МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії

програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ІНТЕРНЕТ-МАГАЗИНУ ВІДЕОІГОР

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Олександр МАТВІЄНКО**

«__» _____ 20__ р.

Керівник роботи

PhD, доцентка

_____ **Катерина АНТІПОВА**

«__» _____ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Матвієнко Олександр

1. Тема кваліфікаційної роботи 'Вебзастосунок інтернет-магазину відеоігор' затверджена наказом ректора ЧНУ ім. Петра Могили № _349_ від «26» _грудня_ 2025 р.

2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.

3. Очікуваним результатом роботи є вебзастосунок інтернет-магазину продажу та покупки відеоігор, що забезпечує такі функціональні можливості: зручний процес перегляду каталогу цифрових товарів, їх пошуку та фільтрації за поданими даними, додавання даних товарів до кошику, з можливістю видалення їх, подальшим оформленням замовлення для користувачів, також створенням акаунтів чи можливістю увійти до них за допомогою поданих інструментів, можливість

додавати цифрові товари для продавця, а також надавати можливості адміністратору керувати асортиментом товарів, створеними замовленнями та зареєстрованими користувачами.

4. Перелік питань, що підлягають розробці:

– провести аналіз сучасних вебзастосунків для продажу відеоігор;

– дослідити сучасні технології та засоби розробки вебзастосунків;

– зробити проєктування та моделювання вебзастосунку;

– розробити структуру базу даних для зберігання інформації про користувачів, товару та замовлення;

– реалізувати функціональні можливості для додавання, створення та видалення зареєстрованих користувачів;

– реалізувати функціональні можливості для створення каталогу товарів, систему кошика та взаємодії з нею та оформлення замовлення зареєстрованих користувачів;

– реалізувати та забезпечити адаптивний інтерфейс сторінок вебзастосунку для користування користувачем;

– зробити тестування функціоналу вебзастосунка.

5. Перелік графічних матеріалів: презентація.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «_24_» ____січня_____ 2026_ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **Вебзастосунок інтернет-магазину відеоігор**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	20.01.2026	24.01.2026	Виконано
2.	Огляд літератури за темою роботи	26.01.2026	21.02.2026	Виконано
3.	Складання календарного плану КБР	22.02.2026	25.02.2026	Виконано
4.	Аналіз предметної області	26.02.2026	06.03.2026	Виконано
5.	Розробка проєктних рішень	07.03.2026	15.03.2026	Виконано
6.	Моделювання та конструювання ПЗ	16.03.2026	28.03.2026	Виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	29.03.2026	30.05.2026	Виконано
8.	Відгук керівника КБР	26.05.2026	01.06.2026	Виконано
9.	Оформлення КБР та презентації	26.04.2026	05.05.2026	Виконано
10.	Попередній захист	26.05.2026	26.05.2026	Виконано
11.	Рецензування	28.04.2026	19.05.2026	Виконано
12.	Завершення оформлення КБР та презентації	01.06.2026	10.06.2026	Виконано
13.	Захист кваліфікаційної роботи			

Здобувач _____ **Олександр МАТВІЄНКО**

«__» _____ 20__ р.

Керівник роботи

PhD, доцентка

Катерина АНТІПОВА

_____

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Вебзастосунок інтернет-магазину відеоігор»

Здобувач 408 гр.: Матвієнко Олександр

Керівник: Phd, доцентка Антіпова Катерина

Актуальність теми зумовлена розвитком інформаційних технологій у сфері продажу надає можливості переходу бізнесу на більш розширену можливість продажу своїх послуг через онлайн-середовища на більшу аудиторію користувачів на глобальному рівні. Існуючі вебзастосунки для продажу відеоігор не завжди забезпечують зручність та зрозумілість користування покупцеві чи продавцю, персоналізації рекомендованого контенту, доступність інформації про продукт чи її прозорість для покупця.

Об’єктом роботи є процес електронного продажу та покупки ліцензій та фізичних копій відеоігор.

Предметом роботи є інструменти розробки вебзастосунку для продажу відеоігор.

Метою роботи є розробка вебзастосунку автоматизації продажу відеоігор для покращення процесів продажу та покупки цифрового товару.

Робота складається зі вступу, аналітичного розділу з оглядом предметної області, розділу проєктування системи, розділу програмної реалізації (з використанням React, NodeJS, SQL Server) та висновків.

У вступі обґрунтовано актуальність теми, сформульовано мету, завдання, об’єкт та предмет дослідження роботи. У першому розділі проведено системний аналіз предметної області, розглянуто існуючи аналоги програмного забезпечення та сформульовані завдання до системи, що розробляється. У другому розділі були сформульовані специфікації вимог та описані побудовані діаграми класів та послідовності. У третьому розділі описано реалізацію архітектури системи, обґрунтовано вибір обраних інструментів для розробки рішення та описано інтерфейси ПЗ. В останньому розділі описано процес створення та тестування розробленого ПЗ. У висновках було підсумовувано результати виконаної роботи,

підтверджено виконання всіх поставлених завдань та досягнення мети дослідження роботи.

Розроблений продукт може бути впроваджений самостійний комерційний веб-майданчик продажу відеоігор на локальному або онлайн ринку.

КБР викладена на _84_ сторінки, містить _4_ розділи, _37_ ілюстрацій, _9_ таблиці, _20_ джерел в переліку посилань.

Ключові слова: база даних, вебзастосунок, інтернет-магазин, клієнт-серверна архітектура, NodeJS, React, Socket, SPA, SQL.

ASTRACT

of the Bachelor's Thesis

«Video game online store web application»

Student of group 408: Matviienko Oleksandr

Supervisor: Phd, associate professor Antipova Katerina

Relevance of the topic: The development of information technologies in the field of sales provides businesses with opportunities to expand their services by moving to online environments and reaching a larger audience of users on a global level. Existing web applications for selling video games do not always ensure convenience and clarity of use for buyers or sellers, personalization of recommended content, availability of product information, or transparency of such information for the customer.

Object of the study: the process of electronic sale and purchase of licenses and physical copies of video games.

Subject of the study: tools for developing a web application for selling video games.

Purpose of the study: to develop a web application for selling video games in order to improve the processes of selling and purchasing digital goods.

The work consists of an introduction, an analytical section with a review of the subject area, a system design section, a software implementation section (using React, NodeJS, SQL Server), and conclusions.

The developed product can be implemented as an independent commercial platform for the local market.

The work consists of an introduction, an analytical section with a review of the subject area, a system design section, a software implementation section (using React, NodeJS, SQL Server), and conclusions.

The introduction substantiates the relevance of the topic and formulates the purpose, objectives, object, and subject of the research. The first section presents a system analysis of the subject area, reviews existing software analogues, and formulates the tasks for the system being developed. The second section defines the requirement specifications and describes the developed class and sequence diagrams. The third section describes the

implementation of the system architecture, justifies the choice of the selected development tools, and presents the software interfaces. The last section describes the process of developing and testing the implemented software. The conclusions summarize the results of the work performed, confirm the completion of all assigned tasks, and demonstrate the achievement of the research objectives.

The developed product can be implemented as an independent commercial web platform for selling video games in the local or online market.

The bachelor's thesis is presented on _84_ pages and contains _4_ sections, _37_ pictures, _9_ tables, and _20_ sources in the list of references.

