2. CIL – Common Intermediate Language. URL: https://www.scriptol.com/programming/cil.php

3. Язык C# и платформа .NET Core. URL: https://metanit.com/sharp/tutorial/1.1.php

4. Common Language Runtime. (CLR) overview. URL: https://docs.microsoft.com/en-us/dotnet/standard/clr

5. ADO.NET. URL: https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/

6. What is Entity Framework? URL: https://www.entityframeworktutorial.net/what-is-entityframework.aspx

7. Документация ASP.NET. URL: https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0

8. Overview of .NET in Unity. URL: https://docs.unity3d.com/Manual/overview-of-dot-net-in-unity.html

9. Что такое Xamarin? URL: https://docs.microsoft.com/ru-ru/xamarin/get-started/what-is-xamarin

Ключові слова: .NET, кросплатформність, C#, веб, CLR.

Ключевые слова: .NET, кроссплатформность, C#, веб, CLR.

Key words: .NET, cross platform, C#, web, CLR.

Науковий керівник: *к.т.н., доц. Трофименко О. Г.*

Люлік Анастасія Юріївна

Національний університет «Одеська юридична академія»,

студентка 1-го курсу факультету кібербезпеки та інформаційних технологій

ЗАСОБИ РОЗРОБКИ МОБІЛЬНИХ ЗАСТОСУНКІВ

У наші технологічні часи більшість людей вже не може уявити своє життя без смартфонів. Нині під час карантину через пандемію COVID-19 люди

щоденно використовують мобільні застосунки для праці, замовлень на дім та відпочинку, а організації прагнуть поширити свій бізнес і в цифрові мережі. За даними статистики 2018 року світовий дохід від мобільних застосунків склав понад 365 млрд доларів США, а за прогнозами у 2023 році мобільні застосунки принесуть понад 935 млрд доларів доходу від платних завантажень і реклами в застосунках [1].

Через це розробка мобільних застосунків є вельми актуальною. Для її реалізації, перш за все, потрібно визначитися з типом створюваного мобільного застосунку, а вже потім з мовою програмування. Розрізняють три різновиди програмних мобільних застосунків: нативні, веб та гібридні. Розглянемо їх.

Нативний тип застосунків (Native App) розробляється під конкретну апаратно-програмну платформу, наприклад, iOS або Android, і пишуться мовами, створеними для цієї платформи. Відповідні платформи мають свої набори засобів розробки (SDK – Software Development Kit) і свій стек технологій, зав'язаний на певну мову програмування. Так, рідними мовами для Android є Java та Kotlin, а для iOS, відповідно – Swift та Objective-C [2]. Специфікою нативних застосунків є те, що вони тонко налаштовані під відповідні операційні системи і можуть отримувати вільний доступ до всіх служб, сервісів та ресурсів телефону: камери, мікрофона, геолокатора, акселерометра, календаря, медіафайлів, повідомлень тощо. Через це перевагами «нативок» є: гарна продуктивність, що забезпечує плавну анімацію і високу швидкість завантаження відповідного застосунку; можливість роботи в автономному режимі (в офлайні); економне витрачання ресурсів телефону (батареї); забезпечення надійного захисту даних, завдяки різним рівням захисту операційної системи і максимально ефективному використанню апаратних ресурсів; покращений користувальницький досвід, завдяки налаштованим на платформу компонентам UI/UX. Попри всі переваги варто враховувати те, що для кожної платформи потрібно розробляти свою версію нативного веб-застосунку. При цьому: розробка «нативок» реалізується швидше (оскільки вона для однієї конкретної платформи), легше запобігти помилкам і технічним проблемам, немає залежності від сторонніх фреймворків і бібліотек, а простіша база коду та екосистема суттєво покращують

продуктивність. З іншого боку, розробка Native App для більш ніж однієї операційної системи, наприклад, для iOS та Android, через те, що рідні мови програмування для них вельми специфічні, потребує найняти розробників з відповідними наборами навичок. У підсумку, для завершення нативного проєкту потрібно більше часу і відповідно більше коштів. Тому розробка нативних застосунків є хорошим вибором, якщо потрібно створити застосунок для однієї цільової платформи, наприклад, ігровий застосунок або програму, яка має «важку» анімацію. Тоді нативна розробка програми допоможе впоратися з цим, завдяки швидшій продуктивності.

