УДК 681.3.07

С. М. Бурбело
Г. О. Черноволик
Л. М. Круподьорова
О. В. Гаврилюк
Н. Є. Барчук
Л. В. Райська

АНАЛІЗ ОСОБЛИВОСТЕЙ ЗАСТОСУВАННЯ «ЧИСТОЇ АРХІТЕКТУРИ» ANDROID-ДОДАТКІВ

Вінницький національний технічний університет

Анотація
У роботі розглянуто застосування принципів clean architecture при розробці мобільних додатків для платформи android.
Ключові слова: чиста архітектура, андроїд, додаток, правила залежностей,чистий код, програмування.

Abstract
The paper considers the application of the principles of clean architecture in the development of mobile applications for the android platform.
Keywords: clean architecture, android, application, dependency rule, clean code, programming.

Вступ

Написання якісного програмного забезпечення – справа не з легких. Окрім задоволення поставлених вимог, потрібно приділяти увагу надійності написаного коду, його тестуванню та гнучкості, щоб забезпечити адаптування до зростання та змін.

Існує досить велика кількість підходів для побудови складних систем з розвиненою архітектурою. Спільним для них залишається те, що всі вони задають різні способи розбиття продукту на окремі модулі. Тому з'являється поняття «чистої архітектури» як прогресивного підходу для розробки будь-якого програмного продукту.

Аналіз особливостей розробки «чистої архітектури» android-додатку

Архітектура android-додатку зазвичай не буває надто складною, проте принципи «чистої архітектури» (clear architecture) досить швидко набули популярності серед розробників мобільних додатків.

В основі Clear Architecture покладено ідеї декількох архітектурних підходів, які полягають у тому, що архітектура має підлягати [1]:

- тестуванню;
- не залежати від UI;
- не залежати від бази даних, зовнішніх фреймворків і бібліотек.

Це досягається шляхом розподілу на шари і слідуванню правилу залежностей (Dependency Rule). Відбувається поділ додатку на 3 ключових шари (рисунок 1):

- шар даних (Data Layer);
- шар бізнес-логіки (Domain Layer);
- шар уявлення (Presentation Layer).

При цьому задля забезпечення незалежності цих шарів на кожному з них використовується своя модель даних, яка конвертується при взаємодії між шарами.

У результаті з точки зору роботи з системою відбувається взаємодія з певними сутностями, які визначаються вимогами системи. За своєю суттю ці бізнес-об'єкти – це класи моделей з певними

методами або ж набір якихось структур даних. Саме вони повинні бути першочергово протестованими та незалежними від інших елементів, оскільки бізнес-логіка вважається внутрішнім шаром і змінюватись буде в останню чергу, відповідно коли буде прийняте рішення змінити сам зміст системи. Цей шар є певним об'єднанням шарів сценаріїв взаємодії і бізнес-логіки в оригінальній «чистій архітектурі». Саме до цього шару звертається шар уявлення для виконання запитів і отримання даних.

У шарі уявлення здійснюється перетворення даних з формату, який використовують бізнес-об'єкти або сценарії взаємодії, в формат, необхідний для роботи системи. Під роботою системи в даному випадку мається на увазі передача даних для відображення користувачеві або іншій службі. Саме цей шар прив'язується до екранів і допомагає організувати взаємодію з шаром бізнес-логіки і роботу з даними.

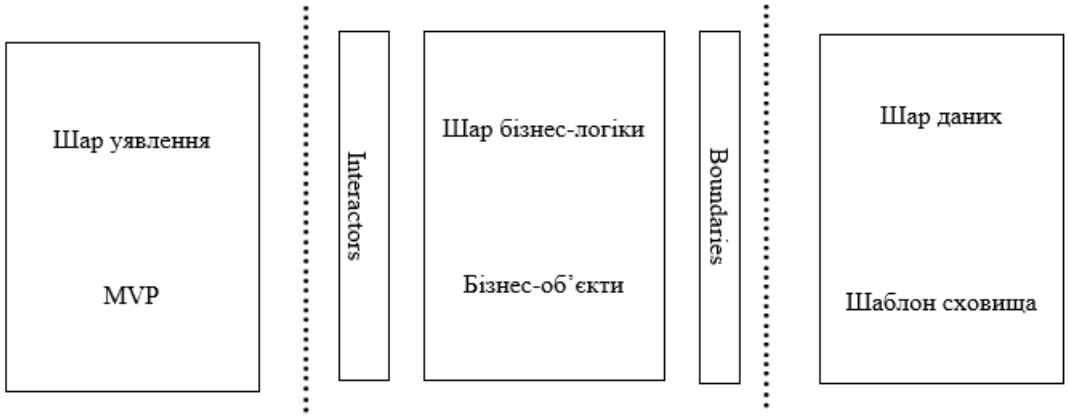

Рис. 1. Схема поділу на шари

Шар даних відповідає, в першу чергу, за отримання даних з різних джерел і їх кешування. Він реалізується за рахунок паттерна Repository, його загальна схема подана на рисунку 2:

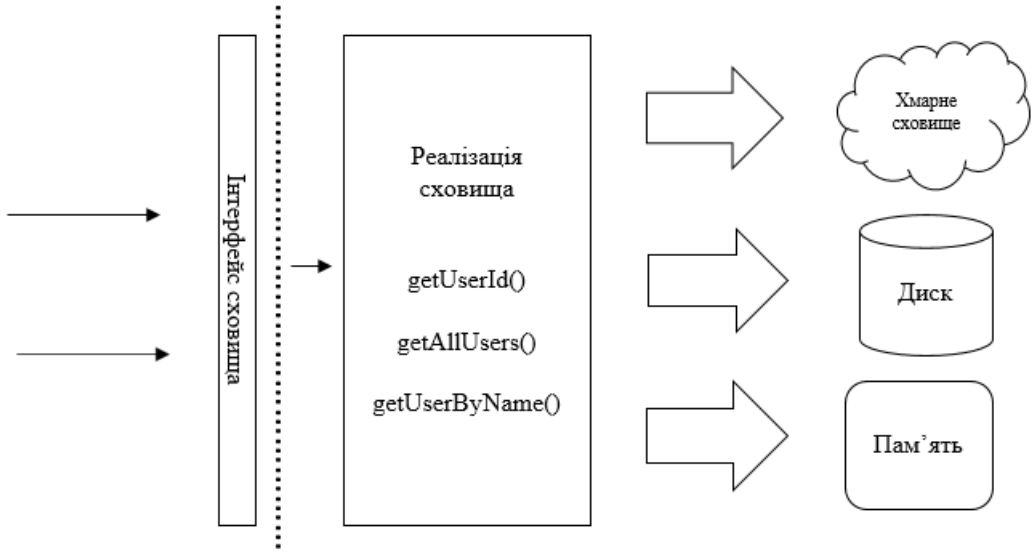

Рис. 2. Схема шару даних

Існує кілька плюсів від використання такого підходу. По-перше, інші шари, які запитують дані, не знають про те, звідки ці дані надходять. Більш того, їм не потрібно цього знати, тому що це ускладнює

логіку роботи і модуль бере на себе зайву відповідальність. По-друге, шар даних у такому випадку виступає єдиним джерелом інформації, що, як обговорювалося раніше, дуже зручно [2].

Висновок

В останні роки прослідковується тенденція до застосування принципів «чистої архітектури» (clean architecture) android-додатків. Така архітектура дозволяє змінювати оточення і тестувати модулі в різних умовах. Тут, у першу чергу, мова йде про принцип Dependency Injection. Код повинен бути чистішим і зрозумілішим. Коли вся логіка серверних запитів, логіка додатків, обробка взаємодії з користувачем і відображення даних знаходяться в одній Activity, стає досить важко розуміти і підтримувати такий код. Тому потрібно розділяти такі Activity на класи за логічними блоками: наприклад, клас для роботи з життєвим циклом, клас для відображення даних, клас, що містить логіку програми, і так далі. Такий підхід дозволяє спростити архітектуру android-додатків та забезпечити зручність її реалізації.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Fernando Cejas. Architecting Android...The clean way? [Електронний ресурс]. – Режим доступу: https://fernandocejas.com/blog/engineering/2014-09-03-architecting-android-the-clean-way/.
2. Архітектура андроїд програми. Clean Architecture. [Електронний ресурс]. – Режим доступу: https://www.fandroid.info/lektsiya-4-po-arhitekture-android-prilozheniya-clean-arcitecture.

Бурбело Сергій Михайлович **–** кандидат технічних наук, доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: smburbelo@gmail.com.

Черноволик Галина Олександрівна **–** кандидат технічних наук, доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: lina2433@gmail.com.

Круподьорова Людмила Михайлівна **–** старший викладач кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: krupodlm@gmail.com.

Гаврилюк Олена Віталіївна **–** асистент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: kafedra_pz_2105@ukr.net.

Барчук Наталія Євгенівна **–** асистент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: kafedra_pz_2105@ukr.net.

Райська Людмила Василівна – студентка групи 1ПІ-19б, Факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця**,** e-mail: milaraiska@gmail.com.

Sergii Burbelo – Ph.D., Associate Professor of Software Engineering, Vinnitsa National Technical University, Vinnitsa, e-mail: smburbelo@gmail.com.

Galyna Chernovolyk – Ph.D., Associate Professor of Software Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: lina2433@gmail.com.

Liudmyla Krupoderova – Senior Lecturer in Software Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: krupodlm@gmail.com.

Olena Gavruluik – Assistant of Software Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: kafedra_pz_2105@ukr.net.

Natalia Barchuk – Assistant of Software Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: kafedra_pz_2105@ukr.net.

Liudmyla Raiska – student of group 1ПІ – 19b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: milaraiska@gmail.com.