Keywords: client–server architecture, database, NodeJS, online store, React, Socket, SPA, SQL, web application.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API – application programming interface (прикладний програмний інтерфейс)

CRUD – create, read, update, delete (створити, прочитати, оновити, видалити)

CSS – cascading style sheets (каскадні таблиці стилів)

DLC – downloadable content (завантажуваний вміст)

DRM – digital rights management (керування цифровими правами)

HTML5 – hypertext markup language (мова розмітки гіпертексту)

HTTP – hypertext transfer protocol (протокол передачі гіпертексту)

JSON – JavaScript object notation (текстовий формат обміну даних)

JWT – JSON web token (веб-токен формату JSON)

REST API – representational state API

SPA – single page application (односторінковий вебзастосунок)

SQL – structure query language (мова структурованих запитів)

URL – uniform resource locator (уніфікований локатор ресурсу)

VR – virtual reality (віртуальна реальність)

XML – exstensible markup language (розширювання мова розмітки)

ПЗ – програмне забезпечення

ВСТУП

Актуальність теми кваліфікаційної бакалаврської роботи зумовлена розвитком інформаційних технологій у сфері продажу, що надає можливості переходу бізнесу на більш розширену можливість продажу своїх послуг через онлайн-середовища на більшу аудиторію користувачів на глобальному рівні. Одним із багатонадійних галузей для розвитку торгівлі в цифровому середовищі є розробка вебзастосунку для продажу відеоігор. Даний напрям є одним із тих, попит на котрий, постійно зростає через популяризацію розваг в цифровому середовищі та розширення можливостей продажу та покупки через інтернет-середовище. Існуючі вебзастосунки для продажу відеоігор не завжди забезпечують зручність та зрозумілість користування покупцеві чи продавцю, персоналізації рекомендованого контенту, доступність інформації про продукт чи її прозорість для покупця.

Об’єктом роботи є процес електронного продажу та покупки ліцензій та фізичних копій відеоігор.

Предметом роботи є інструменти розробки вебзастосунку для продажу відеоігор.

Метою роботи є розробка вебзастосунку автоматизації продажу відеоігор для покращення процесів продажу та покупки цифрового товару.

Для досягнення визначеної мети необхідно вирішити такі завдання:

– провести аналіз сучасних вебзастосунків для продажу відеоігор;
– дослідити сучасні технології та засоби розробки вебзастосунків;
– зробити проєктування та моделювання вебзастосунку;
– розробити структуру базу даних для зберігання інформації про користувачів, товару та замовлення;
– реалізувати функціональні можливості для додавання, створення та видалення зареєстрованих користувачів;

– реалізувати функціональні можливості для створення каталогу товарів, систему кошика та взаємодії з нею та оформлення замовлення зареєстрованих користувачів;

– реалізувати додатковий функціонал для перевірки властивостей та відповідностей можливості ролей;

– реалізувати функціонал для отримання, надсилання та зберігання користувацький та сторонніх відгуків та оцінок;

– реалізувати та забезпечити адаптивний інтерфейс сторінок вебзастосунку для користування користувачем;

– зробити тестування функціоналу вебзастосунка.

1 АНАЛІЗ СФЕРИ ПРОДАЖУ ВІДЕОІГОР

1.1 Об’єкт і предмет роботи

Розвиток продажу товарів у електронній сфері дуже сильно пов’язана з розвитком інформаційних технологій, що суттєво впливають на можливості розповсюдження способу показу та продажу будь-якого цифрового контенту, від фізичних товарів, як наприклад телефони чи комп’ютери, так і цифрові товари, як наприклад фотографії чи відеозаписи, через можливості цифрового каналу. Якщо, наприклад, раніше основним способом, через котрий покупці могли здійснювати покупки тільки у фізичних специфічних магазинах, то зараз набувають поширення цифрові платформи, що дозволяють користувачам оформлювати та здійснювати покупки миттєво та отримувати доступ до продукту безпосередньо після оплати чи доставки.

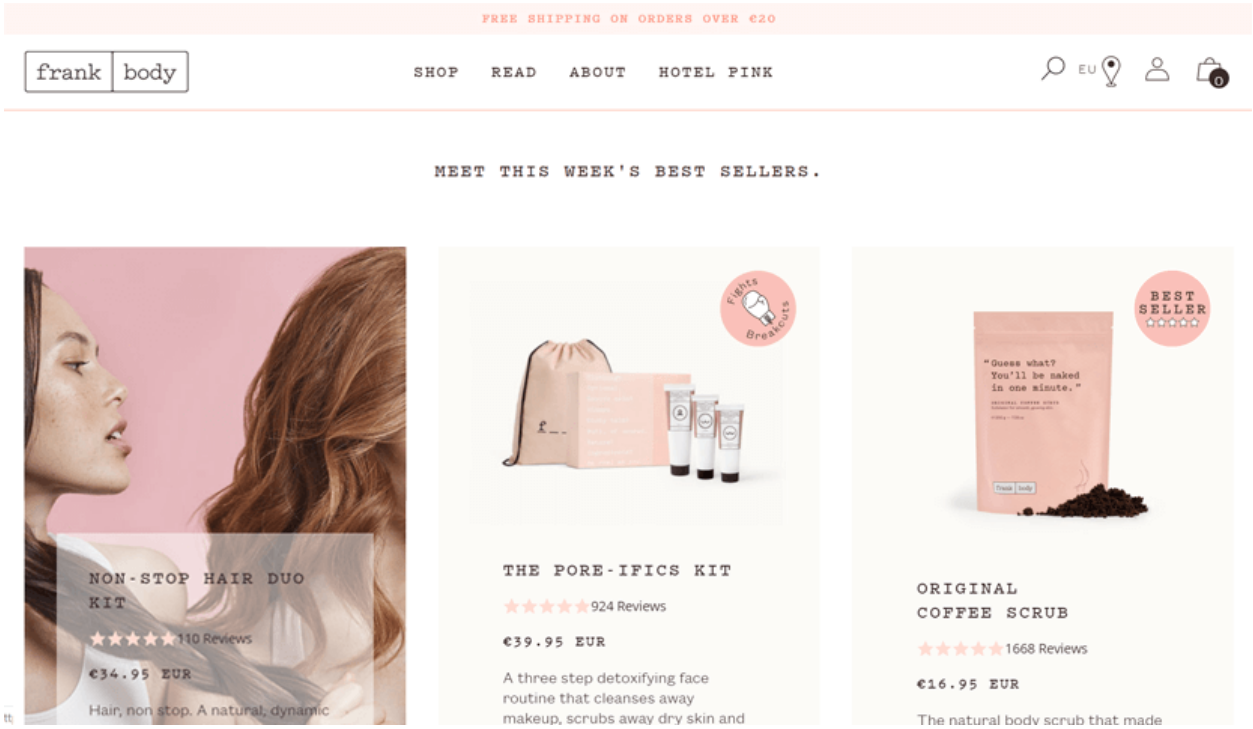

Рисунок 1.1 – Приклад електронного магазину

Об’єктом у даній роботі є процес електронного продажу та покупки ліцензій і фізичних копій відеоігор. Даний процес описує повний цикл взаємодії користувача/покупця із системою магазину: від моменту пошуку, вибору та огляду

продукту, до моменту його придбання, оформлення та отримання. У випадку можливостей цифрових товарів, що передбачає миттєве надання доступу покупцеві ліцензії товару або акаунту, в іншому випадку, для фізичних товарів – це організація доставки та логістики продавцем.

Особливістю об’єкту роботи є комбінованість особливостей різних сфер та можливостей, оскільки буде поєднуватись елементи взаємодії та зв’язку інформаційних технологій, платіжних сервісів, логістики та взаємодія з користувачем/покупцем. Також, основними аспектами, на котрі потрібно буде приділити багато уваги, це забезпечення безпеки даних елементів магазину, наприклад користувачів, а саме персональні дані покупців чи продавців, та товарів.

