УДК 004.41 **doi: 10.26906/SUNZ.2025.2.169-174**

В. Д. Первєєв, Т. В. Філімончук, А. М. Бугрій, С. О. Партика

Харківський національний університет радіоелектроніки, Харків, Україна

МОДЕЛЬ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ОС ANDROID

Анотація. Актуальність цього дослідження зумовлена зростаючими вимогами до продуктивності, гнучкості та можливості розширення мобільних застосунків у сучасних умовах за рахунок ефективного використання архітектурних рішень. Вибір фреймворку Flutter як середовища розробки обумовлений його можливістю забезпечувати швидкий цикл розробки, кросплатформну сумісність та інтеграцію платформозалежних сервісів. **Об'єктом дослідження** є архітектурна модель мобільного застосунку Android, яка може бути сформована обиранням окремих складових відповідно до потреб. Особливу увагу приділено можливостям розширення функціональності через впровадження додаткових модулів, таких як системи аутентифікації, аналітики та хмарних сервісів. **Предметом** дослідження виступає взаємодія компонентів мобільного застосунку в середовищі Flutter, а також вплив архітектурних рішень на продуктивність, масштабованість та ефективність розробки. У ході дослідження проаналізовано сучасні підходи до модульної архітектури мобільних застосунків, що забезпечують стабільність, швидкодію та гнучкість. **Результати** роботи демонструють, що запропонована модель здатна ефективно інтегрувати базові сервіси Android, підтримувати їх розширення та адаптацію до змінних потреб користувачів. **Висновки.** Дослідження сприяє визначенню оптимальних шляхів реалізації функціоналу мобільних застосунків, підвищенню їх продуктивності, масштабованості та конкурентоспроможності.

Ключові слова: модель мобільного застосунку, середовище розробки, Flutter, сервіси Android, модульність, продуктивність, безпека даних, масштабованість, тестування.

Вступ

Постановка проблеми. Розвиток інформаційних технологій набирає дедалі більших обертів, змінюючи різні сфери життя людини. Однією з найбільш прогресивних галузей на даний час стала розробка мобільних застосунків, які суттєво впливають на взаємодію людей із цифровими сервісами. Мобільні застосунки стали невід'ємною частиною повсякденного життя, надаючи користувачам доступ до різноманітних послуг у реальному часі. Водночас, високий рівень конкуренції на ринку програмного забезпечення вимагає від розробників забезпечення не лише інноваційного функціоналу, але й високої продуктивності, зручності та масштабованості своїх продуктів. Одним із провідних інструментів для кросплатформної розробки мобільних застосунків є фреймворк Flutter, який дозволяє створювати універсальні рішення для ОС Android та iOS. Використання Flutter надає розробникам можливість швидкої інтеграції платформозалежних сервісів, таких як геолокація, сповіщення, камера, а також реалізації унікального дизайну. Завдяки своєму інструментарію та архітектурі Flutter відкриває широкі перспективи для підвищення ефективності процесів розробки. Дана робота присвячена дослідженню складових моделі мобільного застосунку Android, яка орієнтована на використання фреймворку Flutter.

При розробці проєкту мобільного застосунку головну увагу слід приділити визначенню архітектурних компонентів моделі, які реалізують логіку застосунку та відповідають за зберігання даних, а також інтерфейсу користувача та інтерфейсу програмування, який інтегрується із платформозалежними сервісами.

Аналіз останніх досліджень і публікацій. У контексті дослідження можливостей розширення моделі мобільного застосунку варто звернути увагу на кілька важливих публікацій, де розглядаються ключові аспекти архітектури мобільних застосунків, зокрема їх модульний підхід, що дозволяє інтегрувати нові функції та адаптувати програму до сучасних вимог ринку. Особливу увагу приділено гнучкості архітектурних рішень, які забезпечують стабільність та масштабованість продукту.

