КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття ступеня бакалавра
за спеціальністю 121 Програмна інженерія
на тему:

ПРОГРАМНИЙ ЗАСІБ ДЛЯ ВСТАНОВЛЕННЯ ІГРОВИХ ДОДАТКІВ HTML5 В ОФЛАЙН-РЕЖИМІ НА ANDROID

Виконав студент 4-го курсу
Юрій ГРИЩЕНКО _____
(підпис)

Науковий керівник:
кандидат фізико-математичних наук, асистент
Костянтин ЖЕРЕБ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«____» _____ 202_ р.,
протокол № ___
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

КИЇВ-2022

РЕФЕРАТ

Обсяг роботи 41сторінка, 14 ілюстрацій, 25 джерел посилань.

ВЕБ-ДОДАТОК, ВСТАНОВЛЕННЯ ДОДАТКІВ, ДИСТРИБУЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ДОДАТОК ANDROID, ОФЛАЙН-РЕЖИМ

Об’єктом роботи є додатки, розроблені з використанням веб-технологій, здатні до роботи без доступу до мережі Інтернет. Предметом роботи є програмний засіб для завантаження додатків, призначених для роботи в режимі онлайн, та їх пристосування до роботи в режимі офлайн на операційній системі Android.

Метою роботи є створення програмного засобу для завантаження веб-додатків для пристроїв на основі операційної системи Android, зокрема комп’ютерних ігор, для їх подальшого використання в режимі офлайн.

Методи розроблення: розробка програмного продукту на основі еволюційної моделі та відгуків користувачів. Інструменти розроблення: безкоштовне, вільно поширюване інтегроване середовище розробки Android Studio, мова програмування Kotlin, що працює на JVM.

Результати роботи: виконано загальний огляд існуючих підходів до встановлення веб-додатків у режимі офлайн, та відомих джерел, що їх розповсюджують, сформульовано вимоги для програмного засобу для їх встановлення, спроектовано архітектуру такого програмного засобу, розроблено та протестовано додаток.

Програмний продукт Mitch може застосовуватися в повсякденному житті в цілях розваги, як альтернатива Google Play.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

API — Application Programming Interface;

APK — Android Package;

CDN — Content Delivery Network

CSS — Cascading Style Sheets;

FAB — Floating Action Button;

HTML5 — HyperText Markup Language 5;

HTTPS — HyperText Transfer Protocol Secure;

IDE — Integrated Development Environment;

JVM — Java Virtual Machine

PWA — Progressive Web Application;

URL — Uniform Resource Locator;

W3C — World Wide Web Consortium;

ОС — операційна система;

ПК — персональний комп’ютер

ВСТУП

Оцінка сучасного стану об’єкта розробки. Смартфони є найбільш популярним форм-фактором персональних комп’ютерів, приблизно 6 мільярдів людей мають смартфон станом на 2022 рік[1], тобто більше 80% населення світу. За даними тих же дослідників[2], кількість людей, підключених до мережі Інтернет, становить лише 60% населення.

Окрім кількості підключень до мережі, слід врахувати якість зв’язку. В багатьох випадках, здебільшого у сільській місцевості, країнах, що розвиваються, або, у випадку розвинених країн, у метро чи підвалах, зв’язок до мережі Інтернет може не завжди бути присутнім, або ж мати значні обмеження швидкості.[3]

Ці фактори ставлять під сумнів доречність залежності мобільного програмного забезпечення від постійного доступу до мережі.[4]

З іншого боку, веб-технології, такі як HTML5, CSS, JavaScript, WebAssembly, мають неабиякі переваги для розробників програм, націлених на глобальний ринок, зокрема мультиплатформність та адаптивність. Це власне і дозволяє користувачам з усього світу запускати ці програми на різноманітних пристроях та операційних системах. У випадку власне веб-додатків, тобто інтерактивного програмного забезпечення, що працює як веб-сайт, також отримуємо перевагу в простоті використання, оскільки немає необхідності виконання окремого попереднього кроку встановлення.

Багато уваги приділяється процесам розробки веб-застосунків, які мали б вище зазначені переваги, при цьому позбуваючись недоліків, а власне — залежності від постійного доступу до мережі Інтернет. Провідною в цій сфері є компанія Google, яка:

- по-перше, розробила найбільш поширену операційну систему для смартфонів Android, і надає розробникам програмного забезпечення на її основі рекомендації “building for billions” зокрема для роботи з обмеженим доступом до мережі[4]

- по-друге, Google активно розробляє веб-стандарти, на основі яких будуються так звані поступові веб-застосунки (Progressive Web Applications, PWA), та імплементує ці стандарти в своїх браузерах.[5]

Актуальність роботи та підстави для її виконання. Незважаючи на наявність принципів розробки веб-застосунків, придатних до роботи в режимі офлайн, та платформ для їх розповсюдження, багато розробників досі не приділяють цьому великої уваги.

Для цього є декілька причин[5], зокрема:

- можлива недоцільність офлайн-режиму для веб-застосунків, оскільки їх основний функціонал напряму залежить від підключення до мережі (написання та читання електронних листів, перегляд відео-контенту тощо)
- потреба в докладанні певних додаткових зусиль для пристосування веб-застосунку до роботи в режимі офлайн

Ігрові застосунки здебільшого уникають першого недоліку, оскільки навіть ті ігри, що створюються за допомогою веб-технологій, часто навіть не мають можливості грати в режимі онлайн, або ж принаймні підтримують другорядний офлайн-режим.

Проте залишається друга проблема з вище зазначених, а саме необхідність докласти певних зусиль, аби гравці мали доступ до застосунку, створеного на основі HTML5 і т.д., не використовуючи мережі Інтернет.

Отже, доцільним є створення програмного засобу, який би дозволив встановлювати такі веб-додатки для роботи в режимі офлайн без зайвих зусиль від їх розробників.

Мета і завдання роботи. Метою роботи є створення програмного засобу для завантаження веб-додатків для пристроїв на основі операційної системи Android, зокрема комп’ютерних ігор, для їх подальшого використання в режимі офлайн. Для досягнення цієї мети поставлено такі завдання:

- Дослідити сучасні підходи до встановлення веб-додатків у режимі офлайн
- Дослідити відомі джерела, що їх розповсюджують

- Сформулювати вимоги для програмного засобу для їх встановлення та спроектувати його архітектуру
- Розробити та протестувати цей додаток

Об’єкт, методи й засоби розроблення. Об’єктом розроблення є додатки, розроблені з використанням веб-технологій, здатні до роботи без доступу до мережі Інтернет.

Розробці програмного засобу, здатного встановлювати такі додатки, передував аналіз роботи обраного джерела додатків за допомогою певних інструментів, таких як Інструментів розробника в Firefox, та командної утиліти Curl.

В якості інструменту створення власне програмного засобу було обрано Android Studio — інтегроване середовище розробки (IDE) мовами програмування Java та Kotlin, яке є безкоштовним, вільно поширюваним, і рекомендованим власне розробниками операційної системи Android.[6]

Було обрано саме мову програмування Kotlin, оскільки вона компілюється у Java bytecode і може виконуватися на JVM, але при цьому надає додаткові можливості, що пришвидшують написання коду, роблять його коротшим та простішим у розумінні. Google, розробники Android та Android Studio, також рекомендують використовувати Kotlin замість Java для розробки нових додатків. [7]

Можливі сфери застосування. Кінцевий продукт може використовуватись за призначенням, як додаток для завантаження та оновлення програмного забезпечення, і для більш загальної взаємодії з порталом itch.io. Продукт надає доступ до додатків у сфері розваг, здебільшого мобільних та комп’ютерних ігор.

1 РОЗРОБКА ТА ПУБЛІКАЦІЯ ВЕБ-ДОДАТКІВ

1.1 Поняття веб-застосунку

Існує декілька розумінь поняття веб-застосунку: значення цього терміну змінювалося з часом, під впливом стрімкого росту поширення мережі Інтернет, і спричиненого цим розвитку технологій, на яких будуються такі застосунки.[5][8]

Веб-застосунок у класичному розумінні — це застосунок (тобто користувацька комп'ютерна програма, що дає змогу вирішувати конкретні прикладні задачі користувача) з розподіленою архітектурою, в якій клієнтом виступає браузер, а сервером — вебсервер.[8]

Браузер в цьому випадку розглядається як основа для побудови так званих тонких клієнтів — тобто основна логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у відображенні інформації, завантаженої мережею з сервера, а також у зворотній передачі даних, тобто відправленні до сервера даних, введених користувачем.

Основною перевагою такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача. Усі поширені браузери здебільшого імплементують ті самі стандарти (з деякими винятками, розраховуючи на стрімкий розвиток та поповнення веб-стандартів — браузери далеко не завжди імплементують найновіші стандарти, або ж навпаки, в якості експерименту додають нові можливості, які ще не є остаточно стандартизованими). При цьому всі поширені користувацькі операційні системи мають або вбудований веб-браузер, або ж асортимент доступних браузерів, створених сторонніми розробниками. Тобто є можливість користуватися веб-застосунками не тільки на настільних ПК (на основі ОС Windows, macOS, Linux та ін.) та мобільних пристроях (Android, iOS), а останнім часом також на інших видах пристроїв, таких як автомобілі, Smart TV (інколи на основі більш специфічних операційних систем, наприклад Firefox OS чи MeeGo).

Властивість програмного забезпечення працювати більш ніж на одній програмній та апаратній платформі (в тому числі — на різних операційних системах) називають багатоплатформністю, а також кросплатформністю або мультиплатформністю.[9] Інколи так називають програми, чий код розрахований на кілька платформ, проте потребує окремо компіляції для кожної платформи. Такі програми мають перевагу в швидкодії та інтеграції з середовищем виконання, проте вимога до окремої компіляції вважається недоліком через збільшення витрат часу на компіляцію та підвищення важливості тестування на кожній окремій платформі.

Також розробка багатоплатформних застосунків ускладнюється тим, що під поняттям платформи часто розуміють саме комбінації програмних та апаратних платформ, тобто ОС Microsoft Windows на архітектурі процесора x86 часто виділяється як окрема платформа від Windows на ARM.

