УДК 004.4

А.А. Пакула

С.М. Довгалець

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ-ГРИ З АРХІТЕКТУРОЮ MVVM

Вінницький національний технічний університет

Анотація

Розглянуто та проаналізовано архітектурний підхід для створення надійного, гнучкого та легко масштабуючого додатку-гри для пристроїв на платформі Android. Така архітектура надає можливість зручно підтримувати, масштабувати та тестувати програмний код додатку .

Ключові слова: Архітектура, MVVM,ViewModel, LiveData, Android додаток.

Abstract

The architectural approach to create a reliable, flexible and eaiy scalable game application for Android devices. This architecture makes it easy to support, scale and test the application code

Keywords: Architecture, MVVM, ViewModel, LiveData, Android application.

Вступ

Екосистема засобів розробки мобільних додатків під платформу Android розвивається дуже швидко. Щотижня з'являються нові інструменти, оновлюються існуючі бібліотеки, публікуються нові версії Support Library або Google Play Services. Враховуючи це, слід постійно слідкувати за новими технологіями та їх можливостями. Це дозволить підвищити ефективність та актуальність додатку. Однак, додавання нових технологій або ж функціоналу часто може виявитись складним, та потребує хорошої організованості проекту.

Добре продумана архітектура дозволяє зекономити розробникам багато часу та сил. Програму з хорошою архітектурою легше розширювати і змінювати, а також тестувати, налагоджувати і розуміти.

Аналіз критеріїв якісної архітектури

Якісна архітектура - це перш за все вигідна архітектура, що робить процес розробки і супроводу програми більш простим і ефективним. Тому можна з легкістю сформулювати список цілком розумних і універсальних критеріїв:

- ефективність;
- гнучкість.;
- масштабування.

Ефективність. В першу чергу, програма повинна виконувати поставлені завдання і свої функції якісно в різних умовах. Сюди також можна віднести надійність, безпеку та продуктивність. Додаток повинен швидко реагувати на події викликані системою, або ж користувачем.

Гнучкість. Додавання змін в існуючий функціонал не повинно викликати проблем та призводити до подальшої модифікації проекту. Враховуючи умови середовища Android, компоненти програми можуть запускатися окремо і не послідовно, а операційна система або користувач можуть знищити їх в будь-який час. Оскільки ці події не знаходяться під контролем додатку, компоненти програми не повинні залежати один від одного.

Масштабування. Можливість розширити або змінити поведінку системи без зміни чи переписування вже існуючих частин. Інакше кажучи, додавання нових функцій та класів не повинно порушувати основну структуру проекту. Дотримання цього принципу (Single-Responsibility Principle),

не слід зберігати будь-які дані в компонентах додатку. Компоненти програми не повинні залежати один від одного.

Архітектура MVVM в мобільних ігрових додатках

Протягом багатьох років, найпопулярнішим рішенням була архітектура MVP. MVP містить три головних компоненти: Model(збереження та обробка даних), View(користувацький інтерфейс) та Presenter(містить усю логіку користувацького інтерфейсу). Головним недоліком даної архітектури є нагромадження коду великою кількістю інтерфейсів, а також складністю використання класу Presenter, оскільки він пов’язаний з певним View.

Наразі набуває популярності архітектурне рішення MVVM, де замість компонента View використовується ViewModel. Цей підхід не має таких недоліків, які є в MVP та дозволяє легко досягнути незалежності між компонентами.

Ключовою особливістю підходу MVVM є те, що жоден компонент (Model, View, ViewModel) не знає про інший явно. Ці компоненти взаємодіють між собою за рахунок механізму зв'язування даних (DataBinding). При цьому зміна даних у ViewModel автоматично змінює дані, які відображаються у View. Аналогічно, будь-яка подія або зміна даних у View (натискання на кнопку, введення тексту та інше) змінює дані в ViewModel. Це дозволяє не зберігати явні посилання на View у ViewModel і навпаки, а також тримати ці компоненти дуже слабо пов'язаними, що зручно при тестуванні.

