УДК 004.43
DOI: https://doi.org/ 10.17721/3041-2323.2025.107-121

Вікторія ГОЛОТЮК, студ.
ORCID ID: 0009-0008-3947-4613
e-mail: viktoriia_holotiuk@knu.ua
Київський національний університет
імені Тараса Шевченка, Київ, Україна

Олена ВАЩІЛІНА, канд. фіз.-мат. наук, доц.
ORCID ID: 0000-0001-6867-6216
e-mail: olenavashchilina@knu.ua
Київський національний університет
імені Тараса Шевченка, Київ, Україна

КРОСПЛАТФОРМНА РОЗРОБКА МОБІЛЬНИХ ЗАСТОСУНКІВ: ОСНОВНІ ПРИНЦИПИ, ТЕХНОЛОГІЇ ТА ПРАКТИЧНІ ПІДХОДИ

У статті розглянуто основні принципи та технології кросплатформної розробки мобільних застосунків. Висвітлено ключові аспекти цього підходу, а також проаналізовано його переваги й обмеження, що впливають на якість та ефективність створення програмних продуктів. Приділено увагу етапам життєвого циклу застосунку – від проєктування до супроводу, що дозволяє забезпечити стабільність і конкурентоспроможність рішень. Проведено огляд відомих фреймворків – React Native, Flutter і Xamarin – із порівняльною характеристикою їхніх можливостей. Наведено приклади практичного використання цих технологій у відомих мобільних продуктах (Instagram, Google Ads, Alaska Airlines), які демонструють ефективність і перспективність кросплатформного підходу. Описано власний досвід розроблення мобільного застосунку на Flutter, що підтверджує практичну доцільність обраного технологічного стеку в освітньому проєкті.

К л ю ч о в і с л о в а: ***Ккросплатформна розробка, мобільні застосунки, React Native, Flutter, Xamarin.***

Вступ

У сучасному цифровому середовищі мобільні застосунки стали головним інструментом взаємодії між користувачами та

сервісами, охоплюючи сфери комунікації, бізнесу, освіти та розваг. Зростаюча потреба в універсальних рішеннях, доступних на різних платформах, зумовила активний розвиток кросплатформної розробки – методології, що дозволяє створювати програмне забезпечення з єдиною кодовою базою для кількох операційних систем, зокрема iOS та Android.

Кросплатформні фреймворки, такі як React Native, Flutter і Xamarin, забезпечують інструменти для адаптації коду до специфіки кожної платформи, водночас зберігаючи спільну логіку та структуру застосунку. Вони дозволяють оптимізувати процес розробки, скоротити витрати, спростити підтримку та забезпечити ширше охоплення аудиторії. Проте, попри численні переваги, кросплатформні рішення мають і певні обмеження, пов'язані з продуктивністю, доступом до нативних функцій та адаптацією інтерфейсу.

У статті розглянуто основні концепції кросплатформної розробки, проведено порівняльний аналіз відомих фреймворків, а також проаналізовано основні етапи життєвого циклу мобільного застосунку – від проєктування до супроводу. Особливу увагу приділено практичним кейсам, що демонструють ефективність використання React Native, Flutter і Xamarin у реальних проєктах. Матеріал публікації спрямований на формування системного уявлення про сучасні підходи до створення кросплатформних мобільних рішень.

Принципи та технології кросплатформної розробки

Ключові концепції. Кросплатформна розробка мобільних застосунків – це методологія створення програмного забезпечення, здатного функціонувати на кількох операційних системах (зокрема iOS та Android) на основі єдиного програмного коду. Такий підхід дозволяє оптимізувати процес розробки, зменшити витрати часу та ресурсів, а також забезпечити ширше охоплення користувачів без потреби в окремій реалізації для кожної платформи.