У роботі [1] розглядаються основні принципи побудови архітектури мобільних застосунків з акцентом на їх ефективність та можливість масштабувати при необхідності. Автори підкреслюють значення модульного підходу, який дозволяє легко додавати нові функціональні можливості та адаптувати продукт до зростаючих потреб користувачів. Також зазначається важливість гнучкої архітектури для інтеграції нових сервісів, таких як хмарні рішення та інструменти аналітики, які забезпечують конкурентоспроможність та стійкість застосунку в умовах ринку, що швидко змінюється. У роботі [2] висвітлено процес побудови архітектури мобільних застосунків, який орієнтовано на використання модульного підходу, що забезпечує адаптивність та гнучкість системи. Автор наголошує на ролі окремих модулів, таких як інтерфейс користувача, який відповідає за візуальну складову та взаємодію з користувачем, логіку застосунку, що реалізує основний функціонал програми та базу даних, яка орієнтована на зберігання інформації. Особливе місце в роботі відведено інтерфейсу програмування, який виступає посередником між клієнтською та серверною частинами, забезпечуючи обмін даними. Також у роботі розглядається інтеграція модулів безпеки для організації конфіденційності та захисту інформації, а також пропонується можливість додавання необхідних стандартних розширень, які дозволяють легко впроваджувати новий функціонал. Завдяки такому напряму модульна архітектура спрощує масштабування застосунку, інтеграцію хмарних сервісів та підтримку змінних вимог ринку, що забезпечує конкурентоспроможність продукту. У роботі [3] розглядаються основні принципи створення мобільних застосунків із використанням архітектурних патернів для ОС Android та iOS.

Особливу увагу приділено аналізу шаблонів, таких як MVC (Model-View-Controller) та MVVM (Model-View-ViewModel), які дозволяють розробникам ефективно розподіляти функціональність між компонентами. У роботі описано переваги та недоліки цих архітектурних підходів, їх вплив на гнучкість та масштабованість кінцевих продуктів. У роботі також акцентовано на важливості модульного підходу, який забезпечує легку інтеграцію нових функціональних модулів та компонентів. Окремо розглянуто кросплатформні рішення, які дозволяють мінімізувати витрати та прискорити розробку, а також надаються рекомендації щодо вибору архітектури залежно від типу мобільного застосунку, його складності та потреб кінцевих користувачів.

У роботі [4] автори аналізують засоби та підходи до створення мобільного програмного забезпечення за допомогою мови програмування Kotlin. Особливий акцент зроблено на архітектурних рішеннях, таких як патерни MVC та MVVM, які забезпечують адаптивність, а також можливість масштабування застосунків за потреби. У роботі розглядається інтеграція базових сервісів, включаючи роботу з базами даних, інтерфейс програмного застосунку та клієнт-серверна архітектура. Автори наполягають на використанні при розробці мобільних застосунків модульного підходу, який спрощує додавання нових або модифікованих функцій та необхідного розширення можливостей додатку. Також підкреслюється важливість організації безпеки даних шляхом впровадження механізмів аутентифікації та авторизації. Завдяки використанню мови Kotlin та сучасних інструментів розробки, автори демонструють можливості створення ефективних та сучасних мобільних застосунків, що відповідають високим стандартам.

Робота [5] присвячена дослідженню архітектури мобільних застосунків з акцентом на безпеку. Автор аналізує основні вразливості архітектури, такі як небезпека втрати даних під час передачі, ненадійна автентифікація користувачів та низька стійкість до атак. У роботі особливу увагу приділено модулю безпеки, який інтегрується в загальну архітектуру мобільного застосунку. Автор пропонує використання систем з нульовим знанням для забезпечення конфіденційності та цілісності даних, що передаються між клієнтом та сервером, що дозволяє значно підвищити рівень захисту без порушення загальної функціональності застосунку. Інші модулі архітектури, такі як інтерфейс користувача, бізнес-логіка та база даних, інтегруються із модулем безпеки для забезпечення комплексного підходу до захисту застосунку.

У роботі [6] автори аналізують сучасні підходи до побудови архітектури мобільних застосунків для платформи Android. Робота зосереджується на проєктуванні модульної архітектури з використанням середовища розробки Android Studio та мови програмування Java. Особливу увагу приділено інтеграції компонентів, таких як інтерфейс користувача, база даних та бізнес-логіка, які забезпечують функціональність застосунку. Наведене дослідження також акцентує на важливості модульного підходу при розробці застосунку, який дозволяє адаптувати його до потреб користувачів. Крім того, автори розглядають особливості проєктування інтерфейсу та організації структури даних, підкреслюючи їх значення для забезпечення гнучкості та стабільності мобільного застосунку.

