МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

СТУКАН АНДРІЙ ОЛЕГОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ О. В. Зелінська
«____»_____ 2025 р.

РОЗРОБКА ІНФОРМАЦІЙНО-ПОШУКОВОЇ СИСТЕМИ ДЛЯ КОМП'ЮТЕРНИХ ІГОР

Спеціальність 122 Комп'ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Р. М. Бабаков, професор
кафедри інформаційних технологій,
д.т.н., доцент
_____

Оцінка:____/ ____/ _____

(бали за шкалою СКТС/за національною шкалою)

Голова ЕК: _____

(підпис)

Вінниця 2025

АНОТАЦІЯ

У бакалаврській роботі представлено розробку інформаційно-пошукового вебзастосунку «Інформаційно-пошукова система для комп'ютерних ігор», що дозволяє користувачам здійснювати пошук комп'ютерних ігор, переглядати новини та зберігати улюблені проєкти. Для реалізації функціоналу використано сучасні технології веброзробки: HTML5, CSS3, JavaScript, а також інтеграцію з RAWG Video Games Database API. Застосунок підтримує мультимовний інтерфейс (українська/англійська), адаптивну верстку та збереження даних у localStorage. Основну увагу приділено зручності інтерфейсу, персоналізації контенту та можливостям розширення. Робота демонструє практичне застосування теоретичних знань з розробки клієнтських вебінтерфейсів.

Ключові слова: вебзастосунок, ігри, пошук, API, localStorage, JavaScript, RAWG, мультимовність.

ABSTRACT

The bachelor's thesis presents the development of an information and search web application titled "Information and Search System for Computer Games", which allows users to search for PC games, view gaming news, and save favorite projects. The application is implemented using modern web technologies: HTML5, CSS3, JavaScript, and integration with the RAWG Video Games Database API. It features a multilingual interface (Ukrainian/English), responsive layout, and localStorage support for saving data. The focus is placed on user-friendly design, personalized content, and extendability. The project demonstrates practical implementation of client-side web development concepts.

Keywords: web application, games, search, API, localStorage, JavaScript, RAWG, multilingual.

ЗМІСТ

ВСТУП

У сучасному світі ігрова індустрія стала однією з найдинамічніших і найвпливовіших сфер цифрових технологій. З огляду на щоденні оновлення, новинки та релізи, користувачам важливо мати зручну платформу для перегляду новин, пошуку ігор та формування власної колекції обраного. У зв'язку з цим, тема створення вебзастосунку «Information and Search System for Computer Games» є актуальною.

Інформаційно-пошукові вебсервіси відіграють ключову роль у забезпеченні швидкого доступу до великих обсягів даних. У контексті ігрової індустрії це означає можливість своєчасно дізнаватися про нові релізи, знижки, популярні жанри та огляди. Водночас користувачі очікують інтуїтивно зрозумілий, візуально привабливий і мобільно-дружній інтерфейс.

З технічної точки зору, створення такого вебзастосунку вимагає знань сучасних вебтехнологій — HTML, CSS, JavaScript, а також уміння інтегрувати зовнішні API, обробляти JSON-дані, зберігати інформацію в localStorage тощо.

Актуальність проєкту також зумовлена потребою створювати SPA-застосунки (Single Page Application), які працюють швидко, без перезавантаження сторінки, та забезпечують плавну взаємодію користувача з системою.

Метою роботи є розробка сучасного вебзастосунку, який дозволяє користувачам:

- переглядати новини зі світу ігор;
- здійснювати пошук ігор через публічний API;
- відфільтровувати ігри за жанрами і платформами;
- зберігати улюблені проєкти в обране;
- переходити на зовнішні ресурси (офіційні сайти або сторінки у маркетплейсах) для перегляду або завантаження гри.

Досягнення цієї мети супроводжується виконанням низки завдань, серед яких:

- аналіз вимог до структури застосунку;
- вибір відповідних технологій для реалізації функціоналу;
- реалізація динамічного пошуку та фільтрації за допомогою API;
- створення зручного користувацького інтерфейсу з мультимовною підтримкою;
- тестування працездатності системи та її адаптація до різних типів пристроїв.

Результатом проєкту є вебзастосунок, що не лише відповідає сучасним вимогам функціональності та зручності, а й слугує прикладом практичного застосування знань у сфері веброзробки.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ВЕБЗАСТОСУНКІВ

1.1 Актуальність теми

Індустрія комп'ютерних ігор стрімко розвивається, перетворившись на одну з провідних галузей цифрової економіки, що охоплює не лише розваги, а й освіту, мистецтво, соціальну взаємодію та електронний спорт. Згідно зі звітами аналітичних компаній, щороку обсяги ринку відеоігор зростають на 8–12%, а кількість активних гравців у світі перевищує 3 мільярди. Такі масштаби зумовлюють необхідність створення потужних інформаційно-пошукових систем, які б дозволяли користувачам орієнтуватися в океані доступного контенту. [8]

Сучасні геймери мають справу з великою кількістю нових релізів, оновлень, акцій, платформ, жанрових змін тощо. Щоденно у Steam, Epic Games Store, PlayStation Store, Nintendo eShop та інших сервісах з’являються десятки нових ігор. Крім того, існує безліч незалежних ігор (indie), які не мають великої рекламної підтримки, але заслуговують уваги. У такому середовищі користувачам важливо мати інструмент, який дозволить швидко фільтрувати контент за жанрами, платформами, датами виходу, рейтингами або іншими параметрами, а також зберігати вподобане для подальшого перегляду чи завантаження.

З погляду інформаційних технологій, така потреба формує попит на створення вебзастосунків, які є не лише пошуковими механізмами, а й персональними асистентами в інформаційному просторі індустрії відеоігор. Ці застосунки мають забезпечувати інтерактивний інтерфейс, адаптивність до мобільних пристроїв, гнучкість фільтрації, можливість локального збереження інформації та інтеграцію з надійними джерелами даних — наприклад, RAWG API, Steam API, IGDB тощо. [9]

Важливою тенденцією є перехід користувачів до односторінкових вебзастосунків (SPA), що дозволяють реалізувати максимально швидку взаємодію з користувачем без необхідності перезавантаження сторінки. Такий підхід сприяє зменшенню навантаження на сервер, покращенню UX та підвищенню ефективності використання клієнтських ресурсів. [10] Крім того, мультимовна підтримка стає ключовим аспектом, особливо для проєктів, які орієнтовані на широке коло користувачів у різних країнах.

Актуальність теми зумовлюється також появою нових технологій: прогресивних вебзастосунків (PWA), serverless-рішень, використання локального зберігання замість традиційних баз даних для певних задач. Ці тенденції створюють передумови для проєктування систем, які поєднують простоту реалізації та високу ефективність на практиці [12].

Розробка вебзастосунку «Information and Search System for Computer Games» у цьому контексті є відповіддю на виклики сучасного світу. Вона дає змогу реалізувати низку ключових технологій, зокрема:

- отримання даних з відкритих API;
- динамічне формування інтерфейсу користувача;
- адаптивний дизайн під мобільні та настільні пристрої;
- можливість збереження обраного контенту;
- перемикання мов інтерфейсу;
- реалізація фільтрів, сортування та інтуїтивної навігації.

Крім практичної користі, такий проєкт є цінним і з навчального погляду: він дозволяє здобути досвід роботи з JavaScript, CSS, HTML, REST API, JSON-структурами, зберіганням даних у браузері та створенням SPA.

Застосунок може бути використаний як шаблон для подібних рішень в інших сферах — кіно, музика, книги, курси тощо. Таким чином, запропонована тема не лише має прикладне значення, а й відкриває можливості для подальших

досліджень та вдосконалень. Вона інтегрує в собі потреби ринку, технологічні тренди та навички майбутніх фахівців у галузі комп'ютерних наук.

У підсумку, можна стверджувати, що створення інформаційно-пошукової системи для комп'ютерних ігор є актуальним, своєчасним та перспективним напрямком, який поєднує інтереси користувачів, потреби індустрії та можливості реалізації за допомогою сучасних інструментів веброзробки. [11]

1.2 Основні поняття веброзробки

Веброзробка є фундаментальною складовою сучасного цифрового середовища, яка охоплює створення вебінтерфейсів, динамічних сторінок та повноцінних вебзастосунків, що функціонують у середовищі браузера на різних типах пристроїв. Веброзробка є міждисциплінарною галуззю, яка поєднує знання з програмування, дизайну, мережевих технологій, безпеки та взаємодії з користувачем (UX/UI).

Класична структура сучасного вебзастосунку включає такі основні компоненти:

- **HTML (HyperText Markup Language)** — мова розмітки, що визначає структуру вебсторінки. За її допомогою створюються основні блоки: заголовки, абзаци, списки, кнопки, зображення тощо.[1]
- **CSS (Cascading Style Sheets)** — мова стилів, яка використовується для оформлення сторінки, включаючи кольори, шрифти, розміщення елементів, а також для адаптивної верстки під різні екрани.[2]
- **JavaScript** — мова програмування, яка відповідає за динамічну поведінку елементів на сторінці. Вона дозволяє реалізувати інтерактивність, взаємодію з API, обробку подій користувача, зміну вмісту DOM у реальному часі.[3]
- **LocalStorage / SessionStorage** — API браузера, що дозволяє зберігати дані безпосередньо на клієнтському боці. Це забезпечує швидкий

доступ до персоналізованого контенту без звернення до серверної частини.[4]

- **API (Application Programming Interface)** — набір інтерфейсів, що дозволяє підключатися до зовнішніх джерел даних (наприклад, баз ігор, новин, користувацьких профілів тощо). Обробка API-запитів здійснюється за допомогою fetch або axios.

У сучасній веброзробці все більше застосовується підхід **SPA (Single Page Application)** — односторінковий застосунок, де взаємодія з користувачем відбувається без повного перезавантаження сторінки. Завдяки цьому забезпечується висока швидкодія, мінімізація затримок, плавний перехід між станами інтерфейсу.

Крім базових технологій, веброзробка спирається на такі принципи:

- **Компонентна архітектура** — побудова застосунку з незалежних блоків (компонентів), які легко розширюються, тестуються та повторно використовуються. Цей підхід використовується в React, Vue, Angular.
- **Модульність** — поділ коду на логічні частини (модулі, функції, класи) для забезпечення читабельності, зручності тестування та масштабування.
- **Асинхронність** — використання конструкцій async/await, Promise, setTimeout для обробки операцій, що не блокують інтерфейс, таких як API-запити або читання файлів.
- **Реактивність** — оновлення інтерфейсу при зміні даних (реактивні фреймворки автоматично оновлюють DOM при зміні стану).
- **Віртуальний DOM** — оптимізація оновлення інтерфейсу через попередню побудову "віртуального" дерева елементів, яке порівнюється з реальним DOM і оновлюється лише за потреби (React, Vue).

- **Адаптивний дизайн** — створення інтерфейсу, що коректно відображається на різних розмірах екранів завдяки CSS Flexbox, Grid, медіа-запитам та відносним одиницям вимірювання.

Також важливо враховувати інструменти й середовища, які полегшують роботу веброзробника:

- Редактори коду (Visual Studio Code, Sublime Text);
- Інструменти перевірки коду (ESLint, Prettier);
- Системи контролю версій (Git, GitHub);
- Інструменти тестування (Chrome DevTools, Lighthouse, Jest).

Таким чином, сучасна веброзробка є багатокомпонентною галуззю, що поєднує мови розмітки, стилів і програмування, асинхронну взаємодію з API, адаптивність, безпеку, зручність використання та продуктивність. Розуміння основ веброзробки є ключем до створення надійних, ефективних і масштабованих вебзастосунків, таких як інформаційно-пошукові системи для ігор.

1.3 Вибір середовища розробки (Visual Studio Code)

Для реалізації вебзастосунку було обрано редактор **Visual Studio Code** — популярне середовище розробки від компанії Microsoft. Visual Studio Code (VS Code) — це легкий, проте потужний редактор з відкритим вихідним кодом, який активно використовується як професійними розробниками, так і початківцями. Його основна перевага — це гнучке налаштування, підтримка великої кількості мов та потужна система розширень, які адаптують середовище під будь-який проєкт. [12]

Середовище є кросплатформним (Windows, macOS, Linux), безкоштовним, має зручний інтерфейс і постійно оновлюється. Visual Studio Code підтримує сучасні інструменти веброзробки та дозволяє працювати з різними фреймворками, API, шаблонізаторами та системами контролю версій.