Компонент ViewModel реалізується з використанням однойменного класу. Цей клас призначений для зберігання та управління даними, пов’язаними з користувацьким інтерфейсом, у свідомому життєвому циклі. Гнучкість досягається за рахунок того, що на відміну від MVP, де кожен «презентер» напряму пов’язаний з відповідним «екраном» користувацького інтерфейсу, на один екземпляр ViewModel можуть бути підписані одночасно декілька View. В цьому допомагає два інструмента: LiveData та DataBinding.

LiveData – це бібліотека, яка представляє собою сховище даних, що працює за принципом паттерна Observer (спостерігач). Це сховище вміє робити дві речі:

- в нього можна помістити який-небудь об'єкт;
- на нього можна підписатися і отримувати об'єкти, які в нього поміщають.

Одною із найважливіших переваг використання бібліотеки - її вміння визначати активний підпісник чи ні, і відправляти дані лише активним підписникам.

DataBinding - це фреймворк від Google, який дозволяє зв’язувати компоненти користувацького інтерфейсу з джерелами даних в додатку. При цьому можна повністю позбутися від роботи з інтерфейсом користувача в коді.

ViewModel повністю відділяє логіку представлення від бізнес-логіки, чим забезпечує гнучкість та зручне масштабування. Використовуючи бібліотеку LiveData, компоненти ViewModel можуть отримувати дані від Model та надавати їх View, що в свою чергу оновлює користувацький інтерфейс без зайвого коду. Така система легко тестується, оскільки немає залежностей між компонентами. При необхідності внести зміни до користувацького інтерфейсу, наприклад оновити дизайн або ж додати нові анімації, компонент View може розглядатись, як повністю незалежний.

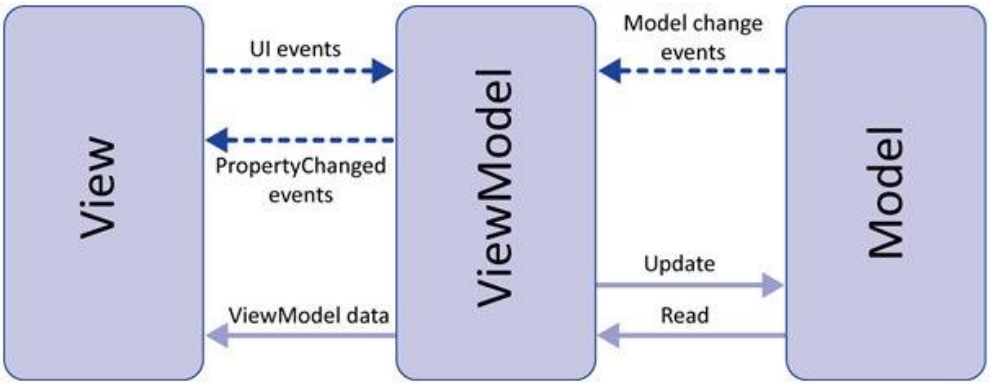

Рис. 1 – Приклад використання архітектури MVVM

Висновки

Використання запропонованого підходу дозволить розробнику створити надійний та гнучкий додаток-гру, з можливістю легкого масштабування та тестування. Незалежність компонентів дозволить без жодних труднощів додавати різні анімації та нові механіки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Guide to app architecture [Електронний ресурс]:[Веб-сайт] – Електронні дані. — Режим доступу: https://developer.android.com/jetpack/guide
2. Data Binding Library [Електронний ресурс]:[Веб-сайт] – Електронні дані. — Режим доступу: https://developer.android.com/topic/libraries/data-binding
3. LiveData Overview [Електронний ресурс]:[Веб-сайт] – Електронні дані. — Режим доступу: https://developer.android.com/topic/libraries/architecture/livedata

Пакула Антон Артурович — студент групи 1АКІТ-17б, факультет комп'ютерних систем і автоматики, Вінницький національний технічний університет, м .Вінниця, e-mail: anton.pakula.2000@gmail.com

Довгалець Сергій Михайлович — професор кафедри автоматизації та інтелектуальних інформаційних технологій, Вінницький національний технічний університет, м. Вінниця

Pakula Anton A. — student of group 1AKIT-17b, faculty of computer Systems and Automation, Vinnytsia National Technical University, Vinnytsia.

Dovgalets Sergeiy M. — Professor of Automation and Intelligent Information Technology department, Vinnytsia National Technical University, Vinnytsia.