DOI: https://doi.org/10.36910/6775-2524-0560-2024-56-30
УДК: 004.438:004.4'042.42/.43:004.432.4:004.432.23:004.382.745
Пахомов Сергій Володимирович[1], архітектор рішень
https://orcid.org/0009-0007-1571-4611
Муляревич Олександр Володимирович[2], к.т.н., доцент
https://orcid.org/0000-0002-4644-7962
Боярінова Юлія Євгенівна[1], к.т.н., с.н.с., доцент
https://orcid.org/0000-0002-8974-529X
[1] Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ, Україна
[2] Національний університет «Львівська політехніка», м. Львів, Україна

РОЗРОБКА ТА ОПТИМІЗАЦІЯ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ РІЗНИХ ПЛАТФОРМ ІЗ ВИКОРИСТАННЯМ МОВ ПРОГРАМУВАННЯ C, C++, C#, JAVA

Пахомов С.В., Муляревич О.В., Боярінова Ю.Є. Розробка та оптимізація мобільних додатків для різних платформ із використанням мов програмування C, C++, C#, Java. У процесі роботи було здійснено огляд методів розробки та оптимізації мобільних додатків для різних платформ на основі використання різноманітних мов програмування C, C++, C#, Java. Виявлено основні вимоги платформ та їх користувачів до мобільних додатків. Згадано фреймворки та бібліотеки, що допомагають реалізувати мультиплатформність та переваги й недоліки їх використання. Також запропоновано застосування обраних мов програмування відповідно до особливостей кожної з них, що зробить розробку мобільних додатків простішою та ефективнішою, на основі чого були надані практичні рекомендації для оптимізації мобільних застосунків. Ці рекомендації можуть бути задіяні як на рівні відділів корпорацій, так і індивідуальними розробниками чи фрілансерами, що робить їх універсальними. Отримані рекомендації можуть стати основою для зміни парадигми розробки та оптимізації мобільних додатків, що здатне змінити парадигму програмування з використанням мов C, C++, C#, Java загалом. Саме питання зміни парадигми програмування з використанням мов C, C++, C#, Java не тільки для мобільних додатків, але й для десктопних застосунків (програм, які встановлюються на локальний комп'ютер або ноутбук) визначено актуальним для наступних досліджень.
Ключові слова: смартфони**,** парадигми програмування, мобільні застосунки, операційна система, бібліотеки, фреймворки

Pakhomov S., Muliarevych O., Boiarinova Yu. Development and optimization of mobile applications for various platforms using C, C++, C#, Java programming languages. In the course of the work, a review of methods for developing and optimizing mobile applications for different platforms was conducted based on the use of various programming languages C, C++, C#, Java. The main requirements of platforms and their users to mobile applications are identified. The author mentions frameworks and libraries that help to implement multiplatform and the advantages and disadvantages of using them. It is also proposed to use the selected programming languages in accordance with the features of each of them, which will make the development of mobile applications easier and more efficient, based on which practical recommendations for optimizing mobile applications were provided. These recommendations can be used both at the level of corporate departments and by individual developers or freelancers, which makes them universal. The obtained recommendations can become the basis for changing the paradigm of mobile application development and optimization, which can change the paradigm of programming using C, C++, C#, and Java languages in general. The issue of changing the programming paradigm using C, C++, C#, and Java languages not only for mobile applications but also for desktop applications (programs that are installed on a local computer or laptop) is identified as relevant for further research.
Keywords: smartphones, programming paradigms, mobile applications, operating system, libraries, frameworks

Постановка наукової проблеми. Станом на 2022 рік 67% людей у всьому світі користувалися мобільними телефонами. Смартфони мали 80% з них, а з інтернет-користувачів – 96% [1]. Це свідчить про важливість, яку ці гаджети відіграють у суспільстві: від дзвінків та ігор до отримання кредитної інформації від банку чи прогнозу погоди. Зараз створюється все більше різноманітних мобільних застосунків, які допомагають їх користувачам виконувати широкий спектр завдань.

Щоб забезпечити успішність використання мобільного застосунку, розробники прагнуть максимізувати якість користувацького досвіду, а отже, і задоволеність користувачів [2]. Було виявлено що успіх мобільних застосунків, в контексті кількості їх завантажень, корелює з рейтингом, який отримує додаток в магазині додатків (наприклад, Google Play чи Apple Store) [3]. У 2018 році загальна кількість завантажених мобільних застосунків склала 194 мільярди, при цьому кожен користувач смартфона має багато встановлених на своєму телефоні додатків [4].

Головними проблемами в роботі мобільних застосунків зараз стали питання безпеки, складності обчислень та високого енергоспоживання. Серед усіх причин низьких оцінок додатків надмірне споживання ними енергії спричинило найбільшу кількість видалень зі смартфонів [5].