Ця методологія базується на використанні спеціалізованих фреймворків і інструментів, які компілюють або транслюють спільний код у нативний формат, забезпечуючи сумісність із

різними платформами. В основі кросплатформної розробки є такі ключові концепції:

- Єдина кодова база – створення одного набору коду, який адаптується або компілюється для різних операційних систем, що дозволяє зберігати узгодженість логіки та функціональності застосунку.
- Адаптація через фреймворки – використання інструментів і бібліотек, які забезпечують інтеграцію з платформними особливостями, дозволяючи зберігати основний код спільним і водночас реалізовувати специфічні функції.
- Архітектурні патерни – застосування структурних моделей, таких як Model-View-Controller (MVC) або Model-View-ViewModel (MVVM), що сприяють чіткому розділенню логіки, інтерфейсу та даних, забезпечуючи масштабованість і підтримуваність коду.
- Віртуальні машини та компілятори – використання середовищ виконання або компіляторів, які дозволяють запускати код на різних платформах. Наприклад, Xamarin використовує Mono Runtime для виконання .NET-коду поза межами Windows, тоді як Flutter компілює код на Dart безпосередньо у нативний формат для кожної платформи.

Ці концепції забезпечують створення гнучких, масштабованих і ефективних мобільних рішень, адаптованих до сучасного цифрового середовища (Jošt & Taneski, 2025).

Етапи життєвого циклу. Розроблення кросплатформного мобільного застосунку охоплює низку послідовних етапів, кожен з яких має критичне значення для забезпечення якості, стабільності та ефективності кінцевого продукту (WebCase, н.д.).

Початковий етап – проєктування – є фундаментом для подальшого розроблення, оскільки саме на цьому етапі визначаються функціональні вимоги, архітектура та дизайн інтерфейсу. Особливої уваги потребують аспекти UX (User Experience) та UI (User Interface), адже застосунок має забезпечувати комфортну взаємодію з користувачем незалежно від типу пристрою чи платформи. Розробники можуть використовувати універсальні компоненти, що однаково працюють на всіх платформах, або адаптивні елементи, які

змінюють свою поведінку відповідно до специфіки операційної системи.

На етапі реалізації важливо забезпечити ефективну інтеграцію нативних функцій пристрою, таких як камера, GPS, сенсори тощо. Кросплатформні фреймворки надають відповідні API та плагіни, що дозволяють розширити функціональність застосунку без потреби в окремій реалізації для кожної платформи. Використання спеціалізованих бібліотек для роботи з мультимедіа або геолокацією значно спрощує процес розробки та підвищує її ефективність.

Після завершення програмної реалізації застосунок проходить етап тестування і налагодження, який включає перевірку працездатності на різних пристроях і версіях операційних систем. Застосування методів функціонального та автоматизованого тестування дозволяє виявити та усунути помилки, забезпечуючи стабільну роботу продукту в реальних умовах експлуатації.

Фінальним етапом є впровадження та подальша підтримка застосунку. Публікація в офіційних маркетплейсах (App Store, Google Play) супроводжується регулярними оновленнями, спрямованими на покращення функціональності, підвищення рівня безпеки та адаптацію до нових версій операційних систем. Такий підхід дозволяє підтримувати актуальність і конкурентоспроможність мобільного застосунку протягом усього його життєвого циклу.

Переваги та обмеження. Кросплатформні рішення мають низку суттєвих переваг, що зумовлюють їхню популярність у сучасній мобільній розробці. Насамперед, використання єдиної кодової бази для різних операційних систем (iOS, Android тощо) дозволяє значно скоротити час і витрати на розробку. Замість створення окремих версій застосунку для кожної платформи, розробники можуть зосередитися на реалізації функціональності та оптимізації одного універсального рішення.

Централізоване оновлення та підтримка також є важливою перевагою: зміни в коді вносяться один раз і автоматично застосовуються до всіх платформ, що спрощує процес супроводу та знижує ризик розбіжностей між версіями. Крім того, охоплення широкого спектра пристроїв забезпечує доступ до

більшої аудиторії, сприяючи розширенню ринку та підвищенню конкурентоспроможності продукту.

Втім, кросплатформна розробка має і певні обмеження, які слід враховувати. Деякі специфічні функції або можливості платформи можуть бути недоступними або реалізованими частково через технічні обмеження фреймворків. Додаткові шари абстракції, необхідні для забезпечення універсальності, можуть негативно впливати на продуктивність застосунку, особливо при виконанні ресурсомістких операцій.

Також є ризик некоректного відображення інтерфейсу на різних пристроях та версіях операційних систем, що потребує додаткового тестування та адаптації. Оновлення кросплатформних фреймворків можуть відставати від оновлень нативних SDK, що призводить до затримок у впровадженні нових функцій або виправленні критичних помилок.

