Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка платформи цифрової дистрибуції програмного забезпечення та ігор з використанням технології Chromium Embedded Framework

Виконав: студент IV курсу, групи СП-42
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Харлан А. А.
(підпис) (прізвище та ініціали)

Керівник _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2023

АНОТАЦІЯ

Кваліфікаційна робота на здобуття освітнього ступеню «бакалавр» за спеціальністю 121 – Інженерія програмного забезпечення. Тернопільський національний технічний університет ім. Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42, 2023 рік. Пояснювальна записка до кваліфікаційної роботи на здобуття освітнього ступеню «бакалавр» містить: 71 с., 37 рис., один додаток.

Тема: Розробка платформи цифрової дистрибуції програмного забезпечення та ігор з використанням технології Chromium Embedded Framework.

В атестаційній роботі бакалавра висвітлено ключові елементи в розробці платформи цифрової дистрибуції за допомогою такої технології, як Chromium Embedded Framework , а також використання HTML, CSS та JavaScript для створення інтерфейсу користувача програми, створення власних частин програми на основі мови C++ . Зображено макети сторінок та можливостей додатку. Розроблено діаграми варіантів використання, класів та послідовностей. Створено додаток, що демонструє властивості Chromium браузера в поєднанні з вебдодатком.

Ключові слова: додаток, CEF, платформа цифрової дистрибуції, API, JavaScript, C++.

ANNOTATION

Qualification work for the educational degree "Bachelor" of the specialty 121 - Software Engineering. Ivan Puluj Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, group SP-42, 2023. Explanatory note to the qualification work for the bachelor's degree contains: 71 p., 37 figures, one addition.

Topic: Development of a digital distribution platform of software and games using Chromium Embedded Framework technology.

The bachelor's thesis highlights the key elements in the development of a digital distribution platform using technology such as the Chromium Embedded Framework, as well as the use of HTML, CSS and JavaScript to create a user interface for the program, creating your own parts of the program based on the C++ language. The layouts of pages and application features are shown. Diagrams of use cases, classes, and sequences are developed. An application is created that demonstrates the properties of the Chromium browser in combination with a web application.

Keywords: application, CEF, digital distribution platform, API, JavaScript, C++.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – прикладний програмний інтерфейс.

CEF (Chromium Embedded Framework) – простий фреймворк для вбудовування браузерів на основі Chromium в інші програми.

CSS (Cascading Style Sheets) – спеціальна мова стилю сторінок.

GUI (Graphical user interface) – графічний інтерфейс користувача.

HTML (HyperText Markup Language) – мова розмітки гіпертексту.

UML (Unified Modeling Language) – уніфікована мова моделювання.

ДНАОП – державні нормативні акти з охорони праці.

ООП – об'єктно-орієнтоване програмування.

ПЗ – програмне забезпечення.

ПК – персональний комп'ютер.

ВСТУП

Додаток під назвою “Onirique”, що можна перекласти з французької, як "мрійливий" або "фантастичний", розроблений як платформа-посередник для розповсюдження відеоігор, де користувачі можуть переглядати, купувати продукти за допомогою посилань на магазини. Індустрія ігрових магазинів поділяється на: ПК, консолі та мобільні ігри. Ця платформа цифрової дистрибуції буде націлена на ПК ринок, так як це найбільш можливо реалізувати. У той час як у сфері консольних існують магазини, що є єдиними доступними на платформі, такі як PlayStation Store (Sony) та Microsoft Store (Microsoft), тощо, у сфері комп'ютерних ігор є реальна можливість створення власної платформи для дистрибуції ігор.

Розробка платформи цифрової дистрибуції програмного забезпечення та ігор є створенням системи розповсюдження ігор, не порушуючи при цьому ліцензійну угоди, як це можливо у випадку з піратством. Наданий додаток допоможе полегшити купівлю та обговорення програмного забезпечення, ігор, тощо. Інструменти, що будуть орієнтиром в програмі, такі як пошук, список бажаного, список рекомендацій, а також необов’язкові способи зміни дизайну додатку.

Щодо мною поставлених завдань є реалізація системи, яка буде виконувати функцію магазину-посередника, що буде об’єднувати в собі ігри з інших платформ цифрової дистрибуції, можливість їх перегляду, розуміння на якій платформі існує можливість купівлі певного програмного забезпечення або гри всередині цього додатку, тобто буде певним довідником, провідником та об’єднанням багатьох магазинів в один.

Для тестового режиму створено реєстрацію за допомогою акаунту Facebook, який дасть можливість додавати певний продукт у список бажань користувача додатку.

Для незареєстрованих користувачів функціонал програми буде обмежений. Без автентифікації буде лише можливість перегляду додатку.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд конкурентів

Так як індустрія поділяється на ринок ПК, консолей та мобільний, то можна виділити декілька великих та потужних конкурентів. Для початку можна згадати конкурентів, що не є основними. Це такі, як: Playstation Store, Nintendo eShop (рис. 1.1), в яких можна купувати з ПК, але продукт буде лише доступний на відповідній платформі (Playstation 4/5, Nintendo Switch).

Рисунок 1.1 – Логотипи крамниць Playstation Store та Nintendo eShop

А також App Store, Google Play, ситуація яких така сама, але продукти доступні на мобільних телефонах Apple та Android відповідно (рис. 1.2).

Рисунок 1.2 – Логотипи магазинів App Store та Google Play

В списку до консолей можна було б згадати Microsoft Store (рис. 1.3), що представлений на платформі Xbox. Також існує на ПК у вигляді стандартного магазину операційної системи Windows [1].

Рисунок 1.3– Логотип Microsoft Store

Найбільша його зручність полягає у можливості купівлі підписки на внутрішній сервіс – Xbox Game Pass, який за невелику ціну пропонує широкий асортимент відеоігор. Сам інтерфейс програми є досить примітивним, але зручним. На головній сторінці показано все, пошук, пропозиції додатку, списки безкоштовних та платних ігор, бестселерів тощо [1].

Наступним конкурентом можна виділити Epic Games Store (рис. 1.4), що є відносно новим гравцем на ринку, всього з 2018 року.

Рисунок 1.4– Логотип Epic Games Store

Свою популярність набув за допомогою безкоштовних роздач однієї платної гри, а інколи і двох, кожного тижня. Інтерфейс програми є простим та зрозумілим, але багато користувачів скаржаться примітивний дизайн, повільну роботу програми та процес перегляду ігор. Але на даний момент є одним з найбільших сервісів, яким користуються люди. Частота знижок, можливість створення бібліотеки ігор лише на безкоштовній роздачі [2].

Головним ж конкурентом можна назвати платформу Steam (рис. 1.5), що є гігантом індустрії серед цифрових магазинів ПК ринку.

Рисунок 1.5– Логотип Steam

Її розробила компанія Valve. Відкривши Steam, можна побачити головну сторінку, де крамниця одразу рекомендує ігри, які можуть сподобатись користувачу, великий банер, який повідомляє про знижки на даний момент або акції. Також в іншій вкладці можна побачити бібліотеку Steam, в якій є можливість перегляду колекції ігор, досягнень, власної статистики. Також ця платформа є досить соціальною, можливо взнати чим займаються ваші друзі, побачити їх профіль [3]. Також є можливість використання Steam для встановлення модифікацій і доповнень. Ви також можете додавати ігри, які не підтримуються Steam. У Steam можна придбати ігри, доповнення та пакети розширення. Ви можете переглядати різні категорії, або, якщо ви знаєте, яка саме гра вам потрібна, ви можете здійснити її пошук. Є можливість купівлі або дарування гри одному з

друзів. Однією з найунікальніших функцій Steam є власний внутрішній ринок, на якому можна продавати, купляти та обмінювати речі з особистого інвентарю користувача. Також від інших подібних магазинів, Steam відрізняє великий спектр змін власного профілю. Це можуть бути як шпалери особистої сторінки, так і найрізноманітніші способи змінити її. На платформі представлені майже всі ігри минулого та сьогодення [3].

Якщо розглядати даних конкурентів зі сторони цінової політики, більшість з них мають схожі ціни на одні і ті ж ігри, тому користувачам більше до вподоби платформа Steam, яка має набагато більший функціонал, якщо порівнювати з попередніми згаданими крамницями.

1.2 Обґрунтування вибору напрямку дослідження

На роль формату платформи цифрової дистрибуції було обрано додаток для ПК з операційною системою Windows. Хоч конкурентні програми існують, як у вигляді веб-сторінок, так і у вигляді додатків, але в цьому випадку буде розроблена лише додаток.

Створюватись це все буде за допомогою Chromium Embedded Framework [4]. Це програмний фреймворк з відкритим вихідним кодом для вбудовування браузера Chromium в інший додаток. Це дозволяє додавати до свого додатку функцію веб-перегляду, а також можливість використовувати HTML, CSS та JavaScript для створення користувацького інтерфейсу додатку. CEF - це проект з відкритим вихідним кодом заснований на проекті Google Chromium в 2008 році. На відміну від самого проекту Chromium, який зосереджений в основному на розробці додатків для Google Chrome, CEF фокусується на полегшенні використання та вбудовання браузера в додатки сторонніх розробників. CEF пропонує стабільні API виробничої якості, гілки релізів, що відстежують конкретні випуски Chromium, та бінарні дистрибутиви. Більшість функцій у CEF мають реалізацію за

замовчуванням, яка забезпечує багату функціональність, вимагаючи від користувача мінімальної роботи з інтеграції або взагалі не вимагаючи її. Наразі у світі налічується понад 100 мільйонів інстальованих екземплярів CEF, вбудованих у продукти широкого спектру компаній та галузей [4].

Деякі приклади використання CEF включають

- Вбудовування HTML5-сумісного елемента керування веб-браузером в існуючий нативний додаток.
- Створення легкої нативної програми-оболонки, яка містить користувацький інтерфейс, розроблений переважно з використанням веб-технологій.
- Відображення веб-вмісту "поза екраном" в додатках, які мають власні власні фреймворки для малювання.
- Виступати в ролі хоста для автоматизованого тестування існуючих веб-властивостей і додатків.

CEF підтримує широкий спектр мов програмування та операційних систем і може бути легко інтегрований як у нові, так і в існуючі додатки. Він був розроблений з нуля з урахуванням як продуктивності, так і простоти використання. Базовий фреймворк включає програмні інтерфейси C та C++, що розкриваються через власні бібліотеки. Це забезпечує тісну інтеграцію між браузером і хост-додатком, включаючи підтримку користувацьких плагінів, протоколів, об'єктів та розширень JavaScript. За бажанням хост-додаток може керувати завантаженням ресурсів, навігацією, контекстними меню, друком тощо, використовуючи при цьому ту саму продуктивність і технології HTML5, що й у веб-браузері Google Chrome [4].

Однією з декількох мов програмування додатку було обрано JavaScript. Це динамічна, об'єктно-орієнтована, прототипна мова програмування. Багато небраузерних середовищ використовують її. JavaScript - це динамічна мова, що базується на прототипах, підтримує імперативні та декларативні, об’єктно-орієнтовані стилі [5].

Ось декілька причин, чому JavaScript є хорошим вибором:

- Простота – завдяки простій структурі JavaScript легше вивчати та впроваджувати, а також працює швидше, ніж деякі інші мови. Помилки легко помітити і виправити.
- Швидкість – JavaScript виконує скрипти безпосередньо у веб-браузері без попереднього підключення до сервера або компілятора.
- Універсальність – JavaScript сумісний з іншими мовами, такими як PHP, Perl та Java.
- Популярність – існує безліч ресурсів і форумів, які допоможуть початківцям з обмеженими технічними навичками і знаннями JavaScript.
- Навантаження на сервер – ще однією перевагою роботи на стороні клієнта є те, що JavaScript зменшує кількість запитів, що надсилаються на сервер. Перевірка даних може здійснюватися через веб-браузер, а оновлення застосовуються лише до певних розділів веб-сторінки.
- Оновлення – команда розробників постійно оновлює та створює нові фреймворки та бібліотеки, забезпечуючи актуальність у галузі.

Другою мовою програмування програми обрано C++. Це рішення було прийнято так як більша частина клієнту платформи цифрової дистрибуції Steam була написана на цій мові [6].

Перевагами C++ є:

- Переносимість – C++ пропонує функцію переносимості або незалежності від платформи, яка дозволяє користувачеві легко запускати одну і ту ж програму на різних операційних системах або інтерфейсах.
- Об'єктно-орієнтована – однією з найбільших переваг C++ є об'єктно-орієнтоване програмування, яке включає в себе такі поняття, як класи, успадкування, поліморфізм, абстрагування даних та інкапсуляція, що дозволяє багаторазове використання коду і робить програму ще більш надійною.

- Низькорівневі маніпуляції – C++ дозволяє низькорівневі маніпуляції з даними на певному рівні. За допомогою C++ створюються вбудовані системи та компілятори.
- Управління пам'яттю – C++ має абсолютний контроль над управлінням пам'яттю. Це підвищує відповідальність користувача за керування пам'яттю, замість того, щоб нею керував збирач сміття. Ця концепція реалізується за допомогою динамічного виділення пам'яті з використанням вказівників.
- Масштабованість – відноситься до здатності програми до масштабування. Це означає, що програма на C++ здатна працювати як з малими, так і з великими обсягами даних.

Також C++ має свої недоліки [6], такі як:

- Питання безпеки – хоча об'єктно-орієнтоване програмування забезпечує більшу безпеку даних, що обробляються, порівняно з іншими мовами програмування, які не є об'єктно-орієнтованими, такими як C, певні проблеми з безпекою все ще існують через наявність функцій-друзів, глобальних змінних та вказівників.
- Відсутність збирача сміття – як говорилось в перевагах, C++ надає користувачеві повний контроль над управлінням пам'яттю комп'ютера. У C++ відсутня функція збирача сміття для автоматичного відфільтровування непотрібних даних.

Для створення та редагування коду проєкту було обрано середовище розробки ПЗ – Microsoft Visual Studio.

1.3 Технічний аспект проблеми

Розробка з використанням технології Chromium Embedded Framework включає кілька технічних аспектів.

Модель об'єктно-орієнтованого програмування - це парадигма розробки програмного забезпечення, яка обертається навколо концепції об'єктів, що інкапсулюють дані та поведінку. Вона підкреслює модульність, можливість повторного використання та супроводження шляхом організації коду в класи та об'єкти, полегшуючи організацію коду та сприяючи повторному використанню коду. ООП заохочує використання таких принципів, як успадкування, поліморфізм та інкапсуляція для створення модульних та розширюваних програмних архітектур [7].

При моделі ООП є можливість реалізації різних компонентів додатку для означення програмного забезпечення та ігор як об'єктів. Наприклад, можуть бути класи, що представляють користувачів, ігри, користувацькі інтерфейси та мережеві функції. Використання ООП сприяє ясності коду, розділенню завдань і можливості масштабування та розширення платформи в майбутньому [7].

Розробка платформи цифрової дистрибуції вимагає добре продуманого користувацького інтерфейсу, щоб забезпечити безперебійну та інтуїтивно зрозумілу роботу користувачів. За допомогою CEF користувацький інтерфейс створений з використанням HTML, CSS та JavaScript, застосовуючи переваги веб-технологій для розробки інтерактивних та візуально приємних інтерфейсів.

Таким чином, технічні аспекти розробки платформи для розповсюдження цифрового програмного забезпечення та ігор з використанням Chromium Embedded Framework включають модель ООП, використання архітектури “модель-вид-контролер”, розробка відповідного інтерфейсу за допомогою HTML, CSS. Використання функцій та можливостей Chromium Embedded Framework для інтеграції веб-контенту, а також застосування принципів ООП для проектування та реалізації модульних та розширюваних програмних компонентів.

2 РОЗРОБКА МОДЕЛІ ТА ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Проєктування платформи цифрової дистрибуції

Розробка платформи цифрової дистрибуції програмного забезпечення та ігор вимагає ретельного планування та врахування різних факторів для створення бездоганного користувацького досвіду. Платформа повинна задовольняти потреби як розробників, так і користувачів, забезпечуючи ефективне та безпечне середовище для розповсюдження та доступу до цифрового контенту.

Платформа має пропонувати набір комплексних інструментів для розробників та API, які дозволять розробникам легко інтегрувати своє програмне забезпечення та ігри в платформу.

Управління цифровими правами має вирішальне значення для захисту прав інтелектуальної власності та запобігання несанкціонованому розповсюдженню або копіюванню програмного забезпечення та ігор.