Одним зі способів розв'язання цієї потрійної проблеми є створення гібридних додатків, які використовують дві або більше мов програмування. Такі застосунки будуть оптимізувати та використовувати кожну мову програмування. Усе це потребує зміни парадигми програмування та уніфікації, що можливе тільки при використанні бібліотек, фреймворків і розширень, які не конфліктують між собою.

Формулювання мети і завдань дослідження. Метою статті є огляд та виявлення особливостей використання методів розробки та оптимізації мобільних додатків для різних платформ на основі мов програмування C, C++, C# та Java.

Для досягнення поставленої мети були сформульовані такі завдання:

1. З'ясувати особливості мов програмування C, C++, C# та Java в контексті фреймворків, бібліотек і розширень.
2. Дослідити сумісність різних мобільних платформ (операційних систем) та додатків.
3. Визначити головні вимоги потенційних користувачів до мобільних застосунків.
4. Розробити практичні рекомендації для оптимізації мобільних застосунків.

Серед загальнонаукових методів для розв'язання завдань дослідження й досягнення його мети були використані такі:

– метод моніторингу: застосовувався для збору, систематизації та аналізу інформації про мови програмування та мобільні застосунки;

– метод порівняння: використовувався під час дослідження різних мобільних платформ та їх порівняння;

– метод абстрагування: застосовувався для виокремлення основних понять та категорій;

– методи аналізу та синтезу: використовувалися в процесі ідентифікації етапів і факторів розвитку, а також найвпливовіших елементів досліджуваного об'єкта.

Для розв'язання загальних завдань поточного дослідження було використано такі групи спеціальних методів: методи збору інформації; методи обробки інформації; методи проведення аналітичної роботи; метод обґрунтувань.

Актуальність досліджуваної теми аргументується важливістю питання контролю якості мобільних застосунків, що дозволяє користувачам робити своє життя простішим та зручнішим, а розробникам – залучати кошти і, на основі попереднього досвіду, створювати все якісніші та сучасніші мобільні застосунки, які позбавлені минулих недоліків та помилок.

Аналіз останніх досліджень та публікацій. Сьогодні тема розробки та оптимізації мобільних додатків для різних платформ із використанням мов програмування C, C++, C# та Java перебуває у фокусі наукової уваги багатьох вітчизняних і закордонних дослідників та практиків у галузі інформаційних технологій. Серед українських учених необхідно згадати О. Сікору, Т. Кобильника, Н. Єрьоміну, М. Шапошника, В. Хамбіра, Н. Ічанську, С. Улько, І. Гунька та інших.

У статті О. Сікори та Т. Кобильника розглянуто питання вибору мови програмування в шкільній освіті. Визначено, що такою може бути Java, не в останню чергу через можливість її використання для створення мобільних додатків [6].

Н. Єрьоміна та М. Шапошник вивчали методи тестування мобільних застосунків, де Java була визнана як зручна мова програмування для написання тестів (автоматизованого тестування), оскільки саме Java-додатки набагато простіше запускати на різних платформах [7]. В. Хамбір також згадував про критичну важливість автоматизації тестування мобільних застосунків для їх подальшої оптимізації [8].

Н. Ічанська та С. Улько описали основні відмінності нативної й багатоплатформної розробок мобільних додатків у контексті мов програмування, бібліотек, фреймворків та розширень [9].

У своїй науковій праці І. Гунько виклав своє бачення питання тестування програмного забезпечення через призму передових тенденцій: використання засобів штучного інтелекту та машинного навчання, практик DevOps та блокчейну [10].

З іноземних дослідників питання розробки та оптимізації мобільних додатків для різних платформ із використанням мов програмування C, C++, C# та Java варто зазначити таких учених, як M. Hort, M. Kechagia, F. Sarro, M. Harman, M. Farooq, S. Riaz, A. Alvi, A. Ali, I. Rehman, J. Ogala та D. Ojie.

У статті таких науковців, як M. Hort, M. Kechagia, F. Sarro, M. Harman представлено всебічний огляд оптимізації нефункціональної продуктивності для Android-додатків на основі показників швидкості відгуку, часу запуску, споживання пам'яті та енергії [11].

M. Farooq, S. Riaz, A. Alvi, A. Ali та I. Rehman розглядали підходи та інструменти розробки кросплатформних мобільних додатків. Ними також були визначені основні виклики у сфері кросплатформності та шляхи їх подолання [12].

J. Ogala та D. Ojie провели дослідження мов програмування C, C++, C# та Java. У процесі їхньої роботи було узагальнено технічні особливості цих чотирьох мов програмування та здійснено їх порівняння шляхом реалізації бенчмарку для кожної з них [13].