Таким чином, хоча кросплатформна розробка є ефективним інструментом для створення універсальних мобільних рішень, її застосування потребує зваженого підходу з урахуванням технічних і бізнесових особливостей проєкту (Foxminded, 2024).

Огляд фреймворків. У сучасній кросплатформній розробці мобільних застосунків використовується низка технологій, які дозволяють створювати універсальні рішення для різних операційних систем. Серед них особливої уваги заслуговують три фреймворки – React Native, Flutter та Xamarin – які демонструють різні підходи до реалізації кросплатформної логіки, мають власні архітектурні особливості, переваги та обмеження (Foxminded, 2024).

React Native, розроблений компанією Meta (раніше Facebook), базується на мові JavaScript та бібліотеці React. Його архітектура передбачає використання JavaScript для опису логіки та інтерфейсу, тоді як рендеринг UI здійснюється через нативні компоненти, що дозволяє створювати застосунки для iOS і Android з єдиною кодовою базою, зберігаючи при цьому нативний вигляд і поведінку.

Перевагами фреймворку є швидкий старт для веб-розробників завдяки JavaScript; велика спільнота та багата екосистема плагінів, можливість інтеграції нативного коду при потребі. До

обмежень слід віднести проблеми продуктивності при складних UI або високих навантаженнях, обмежений доступ до деяких нативних API без додаткових модулів, залежність від мосту між JavaScript і нативним кодом, що може спричинити затримки (Meta, 2020; Wezom, 2022).

Flutter – фреймворк від Google, який використовує мову Dart та власний графічний рушій для рендерингу інтерфейсу. На відміну від React Native, Flutter не покладається на нативні компоненти, а створює UI самостійно, що забезпечує високу продуктивність і повний контроль над виглядом застосунку. Перевагами фреймворку є власний набір віджетів для створення адаптивного та кастомізованого інтерфейсу; висока швидкість рендерингу та плавність анімацій; єдина логіка для UI на всіх платформах. До обмежень слід віднести нову екосистему, менше готових рішень порівняно з React Native; потреба в глибокому знанні Dart, менш поширеної мови; збільшений розмір застосунку через вбудований рушій (Google, н.д.).

Xamarin – технологія від Microsoft, яка дозволяє створювати мобільні застосунки на основі C# та платформи .NET. Вона підтримує два підходи: Xamarin.Forms – для створення спільного UI, та Xamarin.Native – для реалізації інтерфейсів окремо для кожної платформи.

Перевагами фреймворку є повний доступ до нативних API та можливість використання бібліотек .NET; висока інтеграція з екосистемою Microsoft; єдина мова програмування для всіх платформ. До обмежень слід віднести вищу складність при реалізації складного UI; менша гнучкість у дизайні порівняно з Flutter; потреба в додатковій оптимізації для досягнення нативної продуктивності (Microsoft, н.д.; Wezom, 2022). У таблиці 1 наведено порівняльну характеристику розглянутих вище популярних фреймворків для кросплатформної розробки мобільних застосунків. Порівняння охоплює ключові критерії, такі як мова програмування, підхід до рендерингу інтерфейсу, продуктивність, доступ до нативних API, гнучкість дизайну, розмір і активність екосистеми та рівень підтримки. Такий підхід дозволяє наочно оцінити сильні та слабкі сторони кожного фреймворку і допомагає розробникам обрати найбільш

відповідний інструмент залежно від вимог проєкту, ресурсів команди та очікувань кінцевих користувачів.

Таблиця 1

Порівняльна характеристика популярних фреймворків для кросплатформної розробки мобільних застосунків

Критерій	React Native	Flutter	Xamarin
Мова програмування	JavaScript	Dart	C#
Рендеринг UI	Нативні компоненти	Власний рушій	Нативні / загальні UI
Продуктивність	Середня	Висока	Висока (з оптимізацією)
Доступ до нативних API	Через модулі	Через плагіни	Повний доступ
Гнучкість дизайну	Висока	Дуже висока	Обмежена (Forms)
Екосистема	Розвинена	Активно зростає	Стабільна, але вузька
Підтримка	Meta	Google	Microsoft

