Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель Android-застосунку з використанням базових сервісів

(тема)

Виконав:

здобувач 2 року навчання,

групи СПм-23-3

Володимир ПЕРВЄЄВ
(власне ім'я, прізвище)

Спеціальність

123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Тетяна ФІЛІМОНЧУК
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Первєєву Володимиру Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Модель Android-застосунку з використанням базових сервісів

затверджена наказом по університету від " 21 " квітня 2025 р. № 296 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 16 червня 2025 р.

3. Вхідні дані до роботи Види архітектур, архітектурні модулі, фреймворк Flutter, платформа Android, середовище розробки – Android Studio

4. Перелік питань, що потрібно опрацювати у роботі
Порівняльний аналіз архітектурних моделей
Проєктування моделі мобільного застосунку
Перелік функціональних вимог до мобільного застосунку
Аналіз технологій, що використано для реалізації мобільного застосунку

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Огляд архітектурних моделей	22.04.25-29.04.25	
2	Вибір та обґрунтування методики дослідження	30.04.25-05.05.25	
3	Вибір інструментальних засобів	06.05.25-09.05.25	
4	Розробка архітектурної моделі	10.05.25-21.05.25	
5	Проведення експериментів	22.05.25-02.06.25	
6	Оформлення матеріалів кваліфікаційної роботи	03.06.25-05.06.25	
7	Подання кваліфікаційної роботи керівникові та	06.06.25-09.06.25	
	її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	10.06.25-12.06.25	
	вання		

Дата видачі завдання " 21 " квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тетяна ФІЛІМОНЧУК
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 70 с., 2 рисунка, 1 табл., 23 джерел.

МОБІЛЬНИЙ ЗАСТОСУНОК, FLUTTER, ANDROID, АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ІНТЕРФЕЙС КОРИСТУВАЧА, ІПЗ, БАЗА ДАНИХ, МОДУЛЬНІСТЬ, РОЗШИРЮВАНІСТЬ, КРОСПЛАТФОРМНІСТЬ, ПЛАГІН, ІНТЕГРАЦІЯ, ПРОДУКТИВНІСТЬ.

Метою кваліфікаційної роботи є модифікація архітектурної моделі мобільного застосунку на платформі Android із використанням фреймворку Flutter, що дозволяє забезпечити високу гнучкість, масштабованість, стабільність та продуктивність при розробці сучасних застосунків.

У ході виконання кваліфікаційної роботи було здійснено аналіз архітектурних моделей мобільних застосунків, розглянуто особливості використання Flutter у різних сценаріях та визначено ключові критерії для порівняння архітектур: гнучкість, стабільність, продуктивність, зручність обслуговування, модульність тощо. Було досліджено переваги й недоліки базової, розширеної, корпоративної, гнучкої та інноваційної моделей. За результатами аналізу проведено порівняння моделей за визначеними критеріями, а також візуалізовано отримані дані у вигляді графіків. На основі отриманих результатів запропоновано рекомендації щодо вибору архітектурної моделі залежно від потреб конкретного проєкту. Результати дослідження підтвердили, що використання Flutter у поєднанні з гнучкими архітектурами дозволяє ефективно реалізувати вимоги до сучасних мобільних застосунків.

ABSTRACT

Master’s thesis: 70 pages, 2 figures, 1 tables, 23 sources.

MOBILE APPLICATION, FLUTTER, ANDROID, SOFTWARE ARCHITECTURE, USER INTERFACE, IPI, DATABASE, MODULARITY, EXTENSIBILITY, CROSS-PLATFORMITY, PLUG-IN, INTEGRATION, PRODUCTIVITY.

The purpose of the qualification paper is to modify the architectural model of a mobile application on the Android platform using the Flutter framework, which allows for high flexibility, scalability, stability, and performance in the development of modern applications.

In the course of the qualification paper, an analysis of architectural models for mobile applications was carried out, the specifics of using Flutter in various scenarios were considered, and key criteria for comparing architectures were defined: flexibility, stability, performance, maintainability, modularity, etc. The advantages and disadvantages of basic, extended, enterprise, flexible, and innovative models were investigated. Based on the results of the analysis, a comparison of the models according to the defined criteria was performed, and the obtained data was visualized in the form of graphs. Based on the results obtained, recommendations for choosing an architectural model depending on the needs of a specific project were proposed. The research results confirmed that using Flutter in combination with flexible architectures allows for the effective implementation of requirements for modern mobile applications.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ 8
ВСТУП 9
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 10
2 АРХІТЕКТУРА, ЯК ОСНОВНИЙ КОМПОНЕНТ МОДЕЛІ ЗАСТОСУНКУ 15
2.1 Сутність, роль та цілі архітектури програмного забезпечення у розробці мобільних застосунків 15
2.2 Аналіз існуючих архітектурних моделей 18
2.2.1 Базова модель для простих застосунків 21
2.2.2 Розширена модель із підтримкою масштабування 23
2.2.3 Корпоративна модель з підвищеними вимогами безпеки 24
2.2.4 Модель для застосунків із великим мультимедіа-контентом 25
2.2.5 Високонавантажена модель з великою кількістю запитів 26
2.2.6 Гнучка модель для проєктів з частими оновленнями 27
2.2.7 Інноваційна модель з підтримкою IoT або Smart-сервісів 28
3 МОДЕЛЬ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ОС ANDROID 30
3.1 Аналіз досліджень та публікацій 30
3.2 Модифікована модель мобільного застосунку для ОС Android 33
3.3 Обґрунтування складових моделі мобільного застосунку 34
4 ПОРІВНЯННЯ ЗАПРОПОНОВАНОЇ МОДЕЛІ З ІСНУЮЧИМИ АРХІТЕКТУРНИМИ РІШЕННЯМИ 44
4.1 Критерії порівняння архітектурних моделей 44
4.2 Порівняння існуючих моделей с запропонованою 48
4.2.1 Порівняння моделей за критеріями продуктивності та швидкодії 49
4.2.2 Порівняння моделей за критерієм стабільності та надійності 53
4.3 Визначення конкурентних переваг запропонованої моделі 55
ВИСНОВКИ 60

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 61
ДОДАТОК А Графічний матеріал кваліфікаційної роботи..... 64

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АПЗ – архітектура програмного забезпечення

ІАЦ – інформаційно-аналітичний центр

ІК – інтерфейс користувача

ІПЗ – інтерфейс програмування застосунку

ІТ – інструменти тестування

КІ – компоненти інтерфейсу: кнопки, списки, анімації, які забезпечують інтуїтивність та зручність використання

МБД – модуль безпеки даних

МЛЗ – модуль логіки застосунку

МОЗД – модуль, що відповідає за організацію зберігання даних

МП – мова програмування

МРФМЗ – розширення функціональних можливостей застосунку

МТ – модуль тестування

ОМ – XML-файли для опису макетів

СР – середовище розробки

ВСТУП

Розробка мобільних застосунків останніми роками стала однією з найдинамічніших та найперспективніших сфер у галузі інформаційних технологій. Зі зростанням популярності мобільних платформ з'являються все нові вимоги до швидкості розробки, функціональності, безпеки та масштабованості застосунків [1]. У зв'язку з цим розробники шукають архітектурні рішення, які не тільки забезпечать стабільність та продуктивність, але й дозволять легко адаптуватися до нових викликів ринку.

Одним із ефективних інструментів для створення кросплатформних мобільних застосунків є фреймворк Flutter. Його популярність пояснюється можливістю одночасної розробки для Android та iOS із використанням єдиного коду, що значно пришвидшує процес та знижує витрати. Водночас архітектура мобільного застосунку, побудована на Flutter, повинна враховувати інтеграцію платформозалежних сервісів, таких як геолокація, робота з камерою, надсилання сповіщень та інші функції, без яких важко уявити сучасний мобільний продукт.

Сучасний мобільний застосунок складається з багатьох компонентів, кожен з яких виконує окрему роль. До таких компонентів належать інтерфейс користувача, бізнес-логіка, організація зберігання даних, інтерфейс програмування застосунків, модуль, який організує безпеку використання даних та розширення функціональних можливостей. Розробка архітектури повинна забезпечувати гнучку взаємодію цих складових та можливість розширення за допомогою додаткових модулів, що дозволяє застосункам швидко адаптуватися до нових функціональних потреб.

У процесі створення мобільного застосунку важливу роль відіграє правильний вибір методології, що впливає на якість, надійність та терміни реалізації проєкту. Під час розробки використовується структуроване планування та поетапне тестування, що дозволяє контролювати весь життєвий цикл застосунку від ідеї до виходу на ринок.

Метою кваліфікаційної роботи є дослідження та модифікація моделі архітектури мобільного Android-застосунку у середовищі Flutter, яка враховує інтеграцію базових сервісів та можливість подальшого розширення. Об’єктом дослідження є взаємодія компонентів моделі, а предметом – підходи до побудови архітектури, яка забезпечує продуктивність, стабільність та масштабованість застосунку.

Актуальність теми визначається зростаючою потребою у створенні гнучких та безпечних мобільних застосунків, які можуть відповідати високим стандартам ринку та задовольняти потреби користувачів. Вивчення архітектурних підходів до побудови моделі мобільного застосунку дозволить розробникам формувати універсальні рішення, що поєднують ефективність, зручність та технологічну готовність до змін.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Сучасна розробка мобільних застосунків є однією з найбільш динамічних та конкурентних сфер у галузі програмної інженерії. Щорічно кількість мобільних застосунків зростає, а користувачі очікують від продуктів не лише функціональності, а й стабільної роботи, високої продуктивності та зручності. Мобільні застосунки стали невід'ємною частиною повсякденного життя: вони використовуються для спілкування, роботи, навчання, розваг та керування побутовими процесами. У зв'язку з цим підвищуються вимоги до якості застосунків, їхньої безпеки, гнучкості та швидкості виходу на ринок.

Актуальні тенденції демонструють, що користувачі надають перевагу продуктам з інтуїтивно зрозумілим інтерфейсом, швидкою роботою, підтримкою персоналізованих сервісів та стабільністю навіть під високим навантаженням. Також важливою стала підтримка хмарних функцій, синхронізації між пристроями та можливість легкої інтеграції з іншими сервісами. Крім того, користувачі очікують постійного оновлення функціоналу без втрат у продуктивності та якості роботи застосунку.

Одним із важливих напрямів розвитку стала кросплатформна розробка. Порівняно з нативними рішеннями, де створення застосунків для кожної операційної системи потребує окремої роботи, кросплатформні фреймворки дозволяють створювати єдиний код для кількох платформ одночасно. Найпопулярнішими рішеннями є Flutter, React Native та Xamarin. Вибір Flutter у сучасних умовах обумовлений його високою продуктивністю, швидкою компіляцією, гнучкістю у створенні кастомних інтерфейсів та широкою підтримкою спільноти. Саме завдяки цим особливостям Flutter стає основою для розробки архітектурних моделей мобільних застосунків, орієнтованих на швидке масштабування та подальше розширення функціональності [2].