Надання оперативної та ефективної підтримки клієнтів має вирішальне значення для задоволення потреб користувачів. Платформа надає різні канали підтримки, такі як електронна пошта або чат на самій платформі, для своєчасного реагування на запити користувачів або технічних проблем.

Нефункціональні вимоги є важливими для забезпечення надійності, продуктивності, безпеки та загальної якості платформи цифрової дистрибуції програмного забезпечення та ігор. Ось кілька важливих нефункціональних вимог:

- продуктивність;
- надійність;
- зручність використання;
- інтерфейс користувача;

Платформа повинна мати інтуїтивно зрозумілий і зручний інтерфейс, який полегшує користувачам перегляд, пошук і знаходження програмного забезпечення

та ігор. Надійна функція пошуку і фільтрування допоможуть користувачам знайти потрібний контент.

Платформа повинна бути спроектована таким чином, щоб витримувати високий трафік і одночасну активність користувачів без значного зниження продуктивності. Вона повинна мати швидкий час відгуку для перегляду, пошуку та завантаження контенту, незалежно від кількості одночасних користувачів.

Платформа повинна бути розроблена з орієнтацією на користувача, з пріоритетом на простоту використання та інтуїтивно зрозумілу навігацію. Чіткі та стислі інструкції, інформативні повідомлення про помилки та доступні елементи дизайну сприяють позитивному користувацькому досвіду.

Розробка моделі предметної області для платформи цифрової дистрибуції передбачає створення структурованого представлення ключових сутностей, атрибутів і зв'язків у предметній області. Ця модель слугує основою для проектування платформи і допомагає гарантувати, що система адекватно охоплює і підтримує необхідні функціональні можливості.

Розпочати потрібно з визначення основних суб'єктів, які є фундаментальними для платформи. Деякі потенційні суб'єкти можуть включати в себе:

- Користувач: Представляє користувачів платформи, включаючи клієнтів, розробників та адміністраторів.
- Продукт: Представляє цифрове програмне забезпечення або ігрові продукти, доступні на платформі.
- Розробник/Видавець: Представник компанії, відповідальний за створення та публікацію програмного забезпечення/ігор.

При потребі змінювати модель на основі відгуків, додаткових вимог або змін у дизайні платформи. Постійно допрацьовувати та вдосконалювати модель предметної області, щоб забезпечити її відповідність потребам платформи, які постійно змінюються.

Розробка платформи цифрової дистрибуції програмного забезпечення та ігор вимагає ретельного планування та врахування різних факторів для створення

бездоганного користувацького досвіду. Платформа повинна задовольняти потреби як розробників, так і користувачів, забезпечуючи ефективне та безпечне середовище для розповсюдження та доступу до цифрового контенту.

2.2 Розробка бізнес моделі

IBM Rational System Architect - це рішення для візуалізації, аналізу та комунікації архітектури підприємства та аналізу бізнес-процесів. Це рішення забезпечує підтримку прийняття рішень, оптимізацію процесів та інтеграцію в процес надання рішень. Rational System Architect охоплює всі аспекти архітектури, включаючи моделювання, публікацію, аналіз і виконання. IBM RSA надає підтримку для створення, візуалізації та роботи з UML-діаграмами (рис. 2.1) [8].

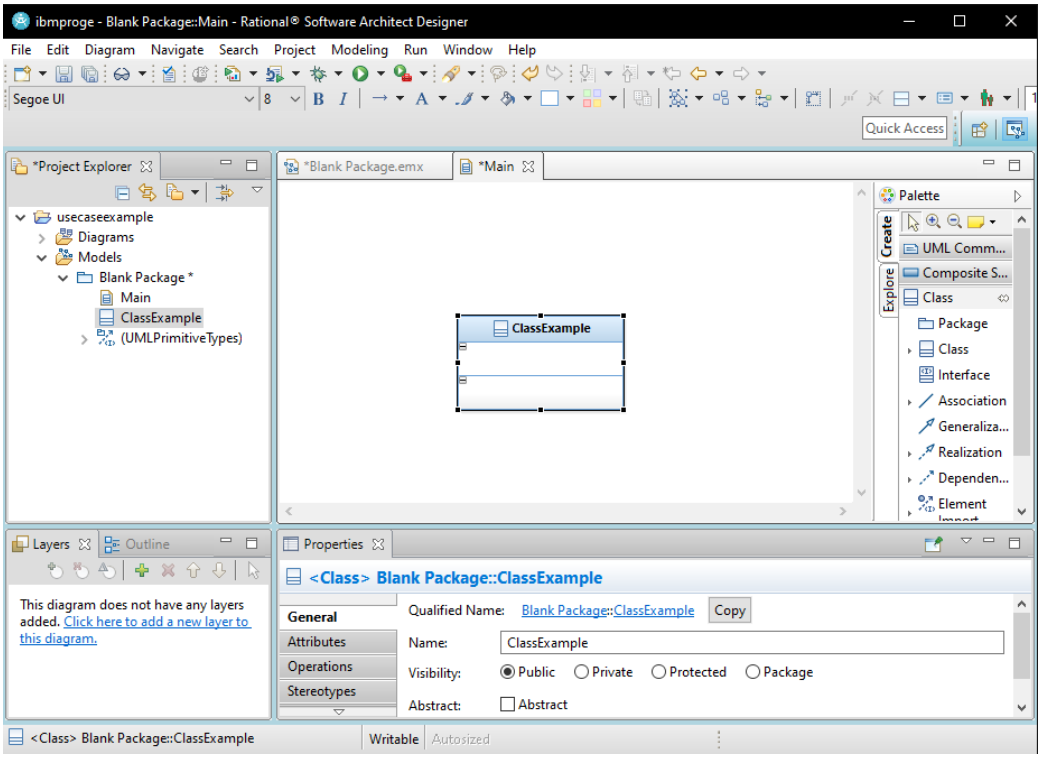

Рисунок 2.1 – Середовище IBM RSA

Діаграма варіантів використання забезпечує візуальне представлення взаємодії між користувачами та системою, демонструючи різні способи, в які система може бути використана для виконання конкретних завдань. У контексті платформи цифрової дистрибуції програмного забезпечення та ігор діаграма зображує ключові функціональні можливості задіяні в роботі платформи. Вона слугує цінним інструментом для розуміння вимог до системи, визначення потреб користувачів і розробки безперебійного користувацького досвіду. Відображаючи потенційні варіанти та їхні взаємозв'язки, діаграма пропонує комплексний огляд додатку [9].

Діаграма ВВ даного додатку була створена за допомогою програми IBM RSA (рис. 2.2).

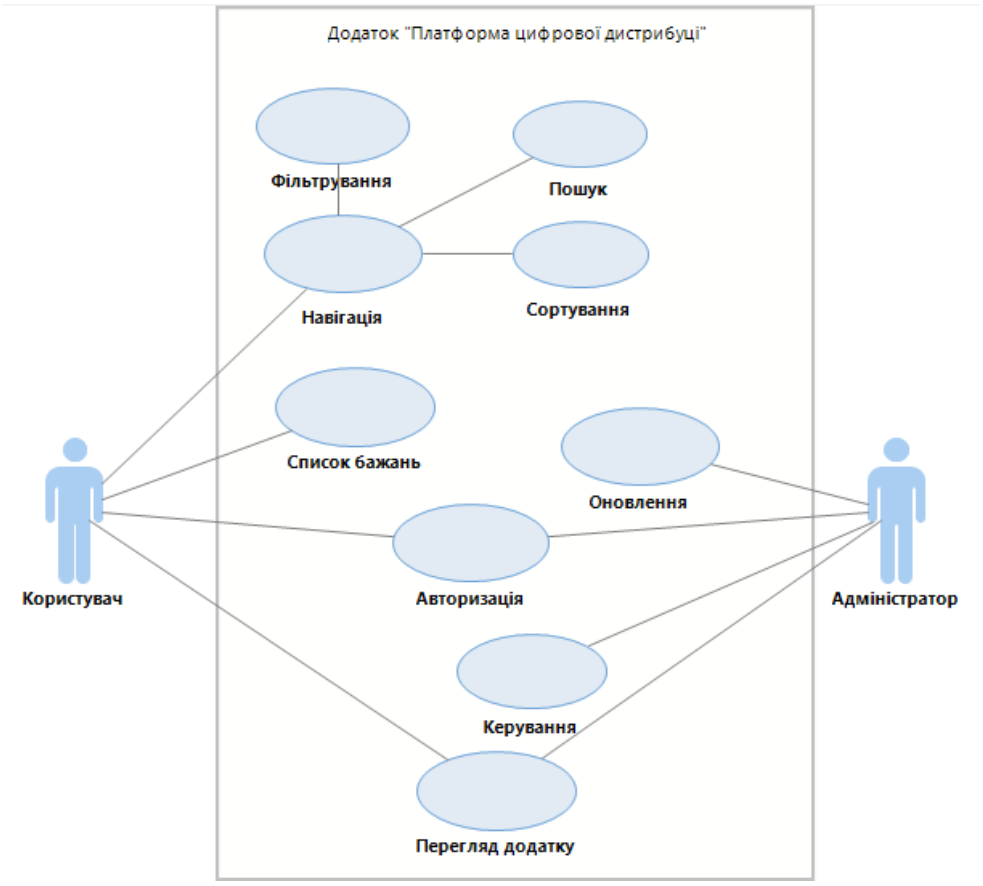

Рисунок 2.2 – Діаграма ВВ додатку

Наступним було сформовано відповідні сценарії опису діаграми варіантів використання.

Опис варіантів використання

Варіант використання "Перегляд додатку"

Короткий опис

Даний варіант використання описує дії користувача з додатком

Основний потік подій

Починає виконуватись після відкриття.

1. Користувач відкриває додаток.
2. Користувач переглядає платформу, бачить інтерфейс.
3. Користувач закриває додаток.

Альтернативний потік подій

1. Користувач натискає на картку з грою або ПЗ.
2. Відкривається вкладка з потрібним продуктом.
3. Користувач переглядає поточну сторінку.
4. Знаходить інформацію, що цікавить.
5. Продовжує користування додатком.

Передумови

Перед початком виконання програма має бути стабільною в роботі.

Післяумови

Якщо все пройшло успішно, платформа працювала без помилок та була зрозумілою користувачу.

Варіант використання "Авторизація"

Короткий опис

Даний варіант використання описує процес автентифікації на сайті.

Основний потік подій

1. Користувач відкриває додаток.
2. Натискає на кнопку "Continue with Facebook".
3. Вводить дані для входу у акаунт Facebook.
4. Програма запитує доступ до даних, користувач натискає кнопку продовжити.
5. Користувач авторизований.

Альтернативний потік подій

1. Програма запитує доступ до даних, користувач відмовляється їх надавати
2. Авторизація не відбувається

Передумови

Має існувати відповідний акаунт Facebook через який можлива автентифікація користувача.

Післяумови

Якщо варіант використання виконано успішно, користувач буде авторизований на сайті.

Варіант використання "Список бажань"

Короткий опис

Даний варіант використання описує процес додавання гри в список бажаного.

Основний потік подій

1. Авторизований користувач відкриває додаток.
2. Знаходить потрібну гру.
3. Натискає символ збереження під назвою гри.
4. Натискає кнопку список бажань вгорі додатку.
5. Відкривається сторінка списку бажань.
6. Користувач бачить всі додані ігри

Альтернативний потік подій

1. Авторизований користувач відкриває сторінку списку бажань
2. Натискає символ збереження на картці гри ще раз
3. Гра більше не перебуває в списку бажань

Передумови

Користувач має бути авторизований або автентифікуватись.

Післяумови

Продукт буде додано або видалено з списку бажань.

Варіант використання "Керування"

Короткий опис

Даний варіант використання описує дії адміністратора щодо керування додатком.

Основний потік подій

1. Один з API більше не діючий.
2. API потрібно оновити.
3. Адміністратор бере новий ключ для потрібного API

Передумови

Інформація, що потребує оновлення.

Післяумови

Успішно відредагована інформація.

Варіант використання "Навігація"

Короткий опис

Даний варіант використання описує процес навігації в додатку.

Основний потік подій

1. Користувач відкриває додаток.
2. Користувач обирає потрібний спосіб зміни списку продуктів.

Передумови

Заповнений список продуктів за допомогою RAWG API.

Післяумови

При успішному використанні, інформацію про продукти на екрані змінить відповідно до маніпуляцій.

Варіант використання "Пошук"

Короткий опис

Даний варіант використання описує процес пошуку продукту в додатку.

Основний потік подій

1. Користувач переглядає додаток.
2. Користувач набирає загальну назву потрібної йому гри.
3. Користувач розпочинає пошук.
4. Додаток виводить список продуктів зі схожою назвою відповідно до заданого пошуку.

Альтернативний потік подій

1. Користувач набирає точну назву продукту.
2. Додаток виводить потрібний продукт в першому рядку списку.

Передумови

Користувач повинен знати назву продукту, що він шукає.

Післяумови

При успішному виконанні, додаток виведе картку продукту.

Варіант використання "Фільтрування"

Короткий опис

Даний варіант використання описує процес фільтрування продуктів в додатку.

Основний потік подій

1. Користувач переглядає додаток.
2. Користувач натискає кнопку "фільтр".
3. Обирає потрібні жанри для пошуку гри.
4. Додаток виводить ігри з відповідними жанрами.

Альтернативний потік подій

1. Користувач набирає натискає кнопку "фільтр".
2. Прокручує меню фільтра вниз
3. Обирає потрібну платформу з чотирьох присутніх
4. Додаток виводить ігри, що присутні на відповідній плафтормі

Передумови

Додаток має бути відкритим у користувача.

Післяумови

При успішному виконанні, відбудеться фільтрування продуктів відповідно до запиту.

Варіант використання "Сортування"

Короткий опис

Даний варіант використання описує процес сортування продуктів додатку.

Основний потік подій

1. Користувач переглядає додаток.
2. Користувач натискає кнопку "сортувати за".
3. Користувач обирає сортування від вищої оцінки до нижчої.
4. Додаток виводить картки продуктів відповідно до запиту.
5. Користувач бачить ігри з найвищими оцінками на початку.

Альтернативний потік подій

1. Користувач натискає кнопку "сортувати за".
2. Користувач обирає сортування від нижчої оцінки до вищої.
3. Додаток виводить картки продуктів відповідно до запиту.
4. Користувач бачить ігри з найнижчими оцінками на початку.

Передумови

Відповідна кнопка повинна працювати, як треба.

Післяумови

При успішному виконанні, відбудеться сортування продуктів відповідно до запиту.

Варіант використання "Оновлення"

Короткий опис

Даний варіант використання описує дії адміністратора з оновленнями

Основний потік подій

1. Адміністратор розпочинає роботу додатку.
2. Оновлює застарілу або непотрібну інформацію у файлах додатку.

Альтернативний потік подій

1. Успішна робота адміністратора.
2. Інформація не потребує оновлення.

Передумови

Інформація, що потребує оновлення.

Післяумови

Успішно відредагована інформація.

2.3 Проєктування архітектури

Діаграма класів відображає статичний вигляд програми. Вона представляє типи об'єктів, що знаходяться в системі, та взаємозв'язки між ними. Клас складається зі своїх об'єктів, а також може успадковуватися від інших класів. Діаграма класів використовується для візуалізації, опису, документування різних аспектів системи, а також для побудови виконуваного програмного коду [9].

Основне призначення діаграм класів - створення статичного вигляду програми. Це єдина діаграма, яка широко використовується для побудови, і її можна відобразити за допомогою об'єктно-орієнтованих мов. Це одна з найпопулярніших діаграм UML [9].

Діаграма класів даного додатку надає вичерпний огляд архітектури програмного забезпечення та його компонентів. CEF, заснований на проекті Chromium з відкритим вихідним кодом, є потужним фреймворком, який дозволяє розробникам вбудовувати повнофункціональний веб-браузер у свої додатки.