Предметом роботи є інструменти та технології розробки вебзастосунку для продажу електронних товарів. До них належать:

– мова програмування (JavaScript, SQL, тощо);

– фреймворки та бібліотеки (для клієнтської та серверної частини проєкту);

– системи управління базами даних;

– підходи до проєктування архітектури програмного забезпечення;

– засоби інтеграції з зовнішніми сервісами (зокрема платіжними системами);

– інструменти для забезпечення безпеки та продуктивності.

Актуальність теми зумовлена розвитком інформаційних технологій у сфері продажу, що надає можливості переходу бізнесу на більш розширену можливість продажу своїх послуг через онлайн-середовища на більшу аудиторію користувачів на глобальному рівні, та необхідністю створення ефективних програмних рішень, котрі здатні забезпечити даний процес на швидкому, доступному, безпечному та зручному для користувачеві рівні, враховуючи всі потрібні сучасні вимоги покупців та продавців на ринку електронної комерції. Через це, готовий продукт роботи повинен мати всі функціональні можливості для забезпечення потреб, як і цільової аудиторії покупців, так і продавців.

Система електронної комерції, в даному випадку продажу відеоігор, є складною, за структурою, інформаційною системою, що включає низку взаємопов’язаних між собою компонентів.

1.1.1 Аналіз структурної організації системи роботи

До основних структурних елементів інформаційної системи можливо віднести:

– клієнтська частина (frontend-частина): забезпечує взаємодії між користувачем/покупцем та системою через можливості вебінтерфейсу. Даний елемент відповідає за можливості відображення каталогу товарів, обробку дій користувача/покупця чи продавця та формування, потрібних для подальшох роботи, запитів до серверної частини проєкту;

– серверна частина (backend-частина): забезпечує реалізацію бізнес-логіки всій системи, обробки запитів клієнтської частини, надісланих від користувачів/покупців чи продавців, взаємодії з базою даних через запити, можливостей та потреб функціювання авторизації, взаємодії з товарами, обробки замовлень та платежів;

– база даних: забезпечує зберігання інформації та даних, у вигляді таблиці, про користувачів, товари, платежі, замовлення, відгуки, та інші елементи магазину. Доступ до елементів даних, а саме отримання інформації, чи внесення змін, забезпечується через можливості запитів SQL;

– платіжні системи: зовнішні системи, що забезпечують можливості проведення фінансових транзакцій між користувачем/покупцем та магазином. Підключення систем відбувається через даний API до серверної частини проєкту;

– система безпеки: системи, що включають механізми аутентифікації, шифрування даних та захисту від спроб несанкціонованих доступу до них.

Через можливу реалізацію узгодженості даних компонентів, елементи, котрих створюють основу для подальшого визначення вимог до реалізації системи електронної комерції.

1.1.2 Аналіз функціональних особливостей системи роботи

Функціональні особливості вебзастосунку проєкту визначаються вимогами як і користувачів/покупців та продавців, так і потребами у модерації вебзастосунку та бізнес-логіки електронної комерції, особливо сторонніх API. До основних функцій, котрі задовольняють потреби у функціюванні вебзастосунку на обох сторонах, можливо віднести:

– авторизація та реєстрація аккаунту: забезпечує створення нового облікового запису користувача та можливість увійти в даний запис, вийти з нього, чи видалити даний запис за допомогою взаємодії функціоналу вебзастосунку. Також надає доступ до подальших персоналізованих функцій, що потребують обліковий запис;

– каталог товарів: забезпечує можливість зображення переліку доступних товарів, в даному випадку відеоігор чи пов’язані товари-елементи (завантажуваний контент чи саундтрек). Також надає можливість функціюванню зручній обробки виводу потрібних товарів користувачеві/покупцеві на клієнтський частині можливостями;

– пошук: забезпечує можливість швидко знаходити товари за потрібним запитом. В залежності від можливостей вебзастосунку, можливо знаходити потрібний товар за частиною назви чи наявності слів в описі товару;

– фільтри: забезпечує можливість швидко знаходити товари за даними в інтерфейсі елементами для фільтрації, наприклад товар повинен мати деякі елементи зі списку, даному та обраними користувачем;

– оформлення замовлення: забезпечує можливість для оформлення покупки, заповнюючи потрібні дані та товари користувачем, та в подальшому підтвердження замовлення, за допомогою можливостей клієнтської частини та серверної частини проєкту;

– оплата: забезпечує можливість реалізації функції (чи імітацію) оплати за допомогою сторонніх API;

– управління замовленнями: забезпечує можливість перегляду історії та деталі покупок для подальшої модерації;

– відгуки та рейтинги: забезпечує можливість користувачам, котрі придбали та мають продукт, формувати та залишати текстові відгуки з можливістю оцінки через елементи інтерфейсу вебзастосунку.

Також особливу увагу слід приділити до підтримці типів товарів, а саме цифрових та фізичних, оскільки йде ускладнення системи через різні особливості типів товарів. Цифрові товари мають перевагу у миттєвій доставці, тобто покупець купує товар, оформлює покупки та після підтвердження оплати користувач отримує відразу доступ до ліцензії товару, що потребує мінімальну логістику між покупцем та магазином. З іншого боку, фізичні товари, навпаки, потребують необхідність обробки адреси доставки магазином, взаємодії з поштовими службами та контроль статусу доставки до отримання товару користувачем, що значно ускладнює логістику та потребує від магазину підтримки різних сценаріїв обробки замовлень.

1.2 Огляд і аналіз сучасного стану програмних рішень

У сучасних умовах вже існують програмні рішення для питання продажу товарів-відеоігор, які відрізняються за функціональністю, архітектурою та бізнес-моделлю. Для більш детального аналізу та подальшого формування завдань, доцільно розглянути відомі приклади платформ окремо, виділити їх недоліки, котрі не повинні бути зроблені при реалізації програмного рішення даного застосунку, та переваги, на котрі також потрібно буде звернути увагу.

Таблиця 1 – Порівняння існуючих сучасних програмних рішень

Характеристика	**Steam**	**Epic Games Store**	**GOG**
Рік запуску	2003	2018	2008
Компанія-розробник	Valve	Epic Games	GOG sp. z o.o.
Основне призначення	Цифрова дистрибуція відеоігор, програм чи обладнання та соціальна платформа	Цифровий дистрибуція відеоігор та ігрового двигуна Unreal Engine	Продаж DRM-free відеоігор

Кінець таблиці 1

Кількість ігор	Найбільша бібліотека серед платформ всіх ігрових платформ	Менша бібліотека, акцент на ексклюзивність товарів	Менша бібліотека, акцент на класичні ігри та ігри без захисту DRM
DRM-захист	Використовується Steam DRM або інший в залежності від потреб видавця чи розробника	Використовується DRM Epic Online Services або інший вид в залежності від потреб видавця чи розробника	Переважно DRM-free
Соціальні функції	Дуже розвинений текстовий та голосовий чат, групи, профілі, форуми продуктів чи вебзастосунку	Наявні форуми вебзастосунку	Наявні форуми вебзастосунку
Система відгуків	Повноцінна система користувацьких відгуків та рейтинг основуючись на відгуках, відображення нагород	Частково реалізована, доступна не всім покупцям	Наявна повноцінна система оцінювання та відгуків
Інтерфейс	Функціональний, багато різної інформації	Мінімалістичний	Простий та класичний, додаткова інформація з сторонніх вебсайтів
Комісія платформи	Приблизно 30%	Приблизно 12%	Приблизно 30%
Основні переваги	Велика бібліотека товарів, соціальні можливості, додаткові можливості до модифікації контенту	Безкоштовні роздачі товарів, сучасний інтерфейс, нижча комісія для розробників	DRM-free, підтримка класичних ігор на нових пристроях, офлайн-встановлення
Основні недоліки	Перевантажений інтерфейс та функціональність	Недостатньо функцій спільноти	Менший вибір сучасних ігор

1.2.1 Платформа Steam

Steam є однією з найпопулярніших систем цифрової дистрибуції відеоігор та програм компанії Valve чи інших розробників та видавців, та товарів пов’язаними з ними (завантажувальний контент чи саундтрек) або продукцію компанії Valve (Steam Deck, Steam Controller, Steam Frame, Steam Machine) у всьому світі. Дана платформа була розроблена компанією Valve у 2002-ому році та на початку використовувалася для видавництва своїх теперішніх та майбутніх проєктів та оновлення даних ігор, особливо своїх багатокористувацьких проєктів, але з часом, набуваючи нові можливості та функції, була трансформована у повноцінний онлайн-магазин.