Проаналізувавши роботи авторів, які було наведено вище, модель мобільного застосунку можна визначити як структурований план, що включає технічні, функціональні та архітектурні елементи [7,8]. У середовищі Flutter така модель [9] може виглядати як набір компонентів (1):

$$M = \{ІК, МЛЗ, МОЗД, ІПЗ\}, \quad (1)$$

де *ІК* – інтерфейс користувача, *МЛЗ* – модуль логіки застосунку, *МОЗД* – модуль, що відповідає за організацію зберігання даних, *ІПЗ* – інтерфейс програмування застосунку.

Ця модель забезпечує основну взаємодію між користувачем та мобільним застосунком, але для більш складних задач вона може бути недостатньою. Наприклад, простота базової моделі може не враховувати взаємодії між компонентами, які впливають на стабільність та можуть обмежувати можливості розширення у подальшому [10]. Слід розуміти, що розробка мобільних застосунків є багатоступеневим процесом, який починається з ідеї та переходить від стадії аналізу до подальшого проєктування, розробки та тестування. На кожному етапі важливу роль відіграє архітектура, яка визначає порядок реалізації встановлених задач, строки виконання проєкту та його кінцеву вартість. Архітектура сприяє підвищенню продуктивності та масштабованості, що є особливо актуальним у кросплатформній розробці [11].

Метою цієї роботи є вдосконалення існуючої моделі для розробки мобільних застосунків через інтеграцію базових сервісів Android та реалізацію розширень у середовищі Flutter. Особливу увагу приділено адаптивності архітектури до змінних потреб користувачів та технологічних тенденцій, які забезпечать ефективність, гнучкість та масштабованість застосунку через впровадження модульного підходу та інтеграцію платформозалежних сервісів.

Основна частина

На основі проведеного аналізу робіт було виявлено, що більшість розглянутих моделей мобільних застосунків, орієнтовані на розробку для конкретної платформи, мають низку обмежень, зокрема складність у підтримці кросплатформного функціоналу, зростаючі витрати на розробку та недостатню гнучкість у розширенні. У зв'язку з цим у роботі запропоновано модель, яка базується на використанні фреймворку Flutter як основного середовища розробки, і складається з наступних складових (2):

$$M = \{ІК, МЛЗ, МОЗД, ІПЗ, МБД, МРФМЗ, МТ, ІТ\}, \quad (2)$$

де *ІК* – інтерфейс користувача, *МЛЗ* – модуль логіки застосунку, *МЗД* – модуль зберігання даних, *ІПЗ* – інтерфейс програмування застосунку, *МБД* – модуль безпеки даних, *МРФМЗ* – розширення функціональних можливостей застосунку, *МТ* – модуль тестування, *ІТ* – інструменти тестування.

У порівнянні з базовою моделлю (1) модель, що запропонована (2), значно розширена та включає нові

складові, які забезпечують повноцінну архітектуру мобільного застосунку. Інтерфейс користувача (ІК) є центральним елементом будь-якого мобільного застосунку, оскільки забезпечує взаємодію користувача із системою і будується на множині (3):

$$ІК = \{ДКІ, ОМ, КІ, СР, МП\}, \quad (3)$$

де *ДКІ* – дизайн-концепція інтерфейсу, *ОМ* – XML-файли для опису макетів, *КІ* – компоненти інтерфейсу: кнопки, списки, анімації, які забезпечують інтуїтивність та зручність використання, *СР* – середовище розробки, *МП* – мова програмування.

У фреймворку Flutter інтерфейс користувача будується за допомогою набору віджетів, які гарантують гнучкість та адаптивність. Віджети дозволяють створювати багатофункціональні інтерфейси, що автоматично налаштовуються під різні роздільні здатності екранів, що значно покращує зручність використання. Дизайн-концепція інтерфейсу (ДКІ) охоплює створення візуального та функціонального стилю інтерфейсу. При розробці мобільних застосунків можливо використовувати UIKit або SwiftUI, які надають розробникам глибокий контроль над елементами інтерфейсу та їх поведінкою. У Flutter даний підхід реалізується за допомогою інструментів, які дозволяють створювати гармонійний дизайн із інтерактивними елементами, такими як кнопки, списки чи навігаційні панелі, забезпечуючи послідовну логіку досвіду користувача. Дуже часто при розробці мобільних застосунків використовуються XML-файли, які дозволяють здійснювати опис макетів та структур інтерфейсу, зокрема в ОС Android. За допомогою цих файлів (ОМ) у розробників є можливість задавати компоненти інтерфейсу, як-от текстові поля, кнопки, зображення, та визначати їхні взаємозв'язки, розміри та розташування, шо спрощує налаштування й масштабування інтерфейсу у подальшому.