Архітектура мобільного застосунку визначає його можливості щодо

масштабованості, стабільності та адаптивності до нових викликів. Однією з головних вимог є продуктивність. Застосунок повинен працювати швидко та без збоїв навіть під високим навантаженням. Це стосується часу відгуку інтерфейсу, обробки запитів та взаємодії з сервером. Продуктивність безпосередньо залежить від правильного розподілу навантаження між клієнтською та серверною частинами, оптимізації запитів до БД та ефективної реалізації бізнес-логіки.

Другою важливою вимогою є масштабованість архітектури. Застосунок повинен легко адаптуватися до збільшення кількості користувачів та обсягу даних без необхідності глобальних змін у коді чи структурі. Масштабованість забезпечується використанням хмарних сервісів, гнучких ІПЗ та правильним проєктуванням баз даних.

Безпека займає окреме місце серед вимог до архітектури мобільних застосунків. Сучасний застосунок повинен гарантувати захист даних користувача від несанкціонованого доступу, забезпечувати шифрування при переданні інформації та застосовувати багаторівневу аутентифікацію та авторизацію. Безпека також передбачає постійний моніторинг, журналювання подій та своєчасне оновлення компонентів для уникнення вразливостей. Окрім продуктивності та безпеки, архітектура повинна бути гнучкою та готовою до інтеграції нових сервісів. Це надає можливість додавання функціоналу без зміни основної логіки або порушення стабільності роботи. Платформа має передбачати підтримку плагінів, розширень та сторонніх модулів.

Окремо слід виділити вимоги до зручності інтерфейсу та якості взаємодії з користувачем. Інтерфейс повинен бути інтуїтивним, доступним та адаптивним до різних типів пристроїв. Якісно спроєктований ІК дозволяє підвищити лояльність користувачів та робить застосунок конкурентоспроможним на ринку. Важливо, щоб архітектура підтримувала швидке оновлення інтерфейсних рішень та гнучку роботу з віджетами без негативного впливу на продуктивність.

Сучасні мобільні застосунки будуються на основі чітко визначених архітектурних моделей, які дозволяють структурувати код, забезпечувати стабільність та можливість подальшого масштабування. Базова модель архітектури мобільного застосунку зазвичай включає три основні компоненти: інтерфейс користувача, бізнес-логіку та базу даних (БД). Ці складові взаємодіють між собою через ІПЗ, що дає змогу налагодити комунікацію між клієнтською та серверною частинами. Однак базова модель часто виявляється недостатньою для складних проєктів, адже вона не передбачає розширень, гнучкості інтеграції нових сервісів та масштабування без кардинальних змін архітектури.

Метою цього дослідження є модифікація архітектурної моделі мобільного застосунку на платформі Android із використанням фреймворку Flutter [3], яка дозволяє інтегрувати базові сервіси, легко масштабуватися та розширювати функціонал відповідно до потреб користувачів та ринкових змін. Для досягнення поставленої мети необхідно вирішити такі задачі:

- розробити модель архітектури мобільного застосунку, яка повинна враховувати сучасні тенденції та вимоги до мобільних застосунків, забезпечуючи зручність у розробці, підтримці та подальшій модифікації;

- передбачити інтеграцію базових сервісів, без яких неможливо уявити сучасний мобільний застосунок. До таких сервісів належать геолокація для визначення місця користувача, надсилання push-сповіщень для оперативної комунікації та взаємодії, а також робота з камерою для реалізації функцій сканування або зйомки зображень;

- провести детальний аналіз можливостей масштабування запропонованої моделі та визначити способи її адаптації до зростаючих навантажень, збільшення кількості користувачів та розширення функціональності. Це передбачає дослідження застосування хмарних технологій, розподілених баз даних та гнучких підходів до інтеграції нових модулів;

- оцінити вплив обраних архітектурних рішень на продуктивність

мобільного застосунку та його гнучкість. Потрібно дослідити, як різні компоненти моделі взаємодіють між собою, які з них впливають на швидкодію, стабільність та можливість оновлень без порушення роботи системи. Ця оцінка дозволить сформувати практичні рекомендації для розробників щодо оптимізації архітектури та підвищення конкурентоспроможності готового продукту.

2 АРХІТЕКТУРА, ЯК ОСНОВНИЙ КОМПОНЕНТ МОДЕЛІ ЗАСТОСУНКУ

2.1 Сутність, роль та цілі архітектури програмного забезпечення у розробці мобільних застосунків

Архітектура програмного забезпечення (АПЗ) – це концепція, яка визначає взаємозв'язки та правила функціонування кожного компонента в межах системи. Вона формує логічну та фізичну структуру застосунку, описує розподіл обов'язків між компонентами, їхню ієрархію та способи комунікації. Архітектура також охоплює принципи масштабування, розширення функціоналу, адаптивності та відмовостійкості.

Основою архітектури є чітке визначення модулів та підсистем, що дозволяє створювати гнучкі рішення, готові до розвитку та модернізації. На етапі архітектурного проєктування приймаються стратегічні рішення: який підхід буде використано (наприклад, монолітна, мікросервісна чи модульна архітектура), які технології та фреймворки будуть застосовані, які стандарти кодування, тестування та безпеки будуть впроваджено.

Для мобільних застосунків архітектура має визначальне значення, адже від її продуманості залежить не лише якість продукту, а й швидкість його виводу на ринок, простота обслуговування та можливість масштабування без зупинок та великих затрат. Архітектурні рішення впливають на зручність оновлення застосунку, роботу з API, взаємодію з хмарними сервісами та інтеграцію нових модулів без шкоди основному функціоналу. Таким чином, АПЗ виступає фундаментом, який об'єднує в собі інженерні, організаційні та бізнес-аспекти розробки. Вона допомагає уникати хаотичності, зменшує ризики помилок на пізніх етапах, дозволяє чітко планувати життєвий цикл продукту та забезпечує цілісність усієї системи в умовах постійних змін.

Основна сутність архітектури полягає у формуванні такої моделі системи, яка не лише відображає внутрішню структуру застосунку, а й задає правила та механізми взаємодії всіх компонентів між собою. Архітектура

створює основу для чіткого розподілу ролей та функціональності між окремими елементами, а також для управління потоками даних.

У контексті мобільних застосунків архітектура передбачає розробку чіткої схеми взаємодії між інтерфейсом користувача, який відповідає за візуальне представлення та сприйняття застосунку, бізнес-логікою, що обробляє запити користувачів та керує основними процесами, а також підсистемами зберігання даних, які відповідають за організацію, збереження та доступ до інформації. Окреме місце займають зовнішні сервіси та API, що забезпечують зв’язок із сервером та іншими сторонніми платформами.

Архітектура має на меті забезпечити таку взаємодію, яка буде стабільною, безпечною та масштабованою. Вона задає механізми обробки помилок, логування подій, контролю доступу та підтримки цілісності даних. Крім того, завдяки продуманій архітектурі можливо безболісно впроваджувати нові функції, додавати модулі, оновлювати окремі компоненти без повного переписування коду.

Гнучкість архітектури дає змогу адаптувати застосунок до змінних ринкових вимог, легко масштабувати його на велику кількість користувачів, а також впроваджувати новітні технології без суттєвих ускладнень. Вона стає не просто технічною схемою, а стратегічною моделлю, яка визначає довгострокову життєздатність та конкурентоспроможність продукту.

Цілі АПЗ у розробці мобільних застосунків спрямовані на створення стійкої основи, яка дозволяє розробникам не лише швидко запускати продукт, але й підтримувати його актуальність та стабільність протягом усього життєвого циклу. Однією з ключових цілей є забезпечення можливості довготривалої підтримки та масштабування без потреби у глобальних змінах коду. Це досягається завдяки чіткому розмежуванню компонентів, їх модульності та можливості повторного використання.

Архітектура також покликана забезпечити стабільність роботи застосунку в умовах зростаючого навантаження, зміни зовнішніх сервісів або оновлення функціоналу. Для мобільних застосунків важливою метою є

адаптивність, яка дозволяє швидко впроваджувати нові функції, не порушуючи основної логіки роботи. Продумана архітектура дає можливість гнучко інтегрувати додаткові сервіси, такі як аналітика, реклама, системи сповіщень або зовнішні API, без складних переробок системи.

Ще одна важлива ціль – полегшення командної роботи. Завдяки чіткій архітектурі задачі розподіляються логічно між розробниками: окремі спеціалісти можуть займатися інтерфейсом користувача, інші – бізнес-логікою або інтеграцією з хмарними сервісами. Це не лише підвищує продуктивність роботи команди, а й дозволяє ефективно контролювати якість розробки та тестування кожної частини системи. Також архітектура відіграє роль орієнтиру для вибору технологічного стеку та визначення стандартів безпеки. Вона визначає, яким чином буде здійснюватися шифрування даних, які протоколи передавання інформації використовуватимуться та як будуть побудовані механізми автентифікації та авторизації. Це забезпечує цілісність та захищеність даних користувачів, що особливо актуально для мобільних застосунків із платіжними або персональними функціями. У результаті, архітектура стає ключовим елементом, який впливає на якість, довговічність та успіх застосунку на ринку.

При розробці мобільних застосунків архітектура виконує ключову роль у досягненні таких важливих характеристик, як висока продуктивність, швидкість відгуку системи та безперебійність функціонування [4]. Вона визначає те, як оптимально будуть організовані обчислювальні процеси, наскільки ефективно відбуватиметься передача даних між клієнтом та сервером та як мінімізуються затримки під час обробки запитів. Від архітектурних рішень залежить можливість масштабування застосунку без втрати продуктивності при зростанні кількості користувачів.

Особлива увага приділяється захисту даних користувача. Архітектура визначає, які механізми шифрування, автентифікації та контролю доступу будуть використані для збереження конфіденційності та цілісності даних. Це дозволяє не лише відповідати вимогам безпеки, але й підвищувати довіру

користувачів до продукту.

Важливою частиною архітектури є ефективне управління ресурсами мобільного пристрою – оперативною пам’яттю, процесором та акумулятором. Завдяки правильно побудованій структурі та оптимізації обчислень можна зменшити навантаження на систему та продовжити час роботи пристрою без підзарядки. Це особливо критично для застосунків, що працюють у фоновому режимі або використовують велику кількість мультимедійних ресурсів. Таким чином, архітектура перетворюється з технічного рішення на стратегічний інструмент, який дозволяє не тільки створити якісний продукт, а й забезпечити його конкурентоспроможність на ринку. Вона допомагає передбачити потенційні ризики, забезпечити гнучкість та швидкість впровадження змін, а також підтримувати актуальність застосунку у довгостроковій перспективі, що безпосередньо впливає на комерційний успіх проєкту.