Рисунок 1.2 – Перша версія платформи Steam, випущена в 2002-ому році

Платформа реалізована як багаторівнева система, що поєднує desktop app (Steam app client) та клієнт-сервер архітектуру з елементами трьохрівневної архітектури (клієнт – сервер – база даних). Основними мовами програмування для

реалізації desktop app не розголошується, але компанія Valve використовує мову програмування C++ для розробки ігрових рушіїв Source та Source 2 для ігрових (і не тільки) проєктів. Також можливо виділити те, що мова C++ найбільше підходить для роботи з пам’яттю, що дуже важливо для завантаження великих проєктів-ігор, роботи з файлами, особливо робота з драйверами системи, та оновлення клієнта. Тому, опираючись на дані факти, можливо дійти висновку що і для Steam була використана мова програмування C++. Для реалізації вебзастосунку були використані HTML, CSS, JavaScript. Також серед функцій магазину можливо виділити вбудований браузер Google Chrome, до якого має доступ користувач, що дає можливість також записати використання Chromium Embedded Framework, що надає можливість додавати браузери побудовані на вебоглядачі Chromium до сторонніх програм.

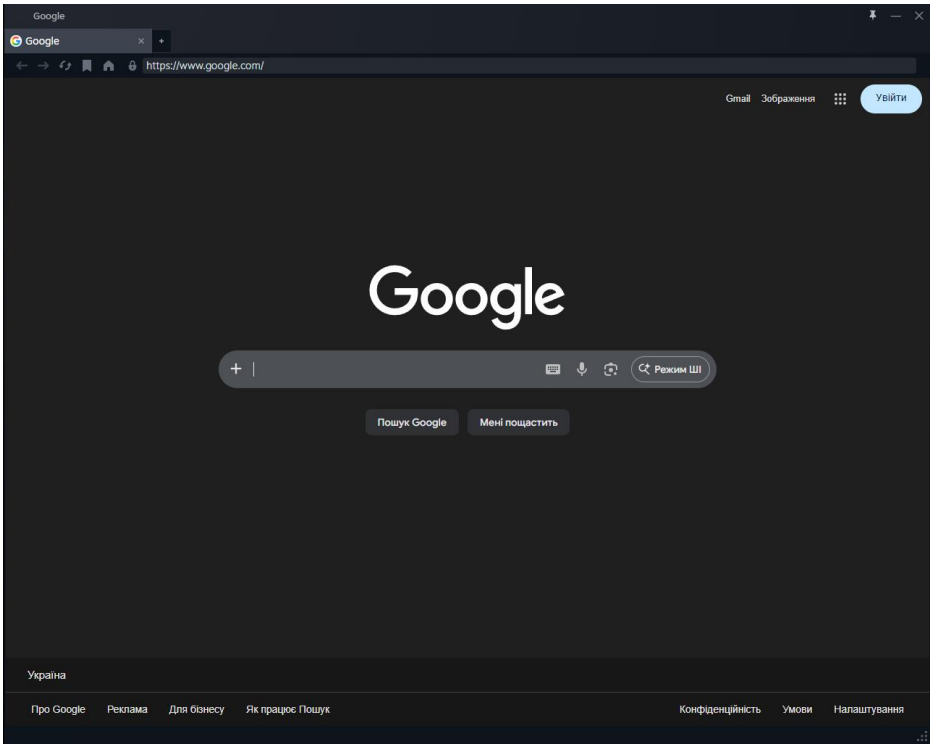

Рисунок 1.3 – Вбудований браузер Google Chrome в інтерфейс Steam

Перша сторінка (якщо користувач не змінить в налаштуваннях домашню сторінку на іншу), котра зустрічає користувача при запуску програми чи переходу на головне посилання (steampowered.com) – це сторінка магазину, яка складається з декілька елементів:

Банер з рекламою товару, котрий обирається розробниками Valve по-принципу що більше сподобається користувачам, чи актуального великого розпродажу, організоване Valve (як-от літні чи зимові розпродажі) чи розпродажи товарів інших видавництв.

Сторінка магазину також має свій header, котрий можливо побачити тільки на сторінках саме магазинної частини програми або сайту, що надає функціональні можливості для користувача, для орієнтації по всієї магазинної частини платформи. Частиною функціоналу даного header є поле для пошуку у крамниці та кнопки ‘Список бажаного’ та ‘Кошик’. Випадаюче меню ‘Огляд (Browse)’ надає можливість перейти до до різних сторінок в магазині, наприклад до новинок чи майбутніх релізів, або до функцій магазину, наприклад список бажаного користувача чи сімейна група (Steam Family).

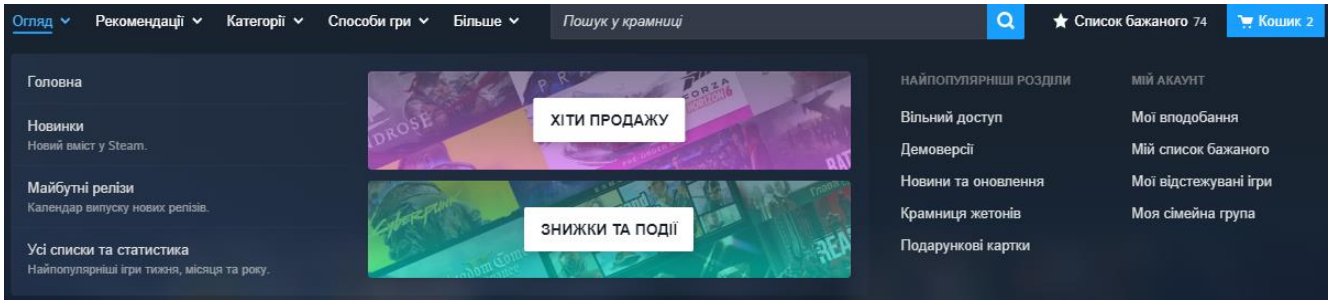

Рисунок 1.4 – Header з можливостями в магазині на платформі Steam

Наступне випадаюче меню, це ‘Рекомендації (Recommendations)’, котре надає можливість побачити персоналізовані, до трьох в кількості, товари, на основі бібліотеки користувача, через вкладку ‘Рекомендовано вам’, персоналізований на користувача календар майбутніх релізів, та посилання на персоналізовані сторінки. Третє випадаюче меню ‘Категорії (Categories)’ надає можливість виводити користувачеві персоналізований вибір категорії товарів, в кількості до шести категорій, або побачити усі. Четверте випадаюче меню ‘Способи гри (Hardware)’ надає спосіб відфільтрувати товари за підтримуваним способом гри в ігри, через вкладки ‘Чудово на (Steam) Deck’, ‘Remote Play’, ‘Продукти з ВР (VR)’, ‘Сумісні з контролерами (controllers)’, ‘Кооперативні’, та додаткові. Останнє випадаюче меню

‘Більше (More)’: надає можливість побачити всі інші розділи, наприклад сторінку ‘Демоверсії’, сторінку ‘Зав. Вміст (DLC)’ чи сторінку ‘Розпродажі’.