Компоненти інтерфейсу (КІ), як правило, включають базові елементи, такі як кнопки, текстові поля, графічні об'єкти, а також анімації та транзіції. У фреймворку Flutter ці компоненти представлені у вигляді віджетів, які забезпечують модульний підхід до розробки. Наприклад, віджети AnimationController та Hero дозволяють створювати плавні переходи між екранами та анімувати елементи, покращуючи досвід користувача та роблячи інтерфейс динамічним і привабливим. Android Studio та Visual Studio Code – два основні середовища розробки (СР) у Flutter. Android Studio забезпечує повну інтеграцію з Flutter та мовою Dart, має вбудований емулятор, потужні інструменти для налагодження, але споживає більше ресурсів та працює повільніше. Visual Studio Code легший, швидший, має гнучку систему розширень, але не містить вбудованого емулятора та менш функціональний у плані аналізу продуктивності. Якщо потрібна повноцінна інтеграція з Android, краще використовувати середовище розробки Android Studio, а для легкої та швидкої роботи з Flutter – VS Code. Слід також зазначити, що багато розробників комбінують обидва середовища залежно від задач, на які вони орієнтовані.

Мовою програмування (МП) у Flutter є Dart, яка забезпечує швидку компіляцію коду та зручну інтеграцію з платформозалежними компонентами. Використання даної мови дозволяє реалізовувати складну логіку взаємодії між елементами інтерфейсу та забезпечує їхню інтерактивність. Іноді у розробників виникає потреба писати код за допомогою мов Java/Kotlin (для ОС Android) або Swift/Objective-C (для iOS), коли необхідно взаємодіяти з нативними API через платформні канали. Використання цього підходу зумовлено тим, що на даний час не існує Flutter-плагіну, який реалізує потрібну функціональність. Також в якості мов програмування можливо використовувати C/C++ для роботи з низькорівневими бібліотеками (наприклад, OpenCV, FFmpeg та ін). Модуль логіки застосунку (МЛЗ) визначає бізнес-логіку, яка реалізує основний функціонал кінцевого продукту (4), включаючи обробку даних, управління бізнес-процесами та їх передачу між клієнтською (КЧ) та серверною частинами (СЧ). Також слід зазначити, що не останню роль в цьому модулі грає сервер БД (СБД):

$$МЛЗ = \{КЧ, СЧ, СБД\}. \quad (4)$$

Модуль бізнес-логіки у мобільному застосунку визначає перелік основних функцій, які забезпечують його роботу. У Flutter логіка організовується через патерни управління станом, такі як BLoC та Provider, які ізолюють бізнес-логіку від інтерфейсу користувача, що забезпечує можливість повторного використання коду, спрощує тестування та обслуговування. Наприклад, у застосунках для електронної комерції бізнес-логіка відповідає за управління кошиком, обробку платежів та оновлення даних про замовлення в реальному часі. Клієнтська частина (КЧ) застосунку є фронтальною частиною, яка реалізує взаємодію користувача з системою (тут виконуються базові операції, такі як введення даних, відображення інформації та взаємодія з API). Завдяки використанню фреймворку Flutter клієнтська частина забезпечує швидке завантаження, адаптивний дизайн та інтерактивність, що робить досвід користувача зручним та привабливим. Серверна частина (СЧ) забезпечує обробку запитів, виконання бізнес-логіки на стороні серверу та управління передачею даних між клієнтською частиною та базою даних. Наприклад, сервер може обробляти запити на авторизацію, здійснювати обчислення, зберігати дані про транзакції та повертати результати в клієнтський застосунок. Висока продуктивність та надійність серверної частини є ключовою особливістю для забезпечення стабільної роботи застосунку. Сервер бази даних (СБД) відповідає за зберігання, управління та забезпечення доступу до даних. У сучасних мобільних застосунках бази даних можуть бути локальними, такими як SQLite або хмарними, такими як Firebase Realtime Database. Сервер бази даних забезпечує швидкий доступ до інформації, підтримує масштабованість та можливість обробки великих обсягів даних, що є критичним для інтерактивних та масштабованих систем.