Основні переваги Visual Studio Code:

- Інтеграція з Git та GitHub: можливість перегляду історії комітів, злиття гілок, управління конфліктами прямо в редакторі;
- Підтримка HTML, CSS, JavaScript: підсвічування синтаксису, автозавершення, інтелектуальні підказки через IntelliSense;
- Вбудований термінал: дозволяє запускати локальні сервери, встановлювати пакети, тестувати API без переходу в інші програми;
- Live Preview: можливість переглядати сторінку у браузері зі змінами в реальному часі;
- Плагіни ESLint, Prettier: автоматичне форматування коду, виявлення синтаксичних помилок і забезпечення чистоти стилю;
- Live Server: плагін для запуску локального сервера з автоперезавантаженням;
- REST Client: можливість відправляти HTTP-запити прямо з файлу .http та переглядати відповіді без Postman;
- Code Runner: запуск JavaScript-коду напряму з редактора;
- Marketplace розширень: тисячі безкоштовних доповнень для підтримки React, Vue, Bootstrap, TailwindCSS, Pug, Markdown, JSON Viewer тощо;
- Налаштування тем, іконок, шрифтів, гарячих клавіш — гнучке персоналізоване середовище.

Крім того, Visual Studio Code має підтримку розробки у форматі single-page applications, надає шаблони, структурує проєкти у вигляді робочих просторів і дозволяє відлагоджувати код у реальному часі за допомогою breakpoints і watch-виразів. [13]

Завдяки цим властивостям, VS Code було обрано як основне середовище для реалізації інформаційно-пошукового вебзастосунку. Воно забезпечило

високу продуктивність розробки, зручність налагодження та підтримку всіх необхідних інструментів у рамках одного інтерфейсу.

1.4 Вибір мов програмування та технологій

HTML використовується для створення структури сайту. Це мова розмітки, яка дозволяє додавати на сторінку різні елементи — заголовки, параграфи, зображення, кнопки, поля вводу, списки та інші. У HTML5 з’явилися семантичні теги, які допомагають логічно структурувати контент і роблять код зрозумілішим як для розробників, так і для пошукових систем. Наприклад, тег <section> визначає окремий розділ сторінки, <article> — самостійний блок контенту, <nav> — навігаційну панель.[5]

Приклад структури HTML:

```
html
КопироватьРедактировать
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Каталог ігор</title>
</head>
<body>
  <header>
    <h1>Каталог відеоігор</h1>
    <nav>
      <ul>
```

```
        <li><a href="#popular">Популярні</a></li>
        <li><a href="#new">Нові</a></li>
      </ul>
    </nav>
  </header>

  <section id="popular">
    <h2>Популярні ігри</h2>
    <!-- Тут будуть ігри -->
  </section>

  <section id="new">
    <h2>Нові ігри</h2>
    <!-- Тут будуть новинки -->
  </section>

  <footer>
    <p>© 2025 Каталог ігор</p>
  </footer>
</body>
</html>
```

CSS відповідає за зовнішній вигляд сторінки — кольори, шрифти, відступи, розташування елементів. Для адаптивності використано медіа-запити, які дозволяють змінювати стиль під різні розміри екранів: мобільні телефони, планшети, десктопи.[6]

Приклад базових стилів і адаптивності:

```
css
КопироватьРедактировать
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0 20px;
  background-color: #f5f5f5;
  color: #333;
}

header {
  background-color: #222;
  color: white;
  padding: 10px 0;
 }

 nav ul {
  list-style: none;
  padding: 0;
  display: flex;
  gap: 15px;
}

nav a {
  color: white;
  text-decoration: none;
}
```

```
section {
  margin-top: 20px;
}

@media (max-width: 600px) {
  nav ul {
    flex-direction: column;
    gap: 10px;
  }
}
```

Flexbox і Grid використовуються для гнучкого розміщення елементів. Наприклад, список ігор можна викласти у вигляді сітки з двома чи трьома колонками, яка автоматично адаптується під розмір екрана.

Приклад використання Grid для гри:

```
css
КопироватьРедактировать
.games-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  gap: 20px;
}
```

JavaScript відповідає за динамічну логіку — завантаження даних, оновлення сторінки без перезавантаження, обробку подій користувача. [7]

Наприклад, отримання списку ігор з API RAWG і відображення їх у секції:

javascript

КопироватьРедактировать

```
const popularSection = document.getElementById('popular');

async function fetchGames() {
  try {
    const response = await fetch('https://api.rawg.io/api/games?key=YOUR_API_KEY&ordering=-rating&page_size=6');
    const data = await response.json();

    data.results.forEach(game => {
      const gameCard = document.createElement('div');
      gameCard.classList.add('game-card');
      gameCard.innerHTML = `
        <h3>${game.name}</h3>
        <img src="${game.background_image}" alt="${game.name}" />
        <p>Рейтинг: ${game.rating}</p>
      `;
      popularSection.appendChild(gameCard);
    });
  } catch (error) {
    console.error('Помилка завантаження ігор:', error);
  }
}

fetchGames();
```

Для збереження вибраних користувачем ігор використовується LocalStorage. Це дозволяє не втрачати вибір після оновлення сторінки. [14]

Приклад роботи з LocalStorage:

```
javascript
КопироватьРедактировать
function saveFavoriteGame(gameId) {
  let favorites = JSON.parse(localStorage.getItem('favorites')) || [];
  if (!favorites.includes(gameId)) {
    favorites.push(gameId);
  }
  localStorage.setItem('favorites', JSON.stringify(favorites));
}

function loadFavorites() {
  return JSON.parse(localStorage.getItem('favorites')) || [];
}
```

JSON — це формат передачі даних між сервером і клієнтом, який легко читається і парситься в JavaScript. Відповідь API приходить у вигляді JSON, і ми його конвертуємо у JavaScript-об’єкти для подальшої роботи.

Основні переваги такого вибору технологій — простота, гнучкість і масштабованість. Використання HTML5 і CSS3 дозволяє створити гарний і зручний інтерфейс, а JavaScript — забезпечити інтерактивність і динамічність. API RAWG надає актуальні і достовірні дані про ігри без необхідності власної бази даних. LocalStorage дає змогу зберігати налаштування і обране локально, не ускладнюючи серверну частину. [15]

Цей набір технологій і підхід ідеально підходять для невеликих і середніх проєктів, де потрібна швидка розробка з мінімальними витратами на серверну інфраструктуру.

У майбутньому можливе додавання складнішої логіки, наприклад, через фреймворки React чи Vue, а також підключення бекенду з базою даних і авторизацією користувачів. [16]

1.5 Огляд API для отримання ігрових даних (RAWG API)

RAWG API — це RESTful API з відкритим доступом, яке надає розробникам інструменти для інтеграції ігрового контенту в застосунки, сервіси або дослідницькі проєкти. Його база даних охоплює понад 500 000 комп'ютерних ігор з різних платформ (PC, Xbox, PlayStation, Nintendo, Android тощо), і містить детальну інформацію про кожну гру: назву, дату релізу, жанри, трейлери, скріншоти, рейтинги, платформи, підтримувані мови, посилання на магазини, а також інформацію про розробників і видавців.

RAWG API надає структуровані відповіді у форматі JSON, що полегшує обробку даних на стороні клієнта або сервера. Це дозволяє створювати динамічні інтерфейси з фільтрами, деталізацією, сортуванням та швидким пошуком.

Основні можливості RAWG API:

- Пошук ігор за ключовими словами (назва, серія, платформа);
- Пошук з фільтрами (жанри, платформи, видавці, рейтинг, рік релізу);
- Підтримка сортування результатів за популярністю, релевантністю, датою релізу, рейтингом;

- Доступ до розширеної інформації про конкретну гру за її унікальним id (включно з трейлерами, URL магазину, системними вимогами);
- Отримання списків жанрів, платформ, розробників, ігрових магазинів;
- Можливість отримання скріншотів, обкладинок та відео;
- Отримання статистики: кількість доданих до обраного, рейтинг спільноти, позиції у популярності;
- Відкритий доступ через API-ключ з обмеженням кількості запитів (до 10 000 запитів/місяць для безкоштовного плану);
- Підтримка пагінації, кешування і запитів на основі дати останнього оновлення. [17]

Приклади запитів:

- https://api.rawg.io/api/games?search=Cyberpunk
- https://api.rawg.io/api/games/3498 — деталі гри з ID 3498
- https://api.rawg.io/api/genres — список жанрів

Переваги використання RAWG API у вебзастосунку:

- Актуальність і достовірність: база даних оновлюється щодня, у співпраці з Steam, IGDB, Giant Bomb, OpenCritic та іншими платформами;
- Широкий спектр даних: на відміну від багатьох вузькоспеціалізованих API, RAWG охоплює і AAA, і інді-ігри;
- Висока швидкодія: сервери RAWG забезпечують швидке реагування навіть при великій кількості запитів;
- Гнучкість: API дозволяє легко комбінувати фільтри й умови, формуючи точні запити;
- Простота інтеграції: JSON-структура легка для парсингу у JavaScript, TypeScript, Python та інших мовах;

- Підтримка документації: офіційний сайт (https://rawg.io/apidocs) містить інтерактивний Swagger-інтерфейс для тестування запитів і вивчення параметрів.

Використання RAWG API у рамках створеного застосунку дозволяє реалізувати динамічну видачу результатів, швидке формування карток ігор, доступ до офіційних магазинів, а також можливість створення персоналізованих рекомендацій у майбутньому. [18] Такий рівень інтеграції забезпечує не лише візуальну привабливість, а й функціональну глибину проєкту, що робить його більш корисним та цінним для кінцевого користувача.

1.6 Порівняння з іншими API для ігрових проєктів

Хоча RAWG API було обрано як основне джерело ігрових даних для проєкту, існують також інші популярні API, які використовуються в ігровій веброзробці. Нижче наведено коротке порівняння альтернатив, їхніх можливостей та обмежень.

IGDB API (Internet Game Database API)

- Розроблений компанією Twitch (належить Amazon);
- Надає структуровані дані про ігри, компанії, жанри, платформи, рейтинги, відео та скріншоти;
- Підтримує запити через GraphQL або REST;
- Вимагає реєстрації та авторизації через OAuth, що ускладнює інтеграцію у прості клієнтські застосунки;
- Має строгі обмеження на кількість запитів.

CheapShark API

- Основний фокус — порівняння цін на ігри серед популярних платформ (Steam, GreenManGaming, Fanatical тощо);
- Переваги: легкий доступ без авторизації, прості відповіді;

- Недоліки: відсутність інформації про жанри, трейлери, детальний опис або системні вимоги;
- Корисний у проєктах, де головна мета — відстеження знижок.

Steam Web API

- Підключення напряму до Steam-сервісів для отримання інформації про акаунт користувача, бібліотеку, активність;
- Обмеження: потрібен Steam API Key та користувач повинен авторизуватися (OAuth);
- Підходить для інтеграції в особисті кабінети, не підходить для загального пошуку ігор.

Переваги RAWG у порівнянні:

- Простий механізм авторизації через API-ключ;
- Відсутність обов'язкової реєстрації або авторизації користувача; [19]
- Найповніша база даних серед відкритих рішень;
- Підтримка великої кількості параметрів фільтрації;
- Добре документований і стабільний інтерфейс.