Нижче на головній сторінці магазину йде вкладка ‘Відібране і рекомендоване’, котре підбирається під користувача на основі тегах вже куплених і зіграних товарів у бібліотеці. Ще нижче відображаються різний вміст для сторінки – знижки, схожі за тегами товари, додатковий контент до існуючого товару, тощо. Також можливо на сторінці відфільтрувати за категоріями ‘Популярні новинки’, ‘Хіти продажу’, ‘Незабаром’, ‘Знижки’ та ‘Популярні з довільним доступом’ чи побачити схожі товари з тими, котрі наявні у користувача в бібліотеці чи списку бажаного. Це дозволяє реалізувати безкінечний скроллінг всієї сторінки (infinite scroll).

Тобто можливо побачити, що платформа реалізує можливість фільтрування та пошуку товари на смак користувача, чи відслідковування статусу за допомогою можливості списку бажаного. Більшість інших функцій є частиною іншої сторони платформи, котра позиціонується більше як соціальна мережа, тому до цих функцій не будуть приділятися більшої уваги, аніж до вже описаних.

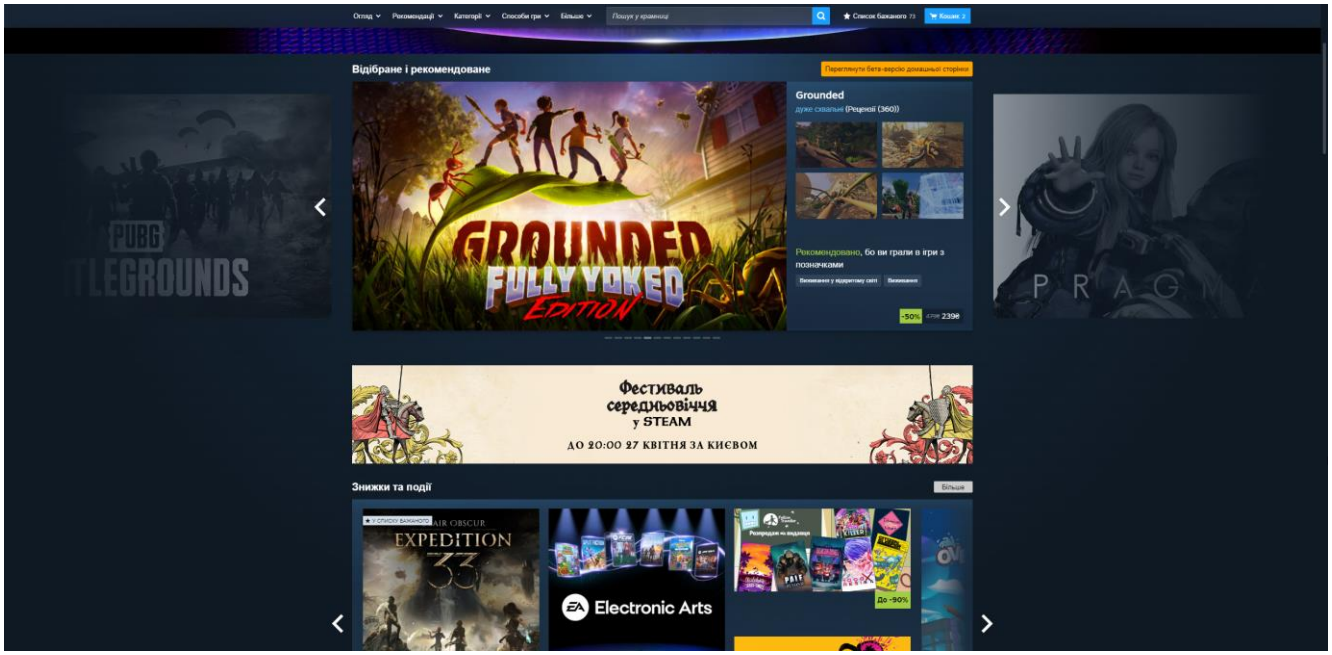

Рисунок 1.5 – Головна сторінка магазину платформи Steam

При переході до сторінки самого товару, клацнувши по блоку товару в магазині, буде відображатися вся доступна інформація пов’язана з даним

продуктом. Більшість функцій потребує умови для виводу, як от наприклад, для поля додавання рецензії, у користувача потрібно щоб продукт був куплений та знаходився в бібліотеці, чи для рекомендації, бажання товару чи наявність від друзів, у користувача повинні бути додані друзі через функції платформи Steam. У верхній частині сторінки розміщується назва гри, банер, а також коротка інформація у вигляді опису та окремі дані про розробника і видавця продукту. Якщо користувачу потрібна коротка інформація про стан продукту від користувачів, то також зображається рейтинг останніх рецензій та всі рецензії, відфільтровані за мовою та їх кількість. Це надає користувачу можливість дізнатися про поточний стан продукту порівняно з попереднім часом, що особливо важливо при можливості додавання масових негативних рецензій чи зміни політики розробника.

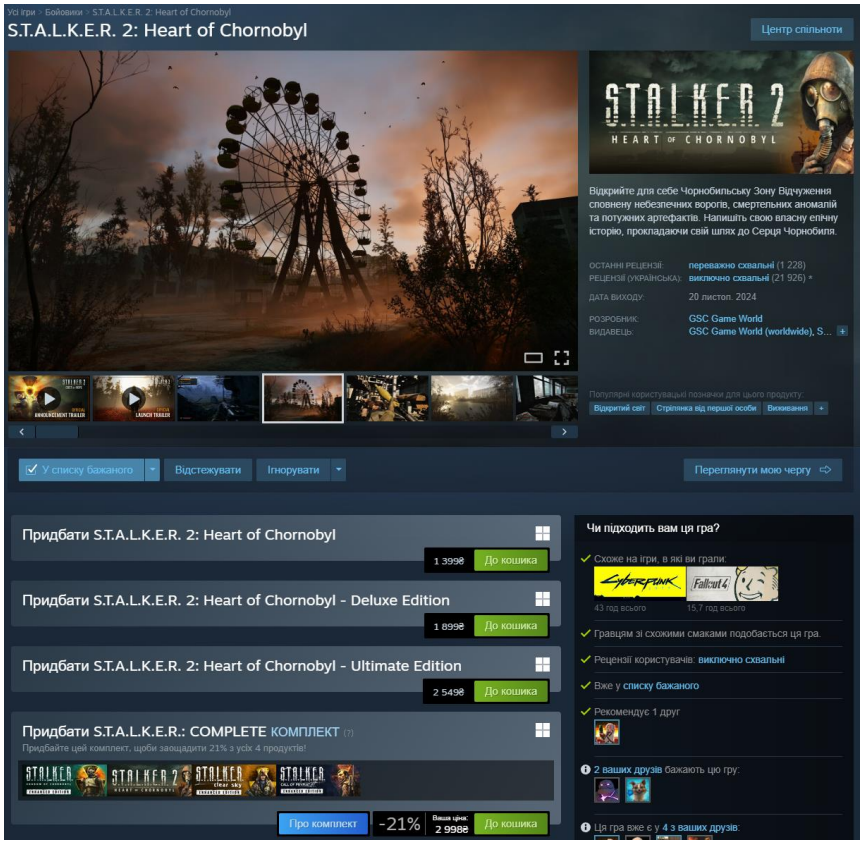

Рисунок 1.6 – Приклад сторінки товару платформи Steam

Найвелике місце займає великий блок із медіаконтентом сторінки, що повинна формувати перше враження про товар, там знаходяться медіа-скріншоти чи відео або кліпи з прямою трансляцією розробника. Далі також є кнопки ‘Список

бажаного’, що додає товар до списку бажаного аккаунту, ‘Відстежувати’ та ‘Ігнорувати’. Також під описом користувачі можуть додати користувацькі позначки до товарів, але котрі не є тегами товару, котрі видавці встановлюють самостійно, теги знаходяться нижче. Наявні кнопки ‘Додати до кошика’ з актуальним цінником та, якщо товар є частиною якогось комплекту, то даний комплект буде відображатися на сторінці товару з цінником, або додатковий контент також буде відображатися нижче для можливості його покупки.