2.2 Аналіз існуючих архітектурних моделей

Архітектура застосунку визначає його структуру, організацію компонентів, логіку взаємодії та інтеграцію зовнішніх сервісів. Від складових архітектури залежить продуктивність, масштабованість, гнучкість та безпека застосунку, що у підсумку впливає на якість досвіду користувача та конкурентоспроможність продукту. У таблиці 2.1 представлені різні комбінації архітектурних компонентів, які відповідають певним потребам та сферам застосування мобільних застосунків. Ці моделі демонструють можливі підходи до побудови архітектури залежно від складності проєкту, кількості користувачів, навантаження, вимог до безпеки та потреб у масштабуванні. Кожна модель включає набір елементів, таких як інтерфейс користувача, логіка застосунку, організація зберігання даних, інтерфейс програмування застосунків, модуль безпеки, розширення функціональних можливостей, хмарні сервіси, а також інструменти аналітики та тестування.

Окремі моделі акцентують увагу на певних характеристиках: безпеці, масштабованості, мультимедійних можливостях або інтеграції з IoT.

Базова модель призначена для простих застосунків, які потребують мінімального набору функцій для швидкого запуску (MVP). Вона включає тільки ключові компоненти, а саме інтерфейс користувача (ІК), логіку застосунку (ЛЗ), організацію зберігання даних (ОЗД), прості таблиці (Т), бази даних (БД), клієнтський застосунок (КЗ) та базовий інтерфейс програмування застосунків (ІПЗ) з мінімальною кількістю протоколів та методів запитів.

Розширена модель розроблена для застосунків середньої складності, таких як маркетплейси або освітні платформи. Окрім основних компонентів, вона містить деталізовану дизайн-концепцію (ДК), XML-файли для створення гнучких макетів, комплексний набір фреймворків та мов програмування (Ф+МП), широкий набір компонентів інтерфейсу (КІ), розвинуту серверну частину (СЗ), сервер бази даних (СБД), повноцінний інтерфейс програмування застосунків з різними форматами даних та проміжним програмним забезпеченням (ППЗ), а також інтеграцію з хмарними сховищами (ХС).

Корпоративна модель передбачає використання у критично важливих рішеннях із підвищеними вимогами до безпеки, таких як банківські чи медичні застосунки. Окрім базових компонентів, акцент зроблений на модуль безпеки (МБ), який включає шифрування (Ш), безпечну передачу даних (БПД), захист від шкідливих атак (ЗША), аутентифікацію та авторизацію (АА), захист ІПЗ та інструменти тестування безпеки (ІТБ). Це гарантує захист даних та відповідність регуляторним вимогам.

Модель для застосунків із великим мультимедійним контентом відрізняється гнучкістю завдяки інтеграції розширень функціональних можливостей (РФР). Вона забезпечує ефективну роботу з фото- та відеоконтентом, а також підтримує аналітику взаємодії користувачів із застосунком (ІА). Дизайн-концепція (ДК) та спеціалізовані компоненти інтерфейсу (КІ) забезпечують високу якість взаємодії з медіафайлами.

Таблиця 2.1 – Можливі архітектурні комбінації мобільних застосунків

№	Тип моделі	Основні компоненти	Сфера застосування
1	Базова модель для простих застосунків	ІК, ЛЗ, ОЗД (Т, БД), ІПЗ (ППД, МЗ), КЗ	Застосунки з мінімальним функціоналом, MVP-версії
2	Розширена модель із підтримкою масштабування	ІК, ДК, XML-файли, Ф+МП, КІ, ЛЗ, КЗ, СЗ, СБД, ОЗД, ІПЗ (ППД, ФД, МЗ, ППЗ), ХС	Застосунки середньої складності: маркетплейси, навчальні платформи
3	Корпоративна модель з підвищеними вимогами безпеки	ІК, ЛЗ, КЗ, СЗ, СБД, ОЗД, ІПЗ (ППД, МЗ, ППЗ), МБ (Ш, БПД, ЗША, АА, ЗІПЗ, ВД), ІТБ	Банківські застосунки, медичні рішення, логістика
4	Модель для застосунків із великим мультимедіа-контентом	ІК, ДК, КІ, ЛЗ, ОЗД (Т, БД, ХС), ІПЗ (ППД, ФД, МЗ), КЗ, СЗ, РФР, ІА	Фото- та відеохостинги, обмін файлами, соціальні мережі
5	Модель для високонавантажених застосунків з великою кількістю запитів	ІК, ЛЗ, КЗ, СЗ, СБД, ОЗД, ІПЗ (ППД, МЗ), МБ (Ш, БПД, АА, ЗІПЗ), РФР, ІТП, ІТБ, ІАТ	E-commerce, онлайн-сервіси з мільйонами користувачів
6	Гнучка модель для проєктів з частими оновленнями	ІК, ДК, XML-файли, Ф+МП, КІ, ЛЗ, КЗ, ОЗД, СЗ, СБД, ІПЗ, МБ, РФР, ТТ (ФТ, ТЮ, ТП), ІРТ	Новинні платформи, блоги, медіаплатформи
7	Інноваційна модель з підтримкою IoT або Smart-сервісів	ІК, ЛЗ, КЗ, СЗ, ОЗД, ІПЗ, МБ, РФР, ХС, ІА, ІТП, ІТБ, ТГ	Смарт-будинки, підключення розумних пристроїв, інтеграція IoT

Високонавантажена модель спрямована на застосунки з великою кількістю користувачів, такі як e-commerce платформи чи масштабні онлайн-сервіси. У ній особливу увагу приділено високопродуктивному інтерфейсу програмування (ІПЗ), хмарним сховищам (ХС) для масштабування, модулю безпеки (МБ), інструментам автоматизації тестів (ІАТ) та тестуванню продуктивності (ІТП), що дозволяє підтримувати стабільність роботи за умов високого навантаження.

Гнучка модель ідеальна для застосунків, що часто оновлюються, таких як новинні платформи чи блоги. Тут ключовими є гнучкі інструменти розробки (XML-файли, Ф+МП), інтегровані розширення функціональних можливостей (РФР), широкий спектр компонентів інтерфейсу (КІ), різні типи тестування (ТТ), зокрема функціональне тестування (ФТ), тестування юзабіліті (ТЮ), тестування продуктивності (ТП) та ручне тестування (ІРТ), що допомагає оперативно реагувати на зміни потреб користувачів.

Інноваційна модель призначена для інтеграції з IoT або Smart-сервісами. Вона забезпечує зв'язок мобільного застосунку з фізичними пристроями та датчиками, використовуючи інтерфейс програмування застосунків (ІПЗ), розширення функціональних можливостей (РФР), хмарні сервіси (ХС) та інструменти аналітики (ІА). Також важливим є тестування геолокації (ТГ) та продуктивності (ІТП), що дозволяє забезпечити стабільну взаємодію з IoT-пристроями у реальному часі.

2.2.1 Базова модель для простих застосунків

Базова модель архітектури мобільного застосунку використовується для розробки невеликих застосунків з мінімальним функціоналом або MVP-версій [5]. Набір компонентів, якими оперує модель формують мінімально необхідний набір для повноцінної роботи застосунку: інтерфейс дозволяє користувачу взаємодіяти із системою, логіка відповідає за обробку дій користувача та внутрішні процеси, база даних зберігає основну інформацію,

а API забезпечує зв’язок між клієнтською частиною та сервером або зовнішніми ресурсами. Додавання складніших модулів на ранніх етапах може призвести до перевантаженості архітектури, зайвих витрат часу та ресурсів на реалізацію функціоналу, який не є критично необхідним.

Також базова модель ідеально підходить для проєктів, де важливо швидко отримати робочу версію та протестувати основні гіпотези. Використання більшої кількості компонентів у такій ситуації недоцільне, оскільки це призводить до надмірної складності проєкту, ускладнює супровід та збільшує ймовірність помилок у початкових реалізаціях. Спрощена архітектура полегшує відлагодження, а також дозволяє швидко внести зміни на основі перших відгуків користувачів, що робить цю модель практичним вибором для стартапів, прототипів та внутрішніх корпоративних рішень, де ключовою є швидкість виходу на ринок.

Однак у порівнянні з більш складними архітектурними моделями базова модель має суттєві недоліки. Вона не забезпечує достатньої масштабованості при зростанні кількості користувачів або функціональних вимог, що призводить до необхідності повного рефакторингу на пізніших етапах. Обмежена гнучкість та відсутність окремих модулів безпеки, аналітики та інструментів моніторингу роблять таку архітектуру менш стійкою до атак, підвищують ризик виникнення непередбачених помилок та сповільнюють реакцію на технічні проблеми. Крім того, відсутність модулів розширень та хмарних сервісів зменшує можливість інтеграції нових функцій без суттєвих змін у кодовій базі та загальної перебудови архітектури.

З точки зору довготривалого розвитку, базова модель не дозволяє ефективно реалізувати додатковий функціонал, підтримувати складні бізнес-процеси або інтегрувати з зовнішніми системами без суттєвих інвестицій у час та ресурси. Саме тому базова модель не підходить для масштабних комерційних проєктів, продуктів із великою аудиторією чи систем, які планують активне розширення функціоналу та вихід на нові ринки.

2.2.2 Розширена модель із підтримкою масштабування

Розширена модель архітектури мобільного застосунку створена для проєктів, які потребують не лише базового функціоналу, але й готовності до зростання, розширення та збільшення навантаження [6]. Набір компонентів, якими оперує модель підібрані оптимально для підтримки масштабування, оскільки дозволяють будувати систему, здатну витримувати зростаючі вимоги без надмірної складності. Використання більшої кількості модулів на етапі масштабованої архітектури недоцільне, адже це може спричинити ускладнення архітектури, збільшити час розробки та знизити керованість системою. Надлишкові модулі можуть впливати на продуктивність, ускладнювати супровід та вимагати додаткових ресурсів на підтримку, що суперечить головній меті цієї моделі – забезпечення гнучкості та готовності до подальшого зростання із збереженням оптимальної складності.

Перевагами моделі є можливість безперервного розвитку та інтеграції нових функцій без суттєвих змін архітектури. Модульність дозволяє легко оновлювати окремі частини системи, адаптувати її під нові бізнес-завдання та мінімізувати ризики під час розширення. Хмарні сервіси забезпечують додаткові можливості для масштабування та зберігання великих обсягів даних. Модулі безпеки захищають дані користувачів, що особливо важливо для фінансових та соціальних застосунків. Використання цієї моделі дозволяє швидко додавати новий функціонал, реагуючи на потреби ринку.