Альтернативним варіантом є розробка застосунків на основі інтерпретованих мов програмування або ж компіляція у портативний байт-код (bytecode), де інтерпретатори та пакунки стандартних бібліотек є аналогічними на всіх підтримуваних платформах.

У цьому випадку, набір інтерпретаторів для однієї і тієї ж мови програмування або байткоду можна вважати окремою апаратною платформою (на що натякає, наприклад, назва Java Virtual Machine), а власне мова та її стандартні бібліотеки відіграють роль програмної платформи.

Це приводить нас до більш сучасного розуміння поняття веб-застосунку, а саме: застосунок, розроблений для веб-браузера в якості платформи.[5]

Таке тлумачення терміну значно змістовно відрізняється від попереднього, оскільки ми відштовхуємося від строгого розподілу на клієнтську та серверну частину.

1995 року Netscape розробило мову програмування скриптів, що використовуються в браузері користувача (англ. client-side scripting). 2005 року було введено поняття Ajax[10], тобто набір технологій та методик програмування

для створення асинхронних веб-застосунків. Дані відсилаються до і від сервера у фоновому режимі, не впливаючи на відображення поточної сторінки. Таким чином від'єднується шар обміну даних від шару презентації, що дозволяє динамічно завантажувати контент не завантажуючи нову сторінку.

Одними із перших повноцінних веб-застосунків у сучасному розумінні цього слова стали Gmail (2004) та Google Maps (2005).[10] З того часу подібне програмне забезпечення ставало ще більш потужним у порівнянні з традиційними нативними застосунками.

Про зріст можливостей таких застосунків та очікувань користувачів до працездатності свідчить створення технології WebAssembly (2015), що є більш низькорівневим проміжним кодом для виконання в браузері програм, скомпільованих з різних мов. Це дозволяє істотно зменшити розмір та збільшити працездатність застосунків у порівнянні з JavaScript.

WebAssembly разом із WebGL (версія 1.0 випущена 2011 року) надає платформі веб достатню працездатність для створення досить складних та реалістичних багатоплатформних комп'ютерних ігор.

1.2. Порівняння сучасних методів розробки та встановлення веб-додатків в офлайн-режимі.

Як ми бачимо, в останні роки веб-технології стають все більш потужними, що призводить до зближення функціоналу побудованих на їх з основі застосунків із класичним (нативним) програмним забезпеченням.

Дійсно, як зазначалося в минулому підрозділі, у сучасному розумінні поняття "веб-застосунок" ключовою характеристикою є їх націлення на конкретну платформу, а саме браузер. Проте браузери самі по собі є застосунками з точки зору користувача, хоча й дуже потужними (сирцевий код найпопулярнішого браузеру Google Chrome містить близько 10 мільйонів рядків коду, приблизно стільки ж має ядро операційної системи Linux).[5]

В деяких ситуаціях було би бажано встановити веб-додатки на одному рівні з іншим користувацьким програмним забезпеченням, наявним у системі.

По-перше, це надало би можливість використовувати такі застосунки в режимі офлайн.

По-друге, веб-браузери виконують свій код у середовищі, певним чином ізольованому від системи користувача (так званий sandboxing або принцип пісочниці): це надає певні гарантії безпеки, наприклад, веб-сайт не може читати та модифікувати файли користувача без явного дозволу на вивантаження чи завантаження — проте зрозуміло, що це накладає певні обмеження на можливі застосування веб-додатків.

По-третє, розробляючи застосунки не в контексті браузера, можна використати веб-технології лише вибірковим чином, наприклад, використати HTML та CSS для презентації сторінок, але обчислення виконувати на скомпільованій мові.

В цій роботі розглянемо сучасні принципи розробки та встановлення веб-застосунків у контекстах, відмінних від стандартних веб-сайтів у браузерах, проте зосередимо увагу саме на офлайн-режимі.

Зазвичай, коли кажуть про встановлення веб-додатків для використання без доступу в Інтернет, мова йде про так звані поступові веб-застосунки (англ. progressive web applications або скорочено PWA). Їх розглядають як певний гібрид звичайної веб-сторінки та мобільного застосунку. Таке програмне забезпечення створюється за допомогою можливостей, наданих сучасними браузерами, (далеко не обов’язково в режимі офлайн), проте їх використання нагадує використання мобільного застосунку.

Поняття PWA не є чітко визначеним, проте виділяють основні критерії[5]:

- поступовість (англ. progressive): можливість використання на базовому рівні у старих браузерах, з набуттям повного функціоналу в останніх версіях
- можуть бути встановленими (англ. installable): дозволяють користувачам додавати на свій робочий стіл додатки, які вважають корисними

- незалежні від з’єднання: працюють в офлайн або в мережах із низькою швидкістю
- адаптивні (англ. responsive): можливе використання на будь-якому пристрої з екраном та сучасним браузером
- безпечні: дані передаються через HTTPS для запобігання перехопленню і гарантії, що контент не підроблено
- на них можна посилатися (англ. linkable): можна “поділитися” застосунком, надіславши його URL
- виявні (англ. discoverable): ідентифікуються як “додатки” завдяки маніфестам W3C, що дозволяє пошуковим рушіям їх знайти
- можуть бути задіяні повторно (англ. re-engageable): можливість сповіщати про наявність нового контенту за допомогою push-повідомлень

Бачимо, що список вимог досить значний. Деякі з них є тривіальними для сучасних веб-застосунків, зокрема можливість посилання та безпека, проте виконання інших вимог потребує певних зусиль, використання специфічних технологій та їх підтримка веб-браузерами.

Зазвичай поступові веб-застосунки використовують поняття “оболонки додатку” (англ. app shell)[11]: спочатку завантажується та зберігається в кеш-пам’ять певний мінімальний користувацький інтерфейс, і лише потім він “наповнюється” контентом. При цьому можна або завантажити останню версію контенту з Інтернету, або, при відсутні зв’язку, показати щось, що раніше збереглося локально.

Кешування контенту відбувається за допомогою Service Worker API, це окремі потоки, що виконують JavaScript, не маючи прямого доступу до документу, а лише перехоплюючи та оброблюючи веб-запити, таким чином виступаючи у ролі проксі-сервера.[11] Наявний інтерфейс CacheStorage дозволяє цим потокам певним чином зберігати дані для подальшого їх надання в режимі офлайн.

Розробники цінують поняття оболонки додатку, оскільки така методика надає перевагу в швидкодії застосунку у випадку повторного використання, тобто коли базовий інтерфейс користувача вже завантажений у сторінку – на схожому принципі засновані так звані односторінкові застосунки, що є більш вужчим поняттям до поступових веб-застосунків. Проте не так багато розробників зацікавлені у створенні повноцінного PWA[12], оскільки використання Service Worker API потребує додаткових зусиль, не пов'язаних з інтерфейсом, а надана можливість працювати в режимі офлайн не завжди є доцільною. Про це свідчить відмова компанії Mozilla від підтримки технології PWA у версіях браузера Firefox для настільних ПК.[12]

Google надає можливість публікувати PWA через свій сервіс Google Play, проте це потребує використання командного інтерфейсу під назвою Bubblewrap, який перетворює PWA у формат Android-застосунків APK. Ці застосунки виконуються всередині так званого Trusted Web Activity, що є ще однією sandbox-прослойкою, яка ускладнює взаємодію веб-додатку з нативники функціями Android.[13]

Альтернативним варіантом для розробників багатоплатформних офлайн-застосунків на основі веб-технологій є фреймворки Electron та React Native.

Electron (раніше відомий як Atom Shell) – фреймворк, розроблений GitHub, націлений на настільні операційні системи. Дозволяє розробку застосунків зі звичним для веб-розробників поділом на бекенд та фронтенд, при тому бекенд виконується локально на основі Node.js, а роль фронтенда відіграє браузерний рушій Chromium.[14]

Використання Node.js та додаткових API (зокрема для взаємодії між процесами) дозволяє таким застосункам повну свободу дій, таку як і для звичайних додатків для підтримуваних операційних систем. Це робить більш доцільним використання таких програм в режимі офлайн, наприклад, текстових редакторів Atom та Visual Studio Code.

Додатки, побудовані на основі Electron, є повністю незалежними від браузерів, оскільки окремий рушій Chromium є вбудованим у їх пакунки та інсталятори. Це призводить до таких недоліків, як великий розмір та повільність інсталятора, а також до надмірного використання оперативної пам’яті.[15]

React Native, розроблений компанією Meta (раніше – Facebook), певною мірою усуває недостатню швидкодію додатків Electron за рахунок використання нативних елементів інтерфейсу, а також додатково підтримує мобільні операційні системи Android та iOS.[16] Проте хоча він і схожий на розповсюджений веб-фреймворк React, що спрощує перетворення веб-застосунків на офлайн-спроможні, React Native не використовує CSS, і взагалі є ще більш специфічною технологією у порівнянні з Electron та стандартними технологіями веб.

Отже, можна зробити висновок, що розробники мають досить широкий вибір методів для розробки офлайн-застосунків із використанням веб-технологій, які різняться схожістю до стандартної розробки веб-сайтів. Проблема постає в тому, що всі розглянуті методи потребують планування та зусиль з боку розробника. У цій роботі ми прагнемо створити метод встановлення певного роду веб-застосунків без участі їх розробників.

1.3. Дослідження джерел, що розповсюджують ігрові веб-додатки

У цій роботі ми зосередимо увагу на ігрові веб-додатки та їх цифрову дистрибуцію. Таке програмне забезпечення має досить зручні для нас характеристики, а саме:

- відносно статична архітектура: невеликі ігри на основі HTML5 часто завантажують весь свій контент під час запуску, аби не переривати ігровий процес передаванням даних — інші види онлайн-застосунків мають пріоритет заощадження трафіку, тому вони завантажують дані динамічно, за потреби, що ускладнює їх збереження для режиму офлайн

- як ми покажемо під час тестування розробленого нами програмного засобу, багато ігрових веб-додатків насправді ніяк не залежать від зв’язку з інтернетом, тобто в них можна грати в офлайн-режимі

Постає питання: де саме знайти ігрові веб-застосунки, до того ж такі, що розраховані на мобільні пристрої. Взагалі такі додатки можна розділити на два типи за методом їх розповсюдження[17]:

- ігри, що публікуються на так званих “порталах”, які слугують джерелом для їх розповсюдження серед користувачів, наприклад, itch.io та Newgrounds[18]
- самостійні веб-сайти або окремі незалежні веб-сторінки, наприклад, Geoguessr, Agar.io, Wordle

Ігри, що публікуються як самостійні веб-сайти, часто також структурно нагадують більш традиційні веб-додатки: наприклад, Geoguessr (гра на основі Google Maps API, ціль якої — вгадати своє місцезнаходження, роздивляючись фото заданої локації) використовує окремі веб-сторінки для різних режимів гри, головного меню і т.д. Функціонал гри тісно залежить від зв’язку з Інтернетом. Ці фактори унеможливлюють запуск такого застосунку в режимі офлайн.