Веб-застосунки (Web App) запускаються через браузери, при цьому ззовні та функціонально вони схожі на нативні, хоча і не потребують окремо пам'яті чи сховища на пристрої користувача. При цьому не варто плутати веб-застосунки і веб-сайти. Веб-застосунки, на відміну від інформаційного веб-сайту, надають додаткову функціональність та інтерактивність. Наприклад, мегаінформативний сайт Вікіпедія по суті лише надає інформацію, а веб-застосунок Facebook має інтерактивний характер. Прогресивні веб-застосунки (Progressive Web Applications, PWA) мають можливість надсилати push-повідомлення, працювати в автономному режимі та завантажувати на головному екрані. На відміну від нативних мобільних застосунків, розробка Web App може бути простою та швидкою, але це залежить від вимог. Переважна більшість розробок Web App виконується за допомогою JavaScript, CSS і HTML5 [2]. Веб-застосунки не треба завантажувати через AppStore, Google Play чи то інші магазини і при цьому вони будуть працездатні в усіх операційних систем. Вони легкі у знаходженні через пошукові системи і доступні для використання на різних платформах. З іншого боку Web App не мають доступу до вбудованих можливостей мобільних пристроїв, чим зумовлена їхня доволі обмежена продуктивність. Здебільшого веб-застосунок розробляють як гарний мінімалістичний спосіб перевірити ідею і швидко отримати веб-програму, схожу на мобільний застосунок, перш ніж інвестувати в його нативну реалізацію. PWA не є заміною мобільних застосунків, а скоріше оновленням поточного веб-UX, оскільки вони не охоплюють всіх основ для мобільного виявлення.

Гібридний тип (Hybrid Mobile App) поєднує в собі характеристики обох попередніх різновидів мобільних застосунків. Він працює як вебзастосунок, а завантажується як нативний. Написаний за допомогою веб-технологій Hybrid Mobile App запускає код у власному контейнері (webview – спрощений браузер безпосередньо у програмі), використовує рушій браузера, але не сам браузер для рендерінга HTML та локальної обробки JavaScript чи іншої обраної мови [3]. Усі переваги гібридних мобільних застосунків випливають з того, що замість того, щоб створювати дві нативні програми для iOS та Android, можна створити одну і налаштовувати її так, щоб вона працювала на обох платформах. Тому для розробки гібридного застосунку потрібна лише одна кодова база для керування, а отже і менше розробників, менше часу, менше коштів. При цьому Hybrid Mobile App має доступ до функцій пристрою, як у нативних застосунках, завдяки PhoneGap, що є певним мостом між SDK і webview, в якому працює програма. З іншого боку, мабуть найбільшим недоліком гібридних програм є продуктивність, що пояснюється завантаженням у webview, продуктивність якого поки що значно поступається нативним застосункам [4]. Не можна не сказати і про складність розробки кросплатформних веб-застосунків. Прилаштувати гібридний застосунок для прийнятної продуктивності на кожній платформі зазвичай вимагає значної роботи і кваліфікації розробника. Інколи загальна вартість може стати порівняною з повністю нативними програмами. Інколи розробники можуть надавати увагу лише одній операційній системі, через що на іншій можуть з’являтися помилки та погіршуватися якість. Тому слід виконувати тестування на кількох пристроях, щоб переконатися у справності програми.

Отже, кожен з типів мобільних застосунків має свої переваги та недоліки. Обираючи, який саме розробляти, варто зважати на них. Звичайно, все залежить від ідеї, котру треба реалізувати. Після визначення з типом мобільного веб-застосунку наступний крок – вибір мови програмування.

Для розробки нативних застосунків для iOS на пристроях від Apple частіше перевагу віддають мовам Objective-C та Swift. Мова Swift, на відміну від Objective-C, має спрощений синтаксис, адже вона новіша і створена з

урахуванням позитивного досвіду попередників. Ще Swift виділяється своєю швидкістю, численні тести показали продуктивність, близьку до C++ [5]. Якщо орієнтація створюваного веб-застосунку спрямована на операційну систему Android, популярними мовами є Java та Kotlin. Ці дві мови програмування мають свої особливості. Порівнюючи Java і Kotlin, думки розробників сильно різняться [6], адже Java, хоча і вважається доволі складною, вже перевірена часом і має потужні бібліотеки та інструменти, а «молода» мова Kotlin є зручною і лаконічною, що дозволяє спростити написання великих проектів, і при цьому уникнути численних помилок. Крім того, існує кілька платформ, які дозволяють розробляти кросплатформні нативні мобільні застосунки, наприклад: Xamarin, React Native, PhoneGap, Titanium і Flutter від Google. Так, за допомогою Xamarin були розроблені Slack, Pinterest і Honeywell. Засоби React Native використовувались при розробленні Facebook, Walmart, Tesla та Airbnb. Деякі з найбільш відомих програм, створених за допомогою Titanium, – це eBay, ZipCar, PayPal і Khan Academy [4].

