УДК 004.41

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

ВИКОРИСТАННЯ АРХІТЕКТУРНИХ ПАТЕРНІВ ДЛЯ ПРОЕКТУВАННЯ ДОДАТКІВ

Архітектурні патерни відіграють ключову роль у розробці програмного забезпечення, зокрема мобільних додатків. Вони використовуються для покращення організації коду, спрощення його підтримки, тестування та масштабування. Крім того, правильний вибір архітектурного патерну допомагає ефективно поєднати інтерфейс користувача, моделі даних і бізнес-логіку, що особливо важливо для складних мобільних застосунків.

Більшість мобільних додатків створюються з низьким рівнем структурованості та без чіткої архітектури, що може ускладнювати їхню підтримку. Використання відповідного патерну робить код структурованим і зручним для модифікації. Найпоширенішими архітектурними патернами для мобільних застосунків є Model-View-Controller (MVC), Model-View-Presenter (MVP), Model-View-ViewModel (MVVM) та Model-View-Intent (MVI).

MVC використовується для невеликих додатків або проєктів із простою логікою. Він поділяє програму на три основні компоненти: модель, представлення та контролер. Таке розділення покращує модульність коду та гнучкість. Модель зберігає та обробляє дані й бізнес-логіку, представлення відповідає за відображення інформації для користувача, а контролер керує введеними даними та визначає, як вони передаватимуться між моделлю та представленням. Це рішення просте у реалізації, але з ростом проєкту контролер може перевантажуватися.

У складніших застосунках, де важливе чітке розділення логіки та зручність тестування, доцільно використовувати MVP. Цей патерн, крім моделі та представлення, включає презентер, який не лише передає дані між моделлю та представленням, а й бере на себе відповідальність за обробку логіки відображення. Основна перевага цього патерну полягає в покращеному тестуванні інтерфейсу та можливості легкого внесення змін. Однак для невеликих проєктів MVP може виявитися занадто складним через необхідність написання додаткового коду.

Ще одним популярним рішенням є MVVM, який широко використовується у мобільній розробці, зокрема у фреймворках Android Jetpack та Xamarin. Ця архітектура розділяє код на модель, представлення і ViewModel, що містить всю логіку, необхідну для взаємодії представлення з даними, зменшуючи залежність між цими компонентами. Основною перевагою є двостороння прив’язка даних, яка спрощує оновлення інтерфейсу. Проте для невеликих проєктів MVVM може бути надмірно складним через додаткову логіку.

Для складних застосунків, які вимагають уніфікованого управління станом, часто використовується MVI. Основна ідея цього патерну полягає в потоковій обробці даних за допомогою намірів, що робить поведінку програми передбачуваною. MVI забезпечує чітке керування станом, зменшує побічні ефекти та покращує тестованість коду. Водночас, цей підхід потребує більше ресурсів, тому його складність виправдана лише у великих проєктах.

Використання оптимально обраного архітектурного патерну впливає на продуктивність, підтримку та тестування мобільного застосунку. Враховуючи обмежені ресурси мобільних пристроїв, він має бути оптимізованим для ефективного використання пам’яті та процесорної потужності. Також важливо забезпечити адаптивний інтерфейс, автономний режим роботи та високий рівень безпеки даних. Окрім цього, необхідно передбачити можливість оновлення та підтримки програми без втрати даних користувачів.

Отже, використання архітектурних патернів у мобільних додатках залежить від аналізу конкретних вимог проєкту, його масштабованості, необхідності тестування та переваг команди розробників. А оскільки технології швидко розвиваються, розробникам необхідно адаптувати свої підходи, експериментувати з новими інструментами та враховувати сучасні тенденції для створення надійних мобільних додатків.

Список літератури

1. Про архітектуру додатків. URL: https://foxminded.ua/arkhitektura-zastosunku/
2. Що таке Патерн? URL: https://refactoring.guru/uk/design-patterns/what-is-pattern
3. Architecture for Mobile Development.URL: https://www.geeksforgeeks.org/architecture-for-mobile-development-design-patterns/