УДК 681.3.01

С.Г. Московко
І.В. Богач

АНАЛІЗ АРХІТЕКТУРНИХ ШАБЛОНІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вінницький національний технічний університет

Анотація

В роботі проведений аналіз існуючих архітектурних шаблонів, досліджені принципи та види різних архітектур, виділено основні критерії оцінки спроектованої архітектури, проведена оцінка переваг та недоліків поширених шаблонів, визначено підхід до вибору архітектури ПЗ відповідно до поставленої задачі.

Ключові слова: архітектура програмного забезпечення, архітектурний шаблон, багаторівневий шаблон, шаблон мікросервісів, шаблон MVC.

Abstract

In this work there are analyzed existing architectural patterns, studied the principles and types of different architectures, identified the main criteria for evaluating of the designed architecture, evaluated the advantages and disadvantages of common patterns, defined the approach to choosing software architecture according to the task.

Keyword: *software architecture, architecture pattern, layered pattern, microservices architecture pattern, MVC pattern.*

Вступ

Розробка програм – досить складний і трудомісткий процес, в якому проектування коректної і надійної архітектури грає ключову роль. Розподіл і координація зусиль зі створення програм в групі розробників часто виявляються найбільш відповідальними і важкими рішеннями, так як впливають на основний результат. Метою проектування архітектури є визначення внутрішніх властивостей програмного забезпечення і деталізація його зовнішніх властивостей на основі виданих замовником і проаналізованих вимог.

Метою даного дослідження є аналіз існуючих архітектурних рішень та виявлення переваг і недоліків їх використання у реальних проектах.

Результати дослідження

Правильно вибрана архітектура ПЗ допомагає зробити процес розробки та забезпечення програм більш простими та ефективними. Отже виділимо основні критерії для оцінки добре спроектованої архітектури:

- *Ефективність програми*. В першу чергу програма повинна вирішувати поставлені завдання і якісно виконувати свої функції при різних початкових умовах та в різних середовищах виконання. Сюди можна віднести такі характеристики, як надійність, безпеку, продуктивність, здатність справлятися зі збільшенням навантаження, тощо.
- *Змінність програми.* Будь-яка програма піддається змінам з плином часу. Чим швидше і зручніше можна змінити програму, тим вона гнучкіша. Зміна одного модуля програми не має впливати на інші модулі.
- *Можливість розширення програми.* Можливість додавати в програму новий функціонал, не порушуючи її основної структури. На першому етапі проектування в програму необхідно додавати тільки самий необхідний функціонал, але при цьому архітектура повинна дозволяти легко додавати новий функціонал в міру необхідності.

- *Можливість повторного використання.* Програму бажано проектувати так, щоб її модулі можна було повторно використовувати в інших проектах.
- *Можливість підтримки.* Архітектура ПО повинна давати можливість легко і швидко розбиратися в програмі, повинна бути структурована, не містити дублювання коду.

За отриманими критеріями оцінки проведемо аналіз та виділимо основні переваги та недоліки найбільш розповсюджених архітектурних шаблонів.

Багаторівневий шаблон (Layered pattern). Компоненти всередині багаторівневого шаблона організовані в горизонтальні рівні, кожен рівень виконує певну роль у програмі (наприклад, логіка представлення або бізнес-логіка). Хоча шаблон багаторівневої архітектури не визначає кількість та типи рівнів, які повинні існувати в шаблоні, більшість багаторівневих архітектур складаються з чотирьох стандартних рівнів: представлення, бізнесу, персистентності та бази даних (рис. 1).