З іншого боку, ігри, що публікуються на “порталах”, таких як itch.io, вбудовуються в HTML-елемент iframe, що наводить певні обмеження: зокрема, більшість таких ігор містять лише одну HTML-сторінку, а функціонал надається суто JavaScript або WebAssembly на стороні клієнта. Ці самі скрипти відповідають за завантаження усього необхідного контенту, такого як звуки, зображення, музика і т.д., що виконуються відразу після запуску гри.

Також важливим фактором є відносна стандартизація веб-сторінок, на яких розміщуються опубліковані ігри. Для програмного засобу, який би встановлював такі додатки, важливо мати певні метадані, такі як назва гри, логотип, автор та інші. В загальному випадку, автоматично отримати такі дані для будь-якої веб-сторінки неможливо, а сайти itch.io та Newgrounds надають таку інформацію для кожної опублікованої гри, і її можна здобути за допомогою парсингу HTML.

Отже, доцільно обрати одне таке джерело, що розповсюджує ігри, та саме для них розробити програмний засіб для встановлення і використання в офлайн-режимі. Порівняємо деякі популярні сервіси[18]:

- itch.io — сайт, відкритий з 2013 року, націлений на незалежних розробників та публікації невеликих ігор. Також дозволяє розміщувати інший онлайн-контент, такий як музику, електронні журнали, тощо: станом на червень 2022 року має асортимент з майже 600,000 товарів, серед них 560,000 ігор. Видавці можуть вимагати оплату за ігри та розповсюджувати різні види файлів, наприклад, .apk та .exe, проте сайт також підтримує HTML5-ігри та містить категорії веб-ігор, оптимізованих для смартфонів та пристроїв із сенсорними екранами, що містить більше 6000 ігор.
- Newgrounds — розважальний веб-сайт та компанія, заснована у 1995 році. Має чотири категорії контенту: ігри, фільми, аудіо та образотворче мистецтво. Кількість ігор приблизно така ж, проте сайт не має окремої категорії ігор для смартфонів, до того ж переважна більшість контенту використовує застарілу технологію Adobe Flash, тому важко оцінити доцільність створення програмного засобу.
- Також існують інші сайти, аналогічні Newgrounds, проте менщ популярні: наприклад, Armor Games розміщує близько 1500 ігор.

У цій роботі створимо програмний засіб для встановлення в офлайн-режим HTML5-ігор саме з itch.io, оскільки це єдиний з розглянутих сайтів, що має окрему категорію для подібних додатків, оптимізованих для смартфонів. Також перевагою є можливість розміщувати інші види файлів, зокрема .apk — формат нативних додатків для системи Android, проте ми зосередимо увагу на веб-іграх.

2 ДОСЛІДЖЕННЯ РОБОТИ СЕРВІСУ ITCH.IO

2.1 Початок дослідження

Перше, що було зроблено перед початком розробки програмного засобу Mitch — це отримання дозволу від власників сервісу itch.io. Оскільки ним володіє невелика компанія, отримання дозволу виявилося нескладним.[19] Це варто було зробити з двох причин:

По-перше, було невідомо, чи є у команди itch.io плани на створення власного офіційного Android-додатку; в такому разі проект Mitch виявився би набагато менш корисним.

По-друге, можливо, створення неофіційних фронт-ендів певних чином порушує Умови використання itch.io чи певні інші правила.

На щастя, виявилося, що розробники itch.io поки не планують створювати власний додаток, і вони з радістю дозволять іншим людям зробити неофіційний фронтенд, за умови, що код буде відкритим.

Також один із розробників звернув увагу на те, що вже існує клієнт для персональних комп’ютерів з вільним кодом, і що варто розглянути його код, аби зрозуміти, як саме працювати з їх API.[19]

Дійсно, найкращим варіантом було б розглянути код, що вже існує, для авторизації, завантаження, встановлення, оновлення файлів з itch.io, тощо. В такому разі ми маємо два варіанти: можна або напряму запускати існуючий код всередині нашого додатку, або ж можна переписати код під систему Android.

Перший варіант виявився досить складним, оскільки наявний код, яким користується офіційний клієнт для Windows, macOS та Linux, написаний на мові Go.[20] Єдиними офіційно підтриманими мовами для операційної системи Android є Java та Kotlin.

Існує проект Go Mobile, який дозволяє запускати код Go на мобільних пристроях, проте у нього є недоліки:[21]

- Проект є експериментальним.

- Він передбачає або створення додатку повністю на Go, або ж створення додатку на Java/Kotlin з викликом деяких функції Go. Створення додатку повністю на Go значно ускладнює розробку користувацького інтерфейсу.
- Код, розроблений itch.io, розраховний на встановлення стандартних додатків, тобто запуск файлів .exe на Windows, розпакування архівів .tar.gz у відповідні теки Linux, і так далі. Цей код не підтримує ні HTML5, ні взагалі використання на системі Android, яка значно обмежує доступ до файлової системи пристрою.

Другий варіант — власноруч переписати код на мову Kotlin — також є досить складним. Кількість строк коду в їх проекті Go перевищує 10,000, до того ж не завжди є документація та необхідні можливості (зокрема підтримка HTML5 та Android).

Насправді існує і третій варіант, який і буде застосовано в цій роботі: замість того, щоб брати за основу клієнт itch.io, написаний на Go, можна дослідити роботу їх веб-інтерфейсу. Цей підхід має свої переваги:

- Досліджувати роботу додатків, основаних на Web-технологіях, дуже просто, завдяки таким інструментам, як Firefox Developer Tools, або Curl.
- Веб-інтерфейс можна вбудувати в Android-додаток за допомогою елементів WebView, що спростить розробку, оскільки не доведеться самому писати більшу частину користувацького інтерфейсу.

Такий підхід має також недоліки, проте їх розглянемо пізніше, а зараз розглянемо конкретні приклади дослідження поведінки веб-інтерфейсу під час запуску та завантаження гри.

2.2 Приклади дослідження роботи веб-додатку itch.io

На етапі дослідження будемо використовувати версію сайту для настільних комп’ютерів, оскільки це робить зручнішим використання інструментів розробника, які вбудовані в браузери для таких систем (в нашому випадку Mozilla Firefox для ОС Linux)

2.2.1 Дослідження поведінки при запуску онлайн-ігор

В якості прикладу використаємо гру Celeste Classic — платформер, випущений 2016 року, попередник більш знаменитої гри Celeste. Версія Classic доступна на сайті лише як HTML5-гра (рисунок 1).

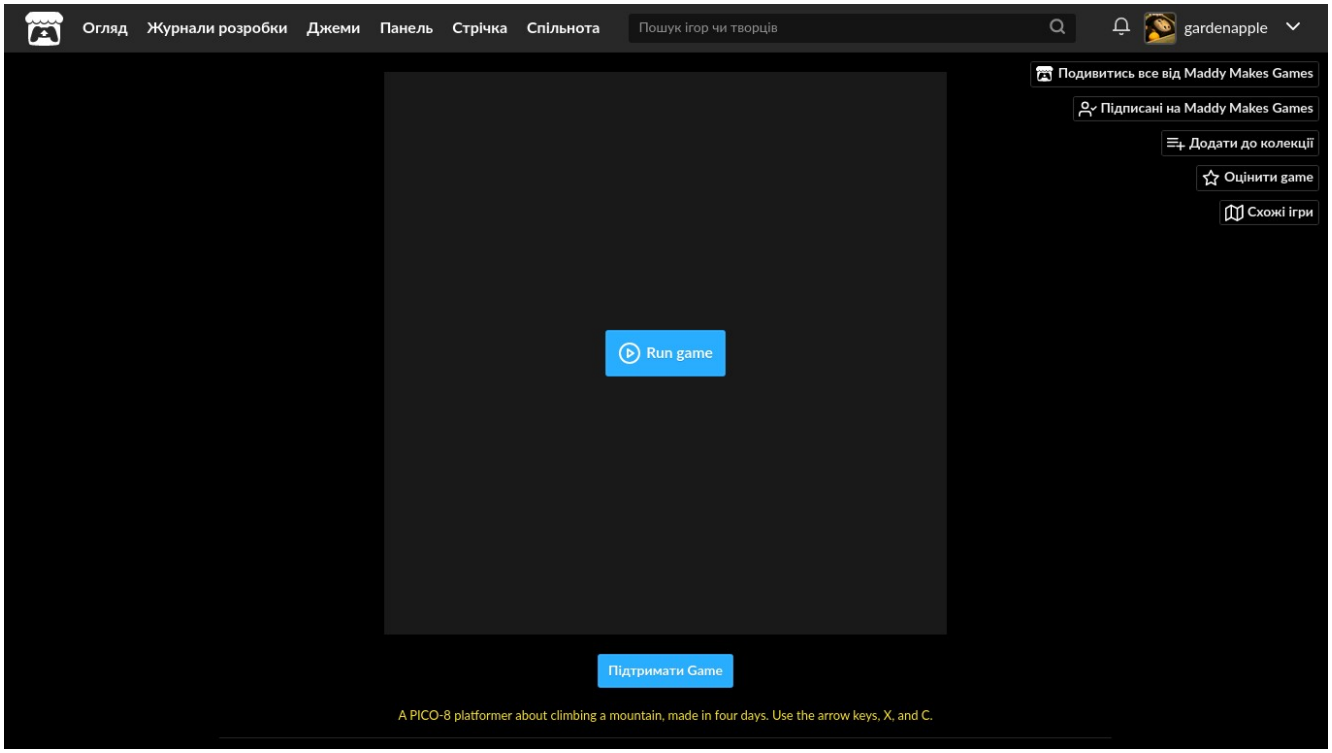


Рисунок 1 — Веб-сторінка з HTML5-грою Celeste Classic

Бачимо, що сама гра займає лише частину сторінки, окрім неї присутні стандартні елементи користувацького інтерфейсу сайту (зверху), а також опис гри та кнопка “підтримати гру” (знизу), що дозволяє переказати гроші на рахунок розробника. Пропонується запуск гри, це відбувається не відразу, в цілях можливого заощадження трафіку у випадку коли опис не привабив користувача.