Діаграму класів для даного додатку було побудовано в середовищі IBM RSA (рис. 2.3)

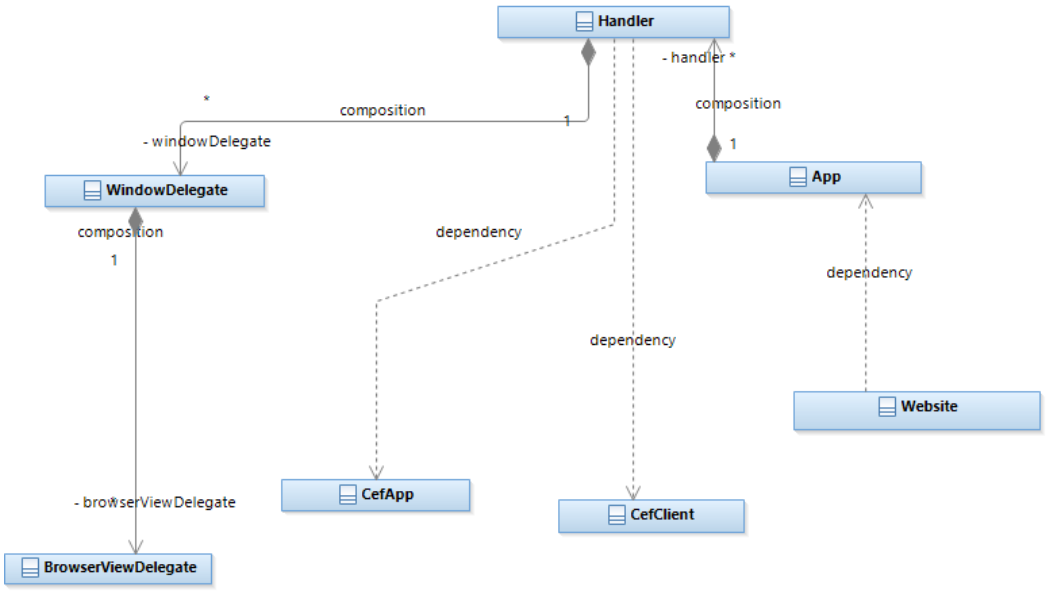

Рисунок 2.3 – Діаграма класів додатку

Клас Handler:

1. Успадковує від стандартних класів бібліотеки CEF CefClient, CefDisplayHandler, CefLifeSpanHandler та CefLoadHandler.
2. Являє собою обробник для зворотних викликів на рівні додатку.
3. Містить функції-члени, такі як OnTitleChange, OnAfterCreated, DoClose та OnLoadError, які обробляють різні події, пов'язані з роботою додатку.
4. Надає статичні функції-члени, такі як GetInstance та IsChromeRuntimeEnabled.

Клас App:

1. Успадковує від стандартних класів бібліотеки CEF CefApp та CefBrowserProcessHandler.
2. Являє собою зворотні виклики на рівні додатку для його процесу.
3. Містить функції-члени, такі як OnContextInitialized та GetDefaultClient, які обробляють ініціалізацію програми та створення браузера.

Клас Website:

1. Представляє собою вебсайт з цифровою дистрибуцією програмного забезпечення та ігор. Але клас Website не є однозначною частиною CEF. Браузер CEF надає фреймворк для вбудовування вебсайту у додаток.
2. Клас Website є додатковим класом для представлення вебсайту з функцією цифрового розповсюдження.
3. Містить функції такі, як showGames, showScore, showPlatformsCard, getPlatforms та інші.

Клас WindowDelegate:

1. Успадковує від стандартного класу CefWindowDelegate.
2. Забезпечує реалізацію делегату для CefWindow, у якому розміщено додаток.
3. Містить функції-члени, такі як OnWindowCreated, OnWindowDestroyed та CanClose, які обробляють події, пов'язані з вікном додатку.

Клас BrowserViewDelegate:

1. Успадковує від CefBrowserViewDelegate.
2. Забезпечує реалізацію делегату для CefBrowserView, який містить спливаючий браузер.
3. Містить функцію-член OnPopupBrowserViewCreated, яка обробляє створення спливаючих вікон.

Діаграма послідовності зображує візуальне представлення взаємодії та потоку подій на платформі цифрової дистрибуції. Вона пропонує комплексний огляд того, як різні компоненти, такі як користувач, сторона CEF та вебсайту, співпрацюють, щоб сприяти безперешкодній роботі. Діаграма послідовності дає змогу глибше зрозуміти основні процеси додатку.

Для початку було створено діаграму послідовності для сценарію ВВ "Перегляд додатку" (рис. 2.4).

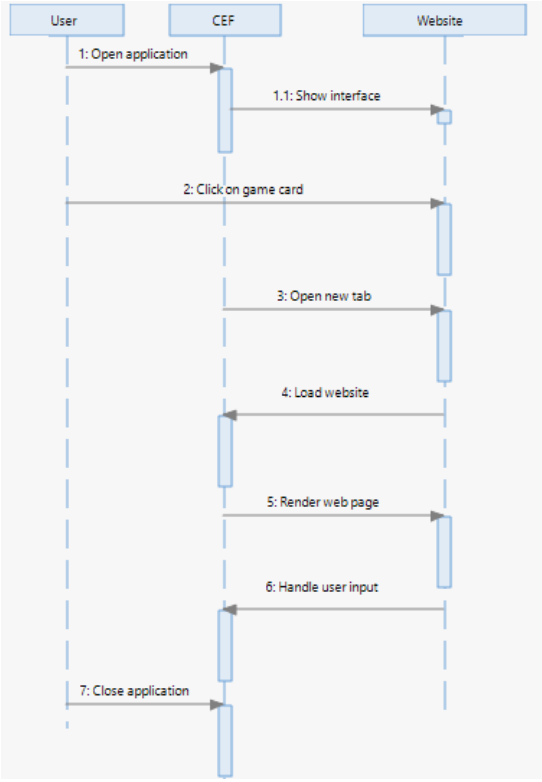

Рисунок 2.4 – Діаграма послідовності сценарію ВВ "Перегляд додатку"

Далі створено діаграму послідовності для сценарію ВВ "Пошук" (рис. 2.5).

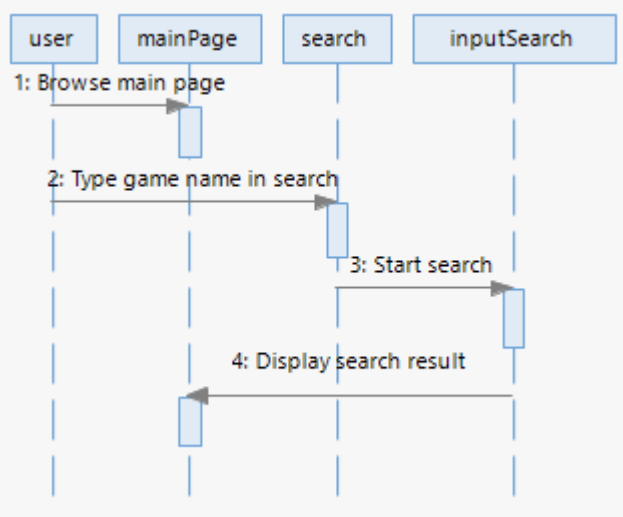

Рисунок 2.5 – Діаграма послідовності сценарію ВВ "Пошук"

Третьою та останньою було створено створено діаграму послідовності на основі сценарію ВВ "Сортування" (рис. 2.6).

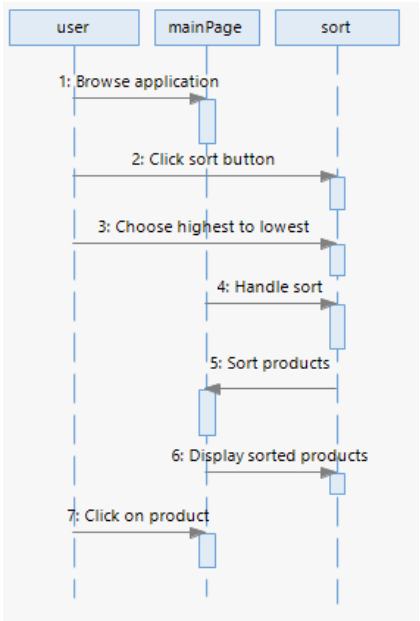

Рисунок 2.6 – Діаграма послідовності сценарію ВВ "Сортування"

Так було створено та оформлено діаграму класів додатку та діаграми послідовності на основі сценарііїв варіантів використання.

3 КОНСТРУЮВАННЯ ДОДАТКУ

3.1 Реалізація ключових класів

Впровадження ключових класів у додаток є важливим кроком на шляху до створення безперебійного та ефективного користувацького досвіду. У цьому параграфі буде розглянуто деякі з фундаментальних класів, які формують основу надійної платформи цифрової дистрибуції, закладаючи фундамент для успішного додатку.

Класи даного додатку були реалізовані за допомогою середовища Microsoft Visual Studio 2022.

Клас Handler відповідає за обробку різних подій і взаємодій всередині програми. Цей клас є похідним від декількох класів-обробників CEF (Chromium Embedded Framework), зокрема CefClient, CefDisplayHandler, CefLifeSpanHandler та CefLoadHandler. Він слугує мостом між логікою програми та базовою функціональністю браузера. Клас Handler визначає такі методи, як OnAfterCreated, OnBeforeClose та OnLoadEnd для обробки певних подій протягом життєвого циклу браузер.

Він забезпечує реалізацію різних методів-обробників, що використовуються у фреймворку CEF, таких як обробка змін заголовків, створення, закриття браузера та помилок завантаження. Він також містить додаткові методи для керування вікнами браузера.

Детальніше:

- Керує вікнами браузера, обробляє події, пов'язані зі зміною заголовків, створенням, закриттям браузера та обробкою помилок.
- Містить список наявних вікон браузера (browser_list_).
- Надає такі методи, як OnTitleChange, OnAfterCreated, DoClose, OnBeforeClose, OnLoadError, CloseAllBrowsers та IsChromeRuntimeEnabled.

- Ключову частину коду реалізації класу Handler наведено на рисунку 3.1, а решта у файлі handler.h, який можна знайти в додатку А.

```
class SimpleHandler : public CefClient,
                      public CefDisplayHandler,
                      public CefLifeSpanHandler,
                      public CefLoadHandler {
public:
  explicit SimpleHandler(bool use_views);
  ~SimpleHandler();

  // Provide access to the single global instance of this object.
  static SimpleHandler* GetInstance();

  // CefClient methods:
  virtual CefRefPtr<CefDisplayHandler> GetDisplayHandler() override {
    return this;
  }
  virtual CefRefPtr<CefLifeSpanHandler> GetLifeSpanHandler() override {
    return this;
  }
  virtual CefRefPtr<CefLoadHandler> GetLoadHandler() override { return this; }

  // CefDisplayHandler methods:
  virtual void OnTitleChange(CefRefPtr<CefBrowser> browser,
                             const CefString& title) override;
```

Рисунок 3.1 – Ключовий код класу Handler

Клас App представляє сам додаток. Він відповідає за ініціалізацію та налаштування фреймворку CEF. Клас App успадковує від класу CefApp і перевизначає декілька методів для забезпечення користувацької поведінки під час фаз ініціалізації та завершення роботи програми.

Він представляє зворотні виклики на рівні програми для процесу браузера. Він реалізує метод OnContextInitialized, який викликається при ініціалізації контексту CEF, і метод GetDefaultClient, який повертає клієнта за замовчуванням для браузера.

Детальніше:

- Успадковується від CefApp та CefBrowserProcessHandler.
- Реалізовує методи GetBrowserProcessHandler. OnContextInitialized та GetDefaultClient.
- Створює екземпляр браузера та встановлює його налаштування.

- Обробляє створення вікон браузера з використанням фреймворку Views або нативного фреймворку платформи.
- Ключову частину коду реалізації класу App наведено на рисунку 3.2, а решту можна знайти в додатку А.

```
class App : public CefApp, public CefBrowserProcessHandler {
public:
 App();

 // CefApp methods:
 CefRefPtr<CefBrowserProcessHandler> GetBrowserProcessHandler() override {
  return this;
 }

 // CefBrowserProcessHandler methods:
 void OnContextInitialized() override;
 CefRefPtr<CefClient> GetDefaultClient() override;
```

Рисунок 3.2 – Ключовий код класу App

Клас WindowDelegate слугує делегатом для керування поведінкою та взаємодією головного вікна програми. Цей клас є похідним від CefWindowDelegate і використовується при використанні фреймворку Views, Клас WindowDelegate перевизначає такі методи, як OnWindowCreated, OnWindowDestroyed та OnPaint для обробки специфічних подій, пов'язаних з вікном. Він забезпечує реалізацію делегатів для CefWindow, у якому розміщено додаток на основі Views.

Детальніше:

- Реалізовує інтерфейс CefWindowDelegate.
- Надає методи делегатів для CefWindow, що містить додаток на основі Views.
- Обробляє події створення та закриття вікна.
- Код реалізації WindowDelegate знаходиться в анонімному просторі імен.
- Ключову частину коду реалізації класу WindowDelegate наведено на рисунку 3.3, а решту можна знайти в додатку А.

```
class WindowDelegate : public CefWindowDelegate {
public:
 explicit WindowDelegate(CefRefPtr<CefBrowserView> browser_view)
     : browser_view_(browser_view) {}

 void OnWindowCreated(CefRefPtr<CefWindow> window) override {
  // Add the browser view and show the window.
  window->AddChildView(browser_view_);
  window->Show();

  // Give keyboard focus to the browser view.
  browser_view_->RequestFocus();
 }

 void OnWindowDestroyed(CefRefPtr<CefWindow> window) override {
  browser_view_ = nullptr;
 }

 bool CanClose(CefRefPtr<CefWindow> window) override {
  // Allow the window to close if the browser says it's OK.
  CefRefPtr<CefBrowser> browser = browser_view_->GetBrowser();
  if (browser)
    return browser->GetHost()->TryCloseBrowser();
  return true;
 }
```

Рисунок 3.3 – Ключовий код класу WindowDelegate

Клас BrowserViewDelegate діє як делегат для браузерного представлення програми. Він визначає такі методи, як GetContextMenuHandler, OnBrowserCreated та OnBrowserDestroyed для обробки специфічних подій, пов'язаних з браузерним представленням.

Клас BrowserViewDelegate є похідним від CefBrowserViewDelegate, який є частиною CEF API. Він надає реалізацію делегата для обробки створення спливаючих вікон браузера.

Детальніше:

- Реалізує інтерфейс CefBrowserViewDelegate.
- Надає методи делегатів для створення спливаючих вікон браузера.
- Створює нові вікна верхнього рівня для спливаючих браузерів.
- Код реалізації BrowserViewDelegate знаходиться в анонімному просторі імен.
- Ключову частину коду реалізації класу BrowserViewDelegate наведено на рисунку 3.4, а решту можна знайти в додатку А.

```
class BrowserViewDelegate : public CefBrowserViewDelegate {
public:
 BrowserViewDelegate() {}

 bool OnPopupBrowserViewCreated(CefRefPtr<CefBrowserView> browser_view,
                                CefRefPtr<CefBrowserView> popup_browser_view,
                                bool is_devtools) override {
  // Create a new top-level Window for the popup. It will show itself after
  // creation.
  CefWindow::CreateTopLevelWindow(
      new SimpleWindowDelegate(popup_browser_view));

  //  created the Window.
  return true;
 }
```

Рисунок 3.4 – Ключовий код класу BrowserViewDelegate

Якщо підсумувати, то клас Handler виступає в ролі головного обробника додатку, заповнюючи прогалину між кодом і вбудованим браузером. Він дозволив налаштувати поведінку браузера, перевизначаючи такі методи, як OnAfterCreated та OnBeforeClose. Клас App ініціалізував та керує фреймворком CEF, виконуючи такі завдання, як ініціалізація, аргументи командного рядка. Клас WindowDelegate обробляє події, пов'язані з тривалістю життя вікна браузера, контролюючи створення, закриття тощо. Клас BrowserViewDelegate перехоплює та модифікує запити браузера за допомогою методів OnBeforeBrowse та OnBeforeResourceLoad. Ці класи надають гнучкість у керуванні та налаштуванні поведінки вбудованого браузера у створеному додатку.

Також стандартні класи бібліотеки CEF відіграли вирішальну роль в інтеграції CEF у додаток. Ці класи дозволили налаштовувати поведінку браузера, перевизначаючи певні методи та надаючи власні реалізації.

3.2 Розробка GUI

Розробка графічного інтерфейсу відіграє ключову роль розробці додатку. Він є важливим аспектом створеного додатку. Завдяки цьому існує інтуїтивно зрозумілий та зручний досвід для споживачів. Графічний інтерфейс користувача

слугує сполучною ланкою між платформою та користувачем, забезпечуючи безперешкодну навігацію та взаємодію з додатком.