Недоліком цієї моделі є те, що вона все ще може бути недостатньо глибокою для високонавантажених проєктів із глобальними масштабами. Відсутність у структурі таких складних механізмів, як автоматизовані системи тестування продуктивності або модулі розподілу навантаження, може обмежувати можливість обробки мільйонних запитів одночасно. Крім того, під час інтеграції великої кількості зовнішніх сервісів можуть виникати труднощі з керуванням залежностями та конфігурацією. У порівнянні з більш складними моделями розширена архітектура поступається у можливості

побудови модульних мікросервісних систем та глибокої кастомізації логіки для специфічних галузевих рішень. Вона є компромісом між простотою та функціональністю, підходить для середніх та великих проєктів, але для критично важливих корпоративних рішень може вимагати подальшого розвитку та поглибленої індивідуалізації.

2.2.3 Корпоративна модель з підвищеними вимогами безпеки

Корпоративна модель архітектури мобільного застосунку орієнтована на проєкти з високим рівнем складності, великою кількістю користувачів та підвищеними вимогами до безпеки та стабільності [7]. Набір компонентів, якими оперує модель обумовлен необхідністю створення максимально захищеного та надійного середовища для обробки чутливих даних та стабільного функціонування під значним навантаженням. Модулі ІТП та ІТБ дозволяють системно тестувати застосунок під великими навантаженнями, оцінювати стійкість та оперативно виявляти потенційні загрози безпеці. Присутність відновлення даних забезпечує безперервність роботи та мінімізує ризики втрат інформації при збої системи. Захист інтерфейсів програмування відповідає за обмеження доступу, логування та моніторинг усіх запитів, що унеможливлює несанкціоновані дії. Вибір саме цих модулів пояснюється потребою у балансі між високою складністю архітектури та її керованістю. Додавати ще більше компонентів, таких як мікросервісні шарування чи надмірно спеціалізовані обробники, було б зайвим, оскільки це ускладнило б обслуговування, збільшило час розробки та створило б точки відмови. У корпоративних системах надмірна складність підвищує ймовірність помилок та вимагає великих витрат на навчання та підтримку.

Серед переваг моделі є максимальна безпека, гнучкість та стійкість до збоїв. Використовуючи цю модель можливо обробляти великі обсяги даних, підтримувати інтеграцію з багатьма зовнішніми сервісами та забезпечувати збереження конфіденційності. Аналітичні інструменти надають можливість

приймати управлінські рішення на основі даних у реальному часі.

Недоліки корпоративної моделі пов’язані з її високою складністю та вартістю. Її розробка вимагає залучення великої команди кваліфікованих фахівців та значних фінансових витрат. Процес впровадження може займати багато часу, що не завжди виправдано для проєктів із меншими вимогами. Підтримка та оновлення такої архітектури потребують постійних зусиль, а перенавчання співробітників та складна документація можуть стати додатковим бар’єром для гнучкості. Порівняно з більш простими моделями вона менш мобільна та не підходить для проєктів із обмеженим бюджетом або короткими термінами реалізації.

2.2.4 Модель для застосунків із великим мультимедіа-контентом

Дана модель розроблена для мобільних застосунків, орієнтованих на активне використання великої кількості графіки, відео, аудіо та інтерактивних елементів [8]. Набір компонентів, якими оперує модель дозволяє швидке завантаження великих обсягів даних, підтримки високої продуктивності та адаптивність до різних типів пристроїв та роздільних здатностей екранів. Додатково в архітектуру впроваджуються модулі безпеки (Ш, БПД, ЗША), які спрямовані на захист авторських прав, цілісність мультимедіа-контенту та конфіденційність даних користувачів. Наявність кешування значно знижує навантаження на сервери та мережу, підвищуючи загальну швидкодію та стабільність роботи.

Перевагами моделі є можливість забезпечити якісну роботу навіть із дуже великим обсягом контенту, гнучкість у масштабуванні, підтримка адаптивних інтерфейсів та мінімізація часу завантаження даних. Це робить її ідеальним рішенням для стримінгових сервісів, медіаплатформ, мобільних ігор з великим графічним наповненням та візуальних онлайн-галерей.

Недоліками такої моделі є підвищена складність конфігурації системи, високі вимоги до оптимізації контенту та необхідність постійного контролю

продуктивності й оновлення інструментів обробки мультимедіа. Порівняно з корпоративною моделлю вона менш захищена з точки зору комплексної безпеки і не підходить для критично важливих фінансових або державних застосунків, де пріоритетом є максимальний рівень захисту та надійності. Проте для комерційних рішень, що потребують швидкості, високої продуктивності та масштабованості, ця модель демонструє свої сильні сторони та здатна ефективно задовольняти потреби кінцевих користувачів.

2.2.5 Високонавантажена модель з великою кількістю запитів

Модель для високонавантажених застосунків з великою кількістю запитів призначена для застосунків, які обробляють значний обсяг одночасних запитів від великої кількості користувачів у режимі реального часу [9]. Особливістю цієї моделі є необхідність забезпечення стійкості до високих навантажень та відмовостійкості. Система здатна обробляти тисячі запитів одночасно без втрати продуктивності завдяки впровадженню балансувальників навантаження та горизонтально масштабованих серверів. Використання кешування та черг повідомлень дозволяє розвантажувати основні сервіси та забезпечувати безперервну обробку запитів.

Серед переваг моделі – можливість обробки великих потоків даних без зниження швидкодії, гнучке масштабування інфраструктури та підвищена надійність. Така архітектура чудово підходить для фінансових платформ, біржових застосунків, сервісів доставки та систем бронювання.

Недоліками є складність налаштування та обслуговування, висока вартість інфраструктури та постійна потреба в контролі за безпекою й оптимізацією ресурсів. Крім того, порівняно з мультимедійною моделлю, ця архітектура менш орієнтована на роботу з великим обсягом статичного контенту та складних графічних рішень, фокусуючись насамперед на швидкій обробці даних та надійності комунікацій.

Всі компоненти моделі оптимізовані для роботи з максимальною

кількістю запитів та великим навантаженням. Вона включає використання багаторівневого кешування, розподілених черг для асинхронної обробки задач та адаптивних балансувальників навантаження, які автоматично масштабують потужності. Також передбачено глибокий моніторинг стану системи з можливістю прогнозування навантажень та запобігання піковим відмовам. Модель добре підходить для критичних застосунків із високою частотою оновлень даних, однак вимагає серйозних фінансових вкладень у інфраструктуру та постійної підтримки з боку досвідчених інженерів.

2.2.6 Гнучка модель для проєктів з частими оновленнями

Гнучка модель для проєктів з частими оновленнями орієнтована на застосунки, які регулярно оновлюються, змінюють структуру відповідно до зворотного зв’язку користувачів та ринкових тенденцій [10]. Перевагою моделі є її здатність адаптуватися до нових вимог без необхідності глобальних змін у всій системі. Завдяки використанню контейнеризації та мікросервісної архітектури нові модулі та сервіси інтегруються швидко та безболісно, що дозволяє проєкту розвиватися поступово та уникати великих простоїв при оновленнях.

Серед недоліків можна виділити підвищену складність початкової конфігурації, потребу у чітко регламентованому контролі версій та складнішому тестуванні при великій кількості змін. Крім того, у порівнянні з високонавантаженими або корпоративними моделями, гнучка модель поступається в продуктивності під час пікових навантажень, тому її краще використовувати для проєктів із середньою інтенсивністю запитів та фокусом на регулярне розширення функціоналу.

Гнучка модель для проєктів з частими оновленнями базується на принципах модульності, мікросервісної архітектури та гнучкого розгортання. Її перевагами є можливість безперервного вдосконалення продукту, швидке впровадження нових функцій і мінімізація ризиків під час оновлень. Така

модель дозволяє легко масштабувати окремі компоненти без зміни всієї структури, а завдяки аналітиці й моніторингу швидко реагувати на зміни поведінки користувачів та коригувати функціонал. У порівнянні з базовою моделлю, вона набагато гнучкіша та краще підходить для динамічних проєктів. Порівняно з розширеною моделлю, вона надає більше можливостей для дрібних частих оновлень, хоча й не орієнтована на роботу з великими навантаженнями. У порівнянні з корпоративною або високонавантаженою моделлю, гнучка архітектура менш захищена та потребує ретельнішої організації процесів контролю версій та тестування. Її недоліком є те, що при неправильному управлінні мікросервісами може виникати плутанина в залежностях та проблеми з інтеграцією. Крім того, у порівнянні з моделлю для мультимедійних застосунків вона не оптимізована для роботи з великим обсягом контенту, фокусуючись більше на стабільності змін та легкості інтеграції нових модулів.

2.2.7 Інноваційна модель з підтримкою IoT або Smart-сервісів

Інноваційна модель з підтримкою IoT або Smart-сервісів призначена для мобільних застосунків, які взаємодіють із широким спектром підключених пристроїв, сенсорів та розумних систем [11]. Основною причиною вибору такої моделі є потреба у швидкому та стабільному обміні даними між мобільним застосунком та пристроями, що підключено, обробка великої кількості дрібних запитів та повідомлень і можливість масштабування з урахуванням росту кількості підключених девайсів. Хмарна архітектура із брокерами повідомлень дозволяє ефективно управляти потоками інформації.

Перевагами цієї моделі є висока гнучкість у роботі з розумними пристроями, можливість швидко інтегрувати нові типи девайсів та протоколів, відмінна масштабованість та адаптивність під завдання різних сфер: від розумних будинків до промислових рішень. Модель можна

застосовувати у розробці застосунків для моніторингу енергоспоживання, управління смарт-пристроями, контролю виробничих процесів.

До недоліків моделі належать складність налаштування архітектури, необхідність постійного моніторингу безпеки через велику кількість взаємодій, підвищені вимоги до продуктивності мережі та інфраструктури. У порівнянні з гнучкою моделлю, інноваційна модель вимагає більшої уваги до протоколів взаємодії та забезпечення стабільності підключень. Порівняно з моделлю для високонавантажених застосунків, вона більш орієнтована на роботу з великою кількістю невеликих запитів від різних пристроїв і менш придатна для одночасної обробки важких транзакцій або складних обчислень. Однак для застосунків, які мають вихід у сферу розумних систем та IoT, ця модель є однією з найбільш перспективних та універсальних.

Відмінністю інноваційної моделі з підтримкою IoT або Smart-сервісів від інших розглянутих моделей є її орієнтація на динамічну взаємодію з безліччю зовнішніх пристроїв та систем. На відміну від базової моделі, де фокус зосереджений на мінімальних функціональних зв’язках між компонентами, ця модель передбачає складну структуру обміну даними з зовнішнім світом. У порівнянні з розширеною моделлю, вона враховує додаткові комунікаційні протоколи та здатна обробляти більшу кількість паралельних з’єднань. Корпоративна модель, хоч і надає високий рівень безпеки, поступається інноваційній у питанні масштабованості під потреби великої кількості підключених пристроїв. Модель для мультимедійних застосунків сфокусована на обробці важких контентних даних, у той час як інноваційна модель віддає пріоритет швидкості та стабільності передачі дрібних запитів. Модель для високонавантажених застосунків сконцентрована на транзакціях та продуктивності при великих обсягах даних, а інноваційна – на стабільній роботі з великою кількістю одночасних підключень. Гнучка модель забезпечує зручність оновлень, але не пристосована до швидких змін у мережній інфраструктурі та підтримки спеціалізованих IoT-протоколів.