Модуль, орієнтований на організацію зберігання даних (МОЗД) може включати наступні компоненти для управління даними (5): таблиці (Т), бази даних (БД), системи управління базами даних (СУБД), а також хмарні сховища (ХС). Використання

наведених складових забезпечує надійне зберігання, безперебійний доступ до даних та їх синхронізацію:

$$МОЗД = \{Т, БД, СУБД, ХС\}. \quad (5)$$

Таблиці (Т) виступають базовими структурами, де організовуються дані у вигляді рядків та стовпців. Вони забезпечують просту та ефективну організацію даних, дозволяючи виконувати базові операції, такі як пошук, фільтрація та сортування. Бази даних (БД) є більш складними структурами, що включають множину таблиць та підтримують зв'язки між ними. Наприклад, для мобільного застосунку бази даних можуть використовуватися для зберігання профілів користувачів, історії транзакцій або інформації про товари. Сучасні мобільні застосунки зазвичай застосовують як реляційні бази даних (SQLite), так і нереляційні (Firebase Firestore) залежно від потреб проєкту.

Системи управління базами даних (СУБД) забезпечують інструменти для управління базами даних та доступу до них. Вони дозволяють створювати, змінювати, видаляти структури даних, виконувати складні запити та управляти транзакціями. Наприклад, для локальних рішень часто використовується SQLite, а для хмарних – Firebase Realtime Database або Amazon DynamoDB, які дозволяють забезпечити синхронізацію в реальному часі.

Хмарні сховища (ХС) надають можливість зберігати дані в хмарі, забезпечуючи доступ до них з будь-якого пристрою та підтримуючи синхронізацію між клієнтськими застосунками. Наприклад, Google Cloud Storage або Amazon S3 дозволяють інтегрувати зберігання файлів, мультимедіа або великих обсягів даних у мобільний застосунок, що забезпечує масштабованість та стабільність навіть за умов високих навантажень, що робить хмарні сховища невід'ємною частиною сучасних мобільних систем. Інтерфейс програмування застосунків (ІПЗ) відповідає за взаємодію між клієнтською та серверною частинами застосунку (6), забезпечуючи передачу даних та виконання відповідних операцій завдяки використанню протоколів передачі даних (ППД), форматів даних (ФД), методів запитів (МЗ) та проміжного програмного забезпечення (ППЗ) для аутентифікації, валідації та логування:

$$ІПЗ = \{ППД, ФД, МЗ, ППЗ\}. \quad (6)$$

Протоколи передачі даних (ППД), такі як HTTP/HTTPS, WebSockets або gRPC визначають правила обміну інформацією між клієнтом та сервером. HTTPS, зокрема, гарантує безпечність передачі даних, використовуючи шифрування для захисту від стороннього доступу. Формати даних (ФД) задають структуру інформації, яка передається між компонентами системи. Найпоширенішими форматами на даний час є JSON, XML та Protocol Buffers (Protobuf). JSON забезпечує простоту та читабельність коду, що робить його популярним для роботи з API, тоді як Protobuf надає високу ефективність завдяки компактному представленню даних. Методи запитів (МЗ) дозволяють клієнту виконувати різні операції на сервері. Наприклад, метод GET використовується для отримання інформації, POST – для створення нових ресурсів, PUT або PATCH – для оновлення даних, а DELETE – для видалення. Ці методи є основою взаємодії між клієнтом та сервером, забезпечуючи гнучкість у реалізації функціональності.

Проміжне програмне забезпечення (ППЗ) додає додаткові рівні обробки запитів, зокрема аутентифікацію, логування та валідацію даних. Наприклад, аутентифікація гарантує, що тільки авторизовані користувачі отримають доступ до ресурсів, логування дозволяє відстежувати активність запитів та відповідей, а валідація забезпечує перевірку коректності переданих даних, запобігаючи можливим помилкам і зловживанням. Ці елементи роблять ІПЗ критично важливим компонентом для стабільної та безпечної роботи мобільного застосунку. Модуль безпеки даних (МБД) може забезпечувати захист даних у застосунку (7) через ряд інструментів: шифрування (Ш), безпечну передачу даних (БПД), захист від шкідливих атак (ЗША), аутентифікацію та авторизацію (АА), а також механізми відновлення даних (МВД):

$$МБД = \{Ш, БПД, ЗША, АА, МВД, К\}. \quad (7)$$

Модулі безпеки в архітектурі мобільних застосунків відіграють важливу роль у забезпеченні захисту даних та безпечної роботи системи. Процес шифрування (Ш) є одним із ключових елементів, який гарантує конфіденційність інформації, що передається між клієнтом та сервером, який забезпечує захист даних від несанкціонованого доступу за допомогою таких методів, як AES або RSA, що широко використовуються в сучасних мобільних застосунках.