Проаналізувавши матеріали, які на сьогодні є у відкритому доступі, можна зробити висновок, що тема розробки та оптимізації мобільних додатків для різних платформ із використанням мов програмування C, C++, C# та Java є актуальною та цікавою для багатьох дослідників. Проте здійснений аналіз дозволяє констатувати відсутність єдиного універсального підходу до вивчення питання трансформації сучасної парадигми програмування.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження. Для ґрунтовного висвітлення обраної проблематики дослідження необхідно визначити особливості мов програмування C, C++, C# та Java. C – це мова програмування загального призначення, що була створена в 1970-х роках Деннісом Рітчі й залишається широко використовуваною та впливовою навіть зараз. За задумом її творця, функції мови C чітко відображають можливості цільових процесорів. Вона знайшла тривале застосування в коді операційних систем (особливо в ядрах), драйверах пристроїв і стеках протоколів, але її використання в прикладному програмному забезпеченні зменшується. Також вона широко використовується в комп'ютерних архітектурах, які варіюються від найбільших суперкомп'ютерів до найменших мікроконтролерів та вбудованих систем [14].

Потрібно зазначити що C – це імперативна процедурна мова, що підтримує структуроване програмування, лексичну область видимості змінних та рекурсію зі статичною системою типів. Вона була розроблена для компіляції, щоб забезпечити низькорівневий доступ до пам'яті та мовних конструкцій, які ефективно відображаються в машинній інструкції, і все це з мінімальною підтримкою під час виконання. Попри свої низькорівневі можливості, мова була розроблена для заохочення кросплатформного програмування. Відповідно до стандартів програма C, написана з думкою про переносність, може бути скомпільована для широкого спектра комп'ютерних чи мобільних платформ та операційних систем з невеликими змінами у вихідному коді [15].

Мова програмування C призначена не лише для настільних комп'ютерів та серверів. При правильному підході вона може стати цінним інструментом для розробки мобільних додатків. Дуже важливо мати сприятливе середовище для розробки. Це гарантує, що процес програмування буде безперебійним і без непотрібних затримок. Першим кроком є вибір інтегрованого середовища розробки (IDE), придатного для програмування мовою C. Існує багато IDE, але найпопулярнішими з них для C є Eclipse CDT, Code::Blocks та CLion. Особливості їх використання представлені в табл. 1.

Таблиця 1. Особливості інтегрованого середовища розробки Eclipse CDT, Code::Blocks та CLion

№	Назва IDE	Головні особливості
1	Eclipse CDT	Підтримує статичний аналіз коду
		Має чудовий графічний інтерфейс користувача з функцією перетягування.
		Інтегрується з Git
2	Code::Blocks	Інтерфейс із вкладками, завершення коду, зручна навігація
		Підтримує різні компілятори – GCC, Clang та Visual C++
		Доступна повна підтримка точок зупинки
3	CLion	Забезпечує безпечний рефакторинг
		Дозволяє тестувати окремі одиниці вихідного коду
		Містить вбудований термінал
		Дозволяє проводити автоматичний аналіз коду

Розробка мобільних додатків часто вимагає спеціальних бібліотек для взаємодії з мобільною операційною системою. Наприклад, якщо розробка створена для Android, то може знадобитися NDK (Native Development Kit). Він дозволяє запускати код написаний на мовах програмування C та

C++ на пристроях Android. Компілятор необхідний для перетворення С-коду в машинний код. Для розробки мобільних додатків необхідно переконатися, що компілятор налаштований на відповідну мобільну архітектуру, наприклад ARM або x86.

Емулятор дозволяє протестувати створений мобільний додаток на комп'ютері без використання реального пристрою. Такі інструменти, як Android Studio, мають вбудовані емулятори для пристроїв Android. Важливо переконатися, що емулятор може безперебійно запускати програми на мові С. При розробці мобільних застосунків на мові програмування С бібліотеки та фреймворки, які обирає програміст, можуть суттєво вплинути на продуктивність, можливості та швидкість розробки його додатку, тому правильний вибір цих інструментів має вирішальне значення. Якщо для розробки на Android незамінним є NDK (Native Development Kit), то для iOS потрібно обрати бібліотеки, які взаємодіють з Objective-C, оскільки вона є підмножиною С.

Такі фреймворки, як SDL (Simple DirectMedia Layer) та Cocos2d-x, дозволяють програмісту написати додаток один раз і розгорнути його на різних платформах. Вони особливо корисні, якщо творці мобільного застосунку прагнуть орієнтуватися як на Android, так і на iOS без переписування коду. Саме ці фреймворки допомагають реалізувати мультиплатформність. Одним із найсучасніших універсальних фреймворків, що підтримує всі основні мови програмування та сумісний з Objective-C, є React Native. Як і будь-яка технологія його використання має свої переваги та недоліки (табл. 2).