Практичні кейси використання фреймворків

Вивчення реальних прикладів успішного застосування кросплатформних технологій дає змогу не лише оцінити їхню ефективність у практичному середовищі, а й виявити ключові аспекти, що впливають на вибір фреймворку, архітектурних рішень та підходів до розробки. Аналіз таких кейсів дозволяє розробникам і компаніям краще зрозуміти переваги та обмеження конкретних технологій, а також адаптувати їх до власних потреб. У цьому контексті доцільно розглянути приклади застосунків, реалізованих із використанням розглянутих вище кросплатформних фреймворків – React Native, Flutter і Xamarin – кожен із яких демонструє унікальні можливості відповідної платформи та її здатність забезпечити стабільну, продуктивну і масштабовану роботу мобільного програмного забезпечення.

Instagram активно використовує фреймворк React Native для розробки свого мобільного застосунку, що дозволяє компанії оперативно впроваджувати нові функції, оновлення та вдосконалення інтерфейсу (Meta, 2020). Завдяки використанню

єдиної кодової бази та гнучкої архітектури, React Native забезпечує швидкий і плавний користувацький досвід, ефективне управління станом застосунку та адаптацію компонентів до різних платформ. Це дає змогу підтримувати високу якість відображення та функціональності як на пристроях з iOS, так і з Android. Скріншоти інтерфейсу Instagram (рис. 1), створеного за допомогою React Native, демонструють здатність технології забезпечити візуальну узгодженість, продуктивність і зручність користування незалежно від операційної системи.

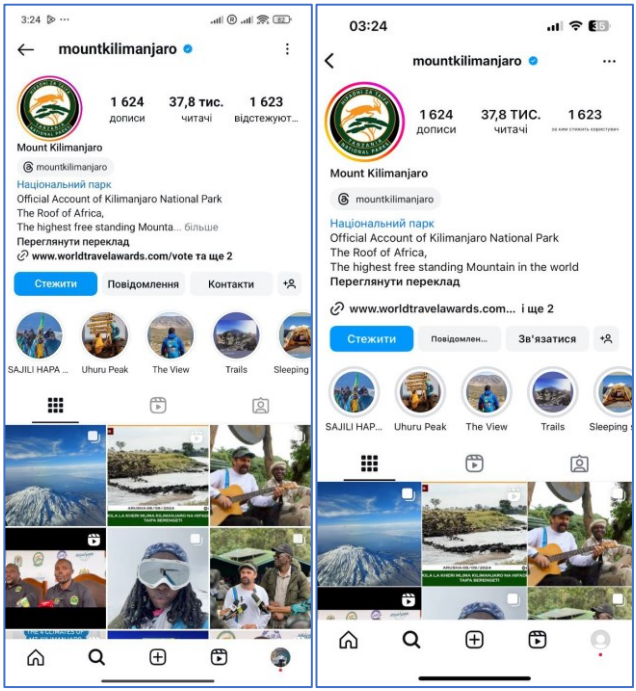


Рис. 1. *Мобільний застосунок Instagram*
зліва – версія для iOS, справа – версія для Android

Google Ads – приклад застосунку, успішно реалізованого за допомогою фреймворку Flutter, який відіграє ключову роль у створенні адаптивного та функціонального інтерфейсу, та коректно працює на різних операційних системах (Google, н.д.). Єдина кодова база дозволяє забезпечити стабільність, високу продуктивність і швидкість розробки. Скріншоти Google Ads (рис. 2) демонструють практичний підхід до дизайну та зручність користування, що стало можливим завдяки гнучкості Flutter у поєднанні з іншими технологічними рішеннями.