На кінці сторінки товару відображаються рецензії користувачів та мінімальні чи рекомендовані системні вимоги до продукту, задані розробником. У відгуках користувачі можуть залишати коментарі та ставити ‘Рекомендую’ чи ‘Не рекомендую’, що формує відсотковий рейтинг товару та надає можливість відобразити це на початку сторінки.

Рисунок 1.7 – Приклад нижньої сторінки (footer) платформи Steam

На нижній частині всіх сторінок (footer) присутні всі потрібні юридичні та контактні інформація та посилання, наприклад про службу підтримки чи подарункові картки. Також там надається можливість пройти до сторінки Steamworks. Це API, за допомогою котрого можливо отримати та використовувати деяку інформацію про продукти, до котрої надає доступ компанія Valve, наприклад виводити онлайн гравців використовуючи даний продукт чи історія змін. Все це надає можливість користувачам отримувати більш гнучку та розширену інформацію про продукт.

1.2.2 Платформа Epic Games Store

Epic Games Store є сучасною та відносно молодою платформою для продажу, акцент поставлений не на кількість функціональних можливостей не пов’язаних з

магазином, а саме на простоті використання та вигідні умови розробникам та покупцям.

Epic Games Store також використовує багаторівневу систему, що поєднує desktop app (Epic Games Launcher) та клієнт-сервер архітектуру з елементами трьохрівневної архітектури. Для створення вебзастосунку використовуються мови HTML, CSS та JavaScript. Розробник Epic Games також не розголошує на якій мові програмування був розроблений додаток, але виходячи з попередніх фактів та того, що ігровий рушій Unreal Engine використовує мову C++, а сам лаунчер при збої роботи видає помилку саме даного рушія, можливо дійти висновку, що лаунчер розроблений з використанням рушія Unreal Engine, тому лаунчер також використовує мову програмування C++.

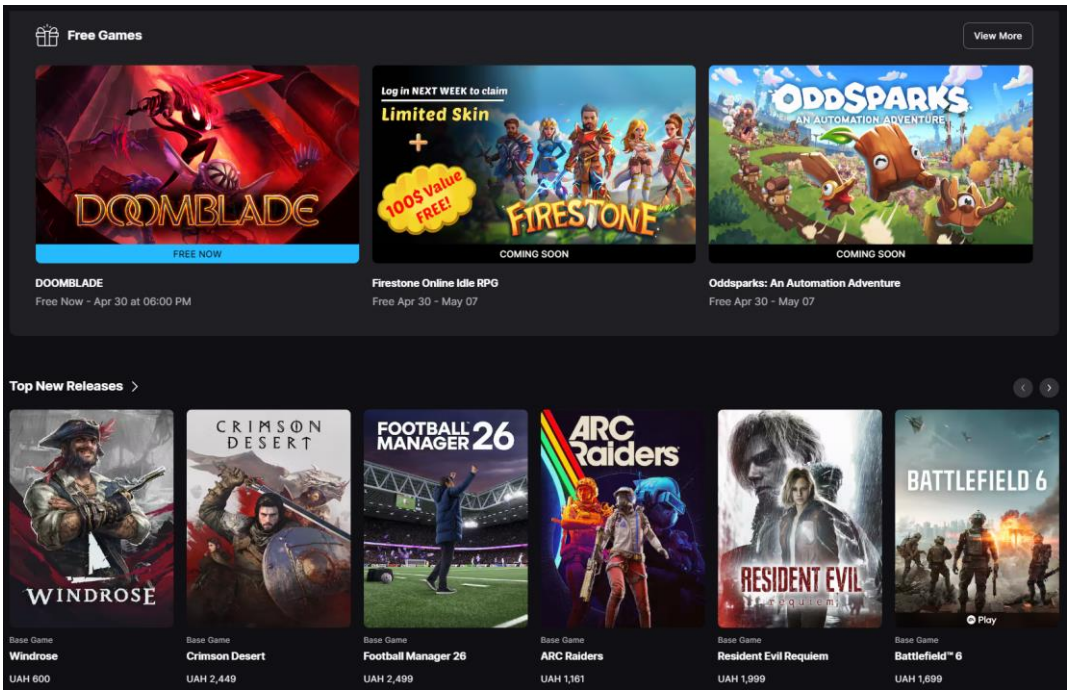

Рисунок 1.8 – Головна сторінка магазину платформи Epic Games Store

На головній сторінці зображуються різні товари з актуальними цінами та блок з безкоштовними товарами. Також, на сторінці, наявний header, в котрому є поле для пошуку товарів по магазину, кнопки на сторінки ‘Discover’, ‘Browse’, ‘News’, та зліва частини є кнопки для використання списку бажаного, подарунків, кошику та профіля користувача.

Як і попередній приклад, так і наступні приклади, платформа забезпечує всі потрібні функціональні можливості системи продажу товарів-відеоігор, але більш спрощено порівняно з іншими платформами. Наявне забезпечення автоматичного оновлення ігор та інтеграцію з обліковим записом Epic Games. Особливою характеризуючою особливістю є регулярні безкоштовні роздачі ігор, що стимулює залучення нових користувачів, та зменшену комісію для розробників, котрі використовують ігровий рушій Unreal Engine, розроблений компанією Epic Games, одного з популярніших ігрових рушіїв для використання з новими технологіями та постійною підтримкою. Це надає можливість залучати нових користувачів до платформи, але через брак додаткового функціоналу чи дуже довгу реалізацію нових функцій до магазину, платформі дуже складно утримувати користувачів через відсутність стримуючого фактора.

Рисунок 1.9 – Приклад сторінки товару платформи Epic Games Store

Сторінка товару має більше деталей та функціональності для повного представлення продукту. У верхній частині сторінки розміщується основна інформація про гру: назва, рейтинг та мітки.

Найбільшим, тому і важливим, елементом є медіагалерея, яка включає скріншоти та відеоматеріали. Це дозволяє користувачу оцінити візуальний стиль гри та її геймплей ще до покупки. Біля медіагалереї можливо виділити відображення тип продукту, його актуальна ціна, розробник та видавник, платформа та кнопки для додавання в кошик, оформлення як подарунку чи додавання в список бажаного.

Нижче йде текстовий опис товару, оформлений видавцем, та ключові характеристики, наприклад жанр чи можливості. Цей блок допомагає користувачу краще зрозуміти, чи відповідає продукт його очікуванням.

Не менш важливим є розділ системних вимог та рецензій. Системні вимоги містить мінімальні та рекомендовані параметри для запуску гри. Щодо рецензій, то Epic Games Store використовує змішаний підхід до даного питання. Використовуються як і рецензії користувачів, так і критиків. Рецензію користувач залишити сам не зможе, магазин надає можливість декільком користувачам залишити після того, як гравець вийшов з гри. Для отримання рецензій критиків магазин використовує можливості вебсайту OpenCritic.