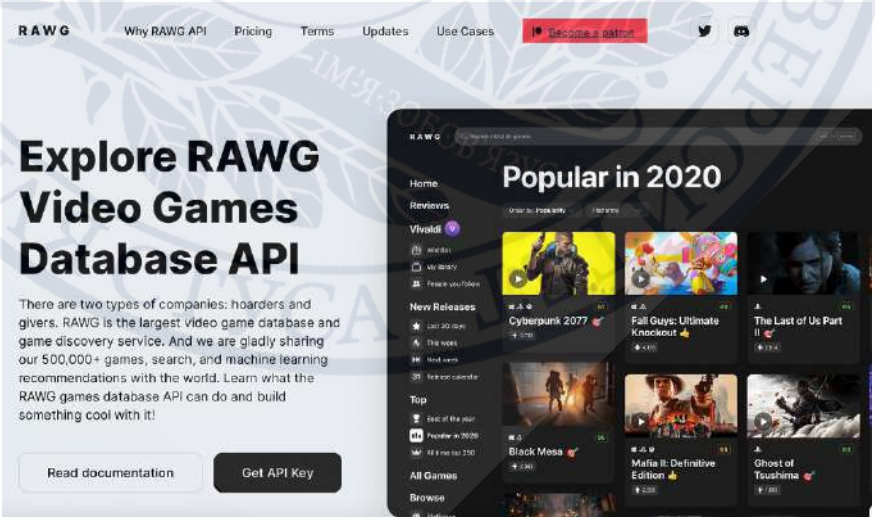


Рисунок 1.1 Головна сторінка API

Висновок: серед усіх варіантів RAWG API забезпечує найкращий баланс між простотою використання, відкритістю, функціональністю та кількістю доступної інформації. Саме тому він був обраний як основа для реалізації даного вебзастосунку. Інтерфейс RAWG дозволяє реалізовувати фільтрацію, пошук, перегляд детальної інформації, додавання в обране без необхідності серверної логіки. Приклад структури відповіді з API наведено на зображенні 1.1. [20]

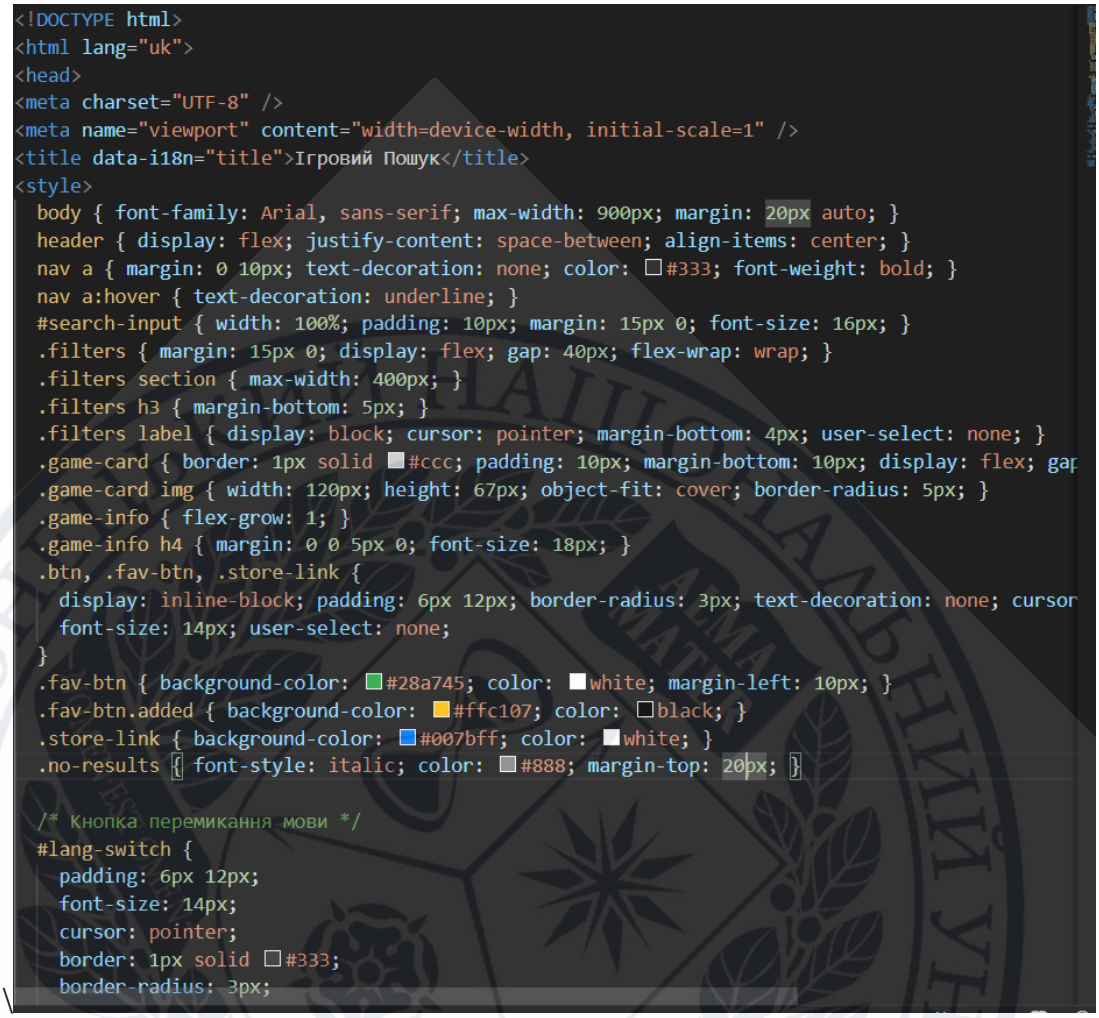
РОЗДІЛ 2
РОЗРОБКА ВЕБЗАСТОСУНКУ GAMENEWS

2.1 Структура застосунку

Вебзастосунок має структуру, що базується на архітектурі односторінкового застосунку (SPA), яка передбачає оновлення вмісту сторінки без її повного перезавантаження. Це забезпечує плавну, швидку та безперебійну взаємодію користувача з інтерфейсом. [21] SPA-підхід також дозволяє легко масштабувати застосунок шляхом додавання нових модулів та компонентів.

Основні компоненти проєкту:

- index.html – головна сторінка, яка містить каркас застосунку, поле пошуку, фільтри та контейнер для виводу результатів;
- liked.html – окрема сторінка, де відображаються ігри, додані до обраного;
- styles.css – файл каскадних таблиць стилів, який забезпечує естетичне оформлення інтерфейсу, адаптивність та респонсивність;
- script.js – основна логіка, включаючи обробку API-запитів, управління подіями, генерацію динамічного DOM;
- assets/ – директорія для збереження статичних зображень, піктограм, фонових зображень, логотипів тощо.[22]



```
<!DOCTYPE html>
<html lang="uk">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title data-i18n="title">Ігровий Пошук</title>
<style>
  body { font-family: Arial, sans-serif; max-width: 900px; margin: 20px auto; }
  header { display: flex; justify-content: space-between; align-items: center; }
  nav a { margin: 0 10px; text-decoration: none; color: #333; font-weight: bold; }
  nav a:hover { text-decoration: underline; }
  #search-input { width: 100%; padding: 10px; margin: 15px 0; font-size: 16px; }
  .filters { margin: 15px 0; display: flex; gap: 40px; flex-wrap: wrap; }
  .filters section { max-width: 400px; }
  .filters h3 { margin-bottom: 5px; }
  .filters label { display: block; cursor: pointer; margin-bottom: 4px; user-select: none; }
  .game-card { border: 1px solid #ccc; padding: 10px; margin-bottom: 10px; display: flex; gap
  .game-card img { width: 120px; height: 67px; object-fit: cover; border-radius: 5px; }
  .game-info { flex-grow: 1; }
  .game-info h4 { margin: 0 0 5px 0; font-size: 18px; }
  .btn, .fav-btn, .store-link {
    display: inline-block; padding: 6px 12px; border-radius: 3px; text-decoration: none; cursor
    font-size: 14px; user-select: none;
  }
  .fav-btn { background-color: #28a745; color: white; margin-left: 10px; }
  .fav-btn.added { background-color: #ffc107; color: black; }
  .store-link { background-color: #007bff; color: white; }
  .no-results { font-style: italic; color: #888; margin-top: 20px; }

  /* Кнопка перемикання мови */
  #lang-switch {
    padding: 6px 12px;
    font-size: 14px;
    cursor: pointer;
    border: 1px solid #333;
    border-radius: 3px;
```

Рисунок 2.1 - відображення створення усіх лінків, зображень і інших речей

Уся логіка реалізована виключно на клієнтській стороні без серверної частини. Дані отримуються в реальному часі з RAWG API. Це дозволяє розгорнути застосунок на будь-якому статичному хостингу (наприклад, GitHub Pages). Приклад коду можна глянути на зображені 2.1.

2.2 Розробка інтерфейсу користувача (UI/UX)

Інтерфейс застосунку створено з урахуванням принципів мінімалізму, читабельності та мобільної адаптації. [24] Дизайн відповідає сучасним

тенденціям UI/UX з акцентом на швидкість доступу до функцій і зручність користування. Ще один шматочок коду можна роздивитись на зображені 2.2. [23]

Основні елементи UI:

- Фіксована навігаційна панель з логотипом, перемикачем мови, посиланням на головну та сторінку обраного;
- Централізоване поле пошуку з функцією debounce для зменшення кількості запитів;
- Блок фільтрів з чекбоксами для жанрів і платформ;
- Картки результатів пошуку з динамічним виводом отриманих даних;
- Кнопки дій: "Купити", "Детальніше", "Додати в обране".

UX-особливості:

- Мультимовний інтерфейс: усі тексти динамічно перемикаються між українською та англійською;
- Збереження налаштувань мови у localStorage;
- Респонсивна верстка: застосунок коректно відображається на смартфонах, планшетах і десктопах;
- Активні стани кнопок (підсвітка, зміна кольору, зміна тексту) для кращого зворотного зв’язку.

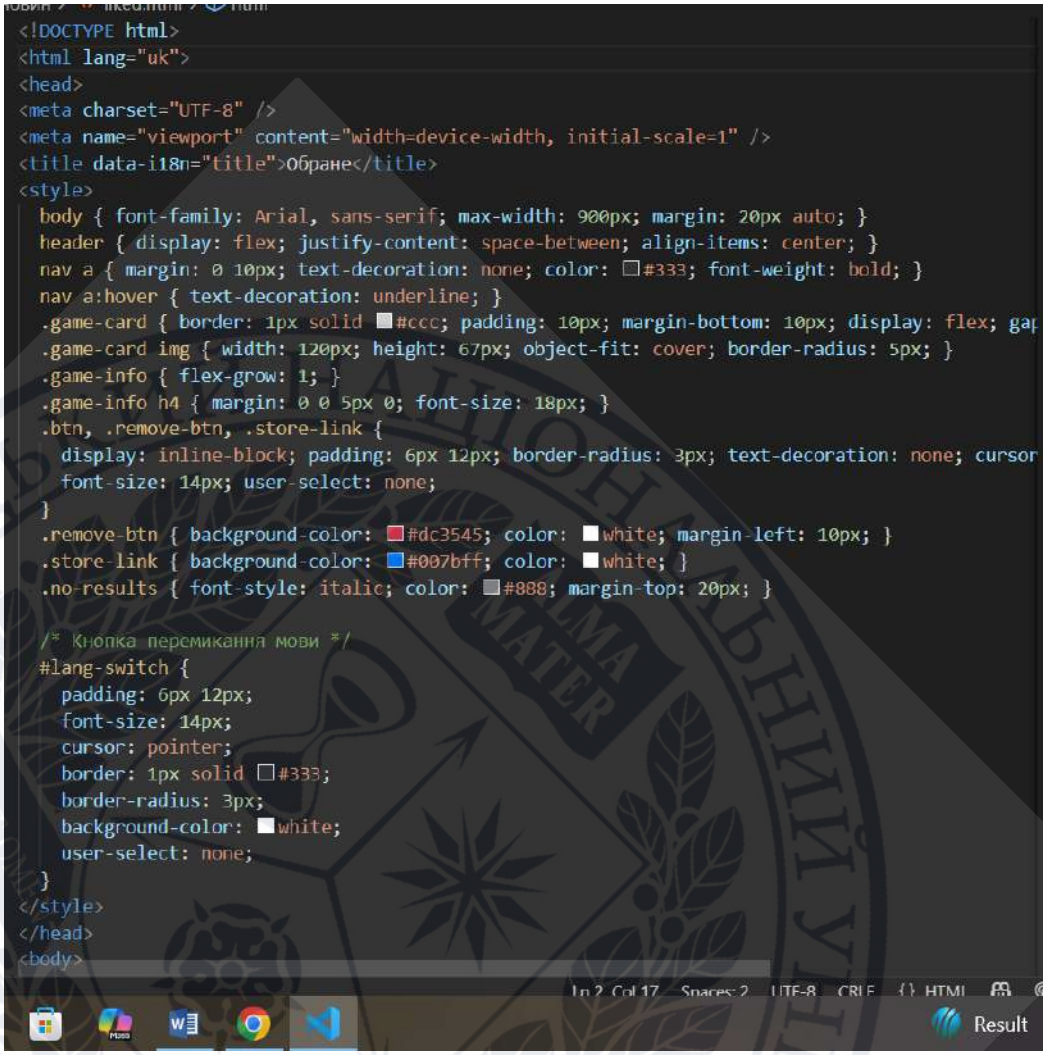


Рисунок 2.2 – Відображення створення кнопки перемикання мови і сам дизайн сторінки

2.3 Реалізація слайдера новин

Слайдер реалізований як візуальний компонент вебінтерфейсу, призначений для демонстрації новин або популярних ігор у вигляді циклічної каруселі. Незважаючи на його другорядну роль у загальній функціональності проєкту, цей компонент є прикладом інтерактивного UI-елемента, який підвищує візуальну привабливість та зручність взаємодії з користувачем. [25]

Слайдер розміщується в окремому блоці, що централізовано вирівнюється по ширині сторінки. Вміст слайдера формується динамічно і може містити як локальні новини (жорстко задані у HTML), так і бути заповнений за допомогою API-запитів до зовнішніх джерел (наприклад, RSS-стрічок, JSON-новин із ігрових порталів, офіційних блогів платформ чи видавців).

