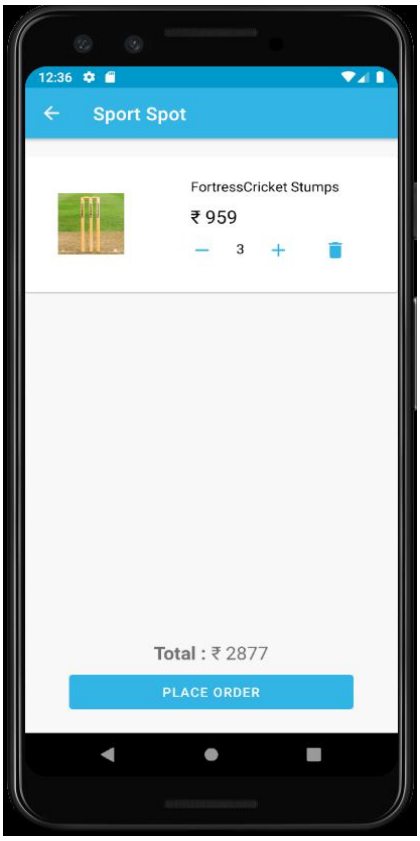

Fig -6: Product cart screen

3.4 Repository

Repository acts as a mediator between local storage and the server. Repository is the place where it is decided whether remote data should be cached. Repository is also the single source of truth for View Models. When View Model wants data, it gets it from the Repository. Repository can fetch the data (observe) from the local data base or web service. View model is not concerned with that. In this app there is a repository called ProductsRepository which fetches data from room (SQ-lite) database.

3.5 Model

All the business logic is written in the model. Repository acts as the intermediate step between view model and model.

4. ADVANTAGES AND DISADVANTAGES OF MVVM ARCHITECTURE

➢ ADVANTAGES

- There is no tight coupling between the view, view model. Repository and model make them loosely coupled.
- Unit testing can be done easily as external and internal dependences are in separate pieces of code from the parts with the core logic.
- There are no interfaces between view Model and view.
- Any change can be done very easily as all the components are loosely coupled(extensibility).
- This structure can be maintained easily and is preferred for agile development as releases can be made very often easily.

➢ DISADVANTAGES

- Observables must be created for each UI component.
- When there is complex data-binding, debugging can be difficult.
- Code size will be pretty large.

5. CONCLUSIONS

Android is a stack for mobile devices which have an operating system, middleware and applications. Android SDK provides API's and tools which are required to develop applications on Android platform using the Kotlin/Java .This paper gives an introduction to android platform architecture and a detailed introduction to MVVM architecture for loose coupling. Finally, a sports retail app developed using native approach with MVVM architecture adopted and material design used for the UI and kotlin programing language for the development is used as an example to illustrate the MVVM architecture.

REFERENCES

[1] Jianye Liu; Jiankun Yu, Research on Development of Android Applications, International Conference on Intelligent Networks and Intelligent Systems,2011.

[2] OL. Google Android Developers, Android Develop Guide, https://developer.android.com/guide/platform.

[3] A. Kathuria and A. Gupta, "Challenges in Android Application Development: A Case Study" International Journal of Computer Science and Mobile Computing, vol. 4, no. 5, pp. 294–299, May 2015.

[4] Ribeiro and A. R. D. Silva, "Survey on Cross-Platforms and Languages for Mobile Apps," Eighth International Conference on the Quality of Information and Communications Technology, 2012.

[5] Pohares, V. C. Kulloli, T. Bhattacharyya, and S. Bhure, "Cross Platform Mobile Application Development," International Journal of Computer Trends and Technology, vol. 4, no. 5, pp. 1095–1100, 2013.

[6] J. Dongjiu Geng, Yue Suo, Yu Chen, Jun Wen, Yongqing Lu, Remote Access and Control System Based on Android Mobil Phone'vol.2. Journal of Computer Applications, pp. 560-562, 2011.