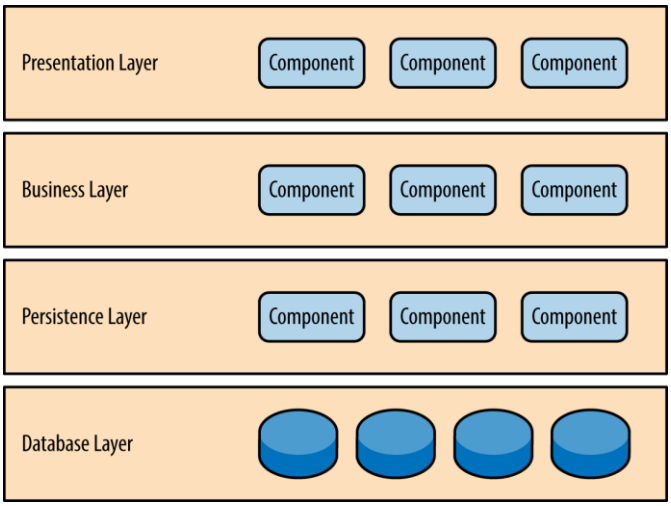

Рисунок 1 – Схематичне зображення багаторівневого шаблону

Однією з потужних особливостей багаторівневої архітектури є розподіл проблем між компонентами. Компоненти в межах певного рівня мають справу лише з логікою, яка стосується цього рівня. Наприклад, компоненти на рівні представлення мають справу лише з логікою представлення, тоді як компоненти, що перебувають на рівні бізнесу, мають справу лише з бізнес-логікою. Цей тип класифікації компонентів дозволяє легко вбудувати ефективні моделі ролей та моделей відповідальності у вашу архітектуру, а також спрощує розробку, тестування, управління та підтримку програм, що використовують цей шаблон архітектури, завдяки чітко визначеним інтерфейсам компонентів та обмеженому обсягу компонентів.

Недоліками даного підходу є низька ефективність через необхідності проходження декількох шарів архітектури для виконання бізнес-запиту і низька масштабованість через тісно пов’язану, та монолітну реалізацію цього шаблону.

Шаблон мікросервісів (Microservices architecture pattern). Незалежно від обраної топології чи стилю реалізації, існує кілька загальних основних концепцій, які застосовуються до загальної схеми архітектури. Перша з цих концепцій - це поняття окремо розгорнутих підрозділів. Як показано на рисунку 2, кожен компонент архітектури мікросервісів розгортається як окремий блок, що дозволяє полегшити розгортання за допомогою ефективного та впорядкованого конвеєра доставки, підвищеної масштабованості та високого ступеня роз'єднання програм та компонентів у вашому додатку.

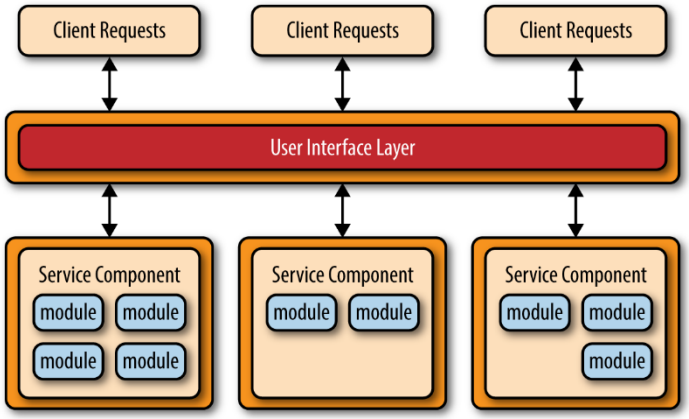

Рисунок 2 – Схематичне зображення мікросервісної архітектури

Ще однією ключовою концепцією архітектури мікросервісів є те, що це розподілена архітектура, що означає, що всі компоненти в архітектурі повністю відокремлені один від одного і доступ до них здійснюється за допомогою якогось протоколу віддаленого доступу (наприклад, JMS, AMQP, REST, SOAP, RMI тощо). Розподілена природа цієї архітектури полягає в тому, як вона досягає деяких зі своїх чудових масштабованих характеристик.

Єдиним недоліком даної архітектури є низька ефективність. Хоча можна створювати додатки, реалізовані на основі цього шаблону, які працюють дуже якісно, в цілому цей шаблон, природно, не піддається високопродуктивним додаткам через розподілений характер шаблону мікросервісів.