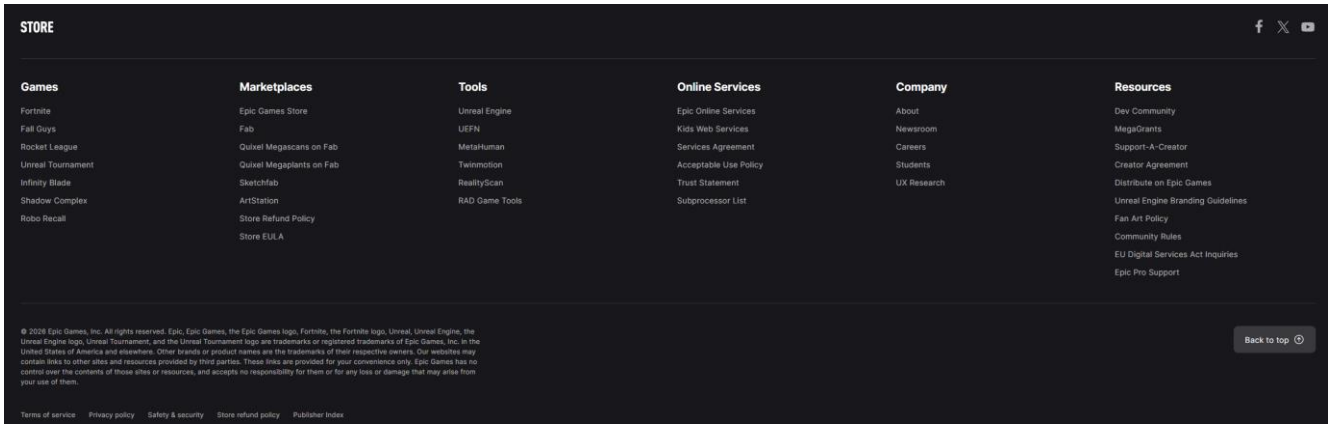

Рисунок 1.10 – Приклад нижньої сторінки (footer) платформи Epic Games Store

Нижня частина всіх сторінок (footer) присутня тільки у вебзастосунку, у лаунчер-версії відсутній, деякі посилання знаходяться через випадаюче меню до профілю користувача в header-частині лаунчера, але вони відкривають потрібну сторінку в браузері, за замовчуванням, користувача. На вебзастосунку зображені всі потрібні юридичні та контактні інформація та посилання, пов’язаних з Epic

Games, наприклад можливо знайти всі посилання на інструментарії розроблені компанією чи відеоігри. Також можливо знайти посилання про детальну інформацію пов’язану з компанією чи її ресурсами, наприклад підтримка автора чи умови та можливості видання товарів на платформі Epic Games Store.

1.2.3 Платформа GOG.com

Платформа GOG (Good Old Games) є платформою для продажу відеоігор, котра займає свою нішу на цифровому ринку відеоігор. Розробники роблять акцент на продаж товарів без DRM-захисту, що є дуже сильним фактором для приваблення нових користувачів та покупців до платформи.

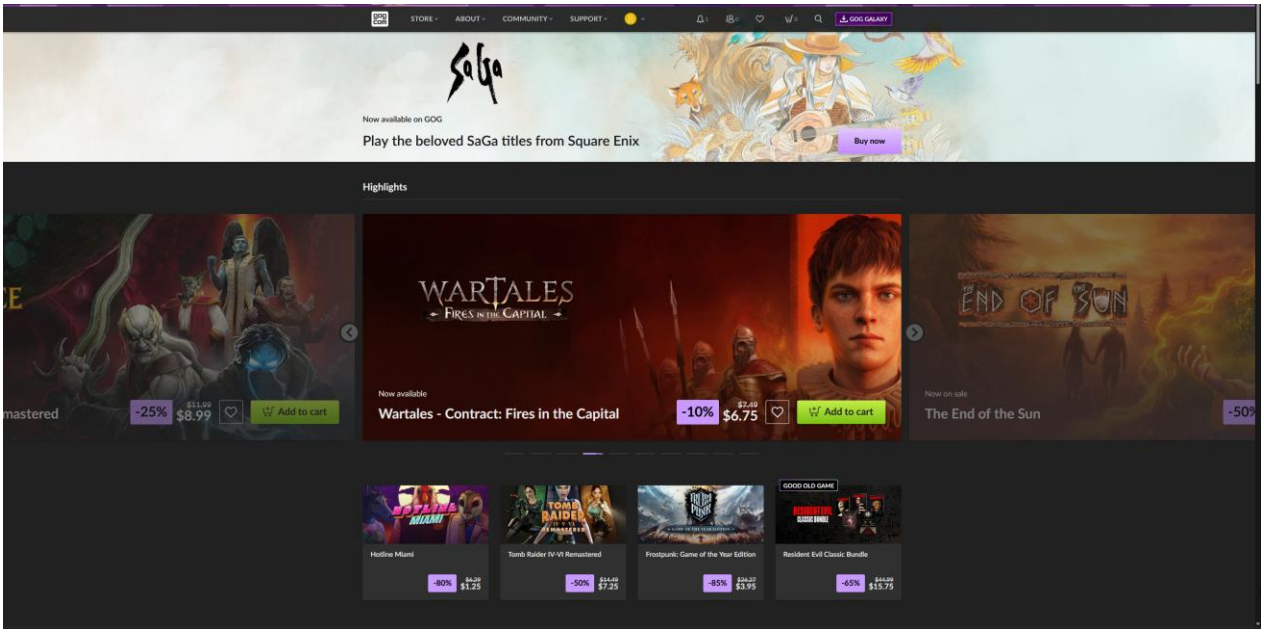

Рисунок 1.11 – Головна сторінка магазину платформи GOG.com

Як і попередні приклади, для GOG використовується багаторівнева система, що поєднує desktop app (GOG Galaxy) та клієнт-сервер архітектуру з елементами трьохрівневної архітектури. Для створення вебсайту використовуються мови HTML, CSS та JavaScript. Розробник GOG не розголошує на якій мові програмування був розроблений додаток, але роблячи ставку на попередні факти, можливо дійти висновку, що для даного додатка також використовується мова C++.

Всі сторінки вебзастосунку мають свою верхню частину (header), з можливістю навігації через подані кнопки для переходу між сторінками вебсайту, наприклад до сторінок магазину, інформації про вебсайт, форуми та служби

підтримки. Також наявні кнопки для повідомлень, списку друзів, списку бажаного, кошику, пошуку товарів по всьому магазину та кнопка для завантажування інсталятора для лаунчера ‘GOG Galaxy’.

Вся основна сторінка магазину заповнена тематичними секціями по-типу ‘Highlights’, ‘Games for you’, ‘Now on sale’ , ‘GOG Preservation Program’ та інші, що дають можливість користувачі швидко зорієнтуватися на можливості та продукцію вебсайту.

У нижній частині всіх сторінок (footer) знаходяться посилання на потрібну юридичну чи контактну інформацію, а саме служба підтримки, політика конфіденційності, соціальні мережі (Facebook, Twitch, Youtube, X/Twitter), можливість обрати мову вебзастосунку, а також додаткові розділи платформи. Загалом дизайн головної сторінки GOG є мінімалістичним, з акцентом на візуальний контент і зручну навігацію. Цей елемент сторінок присутній тільки у вебзастосунку, у додатку GOG Galaxy він відсутній, але частина посилань знаходиться у кнопки налаштувань в додатку.