Подивимося, яку інформацію можна отримати з HTML-коду цієї сторінки. Напевно, цей код згенеровано певним фреймворком, оскільки його важко читати через відсутність розбиття на строки. На щастя, інструменти розробника в браузері надають деревоподібне представлення коду, а також можливість дослідити певні елементи, клацнувши на них.

На рис. 2 побачимо код для рамки з кнопкою “Run game”.

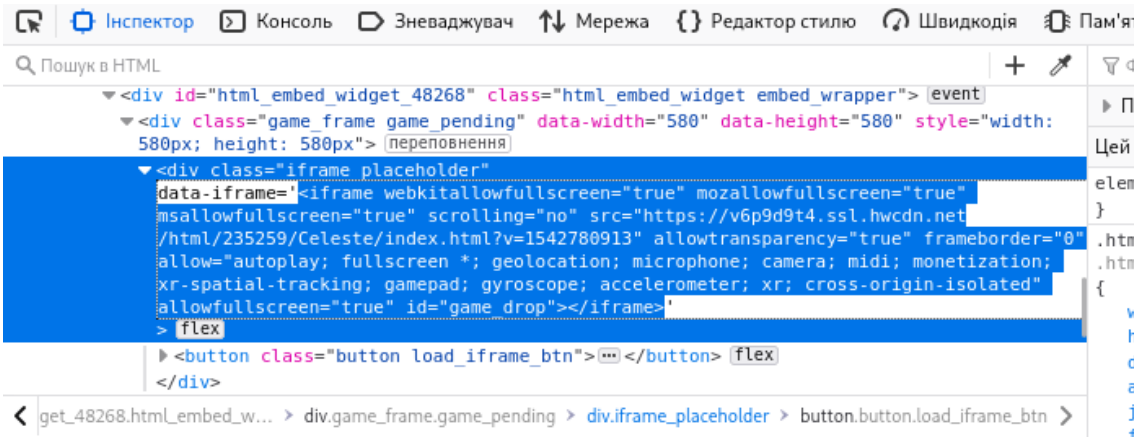


Рисунок 2 — HTML-код елементів, що нас цікавлять

Бачимо деякі нестандартні атрибути HTML5, які починаються зі слова “data-”, таким чином можна зберігати довільні дані в коді документу, які будуть оброблюватися JavaScript на стороні клієнта.

Нас не цікавить код скриптів (все одно його важко розібрати, бо він мінімізований), проте його дії досить очевидні. Атрибути data-width та data-height вказують на розмір рамки гри в пікселях, який ідентичний розміру рамки: 580 на 580 пікселів. Елемент класу “iframe_placeholder” містить атрибут data-iframe, який в свою чергу містить HTML-код для рамки гри, в текстовому вигляді, аби потім вставити його у сам документ.

Дійсно, після натискання кнопки “Run game” бачимо гру, а інструменти розробника засвідчують про очікувані зміни коду сторінки (рисунок 3).

Рисунок 3 — Запущена гра та відповідний елемент HTML iframe

Тепер найцікавішою частиною коду сторінки є атрибут src для елементу рамки гри, він власне і вказує на те, звідки беруться дані гри (на рисунку бачимо підкреслену URL-адресу).

Перейшовши за цією адресою, дійсно отримуємо HTML-код самої гри, а разом із ним і посилання на всі її скрипти. У випадку Celeste Classic усі посилання вказують на те саме доменне ім’я, отже весь контент гри зберігається та видається з однієї локації.

Було б дуже зручно мати можливість перейти до URL адреси на рівень вище, і отримати доступ до теки, що містить перелік усіх файлів гри, проте на жаль, нам такий доступ не надається, на запит відповідають кодом 403 Forbidden.

На основній сторінці гри itch.io також бачимо інші атрибути в елементах meta, такі як назву гри, короткий опис гри, певний ідентифікаційний номер, посилання на зображення-ескіз гри тощо. Деякі з цих даних можуть використовуватися офіційним додатком itch.io для настільних ПК, інші ж імплементують протокол OpenGraph, що дозволяє соцмережам та месенджерам показати, наприклад, заголовок статті з описом, коли на неї посилається користувач.

2.2.2 Додатковий приклад: завантаження файлів платних ігор

Як зазначалося в розділі 1, сайт itch.io дозволяє розміщувати не лише HTML5-ігри, а й інші види файлів, при чому з можливістю вимагати за них оплату. Завантаження та встановлення таких файлів не є предметом дослідження нашої кваліфікаційної роботи, проте цікаво навести приклад використання інших інструментів, зокрема curl, для дослідження складніших процесів, коли дані не вбудовані в HTML, а обмінюються в інших форматах.

Розглянемо гру Super Hexagon, що коштує 3 долари США. Після того, як користувач купив гру, можна зайти на основну сторінку гри (в цьому випадку https://terrycavanagh.itch.io/super-hexagon). На сторінці бачимо кнопку, що дозволяє завантажити гру (рис. 4).

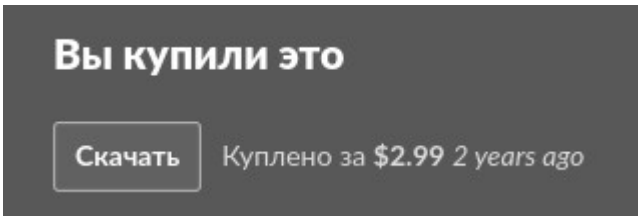


Рисунок 4 — Елемент веб-сторінки з посиланням для завантаження купленої гри

Це посилання має форму https://terrycavanagh.itch.io/super-hexagon/download/ [ідентифікаційний код транзакції]. Перейшовши за посиланням, бачимо сторінку, зображену на рисунку 5.

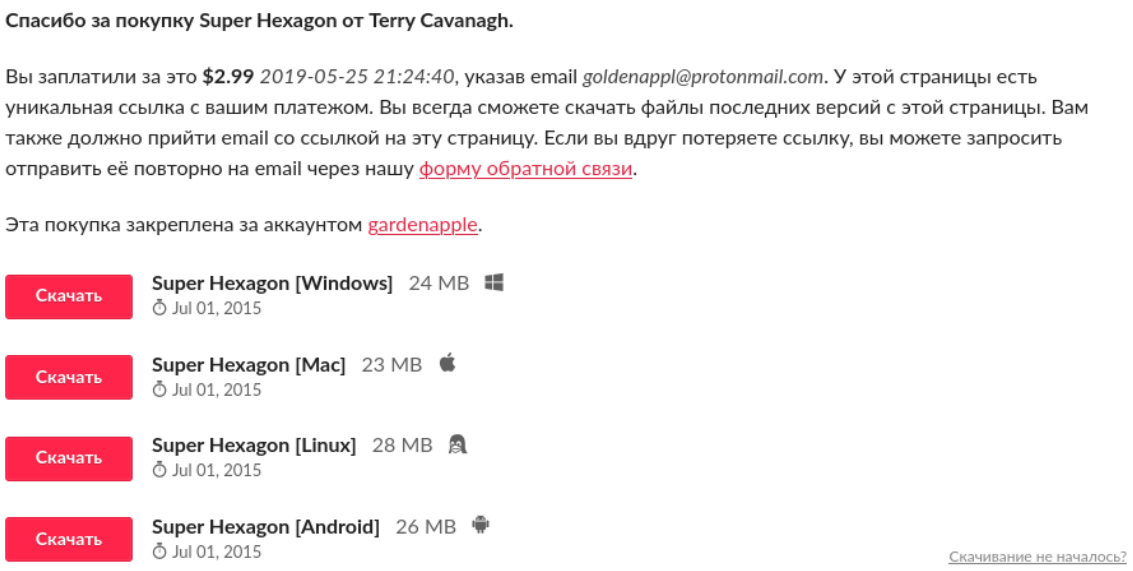


Рисунок 5 — Власне сторінка, що дозволяє завантажити гру

Тепер бачимо чотири посилання, що дозволяють завантажити конкретну версію гри, в цьому випадку це посилання на версії ігор для Windows, macOS, Linux та Android.

Поруч із кожною кнопкою наявна деяка інформація, зокрема дата видання, розмір файлу, та платформи, для яких призначений конкретний файл. Всі ці елементи мають власні HTML-класи та атрибути, що дають можливість автоматично обробити цю інформацію.

Розглянемо кнопки з написом “Скачать”: на цей раз вони не вказують на конкретну адресу, а виконують певний код, написаний на JavaScript. Бачимо POST запит (на рис. 6) без HTTP тіла, де адреса запиту має формат https://terrycavanagh.itch.io/super-hexagon/file/[ідентифікаційний код файлу]?key=[ідентифікаційний код транзакції]. Код транзакції співпадає з тим, що знаходиться у URL сторінки завантаження, отже він нам відомий.