Таблиця 2. Переваги та недоліки кросплатформного фреймворку React Native

№	Переваги	Недоліки
1	Багаторазове використання коду	Обмежений нативний доступ
2	Нативна продуктивність	Обмеження продуктивності
3	Гаряче перезавантаження	Відсутність кастомізації

Пояснимо переваги та недоліки фреймворку React Native більш детально:

1. Багаторазове використання коду: фреймворк React Native дозволяє розробнику написати код один раз і використовувати його на обох мобільних платформах (Android та iOS).
2. Нативна продуктивність: розробник може створювати додатки, які мають такий же рівень продуктивності, як і нативні.
3. Гаряче перезавантаження: фреймворк має функцію гарячого перезавантаження, яка дозволяє програмісту виправляти код з огляду на зміни в реальному часі.
4. Обмежений нативний доступ: розробник може стикатися з певними обмеженнями при створенні застосунків, які вимагають доступу до нативних API.
5. Обмеження продуктивності: хоч фреймворк і дозволяє створювати додатки, які мають ознаки нативних, але можуть даватися взнаки обмеження продуктивності при створенні саме складних застосунків.
6. Відсутність кастомізації: React Native використовує раніше зібрані компоненти, які не можуть бути налаштовані так, як цього потребують деякі додатки.

Для обчислювально-інтенсивних завдань можна використовувати такі бібліотеки, як OpenCL та OpenGL. Вони допомагають досягти кращого рендерингу графіки та паралельних обчислень на мобільних пристроях. Важливим також є питання створення надійної та масштабованої архітектури мобільного додатку, оскільки це має вирішальне значення для його подальшого успіху. Модульний дизайн застосунку гарантує, що кожен його компонент функціонує незалежно, що забезпечує кращу масштабованість та придатність до виправлень. Також додаток необхідно розділити на різні шари, такі як презентація, бізнес-логіка та доступ до даних, що гарантує незмінність усього додатку при зміні в одному з них. Для підвищення продуктивності роботи додатка потрібно використати принцип паралелізму, оскільки більшість сучасних смартфонів мають багатоядерні процесори [17].

Після створення застосунку необхідно його оптимізувати, зважаючи на обмеження щодо пам'яті, місткості акумулятора та обчислювальної потужності смартфонів. Жодний із сучасних додатків не може стати успішним серед користувачів, якщо буде мати критичні проблеми з надмірним споживанням енергії, оскільки це не дозволить користуватися ним регулярно. Найважливіші шляхи оптимізації застосунку, що створений із використанням мови С, показано на рис. 1.

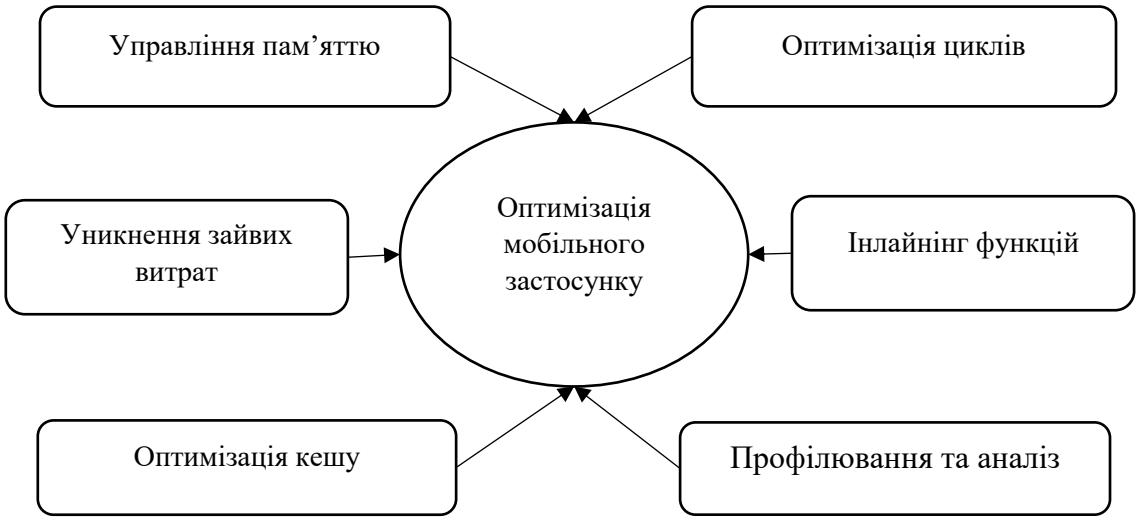

Рис. 1. Шляхи оптимізації мобільного застосунку

Сутність оптимізації мобільних застосунків, що розроблені на основі мови С, пояснена далі детальніше:

1. Управління пам'яттю: ефективне управління пам'яттю має вирішальне значення, тому потрібно уникати витоків пам'яті й переконатися в тому, що динамічно виділена пам'ять звільняється, коли вона більше не потрібна.
2. Оптимізація циклів: розгортання циклів та мінімізація накладних витрат на цикли може призвести до значного прискорення роботи.
3. Уникнення зайвих витрат: деякі операції, такі як ділення або математичні функції, можуть бути витратними, тому там, де це можливо, необхідно використовувати побітові операції або таблиці пошуку.
4. Інлайнінг функцій: вбудовування невеликих функцій може зменшити накладні витрати на виклики функцій, що призводить до швидшого виконання.
5. Оптимізація кешу: оптимізація використання кешу може значно підвищити продуктивність. Відповідно, треба переконатися, що дані, до яких часто звертаються, зберігаються безперервно, щоб скористатися перевагами локальності кешу.
6. Профілювання та аналіз: такі інструменти, як gprof або Valgrind, можуть допомогти визначити області, які потребують оптимізації [18].

Узагальнюючи питання оптимізації мобільних застосунків, які розроблені на основі мови С, потрібно зауважити, що концептуально вона визначається як поєднання розуміння апаратних обмежень та використання низькорівневих можливостей С.

На жаль, Java та C++ не мають такої універсальності в контексті створення додатків як С та C#, які можуть використовуватися для роботи з платформами Android і iOS. Java та C++ можна використовувати виключно для створення застосунків для Android, що є серйозним обмеженням для розробників. Саме тому для багатоплатформної розробки застосовують фреймворки, де основою є такі мови, як С та C#. Існує чотири основні способи створення додатків для Android та iOS:

1. Окремі нативні додатки для кожної операційної системи: при створенні нативних додатків розробники створюють додаток для певної операційної системи й покладаються на інструменти та мови програмування, розроблені спеціально для однієї платформи: Kotlin або Java для Android, Objective-C або Swift для iOS.
2. Прогресивні вебдодатки (PWA): розробники створюють PWA за допомогою вебтехнологій, таких як JavaScript, HTML, CSS і WebAssembly. Вони не потребують окремого пакування чи поширення й можуть бути опубліковані в Інтернеті та доступні через браузер на комп'ютері, смартфоні, планшеті. До того ж їх не потрібно встановлювати через Google Play або App Store.

3. Кросплатформні додатки: багатоплатформні застосунки призначені для однакової роботи на різних мобільних платформах. Кросплатформні фреймворки дозволяють писати спільний і багаторазовий код для розробки таких додатків.

4. Гібридні додатки: гібридна розробка додатків – це підхід, який поєднує нативні та вебтехнології. Він вимагає вбудовування коду, написаного мовою веброзробки, наприклад HTML, CSS або JavaScript, у нативний додаток. Це можна зробити за допомогою фреймворків, таких як Ionic Capacitor та Apache Cordova, використовуючи додаткові плагіни для доступу до нативних функцій платформ [19].

Вибір між кросплатформною та нативною розробкою залежить від цілей самого розробника застосунку. Якщо мета проєкту – довгостроковий розвиток, де необхідна безперебійна робота й швидке реагування, то бажано обирати нативну розробку. Також, хоч офіційно і вважається що кросплатформна розробка робить сам процес менш затратним, але вона призводить до збільшення необхідного часу розробки та інших неявних негативних факторів у процесі створення застосунку.

Попри те, що для створення мобільних застосунків можуть використовуватися різні мови програмування, фреймворки та бібліотеки, користувачі здебільшого мало цікавляться деталями розробки або й зовсім нічого не розуміють у цьому. Їх вимоги до роботи застосунку відрізняються залежно від особистих уподобань, але можна виокремити декілька основних із них, без задоволення яких мобільний додаток не здобуде успіху.

Вимоги користувачів часто збігаються з вимогами магазинів різних платформ, таких як AppStore та PlayMarket. Ці вимоги переважно базуються на гайдлайнах – збірниках головних правил, які обов'язкові до виконання для створення додатка. Так прийнято називати вимоги до дизайну застосунку та принципів зворотного зв'язку, якими повинні керуватися розробники при створенні додатка.

Більшість магазинів висувають досить реалістичні та інтуїтивно зрозумілі вимоги до розроблених додатків. Для розробки додатків під iOS використовуються Human Interface Guidelines, для Android – Material Design. Основні вимоги до дизайну додатків згідно з гайдлайнами зображено на рис. 2.

Рис. 2. Вимоги до дизайну додатків згідно з гайдлайнами

Серед додаткових вимог AppStore та PlayMarket до застосунку є наявність брендингу (кольори, формулювання, шрифти чи анімації розробників), оптимізації, свободи вибору й високого рівня безпеки.