3 МОДЕЛЬ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ОС ANDROID

3.1 Аналіз досліджень та 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 [12] 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 стійкість та конкурентоспроможність застосунку в умовах ринку, що швидко змінюється.

У роботі [13] 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 [14] 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 [15] 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 [16] 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 [17] 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 [18,19]. У середовищі Flutter така модель [20] може виглядати як набір компонентів (3.1):

$$М=\{ІК, МЛЗ, МОЗД, ІПЗ\} \quad (3.1)$$

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 [21].

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 [22].

3.2 Модифікована модель мобільного застосунку для ОС Android

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 [23], яка базується на використанні фреймворку Flutter як основного середовища розробки, і складається з наступних складових (3.2):

$$М = \{ІК, МЛЗ, МОЗД, ІПЗ, МБД, МРФМЗ, МТ, ІТ\}, \quad (3.2)$$

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 (3.1) модель, що запропонована (3.2), значно розширена та включає нові складові, які забезпечують повноцінну архітектуру мобільного застосунку. Розглянемо більш докладно складові моделі мобільного застосунку для ОС Android.

3.3 Обґрунтування складових моделі мобільного застосунку

Інтерфейс користувача (ІК) є центральним елементом будь-якого мобільного застосунку, оскільки забезпечує взаємодію користувача із системою і будується на множині (3.3):

$$ІК = \{ДКІ, ОМ, КІ, СР, МП\}, \quad (3.3)$$

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.

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 (3.4), 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):

$$МЛЗ = \{КЧ, СЧ, СБД\}. \quad (3.4)$$

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 ІПЗ). 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 (3.5): 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:

$$МОЗД = \{Т, БД, СУБД, ХС\}. \quad (3.5)$$

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 БД та доступу до них: 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 (3.6), 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:

$$\text{ІПЗ} = \{\text{ППД}, \text{ФД}, \text{МЗ}, \text{ППЗ}\}. \quad (3.6)$$

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 та 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 (3.7) через ряд інструментів: шифрування (Ш), безпечну передачу даних (БПД), захист від шкідливих атак (ЗША), аутентифікацію та авторизацію (АА), а також механізми відновлення даних (МВД):

$$МБД = \{Ш, БПД, ЗША, АА, МВД, К\}. \quad (3.7)$$

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 (3.8): 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:

$$МРФМЗ = \{ДМА, СС, ІЕ\}. \quad (3.8)$$

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 (3.9) за рахунок використання різноманітних типів тестів: функціональне тестування (ФТ), тестування юзабіліті (ТЮ), тестування продуктивності (ТП), тестування безпеки (ТБ):

$$МТ = \{ФТ, ТЮ, ТП, ТБ\}. \quad (3.9)$$

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 (3.10):

$$ІТ = \{ІАТ, ІРТ, ІТП, ІТБ\}, \quad (3.10)$$

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 модулів (3.8, 3.9) забезпечується стабільність, продуктивність та безпека мобільного застосунку, що є критично важливим для успішного функціонування на ринку.

В якості висновку можливо зазначити, що запропонована архітектурна модель мобільного застосунку, яка реалізована за допомогою платформи Flutter, забезпечує високі показники гнучкості, продуктивності та масштабованості. Завдяки інтеграції базових Android-сервісів можливо створювати ефективні функціональні рішення, що відповідають актуальним потребам користувачів.

4 ПОРІВНЯННЯ ЗАПРОПОНОВАНОЇ МОДЕЛІ З ІСНУЮЧИМИ АРХІТЕКТУРНИМИ РІШЕННЯМИ

4.1 Критерії порівняння архітектурних моделей

Розглянемо ключові критерії, які дозволяють об’єктивно оцінити та порівняти архітектурні моделі мобільних застосунків. Такий порівняльний аналіз допоможе визначити сильні та слабкі сторони кожної моделі, а також з’ясувати, чому запропонована модель є найкращою.

Критерій №1: продуктивність та швидкодія. Це один із найважливіших аспектів під час оцінювання архітектури мобільного застосунку. Він включає швидкість відповіді на запити користувачів, здатність до обробки великих обсягів даних та стійкість до зниження продуктивності під час пікових навантажень. Продуктивність також залежить від обраної технології (наприклад, Flutter, нативна розробка тощо), використаних підходів до кешування даних та оптимізації мережних запитів. Цей критерій визначає, чи зможе застосунок працювати стабільно при великій кількості одночасних користувачів і чи не виникатимуть затримки або збої, що можуть вплинути на якість досвіду користувача.

Критерій №2: гнучкість та можливість модифікації. Цей критерій відображає, наскільки архітектура дозволяє легко вносити зміни або доповнення у функціонал мобільного застосунку без ризику порушення стабільності або функціонування інших його частин. Гнучкість означає, що архітектурне рішення має чітку модульну структуру, де кожен компонент (наприклад, інтерфейс користувача, база даних, API) працює незалежно від інших і може бути оновлений або замінений у разі потреби. Це особливо важливо для проєктів, які швидко еволюціонують та потребують частих оновлень: як-от впровадження нових функцій, зміни бізнес-логіки чи виправлення помилок. Гнучка архітектура дає змогу швидко інтегрувати нові модулі або сторонні сервіси (аналітика, оплати, інтеграція зі смарт-

пристроями) без необхідності глобального переписування коду або зупинки роботи застосунку. Крім того, вона дозволяє масштабувати окремі частини системи, не зачіпаючи інші. Наприклад, додати підтримку нових платформ (iOS, Android) або підключити хмарні сервіси. У проєктах, де постійно змінюються потреби користувачів або вимоги ринку, саме гнучкість є ключовою перевагою архітектури. Вона дозволяє підтримувати конкурентоспроможність застосунку, швидко адаптуватися до нових умов та уникати дорогих та складних переробок системи у майбутньому.

Критерій №3: масштабованість. Цей критерій характеризує здатність архітектури мобільного застосунку без проблем адаптуватися до зростаючих навантажень та потреб бізнесу. Масштабованість означає, що зі збільшенням кількості користувачів, транзакцій чи запитів застосунок продовжує працювати стабільно, без значного зниження продуктивності. Щоб досягти цього, архітектура повинна передбачати можливість розподілення навантаження між кількома серверами або сервісами. Зокрема, це стосується використання балансувальників навантаження, які рівномірно розподіляють запити між різними вузлами системи, знижуючи ризики перевантаження одного з них. Також важливою є гнучкість у роботі з базами даних. Наприклад, підтримка реплікації або розділення навантаження між читанням та записом. Масштабована архітектура дозволяє безперешкодно додавати нові сервери чи ресурси для обробки запитів у разі зростання аудиторії або розширення функціоналу. Завдяки цьому розробники можуть не турбуватися про те, що застосунок «впаде» або стане повільним у критичні моменти, наприклад, під час акційних кампаній або великих оновлень. Масштабованість також важлива для підтримки глобального зростання: якщо застосунок орієнтований на міжнародну аудиторію, архітектура має дозволяти гнучко розгортати інфраструктуру в різних регіонах для зменшення затримок та підвищення якості користувацького досвіду.

Критерій №4: інтеграція базових та сторонніх сервісів. Цей критерій оцінює, наскільки зручно архітектура дозволяє підключати необхідні сервіси

та API, такі як геолокація, сповіщення, аналітика чи платіжні системи. Важливо, щоб модель мала чітко визначені точки взаємодії, які спрощують інтеграцію і не створюють «пляшкових горлечок» у продуктивності чи безпеці.

Критерій №5: безпека та захист даних. Цей критерій є одним із найважливіших у виборі архітектури мобільного застосунку, адже він безпосередньо впливає на довіру користувачів та репутацію продукту. Він передбачає наявність ефективних механізмів для запобігання несанкціонованому доступу, втраті чи модифікації даних. Безпечна архітектура застосунку повинна забезпечувати шифрування даних, щоб захистити їх під час зберігання та передачі; аутентифікацію користувачів для перевірки їхньої особистості; авторизацію, що дозволяє визначати права доступу до різних частин системи та функцій застосунку. Також важливо впроваджувати заходи для захисту від шкідливих атак: наприклад, налаштування вебфільтрів, перевірку вхідних даних для запобігання ін’єкціям та кібератакам, а також моніторинг підозрілих активностей та швидке реагування на загрози. Надійна архітектура передбачає наявність резервного копіювання та механізмів швидкого відновлення даних у разі збоїв або аварій. Це особливо критично для застосунків, які обробляють конфіденційні або фінансові дані: наприклад, у банківських сервісах, застосунках для електронної комерції чи медичних системах. Недостатня увага до безпеки може призвести до витоку даних, фінансових втрат та серйозної шкоди репутації компанії. Тому цей критерій завжди повинен бути у фокусі під час проєктування архітектури мобільного застосунку.

Критерій №6: стабільність та надійність. Цей критерій є визначальним для будь-якого мобільного застосунку, оскільки від його виконання залежить безперервна робота та зручність використання продукту. Під стабільністю розуміють здатність застосунку функціонувати без збоїв протягом тривалого часу навіть за умов підвищеного навантаження чи нештатних ситуацій. Надійність передбачає мінімізацію ризику раптових помилок, збоїв або

падінь застосунку, що може статися через несподівані обставини: збій мережі, проблеми з апаратними ресурсами чи помилки у логіці застосунку. Надійна архітектура повинна містити механізми для моніторингу працездатності ключових компонентів, вчасно виявляти аномалії й автоматично відновлювати роботу після збоїв. Також важливо мати продумані плани резервного копіювання, щоб у разі втрати або пошкодження даних їх можна було швидко відновити. У стабільному застосунку враховано як перевірку на помилки, так і обробку виняткових ситуацій (наприклад, неправильне введення даних або відсутність підключення до сервера), що дозволяє уникати критичних відмов та забезпечує користувачам безперервний та передбачуваний досвід. Порівняно з іншими критеріями, стабільність та надійність часто є базою для досягнення високої продуктивності та безпеки, оскільки проблеми у цих сферах можуть звести нанівець будь-які переваги швидкодії чи масштабованості. Таким чином, при виборі архітектури цей критерій завжди має бути у центрі уваги, особливо для застосунків, що працюють із критично важливою інформацією або обслуговують велику кількість користувачів.