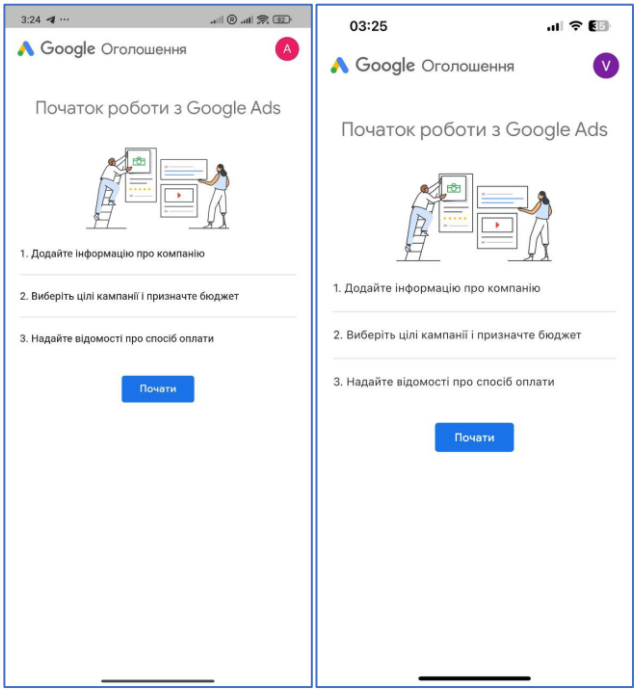


Рис. 2. *Мобільний застосунок Google Ads*
зліва – версія для iOS, справа – версія для Android

Зображення демонструє узгодженість дизайну, адаптивність компонентів та ефективну реалізацію функціональності на обох платформах. Крім того, використання Flutter дозволяє легко

масштабувати застосунок, додаючи нові функції без необхідності дублювання коду для кожної операційної системи. Це особливо важливо для великих компаній, таких як Google, які прагнуть забезпечити однаковий користувацький досвід незалежно від пристрою. Такий підхід сприяє швидкому оновленню інтерфейсу, покращенню продуктивності та зниженню витрат на підтримку застосунку.

Alaska Airlines використовує Xamarin для розробки свого мобільного застосунку, що дозволяє об'єднати нативні функції – такі як GPS, push-сповіщення та доступ до системних ресурсів – в єдину кросплатформну платформу (Microsoft, н.д.).

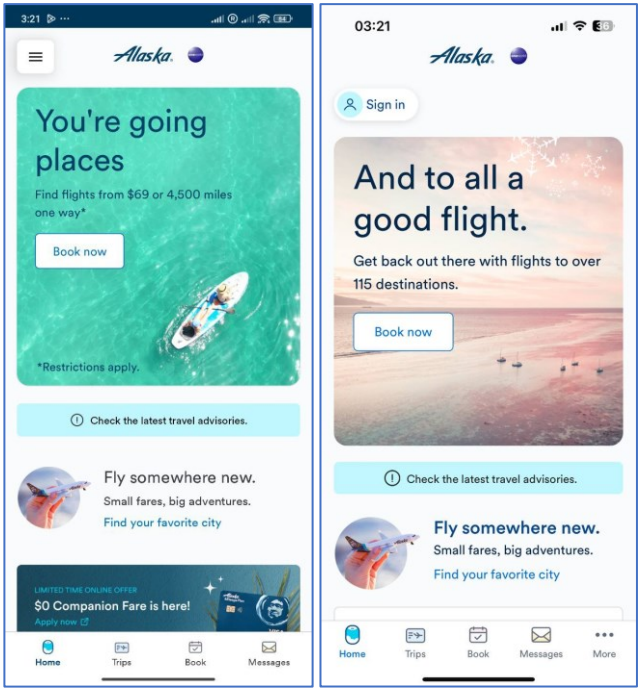


Рис. 3. *Мобільний застосунок Alaska Airlines зліва – версія для iOS, справа – версія для Android*