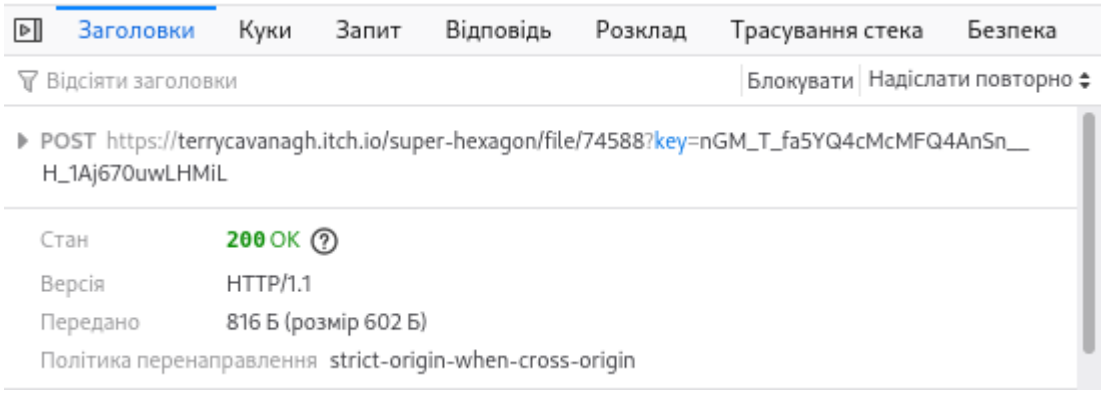


Рисунок 6 —Заголовок запиту, виконаного в JavaScript сайту

Код файлу можна знайти напряму в HTML-документі, бачимо, що з кнопками “Скачать” пов’язаний атрибут data-upload_id=”[ідентифікаційний код файлу]”, що співпадає з кодом у URL для POST-запиту. Ці кнопки також мають свій власний клас download_btn (англ. “download button” — кнопка завантаження), це дозволить нам чітко дізнатися коди для всіх доступних файлів.

Розглянемо відповідь на наш запит, яку надіслав сервіс.

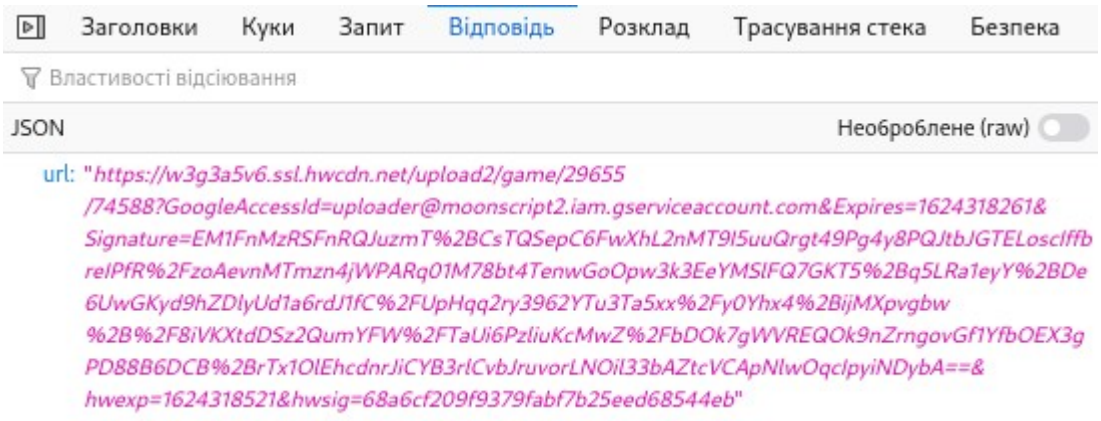


Рисунок 7 — Відповідь на запит з рисунку 6

Бачимо, що сервер надсилає у відповідь дуже довгий URL, і вже перейшовши на нього, браузер починає власне завантажувати потрібний файл. Судячи з наявних параметрів Expire та Signature, це посилання буде дійсним лише протягом певного часу, і також воно певним чином прив’язано до конкретного акаунту.

Дійсно, якщо перейти за цим посиланням на декілька годин пізніше, до доступ до файлу не надається. Також, якщо цей запит надіслати за допомогою інструменту curl, побачимо помилку 401 Unauthorized, що свідчить про те, що необхідно певним чином автентифікуватися.

На щастя, можна легко надіслати певні дані для автентифікації, оскільки вони зберігаються в cookies браузера. Дійсно, якщо скопіювати header Cookie з запиту, виконаного браузером, у запит, виконаний Curl, то отримаємо бажаний результат — посилання дійсно працює, і ми можемо завантажити потрібний контент.

2.3 Результати дослідження

Ми побачили, як влаштовані HTML-сторінки ігор itch.io, які дані в них зберігаються, і як за допомогою цих даних клієнтська частина сервісу взаємодіє з серверною.

На основі цих знанням будемо проектувати та розробляти свій власний програмний засіб, що дозволятиме не тільки запускати веб-ігри, зайшовши на відповідні сторінки, а й зберігати їх та запускати в будь-який момент у режимі офлайн.

Для цього необхідно буде доповнити або певним чином змінити поведінку клієнту, за який ми відповідаємо, проте такий метод розробки набагато простіший, ніж розробка клієнту з нуля. Були розглянуті декілька потенційних випадків використання, які відносяться до теми нашої роботи, проте itch.io підтримує багато інших функцій, наприклад коментарі, форуми, “кабінет творця” зі статистикою щодо ігор, опублікованих акаунтом, тощо — наш програмний

засіб не модифікує цей функціонал, а завдяки використанню вже розробленого фронтенду ми отримуємо до них доступ без зайвих зусиль.

Недоліком розглянутого методу є той факт, що алгоритми, що виконуватимуться на нашій власній клієнтській частині, спираються на специфіку фронтенду (наприклад, конкретні HTML-класи) та недокументованого бекенду (відповіді на запити JavaScript), які ми не контролюємо, і які можуть у будь-який момент змінитися. Вважатимемо, що в цьому випадку можна буде досить швидко оновити додаток, виправивши алгоритми, але в цілому процес збереження ігор не повинен значним чином змінюватися.

3 РОЗРОБКА ДОДАТКУ ANDROID

3.1. Формулювання вимог до програмного засобу

Перед початком розробки програмного засобу, який до цього часу згадували у роботі, варто сформулювати більш чіткі вимоги з точки зору користувача.

З досвіду розробників поступових вебзастосунків (progressive web apps, які раніше описували), можливість запустити такий додаток моментально, без попереднього встановлення, може бути вкрай бажаною: наприклад, компанія Starbucks подвоїла кількість онлайн-замовлень після введення свого PWA як альтернативу стандартному додатку iOS (до того ж із значним заощадженням трафіку, оскільки веб-додаток на 99% менший).[22]

Такі показники властиві не для всіх додатків, наприклад, більш громіздкі програми, що використовуються регулярно, або такі, що мають значнішу потребу в швидкодії, краще встановити як нативне програмне забезпечення.

У випадку розглянутих нами HTML5-ігор ми можемо не отримати значних збережень використаного трафіку, оскільки наші збережені офлайн-версії будуть ідентичними до версій, наданих сайтом; при цьому ігри, опубліковані на itch.io, часто дійсно невеликі, і багато гравців запустять таку гру лише один раз.

Залишається одна значна перевага саме веб-ігор, яку варто було би зберегти в нашому розробленому програмного засобу, а саме можливість моментального запуску без зайвого процесу встановлення. Це особливо підходить для ігор, розроблених командами із 1-2 людей, що є типовим для itch.io.

Отже до списку вимог до програмного засобу, який би встановляв такі додатки для режиму онлайн, слід додати, що такий засіб не повинен ускладнювати встановлення.

Враховуючи невеликий розмір ігор та зріст ємності носів мобільних пристроїв, було би доцільно навіть зберігати ігри в кеш-пам’яті до тих пір, поки

користувач їх сам не видалить. Система Android автоматично сповіщає про недостаток вільної пам’яті на пристрої.

Також важливою є можливість оновлювати веб-ігри — про це не варто задумуватися розробникам та користувачам типових веб-застосунків, оскільки в будь-який момент можна замінити файли, надані сервером, і клієнт їх отримає при повторному з’єднанні. Наш програмний засіб розробляється для запуску в режимі офлайн, проте варто врахувати очікування людей щодо можливості оновлення веб-ігор і окремо це імплементувати.

В результаті ми маємо отримати додаток, який:

- в цілому використовує фронт-енд сайту itch.io для надання повного функціоналу сервісу
- при запуску веб-ігор автоматично їх зберігає для можливості подальшого запуску в офлайн-режимі
- надає користувачу каталог встановлених та доступних в офлайн-режимі ігор
- за можливості, періодично їх оновлює
- надає можливість видаляти встановлені ігри

3.2 Засоби розроблення

Як уже зазначалося раніше, для цієї роботи використовували інтегроване середовище розробки Android Studio та мова Kotlin. Компанія Google, що є розробником системи Android та середовища розробки Android Studio, рекомендує Kotlin для створення нових додатків.[6]

Одна з переваг мови Kotlin над Java є спрощення роботи з мультипоточністю. В мові Kotlin є поняття coroutine (з англ. “співпрограма”) — це програмний модуль, що забезпечує взаємодію з іншими модулями за принципом кооперативної мультизадачності: модуль може призупинити дію на заданому етапі та передати контроль іншому модулю, зберігаючи свій стек та положення. Вони є більш гнучкими за звичайні підпрограми.[23]

В нашому випадку мультипоточність буде використовуватися для виконання наших додаткових вимог під час браузингу сайту, а точніше для парсингу HTML-сторінок для виведення з них метаданих про ігри, збереження їх в локальну базу даних для зображення каталогу, тощо, не перериваючи рендеринг цих сторінок.

Для завантаження HTTP-даних використаємо бібліотеку OkHttp, яка має відкритий висхідний код, і широко застосовується в програмах Java. Однією з її переваг є можливість налаштування кешування, що нам знадобиться для запуску ігор в офлайн-режимі.

Для синтаксичного аналізу (парсингу) HTML використаємо бібліотеку Jsoup, яка також є вільним програмним забезпеченням, що імплементує HTML5, зокрема дозволяє досить швидко і легко шукати елементи використовуючи DOM методи, аналогічні функціям JavaScript та CSS-селекторам у браузерах.

Основна частина користувацького інтерфейсу використовує власне веб-інтерфейс сайту itch.io, завдяки вбудованій у Android можливості WebView. WebView дозволяє показувати, а також маніпулювати веб-сторінки, інтегруючись з нативними елементами інтерфейсу. До зазначених раніше переваг такого методу розробки можна додати ще такі:

- отримуємо можливість легко авторизувати запити до сайту itch.io, зчитуючи Cookie з WebView
- інтерфейс виглядатиме так само, як і мобільний сайт, що може бути зручним для користувача, оскільки не доведеться звикати до зовсім нового інтерфейсу

3.2.1 Специфіка взаємодії з елементом WebView системи Android

Для підвищення рівня безпеки, а також для зменшення розмірів файлів APK, система Android надає один стандартний рушій веб-браузера Chromium, що є спільним для всіх додатків, що використовують елементи інтерфейсу WebView.