Критерій №7: складність розробки та підтримки. Цей критерій показує, наскільки ресурсоємною є реалізація конкретної архітектурної моделі та її подальше обслуговування. Він включає не лише фінансові витрати, а й витрати часу та людських ресурсів. Чим складніша архітектура, тим більше етапів налагодження, тестування та інтеграції потрібно пройти, а це потребує участі кваліфікованих фахівців, зокрема архітекторів, розробників, тестувальників та фахівців із безпеки. Такі архітектури, як правило, вимагають ретельного документування, розробки інтерфейсів взаємодії між компонентами, а також регулярних оновлень та перевірок працездатності. Якщо модель містить складні механізми кешування, розподілену обробку даних або інтеграцію багатьох зовнішніх сервісів, це збільшує ризики появи помилок і потребує постійного контролю. Важливо враховувати, що надмірно складна архітектура, яка перевищує реальні потреби застосунку,

може призвести до значного подорожчання розробки та сповільнення часу виходу на ринок. У той час як більш проста модель може виявитися економнішою й ефективнішою, якщо вона відповідає потребам проєкту. Тому при виборі архітектури мобільного застосунку необхідно зважувати, чи виправдані витрати на складні рішення та наскільки вони відповідають очікуваним бізнес-результатам.

Критерій №8: відповідність вимогам цільового проєкту. Цей критерій розглядає, наскільки архітектурна модель мобільного застосунку здатна задовольнити специфічні потреби та цілі, визначені для конкретного проєкту. Він охоплює такі аспекти, як функціональні вимоги, продуктивність, безпека, інтеграція з іншими системами та готовність до масштабування. Важливо, щоб архітектура враховувала основний сценарій використання: для одних проєктів ключовою вимогою є підтримка великого мультимедіа-контенту (наприклад, стрімінгові платформи чи медіатеки), для інших – швидкі оновлення та гнучкість (як-от застосунки новин або маркетплейси), а для деяких – робота з IoT-пристроями або інтеграція аналітичних модулів (розумні будинки, системи моніторингу). Також цей критерій допомагає зрозуміти, наскільки обрана модель підтримує цілі бізнесу: чи відповідає вона темпам зростання компанії, чи дозволяє оперативно реагувати на запити користувачів, чи відповідає фінансовим та часовим обмеженням. Архітектура, що повністю відповідає цільовим задачам, дозволяє ефективніше використовувати ресурси команди, забезпечує стабільний розвиток застосунку та мінімізує ризики під час його експлуатації.

4.2 Порівняння існуючих моделей с запропонованою

Розглянемо сильні та слабкі сторони кожної з архітектурних моделей, що були проаналізовані раніше, визначимо ключові відмінності між ними та запропонованою комплексною моделлю, яка інтегрує важливі компоненти для забезпечення повноцінної роботи мобільного застосунку. Такий підхід

дозволяє створити універсальну архітектуру, яка відповідає сучасним вимогам ринку та потребам користувачів.

4.2.1 Порівняння моделей за критеріями продуктивності та швидкодії

На рисунку 4.1 наведено порівняльний аналіз продуктивності та швидкодії восьми архітектурних моделей мобільних застосунків. По горизонталі зазначено назви моделей, а по вертикалі – умовна шкала оцінювання (від 0 до 10 балів), де 10 відповідає найвищій продуктивності.

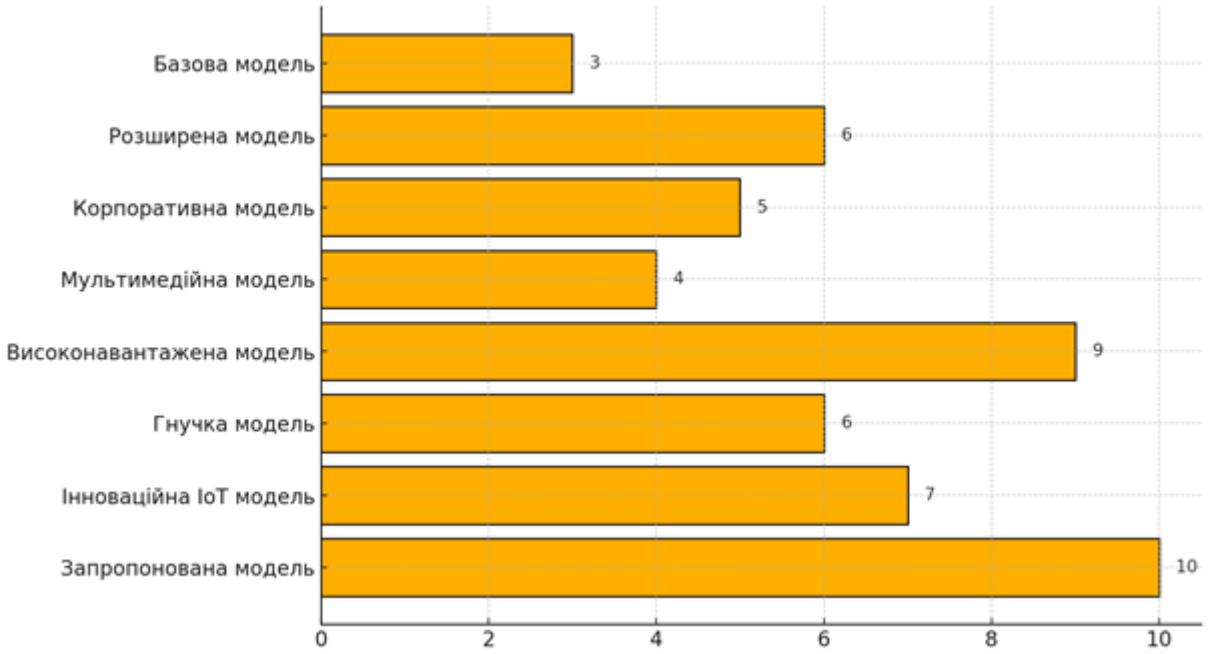

Рисунок 4.1 – Рівень продуктивності та швидкодії моделей

Базова модель отримала найнижчу оцінку продуктивності (3 бали), оскільки вона проєктувалася для найпростіших застосунків, де не передбачається значного навантаження. В архітектурі відсутні механізми обробки великої кількості одночасних запитів, кешування даних реалізовано на мінімальному рівні або взагалі не передбачено. Обробка мережних запитів здійснюється послідовно без оптимізації, що призводить до значних затримок при зростанні кількості користувачів. Також ця модель не враховує можливість горизонтального масштабування або розподілу ресурсів, що

критично для сучасних динамічних застосунків. Вона підходить лише для MVP-версій або внутрішніх інструментів з обмеженим колом користувачів, де швидкодія не є ключовим фактором.

Розширена модель має перевагу над базовою завдяки чіткішому розділенню компонентів та використанню більш структурованої логіки обробки даних. Вона вже передбачає окремі шари для бізнес-логіки та взаємодії з базою даних, що дозволяє краще керувати запитами, кешувати результати та масштабувати окремі частини системи при потребі. Однак, через відсутність повноцінного управління асинхронністю та навантаженням, її продуктивність може знижуватись при одночасній роботі великої кількості користувачів або при виконанні складних запитів. Для досягнення вищої ефективності ця модель потребує додаткових інструментів оптимізації – таких як балансування навантаження, покращене кешування, використання черг для обробки запитів, або перехід на більш складні архітектури.

Корпоративна модель з підвищеними вимогами безпеки демонструє високу стабільність у роботі завдяки впровадженим механізмам безпеки, таким як багаторівнева автентифікація, шифрування даних, контроль сесій та валідація запитів. Вона створена з урахуванням стандартів безпеки, що робить її незамінною для застосунків у фінансовій, медичній або урядовій сферах. Проте така посилена безпека не минає без наслідків для продуктивності. Додаткові етапи перевірок, шифрування/дешифрування даних, обробка логіки авторизації та моніторинг дій користувачів створюють навантаження на систему, що може призвести до зниження швидкодії, особливо під час пікових навантажень або великої кількості одночасних користувачів. Окрім цього, складність інтеграції нових функцій та необхідність суворого дотримання процедур безпеки уповільнюють реакцію системи на зміни. Таким чином, корпоративна модель ідеально підходить для сценаріїв, де пріоритетом є безпека, а не максимальна швидкість обробки запитів. Вона забезпечує надійність та захист, проте поступається іншим моделям у гнучкості й динамічній обробці великих потоків даних.

Модель для застосунків із великим мультимедійним контентом розроблена з урахуванням потреб у зберіганні, передачі та обробці великих обсягів відео, аудіо та зображень. Її основна сила у підтримці мультимедійних форматів, оптимізації буферизації, попереднього завантаження контенту та інтеграції з медіасховищами. Система побудована таким чином, щоб гарантувати плавне відтворення, швидкий доступ до великих файлів, можливість перегляду потокового відео без значних затримок і обривів. Вона включає засоби стиснення та адаптивної передачі контенту залежно від якості мережі, що особливо важливо для користувачів із різною пропускною здатністю інтернету. Однак така модель часто виявляється менш ефективною в загальній продуктивності застосунку, якщо йдеться про обробку великої кількості одночасних запитів, складну логіку користувацької взаємодії або бізнес-операції у реальному часі. Через фокус на медіа її структура може бути менш гнучкою у випадках, коли застосунок повинен швидко реагувати на динамічні зміни, масштабуватись під зростання запитів або забезпечувати миттєвий зворотний зв'язок. Таким чином, ця модель є чудовим рішенням для стрімінгових платформ, галерей, відео/аудіо архівів, однак менш придатна для інформаційних, фінансових або інтерактивних систем, де ключовими є швидкодія, навантаження на сервери й оптимізація складної логіки.

Модель для високонавантажених застосунків з великою кількістю запитів є однією з найефективніших з погляду продуктивності та стійкості до пікових навантажень. Вона спеціально розроблена для підтримки великої кількості одночасних з'єднань, запитів та потоків даних, що робить її ідеальною для застосунків на кшталт маркетплейсів, соцмереж, систем онлайн-замовлень або чатів у реальному часі.

У такій архітектурі передбачено кілька рівнів оптимізації: горизонтальне масштабування, що дозволяє збільшувати кількість серверів за потреби; використання балансувальників навантаження, які розподіляють запити рівномірно між інстансами; системи кешування, що зменшують

кількість звернень до бази даних; асинхронна обробка подій, що прискорює взаємодію між компонентами. Модель також орієнтована на зменшення часу відгуку та забезпечення стабільності при зростанні аудиторії. Вона передбачає моніторинг продуктивності, автоматичне масштабування залежно від завантаження та можливість інтеграції з хмарними сервісами для гнучкого управління ресурсами. Однак складність її реалізації вища за середню. Вона вимагає глибокого проєктування, тестування на стресостійкість та наявності розподіленої інфраструктури. Крім того, надмірна потужність може виявитися зайвою для менш ресурсомістких проєктів, що спричинить зайві витрати.