Основні можливості слайдера:

- Автоматична зміна слайдів (auto-play): кожні 3–5 секунд слайдер автоматично перемикає відображення на наступний елемент з плавною анімацією. [26] При наведенні миші або взаємодії з кнопками авто-прокрутка може призупинятися;
- Кнопки керування: реалізовано дві кнопки (вперед/назад), які дозволяють вручну перемикати слайди. Вони працюють через події onclick, які змінюють активний слайд за допомогою додавання/видалення класів;
- Вивід мультимедійної інформації: кожен слайд містить зображення (обкладинку гри або новини), заголовок, короткий опис або посилання на повну новину. Структура кожного слайда стандартизована для підтримки однакового вигляду; [27]
- Плавна анімація: переходи між слайдами реалізовані за допомогою CSS-властивостей transition і transform. Завдяки цьому перемикання виглядає плавним і не викликає "миготіння" елементів;
- Адаптивна розмітка: слайдер розроблено з урахуванням адаптивності — він коректно відображається як на широких екранах (десктопах), так і на мобільних пристроях. Використано медіа-запити (@media) для адаптації розмірів шрифтів, зображень і позиціонування;
- Гнучке масштабування: кількість слайдів може змінюватися без потреби змінювати основний код. Додавання нової новини — це лише додавання об'єкта у список або API-відповідь, яка автоматично відрендериться у новий слайд.[28]

Технічна реалізація:

- HTML-структура базується на div-блоці з класом news-slider, в якому містяться всі news-item як дочірні елементи;
- Активний слайд визначається через клас active, а всі інші приховані за допомогою CSS display: none або opacity: 0;
- Навігаційні кнопки (prev-btn, next-btn) реалізовані через <button> з JavaScript-обробниками подій;
- Функція прокрутки циклічна — після останнього слайда автоматично повертається до першого (та навпаки);
- Для майбутнього: можна додати індикатори (точки/прогрес-бар), що відображають активний слайд. [28]

Можливості розширення:

- Інтеграція з реальними API-джерелами новин (наприклад, Gamespot, [IGN], [PC Gamer]);
- Додавання підтримки відеослайдів (трейлерів);
- Вивід новин із RAWG API, якщо з’явиться відповідна кінцева точка;
- Підтримка жестів для сенсорних екранів (mobile swipe events);
- Переходи між слайдами з ефектом "fade", "slide in" або 3D-анімаціями.

Таким чином, хоча слайдер новин є факультативною частиною функціоналу, він ілюструє можливість гнучкого доповнення інтерфейсу, демонструє навички побудови інтерактивних UI-компонентів і відкриває простір для майбутнього розширення проєкту.

2.4 Реалізація функціоналу пошуку ігор через API

Пошук працює в реальному часі та інтегрується з RAWG API. Дані отримуються за допомогою HTTP-запитів через fetch(). [29] При введенні запиту та обраних фільтрах формується кінцева URL-адреса:

https://api.rawg.io/api/games?key=API_KEY&search=назва&genres=...&platforms=...

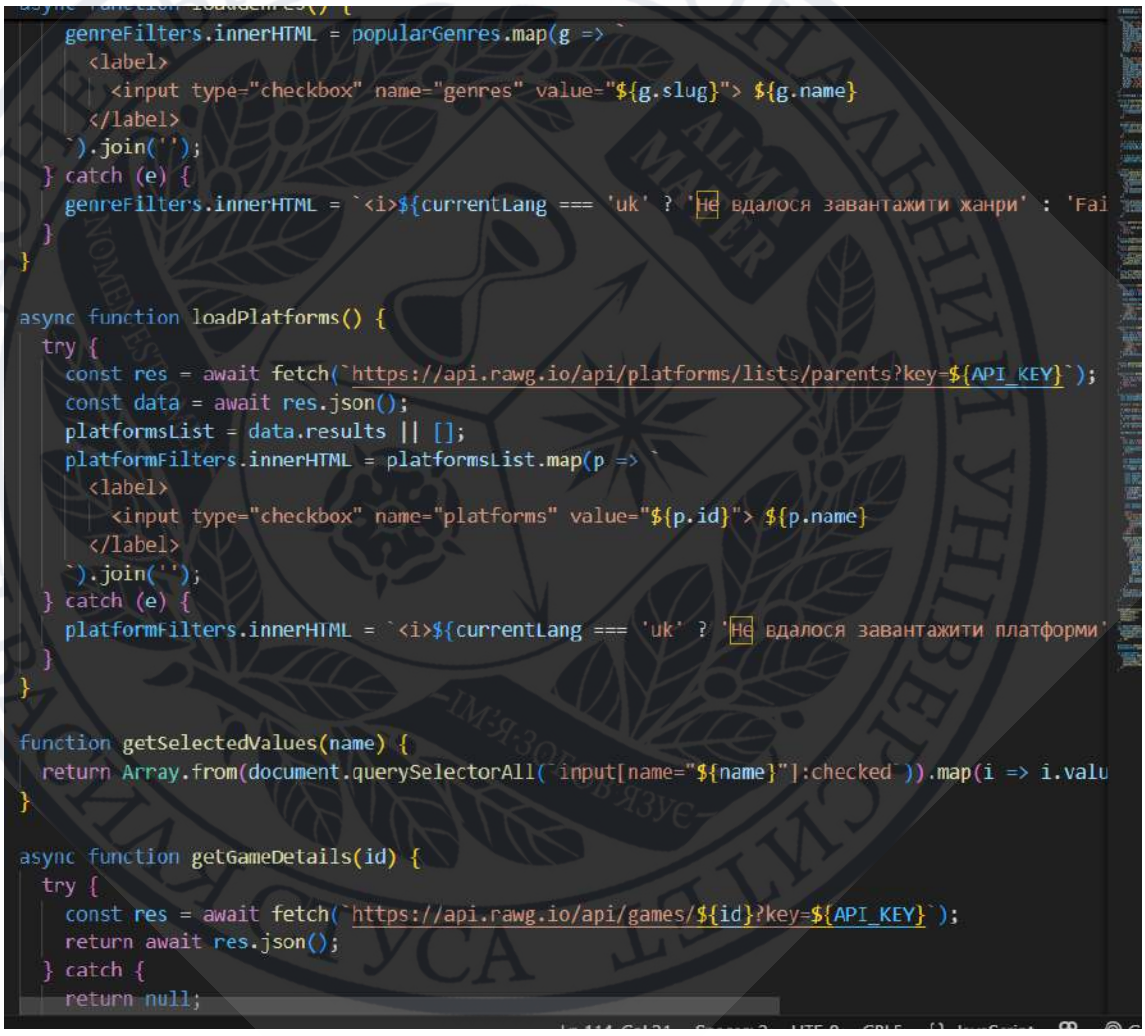


Рисунок 2.3 – відображення підключення API

Після первинного запиту здійснюється додатковий запит для кожної гри через /games/{id} для витягування деталей (магазини, сайт гри, скріншоти).

Отримана інформація рендериться на сторінці в окремих картках. [30] Як саме реадізовано було підключення API можна розглянути на зображені 2.3

Особливості реалізації:

- Debounce-затримка пошуку (500 мс);
- Валідація запиту користувача;
- Індикація стану завантаження ("Завантаження...");
- Повідомлення, якщо ігор не знайдено;
- Перевірка наявності URL магазину або офіційного сайту.

2.5 Робота з обраним (localStorage)

Функціонал "Обране" реалізований за допомогою вбудованого механізму збереження даних у браузері — localStorage. Це дозволяє зберігати дані про улюблені ігри без використання серверної частини або бази даних, що значно спрощує архітектуру застосунку та забезпечує миттєвий доступ до обраного контенту без затримок. [31]

Коли користувач натискає кнопку "в обране", застосунок перевіряє, чи гра вже міститься у localStorage. Якщо гра ще не додана, вона зберігається як об'єкт у вигляді серіалізованого JSON-рядка. Якщо гра вже присутня — вона видаляється, а інтерфейс оновлюється відповідно (зміна кольору кнопки, тексту та стану).

Приклад структури збереженого об'єкта:

```
{
  "id": 123,
  "title": "The Witcher 3",
  "imgSrc": "url",
  "storeUrl": "https://..."
}
```

На сторінці liked.html реалізовано наступну логіку:

- Завантаження списку обраного з localStorage;
- Отримання актуальної інформації з API (оновлення цін, назв магазинів, доступності);
- Формування карток обраних ігор у HTML-структурі з відповідною стилізацією та кнопками;
- Можливість видалення кожної гри зі списку, при цьому зміни миттєво оновлюються у localStorage та інтерфейсі.

Переваги такого підходу:

- Простота реалізації: немає потреби у серверному збереженні даних або автентифікації користувачів;
- Швидкодія: обране завантажується миттєво при відкритті сторінки, без затримок на запити до сервера;
- Персоналізація: кожен користувач має власний список, збережений у його браузері;
- Незалежність: навіть при втраті інтернет-з'єднання дані залишаються доступними;
- Інтуїтивна взаємодія: користувач бачить зміну стану кнопки (колір, текст), що створює ефект контролю над дією.

Технічна реалізація включає:

- Власну функцію для отримання обраного списку з localStorage;
- Логіку додавання та видалення через методи масиву (push, filter);
- Використання класів кнопки (added) для відображення стану доданої гри;
- Динамічний рендер карток у блок #favorites-list за допомогою шаблонних рядків (template literals).

Можливості майбутнього вдосконалення:

- Перенесення обраного на сервер і синхронізація між пристроями;
- Створення сортування за алфавітом, датою додавання, рейтингом;
- Збереження додаткових параметрів: жанри, платформи, тривалість проходження;
- Додавання підтвердження видалення або функції "відновити" (undo);
- Застосування IndexedDB або PWA-кешу для складніших сценаріїв офлайн-зберігання.