Цей рушій вважається окремим пакунком, і він запускається в процесі, ізольованому від додатків, що на нього спираються.

Така архітектура має свої переваги, проте недоліком є ускладнення взаємодії між додатком та процесом браузера. Наш програмний засіб потребує певного втручання в елемент WebView, по-перше, щоб зчитувати дані з HTML-сторінок, відвіданих користувачем, а по-друге, щоб, можливо, модифікувати певні їх елементи.

Можливість редагування структури DOM може бути корисним, скажімо, для прибрання стандартної кнопки “Run game” (див. рисунок 1) і її заміни на кнопку, яка би запускала гру специфічним чином, зберігаючи її для офлайн-режиму.

Відокремлення процесу браузера від нашого додатку означає, що комунікація між ними відбувається через певний API, який має певні обмеження, зокрема немає прямого доступу ні до HTML, ні до DOM-структури. Проте є методи, які дозволять це зробити не напряму:

- Замість зчитування отриманих HTML-сторінок, WebView API надає можливість перехоплювати HTTP-запити. Таким чином ми зможемо взяти на себе відповідальність за отримання сторінок, в окремому потоці (а точніше співпрограмі Kotlin) виконати синтаксичний аналіз і отримати потрібні дані, одночасно надаючи цю сторінку WebView для показу користувачу. Також цей принцип також стане ключовим для збереження ігор в офлайн-режим.
- Маніпулювати показаним документом DOM можна під час перехоплення, проте простіше цим займатися не вдаючись до мультипоточності, а вже після того, як WebView отримав та запарсив документ. Стандартний API надає метод для виконання довільного коду JavaScript — отже можна вставити код, що реагує на подію “DOMContentLoaded”, тобто чекає завершення парсингу документу, і в цей момент починає маніпулювати структурою DOM стандартними функціями браузерного JavaScript.

3.3 Встановлення ігор в офлайн-режим та проектування додатку

За сформульованими нами вимогами до програмного засобу, зберігання ігор для їх подальшого використання в офлайн-режимі повинно відбуватися відразу під час запуску гри. Тому доцільніше розглядати цей процес не як встановлення або завантаження гри, а як кешування.

Під поняттям кешування, у контексті веб-технологій, зазвичай розуміють процес тимчасового зберігання документів, зображень тощо, в цілях зменшення серверних затримок.

Система веб-кешу зберігає копію контенту, що проходить через неї; подальші запити тих же даних можна задовільнити тим, що вже наявне в кеші, за певних умов. Протокол HTTP визначає такі механізми управління кешем[24]:

- Поняття “свіжості” (англ. freshness) дозволяє використати відповідь без повторної її перевірки на сервері. Наприклад, HTTP-заголовок відповіді “Expires” вказує на дату й час, коли документ стане застарілим; а до того моменту можна скористатися наявною версією
- Додаткова “перевірка” (англ. validation) може виконуватися, щоб дізнатись, чи є кешована відповідь ще придатною для використання після того, як вона застаріла. Валідація може виконуватися на основі дати останньої модифікації документу зі сторони сервера (умовний запит із використанням заголовку “If-Modified-Since” на рисунку 8) або ж за допомогою механізму ідентифікації версій файлу ETag (умовний запит “If-None-Match” на рисунку 8).
- Анулювання кешу може бути побічним ефектом запитів, таких як POST, PUT та DELETE

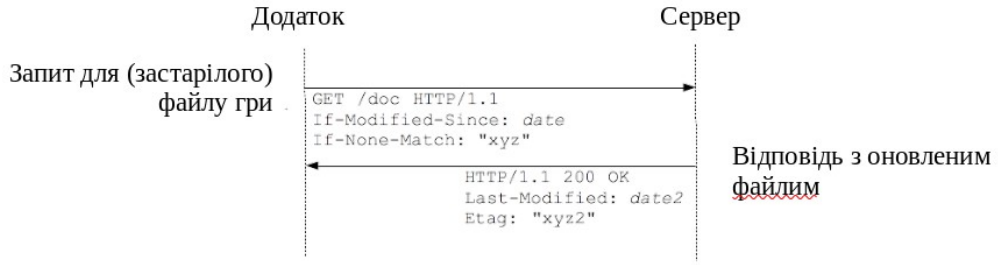


Рисунок 8 — Умовні запити HTTP

Мережа розповсюдження контенту (CDN), яку використовує сервіс itch.io, підтримує всі стандартні механізми кешування, що прискорює завантаження ігор та заощаджує трафік. Для наших цілей слід додати ще одне нестандартне правило:

– У випадку, коли запит не успішний, показуємо кешований файл. В офлайн-режимі краще показати те, що вже збережено, навіть якщо цей контент застарілий. Також в якості оптимізації для мобільних пристроїв можна явно перевірити, чи має пристрій зв'язок з мережею мобільного інтернету, Wi-Fi і т.д.: якщо ні — не будемо чекати помилки від HTTP-запиту, а відразу покажемо збережений контент.

  Для цього створимо обгортку над кешем, наданим бібліотекою OkHttp.

Також важливо, щоб ці дані зберігалися не в розділі власне кеш-пам'яті Android (де передбачається можлива очистка пам'яті в будь-який момент без перешкоди працездатності додатків), а в постійній пам'яті, аби користувач сам міг керувати встановленими іграми та уникнути їх випадкового видалення.

Звісно, що така поведінка недоцільна для загального користування сайтом itch.io, оскільки основний функціонал (перегляд каталогу ігор, їх перший запуск, завантаження файлів тощо) неможливий в режимі офлайн, і немає сенсу в тому, щоб вічно зберігати всі відвідані сторінки. Тому введемо спеціальний режим гри, в який ввійдемо натисканням модифікованої кнопки "Запустити".

Як тільки користувач заходить на сторінку з описом гри, наш програмний засіб зчитує метадані, такі як назву гри, та її унікальний ідентифікаційний номер на сайті itch.io, і зберігає це в локальну базу даних.

При натисканні кнопки "Запустити" програмний засіб створить окремий кеш для гри з даним ідентифікаційним кодом, та переходить в режим гри, тобто саме з цього моменту дані, на які посилається гра, будуть зберігатися у відповідний спеціальний кеш.

Передбачається можливість переліку "встановлених" таким чином ігор: користувач бачить список назв та ескізів ігор, з аналогічними кнопками для запуску.

Слід врахувати, що користувач не завжди буде зацікавлений у грі, опис якої він щойно прочитав, тому також імплементуємо періодичну очистку зайвих рядків у базі даних, тобто видалення даних про ігри, які відомі нашому програмному засобу, проте ніколи не були запущені, а отже не зберігаються в кеш-пам’яті та не відображаються в каталозі.

Ще однією періодичною задачею є оновлення файлів. Раніше ми зазначили, що коли мова йде про онлайн-ігри, як і гравці, так і розробники розраховують на простоту оновлення ігор, тобто їх файли можуть досить часто змінюватися. Звичайно, це тривіально виконати під час запуску гри, оскільки використовуються стандартні механізми валідації кешу, і при наявності Інтернет-зв’язку можна перевірити, чи є певний файл застарілим. Проте через те, що така перевірка виконується під час запуску, можуть виникнути неочікувані затримки. Можливою оптимізацією є виконання автоматичних запитів до файлів гри, наприклад, з інтервалом в 1-2 дні. Така поведінка притаманна стандартним сервісам постачання додатків, таких як Google Play.

На рисунку 9 зображена приблизна архітектура нашого програмного засобу. Нижче пунктирної лінії зображено елементи користувацького інтерфейсу, вище — приховані від користувача компоненти.

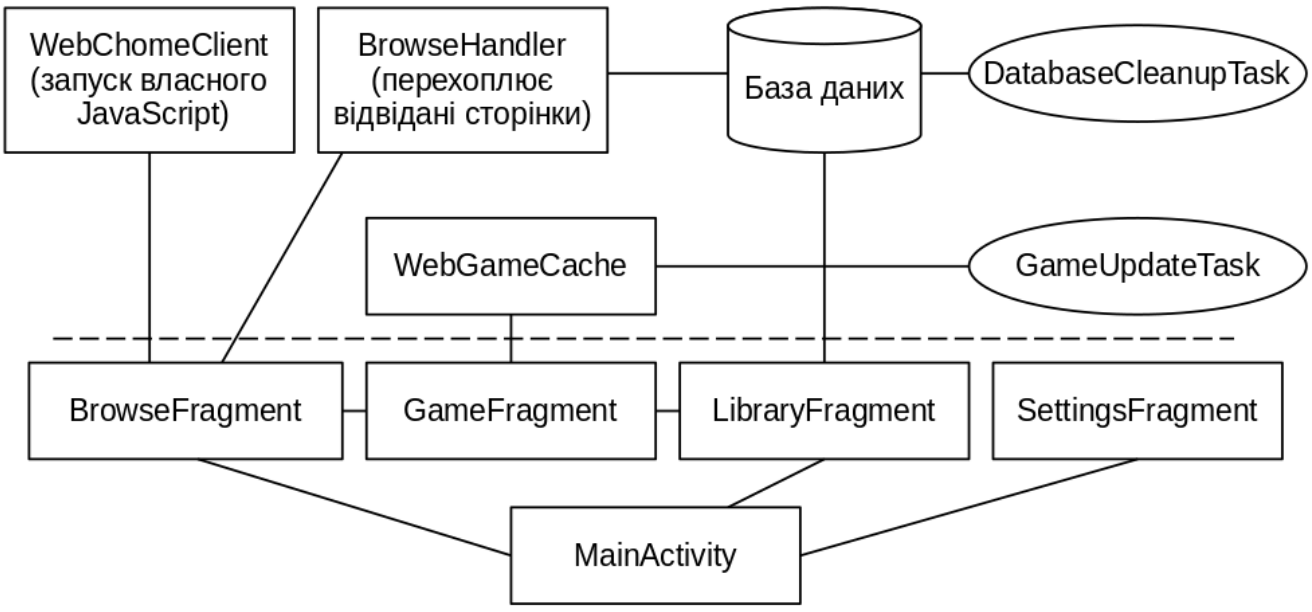


Рисунок 9 — Спроектована архітектура додатку

3.4 Графічний інтерфейс користувача

Оскільки наш додаток призначений для системи Android, варто дотримуватися таких же принципів розробки графічного інтерфейсу, як і інші мобільні додатки, тому притримуємося принципам матеріального дизайну.[25] Ідея дизайну полягає в інтерфейсі, поведінка і вигляд якого наслідують паперові картки в реальному житті.