Безпечна передача даних (БПД) забезпечується за рахунок використання протоколів, таких як HTTPS, що гарантують шифрування трафіку. Додатково, перевірка сертифікатів та мінімізація обсягу переданих даних сприяють підвищенню рівня захисту.

Захист від шкідливих атак (ЗША) включає механізми обфускації коду, мінімізацію видимості критичних компонентів та регулярний аналіз безпеки для виявлення потенційних загроз. Аутентифікація та авторизація (АА) є основою управління доступом до системи. Аутентифікація визначає особу користувача за допомогою токенів, API-ключів або багатофакторної перевірки, тоді як авторизація обмежує доступ до функцій відповідно до ролей користувачів, що запобігає несанкціонованому використанню ресурсів застосунку. Механізм відновлення даних (МВД) передбачає створення резервних копій інформації та розробку планів відновлення у разі збоїв системи, що забезпечує цілісність та доступність даних у разі критичних помилок. Кешування (К) використовується для тимчасового зберігання даних, що дозволяє зменшити навантаження на сервер та прискорити доступ до інформації, яка часто використовується.

Модуль розширення функціональних можливостей застосунку (МРФМЗ) може бути реалізовано за рахунок інтеграції нових функцій (8), таких як додаткові модулі аналітики (ДМА), інтеграція зі сторонніми сервісами (СС) або нові інтерфейсні елементи (ІЕ), що в свою чергу забезпечує гнучкість системи через інтеграцію нових модулів та інструментів:

$$МРФМЗ = \{ДМА, СС, ІЕ\}. \quad (8)$$

Під час розробки мобільного застосунку, як правило, виникають помилки в його функціонуванні.

Для вчасного їх виявлення та подальшого усунення, необхідно проводити тестування.

Комплексне тестування застосунку за рахунок додавання модулю тестування (МТ) гарантує стабільну та швидку роботу в майбутньому, сприяє ефективності та продуктивності розробки (9) за рахунок використання різноманітних типів тестів: функціональне тестування (ФТ), тестування юзабіліті (ТЮ), тестування продуктивності (ТП), тестування безпеки (ТБ):

$$МТ = \{ФТ, ТЮ, ТП, ТБ\}. \quad (9)$$

Функціональне тестування (ФТ) спрямоване на перевірку відповідності застосунку функціональним вимогам. Воно охоплює перевірку коректної роботи всіх модулів та функцій, таких як авторизація, обробка даних та відображення інформації в інтерфейсі користувача.

Тестування юзабіліті (ТЮ) аналізує зручність використання інтерфейсу та логіку взаємодії з користувачем. Цей етап дозволяє виявити проблеми, які можуть перешкоджати інтуїтивному розумінню та використанню застосунку. Юзабіліті-тестування забезпечує створення продукту, який відповідає потребам користувачів.

Тестування продуктивності (ТП) оцінює швидкість роботи застосунку, ефективність використання ресурсів пристрою та здатність обробляти високі навантаження, що дозволяє виявити вузькі місця, такі як повільні запити до серверів або проблеми з продуктивністю бази даних.

Тестування безпеки (ТБ) спрямоване на виявлення вразливостей у системі захисту застосунку. Воно включає перевірку надійності аутентифікації, авторизації, шифрування даних та стійкості до атак, таких як SQL-ін'єкції або XSS (міжсайтові сценарії).

Процес тестування неможливий без використання інструментів тестування (ІТ). Слід зазначити, що на даний час існує два метода тестування мобільних застосунків: ручне та автоматизоване. Для отримання найкращого кінцевого результату слід комбінувати ці підходи (9):

$$ІТ = \{ІАТ, ІРТ, ІТП, ІТБ\}, \quad (10)$$

де *ІАТ* – інструменти автоматизованого тестування, *ІРТ* – інструменти ручного тестування, *ІТП* – інструменти тестування продуктивності, *ІТБ* – інструменти тестування безпеки.

