УДК **004.624**

Ставицький П.В.
Войтко В.В.

ОСОБЛИВОСТІ ПРОЕКТУВАННЯ ПРОГРАМНИХ СИСТЕМ НА БАЗІ МОБІЛЬНИХ ПЛАТФОРМ

Вінницький національний технічний університет

Анотація.
Розглянуто особливості проектування систем на базі мобільних операційних систем. Розглянуто архітектурний підхід, а також принципи та шаблони, на яких вони базуються. Проаналізовано можливості та шляхи реалізації презентаційного рівня систем. Розглянуто шляхи використання реактивної парадигми програмування при реалізації логіки взаємодії системи з логікою користувацького інтерфейсу.
Ключові слова: мобільна система, архітектура, шаблони проектування, реактивне програмування.

Abstract.
Features of mobile software systems are considered. An architectural approach and principles and design patterns on which it is based are described. Analyzed the ways of application of the reactive programming paradigm in the implementation of the communication layer between business logic and presentation logic of the software mobile system.
Keywords: mobile system, architecture, design patterns, reactive programming.

Вступ

Програмні системи, що базуються на мобільних платформах, мають низку особливостей, які необхідно враховувати при їх проектуванні. Однією з найважливіших задач при проектуванні мобільних систем є досягнення ефективного розділення відповідальності програмних компонентів. Серед основних логічних рівнів, на які можна поділити програму, виділимо:

- презентаційний рівень;
- рівень бізнес-логіки;
- рівень даних.

Крім того, додатково можна розглядати рівень платформи, що відповідає за реалізацію конкретного функціоналу на пристроях конкретної платформи.

Результати дослідження

Одним з можливих варіантів організації мобільної системи є використання підходу чистої архітектури «The Clean Architecture» [1]. Такий підхід базується на принципах SOLID [2], що описують розділення та взаємодію компонентів програмних систем. За допомогою такого підходу вдається відокремити якомога більшу кількість платформо-незалежного програмного коду, що в подальшому може бути окремо протестованим та використаним під час розробки платформної частини. Цей підхід передбачає розділення програми на декілька компонентів, серед яких платформо-незалежний рівень бізнес-логіки, а також рівні даних та презентації. Додатково може використовуватися рівень з функціоналом, притаманним конкретній операційній системі, пристрою тощо.

Схожим є підхід VIPER, що здебільшого використовується при розробці систем на базі операційної системи IOS, і його основні компоненти слідують підходу чистої архітектури.

Серед основних програмних компонентів, що використовуються у більшості архітектурних підходів, можна виокремити:

- Interactor – програмний компонент, що реалізує конкретний сценарій використання системи;
- Router – відповідає за навігацію між функціональними компонентами системи;

- Repository – реалізує абстрактний доступ до даних, що використовуються в системі без залежності від конкретної реалізації.

Крім того, важливими є компоненти презентаційної логіки, тобто користувацького інтерфейсу. Найпершим і найосновнішим підходом є Model-View-Controller [3]. В його основі лежить розділення логіки представлення та бізнес-логіки програмної системи. Взаємодія з користувацьким інтерфейсом відбувається за допомогою контролера. Подальшого розвитку даний шаблон набув у вигляді Model-View-Presenter, де контролюючий компонент виконує роль посередника між користувацьким інтерфейсом системи та її функціоналом.

У той час, як реактивна парадигма програмування набула широкої розповсюдженості, підходи Model-View-ViewModel [4] та Model-View-Intent отримали практичне застосування. Їх основною особливістю є інтерпретування процесу взаємодії з системою шляхом використання подій та потоків даних, на які реєструється підписка шляхом використання шаблону спостерігач. При використанні першого підходу відбувається спостерігання за подіями користувацького інтерфейсу. При реалізації другого – автоматизується взаємодія презентаційної логіки та бізнес-логіки системи шляхом спостерігання за «намірами» користувача при використанні того чи іншого сценарію використання додатку.

Альтернативним підходом організації вищевказаних компонентів мобільних систем є використання підходу, розробленого інженерами компанії Uber під назвою RIBs [5]. Цей підхід передбачає моделювання структури системи у вигляді дерев, де кожна з його вершин асоціюється з самодостатнім функціональним компонентом. Вершини-нащадки не залежать від предків і реалізують один з можливих сценаріїв використання мобільної системи.

Висновки

Таким чином, було проведено огляд особливостей проектування програмних систем на базі мобільних платформ, розглянуто основні логічні компоненти таких систем та їх взаємодію. Проаналізовано шляхи побудови архітектури мобільних програмних систем. Крім того, було деталізовано структуру презентаційного рівня мобільних додатків та можливих варіантів його реалізації. Розглянуто роль реактивної парадигми програмування як засобу взаємодії компонентів системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Martin R. Clean Architecture: A Craftsman’s Guide to Software Structure and Design / Robert C. Martin: Prentice Hall, 1 edition, 428, 2017
2. Martin R. Design Principles and Design Patterns / Robert C. Martin, Bernard Wand-Polak School of Engineering, 2000 – URL: https://fi.ort.edu.uy/innovaportal/file/2032/1/design_principles.pdf
3. Reenskaug T. The Model-View-Controller (MVC). Its Part and Present / Trygve Reenskaug, University of Oslo, 2003 – URL: http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern.pdf
4. Ставицький П. Організація архітектури додатків на базі мобільних платформ / П.В. Ставицький, В.В. Войтко. НТКП ВНТУ. Факультет інформаційних технологій та комп’ютерної інженерії : XLVI Науково-технічна конференція факультету інформаційних технологій та комп’ютерної інженерії, 2017 – URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2017/paper/view/2794/2522
5. RIBs Cross-Platform Mobile Architecture [Електронний Ресурс] – Режим Доступу: https://eng.uber.com/new-rider-app-architecture/

Ставицький Павло Валерійович – аспірант кафедри програмного забезпечення, Вінницький національний технічний університет, Вінниця, pavlo.stavytskyi@gmail.com .

Войтко Вікторія Володимирівна – к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, Вінниця, dekanfki@i.ua .

Pavlo Stavytskyi – graduate student of the department of higher mathematics, Vinnytsia National Technical University, Vinnytsia, pavlo.stavytskyi@gmail.com .

Voitko Viktoria – candidate of technical sciences, Associate Professor of the department of software engineering; Vinnytsia National Technical University, Vinnytsia, dekanfki@i.ua .