Багато Android-додатків використовують такий елемент дизайну, як Top app bar — це полоса (“bar”), що знаходиться зверху (“top”) екрана, і показує певний контент та дії, пов’язані з поточним контекстом (“app”).

На рисунку 10 бачимо кнопку у вигляді трьох горизонтальних полосок — це дає доступ до іншого функціоналу додатку. “Page title” є заголовком теперішнього контексту, з яким пов’язані всі інші кнопки.

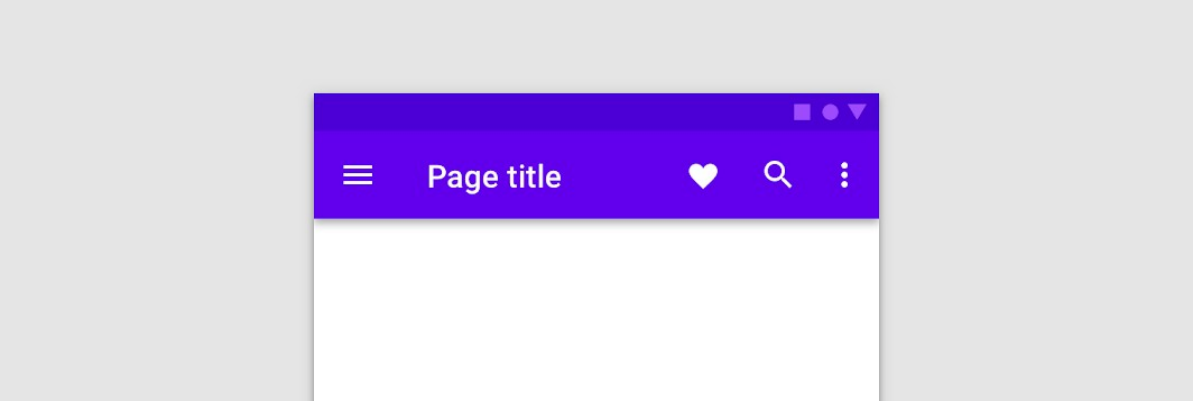


Рисунок 10 — Приклад типового top app bar інтерфейсу

Проблема постає у тому, що сайт itch.io вже має власні полоси зверху (рис. 11). Якщо використовувати веб-інтерфейс сайту, і поверх нього будувати мобільний інтерфейс, то не варто використовувати стандартний app bar. Замість цього використаємо менш вживаний елемент матеріального дизайну — bottom app bar (нижню полосу).

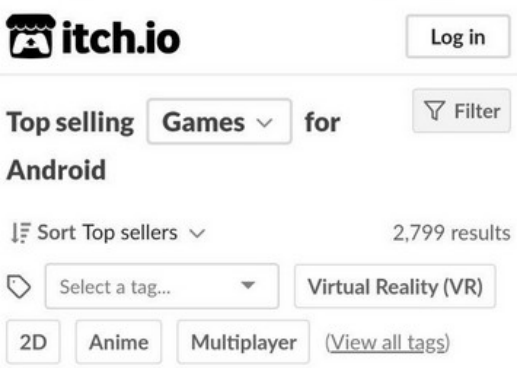
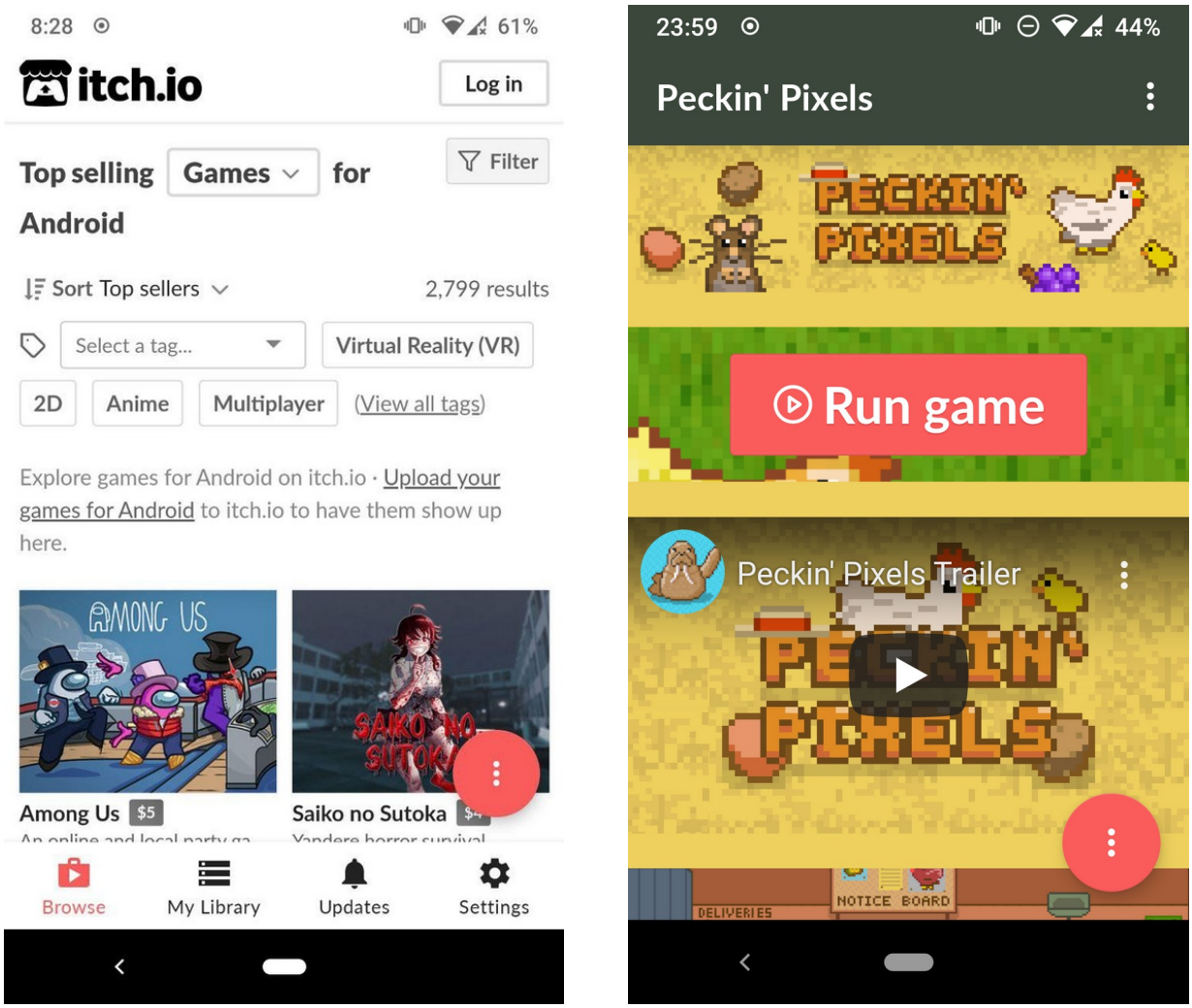


Рисунок 11 — Верхні полоси інтерфейсу сайту itch.io

Кінцевий вигляд оболонки над веб-інтерфейсом, з використанням нижньої полоси, зображено на рисунку 12.



а) б)

Рисунок 12 — Веб-інтерфейс сайту, вбудований в інтерфейс додатку:
а — основна частина сайту; б — сторінка в крамниці для певної гри

В цьому випадку нижня полоса (рис. 12а) дає нам доступ до іншого функціоналу, наприклад, перевірки оновлень. Всі функції, окрім “огляду” (“browse”) мають нативний інтерфейс, тобто інтерфейс, не оснований на веб-сайті. Взагалі нативні елементи мають такі самі кольори, як і сайт, що дозволяє легко змішувати їх із веб-інтерфейсом.

Єдиний виняток — це власне сторінки ігор у крамниці, де видавець чи розробник гри може надати свою палітру кольорів (рис. 12б). В такому разі ми заховаємо bottom app bar, який стилістично не підходить до палітри, а замість цього додамо інші елементи, які співпадають за кольором, і при цьому теж дають доступ до функцій додатку.

У правому нижньому кутку рисунку 12а та 12б бачимо застосований елемент інтерфейсу FAB (“floating action button”), тобто кнопку, що постійно присутня трохи вище нижньої полоси, і при натисканні дає доступ до списку додаткових дій. У нашому випадку це дії, пов’язані з процесом веб-браузингу, а саме оновлення сторінки, можливість поділитися URL-адресою, а також функція пошуку ігор на сайті за назвою.

3.5 Тестування програмного засобу

Оцінити працездатність розробленого нами програмного засобу можна за кількома критеріями: можна розглянути основну ціль програми (встановлення ігор і їх подальший запуск без доступу до інтернету), або ж оцінити взагалі здатність запускати ігри.

Ці два надані критерії дійсно відрізняються один від одного, оскільки, на жаль, пошук та фільтрування ігор, сумісних зі смартфонами, на itch.io не є ідеальним.

Цікавим є те, що itch.io дозволяє розробникам попереджати саме користувачів мобільної версії сайту про те, що їх гра не розрахована на такі пристрої (рис. 13), проте відфільтрувати такі ігри під час пошуку на сайті не можна.