Для створення графічного інтерфейсу було розроблено макети основних компонентів для подальшої розробки додатку.

Для початку був створений макет головної сторінки додатку, що користувачі будуть бачити прямо при відкритті. Визначено ключові елементи:

- Лого;
- Пошук;
- Кнопка “Меню”;
- Кнопка “Авторизації”;
- Кнопка “Фільтр”;
- Кнопка “Сортування”;
- Картки з продуктами;

Розроблений макет зображено на рисунку 3.5.

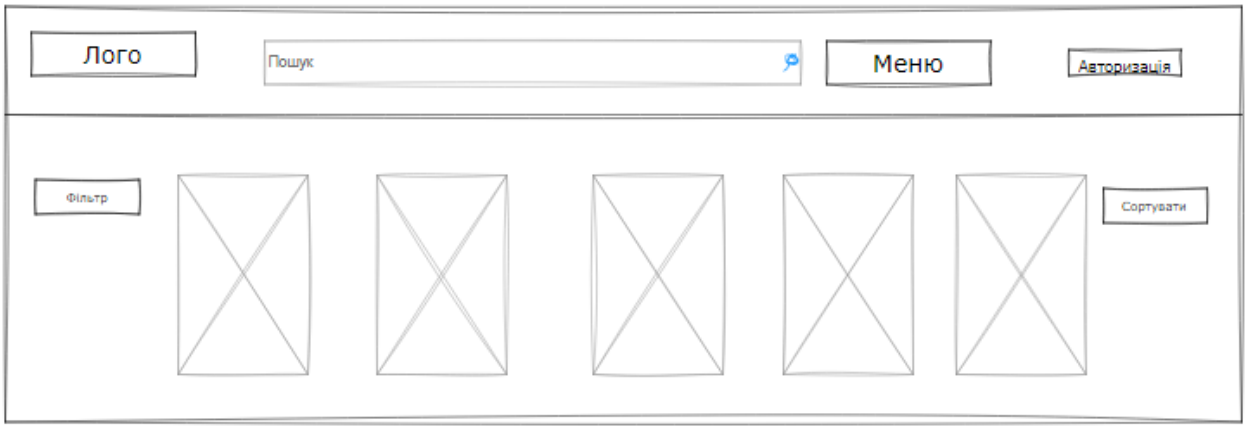

Рисунок 3.5 – Макет головної сторінки

Наступним макетом стала кнопка “Фільтр” за допомогою якої користувачі можуть фільтрувати продукти по жанрам та платформам. Визначено її ключові елементи:

- Меню кнопки
- Заголовок “Жанри” та їх список

- Заголовок “Платформа” та її список

Розроблений макет зображено на рисунку 3.6

Рисунок 3.6 – Макет кнопки “Фільтр”

Далі створений макет кнопки “Сортування” за допомогою якої користувач має можливість сортувати продукти (рис. 3.7)

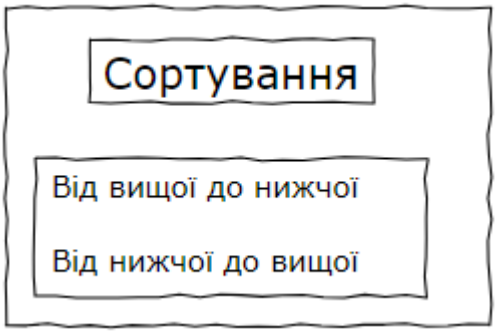

Рисунок 3.7 – Макет кнопки “Сортування”

Після цього, розроблений макет сторінки “Список бажань” (рис. 3.8), де міститись продукти, що були збережені користувачем. Макет містить такі ключові деталі:

- кнопка “Меню”;
- картки збережених продуктів;

- стрілки "Назад" та "Вперед".

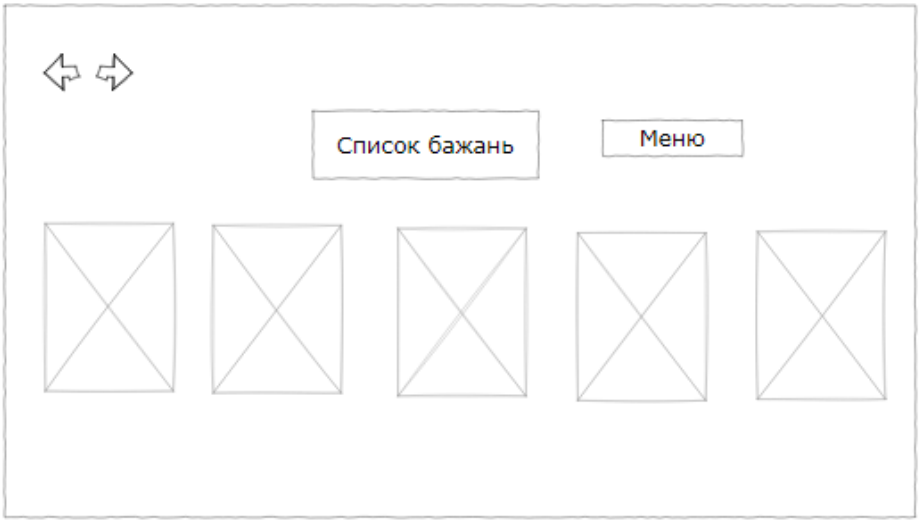

Рисунок 3.8 – Макет сторінки “Список бажань”

Наступним в черзі стала сторінка “Зв’язатись”, де користувач може побачити місцеперебування офісу компанії та контактну інформацію. В ньому є такі ключові деталі:

- карта з точкою, де перебуває офіс компанії;
- контактні дані (електронна пошта, номер телефону);
- колонка, де можна описати свою проблему та відправити адміністратору.

Макет присутній на рисунку 3.9

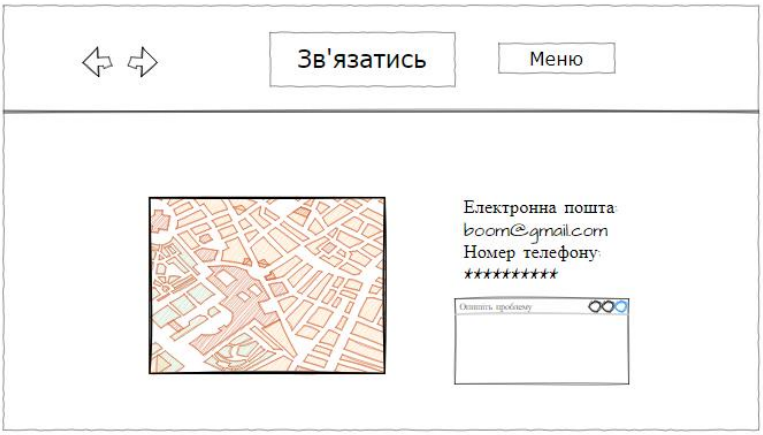

Рисунок 3.9 – Макет сторінки “Зв’язатись”

Після було створено макет картки продукту (рис. 3.10), як він буде виглядати, приблизний розмір, такі характеристики:

- фото продукту;
- назва;
- середній рейтинг;
- жанр;;
- дата випуску
- платформи, де продукт присутній.

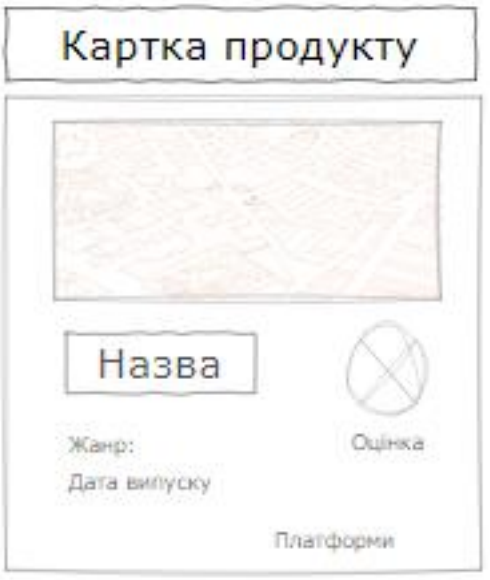

Рисунок 3.10 – Макет картки продукту

Останнім було створено один з головних макетів додатку, це сторінка продукту (рис. 3.11), де користувач бачить назву, опис та деталі, такі, як:

- жанр
- дата випуску
- платформи на яких присутній продукт
- зображення програми
- посилання де можна продукт купити

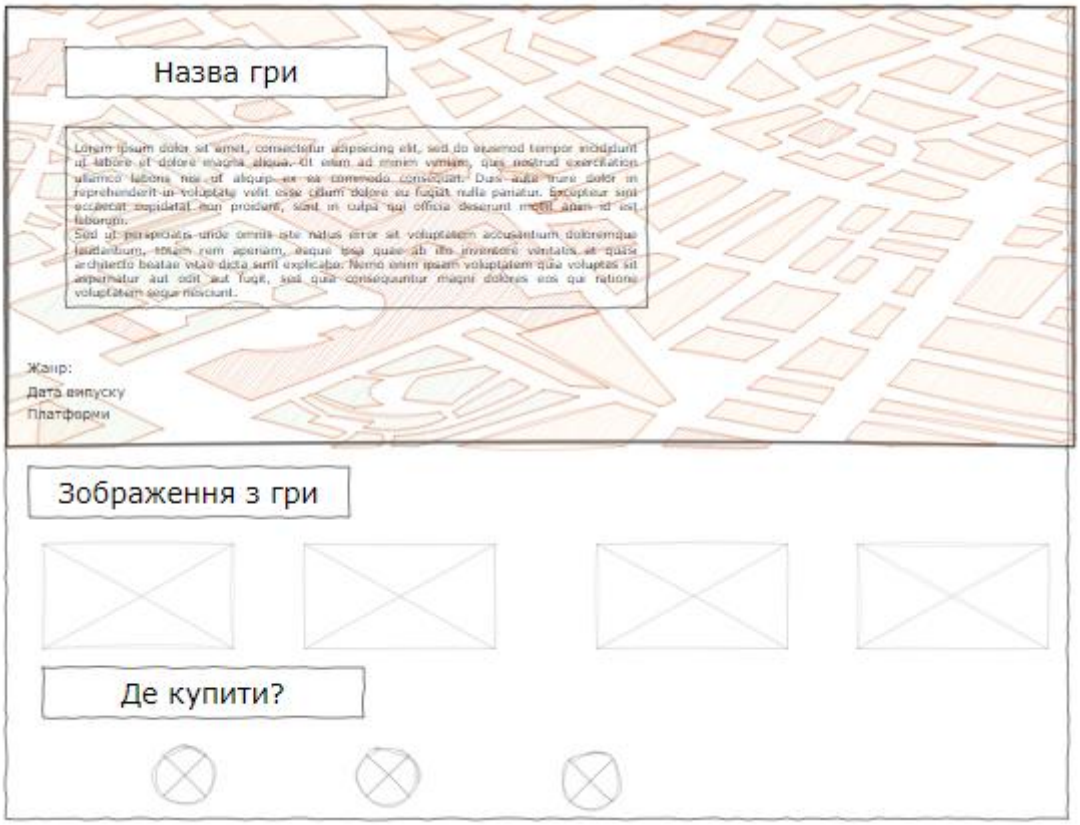

Рисунок 3.11 – Макет сторінки продукту

3.3 Тестування, оцінка якості та результат розробки ПЗ

Розроблений додаток “Onirique”, що є платформою-посередник цифрової дистрибуції та програмного забезпечення є інтегрованим вебдодатком у власний, окремий браузер. Додаток став цілісним, що працює на внутрішніх посиланнях та може опрацьовувати лише потрібний продукт.

Роботу додатку було перевірено на базі платформ:

- Windows 10
- Windows 11

Роботу програми можна оцінити, як стабільну, інтерфейс зрозумілим, проблеми відсутні, функціонал робочий

Сама база всіх продуктів була створена на сайті за допомогою RAWG API. Це найбільша база даних і найкращий API для реалізації потрібних продуктів для у додатку.

Наступним, додаток було запущено, де одразу на головній сторінці видно ось такі ключові елементи:

- лого;
- меню;
- фільтр;
- сортування;
- авторизація через Facebook;
- картки продуктів.

Головну сторінку зображено на рисунку 3.12

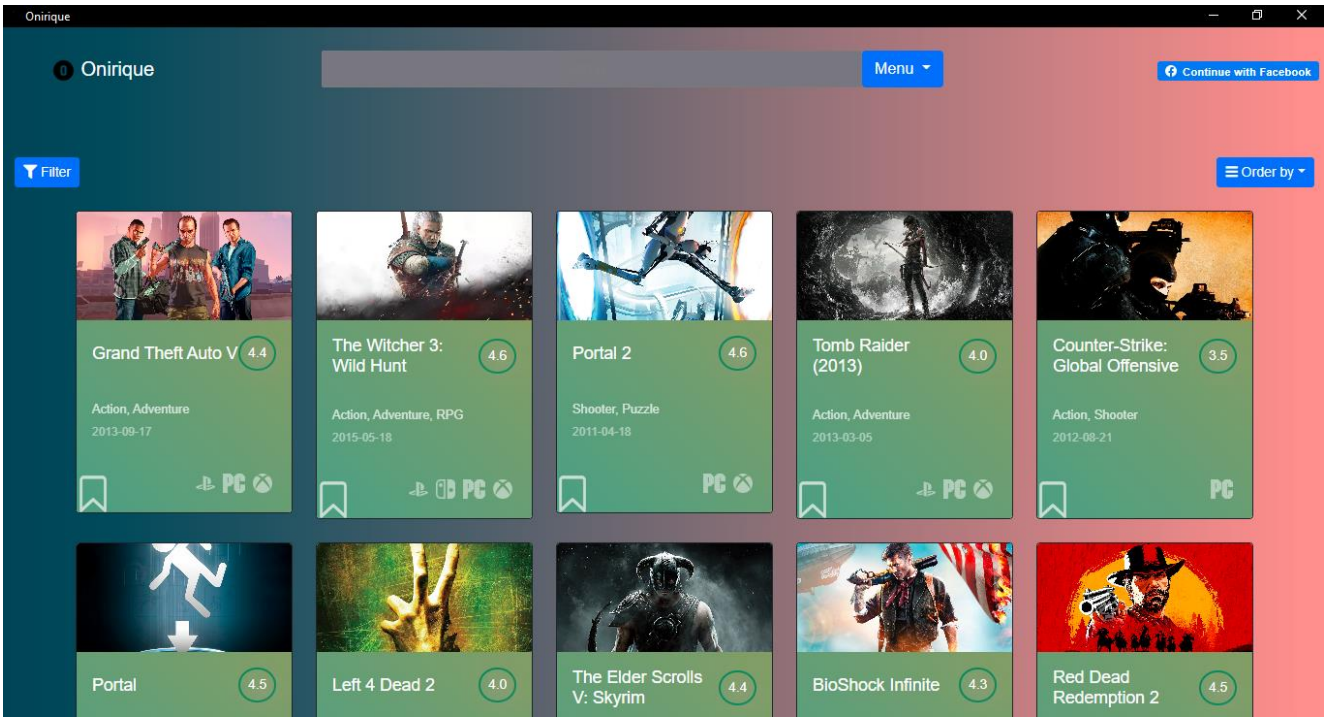

Рисунок 3.12 – Головна сторінка додатку

Далі, якщо йти за порядком, необхідно розглянути створену картку продукту. На ній можна побачити ось такі ключові речі:

- зображення гри;

- назву;;
- її жанри
- рік випуску;
- середній рейтинг;
- платформи, на яких продукт присутній;
- кнопка, за допомогою якої, можна додати гру в список бажань.

Картку продукту зображено на рисунку 3.13

Рисунок 3.13 – Картка продукту

Після цього, потрібно розглянути кнопку (рис. 3.14), що допомагає сортувати продукти за критеріями:

- “Високого балу”, тобто після сортування будуть відображені ігри, що мають найвищий середній рейтинг серед всіх.
- Також за критерієм “Низького балу”, тобто додаток покаже продукти з найнижчим рейтингом або взагалі нулем.
- А також критерій актуальності, що покаже найпопулярніші ігри на даний момент. Головна сторінка завжди сортується за цим.