Інструменти автоматизованого тестування (ІАТ) використовують для проведення трудомістких тестів, які дозволяють отримати швидкі, ефективні та точні результати. Слід зазначити, що одночасно можливо проводити декілька таких тестів на різних пристроях, що, в свою чергу, прискорює процес перевірки працездатності мобільного застосунку.

Інструменти ручного тестування (ІРТ) дозволяють отримати досвід користувача конкретної людини.

Також слід розуміти, що використання автоматизованих сценаріїв тестування для невеликих проєктів може виявитися надто затратним, як за людським ресурсом, так і в грошовому еквіваленті.

Інструменти тестування продуктивності (ІТП) забезпечують автоматизацію перевірок ефективності роботи застосунку. Наприклад, Firebase Performance Monitoring допомагає аналізувати швидкодію, тоді як Apache JMeter дозволяє моделювати навантаження на сервер для виявлення критичних точок.

Інструменти тестування безпеки (ІТБ) допомагають аналізувати вразливості застосунку. OWASP ZAP або Burp Suite можуть бути використані для перевірки безпеки мережних запитів, API та загальної стійкості системи до атак. Цей тип інструментів дозволяє своєчасно виявити та усунути потенційні загрози.

Завдяки комплексному підходу до тестування за допомогою складових модулю (9) забезпечується стабільність, продуктивність та безпека мобільного застосунку, що є критично важливим для успішного функціонування на ринку.

Висновки

В результаті проведених досліджень встановлено, що запропонована архітектурна модель мобільного застосунку, реалізована у середовищі Flutter, забезпечує високий рівень гнучкості, продуктивності та масштабованості.

Інтеграція базових сервісів Android, дозволяє створювати функціональні рішення, адаптовані до сучасних потреб користувачів.

Головною метою при розробці конкретного мобільного застосунку є визначення архітектурних компонентів моделі: для реалізації логіки застосунку, збереження даних, розробки інтерфейсу користувача та використання інтерфейсу програмування мобільних застосунків, який інтегрується із платформозалежними сервісами. Важливим аспектом моделі, що запропонована, є можливість розширення функціоналу через інтеграцію сторонніх модулів, що дозволяє швидко реагувати на зміни ринкових умов та потреби користувачів.

Використання модулю безпеки, який включає аутентифікацію, авторизацію та шифрування даних, гарантує конфіденційність та цілісність інформації, що є критичним для застосунків, які працюють із чутливими даними.

Використання інструментів моніторингу та аналітики дозволить підвищити продуктивність застосунку, виявляти можливі проблеми та своєчасно їх вирішувати.

Хмарні сервіси також надають додаткові переваги у зберіганні даних, синхронізації між пристроями та оптимізації витрат на інфраструктуру.

Загалом, запропонована модель мобільного застосунку демонструє свою ефективність у забезпеченні якісного кінцевого продукту, який відповідає вимогам користувачів та сучасним стандартам. Її гнучкість, адаптивність та надійність роблять її перспективною для широкого використання у розробці мобільних додатків на платформі Android.

Загальний висновок полягає в тому, що в залежності від цілей та направленості мобільного застосунку, який розробляється, може бути обрано різні напрями його проєктування, реалізації, інструментарію та подальшого тестування.