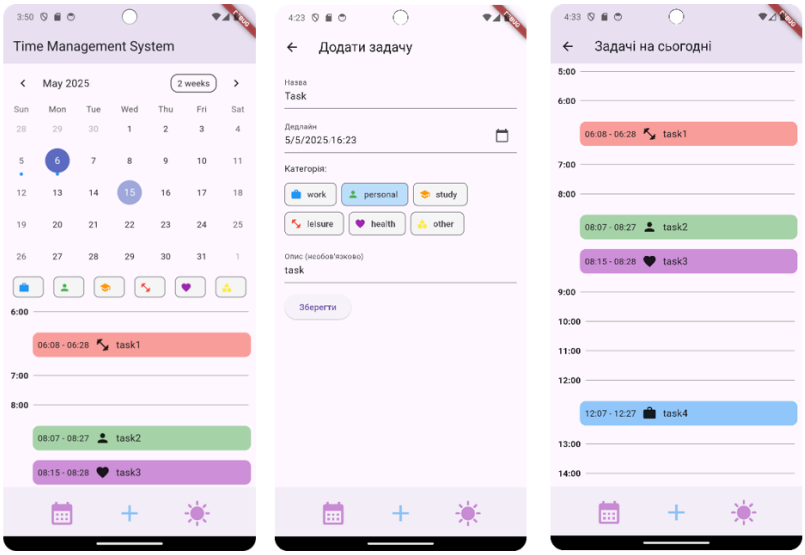
Завдяки Xamarin розробники мають змогу створювати застосунки з високим рівнем продуктивності та інтеграції, використовуючи спільну кодову базу для Android та iOS. Це значно знижує витрати на розробку та спрощує підтримку продукту. Знімки екрану застосунку Alaska Airlines (рис. 3) демонструють зручний інтерфейс і стабільну реалізацію ключових функцій, що забезпечують комфортну взаємодію користувача з сервісом. Крім того, Xamarin дозволяє легко масштабувати застосунок та інтегрувати нові можливості без необхідності дублювати код для кожної платформи. Такий підхід особливо важливий для авіакомпаній, де точність, швидкість оновлень і надійність мають критичне значення для користувацького досвіду.

Власний досвід мобільної розробки. У межах виконання освітнього проєкту було розроблено мобільний застосунок для планування часу. Основною метою проєкту стало створення інструмента, що забезпечує ефективну організацію особистих завдань і підвищує зручність у керуванні повсякденними справами.

Для реалізації було обрано кросплатформний стек технологій, до якого увійшли Flutter Framework, Dart Programming Language, SQLite для роботи з локальною базою даних, а також Visual Studio Code як середовище розробки. Використання цих інструментів дало змогу створити єдину програмну систему, яка коректно функціонує на різних операційних системах, зокрема Android та iOS. Вибір фреймворку Flutter зумовлений його високою продуктивністю, стабільністю інтерфейсу та широким набором готових віджетів, що сприяють створенню сучасного та адаптивного дизайну. Натомість React Native і Xamarin не були використані через складніші механізми налаштування інтерфейсу та обмежену гнучкість у кастомізації UI-компонентів, що поступаються можливостям Flutter у контексті реалізації поставлених завдань.

Розробка передбачала проєктування архітектури системи, створення основної функціональності та інтерфейсу користувача. Окрему увагу було приділено опису алгоритму взаємодії користувача з застосунком, що дозволило чітко продемонструвати

логіку роботи програмної системи. На етапі тестування застосунок було перевірено на пристроях з різними операційними системами. Незважаючи на певні труднощі, пов'язані з налаштуванням середовища та адаптацією інтерфейсу під різні екрани, результати підтвердили стабільність і функціональність системи. На рис. 4 представлено скріншоти роботи розробленого застосунку на платформі Android, зокрема: а) головний екран; б) інтерфейс додавання задачі; в) перегляд задач, запланованих на поточний день.



а) **б)** **в)**

Рис. 4. *Мобільний застосунок Time Management System*
а) головний екран; б) інтерфейс додавання задачі; в) перегляд задач на поточний день

Власний досвід показав, що використання кросплатформних технологій у мобільній розробці є дієвим підходом, що поєднує універсальність, продуктивність і зручність. Flutter забезпечив швидку розробку, багатий набір готових компонентів і простоту роботи з даними через SQLite. У цілому обраний стек технологій виправдав себе в межах освітнього проєкту й продемонстрував

перспективність застосування Flutter для масштабніших рішень у сфері мобільних застосунків.

Дискусія і висновки

Кросплатформна розробка мобільних застосунків відкриває можливості для створення універсальних рішень, що функціонують на різних операційних системах на основі єдиної кодової бази. Такий підхід дозволяє суттєво скоротити витрати часу та ресурсів, спрощує процес підтримки й оновлення, а також сприяє розширенню аудиторії користувачів.

Популярні фреймворки – React Native, Flutter та Xamarin – мають власні переваги й обмеження, що підтверджується успішними практичними кейсами (Instagram, Google Ads і Alaska Airlines). Вибір конкретної технології залежить від вимог проєкту, особливостей дизайну, очікуваної продуктивності та потреби в інтеграції з нативними можливостями платформи.