Гнучка модель для проєктів з частими оновленнями проєктується з урахуванням потреб у швидкій адаптації функціоналу та швидкому циклі релізів. Це ідеальний варіант для стартапів, MVP-проєктів або застосунків, які постійно доповнюються новими можливостями згідно з фідбеком користувачів. Архітектура такої моделі зазвичай має чітку модульну структуру, яка дає змогу змінювати окремі компоненти без впливу на інші. Вона підтримує CI/CD-процеси, гнучке тестування та інтеграцію нових API або сервісів у стислі строки. Проте саме через часті оновлення може виникати ризик зниження стабільності та погіршення продуктивності, якщо не впроваджено належної стратегії регресійного тестування та моніторингу. Крім того, архітектура зазвичай менш орієнтована на оптимізацію для пікових навантажень або великих обсягів трафіку, що обмежує її застосування у високонавантажених середовищах. Загалом, ця модель балансує між гнучкістю та продуктивністю, де акцент зроблено на швидкість реакції на зміни, а не на абсолютну обчислювальну ефективність.

Інноваційна модель з підтримкою IoT або Smart-сервісів вирізняється здатністю ефективно взаємодіяти з численними пристроями в режимі реального часу. Вона передбачає архітектуру, яка оптимізована для асинхронної обробки подій, миттєвого реагування на сенсорні сигнали, підтримки потоків даних від зовнішніх пристроїв та масштабування у

хмарних середовищах.

Її висока продуктивність забезпечується завдяки використанню сучасних протоколів обміну (наприклад, MQTT, WebSockets), а також можливістю делегувати частину обчислювального навантаження на периферійні пристрої (edge computing). Завдяки цьому застосунки на основі цієї моделі можуть забезпечувати мінімальну затримку при обробці подій та зберігати стабільну швидкодію навіть при великій кількості запитів. Однак значним викликом є складність налаштування системи. Щоб досягти такої продуктивності, потрібне продумане планування інфраструктури, включаючи хмарні сервіси, балансувальники навантаження, резервування каналів зв’язку та надійне управління конфігураціями пристроїв. Це робить модель дорогою у впровадженні та підтримці, особливо для команд з обмеженими ресурсами.

Запропонована модель демонструє високу продуктивність завдяки комплексному охопленню всіх ключових архітектурних аспектів, що впливають на швидкодію та стабільність мобільного застосунку. Вона поєднує ефективну бізнес-логіку, оптимізовану організацію зберігання даних з використанням кешування, продуману інтеграцію з API, а також включає модулі для автоматизованого тестування і моніторингу. Завдяки цьому забезпечується стабільна робота навіть у складних сценаріях з великою кількістю користувачів та динамічним контентом. Гнучка архітектура дозволяє масштабувати застосунок без ризику зниження продуктивності чи збоїв. Модель добре пристосована до подальшого розвитку й зростання проєкту, не потребуючи повної перебудови системи, що робить її універсальним і конкурентоспроможним рішенням у сучасних умовах.

4.2.2 Порівняння моделей за критерієм стабільності та надійності

Порівняльний аналіз моделей за критерієм стабільності та надійності (рисунок 4.2) є важливим етапом у розробці програмного забезпечення, зокрема мобільних застосунків, з кількох ключових причин. Такий аналіз

дозволяє визначити, яка модель забезпечує більшу стійкість до помилок, меншу кількість збоїв та кращу поведінку в умовах високого навантаження або нештатних ситуацій. Це критично для застосунків, які працюють із великою кількістю користувачів або чутливими даними.

Слід розуміти, що стабільна та надійна архітектура зменшує ризики пов'язані з майбутніми оновленнями, масштабуванням чи інтеграцією нових функцій. Аналіз допомагає обрати ту модель, яка не буде вимагати частих доопрацювань або виправлень. Надійна модель дозволяє гарантувати постійну доступність та стійкість до збоїв, що особливо важливо для застосунків у сферах фінансів, охорони здоров’я, логістики тощо.

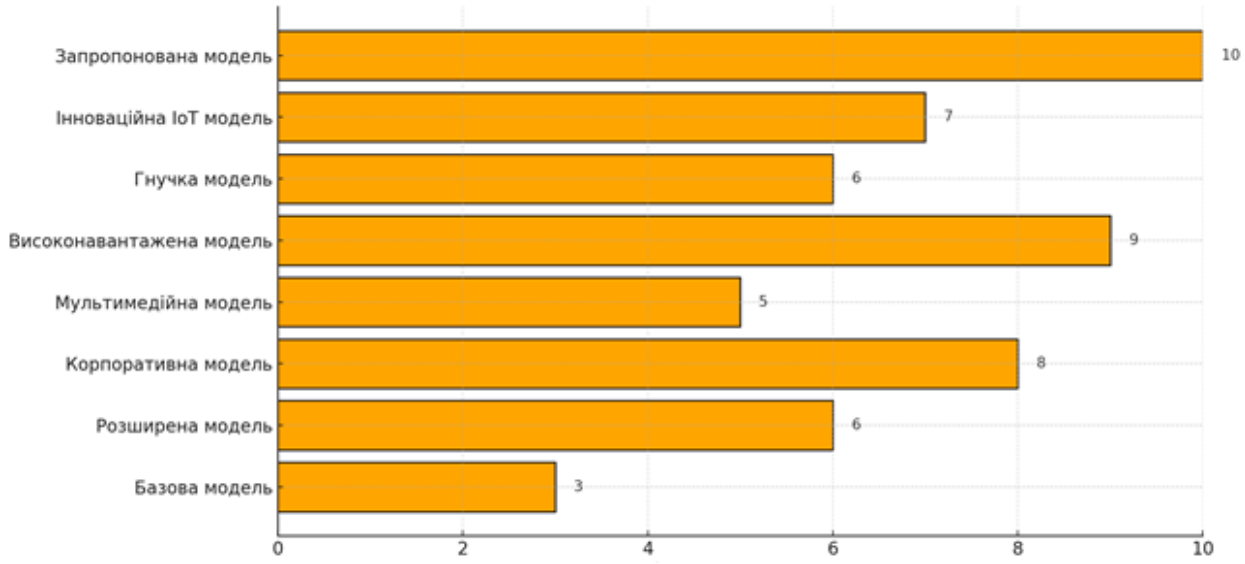

Рисунок 4.2 – Стабільність та надійність

За проведеним аналізом можливо робити наступні висновки:

- базова модель найменш надійна, тому що не має систем моніторингу, відновлення чи обробки збоїв;
- розширена модель має кращу структуру, але без глибоких механізмів захисту та самовідновлення;
- корпоративна модель має високий рівень надійності завдяки використання система доступу, контролю та шифрування;
- мультимедійна модель надійна при роботі з контентом, але

нестабільна при помилках з’єднання або оновленнях;

- високонавантажена модель може витримувати багато паралельних запитів, з резервними сценаріями;
- гнучка модель легка в адаптації, але залежна від цілісності модулів;
- інноваційна IoT модель вимагає стабільної інфраструктури, але добре справляється з розподіленими з’єднаннями;
- запропонована модель має найвищу оцінку завдяки комплексному підходу: резервування, тестування, відновлення, моніторинг.

Як висновок можливо зазначити, що порівняльний аналіз за критерієм стабільності та надійності є необхідним для прийняття технічно обґрунтованих рішень, що дозволяє створити якісний, довговічний та конкурентоспроможний мобільний застосунок.

4.3 Визначення конкурентних переваг запропонованої моделі

Запропонована модель демонструє низку ключових конкурентних переваг, які роблять її універсальним та стратегічно вигідним рішенням для створення мобільних застосунків у сучасних умовах.

Універсальність та повнота – це одна з головних переваг запропонованої моделі. Вона ґрунтується на об’єднанні всіх ключових архітектурних компонентів, які традиційно розглядаються як окремі або спеціалізовані блоки. Завдяки цьому підходу модель перетворюється на цілісну архітектурну систему, де кожен модуль відповідає конкретним завданням, але водночас взаємодіє з іншими складовими для досягнення спільної мети.

Інтерфейс користувача забезпечує інтуїтивно зрозумілу та привабливу взаємодію з застосунком, що особливо важливо для залучення та утримання користувачів. Бізнес-логіка визначає основний функціонал та правила роботи застосунку, а організація зберігання даних гарантує ефективну обробку та збереження інформації. Інтерфейс програмування застосунків відповідає за

безперебійну інтеграцію з зовнішніми сервісами, що розширює можливості застосунку.

Модулі безпеки формують захисний каркас застосунку, включаючи шифрування даних, аутентифікацію та авторизацію. Компонент МРФМЗ забезпечує обробку мультимедійного контенту, що стає дедалі актуальнішим у сучасних мобільних сервісах. Модулі тестування та інструменти тестування підвищують стабільність і надійність, дозволяючи виявляти й виправляти помилки ще до того, як вони вплинуть на кінцевого користувача. У сукупності це створює комплексну архітектуру, яка підходить для реалізації застосунків будь-якого рівня складності: від простих до високонавантажених чи мультимедійних. Універсальність означає, що розробник може легко адаптувати цю модель до конкретних завдань, не жертвуючи стабільністю або безпекою. А повнота – це впевненість у тому, що застосунок буде готовий до будь-яких викликів ринку та потреб користувачів.

Гнучкість та масштабованість – це друга ключова конкурентна перевага запропонованої моделі. Завдяки своїй модульній структурі, де кожен компонент чітко виконує власні завдання, архітектура легко піддається змінам без ризику дестабілізувати всю систему. Це означає, що розробники можуть вносити нові функціональні можливості або покращувати вже існуючі модулі окремо, не потребуючи глобального переписування застосунку чи довготривалих зупинок роботи.

Також модель дозволяє легко адаптуватися до швидких змін ринку чи побажань користувачів. Наприклад, якщо необхідно додати новий сервіс або функцію достатньо розробити та підключити відповідний модуль, не зачіпаючи решту архітектури. Це особливо важливо для стартапів і компаній, які працюють у конкурентному середовищі й постійно отримують нові запити від клієнтів.

Ще однією перевагою такої архітектури є її масштабованість. Коли кількість користувачів або запитів зростає, застосунок може розширюватися без зниження продуктивності. Завдяки чіткому поділу відповідальностей між

модулями та підтримці горизонтального масштабування, можна легко додавати нові сервери або обчислювальні ресурси, щоб витримати навантаження й уникнути збоїв.

Завдяки поєднанню гнучкості та масштабованості запропонована модель здатна ефективно підтримувати розвиток застосунку протягом його всього життєвого циклу, знижуючи ризики втрати конкурентоспроможності й забезпечуючи високу стабільність і швидкість роботи.

Високий рівень безпеки є одним із ключових аспектів запропонованої моделі, який робить її особливо цінною для проєктів, що працюють із критичними або конфіденційними даними. Модель включає окремі модулі, кожен із яких відповідає за різні рівні захисту: від базових перевірок доступу до складної обробки й шифрування даних.