СПИСОК ЛІТЕРАТУРИ

1. Бідник Д.С., Цибульник С.О. (2021), “Архітектура мобільного додатку”, Матеріали XIV Всеукраїнської НПК студентів, аспірантів та молодих вчених “Погляд у майбутнє приладобудування”. К: КПІ ім. Ігоря Сікорського. С. 19-22.
2. Сідельнікова Д.С. (2022), “Етапи створення дизайну мобільного застосунку”, Мультимедійні технології в освіті та інших сферах діяльності: науково-практична конференція з міжнародною участю. К.: НАУ. С.105-109.
3. Ставицький П.В., Войтко В.В. (2017), “Організація архітектури додатків на базі мобільних платформ”, Матеріали XLVI науково-технічної конференції підрозділів ВНТУ, Вінниця. Електрон. текст. дані. URL: https://conferences.vntu.edu.ua/ index.php/all-fitki/all-fitki-2017/paper/view/2794.
4. Козуб Г.О., Козуб Ю.Г., Могильний Г.А., Жуков А.В. (2021), “Розробка мобільного Aandroid-додатку з застосуванням принципів Clean Architecture”, *Вісник Східноукраїнського національного університету імені Володимира Даля*. №5 (269). С. 5-10. doi: https://doi.org/10.33216/1998-7927-2021-269-5-5-10.
5. Санак О.Є. (2018), Захист мобільних застосунків на основі систем з нульовим знанням: магістерська дис.: 125 Кібербезпека. Київ. 121 с.
6. Дворецький М.Л., Нездолій Ю.О., Дворецька С.В., Кандиба І.О. (2021), “Розробка мобільних застосунків для OS Android”: навч. посіб. Миколаїв : Вид-во ЧНУ ім. Петра Могили. 140 с.
7. Яшина К.В., Ялова К.М., Сугаль Є.О. (2017),Огляд гнучких методологій розробки програмного забезпечення”, *Збірник наукових праць Дніпровського державного технічного університету*. Том 1, №30. С. 153-156.
8. Вавіленкова А.І. (2021), “Аналіз гнучких методологій розробки програмного забезпечення для реалізації у командних проєктах”, *Вісник Національного технічного університету "ХПІ"*. Харків: НТУ "ХПІ", 2021. №1 (7). С. 39-46.
9. Сеспедес Гарсія Н.В., Сеспедес Гарсія П.Д. (2023), “Моделі життєвого циклу розробки програмного забезпечення”, *Молодий вчений*, №2 (114). С. 17-20. doi: https://doi.org/10.32839/2304-5809/2023-2-114-4.
10. Sommerville I. (2015), “Software Engineering”. Publisher: Pearson, 10th edition. 816 p.
11. Pressman R.S. (2019), “Software Engineering: A Practitioner's Approach”. Publisher: McGraw-Hill Higher Education; 9th edition. 692 p.

Received (Надійшла) 03.04.2025
Accepted for publication (Прийнята до друку) 28.05.2025

ВІДОМОСТІ ПРО АВТОРІВ/ ABOUT THE AUTHORS

Первєєв Володимир Дмитрович – магістрант кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Volodymyr Pervieiev – master’s student of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: volodymyr.pervieiev@nure.ua; ORCID Author ID: https://orcid.org/0009-0004-9776-1283.

Філімончук Тетяна Володимирівна – кандидат технічних наук, доцентка, доцент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Tetiana Filimonchuk – PhD, Associate Professor of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: tetiana.filimonchuk@nure.ua; ORCID Author ID: http://orcid.org/0000-0002-4380-504X;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57190949991.

Бугрій Андрій Миколайович – кандидат технічних наук, старший викладач кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Buhrii Andrii – Associate Professor of the Department of Electronic Computers, Senior lecturer at the Department of Electronic Computers, Kharkiv National University of Radio Electronics; Kharkiv, Ukraine;
e-mail: andrii.buhrii@nure.ua; ORCID Author ID: http://orcid.org/0009-0002-9059-3200.

Партика Станіслав Олександрович – старший викладач кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Stanislav Partyka – Senior lecturer of the Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: stanislav.partyka@nure.ua; ORCID Author ID: http://orcid.org/0000-0002-7376-8980;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57204560890.

Model of an android mobile application with integration of basic services and extensions

Volodymyr Pervieiev, Tetiana Filimonchuk, Andriy Buhrii, Stanislav Partyka

Abstract. The relevance of this research is driven by the growing demands for the performance, flexibility, and scalability of mobile applications in modern conditions. The choice of Flutter as the development environment is justified by its ability to provide a fast development cycle and cross-platform compatibility**. The object of the study** is the architectural model of an Android mobile application that includes the integration of basic services such as geolocation, notifications, and the camera, offering approaches for extending functionality through the addition of new modules. The model consists of interconnected components, including the user interface, business logic, access to ШЗs, and platform-specific services. Particular attention is paid to the modularity of the architecture, ensuring the model's adaptability to changing user needs and technological trends. **The subject of the study** is the interaction of components within the mobile application model in the Flutter environment. The study examines the impact of architectural decisions on performance, scalability, and development efficiency. **Results**. The proposed model demonstrates the ability to integrate basic Android services and support their extensions, ensuring stability and high product quality. **Conclusions**. Research into the mobile application model in the Flutter environment helps identify optimal ways to implement functionality, improve performance, and enhance the competitiveness of mobile applications.

Keywords: mobile application, application model, Flutter, architectural model, Android services, modularity.