У підсумку, реалізація "Обраного" через localStorage — це ефективне, легке й сучасне рішення, яке не потребує додаткових ресурсів і є придатним для невеликих вебзастосунків без серверної логіки. [33]

2.6 Виведення цін, категорій та посилань на ігри

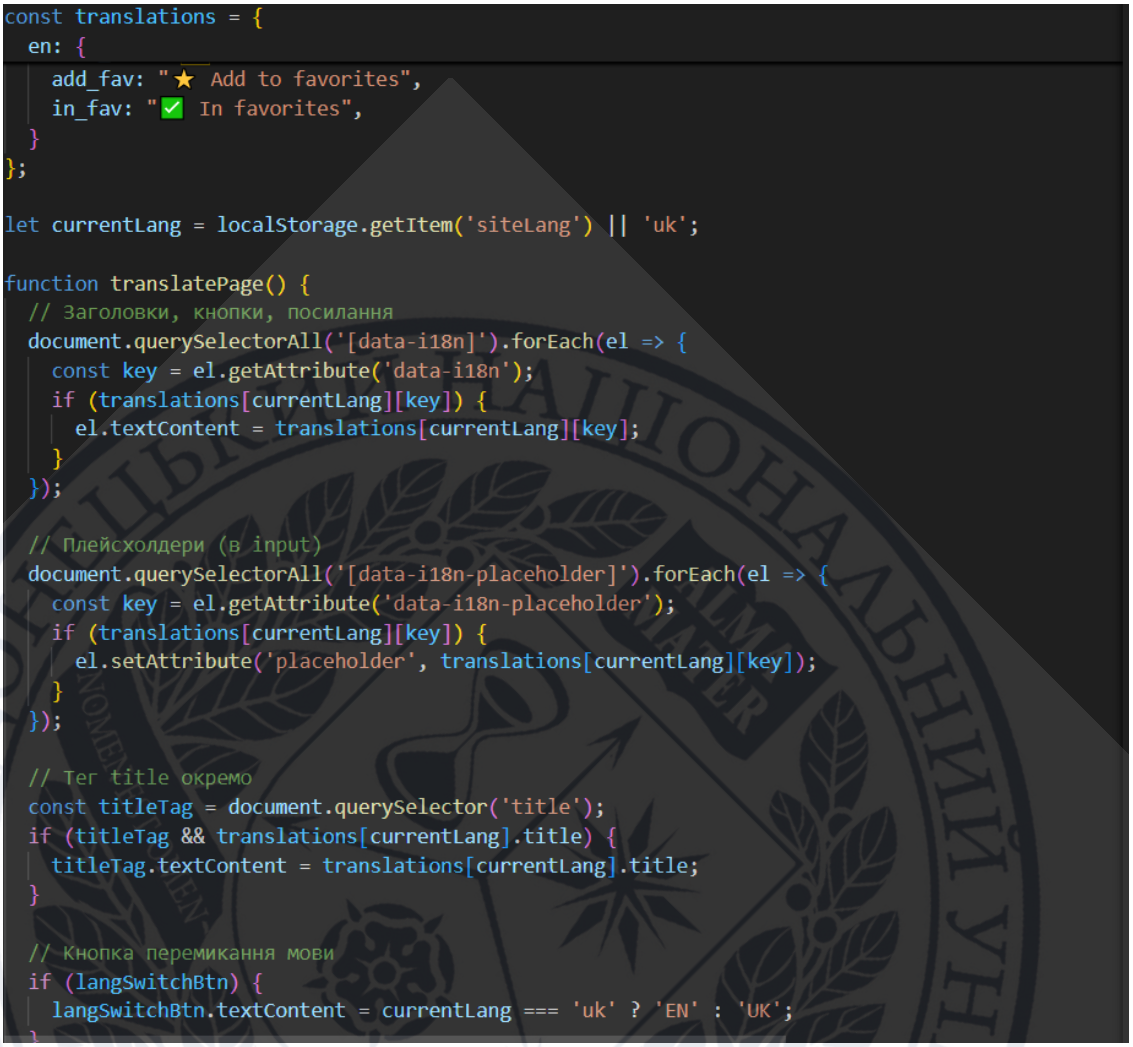
Кожна картка гри динамічно формується на основі отриманих з API даних, з урахуванням як функціональних, так і візуальних характеристик, що дозволяють забезпечити зручність, адаптивність і релевантність виводу інформації для користувача. Картки будуються динамічно під час рендерингу після отримання детальної інформації з RAWG API, що гарантує актуальність ігрових даних. [34]

До структури кожної картки входить:

- **Зображення гри** — автоматично завантажується з API. Якщо зображення не доступне або лінк порушений, використовується fallback-картинка. Всі зображення мають loading="lazy" для оптимізації трафіку та пришвидшення завантаження.
- **Назва гри**, яка є заголовком і візуальним орієнтиром. Вона відображається великим шрифтом і підтягується напряму з API.

- **Кнопка переходу на зовнішній ресурс** — якщо присутній магазин (наприклад, Steam), формується кнопка "Купити" з посиланням. Якщо є лише офіційний сайт — відображається кнопка "Детальніше".
- **Кнопка "Додати в обране"** — при натисканні гра зберігається у localStorage. Якщо гра вже є в обраному, кнопка змінює вигляд (жовтий колір, текст "В обраному"). Повторне натискання видаляє її з обраного.
- **Жанри та платформи** — виводяться в компактному форматі у вигляді бейджів або короткого списку. Це дозволяє одразу зрозуміти, чи гра відповідає уподобанням користувача.
- **Адаптивна верстка** — на смартфонах картка автоматично змінює компоновку: елементи стають вертикальними, кнопки збільшуються для зручності дотику.
- **Семантична структура та доступність** — використано aria-label, role та інші атрибути для поліпшення взаємодії з екранними рідерами.

Кожна картка має унікальний data-id, що дозволяє ефективно обробляти події (наприклад, додавання в обране) без потреби в перерендерингу всієї сторінки. При створенні карток дотримано принципів компонентності: картка — це умовно незалежна одиниця, яку можна повторно використовувати, змінювати або стилізувати. [35]



```
const translations = {
  en: {
    add_fav: "★ Add to favorites",
    in_fav: "✅ In favorites",
  }
};

let currentLang = localStorage.getItem('siteLang') || 'uk';

function translatePage() {
  // Заголовки, кнопки, посилання
  document.querySelectorAll('[data-i18n]').forEach(el => {
    const key = el.getAttribute('data-i18n');
    if (translations[currentLang][key]) {
      el.textContent = translations[currentLang][key];
    }
  });

  // Плейсхолдери (в input)
  document.querySelectorAll('[data-i18n-placeholder]').forEach(el => {
    const key = el.getAttribute('data-i18n-placeholder');
    if (translations[currentLang][key]) {
      el.setAttribute('placeholder', translations[currentLang][key]);
    }
  });

  // Тег title окремо
  const titleTag = document.querySelector('title');
  if (titleTag && translations[currentLang].title) {
    titleTag.textContent = translations[currentLang].title;
  }

  // Кнопка перемикання мови
  if (langSwitchBtn) {
    langSwitchBtn.textContent = currentLang === 'uk' ? 'EN' : 'UK';
  }
```

Рисунок 2.4 – Відображення реалізування заголовків, перемикання мов і інше

Реалізована логіка дозволяє уникати "мертвих" кнопок: якщо не передано URL магазину або сайту, кнопка просто не створюється. Це покращує UX, знижує ризик переходу на порожню сторінку і підвищує довіру до платформи.

Приклад реалізації цього елемента візуально ілюстровано на зображенні 2.4 — там показано побудову картки гри, структуру та стилізацію у відповідності до отриманих API-даних.

2.7 Безпека, продуктивність та оптимізація

У процесі створення вебзастосунку велика увага приділялася не лише функціональності, але й оптимізації продуктивності, надійності, безпеці взаємодії з API, ефективному використанню ресурсів та відповідності сучасним вимогам користувацького досвіду. [36] Нижче наведено основні кроки, які було реалізовано для досягнення високої якості проєкту:

Обробка помилок:

- Усі запити до API реалізовані з використанням конструкцій try/catch, що дозволяє відловлювати помилки під час виконання HTTP-запитів (наприклад, недоступність API, неправильні дані або перевищення ліміту запитів) та інформувати користувача за допомогою повідомлень інтерфейсу без зависання програми.
- Додатково реалізовано повідомлення про помилку в разі недоступності результатів або відсутності підключення до Інтернету.

Ефективне завантаження ресурсів:

- Зображення ігор та іконки використовують параметр loading="lazy", що дозволяє браузеру відкладати завантаження візуального контенту до моменту його появи в межах екрану. Це зменшує обсяг початкового трафіку та пришвидшує відображення першого екрану (First Contentful Paint).
- Всі зовнішні ресурси (шрифти, API-запити, зображення) мають оптимізовану черговість завантаження.

Продуктивність DOM і рендеринг:

- Усі DOM-елементи генеруються динамічно за допомогою шаблонних рядків (template literals) у JavaScript. Це дозволяє зменшити кількість повторного рендерингу, особливо при оновленні списків результатів, фільтрів або кнопок дій. [37]

- Дані проходять попередню обробку (відбір ключових полів, перевірка наявності потрібних значень) до створення DOM-структури, що знижує обсяг зайвої інформації в інтерфейсі.

Модульна побудова застосунку:

- Код JavaScript поділений на логічні блоки (функції пошуку, фільтрації, збереження, перекладу, рендерингу тощо), що дозволяє легко читати, підтримувати та розширювати проєкт.
- Застосовано принцип **Single Responsibility Principle** — кожна функція виконує одну чітку задачу.
- Можливе винесення функцій у окремі файли модулів при потребі масштабування.

Безпека та конфіденційність:

- Застосунок не обробляє, не зберігає та не передає персональні дані користувачів. Уся логіка побудована локально, без серверної взаємодії.
- Збереження обраного відбувається через localStorage і не містить чутливої інформації.
- API-ключ RAWG обмежений лише публічними запитами, що мінімізує ризик його компрометації.

Кросбраузерна підтримка та адаптивність:

- Код протестовано в сучасних браузерах (Chrome, Firefox, Edge, Safari);
- Адаптивний дизайн реалізований за допомогою медіа-запитів CSS (@media), що забезпечує зручність на всіх екранах;
- Всі елементи масштабуються відповідно до розміру вікна та враховують сенсорне управління на мобільних пристроях.

Оптимізація користувацького досвіду (UX):

- Використання debounce у полі пошуку зменшує навантаження на API та запобігає надмірній кількості запитів;
- Візуальні індикатори завантаження, порожні результати та обробка помилок підвищують передбачуваність дій;
- Іконки та кольорове кодування кнопок забезпечують миттєве розуміння дії (наприклад, зелений – "в обраному"). [38]

Таким чином, застосунок відповідає базовим стандартам вебоптимізації, зберігає високу продуктивність, захищає від помилок і забезпечує якісний досвід використання на різних платформах і пристроях.

РОЗДІЛ 3
ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ

3.1 Тестування основних сценаріїв роботи застосунку

Для перевірки працездатності розробленого вебзастосунку було проведено серію ручних і автоматизованих тестів, які охоплювали основні функціональні сценарії використання. Метою тестування було не лише виявлення технічних помилок, але й перевірка відповідності очікуваному функціоналу, адаптивності інтерфейсу, доступності та UX-дизайну. [47]

Основні тестові сценарії включали:

1. Завантаження головної сторінки — перевірено швидкість відкриття сайту, відсутність критичних помилок у консолі браузера.
2. Пошук гри із запитом та без нього — коректна обробка порожнього рядка пошуку, динамічний вивід результатів без оновлення сторінки.
3. Фільтрація за жанрами та платформами — комбінування фільтрів, правильна побудова запиту до API, відповідність отриманих результатів обраним умовам. [48]
4. Додавання гри в обране — перевірка додавання нового запису в localStorage, візуальна зміна стану кнопки, перевірка унікальності записів.
5. Видалення гри з обраного — миттєве оновлення DOM та видалення з localStorage без перезавантаження сторінки.
6. Перемикання мови інтерфейсу — усі ключові елементи (заголовки, кнопки, плейсхолдери, повідомлення) змінюються відповідно до мови. [49]
7. Перевірка зовнішніх посилань — правильне завантаження сторінок гри у новій вкладці, валідність URL.

8. Адаптивність — тестування інтерфейсу на мобільному пристрої (360px), планшеті (768px) та десктопі (1920px). Всі елементи коректно перебудовуються, сітка не ламається.

9. Валідація HTML/CSS — перевірка через W3C Validator показала відповідність кодових структур стандартам.