На основі попередньо розглянутих даних можна надати низку рекомендацій, що допоможуть розробникам ефективно оптимізувати їхні мобільні застосунки. Оптимізація продуктивності мобільних додатків передбачає підвищення швидкості їх роботи та відгуку на дії користувачів і збільшення ефективності роботи програми. Добре оптимізований додаток не тільки покращує користувацький досвід, але й сприяє успіху бізнесу та розробників. Для покращення ефективності оптимізації мобільних додатків треба використовувати такі практики:

– проводити тестування продуктивності перед запуском застосунку (допомагає визначити час відгуку та обсяг використання пам'яті);

– видаляти невикористаний код і ресурси, стискати зображення та відео- і аудіофайли (чим менший розмір програми, тим швидше вона завантажується);

– зменшити кількість HTTP-запитів, мінімізувати споживання ресурсів і кешування даних, які часто використовуються (цей метод оптимізації допомагає підвищити швидкість програми та скоротити час її завантаження);

– використовувати мережу доставки вмісту (CDN), що поширює вміст програми серед користувачів на основі їхнього географічного розташування (покращує взаємодію з користувачем та зменшує навантаження на сервер);

– регулярно оновлювати додаток та виправляти помилки (допомагає покращити продуктивність програми);

– використовувати ефективні методи кодування, обмежити фонові процеси та зменшити використання анімації та візуальних ефектів (менше споживання заряду батареї).

Підсумовуючи, зазначимо, що оптимізація продуктивності мобільного додатка має вирішальне значення для його популярності серед користувачів та успіху бізнесу розробників. Проводячи тестування продуктивності, зменшуючи розмір програми, оптимізуючи код програми, використовуючи CDN, впроваджуючи оновлення, мінімізуючи розряд акумулятора та оптимізуючи для різних пристроїв, розробники здатні покращити продуктивність програми та вивести взаємодію з користувачем на більш високий рівень.

Висновки та перспективи подальшого дослідження. У сучасних умовах інтенсивного розвитку інформаційних технологій неможливо уявити життя людини без мобільних застосунків. У процесі роботи було здійснено огляд методів розробки та оптимізації мобільних додатків для різних платформ на основі використання різноманітних мов програмування C, C++, C#, Java, зокрема нативних та кросплатформних методів. Визначно основні вимоги платформ та їх користувачів до мобільних додатків, що збігаються з вимогами гайдлайнів магазинів. Згадано фреймворки та бібліотеки, що допомагають реалізувати мультиплатформність та переваги й недоліки їх використання. Також запропоновано застосування обраних мов програмування відповідно до особливостей кожної з них, що зробить розробку мобільних додатків простішою та ефективнішою. На цій підставі було надано практичні рекомендації для оптимізації мобільних застосунків. Ці рекомендації можуть бути задіяні як на рівні відділів корпорацій, так і індивідуальними розробниками чи фрілансерами, що робить їх універсальними. Отримані рекомендації повинні стати основою для зміни парадигми розробки та оптимізації мобільних додатків, що здатне змінити парадигму програмування з використанням мов C, C++, C#, Java загалом. Питання зміни парадигми програмування з використанням мов C, C++, C#, Java не тільки для мобільних додатків, але й для десктопних застосунків (програм, які встановлюються на локальний комп'ютер або ноутбук) визначено актуальним для наступних досліджень.