Шаблон MVC (Model-View-Controller pattern). Компоненти всередині шаблону MVC (модель даних програми, призначений для користувача інтерфейс і взаємодія з користувачем) розділені на три окремих частини так, що модифікація одного з них має мінімальний вплив на інші. Модель надає дані і методи роботи з ними, реагує на запити, змінюючи свій стан. Представлення відповідає за візуалізацію. Часто в якості представлення виступає форма з графічними елементами. Контролер забезпечує введення даних користувачем і використовує модель і представлення для реалізації необхідної реакції. Схема взаємодії компонентів наведена нижче (рис. 3).

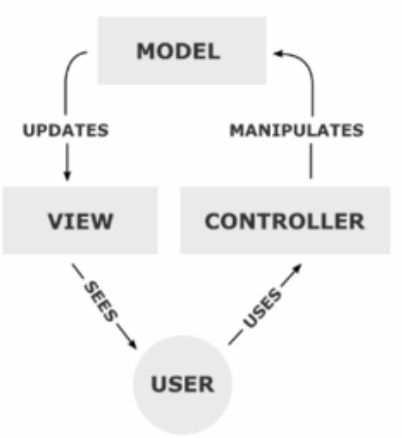

Рисунок 3 – Схематичне зображення шаблону MVC

Головною перевагою концепції MVC є поділ логіки управління додатка, отримання даних і їх відображення. MVC чітко відокремлює модель від користувача компоненти інтерфейсу і, отже, може бути кілька представлень впроваджено та використано з єдиною моделлю. MVC також дозволяє створювати "приєднувальні" представлення та контролери. Концептуальне розділення MVC дозволяє знайти заміну

представлення та об'єкта контролера. Користувацькі інтерфейси об'єкта можна навіть замінити під час виконання.

Детальний моніторинг структури MVC не завжди є найкращим способом побудувати інтерактивний додаток, оскільки використання окремих компонентів моделі, представлення та контролеру збільшують складність без отримання великої гнучкості. Використання цього шаблону також створює можливість надмірної кількості оновлень, що значно зменшує кінцеву продуктивність програми.

Висновки

Були розглянуті критерії оцінки добре спроектованої архітектури та виділені основні переваги, та недоліки поширених архітектурних шаблонів.

При проектуванні ПЗ розробник повинен керуватися пропонованими критеріями. При створенні великого проекту краще використовувати архітектуру, яка передбачає декомпозицію програмного проекту на модулі за функціональним призначенням. Даний спосіб побудови ПЗ дозволяє спростити процес програмування і налагодження проекту.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. What is a software architecture? [Електронний ресурс]. Електронні дані. URL: https://www.ibm.com/developerworks/rational/library/feb06/eeles/index.html

2. Караванов А. В., Иванов Н.Д. Архитектура программного обеспечения для высоконадежных систем / Космическое приборостроение. 2018. URL: http://www.journal-niss.ru/journal/archive/24/paper6.pdf.

3. Симонова О. Н. Шаблон проектирования MVC как эффективное средство построения архитектуры программной системы / О. Н. Симонова, Д. Н. Лясин. // Современные наукоемкие технологии. №5, 2014. URL: https://www.top-technologies.ru/pdf/2014/5-2/33980.pdf.

4. Plakalović D. Applying MVC and PAC patterns in mobile applications / D. Plakalović, D. Simić. // Journal of computing. – 2010.

5. Richards M. Software Architecture Patterns / M. Richards. – Sebastopol, CA: O'Reilly Media, 2015. – 47 с. – ISBN 9781491924242.

Московко Сергій Геннадійович — студент групи 1АКІТ-17б, факультет комп'ютерних систем і автоматики, Вінницький національний технічний університет, м.Вінниця, e-mail: fkca.1akit.msg@gmail.com.

Богач Ілона Віталіївна — доцент кафедри автоматизації та інтелектуальних інформаційних технологій, Вінницький національний технічний університет, м.Вінниця, e-mail: ilona.bogach@gmail.com.

Moskovko Serhii G. — Department of Computer System and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: fkca.1akit.msg@gmail.co.

Supervisor: ***Bogach Ilona V.*** — Associate Professor of Automation and Intelligent Information Technology Department, Vinnytsia National Technical University, Vinnytsia, e-mail: ilona.bogach@gmail.com.