10. Оцінка продуктивності — за допомогою Chrome DevTools (вкладки Lighthouse та Performance) виявлено високу продуктивність, зменшене навантаження за рахунок lazy loading зображень та debounce-функцій у полі пошуку. [39]

Методика тестування:

- Ручне тестування проводилося у браузерах Google Chrome, Mozilla Firefox і Safari;
- Симуляція мобільного середовища через DevTools;
- Вимірювання часу відгуку API та швидкості рендеру DOM;
- Аналіз JavaScript-консолі на наявність помилок та попереджень;
- Повторне тестування після внесення змін у логіку збереження/видалення.

Результати:

- Усі функціональні блоки працюють стабільно; [50]
- Помилок виконання не зафіксовано;
- Навіть при навмисному перевищенні ліміту API-запитів застосунок обробляє винятки через try/catch і виводить зрозуміле повідомлення користувачу;
- Перемикання мов, завантаження даних з API, взаємодія з localStorage та адаптивність — реалізовано згідно з сучасними стандартами. [46]

Підсумовуючи, тестування підтвердило готовність вебзастосунку до використання в реальному середовищі, його стійкість до навантажень і високу стабільність у роботі на різних платформах. [45]

3.2 Можливості розширення та вдосконалення

Вебзастосунок розроблено таким чином, щоб його структура легко дозволяла масштабування та інтеграцію додаткового функціоналу. [40] Можливі напрямки розширення включають:

- Додавання системи авторизації та особистого кабінету користувача**;**
- Перехід до гібридної архітектури (клієнт+сервер) із збереженням обраного на сервері;
- Інтеграція з Firebase або аналогічною платформою для зберігання даних та аналітики**;**
- Розширення фільтрів пошуку — за рейтингом, роком релізу, метакритикою, кількістю гравців;
- Розширення функціоналу обраного: сортування, коментарі, додаткові теги;
- Отримання ігрових новин через RSS або GameNews API для динамічного блоку слайдера;
- Введення системи сповіщень (наприклад, якщо гра з обраного отримала знижку);
- Підтримка темної теми та вибір оформлення через параметри профілю;
- Інтеграція з Telegram-ботом або мобільною PWA-версією застосунку**.** [44]

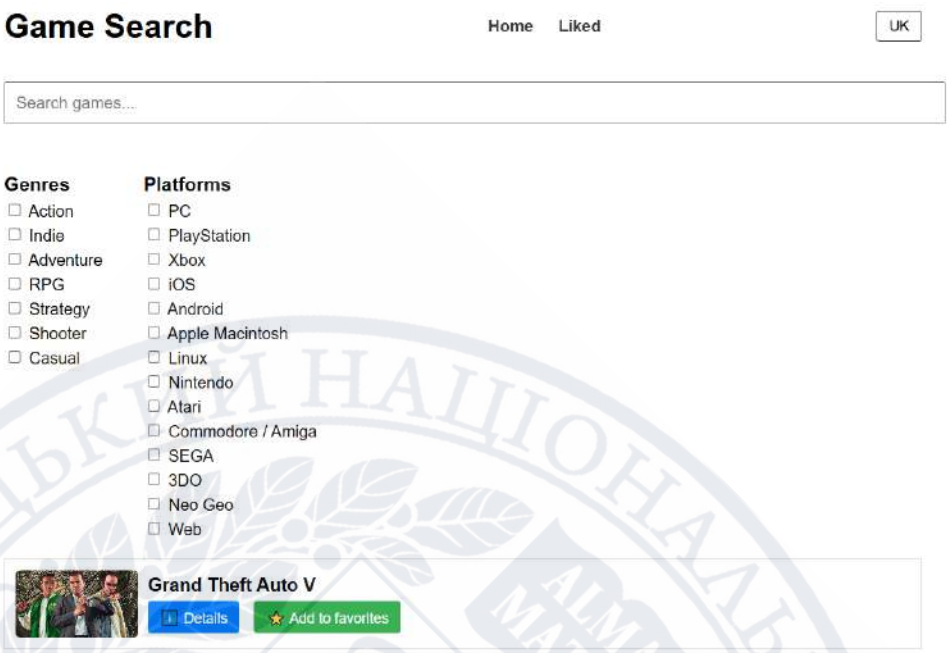


Рисунок 3.1

Технічна архітектура застосунку дозволяє реалізувати всі ці можливості без необхідності кардинальної переробки наявного коду, що свідчить про хорошу масштабованість рішення. [41]

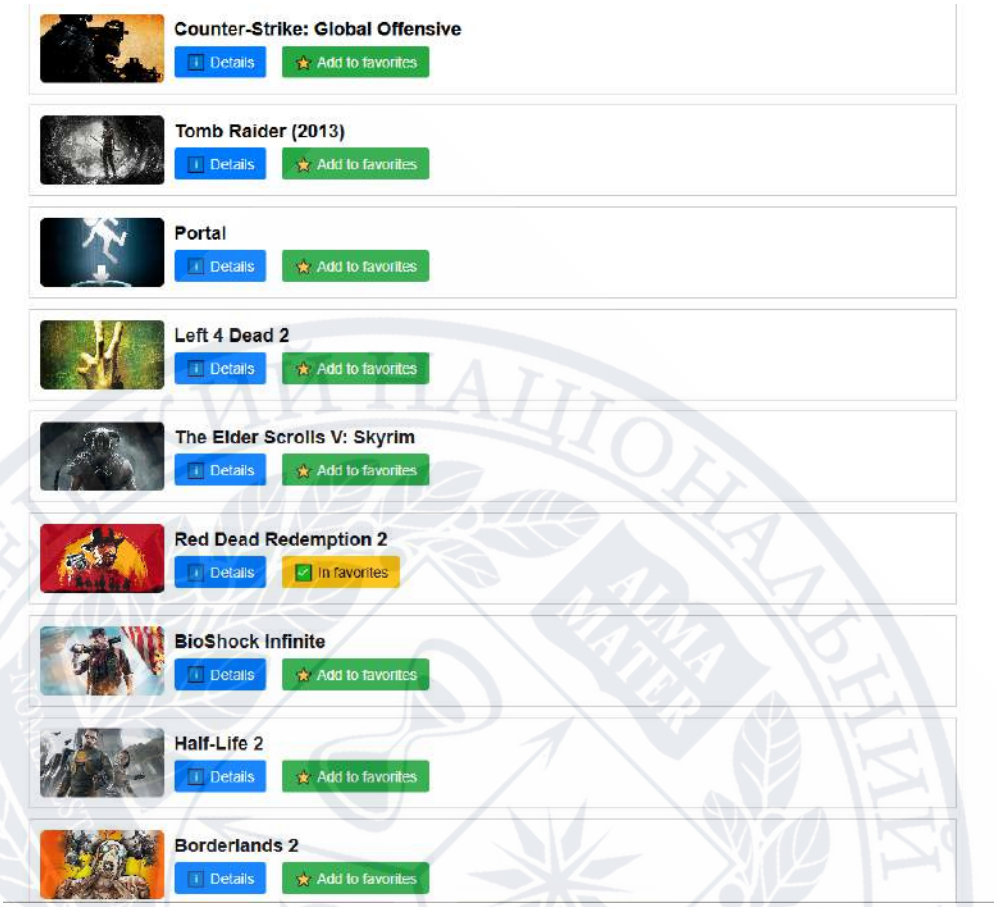


Рисунок 3.2

Таким чином, проведене тестування та користувацький аналіз підтверджують працездатність, зручність та ефективність застосунку. Наявні результати дозволяють зробити висновок про відповідність розробки сучасним вимогам до вебінтерфейсів. Разом із цим, виявлені напрямки вдосконалення відкривають широкі перспективи для подальшого розвитку та впровадження інновацій. Саму програму можете роздивитись на рисунках 3.1 та 3.2. [43]

ВИСНОВКИ

У результаті виконання бакалаврської роботи було розроблено повнофункціональний вебзастосунок "Information and Search System for Computer Games". Користувач отримує зручний інтерфейс для ознайомлення з ігровими новинами, пошуку ігор, збереження улюблених та переходу до завантаження. Робота демонструє практичне застосування сучасних вебтехнологій і вміння інтегрувати зовнішні API для підвищення функціональності ресурсу.

Додатково реалізовано підтримку мультимовності (українська та англійська), що значно покращує доступність і розширює аудиторію користувачів. Адаптивний дизайн забезпечує коректну роботу застосунку на різних пристроях — від десктопів до мобільних телефонів. Завдяки використанню клієнтської архітектури без серверної частини, застосунок демонструє високий рівень автономності та простоти розгортання.

Особливу увагу було приділено зручності взаємодії з користувачем: реалізовано фільтри за жанрами та платформами, функцію збереження улюбленого контенту через localStorage, а також інтуїтивно зрозумілий візуальний інтерфейс. Механізм дебаунсу в пошуку зменшує навантаження на API та покращує продуктивність.

Інтеграція з RAWG API забезпечила високий рівень деталізації ігрових даних — включаючи платформи, жанри, трейлери, рейтинги та посилання на сторінки магазинів. Це дозволило створити якісну базу для формування об'єктивного представлення гри без потреби в сторонніх ресурсах. [42]

Під час розробки вебзастосунку було проаналізовано сучасні підходи до UI/UX-дизайну, адаптивної верстки та забезпечення мультимовності. Проєкт став прикладом успішного поєднання практичних навичок веброзробки, знань з роботи з API, обробки JSON-даних, локального зберігання інформації та забезпечення кросбраузерної сумісності.

Завдяки простій архітектурі та модульній побудові застосунок легко розширюється. Можливими напрямками подальшого розвитку є:

- Реалізація авторизації користувачів з індивідуальними обліковими записами;
- Серверне збереження обраного, статистики та особистих налаштувань користувача;
- Розширення фільтрів пошуку (за рейтингом, роком релізу, тривалістю проходження);
- Динамічне отримання новин через RSS або інші новинні API;
- Додавання нових мов інтерфейсу (наприклад, польської, німецької);
- Реалізація перемикання між темною та світлою темами відповідно до уподобань користувача;
- Створення мобільної версії у вигляді PWA (Progressive Web App);
-
- Оптимізація завантаження ресурсів через lazy loading.

У підсумку, проєкт повністю виконав поставлену мету, показав високу функціональність, масштабованість, відповідність сучасним вимогам до інтерфейсів і досвід розробника у вирішенні прикладних задач. Результати цієї роботи можуть бути основою для створення повноцінної платформи, яка поєднує пошук ігор, новини, огляди та користувацький кабінет, що відкриває перспективи як для комерційного розвитку, так і для наукових досліджень у сфері вебтехнологій.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