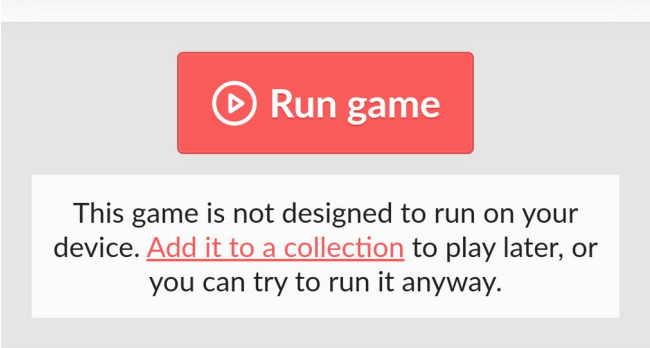


Рисунок 13 — Попередження про можливу несумісність гри з мобільним пристроєм

До цього моменту ми казали про фільтрацію ігор “для пристроїв із сенсорним екраном”, проте це є ширшою категорією, ніж смартфони чи навіть мобільні пристрої. Незважаючи на це, сайт itch.io все одно дозволяє запускати будь-яку гру на смартфоні, інколи під час ініціалізації вона сам теж попередить про можливі проблеми. Зокрема ігри, основані на рушії Unity, попереджають про те, що Unity WebGL офіційно не підтримує мобільні пристрої (рис. 14), пояснюючи це їх здебільшого недостатньою потужністю та кількістю оперативної пам’яті.

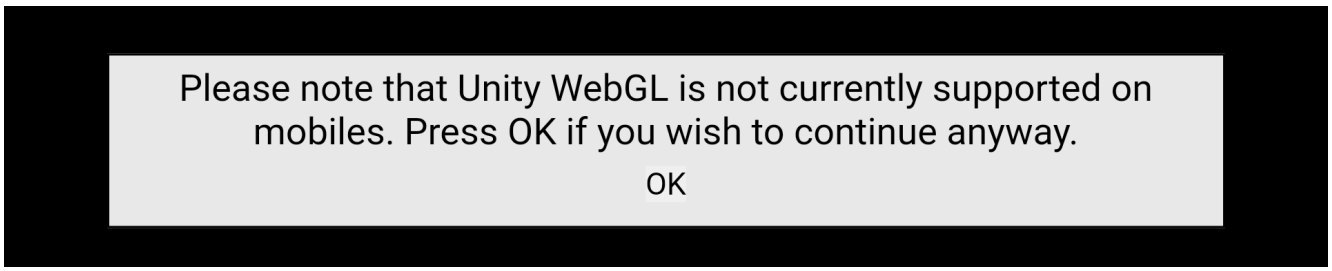


Рисунок 14 — Гра на основі Unity попереджає про відсутність підтримки мобільних пристроїв

Тим не менш, відсутність офіційної підтримки не завжди є перешкодою: успішність запуску конкретної гри чітко залежить лише від її власних технічних вимог та дієздатності мобільного пристрою, тобто сучасніші смартфони рідше стикатимуться з проблемами. Проте в будь-якому випадку це ускладнює процес тестування, оскільки одним із вирішальним фактором стає сам пристрій, на якому проводиться перевірка.

В цій роботі для тестування використаємо Sony Xperia XA2, який вийшов у продаж у 2018 році, тобто на даний момент має чотирьохлітню давність, що за деякими даними є більш ніж середнім показником віку смартфонів (наприклад, в США купують нові смартфони в середньому кожні 2-3 роки).

Візьмемо 20 ігор з категорії “для пристроїв із сенсорним екраном” відсортованих як “топ продажів”, запустимо їх спершу в режимі онлайн, потім вимкнемо доступ до інтернету і запустимо збережену версію гри в офлайн-режимі. Отримали такі результати:

- кількість ігор, що висвічують попередження про запуск на мобільному пристрої: 8
- кількість ігор, що дійсно не запустилися на нашому пристрої: 4
- кількість ігор, що запустилися в онлайн-режимі, але не запустилися в офлайн-режимі: 0

Бачимо, що основна ціль нашого програмного засобу виконана — усі ігри, що працюють в режимі онлайн, також працюють без наявного зв’язку до мережі. Також помітно значно швидший запуск ігор, оскільки додаток автоматично перенаправляє запити відразу в кеш-пам’ять при вимкненому Wi-Fi та доступу до мобільного інтернету. Ігри відносно невеликого розміру запускаються майже миттєво в офлайн-режимі у порівнянні з першим запуском в онлайн-режимі, що займає кілька секунд. Інші, більш ресурсозатратні ігри, зокрема на основі Unity, отримують менш значне прискорення — приблизно 30%.

Недоліком є те, що не вдалося відфільтрувати всі ігри, що не запускаються на середньостатистичному смартфоні, а також наявність хибно позитивної класифікації ігор, що “не підтримуються на даному пристрої”, навіть коли вони запускаються без проблем.

ВИСНОВКИ

В цій роботі ми порівняли різні методи розробки додатків з використанням веб-технологій, здатних до роботи в режимі офлайн, та створили власний програмний засіб під назвою Mitch, що дозволяє встановлювати веб-додатки локально на пристрої Android, не вимагаючи ніяких зусиль від їх розробників. Функціонал цієї програми обмежений, оскільки він здатен встановлювати лише ігрові застосунки з одного сайту itch.io, проте ми показали доцільність та унікальність такого засобу. Додаток Mitch відповідає заданим нами вимогам, а також загальним сучасним вимогам до графічних інтерфейсів.

Додаток Mitch можна використовувати в цілях розваги, оскільки більша частина контенту, яка розповсюджується через сервіс itch.io — це саме відео-ігри. Можливість запускати їх в офлайн-режимі, навіть коли це не передбачалося розробниками, розширює асортимент ігор, доступних у ситуаціях з низькоякісним або відсутнім зв’язком з інтернетом, наприклад, у метро, в підвалах і т.д.

Створений в рамках цієї роботи додаток Mitch було опубліковано на сервісах itch.io та F-Droid; на даний момент має кілька сотень користувачів, і ця кількість зростає.

Оскільки додаток Mitch опублікований і має користувачів, варто продовжити його розробку, додавши новий функціонал, щоб його використання було доцільним у порівнянні з альтернативами. Зокрема варто додати можливість встановлювати не лише веб-додатки, а й інші види файлів, розміщених на сайті itch.io.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Statista: Number of smartphone users worldwide from 2016 to 2026 [Електронний ресурс]. — Режим доступу: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
2. Statista: Internet users in the world 2022 [Електронний ресурс]. — Режим доступу: https://www.statista.com/statistics/617136/digital-population-worldwide/
3. Digital divide persists even as Americans with lower incomes make gains in tech adoption [Електронний ресурс] / Emily A. Vogels // Pew Research Center. — 2021. — Режим доступу: https://www.pewresearch.org/fact-tank/2021/06/22/digital-divide-persists-even-as-americans-with-lower-incomes-make-gains-in-tech-adoption/
4. Android Developers: Build for billions [Електронний ресурс]. — Режим доступу: https://developer.android.com/docs/quality-guidelines/build-for-billions
5. Progressive Web Apps Are Here and They're Changing Everything [Електронний ресурс] / Ladage A. // DEG. — 2018. — Режим доступу: https://www.degdigital.com/insights/progressive-web-apps/
6. Android Studio: An IDE built for Android. Android Developers Blog. [Електронний ресурс] — Режим доступу: https://android-developers.blogspot.in/2013/05/android-studio-ide-built-for-android.html
7. Kotlin is now Google's preferred language for Android app development [Електронний ресурс] // TechCrunch. — Режим доступу: http://social.techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/
8. Веб-застосунок — Київський університет імені Бориса Грінченка [Електронний ресурс]. — Режим доступу: http://wiki.kubg.edu.ua/Веб-застосунок

9. Design Guidelines: Glossary [Електронний ресурс]. — Режим доступу: http://java.sun.com/products/jlf/ed1/dg/higq.htm
10. What Does AJAX Even Stand For? [Електронний ресурс]. — Режим доступу https://thehistoryoftheweb.com/what-does-ajax-even-stand-for/
11. Progressize web app structure [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/App_structure
12. Firefox just walked away from a key piece of the open web [Електронний ресурс] / Newman J. // Fast Company. — 2021. — Режим доступу: https://www.fastcompany.com/90597411/mozilla-firefox-no-ssb-pwa-support
13. Trusted Web Activity: Quick Start Guide [Електронний ресурс]. — Режим доступу: https://developer.chrome.com/docs/android/trusted-web-activity/quick-start/
14. Electron Internals: Using Node as a Library [Електронний ресурс]. — Режим доступу: https://www.electronjs.org/blog/electron-internals-using-node-as-a-library
15. Tauri: Build smaller and more secure desktop applications with a web frontend [Електронний ресурс]. — Режим доступу: https://github.com/tauri-apps/tauri
16. Chapter 1. What Is React Native? [Електронний ресурс] / Bonnie Eisenman // O'Reilly Media, Inc. — 2015. — Режим доступу: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html
17. The rise and rise of .io games [Електронний ресурс] / Castello Jay // Rock Paper Shotgun. — 2018. — Режим доступу: https://www.rockpapershotgun.com/the-rise-and-rise-of-io-games
18. How Flash Games Changed Video Game History [Електронний ресурс] / Ben Reeves // Game Informer. — 2018. — Режим доступу: https://www.gameinformer.com/2018/12/22/how-flash-games-changed-video-game-history

19. How would you feel about an unofficial Android itch.io client? [Електронний ресурс]. — Режим доступу: https://itch.io/t/495651/how-would-you-feel-about-an-unofficial-android-itchio-client
20. Command-line itch.io helper [Електронний ресурс]. — Режим доступу: https://github.com/itchio/butler
21. Go support for Mobile devices [Електронний ресурс]. — Режим доступу: https://github.com/golang/mobile
22. Case Study: Starbucks Progressive Web App [Електронний ресурс]. — Режим доступу: https://formidable.com/work/starbucks-progressive-web-app/
23. Кнут Д. Сопрограммы // Искусство программирования, том 1. Основные алгоритмы, 3-е издание / Кнут Д. — М.: Вильямс, 2006. — 229—236 с
24. Using HTTP Link: Header for Gateway Cache Invalidation [Електронний ресурс] / Kelly, Mike; Hausenblas, Michael // WS-REST. — 2010. — Режим доступу: http://ws-rest.org/files/03-Link%20Header-based%20Invalidation%20of%20Caches.pdf.
25. Introduction – Material Design [Електронний ресурс]. – Режим доступу: https://material.io/design/introduction