Список бібліографічного опису

1. Нагорна А. Дослідження: 63% населення планети мають доступ до інтернету, а в 67% є мобільний телефон. *dev.ua*. URL: https://dev.ua/news/63-naselennia-planety-maiut-internet (дата звернення: 22.07.2024).
2. Al-Subaihin A. A., Sarro F., Black S., Capra L., Harman M. App store effects on software engineering practices. *IEEE Transactions on Software Engineering*. 2019. Vol. 47(2). P. 300–319. DOI: 10.1109/TSE.2019.2891715
3. Sällberg H., Wang S., Numminen E. The combinatory role of online ratings and reviews in mobile app downloads: an empirical investigation of gaming and productivity apps from their initial app store launch. *Journal of Marketing Analytics*. 2023. Vol. 11(3). P. 426–442. URL: https://link.springer.com/article/10.1057/s41270-022-00171-w
4. Simon J. P. The global internet market (s): a reconstruction of the views of the industry. *Digital Policy, Regulation and Governance*. 2020. Vol. 22(2). P. 109–133. URL: https://www.emerald.com/insight/content/doi/10.1108/DPRG-10-2019-0092/full/html (дата звернення: 22.07.2024).
5. Dahdouh M., Bouchi A., Khawatmi S., Ayman M. Programmatic effect of optimized smali code on saving energy of android applications. *Int. J. Comput. Appl*. 2020. Vol. 177(42). P. 33–41. DOI:10.5120/ijca2020919928
6. Сікора О. В., Кобильник Т. П. Java як засіб навчання учнів основ програмування. *Вчені записки ТНУ імені В. І. Вернадського. Серія: Технічні науки*. 2023. № 34(73). С. 224–230. DOI: https://doi.org/10.32782/2663-5941/2023.5/35
7. Єрьоміна Н., Шапошник, М. Методи тестування мобільних застосунків. *Scientific Collection «InterConf»*. 2023. № 178. С. 322–325. URL: https://archive.interconf.center/index.php/conference-proceeding/article/view/4715 (дата звернення: 22.07.2024).
8. Хамбір В. Автоматизація процесів тестування мобільних застосунків. *Computer-integrated technologies: education, science, production*. 2024. № 55. С. 213–224. DOI: https://doi.org/10.36910/6775-2524-0560-2024-55-27
9. Ічанська Н., Улько С. Основні аспекти створення мобільних додатків та вибір інструментів їх розробки. *Системи управління, навігації та зв'язку. Збірник наукових праць*. 2020. Вип. 1(59). С. 74–78. DOI: https://doi.org/10.26906/SUNZ.2020.1.074
10. Гунько І. Software testing in 2023: new trends and challenges. *Herald of Kyiv Institute of Business and Technology*. 2023. Вип. 49(1-2). С. 25–36. DOI: https://doi.org/10.37203/kibit.2023.49.03
11. Hort M., Kechagia M., Sarro F., Harman M. A survey of performance optimization for mobile applications. *IEEE Transactions on Software Engineering*. 2021. Vol. 48(8). P. 2879–2904. URL: https://ieeexplore.ieee.org/abstract/document/9397392 (дата звернення: 22.07.2024).
12. Farooq M. S., Riaz S., Alvi A., Ali A., Rehman I. U. Cross-Platform Mobile Development Approaches and Frameworks. *VFAST Transactions on Software Engineering*. 2022. Vol. 10(2). P. 79–93. DOI: https://doi.org/10.21015/vtse.v10i2.978
13. Ogala J. O., Ojie D. V. Comparative analysis of C, C++, C# and Java programming languages. *GSJ*. 2020. Vol. 8(5). P. 1899–1913. URL: https://unidel.edu.ng/cms/uploads/publications/unidel_pub_1679733436.pdf (дата звернення: 22.07.2024).
14. Klabnik S., Nichols C. The Rust programming language. No Starch Press, 2023. URL: https://ejudge.lksh.ru/lang_docs/The%20Rust%20Programming%20Language.pdf (дата звернення: 22.07.2024).

15. Artigues V., Kormann K., Rampp M., Reuter K. Evaluation of performance portability frameworks for the implementation of a particle-in-cell code. *Concurrency and Computation: Practice and Experience*. 2020. Vol. 32(11). P. 1–23. DOI: https://doi.org/10.1002/cpe.5640
16. GeeksforGeeks. 7 Best IDEs For C/C++ Developers in 2024. URL: https://www.geeksforgeeks.org/best-ides-for-c-c-plus-plus-developers/ (дата звернення: 22.07.2024).
17. MarketSplash. C Programming Mobile App Development: Techniques And Tips. URL: https://marketsplash.com/c-programming-mobile-app-development/ (дата звернення: 22.07.2024).
18. Mobile Growth Association. How to Optimize Your Code in Mobile Applications – 10 Pro Tips. URL: https://mobilegrowthassociation.com/how-to-optimize-your-code-in-mobile-applications-10-pro-tips/ (дата звернення: 22.07.2024).
19. Stoyko T. How to Make an App both for Android and iOS. *Incora*. URL: https://incora.software/insights/make-app-both-iOS-Android (дата звернення: 22.07.2024).