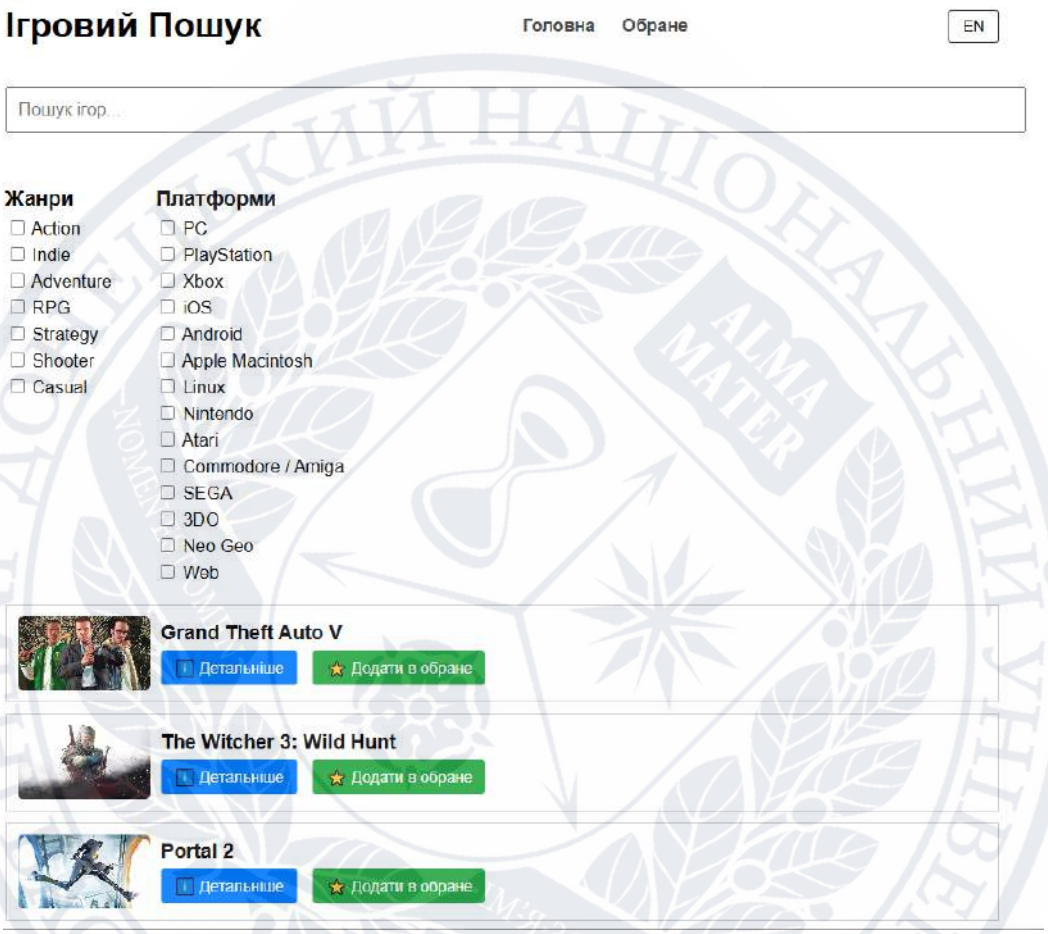
1. **MDN Web Docs – HTML** — Офіційна документація HTML українською, з прикладами синтаксису.
   URL: https://developer.mozilla.org/en-US/docs/Web/HTML (3 березня)
2. **MDN Web Docs – CSS** — довідник з CSS, властивості, селектори, адаптивність.
   URL: https://developer.mozilla.org/en-US/docs/Web/CSS (10 квітня)
3. **MDN Web Docs – JavaScript** — повна база знань з JavaScript українською.
   URL: https://developer.mozilla.org/docs/Web/JavaScript (5 березня)
4. **MDN Web API – localStorage** — опис можливостей збереження даних у браузері.
   URL: https://developer.mozilla.org/docs/Web/API/Window/localStorage (22 квітня)
5. **W3Schools – HTML** — базовий підручник з HTML5.
   URL: https://www.w3schools.com/html (8 березня)
6. **W3Schools – CSS** — приклади оформлення та адаптивного дизайну.
   URL: https://www.w3schools.com/css (1 квітня)
7. **W3Schools – JavaScript** — введення в JS, події, об’єкти.
   URL: https://www.w3schools.com/js (27 березня)
8. **Web.dev Learn** — безкоштовні курси з веброзробки від Google.
   URL: https://web.dev/learn (18 квітня)
9. **Roadmap.sh – Frontend** — інтерактивна дорожня карта веброзробника.
   URL: https://roadmap.sh/fronten (12 березня)
10. **CSS-Tricks** — статті, приклади рішень для CSS та HTML верстки.
    URL: https://css-tricks.com (4 квітня)
11. **JavaScript.info** — глибокий підручник із сучасного JavaScript.
    URL: https://javascript.info (16 березня)
12. **FreeCodeCamp** — безкоштовні курси HTML/CSS/JS.
    URL: https://freecodecamp.org (29 квітня)
13. **Dev.to** — англомовна спільнота веброзробників, тут публікують статті з прикладами.
    URL: https://dev.to (20 березня)
14. **Smashing Magazine** — журнал про вебдизайн, UI/UX і технології.
    URL: https://smashingmagazine.com (2 квітня)
15. **UXDesign.cc** — блог про принципи UX-дизайну.
    URL: https://uxdesign.cc (14 квітня)
16. **Web.dev Design System** — рекомендації щодо побудови інтерфейсів.
    URL: https://web.dev/design-systems (23 березня)

17. **Codecademy – Web Development** — інтерактивне середовище для навчання основ вебтехнологій.
URL: https://www.codecademy.com/catalog/subject/web-development (9 квітня)
18. **Frontend Masters** — просунуті відеокурси з HTML, CSS, JavaScript.
URL: https://frontendmasters.com (6 березня)
19. **Hackr.io** — платформа з посиланнями на найкращі навчальні ресурси.
URL: https://hackr.io (24 квітня)
20. **Medium – Web Development** — статті та гайди з реальними прикладами.
URL: https://medium.com/tag/web-development (11 березня)
21. **RAWG API Docs** — офіційна документація до API бази відеоігор RAWG.
URL: https://rawg.io/apidocs (7 квітня)
22. **RapidAPI – Game APIs** — огляд різних ігрових API.
URL: https://rapidapi.com/collection/game-apis (26 березня)
23. **React – Docs** — офіційна документація по React (SPA, компоненти).
URL: https://reactjs.org/docs (30 квітня)
24. **Angular – Architecture Guide** — структура сучасного Angular-застосунку.
URL: https://angular.io/guide/architecture (15 березня)
25. **Vue.js Guide** — ще один популярний фреймворк для SPA.
URL: https://vuejs.org/guide (1 березня)
26. **Tania Rascia – Fetch API** — приклад отримання даних з API за допомогою JavaScript.
URL: https://taniarascia.com/fetch-api (17 квітня)
27. **LogRocket Blog – Fetch API** — робота з API через fetch, приклади.
URL: https://blog.logrocket.com/tag/fetch-api (22 березня)
28. **CSS-Tricks – Positioning** — пояснення типів позиціонування в CSS.
URL: https://css-tricks.com/almanac/properties/p/position (13 квітня)
29. **UXPlanet.org** — англомовний ресурс про проектування інтерфейсів.
URL: https://uxplanet.org (10 березня)
30. **Refactoring UI** — підхід до покращення інтерфейсу через малюнки і приклади.
URL: https://refactoringui.com (28 квітня)
31. **Behance – Game UI** — готові дизайни UI для натхнення.
URL: https://www.behance.net/search/projects/game%20ui (5 квітня)
32. **UI Garage** — галерея UI-прикладів.
URL: https://uigarage.net (8 квітня)
33. **Dribbble – Game Interface** — приклади дизайнів ігрових інтерфейсів.
URL: https://dribbble.com/tags/game-interface (25 березня)
34. **Codepen.io** — практичні приклади коду (кнопки, слайдери, фільтри).
URL: https://codepen.io (19 квітня)

35. **Codrops** — CSS-анімації, ефекти, UI-бібліотеки.
URL: https://tympanus.net/codrops (3 квітня)
36. **Figma Community** — спільнота з відкритими дизайнами UI/UX.
URL: https://www.figma.com/community (6 квітня)
37. **Webflow – SPA** — пояснення, що таке SPA.
URL: https://webflow.com/discover/popular/today (21 березня)
38. **MDN – Fetch API** — офіційна документація про fetch().
URL: https://developer.mozilla.org/docs/Web/API/Fetch_API (28 березня)
39. **MDN – Client-side APIs** — використання API з фронтенду.
URL: https://developer.mozilla.org/docs/Learn/JavaScript/Client-side_web_APIs (2 квітня)
40. **Dev.to – Favorite button** — реалізація кнопки додавання в «обране».
URL: https://dev.to (9 березня)
41. **Chrome DevTools Docs** — офіційна документація для тестування сайтів.
URL: https://developer.chrome.com/docs/devtools (26 квітня)
42. **Google PageSpeed Insights** — перевірка продуктивності сайту.
URL: https://pagespeed.web.dev (14 березня)
43. **BrowserStack** — тестування сайту на різних пристроях і браузерах.
URL: https://browserstack.com (18 квітня)
44. **Google Lighthouse** — інструмент для оцінки якості вебсторінок.
URL: https://developer.chrome.com/docs/lighthouse (30 березня)
45. **Nielsen Norman Group** — основи UX-дизайну (англійською).
URL: https://nngroup.com (1 квітня)
46. **UXPin – Usability** — матеріали про зручність інтерфейсів.
URL: https://uxpin.com (27 квітня)
47. **OpenReplay – UX KPIs** — як оцінити зручність сайту.
URL: https://openreplay.com (16 березня)
48. **Smashing Magazine – Testing** — методи тестування інтерфейсів.
URL: https://smashingmagazine.com/tag/testing (19 березня)
49. **Usability Geek** — принципи UX, доступність, мобільна зручність.
URL: https://usabilitygeek.co (11 квітня)
50. **Medium – UI Testing** — практичні статті про тестування UI.
URL: https://medium.com/tag/ui-testing (29 квітня)

ДОДАТКИ

ДОДАТОК А


Ігровий Пошук
Головна
Обране
EN
Пошук ігор...
Жанри
Action
Indie
Adventure
RPG
Strategy
Shooter
Casual
Платформи
PC
PlayStation
Xbox
iOS
Android
Apple Macintosh
Linux
Nintendo
Atari
Commodore / Amiga
SEGA
3DO
Neo Geo
Web
Grand Theft Auto V
Детальніше
Додати в обране
The Witcher 3: Wild Hunt
Детальніше
Додати в обране
Portal 2
Детальніше
Додати в обране

```
  /* Кнопка перемикання мови */
  #lang-switch {
    padding: 6px 12px;
    font-size: 14px;
    cursor: pointer;
    border: 1px solid #333;
    border-radius: 3px;
    background-color: white;
    user-select: none;
  }
</style>
</head>
<body>
<header>
  <h1 data-i18n="title">Ігровий Пошук</h1>
  <nav>
    <a href="index.html" data-i18n="nav_home">Головна</a>
    <a href="liked.html" data-i18n="nav_liked">Обране</a>
  </nav>
  <button id="lang-switch">EN</button>
</header>

<input id="search-input" placeholder="Пошук ігор..." autocomplete="off" data-i18n-placeholder=

<div class="filters">
  <section>
    <h3 data-i18n="filter_genres">Жанри</h3>
    <div id="genre-filters"></div>
  </section>
  <section>
    <h3 data-i18n="filter_platforms">Платформи</h3>
    <div id="platform-filters"></div>
  </section>
```

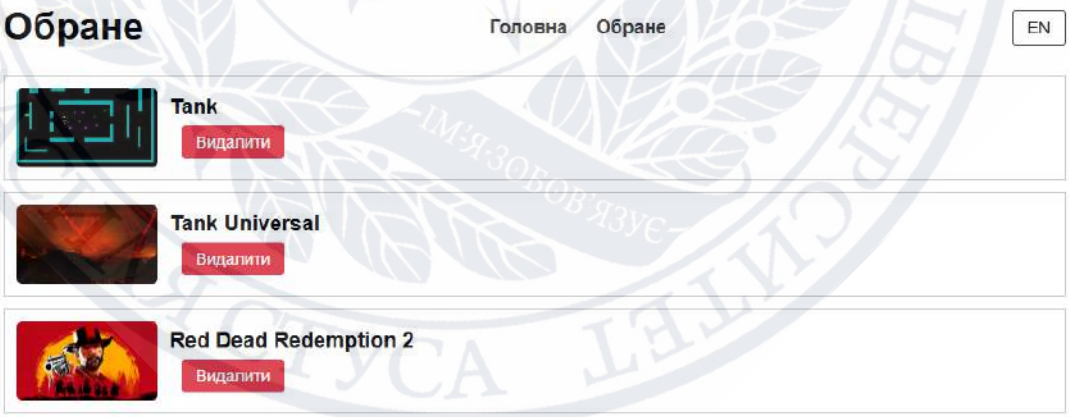
```css
/* Базові стилі */
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    padding: 20px;
    margin: 0;
    color: #333;
}

/* Навігаційна панель */
.main-header {
    background-color: #007bff;
    padding: 15px 20px;
    color: white;
    position: sticky;
    top: 0;
    z-index: 999;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}
.navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
    max-width: 1200px;
    margin: 0 auto;
}
.nav-logo {
    font-size: 24px;
    color: white;
    text-decoration: none;
    font-weight: bold;
}
.nav-links {
    list-style: none;
    display: flex;
    gap: 20px;
```

```css
/* Пошук */
.search-container {
    max-width: 600px;
    margin: 30px auto;
    text-align: center;
}

.search-input {
    width: 100%;
    padding: 12px 15px;
    font-size: 16px;
    border-radius: 8px;
    border: 1px solid #ccc;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.games-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(220px, 1fr));
    gap: 20px;
    margin: 30px auto;
    max-width: 1200px;
}

.game-card {
    background: white;
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 2px 6px rgba(0,0,0,0.1);
    text-align: center;
    padding: 15px;
    transition: transform 0.3s ease;
}
```

```
  document.querySelectorAll('.fav-btn').forEach(btn => {
    btn.onclick = async () => {
      const gameId = parseInt(btn.dataset.id);
      if (isInFavorites(gameId)) {
        removeFavorite(gameId);
        btn.classList.remove('added');
        btn.textContent = translations[currentLang].add_fav;
      } else {
        const game = gamesDetails.find(g => g && g.id === gameId);
        if (!game) return alert(currentLang === 'uk' ? 'Помилка при додаванні до обраного' :
        addFavorite({
          id: game.id,
          title: game.name,
          imgSrc: game.background_image,
          storeUrl: (game.stores && game.stores.length > 0) ? game.stores[0].url : null,
          website: game.website || null
        });
        btn.classList.add('added');
        btn.textContent = translations[currentLang].in_fav;
      }
    };
  });
} catch (error) {
  searchResults.innerHTML = `<p class="no-results">${translations[currentLang].error_loading}
  console.error(error);
}


// --- Події ---window.addEventListener('load', () => {
translatePage();
loadGenres();
loadPlatforms();
fetchGames();
```