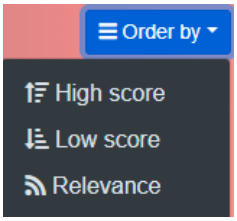

Рисунок 3.14 – Кнопка сортування

На рисунку 3.15 зображено сортування продуктів за найвищим середнім рейтингом.

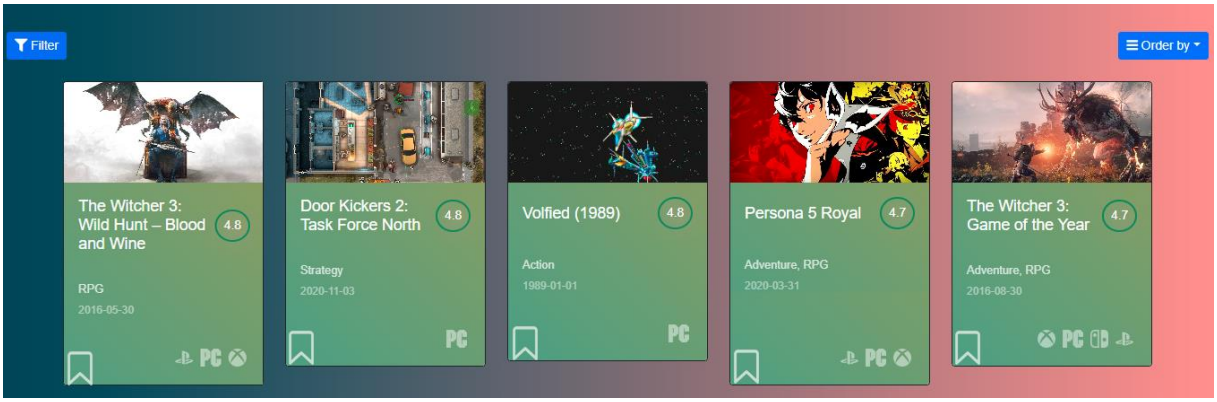

Рисунок 3.15 – Сортування за критерієм “Високого балу

На рисунку 3.16 зображено сортування продуктів за найнижчим середнім рейтингом, в цьому випадку – нулем.

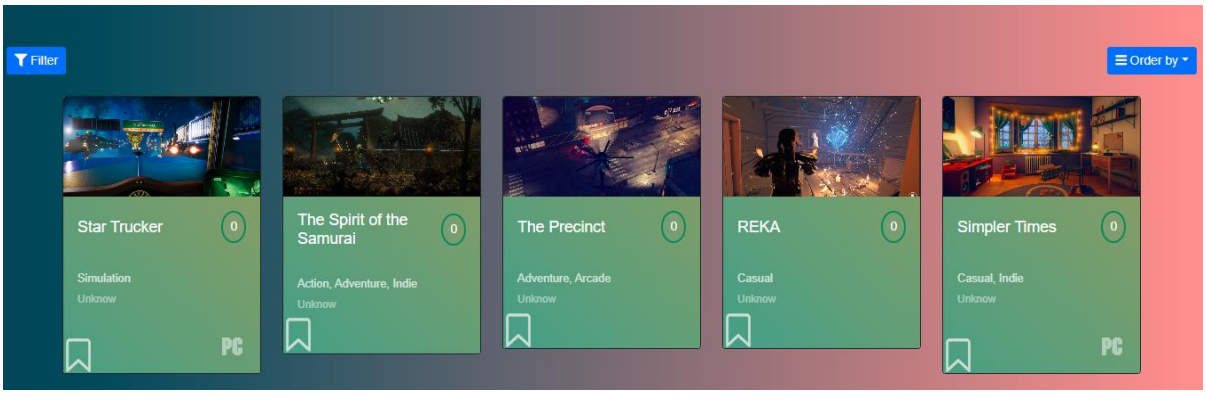

Рисунок 3.16 – Сортування за критерієм “Високого балу

Далі потрібно розглянути автентифікацію на сайті, це можна зробити, при умові, що в користувача існує акаунт Facebook. Створено це було за допомогою

кабінету розробника Facebook та Facebook Authentication API, за допомогою якого авторизація стала можливою. На головній сторінці (див. рис. 3.9) можна побачити кнопку “Continue with Facebook”, після взаємодії з якої відкриється діалогове вікно, де потрібно ввести свій логін та пароль. А вже після цього, додаток попросить дозволи на використання даних користувача (рис. 3.17)

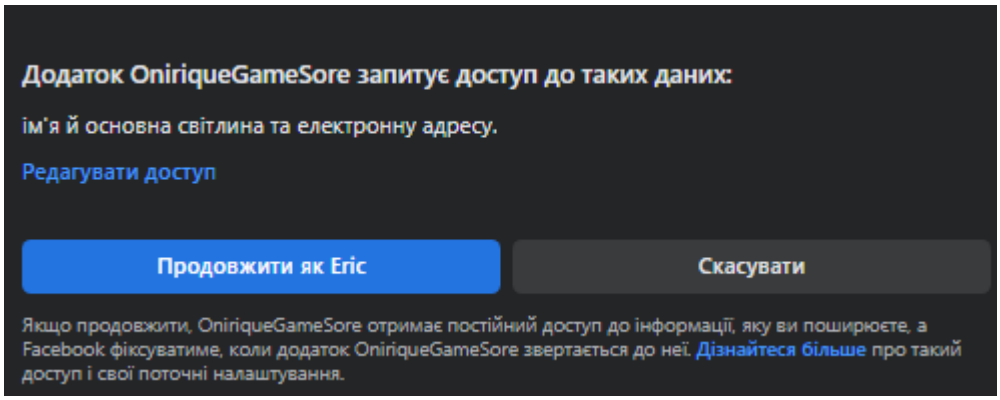

Рисунок 3.17 – Автентифікація через Facebook

Також необхідно розглянути роботу фільтра в додатку. Користувач має можливість фільтрувати продукти за двома категоріями:

- жанр;
- платформа.

Однією з ключових функцій, яка покращує користувацький досвід додатку, є можливість фільтрувати за жанром. Завдяки їх широкому спектру від екшенів і пригод до стратегій і рольових ігор, можливість фільтрувати ігри на основі особистих уподобань дає користувачам можливість легко знаходити і досліджувати свої улюблені жанри.

Після натискання на кнопку фільтра, можна побачити два списки (рис. 3.18)

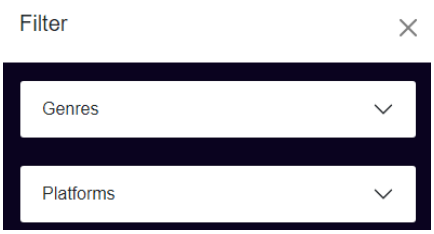

Рисунок 3.18 – Кнопка фільтрування

Результат фільтрування за жанром можна побачити на рисунку 3.19

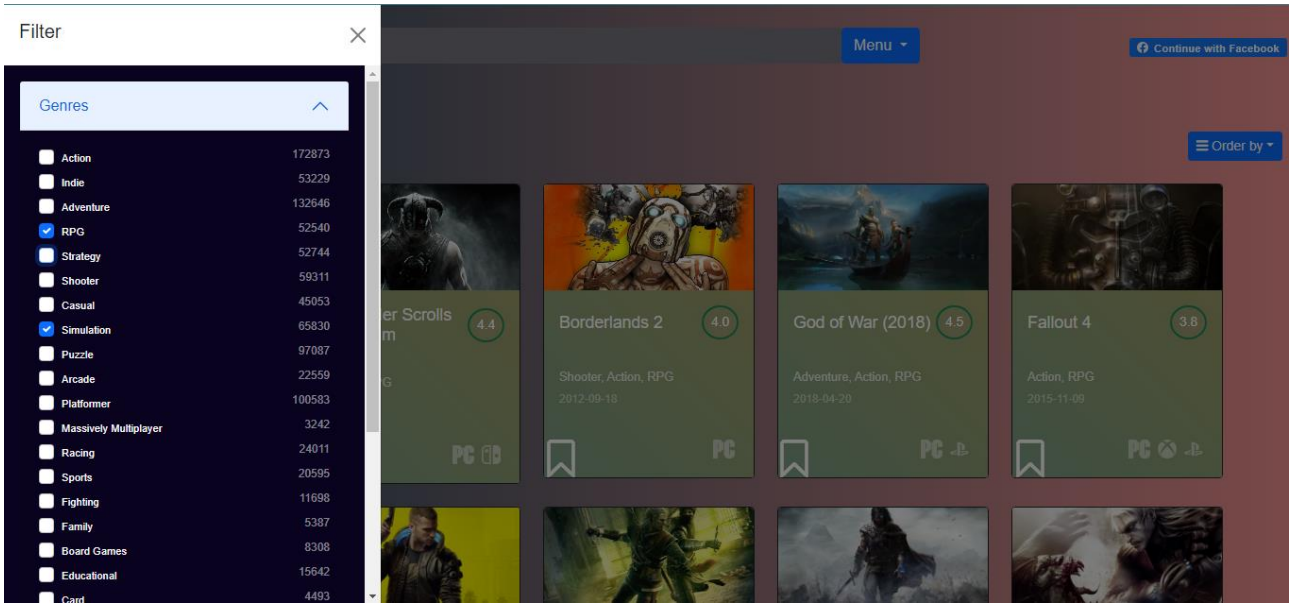

Рисунок 3.19 – Фільтрування за жанром

На додаток до фільтрації ігор за жанрами, ще однією важливою функцією, яка збагачує досвід користування ігровим веб-сайтом, є можливість фільтрувати ігри за платформами. Фільтр за платформою дозволяє звузити пошук і знайти ігри, оптимізовані для конкретної ігрової платформи. Ця функція спрощує шлях користувача (рис. 3.20).

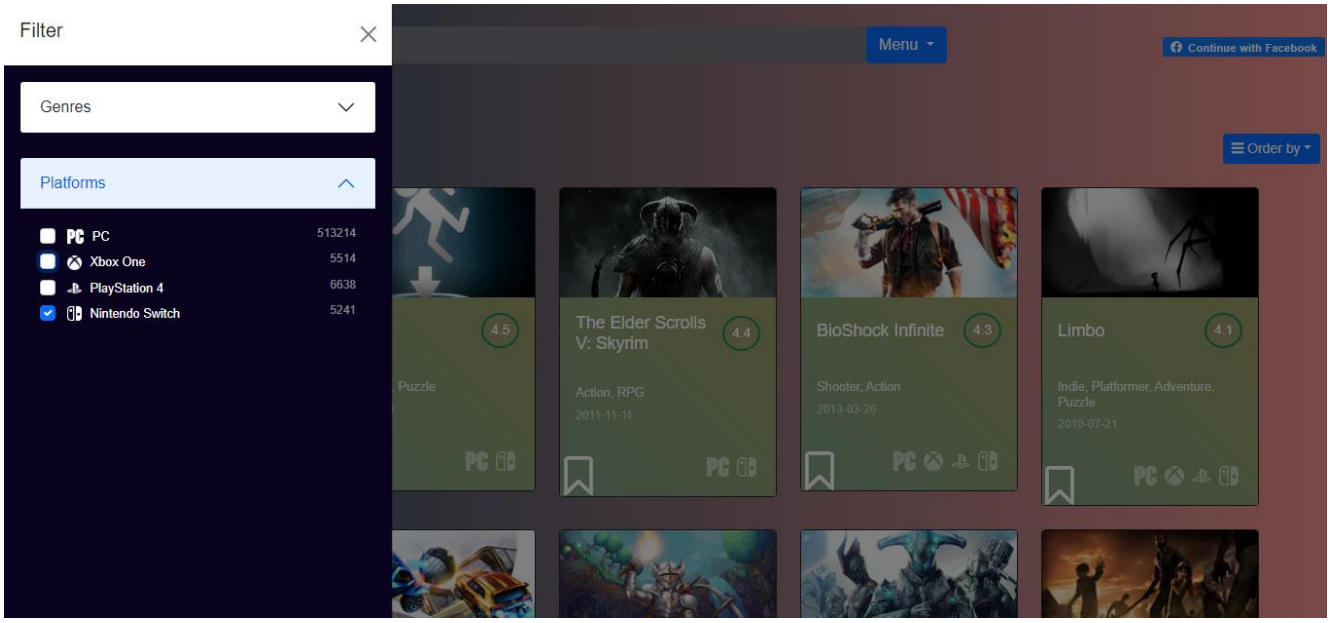

Рисунок 3.20 – Фільтрування за платформою

Функція пошуку на ігровому веб-сайті є цінним інструментом, що дозволяє користувачеві швидко та ефективно знаходити ігри, які він шукає. Завдяки великій базі даних взятій з RAWG API функція пошуку спрощує користувачеві шлях, надаючи інтуїтивно зрозумілий спосіб навігації у великому каталозі ігор.

Користувач може вводити в пошуковий рядок певні ключові слова, такі як назви ігор. Алгоритм пошуку сканує базу даних і надає результати, відображаючи ігри, які відповідають або близькі до введених пошукових термінів (рис. 3.21).

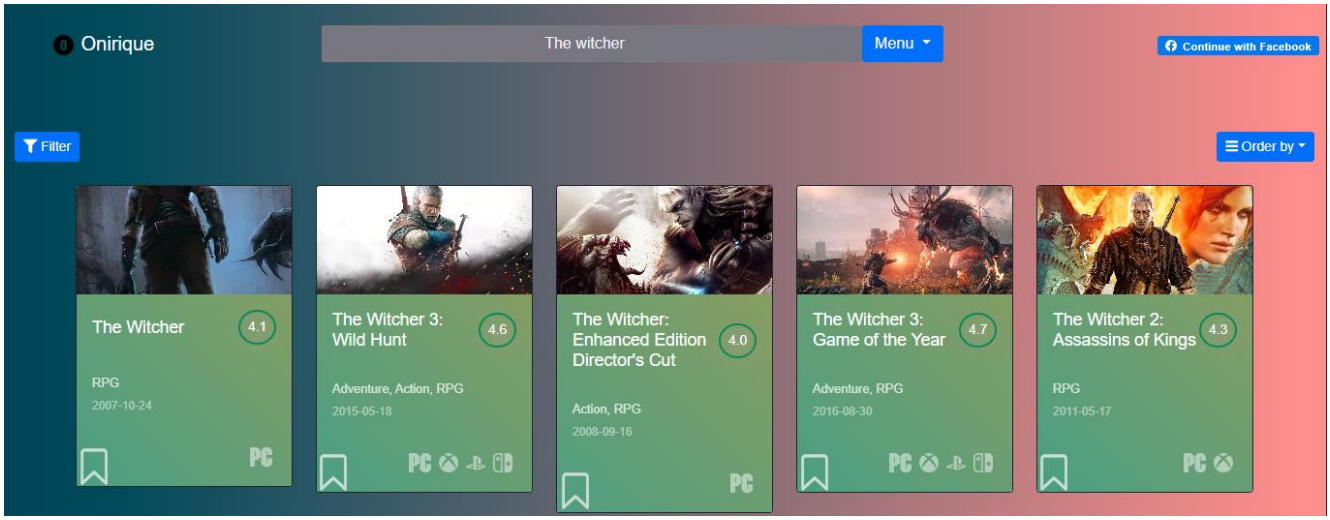

Рисунок 3.21 – Демонстрація роботи пошуку

Наступним потрібно розглянути, що відбувається, коли користувач натисне на одну з карток, в цьому випадку – Terraria (рис. 3.22). Одразу після відкриття, користувача зустрічає:

- Зображення гри, зазвичай ця сторінка містить ключову картинку з гри, яка є її обкладинкою.
- Назва гри, розташоване на зображенні.
- Під назвою гри можна побачити короткий опис сюжету.
- Жанри, які досить точно описують належність гри до певного сетингу.
- Дата виходу.
- Платформи, на яких гра присутня .