Власний досвід розробки мобільного застосунку на Flutter підтвердив ефективність цієї технології як інструмента для швидкої, гнучкої та масштабованої мобільної розробки, що успішно зарекомендував себе в межах освітнього проєкту.

Отже, кросплатформний підхід є перспективним напрямом розвитку мобільного програмного забезпечення, який поєднує гнучкість, економічність і високу швидкість створення цифрових продуктів. Водночас він потребує ретельного планування архітектури та вибору інструментів для досягнення максимальної ефективності.

Список використаних джерел

Foxminded. (2024). *Переваги кросплатформного застосунку*. https://foxminded.ua/krosplatformnyi-zastosunok/

Google. (н.д.). *Flutter Documentation*. Google Developers. https://flutter.dev/docs

Jošt, G., & Taneski, V. (2025). State-of-the-art cross-platform mobile application development frameworks: A comparative study of market and developer trends. Informatics, 12(2), 45. https://doi.org/10.3390/informatics12020045

Meta. (2020). *React Native at Instagram*. Meta Engineering Blog. https://engineering.fb.com/2020/07/13/android/react-native-at-instagram

Microsoft. (н.д.). *Xamarin Fundamentals*. Microsoft Learn. https://learn.microsoft.com/en-us/xamarin

WebCase. (н.д.). *Етапи створення мобільного додатку*. https://webcase.com.ua/uk/blog/etapi-stvorennja-mobilnogo-dodatku

Wezom. (2022). *React Native vs Xamarin: Яка платформа краща для розробки?* https://wezom.com.ua/ua/blog/react-native-vs-xamarin

References

Foxminded. (2024). *Advantages of cross-platform application development* [in Ukrainian]. https://foxminded.ua/krosplatformnyi-zastosunok/

Google. (n.d.). *Flutter documentation*. Google Developers. https://flutter.dev/docs

Jošt, G., & Taneski, V. (2025). *State-of-the-art cross-platform mobile application development frameworks: A comparative study of market and developer trends*. *Informatics, 12*(2), 45. https://doi.org/10.3390/informatics12020045

Meta. (2020). *React Native at Instagram*. Meta Engineering Blog. https://engineering.fb.com/2020/07/13/android/react-native-at-instagram

Microsoft. (n.d.). *Xamarin fundamentals*. Microsoft Learn. https://learn.microsoft.com/en-us/xamarin

WebCase. (n.d.). *Stages of mobile application development* [in Ukrainian]. https://webcase.com.ua/uk/blog/etapi-stvorennja-mobilnogo-dodatku

Wezom. (2022). *React Native vs Xamarin: Which platform is better for development?* [in Ukrainian]. https://wezom.com.ua/ua/blog/react-native-vs-xamarin

Отримано редакцією журналу / Received:29.09.2025
Прорецензовано / Revised:30.09.2025
Схвалено до друку / Accepted:01.10.2025

Viktoria HOLOTIUK, Student
ORCID ID: 0009-0008-3947-4613
e-mail: viktoriia_holotiuk@knu.ua
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

Olena VASHCHILINA, PhD (Phys. & Math.), Assoc. Prof.
ORCID ID: 0000-0001-6867-6216
e-mail: olenavashchilina@knu.ua
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT: CORE PRINCIPLES, TECHNOLOGIES, AND PRACTICAL APPROACHES

The article explores the fundamental principles and technologies of cross-platform mobile application development. It highlights the key aspects of this approach and analyzes its advantages and limitations that affect the quality and efficiency of software product creation. Special attention is given to the stages of the application lifecycle – from design to maintenance – which ensure the stability and competitiveness of solutions. The paper provides an overview of popular

frameworks – React Native, Flutter, and Xamarin – with a comparative analysis of their capabilities. Practical examples of these technologies used in well-known mobile products (Instagram, Google Ads, Alaska Airlines) demonstrate the effectiveness and potential of the cross-platform approach. The author's experience in developing a mobile application using Flutter confirms the practical relevance of the chosen technology stack within an educational project.

Keywords: ***Cross-platform development, mobile applications, React Native, Flutter, Xamarin***

Автори заявляють про відсутність конфлікту інтересів. Спонсори не брали участі в розробленні дослідження; у зборі, аналізі чи інтерпретації даних; у написанні рукопису; в рішенні про публікацію результатів.

The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.