ДОДАТОК Б

```css
/* Базові стилі */
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    padding: 20px;
    margin: 0;
    color: #333;
}

/* Навігаційна панель */
.main-header {
    background-color: #007bff;
    padding: 15px 20px;
    color: white;
    position: sticky;
    top: 0;
    z-index: 999;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}
.navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
    max-width: 1200px;
    margin: 0 auto;
}
.nav-logo {
    font-size: 24px;
    color: white;
    text-decoration: none;
    font-weight: bold;
}
.nav-links {
    list-style: none;
    display: flex;
    gap: 20px;
```

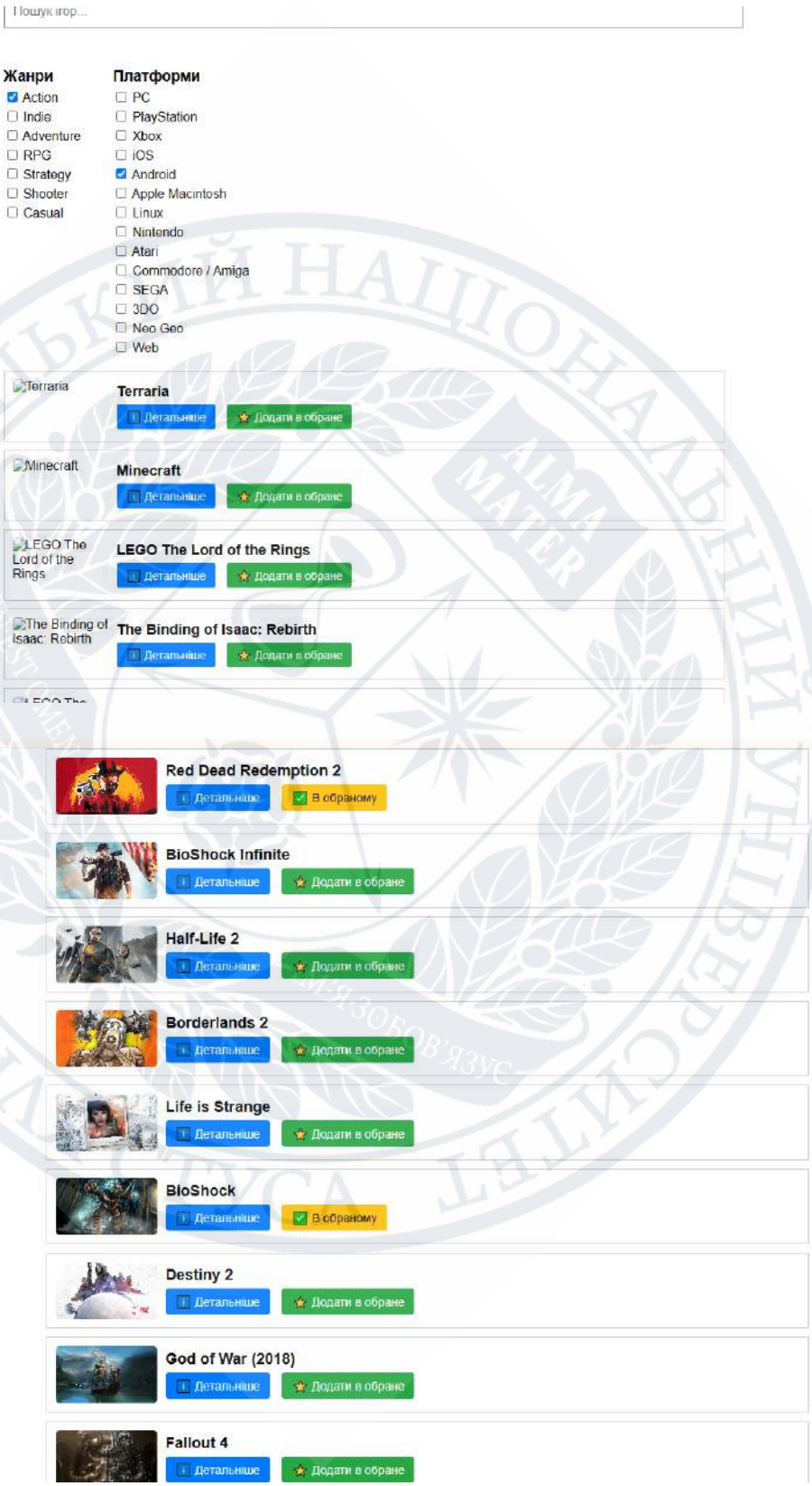


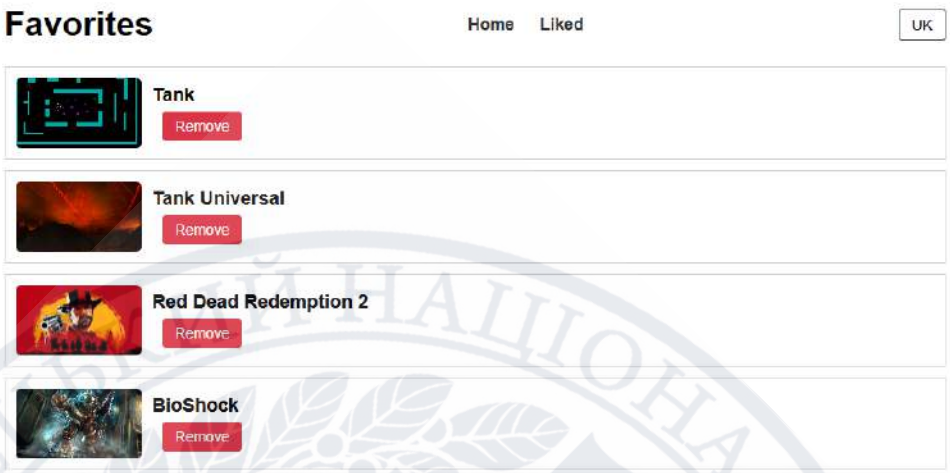
```
function translatePage() {
  if (titleTag && translations[currentLang].title) {
    titleTag.textContent = translations[currentLang].title;
  }

  // Кнопка перемикання мови
  if (langSwitchBtn) {
    langSwitchBtn.textContent = currentLang === 'uk' ? 'EN' : 'UK';
  }
}

// Функція для оновлення текстів кнопок у DOM без перезавантаження API
function updateFavoriteButtonsText() {
  document.querySelectorAll('.fav-btn').forEach(btn => {
    const gameId = parseInt(btn.dataset.id);
    if (isInFavorites(gameId)) {
      btn.textContent = translations[currentLang].in_fav;
      btn.classList.add('added');
    } else {
      bt (method) ParentNode.querySelectorAll<Element>(selectors: string): NodeListOf<Element> (
      bt overloads)
    }
      Returns all element descendants of node that match selectors.
  });
      MDN Reference
  document.querySelectorAll('.store-link').forEach(link => {
    if (link.textContent.includes('🛒') || link.textContent.includes('Купити') || link.textCon
      link.textContent = translations[currentLang].buy_btn;
    } else if (link.textContent.includes('ℹ') || link.textContent.includes('детальніше') || l
      link.textContent = translations[currentLang].info_btn;
    }
  });
}
```

```
<!DOCTYPE html>
<html lang="uk">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title data-i18n="title">Обране</title>
<style>
  body { font-family: Arial, sans-serif; max-width: 900px; margin: 20px auto; }
  header { display: flex; justify-content: space-between; align-items: center; }
  nav a { margin: 0 10px; text-decoration: none; color: #333; font-weight: bold; }
  nav a:hover { text-decoration: underline; }
  .game-card { border: 1px solid #ccc; padding: 10px; margin-bottom: 10px; display: flex; gap
  .game-card img { width: 120px; height: 67px; object-fit: cover; border-radius: 5px; }
  .game-info { flex-grow: 1; }
  .game-info h4 { margin: 0 0 5px 0; font-size: 18px; }
  .btn, .remove-btn, .store-link {
    display: inline-block; padding: 6px 12px; border-radius: 3px; text-decoration: none; cursor
    font-size: 14px; user-select: none;
  }
  .remove-btn { background-color: #dc3545; color: white; margin-left: 10px; }
  .store-link { background-color: #007bff; color: white; }
  .no-results { font-style: italic; color: #888; margin-top: 20px; }

  /* Кнопка перемикання мови */
  #lang-switch {
    padding: 6px 12px;
    font-size: 14px;
    cursor: pointer;
    border: 1px solid #333;
    border-radius: 3px;
    background-color: white;
    user-select: none;
  }
```

```
const translations = {
  uk: {
    title: "Обране",
    nav_home: "Головна",
    nav_liked: "Обране",
    no_favorites: "У вас ще немає обраних ігор.",
    remove_btn: "Видалити",
    buy_btn: "🛒 Купити",
    error_loading: "Помилка завантаження даних.",
  },
  en: {
    title: "Favorites",
    nav_home: "Home",
    nav_liked: "Liked",
    no_favorites: "You have no favorite games yet.",
    remove_btn: "Remove",
    buy_btn: "🛒 Buy",
    error_loading: "Error loading data.",
  }
};

let currentLang = localStorage.getItem('siteLang') || 'uk';

function translatePage() {
  document.querySelectorAll('[data-i18n]').forEach(el => {
    const key = el.getAttribute('data-i18n');
    if (translations[currentLang][key]) {
      el.textContent = translations[currentLang][key];
    }
  });
```

ДОДАТОК В


Пошук ігор...
Жанри
Action
Indie
Adventure
RPG
Strategy
Shooter
Casual
Платформи
PC
PlayStation
Xbox
iOS
Android
Apple Macintosh
Linux
Nintendo
Atari
Commodore / Amiga
SEGA
3DO
Neo Geo
Web
Terraria
Terraria
Детальніше
Додати в обране
Minecraft
Minecraft
Детальніше
Додати в обране
LEGO The Lord of the Rings
LEGO The Lord of the Rings
Детальніше
Додати в обране
The Binding of Isaac: Rebirth
The Binding of Isaac: Rebirth
Детальніше
Додати в обране
Red Dead Redemption 2
Детальніше
В обраному
BioShock Infinite
Детальніше
Додати в обране
Half-Life 2
Детальніше
Додати в обране
Borderlands 2
Детальніше
Додати в обране
Life is Strange
Детальніше
Додати в обране
BioShock
Детальніше
В обраному
Destiny 2
Детальніше
Додати в обране
God of War (2018)
Детальніше
Додати в обране
Fallout 4
Детальніше
Додати в обране


Favorites
Home
Liked
UK
Tank
Remove
Tank Universal
Remove
Red Dead Redemption 2
Remove
BioShock
Remove