УДК 004.41

Д. М. СІЧКАР, студентка, А.А. ТРАЧУК, канд. техн. наук, доц., С.В. БІЛАШЕНКО, асистент
Криворізький національний університет

СУЧАСНІ ТИПИ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У сфері програмного забезпечення вибір архітектури суттєво впливає на розробку додатків, їх масштабованість та зручність. Спосіб взаємодії компонентів впливає на продуктивність, легкість оновлення, безпеку та здатність системи до розвитку. Сучасні підходи до архітектури програмного забезпечення дозволяють програмним рішенням адаптуватися до мінливих ринкових умов і технічних вимог. Метою дослідження є сучасні типи архітектури програмного забезпечення, їх переваги та недоліки, а також порівняння їх для визначення оптимального вибору, виходячи зі специфіки проєктів. Різні архітектурні підходи дозволяють адаптувати систему до унікальних вимог розробки, забезпечити її масштабованість та постійну підтримку.

Архітектура програмного забезпечення є ключовим елементом у створенні будь-якого програмного забезпечення. Вона визначає організацію системи, а також те, як компоненти взаємодіють один з одним і зовнішніми системами. Як наслідок, архітектура програмного забезпечення дозволяє будувати надійні, ефективні та масштабовані системи. Проаналізуємо декілька архітектурних підходів, а саме монолітну та мікросервісну архітектури, Clean Architecture (чиста архітектура), Serverless (серверлесс) архітектуру.

Монолітна архітектура передбачає розробку всього додатку як єдиного цілого. Цей метод пропонує переваги на початковому етапі, але зі зростанням складності додатку виникають проблеми з масштабуванням та обслуговуванням.

Мікросервісна архітектура (мікросервіси) представляє собою сукупність невеликих та незалежних сервісів, які поєднуються в єдиний застосунок. Кожен з сервісів має свій власний процес але вони пов’язані між собою за допомогою механізмів: HTTP, gRPC, AMQP.

На відміну від монолітної, архітектура мікросервісів дозволяє будувати систему з невеликих автономних частин, забезпечуючи більшу гнучкість і масштабованість. Цей підхід, однак, вимагає ретельної конфігурації взаємодії між мікросервісами та управління інфраструктурою. Мікросервісну архітектуру більш доцільно використовувати для великих систем, тоді як монолітну - для невеликих додатків.

Clean Architecture, представляє собою архітектурний шаблон, який має на меті відокремити бізнес-логіку від деталей реалізації. В основі її лежить принцип залежності від абстракцій, а не від конкретних реалізацій. Вона сприяє створенню тестувальних, незалежних та підтримуваних систем. Цей підхід поділяє систему на шари, кожен з яких має свою чітку відповідальність. Внутрішні шари (бізнес-логіка) не залежать від зовнішніх (інфраструктура). Clean Architecture забезпечує гнучкість, адаптивність і легкість тестування, але вимагає більше зусиль на початковому етапі розробки та може призвести до збільшення кількості коду.

Серверлесс архітектура дозволяє розробникам зосередитися на написанні коду, не турбуючись про управління серверами. Обчислювальні ресурси надаються хмарними платформами, такими як AWS Lambda, Azure Functions, Google Cloud Functions, лише під час виконання коду, що дозволяє значно економити на інфраструктурі. Ця архітектура оптимально підходить для обробки подій, створення Application Programming Interface(API) та реалізації бекенду для мобільних додатків. Однак, слід враховувати потенційні проблеми з затримкою при першому запуску та налагодженням, оскільки логування та моніторинг можуть бути ускладнені. Цей підхід краще використовувати для додатків з великою кількістю незалежних операцій, які можуть масштабуватися окремо.

Архітектура програмного забезпечення є життєво важливим елементом у створенні ефективних і масштабованих додатків. Вибір оптимального типу архітектури в залежності від поставлених завдань дозволяє створювати надійні системи, які відповідають потребам бізнесу та користувачів, а також знижують ризики розробки та експлуатації програмних продуктів.