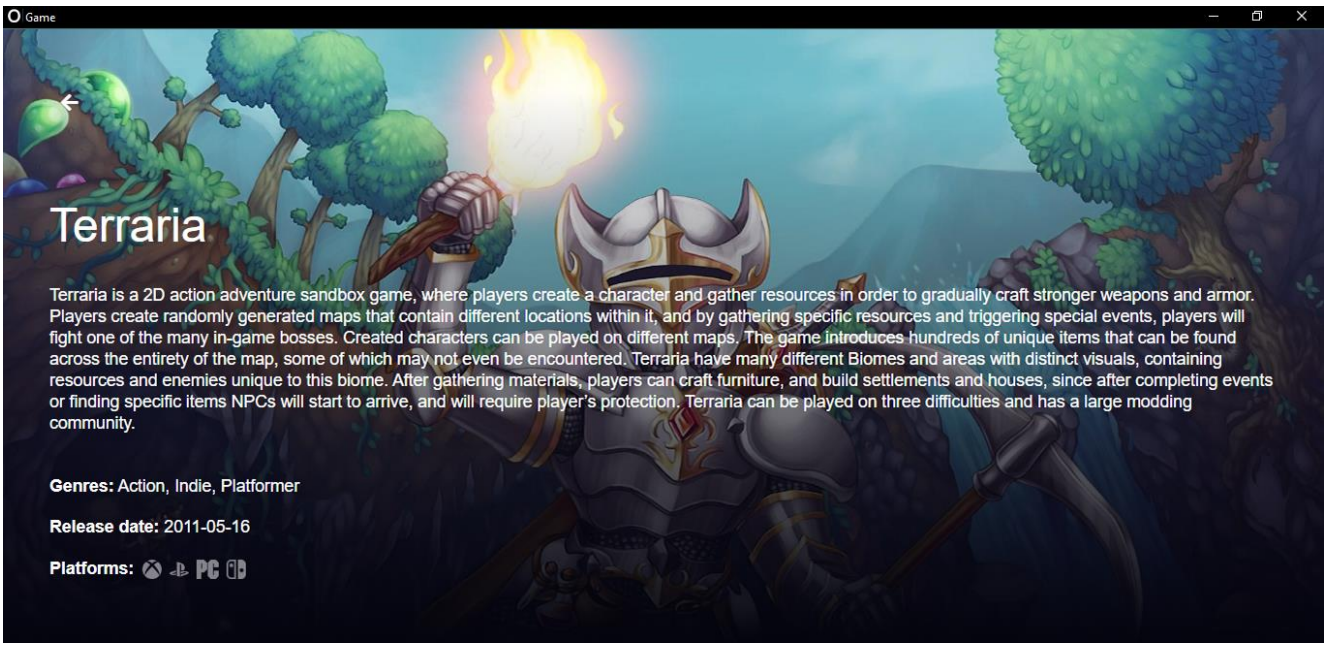

Рисунок 3.22 – Перша половина сторінки продукту

В другій половині сторінки зазвичай можна побачити зображення, які були взяті з ігроладу гри, а також посилання на магазини різних платформ, де можливо офіційно купити гру в прямого представника (рис. 3.23).

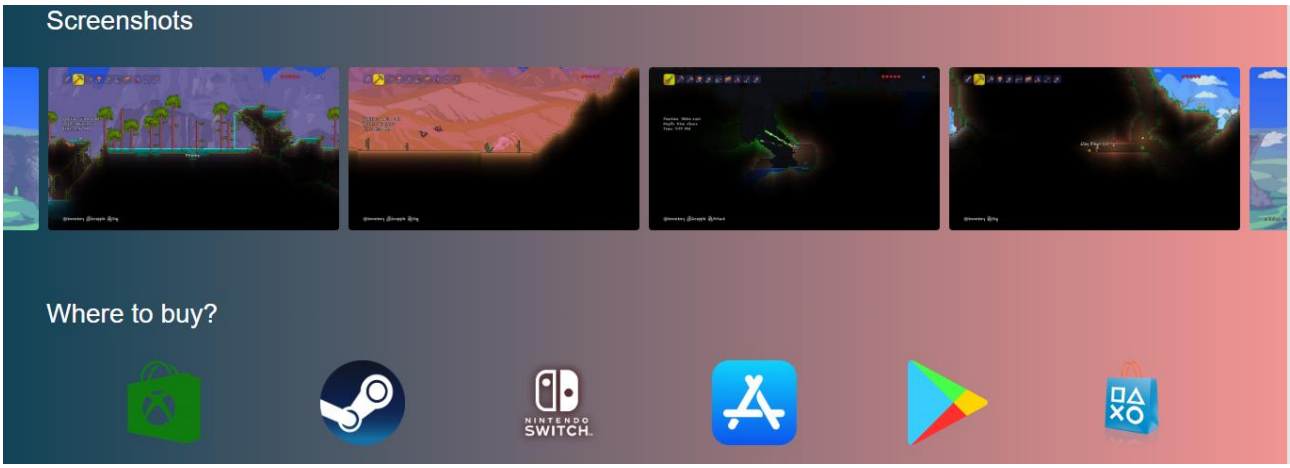

Рисунок 3.23 – Друга половина сторінки продукту

Також в інших продуктах присутній розділ, що показує споріднені ігри, з тієї самої серії або дії яких відбуваються в одному світі. Це продемонстровано на прикладі гри – The Elder Scrolls V: Skyrim та споріднених з нею ігор (рис. 3.24).

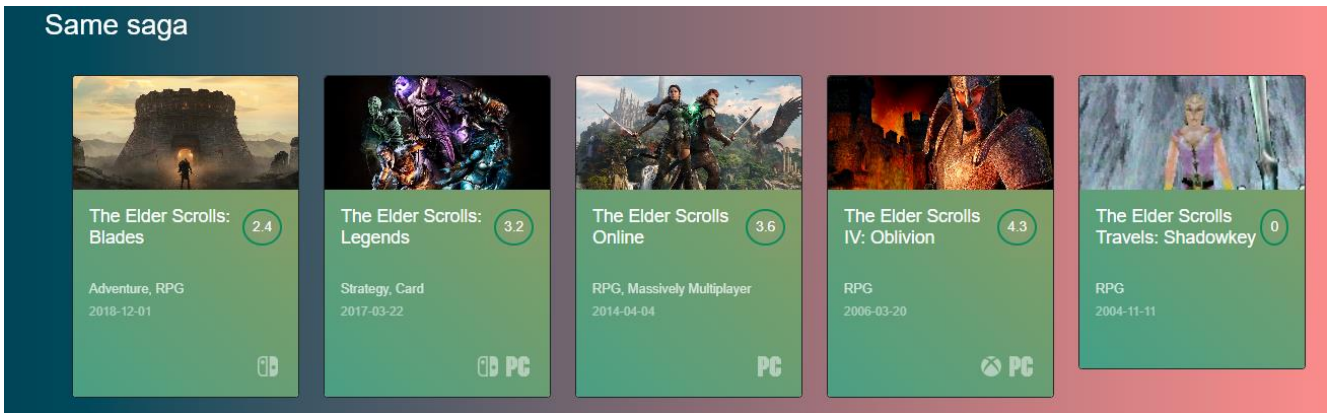

Рисунок 3.24 – Приклад споріднених ігор

Підійшла черга розглянути кнопку меню та її компоненти. За допомогою нею користувач може пересуватись між ключовими сторінками додатку:

- головна сторінка;
- список бажань;
- сторінка, де можна зв’язатись з адміністрацією сайту.

Включення функції списку бажань в додатку додає користувацькому досвіду ще один рівень зручності та персоналізації. Список бажань слугує віртуальним простором, де користувач може зберігати та відстежувати бажані ігри, створюючи персоналізовану колекцію. Загалом, функція списку бажань надає можливість персоналізувати додаток. Користувач може додавати бажаний продукт за допомогою відповідної кнопки на картці. Зі списку бажань користувач їх може видалити за тим самим принципом. Дана сторінка зображена на рисунку 3.25.

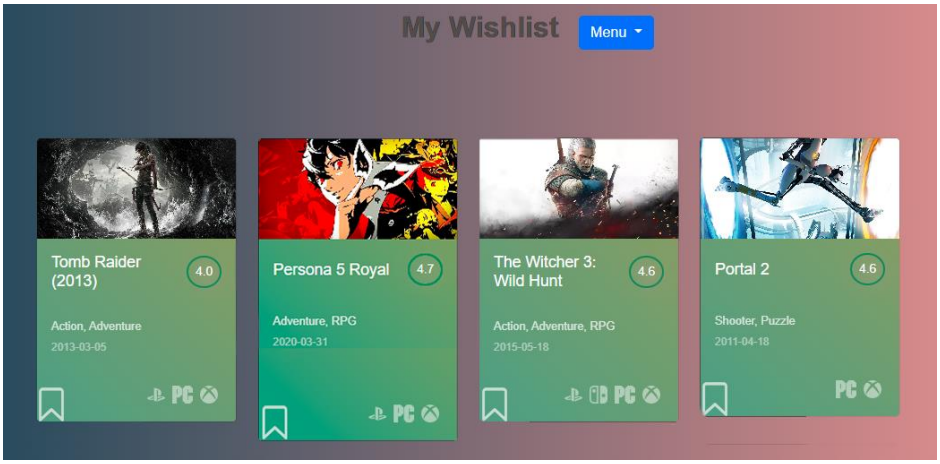

Рисунок 3.25 – Список бажань

Сторінка "Зв'язатися з нами" в додатку слугує важливим каналом зв'язку між адміністрацією сайту та користувачем. Ця сторінка призначена для того, щоб надати користувачам зручний і доступний спосіб звернутися зі своїми запитаннями, відгуками або проблемами. Сторінка містить такі елементи:

- Карту, де розташований офіс компанії, що розробила додаток.
- Контактну форму за допомогою якої користувач може написати свої побажання, проблеми або рекомендації по роботі додатку.
- Адресу електронної пошти та номер телефону, на які можна звертатись.

Дана сторінка зображена на рисунку 3.26

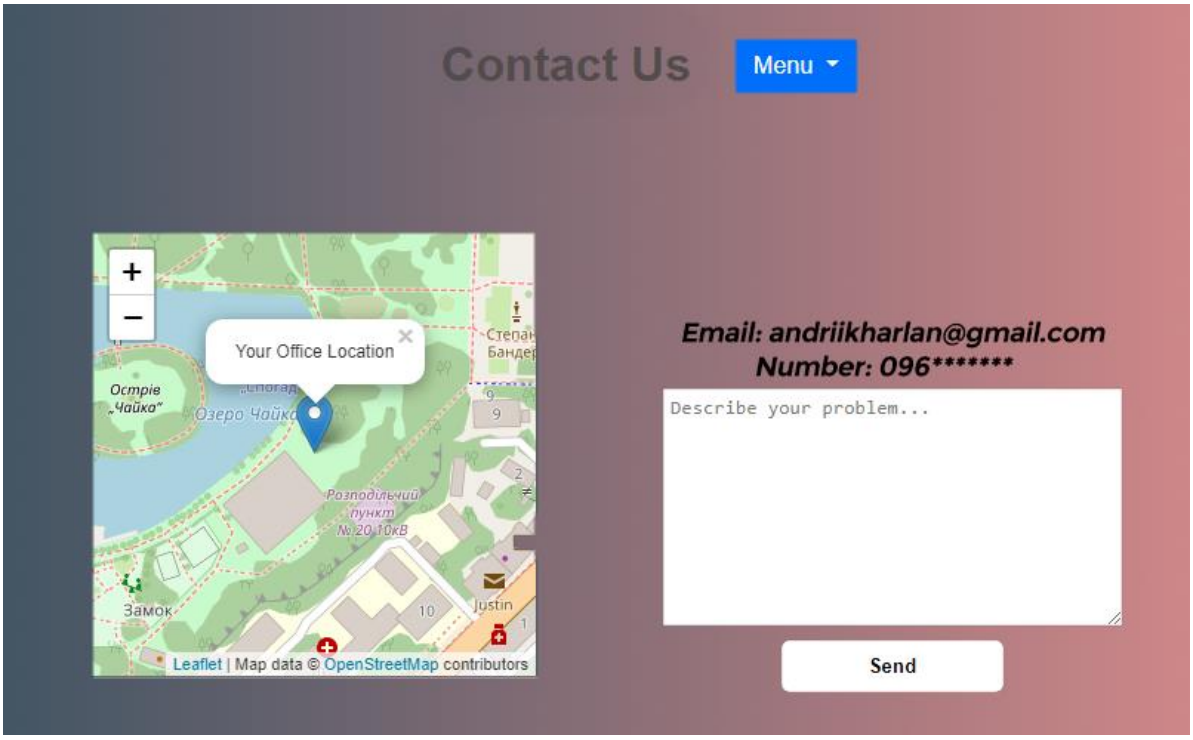

Рисунок 3.26 – Сторінка "Зв'язатися з нами"

На цьому розробка даної платформи цифрової дистрибуції програмного забезпечення та ігор була успішно завершена.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Проведення інструктажів з охорони праці для програміста

Види та порядок проведення інструктажів з охорони праці визначені Типовим положенням про порядок проведення навчання і перевірки знань з питань охорони праці, затвердженим наказом Державного комітету України з нагляду за охороною праці № 15 від 26.01.2005 року [10][11].

Інструкція з охорони праці для програміста [12] розроблена на основі ДНАОП 0.00-6.03-93 “Порядок опрацювання та затвердження власником нормативних актів про охорону праці, що діють на підприємстві” [13], ДНАОП 0.00-4.15-98 “Положення про розробку інструкцій з охорони праці” [14], ДНАОП 0.00-4.12-99 “Типове положення про навчання з питань охорони праці” [15].

Робочим місцем протягом усієї робочої зміни для програміста є спеціально обладнане місце[].

Програміст зобов’язаний:

- піклуватися про особисту безпеку і здоров’я, а також про безпеку і здоров’я оточуючих людей у процесі виконання будь-яких робіт або під час знаходження на території підприємства;
- знати і виконувати вимоги інструкцій з охорони праці і по видах робіт на своєму робочому місці;
- виконувати роботу відповідно до вимог інструкційно-технологічної карти;
- вміти користуватися засобами індивідуального і колективного захисту;
- знати і виконувати Правила поводження з устаткуванням, інвентарем, користуватися технічним паспортом на устаткування;
- знати і виконувати обов’язки з охорони праці, передбачені колективним договором (трудовим договором), правилами внутрішнього трудового розпорядку підприємства, в тому числі:

- вчасно починати і закінчувати роботу, дотримуватися розкладу технологічної і обідньої перерв;
- не виконувати роботи, що не передбачені змінним завданням;
- не перебувати на роботі в неробочій час без відповідного розпорядження керівника;
- дотримуватись правил корпоративного поводження;
- проходити в установленому порядку медичні огляди;
- вміти надати першу допомогу потерпілому від нещасного випадку;
- перед початком роботи перевіряти справність устаткування, огороджень, інженерно-технічних засобів безпеки, інвентарю, засобів пожежогасіння;
- співпрацювати з роботодавцем у справі організації безпечних і нешкідливих умов праці, особисто вживати можливих заходів щодо усунення будь-якої ситуації, що створює загрозу її життю чи здоров'ю або людям, які її оточують та навколишньому природному середовищу;
- при виявленні недоліків чи небезпеки зобов'язана повідомляти безпосереднього керівника або іншу посадову особу.

Загальними вимогами безпеки перед початком робіт потрібно перевірити такі речі:

- справність обладнання, інструменту, приладів;
- наявність і справність достатнього освітлення, вентиляції, обладнання тощо;
- перевірити справність рубильників, розеток, штепсельних з'єднань тощо.

Під час виконання роботи програміст повинен виконувати роботу згідно із своїми посадовими обов'язками. Не залишати без нагляду своє робоче місце, коли обладнання підключено до електромережі.

Після закінчення роботи потрібно:

- Перевірити своє робоче місце.

- Відключити від електромережі електрообладнання.
- Закрити вікна.
- Вжити заходів особистої гігієни: старанно вимити руки, при можливості прийняти душ.

4.2 Долікарська допомога при харчових отруєннях

Важливо, щоб медичну допомогу надавав лікар спеціаліст або фельдшер. Якщо на момент події такого фахівця поряд з постраждалим немає, то будь-хто має надати долікарську допомогу, керуючись відповідними алгоритмами [16].

Долікарська допомога - це невідкладні дії та організаційні заходи, спрямовані на врятування та збереження життя людини у невідкладному стані, мінімізацію наслідків впливу такого стану на її здоров'я, що здійснюються на місці події особами, які не мають медичної освіти [17].

У випадку долікарської допомоги при харчовому отруєнні, вона повинна бути надана якомога швидше. Від швидкості та своєчасних і правильних дій часом залежить не лише здоров'я постраждалого, а і його життя. Харчові інтоксикації частіше всього протікають у гострій формі, симптоми розвиваються швидко і залежать від типу отруєння [18]:

- Харчові токсикоінфекції — отруєння продуктами харчування, що містять мікроби.
- Інтоксикація хімічними речовинами.
- Отруєння отрутами тварин або рослин.