References

1. Nahorna, A. (2022). Doslidzhennya: 63% naselennya planety mayut dostup do internetu, a v 67% ye mobilnyy telefon [Research: 63% of the world's population has access to the Internet, and 67% have a mobile phone]. Retrieved from https://dev.ua/news/63-naselennia-planety-maiut-internet [in Ukrainian].
2. Al-Subaihin, A. A., Sarro, F., Black, S., Capra, L., & Harman, M. (2019). App store effects on software engineering practices. *IEEE Transactions on Software Engineering*, 47(2), 300–319. DOI: 10.1109/TSE.2019.2891715 [in English].
3. Sällberg, H., Wang, S., & Numminen, E. (2023). The combinatory role of online ratings and reviews in mobile app downloads: an empirical investigation of gaming and productivity apps from their initial app store launch. *Journal of Marketing Analytics*, 11(3), 426–442. Retrieved from https://link.springer.com/article/10.1057/s41270-022-00171-w [in English].
4. Simon, J. . (2020). The global internet market (s): a reconstruction of the views of the industry. *Digital Policy, Regulation and Governance*, 22(2), 109–133. Retrieved from https://www.emerald.com/insight/content/doi/10.1108/DPRG-10-2019-0092/full/html [in English].
5. Dahdouh, M., Bouchi, A., Khawatmi, S., & Ayman, M. (2020). Programmatic effect of optimized smali code on saving energy of android applications. *Int. J. Comput. Appl*, 177(42), 33–41. DOI:10.5120/ijca2020919928 [in English].
6. Sikora, O. V., & Kobylnyk, T. P. (2023). Java yak zasib navchannya uchniv osnov prohramuvannya [Java as a means of teaching students the basics of programming]. *Vcheni zapysky TNU imeni V. I. Vernadskoho. Seriya: Tekhnichni nauky – Academic notes of TNU named after V. I. Vernadskyi. Series: Technical sciences*, 34(73), 224–230. https://doi.org/10.32782/2663-5941/2023.5/35 [in Ukrainian].
7. Yeromina, N., & Shaposhnyk, M. (2023). Metody testuvannya mobilnykh zastosunkiv [Methods of testing mobile applications]. *Scientific Collection «InterConf»*, 178, 322–325. Retrieved from https://archive.interconf.center/index.php/conference-proceeding/article/view/4715 [in Ukrainian].
8. Khambir, V. (2024). Avtomatyzatsiya protsesiv testuvannya mobilnykh zastosunkiv [Automation of mobile application testing processes]. *Computer-integrated technologies: education, science, production*, 55, 213–224. https://doi.org/10.36910/6775-2524-0560-2024-55-27 [in Ukrainian].
9. Ichanska, N., & Ulko, S. (2020). Osnovni aspekty stvorennya mobilnykh dodatkiv ta vybir instrumentiv yikh rozrobky [The main aspects of creating mobile applications and the choice of tools for their development]. *Systemy upravlinnya, navihatsiyi ta zvyazku. Zbirnyk naukovykh prats – Control, navigation and communication systems. Collection of scientific works*, 1(59), 74–78. https://doi.org/10.26906/SUNZ.2020.1.074 [in Ukrainian].
10. Hunko, I. (2023). Software testing in 2023: new trends and challenges. *Herald of Kyiv Institute of Business and Technology*, 49(1–2), 25–36. https://doi.org/10.37203/kibit.2023.49.03 [in Ukrainian].
11. Hort, M., Kechagia, M., Sarro, F., & Harman, M. (2021). A survey of performance optimization for mobile applications. *IEEE Transactions on Software Engineering*, 48(8), 2879–2904. Retrieved from https://ieeexplore.ieee.org/abstract/document/9397392 [in English].
12. Farooq, M. S., Riaz, S., Alvi, A., Ali, A., & Rehman, I. U. (2022). Cross-Platform Mobile Development Approaches and Frameworks. *VFAST Transactions on Software Engineering*, 10(2), 79–93. https://doi.org/10.21015/vtse.v10i2.978 [in English].
13. Ogala, J. O., & Ojie, D. V. (2020). Comparative analysis of C, C++, C# and Java programming languages. *GSJ*, 8(5), 1899–1913. Retrieved from https://unidel.edu.ng/cms/uploads/publications/unidel_pub_1679733436.pdf [in English].
14. Klabnik, S., & Nichols, C. (2023). *The Rust programming language*. No Starch Press. Retrieved from https://ejudge.lksh.ru/lang_docs/The%20Rust%20Programming%20Language.pdf [in English].
15. Artigues, V., Kormann, K., Rampp, M., & Reuter, K. (2020). Evaluation of performance portability frameworks for the implementation of a particle-in-cell code. *Concurrency and Computation: Practice and Experience*, 32(11), 1–23. https://doi.org/10.1002/cpe.5640 [in English].
16. GeeksforGeeks. (2024). *7 Best IDEs For C/C++ Developers in 2024*. Retrieved from https://www.geeksforgeeks.org/best-ides-for-c-c-plus-plus-developers/ [in English].
17. MarketSplash. (2023). *C Programming Mobile App Development: Techniques And Tips*. Retrieved from https://marketsplash.com/c-programming-mobile-app-development/ [in English].
18. Mobile Growth Association. (2023). *How to Optimize Your Code in Mobile Applications – 10 Pro Tips*. Retrieved from https://mobilegrowthassociation.com/how-to-optimize-your-code-in-mobile-applications-10-pro-tips/ [in English].
19. Stoyko, T. (2023). How to Make an App both for Android and iOS. *Incora*. Retrieved from https://incora.software/insights/make-app-both-iOS-Android [in English].