Для розробки веб-застосунків поширено використовують мови програмування JavaScript (JS), PHP та Python. Так, швидка і продуктивна скриптова мова JavaScript підтримує усі сучасні браузери, оскільки вона інтегрується з HTML. JS використовується і для фронтенду, і для бекенду, проте вона не відрізняється стабільністю. Потужна мова для створення інтерактивних і динамічних веб-сторінок PHP, на відміну від JS, інтерпретується на стороні сервера в HTML-код, який передається на сторону клієнта. Це з одного боку погіршує інтерактивність і продуктивність сторінок, але краще з точки зору безпеки. Високорівнева універсальна мова Python стійкіша до навантажень сервера, ніж PHP, має простий синтаксис і потужні бібліотеки. Проте ця мова має низьку швидкість, займає багато пам’яті, що може стати суттєвим недоліком у створенні веб-застосунків. Не існує ідеальних засобів розробки, а обґрунтований вибір мови залежить від специфіки створюваного веб-застосунку [7]. До того ж не слід забувати про шаблони та фреймворки для розробки веб-застосунків, як-от: Angular, React, Vue.js тощо.

При розробленні гібридних застосунків можуть знадобитися усі розглянуті вище мови програмування, позаяк, за правилом, інтерфейсу частину розробляють як нативний тип, а ядро створюють як веб-застосунок. Крім того, існують гібридні платформи розробки PhoneGap, Cordova і Canvas. Найвідоміша серед гібридних платформ PhoneGap є, ймовірно, найпростішою для веб-розробника. Наприклад, за допомогою PhoneGap створені гібридні веб-застосунки Sworkit і Untappd. Cordova підтримується Adobe і дозволяє створювати міжбраузерні мобільні застосунки з JavaScript, HTML і CSS. Canvas дозволяє з наявного мобільної вебпрограми або успішного адаптивного сайту створити мобільний застосунок, оптимізувавши його для мобільних пристроїв, як нативну програму для iOS та Android.

Отже, існують різні напрямки розробки мобільних застосунків і різни засоби їх реалізації. Всі вони мають свої плюси і мінуси. При виборі слід враховувати вимоги і специфіку створюваного застосунку, а також можливі обмеження, наприклад, в часі чи грошах, що підштовхне до прийняття певного рішення.

Список використаних джерел:

1. Worldwide mobile app revenues in 2014 to 2023. URL: https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast/
2. Best Language For Mobile App Development. URL: https://www.webiotic.com/best-language-for-mobile-app-development-what-you-need-to-know/
3. What is a Hybrid Mobile App? URL: https://www.telerik.com/blogs/what-is-a-hybrid-mobile-app-
4. Saccomani P. Native Apps, Web Apps or Hybrid Apps? What’s the Difference? URL: https://www.mobiloud.com/blog/native-web-or-hybrid-apps#6
5. Swift-Speed-Test. URL: https://github.com/futursolo/Swift-Speed-Test
6. Google объявила Kotlin приоритетным языком программирования для разработки Android-приложений. URL: https://vc.ru/dev/66728-google-obyavila-kotlin-prioritetnym-yazykom-programmirovaniya-dlya-razrabotki-android-prilozheniy.

7. Трофименко О.Г. Сучасний інструментарій веб-розробника. *Інформаційне суспільство: проблеми та перспективи* : матер. V всеукр. наук.-практ. конф. (22 травня 2020 р.). URL: http://conf.inf.od.ua/doklady-konferentsii/2020/223-trofimenko.

Ключові слова: нативний застосунок, веб-застосунок, гібридний застосунок, мови програмування.

Ключевые слова: нативное приложение, веб-приложение, гибридное приложение, языки программирования.

Key words: native application, web application, hybrid application, programming languages.

Науковий керівник: *к.т.н., доц. Трофименко О. Г.*

Струк Ніkіта Олександрович

Національний університет «Одеська юридична академія»,

студент 2-го курсу факультету кібербезпеки та інформаційних технологій

МОЖЛИВОСТІ БІБЛІОТЕКИ OPENGL У КОМП'ЮТЕРНІЙ ГРАФІЦІ

OpenGL - одна з програмних технологій обробки двовимірної та тривимірної графіки, яка є конкурентною відносно Direct3D від Microsoft. Ця бібліотека має дуже широкий спектр можливостей та інструментів, до того ж вона безкоштовна, реалізована на усіх сучасних платформах, у тому числі мобільних [1].

Специфіка OpenGl організована досить чітко, тобто за допомогою імені функції тут же можна зрозуміти, яка з функцій є універсальною або "місцевою" для якого-небудь набору відеокарт, чи скільки в ній є параметрів [1].