Надання допомоги при харчовому отруєнні базується на чотирьох правилах:

- Очищення (промивання шлунку або внутрішньовенна терапія).
- Адсорбція та виведення токсинів.
- Велика кількість пиття.
- Сувора дієта.

Самостійні дії по очищенню шлунку та нейтралізації отруєння недопустимі у конкретних випадках — виклик швидкої допомоги є необхідним для таких категорій постраждалих [18]:

- Особи похилого віку (старші 60 років).
- Діти віком від народження до 15 років.
- Люди, що мають хронічні захворювання (шлункового-кишкового тракту, кардіологія, цукровий діабет, нефропатії, невралгічні патології, астма, тощо).
- Харчове отруєння отруйними рослинами або грибами.
- У випадках, коли у постраждалого проявляються симптоми паралічу, порушення свідомості.

Якщо інтоксикація визначається, як легка, у хворого немає небезпечних для життя симптомів (безперервне блювання, діарея з кров'ю, падіння артеріального тиску, судоми), можна зробити наступне:

- Постраждалому потрібно випити якомога більше чистої води (кип'ячена, очищена, мінеральна вода без газу). Содові розчини, відвари ромашки та інші “народні” засоби є недоречними і, навіть, можуть нашкодити на першому етапі очищення. Збудник і токсин, які викликали отруєння є невідомими, як і реакції, що відбуваються всередині організму. Якщо у хворого відсутній блювотний рефлекс, можна його активувати шляхом натискання на корінь язика (краще робити це чистою ложкою, а не пальцем руки).
- Випита вода не тільки допоможе очистити травний тракт, а й компенсує втрату рідини, що видаляється із організму разом із блювотними масами та діареєю. Надання допомоги при отруєнні — нейтралізація зневоднення. Постраждалому необхідно випивати не менше 2 л води на добу. Краще, якщо це буде регідратаційний препарат, куплений в аптеці. У домашніх умовах можна приготувати пиття таким чином: на 1 літр очищеної води додають чайну ложку солі та 2 столових ложки цукру.

- Хворому дають випити адсорбуючий токсини препарат, наприклад, активоване вугілля.

Якщо симптоми отруєння через 4-6 годин не стихають, потрібно викликати лікаря, самолікування може погіршити стан постраждалого.

Алгоритм дій є наступним:

- Екстрене виведення токсинів із травного тракту. Це можна зробити промиванням шлунку — прийом великої кількості рідини та активізація блювотного рефлексу. Зверніть увагу, що за допомогою блювання організм чистить шлунок, а діарея очищує кишечник, тому не варто її зупиняти, принаймні, у перші 2-3 години після отруєння.
- Припинення поширення токсинів. Це можна зробити шляхом прийому сорбентів — активованого вугілля, тощо.
- Активоване вугілля
- Забезпечення “відпочинку” органам травлення. Долікарська допомога при харчових отруєннях — це голод протягом першої доби та обмеження у харчуванні (дієта) протягом наступних 5-7 днів. Їжа повинна бути вареною, подрібненою (відвар рису, киселі, легкі супи-пюре).
- Відновлення функції шлунково-кишкового тракту. Це робиться за допомогою двотижневого прийому спеціальних ферментних препаратів та пробіотиків.

Наперекір поширеній думці, очищувальні клізми, прийом антибактеріальних або закріплюючих препаратів можуть погіршити стан хворого та погіршити клінічну картину харчового отруєння. Також, не варто приймати самостійні рішення та дії при небезпечних симптомах, єдине, що потрібно зробити якомога швидше — викликати швидку допомогу.

ВИСНОВКИ

У дипломній роботі було досліджено розробку та впровадження платформи цифрової дистрибуції програмного забезпечення та ігор з використанням технології Chromium Embedded Framework. Так, в ході виконання роботи було створено додаток під назвою “Onirique”, що представляє собою платформу-посередник за допомогою якої користувачі мають велику кількість можливостей взаємодії з додатком. Програма працює на базі власного браузера з внутрішніми посиланнями, які дозволяють вільно та легко користуватись нею.

У розділі з огляду предметної області було проведено аналіз конкурентів, їх функціональності та особливостей. Це дозволило визначити переваги додатку в порівнянні з існуючими аналогами.

Технічний аспект розробленого додатку включав докладний опис архітектури програми, використані технології та інструменти розробки. Було розглянуто переваги використання Chromium Embedded Framework.

Також розробка моделі та програмного комплексу, де відбулось проєктування платформи, основних зв'язків додатку, створення діаграми варіантів використання та її сценаріїв. Діаграм послідовності, які зобразили реалізацію сценаріїв варіантів використання та їх зв'язків, а найголовніше, створено діаграму класів, яка описала атрибути та операції потрібні в системі. Діаграми варіантів використання, послідовності та класів допомогли систематизувати розуміння структури додатку.

В роботі присутня детальна реалізація класів додатку, розроблені макети користувацького інтерфейсу. А найголовніше в цьому розділі був результат – успішно розроблений додаток платформи-посередника.

Загалом, дипломна робота представляє детальне дослідження, розробку та впровадження платформи цифрової дистрибуції програмного забезпечення та ігор з використанням технології Chromium Embedded Framework.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Структура Microsoft Store [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/gaming/gdk/_content/gc/live/get-started/live-xbl-overview
2. Epic Games Store для розробників [Електронний ресурс] – Режим доступу до ресурсу: https://dev.epicgames.com/docs
3. Steam для розробників [Електронний ресурс] – Режим доступу до ресурсу: https://partner.steamgames.com/doc/home
4. CEF документація [Електронний ресурс] – Режим доступу до ресурсу: https://bitbucket.org/chromiumembedded/cef/wiki/Home
5. Документація JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://devdocs.io/javascript/
6. Документація C++ [Електронний ресурс] – Режим доступу до ресурсу: https: https://devdocs.io/cpp/
7. Все про ООП в C++ [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/cpp/cpp_oop.asp
8. Інструкції та опис IBM RSA [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/support/pages/
9. The Unified Modeling Language [Електронний ресурс] – Режим доступу до ресурсу: https://www.uml-diagrams.org/
10. Гандзюк М.П., Желібо Є.П., Халімовський М.О. Основи охорони праці. Підручник – К.: Каравела, 2007. 408 с.
11. Закон України «Про охорону праці». [Електронний ресурс] URL: https://zakon.rada.gov.ua/laws/show/2694-12
12. НПАОП 0.00-4.12-05 «Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці». [Електронний ресурс] URL: https://zakon.rada.gov.ua/laws/show/z0231-05
13. ДНАОП 0.00-6.03-93 «Порядок опрацювання та затвердження власником нормативних актів про охорону праці, що діють на підприємстві».

[Електронний ресурс] URL: https://dnaop.com/html/43271/doc-ДНАОП_0.00-6.03-93

14. ДНАОП 0.00-4.15-98 «Положення про розробку інструкцій з охорони праці». [Електронний ресурс] URL: https://dnaop.com/html/43320/doc-ДНАОАП_0.00-4.15-98

15. ДНАОП 0.00-4.12-99 «Положення про навчання з питань охорони праці». [Електронний ресурс] URL: https://dnaop.com/html/43063/doc-ДНАОП_0.00-4.12-99

16. Желібо Є.П., Зацарний В.В. Безпека життєдіяльності. Підручник. – К.: Каравела, 2006. – 288 с.

17. Закон України «Про екстрену медичну допомогу». [Електронний ресурс] URL: https://zakon.rada.gov.ua/laws/show/5081-17

18. «Порядок надання домедичної допомоги постраждалим при підозрі на гостре отруєння невідомою речовиною». [Електронний ресурс] URL: https://zakon.rada.gov.ua/laws/show/z0356-22

ДОДАТКИ

ДОДАТОК А

Лістинг коду розробленого додатку

Лістинг 1 – Файл cef_win.cc

```
#include <windows.h>
#include "include/cef_command_line.h"
#include "include/cef_sandbox_win.h"
#include "tests/cef/app.h"

#if defined(CEF_USE_SANDBOX)
#pragma comment(lib, "cef_sandbox.lib")
#endif

// Entry point function for all processes.
int APIENTRY wWinMain(HINSTANCE hInstance,
                      HINSTANCE hPrevInstance,
                      LPTSTR lpCmdLine,
                      int nCmdShow) {
  UNREFERENCED_PARAMETER(hPrevInstance);
  UNREFERENCED_PARAMETER(lpCmdLine);

  // Enable High-DPI support
  CefEnableHighDPISupport();
  void* sandbox_info = nullptr;

#if defined(CEF_USE_SANDBOX)
  // Manage the life span of the sandbox information object.
  CefScopedSandboxInfo scoped_sandbox;
  sandbox_info = scoped_sandbox.sandbox_info();
#endif
  // Provide CEF with command-line arguments.
  CefMainArgs main_args(hInstance);

   //function checks the command-line and, if this is a sub-process, executes
the appropriate logic.
  int exit_code = CefExecuteProcess(main_args, nullptr, sandbox_info);
  if (exit_code >= 0) {
    return exit_code;
  }

  CefRefPtr<CefCommandLine>                  command_line                  =
CefCommandLine::CreateCommandLine();
  command_line->InitFromString(::GetCommandLineW());

  //CEF global settings.
  CefSettings settings;

  if (command_line->HasSwitch("enable-chrome-runtime")) {
    settings.chrome_runtime = true;
  }

#if !defined(CEF_USE_SANDBOX)
  settings.no_sandbox = true;
```

```
#endif

  CefRefPtr<App> app(new App);

  // Initialize CEF.
  CefInitialize(main_args, settings, app.get(), sandbox_info);

  // CEF message loop
  CefRunMessageLoop();

  // Shut down CEF.
  CefShutdown();

  return 0;
}
```

Лістинг 2 – Файл app.cc

```
#include "tests/cef/app.h"
#include <string>
#include "include/cef_browser.h"
#include "include/cef_command_line.h"
#include "include/views/cef_browser_view.h"
#include "include/views/cef_window.h"
#include "include/wrapper/cef_helpers.h"
#include "tests/cef/handler.h"
#include "include/cef_browser.h"

namespace {
// object provides the delegate implementation for the CefWindow.
class WindowDelegate : public CefWindowDelegate {
 public:
  explicit WindowDelegate(CefRefPtr<CefBrowserView> browser_view)
      : browser_view_(browser_view) {}

  void OnWindowCreated(CefRefPtr<CefWindow> window) override {
    // Add the browser view and show the window.
    window->AddChildView(browser_view_);
    window->Show();

    // Give keyboard focus to the browser view.
    browser_view_->RequestFocus();
  }

  void OnWindowDestroyed(CefRefPtr<CefWindow> window) override {
    browser_view_ = nullptr;
  }

  bool CanClose(CefRefPtr<CefWindow> window) override {
    // Allow the window to close if the browser says it's OK.
    CefRefPtr<CefBrowser> browser = browser_view_->GetBrowser();
    if (browser)
      return browser->GetHost()->TryCloseBrowser();
    return true;
  }

  CefSize GetPreferredSize(CefRefPtr<CefView> view) override {
    return CefSize(800, 600);
```

```
  }

 private:
  CefRefPtr<CefBrowserView> browser_view_;

  IMPLEMENT_REFCOUNTING(WindowDelegate);
  DISALLOW_COPY_AND_ASSIGN(WindowDelegate);
};

class BrowserViewDelegate : public CefBrowserViewDelegate {
 public:
  BrowserViewDelegate() {}

  bool OnPopupBrowserViewCreated(CefRefPtr<CefBrowserView> browser_view,
                                 CefRefPtr<CefBrowserView>
popup_browser_view,
                                 bool is_devtools) override {
    // Create a new top-level Window for the popup
    CefWindow::CreateTopLevelWindow(
        new WindowDelegate(popup_browser_view));

    //  created the Window.
    return true;
  }

 private:
  IMPLEMENT_REFCOUNTING(BrowserViewDelegate);
  DISALLOW_COPY_AND_ASSIGN(BrowserViewDelegate);
};

}

App::App() {}

void App::OnContextInitialized() {
  CEF_REQUIRE_UI_THREAD();

  CefRefPtr<CefCommandLine> command_line =
      CefCommandLine::GetGlobalCommandLine();

  // Create the browser using the Views framework if "--use-views" is
specified
  // via the command-line. Otherwise, create the browser using the native
  // platform framework.
  const bool use_views = command_line->HasSwitch("use-views");

  //implements browser-level callbacks.
  CefRefPtr<Handler> handler(new Handler(use_views));

  // settings here.
  CefBrowserSettings browser_settings;
  if (command_line->HasSwitch("disable-web-security")) {

    // Disable web security by setting the appropriate command-line flag.
    CefRefPtr<CefCommandLine> new_command_line = command_line->Copy();
    new_command_line->AppendSwitch("--disable-web-security");
  }
```

```
  std::string url;


  if (use_views) {
    // BrowserView.
    CefRefPtr<CefBrowserView>                  browser_view                  =
CefBrowserView::CreateBrowserView(
        handler, url, browser_settings, nullptr, nullptr,
        new BrowserViewDelegate());

    // Create the Window
    CefWindow::CreateTopLevelWindow(new WindowDelegate(browser_view));
  } else {
    CefWindowInfo window_info;

#if defined(OS_WIN)
    window_info.SetAsPopup(nullptr, "cef");
#endif

    //first browser window.
    CefBrowserHost::CreateBrowser(window_info,         handler,         url,
browser_settings,
                                  nullptr, nullptr);
  }
}

CefRefPtr<CefClient> App::GetDefaultClient() {
  return Handler::GetInstance();
}
```

Лістинг 3 – Файл handler.cc

```
#include "tests/cef/handler.h"
#include <sstream>
#include <string>
#include "include/base/cef_callback.h"
#include "include/cef_app.h"
#include "include/cef_parser.h"
#include "include/views/cef_browser_view.h"
#include "include/views/cef_window.h"
#include "include/wrapper/cef_closure_task.h"
#include "include/wrapper/cef_helpers.h"

namespace {

Handler* g_instance = nullptr;

std::string   GetDataURI(const   std::string&   data,   const   std::string&
mime_type) {
  return "data:" + mime_type + ";base64," +
         CefURIEncode(CefBase64Encode(data.data(), data.size()), false)
             .ToString();
}
}

Handler::Handler(bool use_views)
    : use_views_(use_views), is_closing_(false) {
  DCHECK(!g_instance);
```

```
  g_instance = this;
}

Handler::~Handler() {
  g_instance = nullptr;
}

// static
Handler* Handler::GetInstance() {
  return g_instance;
}

void Handler::OnTitleChange(CefRefPtr<CefBrowser> browser,
                                    const CefString& title) {
  CEF_REQUIRE_UI_THREAD();

  if (use_views_) {
    // Set the title of the window using the Views framework.
    CefRefPtr<CefBrowserView> browser_view =
        CefBrowserView::GetForBrowser(browser);
    if (browser_view) {
      CefRefPtr<CefWindow> window = browser_view->GetWindow();
      if (window)
        window->SetTitle(title);
    }
  } else if (!IsChromeRuntimeEnabled()) {
    // Set the title of the window using platform APIs.
    PlatformTitleChange(browser, title);
  }
}

void Handler::OnAfterCreated(CefRefPtr<CefBrowser> browser) {
  CEF_REQUIRE_UI_THREAD();

  // Add to the list of existing browsers.
  browser_list_.push_back(browser);
}

bool Handler::DoClose(CefRefPtr<CefBrowser> browser) {
  CEF_REQUIRE_UI_THREAD();

  if (browser_list_.size() == 1) {
    // Set a flag to indicate that the window close should be allowed.
    is_closing_ = true;
  }
  // Allow the close
  return false;
}

void Handler::OnBeforeClose(CefRefPtr<CefBrowser> browser) {
  CEF_REQUIRE_UI_THREAD();
  // Remove from the list of existing browsers.
  BrowserList::iterator bit = browser_list_.begin();
  for (; bit != browser_list_.end(); ++bit) {
    if ((*bit)->IsSame(browser)) {
      browser_list_.erase(bit);
      break;
    }
```

```
  }
  if (browser_list_.empty()) {
    // All browser windows have closed. Quit the application message loop.
    CefQuitMessageLoop();
  }
}

void Handler::OnLoadError(CefRefPtr<CefBrowser> browser,
                          CefRefPtr<CefFrame> frame,
                          ErrorCode errorCode,
                          const CefString& errorText,
                          const CefString& failedUrl) {
  CEF_REQUIRE_UI_THREAD();

  // Allow Chrome to show the error page.
  if (IsChromeRuntimeEnabled())
    return;

  // Don't display an error for downloaded files.
  if (errorCode == ERR_ABORTED)
    return;

  // Display a load error message using a data
  std::stringstream ss;
  ss << "<html><body bgcolor=\"white\">"
        "<h2>Failed to load URL "
     << std::string(failedUrl) << " with error " << std::string(errorText)
     << " (" << errorCode << ").</h2></body></html>";

  frame->LoadURL(GetDataURI(ss.str(), "text/html"));
}

void Handler::CloseAllBrowsers(bool force_close) {
  if (!CefCurrentlyOn(TID_UI)) {
    // Execute on the UI thread.
    CefPostTask(TID_UI, base::BindOnce(&Handler::CloseAllBrowsers, this,
                                       force_close));
    return;
  }

  if (browser_list_.empty())
    return;

  BrowserList::const_iterator it = browser_list_.begin();
  for (; it != browser_list_.end(); ++it)
    (*it)->GetHost()->CloseBrowser(force_close);
}

// static
bool Handler::IsChromeRuntimeEnabled() {
  static int value = -1;
  if (value == -1) {
    CefRefPtr<CefCommandLine> command_line =
        CefCommandLine::GetGlobalCommandLine();
    value = command_line->HasSwitch("enable-chrome-runtime") ? 1 : 0;
  }
  return value == 1;
```