Модуль аутентифікації дозволяє точно визначати користувачів застосунку, перевіряючи їхню особу через логіни, паролі або сучасні біометричні технології. Авторизація, у свою чергу, контролює, які саме дії або функції доступні конкретному користувачеві, зменшуючи ризики несанкціонованого доступу.

Шифрування даних забезпечує збереження конфіденційності всіх переданих і збережених даних, перешкоджаючи спробам перехоплення або несанкціонованого використання інформації. Додатково модулі захисту від шкідливих атак (ЗША) впроваджують спеціальні фільтри та механізми обробки аномальних запитів, знижуючи ризики DDoS-атак, SQL-ін’єкцій чи інших поширених загроз.

Завдяки цьому багаторівневому підходу запропонована модель відповідає найсуворішим стандартам безпеки, що особливо важливо для застосунків у фінансовому секторі, охороні здоров’я або державних проєктах. Вона гарантує цілісність даних, захист персональної інформації користувачів та зменшує ризики фінансових або репутаційних втрат для бізнесу.

Крім того, модель інтегрує підтримку мультимедійного контенту (МРФМЗ) та інструменти для роботи з IoT (як потенційне продовження

архітектури). Це дозволяє застосунку ефективно працювати з потоковим відео, аудіо, фото та іншими медіафайлами, що особливо важливо для сучасних сервісів, орієнтованих на залучення користувачів через багатий візуальний чи звуковий контент. Наявність окремих механізмів обробки мультимедіа дає змогу зменшити навантаження на сервери й підвищити швидкість відтворення, покращуючи загальний користувацький досвід.

Що стосується IoT, то підтримка «розумних» пристроїв (через спеціалізовані протоколи та інтерфейси) відкриває нові можливості для застосунків у сферах «розумного дому», автоматизації виробництва або моніторингу інфраструктури. Це робить запропоновану модель універсальною платформою, яка здатна інтегруватися з новітніми технологіями та підтримувати динамічний розвиток цифрових екосистем.

Таким чином, така комбінація мультимедійних та IoT-компонентів у межах єдиної архітектури дає змогу застосунку виходити за рамки стандартного функціоналу й бути гнучким інструментом для задоволення потреб найрізноманітніших сегментів ринку: від розваг до промислових рішень.

Не менш важливими є зручність обслуговування та тестування. Компоненти тестування (MT, IT) відіграють ключову роль у підтримці високої якості застосунку на всіх етапах його життєвого циклу. Вони дозволяють розробникам швидко знаходити й усувати помилки ще до того, як вони стануть критичними для користувачів, зменшуючи ризик виникнення серйозних проблем у продуктивній версії застосунку.

Завдяки наявності автоматизованих тестів (MT), можна перевіряти окремі модулі та їх взаємодію в комплексі, що значно знижує час, витрачений на ручне тестування, і мінімізує людський фактор. Інструменти тестування (IT), які можуть включати засоби навантажувального тестування, емулятори пристроїв або моніторингові системи, допомагають розробникам бачити повну картину роботи застосунку в реальних умовах.

Це забезпечує стабільність та прогнозованість роботи застосунку в

умовах різного навантаження, а також дозволяє швидко впроваджувати оновлення або виправлення без ризику порушити роботу всієї системи. У результаті користувачі отримують продукт, який залишається надійним та зручним навіть під час постійного розвитку та вдосконалення.

У підсумку запропонована модель успішно поєднує кращі практики всіх розглянутих архітектур, об’єднуючи їхні сильні сторони в єдине цілісне рішення. Завдяки цьому вона виступає універсальною платформою, яка може бути адаптована до будь-яких умов та потреб проєкту: від роботи з мультимедіа та IoT до обробки високонавантажених запитів чи забезпечення високого рівня безпеки.

Ще однією перевагою цієї моделі є її гнучкість, що дозволяє швидко додавати нові функції чи масштабувати застосунок під зростаючі вимоги бізнесу. Універсальність та модульність забезпечують легкість у впровадженні оновлень, а високий рівень безпеки – надійність у роботі з конфіденційною інформацією.

Такий підхід орієнтований на довгостроковий успіх застосунку, знижує ризики технічних проблем, спрощує подальше обслуговування та дозволяє проєкту залишатися конкурентоспроможним у стрімко змінному ринку. Завдяки цьому розробники можуть зосередитися на основних бізнес-цілях та впевнено будувати масштабовані, надійні та інноваційні рішення, що повністю відповідають сучасним потребам користувачів.

ВИСНОВКИ

Головною метою при розробці мобільного застосунку є визначення архітектурних компонентів моделі: для реалізації логіки, збереження даних, розробки інтерфейсу користувача та інтерфейсу програмування, який інтегрується із платформозалежними сервісами. В результаті проведеного аналізу розглянуто ключові аспекти архітектури мобільних застосунків, зокрема їх модульний підхід, який дозволяє адаптувати кінцевий продукт до вимог ринку. На основі отриманих результатів була модифікована модель Android-застосунку за рахунок розширення її складових, які орієнтовані на забезпечення високого рівня гнучкості, продуктивності та масштабованості.

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 в моделі модулю безпеки, який включає аутентифікацію, авторизацію та шифрування даних, гарантує конфіденційність та цілісність інформації, що є критичним для застосунків, які працюють із чутливими даними. Впровадження інструментів моніторингу та аналітики дозволяє підвищити продуктивність мобільного застосунку, виявляти можливі проблеми та своєчасно їх вирішувати. Додавання хмарних сервісів допоможе використати додаткові переваги у зберіганні даних, синхронізації між пристроями та оптимізації витрат на інфраструктуру. Інтеграція в модель існуючих базових сервісів, дозволяє створювати функціональні рішення, що адаптовані до сучасних потреб користувачів. Загалом, запропонована модель Android-застосунку демонструє ефективність у забезпеченні якісного кінцевого продукту, що відповідає вимогам користувачів та сучасним стандартам. Її гнучкість, адаптивність та надійність роблять її перспективною для широкого використання. Також слід зауважити, що в залежності від цілей та направленості мобільного застосунку, що розробляється, може бути обрано різні напрями його реалізації, інструментарію та подальшого тестування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мельник В.А., Сторожук В.М. Проєктування програмного забезпечення: навч. посіб. Вінниця: ВНТУ, 2018. 220 с.

2. Громовий Ю. В., Андрущенко Т. М. Основи розробки мобільних застосунків з використанням Flutter: навч.-метод. посіб. Київ: НТУУ "КПІ", 2020. 148 с.

3. Сич Н.М., Киричок О.Ю. Розробка мобільного застосунку з використанням Flutter // Вісник ХНУРЕ. Серія: Інформатика. 2021. № 1(71). С. 120-126.

4. Попов С.О., Іващенко О.О. Архітектура мобільних додатків: проблеми побудови та сучасні підходи // Системи обробки інформації. 2020. №2 (157). С. 47-51.

5. Панкратов К.Ю., Гринько А.О. Основи розробки мобільних застосунків: навч. посіб. Київ: КНЕУ, 2020. 172 с.

6. Коваль В.В., Литвин О.О. Мобільні додатки: проєктування, розробка та безпека: навч. посіб. Львів: ЛНУ ім. Івана Франка, 2021. 224 с.

7. Нікітенко А.С., Ігнатенко І.М. Архітектура корпоративних інформаційних систем: навчальний посібник. Київ: КНЕУ, 2019. 312 с.

8. Андрієнко О.В., Приймак С.І. Мультимедійні технології в мобільних застосунках: навчальний посібник. Київ: НТУУ "КПІ", 2020. 204 с.

9. Жуков В. В., Корнієнко О. А. Хмарні технології у високонавантажених системах: монографія. Харків: ХНУРЕ, 2019. 248 с.

10. Мосейчук С.І., Харченко В.С. Адаптивні програмні системи: методи та інструменти створення: монографія. Київ: КНУБА, 2020. 312 с.

11. Гнатюк С.М., Лінник Ю.В., Романишин І.Б. Інтернет речей: архітектура, моделі та протоколи: навчальний посібник. Київ: Видавництво Ліра-К, 2020. 352 с.

12. Бідник Д.С., Цибульник С.О. Архітектура мобільного додатку // Матеріали XIV Всеукраїнської науково-практичної конференції студентів,

аспірантів та молодих вчених "Погляд у майбутнє приладобудування". Київ: КПІ ім. Ігоря Сікорського, 2021. С. 19-22.

13. Сідельнікова Д.С. Етапи створення дизайну мобільного застосунку // Мультимедійні технології в освіті та інших сферах діяльності: науково-практична конференція з міжнародною участю. Київ: НАУ, 2022. С. 105-109.

14. Ставицький П.В., Войтко В.В. Організація архітектури додатків на базі мобільних платформ // Матеріали XLVI науково-технічної конференції підрозділів ВНТУ. Вінниця, 2017. Електрон. текст. дані. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2017/paper/view/2794.

15. Козуб Г.О., Козуб Ю.Г., Могильний Г.А., Жуков А.В. Розробка мобільного Android-додатку з застосуванням принципів Clean Architecture // Вісник Східноукраїнського національного університету ім. В. Даля. 2021. №5 (269). С. 5-10. DOI: https://doi.org/10.33216/1998-7927-2021-269-5-5-10.

16. Санак О.Є. Захист мобільних застосунків на основі систем з нульовим знанням: магістерська дисертація: 125 Кібербезпека. Київ, 2018. 121 с.

17. Дворецький М.Л., Нездолій Ю.О., Дворецька С.В., Кандиба І.О. Розробка мобільних застосунків для OS Android: навчальний посібник. Миколаїв: Вид-во ЧНУ ім. Петра Могили, 2021. 140 с.

18. Яшина К.В., Ялова К.М., Сугаль Є.О. Огляд гнучких методологій розробки програмного забезпечення // Збірник наукових праць Дніпровського державного технічного університету. 2017. Т.1, №30. С. 153-156.

19. Вавіленкова А.І. Аналіз гнучких методологій розробки програмного забезпечення для реалізації у командних проєктах // Вісник Національного технічного університету "ХПІ". 2021. №1 (7). С. 39-46.

20. Сеспедес Гарсія Н. В., Сеспедес Гарсія П. Д. Моделі життєвого циклу розробки програмного забезпечення // Молодий вчений. 2023. №2 (114). С. 17-20. DOI: https://doi.org/10.32839/2304-5809/2023-2-114-4.

21. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. 816 p.

22. Pressman R. S. Software Engineering: A Practitioner’s Approach. 9th ed. New York: McGraw-Hill Higher Education, 2019. 692 p.

23. Первєєв В.Д., Філімончук Т.В., Бугрій А.М., Партика С.О. Модель мобільного застосунку для ОС Android // Системи управління, навігації та зв’язку. Збірник наукових праць. Полтава: ПНТУ, 2025. Випуск 2. С. 170-175.