Лістинг 4 – Файл handler.h

```
#ifndef CEF_TESTS_CEFHANDLER_H_
#define CEF_TESTS_CEFHANDLER_H_
#include "include/cef_client.h"
#include <list>

class Handler : public CefClient,
                    public CefDisplayHandler,
                    public CefLifeSpanHandler,
                    public CefLoadHandler {
 public:
  explicit Handler(bool use_views);
  ~Handler();

  // Provide access to the single global instance of this object.
  static Handler* GetInstance();

  // CefClient methods:
  virtual CefRefPtr<CefDisplayHandler> GetDisplayHandler() override {
    return this;
  }
  virtual CefRefPtr<CefLifeSpanHandler> GetLifeSpanHandler() override {
    return this;
  }
  virtual CefRefPtr<CefLoadHandler> GetLoadHandler() override { return this;
}

  // CefDisplayHandler methods:
  virtual void OnTitleChange(CefRefPtr<CefBrowser> browser,
                             const CefString& title) override;

  // CefLifeSpanHandler methods:
  virtual void OnAfterCreated(CefRefPtr<CefBrowser> browser) override;
  virtual bool DoClose(CefRefPtr<CefBrowser> browser) override;
  virtual void OnBeforeClose(CefRefPtr<CefBrowser> browser) override;

  // CefLoadHandler methods:
  virtual void OnLoadError(CefRefPtr<CefBrowser> browser,
                           CefRefPtr<CefFrame> frame,
                           ErrorCode errorCode,
                           const CefString& errorText,
                           const CefString& failedUrl) override;

  // Request that all existing browser windows close.
  void CloseAllBrowsers(bool force_close);

  bool IsClosing() const { return is_closing_; }

  // Returns true if the Chrome runtime is enabled.
  static bool IsChromeRuntimeEnabled();

 private:
  void PlatformTitleChange(CefRefPtr<CefBrowser> browser,
                           const CefString& title);

  const bool use_views_;
```

```
  // List of existing browser windows. Only accessed on the CEF UI thread.
  typedef std::list<CefRefPtr<CefBrowser>> BrowserList;
  BrowserList browser_list_;

  bool is_closing_;

  // Include the default reference counting implementation.
  IMPLEMENT_REFCOUNTING(Handler);
};
```

Лістинг 5 – Файл game.js

```
const GAME_IMG = "https://api.rawg.io/api/games/" + id + "/screenshots?" +
API;
const GAME_ADDITIONS = "https://api.rawg.io/api/games/" + id + "/additions?"
+ API;
const GAME_SERIES = "https://api.rawg.io/api/games/" + id + "/game-series?"
+ API;
const GAME_STORES = "https://api.rawg.io/api/games/" + id + "/stores?" +
API;
const main = document.getElementById("main");
const screenGames = document.getElementById("screenshosts");
const imgScreen = document.getElementById("imgScreen");
const imgContent = document.getElementById("imgContent");
const market = document.getElementById("market");
const marketContent = document.getElementById("marketContent");
const releatedGames = document.getElementById("releatedGames");
const releatedtContent = document.getElementById("releatedtContent");

// Select the owl-carousel element using jQuery
var owl = $('.owl-carousel');
id ? getGames(GAME) : "";
getScreenshots(GAME_IMG);
getReleated(GAME_SERIES);
getStores(GAME_STORES);

// Function to fetch and display game data
function getGames(url) {
    fetch(url).then(res => res.json()).then(data => {
        if (data != null) {
            document.title += " " + data.name;
            showGame(data);
        }
    })
}

// Function to fetch and display game stores
function getStores(url) {
    fetch(url).then(res => res.json()).then(data => {
        if (data.count != 0) {
            showStores(data.results);
        } else {
            market.classList.add("d-none");
        }
    })
}
```

```
// Function to create and display store elements
function showStores(stores) {
    stores.forEach(store => {
        const market = document.createElement("a");
        market.href = store.url;
        market.target = "_blank"
        if (store.store_id == 2 || store.store_id == 3 || store.store_id ==
6 || store.store_id == 1 || store.store_id == 8 || store.store_id == 4 ||
store.store_id == 7) {
            market.innerHTML = `<img src="img/${store.store_id}.png" alt=""
class="store" height="120" />`;
            marketContent.appendChild(market);
        }
    });
}

// Function to display game information
function showGame(data) {
    main.innerHTML = "";
    const style = document.createElement('style');
    const gameInfo = document.createElement("div");
    const {
        name,
        genres,
        description_raw,
        platforms,
        released,
        background_image
    } = data;
    style.innerHTML = `#main{
        background: linear-gradient(
                to bottom,
                rgba(0, 0, 0, 0) 10%,
                rgba(3, 4, 13, 255)
            ),
            url('${background_image}');
        }`;
    document.head.appendChild(style);

    // Populate the gameInfo element with the fetched data
    gameInfo.classList.add("m-5");
    gameInfo.innerHTML = `
        <h1 class="title text-capitalize">
        ${name}
        </h1>
        <br class="d-none d-sm-block" />
        <p class="description lh-sm col-7 d-none d-sm-block">
        ${description_raw}
        </p>
        <br class="d-none d-sm-block" />
        <p    class="description"><strong>Genres:</strong>    ${genres    ?
showGenres(genres) : "Unknow"}</p>
        <p class="description"><strong>Release date:</strong> ${released ?
released : "Unknow"}</p>
        <p class="description">
        <strong>Platforms:</strong> ${showPlatformsCard(platforms)}
        </p>`
```

```
    // Append the gameInfo element to the main element
    main.appendChild(gameInfo);
}

// Function to format and display genres
function showGenres(genres) {
    var genresGame = "";
    genres.forEach(genre => {
        genresGame += genre.name + ", ";
    });
    return genresGame.substring(0, genresGame.length - 2);
}

// Function to format and display platform cards
function showPlatformsCard(platforms) {
    var platformsGame = "";
    if (platforms) {
        platforms.forEach(platform => {
            if (platform.platform.id == 7 || platform.platform.id == 1 ||
platform.platform.id == 4 || platform.platform.id == 18) {
                platformsGame                    +=                    `<img
src="img/${platform.platform.slug}.png"     alt="${platform.platform.slug}"
class="platformsCard m-1 opacity-50" height="20" />`;
            }
        });
    }
    return platformsGame;
}

// Function to fetch and display game screenshots
function getScreenshots(url) {
    fetch(url).then(res => res.json()).then(data => {
        if (data != null) {
            showScreenshots(data.results)
        }
    })
}

// Function to display game screenshots
function showScreenshots(screenshots) {
    imgScreen.innerHTML = ""
    if (screenshots.length != 0) {
        screenshots.forEach(screenshot => {
            const itemImg = document.createElement("div");
            itemImg.classList.add("item");
            itemImg.innerHTML     =     `<img     src="${screenshot.image}"
class="rounded gallery-item" alt=""/>`
            imgScreen.appendChild(itemImg);
        });
    } else {
        screenGames.classList.add("d-none");
    }
    owl.owlCarousel({
        stagePadding: 50,
        loop: true,
        dots: false,
        margin: 10,
```

```
        autoplay: true,
        autoplayTimeout: 3000,
        responsive: {
            0: {
                items: 1
            },
            600: {
                items: 2
            },
            1000: {
                items: 4
            }
        }
    });
}

owl.on('mousewheel', '.owl-stage', function (e) {
    if (e.deltaY > 0) {
        owl.trigger('next.owl');
    } else {
        owl.trigger('prev.owl');
    }
    e.preventDefault();
});

// Function to fetch and display related games
function getReleated(url) {
    fetch(url).then(res => res.json()).then(data => {
        if (data.count != 0) {
            showReleated(data.results);
        } else {
            releatedGames.classList.add("d-none");
        }
    })
}

// Function to display related games
function showReleated(releateds) {
    releateds.forEach(game => {
        const {
            id,
            name,
            genres,
            rating,
            platforms,
            released,
            background_image
        } = game;
        const card = document.createElement("div");
        card.classList.add("col-auto");
        card.classList.add("mb-4");
        card.innerHTML = `
        <div  id="${id}"  class="card  overflow-hidden  border  border-dark"
style="width: 19rem;" onclick="getInfoAbout(this.id)">
        <div class="bg-img"
            style="background-image:        url('${background_image       ?
background_image                  :                  "https://site.groupe-
```

```
psa.com/content/uploads/sites/3/2016/12/white-background-2.jpg"}'); height:
24rem;">
        </div>
        <div class="card-body">
            <div class="d-flex justify-content-between align-items-center">
                <h5 class="card-title">${name}</h5>
                <h6 class="vote p-2 bg-transparent <!--border border-success
rounded-circle-->">${showScore(rating)}</h6>
            </div>
            <br class="d-none d-sm-block"/>
            <h6  class="opacity-75  genres  d-none  d-sm-block">${genres  ?
showGenres(genres) : "Unknow"}</h6>
            <h6  class="opacity-50  date  d-none  d-sm-block">${released  ?
released : "Unknow"}</h6>
            <br class="d-none d-sm-block"/>
            <div class="d-none d-sm-block">
            <div class="d-flex justify-content-end align-items-center">
                 ${showPlatformsCard(platforms)}
            </div>
            </div>
        </div>
    </div>
        `;
        releatedtContent.appendChild(card);
    });
}

// Function to store the selected game ID and redirect to the game page
function getInfoAbout(id) {
    localStorage.setItem("game", id);
    window.location = "/game";
}

// Function to format and display the game score
function showScore(score) {
    return (score + 0.0).toString().substring(0, 3);
}

// Event listener for clicks on gallery images
document.addEventListener("click", function (e) {
    if (e.target.classList.contains("gallery-item")) {
        const src = e.target.getAttribute("src");
        document.querySelector(".modal-img").src = src;
        var myModal = new bootstrap.Modal(document.getElementById('gallery-
game'));
        myModal.show();
    }
});

// Event listener for window onload event
window.onload = function () {
    // Hide the preloader
    $('#preloader').fadeOut('slow');
    $('#preloader').addClass('visually-hidden');
    $('body').removeClass('hidden');
}
```

Лістинг 6 – Файл index.js

```
// Declare and initialize variables
var selectedGenres = [];
var selectedPlatforms = [];
var checkGeneres = document.getElementsByName("genere");
var checkPlatform = document.getElementsByName("platform");
var nextPage = ""; // Variable to store the URL of the next page
var currentUrl = ""; // Variable to store the current URL

// Function to retrieve games data from a specified URL
function getGames(url) {
    currentUrl = url;
    fetch(url).then(res => res.json()).then(data => {
        if (data.results != 0) {
            nextPage = data.next;
            showGames(data.results);
        }
    });
}

// Function to display games on the webpage
function showGames(data) {
    data.forEach(game => {
        const { id, name, genres, rating, platforms, released,
background_image } = game;
        // Create a new card element
        const card = document.createElement("div");
        card.classList.add("col-auto");
        card.classList.add("mb-4");
        card.innerHTML = `
        main.appendChild(card);
    });
}

// Function to display the score of a game
function showScore(score) {
    return (score + 0.0).toString().substring(0, 3);
}

// Function to display genres of a game
function showGenres(genres) {
    var genresGame = "";
    genres.forEach(genre => {
        genresGame += genre.name + ", ";
    });
    return genresGame.substring(0, genresGame.length - 2);
}

// Function to display platforms of a game in card format
function showPlatformsCard(platforms) {
    var platformsGame = "";
    if (platforms) {
        platforms.forEach(platform => {
            if (platform.platform.id == 7 || platform.platform.id == 1 ||
platform.platform.id == 4 || platform.platform.id == 18) {
```

```
                platformsGame                    +=                    `<img
src="img/${platform.platform.slug}.png"     alt="${platform.platform.slug}"
class="platformsCard m-1 opacity-50" height="20" />`;
            }
        });
    }
    return platformsGame;
}

// Event listener for the search input field
search.addEventListener("keyup", (e) => {
    clearMain();
    uncheckFilters();
    e.preventDefault();
    const searchGame = search.value;
    if (searchGame) {
        getGames(BASE_URL + API + "&search=" + searchGame.replace(" ", "+")
+ "&page=1");
    } else {
        getGames(URL_GAMES);
    }
});

function getGenres(genres) {
    fetch(genres).then(res => res.json()).then(data => {
        if (data.results != 0) {
            showGeneres(data.results);
        }
    });
}

// Function to display genres on the webpage
function showGeneres(genre) {
    filterGeneres.innerHTML = "";
    genre.forEach(gen => {
        const { id, name, games_count } = gen;
        // Create a new checkbox element for the genre
        const checkBox = document.createElement("div");
        checkBox.classList.add("col-12");
        checkBox.innerHTML = `
        <!-- Checkbox HTML structure with genre information -->
        `;
        // Append the checkbox element to the genre filter section of the
webpage
        filterGeneres.appendChild(checkBox);
    });
};

function getPlatforms(platforms) {
    fetch(platforms).then(res => res.json()).then(data => {
        if (data.results != 0) {
            showPlatforms(data.results);
        }
    });
}

// Function to display platforms on the webpage
function showPlatforms(platforms) {
```

```
    filterPlatforms.innerHTML = "";
    platforms.forEach(platform => {
        const { id, slug, name, games_count } = platform;

        if (id == 1 || id == 18 || id == 4 || id == 7) {
            const checkPlatform = document.createElement("div");
            checkPlatform.innerHTML = `
            <!-- Checkbox HTML structure with platform information -->
            `;

            filterPlatforms.appendChild(checkPlatform);
        }
    });
}


// Functions to handle different sorting/filtering options

function highScore() {
    clearMain();
    getGames(BEST_GAMES);
}

function lowScore() {
    clearMain();
    getGames(WORST_GAMES);
}

function relevance() {
    clearMain();
    getGames(URL_GAMES);
}

// Function to filter games based on selected genres and platforms
function filterBy() {
    selectedGenres = [];
    selectedPlatforms = [];
    var genres = "";
    var platforms = "";
    var inputSearch = search.value || "";
    for (var i = 0, n = checkGeneres.length; i < n; i++) {
        if (checkGeneres[i].checked) {
            selectedGenres.push(checkGeneres[i].value);
        }
    }
    for (var i = 0, n = checkPlatform.length; i < n; i++) {
        if (checkPlatform[i].checked) {
            selectedPlatforms.push(checkPlatform[i].value);
        }
    }
    selectedGenres.length    !=    0    ?    genres    =    "&genres="    +
selectedGenres.toString() : "";
    selectedPlatforms.length   !=   0   ?   platforms   =   "&platforms="   +
selectedPlatforms.toString() : "";
    inputSearch ? inputSearch = "&search=" + inputSearch.replace(" ", "+")
: "";
    clearMain();
    getGames(URL_GAMES + genres + platforms + inputSearch);
```

ДОДАТОК Б

Диск з розробленою програмою