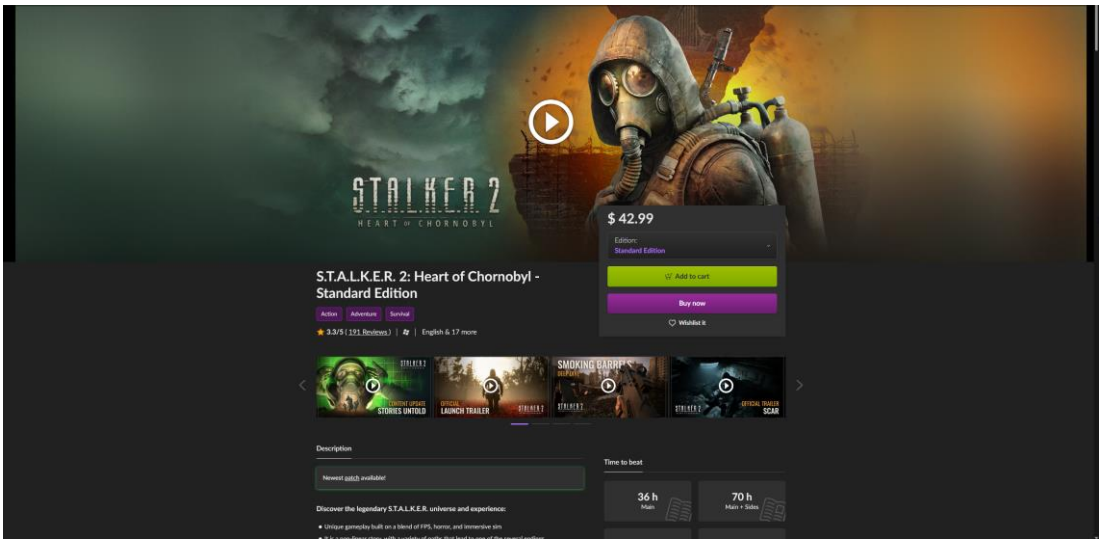

Рисунок 1.12 – Приклад сторінки товару платформи GOG.com

Сторінка товару має більш детальний опис та функціонал для зображення потрібної користувачеві інформацію. У верхній частині сторінки зазвичай розміщується назва гри, обкладинка або відеотрейлер, котрі працюють як фонові зображення, та коротка інформація про рейтинг, доступну платформу та мови.

Також біля даної частини знаходяться актуальний цінник та кнопки для додавання в кошик, купити продукт окремо зараз, все це обираючи тип продукту, та кнопка для додавання в список бажаного.

В середині сторінки знаходяться детальний опис продукту, деталі гри та доступні можливості з мовами, а саме жанри, теги, доступні платформи, дата виходу, компанії розробників та видавців та корисні посилання. Також, за допомогою використання можливостей вебсайту HowLongToBeat, наявна інформація скільки годин буде потрібно гравцеві щоб пройти основну частину гри.

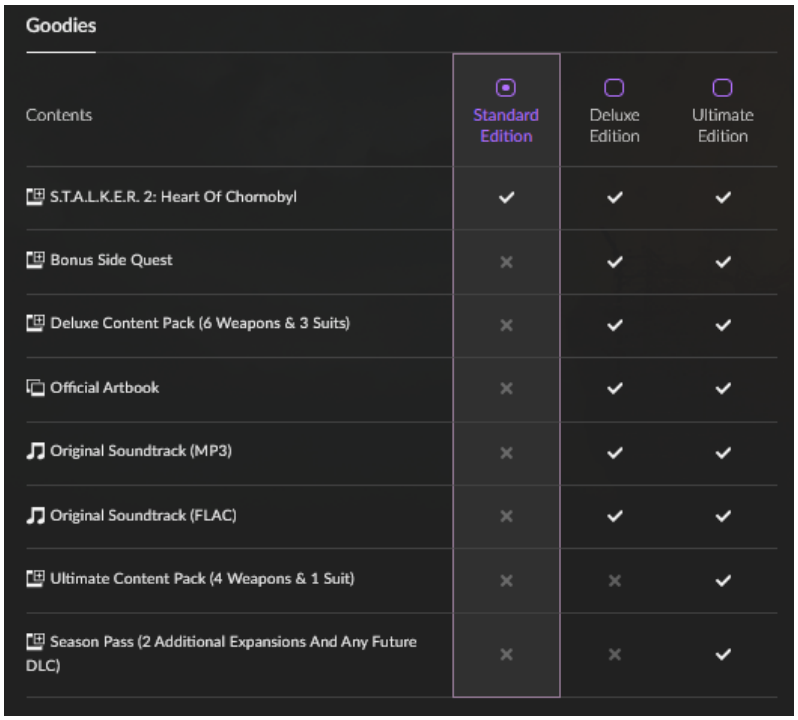

Рисунок 1.13 – Приклад виводу порівняння вмісту різних видань товарів на платформі GOG.com

На нижній частині сторінки присутня таблиця порівняння вмісту різних видань товару, якщо присутні, що допоможе користувачеві знайти потрібне видання для покупки та побачити зміст. Також наявний розділ із відгуками користувачів, де можливо залишати коментарі, та рейтингом критиків, що допомагає сформувати уявлення про якість гри покупцеві. Додатково може бути блок ‘You may also like’, котрий пропонує схожі, за жанром та тегами, товарами.

Висновки до розділу 1

В даному розділі визначені та обгрунтовані об’єкт і предмет дослідження роботи, що пов’язані з темою роботи. Визначено, що об’єктом роботи є процес електронного продажу та покупки ліцензій і фізичних копій відеоігор, через поширення та розширенні можливості цифрових платформ, що дозволяють користувачам здійснювати покупки, як і фізичних, так і цифрових товарів, миттєво в комфортному, для них, місці та коли побажає сам покупець.

Предметом дослідження роботи стали інструменти та технології розробки вебзастосунку для продажу електронних товарів, до котрих належать: мови програмування, фреймворки та бібліотеки, системи управління базами даних, підходи до проєктування архітектури ПЗ, засоби інтеграції з зовнішніми сервісами та інструменти для забезпечення безпеки та продуктивності. Всі ці елементи надають можливості для створення програмного рішення на поставлені задачі.

Були виділені та проаналізовані структурні та функціональні можливості об’єкта роботи, за допомогою котрих можливо поділити об’єкта на логічні частини, котрі будуть співпрацювати між собою та надають можливість постановку вимог до реалізації системи електронної комерції продажу відеоігор.

Було оглянуто та проаналізовано стан існуючих сучасних програмних рішень-аналогів. Серед них можливо було виділити три платформи: Steam, Epic Games Store, GOG.com. Можливо виділити потрібні, для роботи електронного магазину, функції, наприклад кошик, список бажаного, можливість побачити рейтинги товару. Можливо виділити переваги і недоліки, наприклад великим плюсом платформи Steam, є підтримка різних способів гри (ігрові контролери, VR-окуляри, спеціалізовано консолі, спеціалізований окремий інтерфейс для телевізора), але це також може бути і мінусом, через велику кількість функціоналу користувач зможе заплутатись в ньому. Чи наприклад GOG.com, відомий підтримкою відеоігор без DRM-захисту, чи Epic Games Store, відомий своїми безкоштовними роздачами, хоча це не дає таку велику перевагу саме покупцеві.

2 МОДЕЛЮВАННЯ ОБ'ЄКТУ ТА ПРЕДМЕТУ РОБОТИ

2.1 Аналіз сучасного стану інструментарію, моделей та методів

Інформаційні технології є одним із найшвидших сфер, у котрій відбуваються постійні стрімкі зростання, що надають можливості до реалізації нових програмних рішень у різних сферах, що надає можливість розробити потрібний функціонал та ідеї у сфері електронної комерції. У зв'язку з даним стрімким зростанням, частіше з'являються і нові середовища з інструментарієм для реалізації даних ідей та завдань та будуть забезпечувати високу продуктивність, масштабованість, безпеку користувачів та зручність використання поданої системи [1].

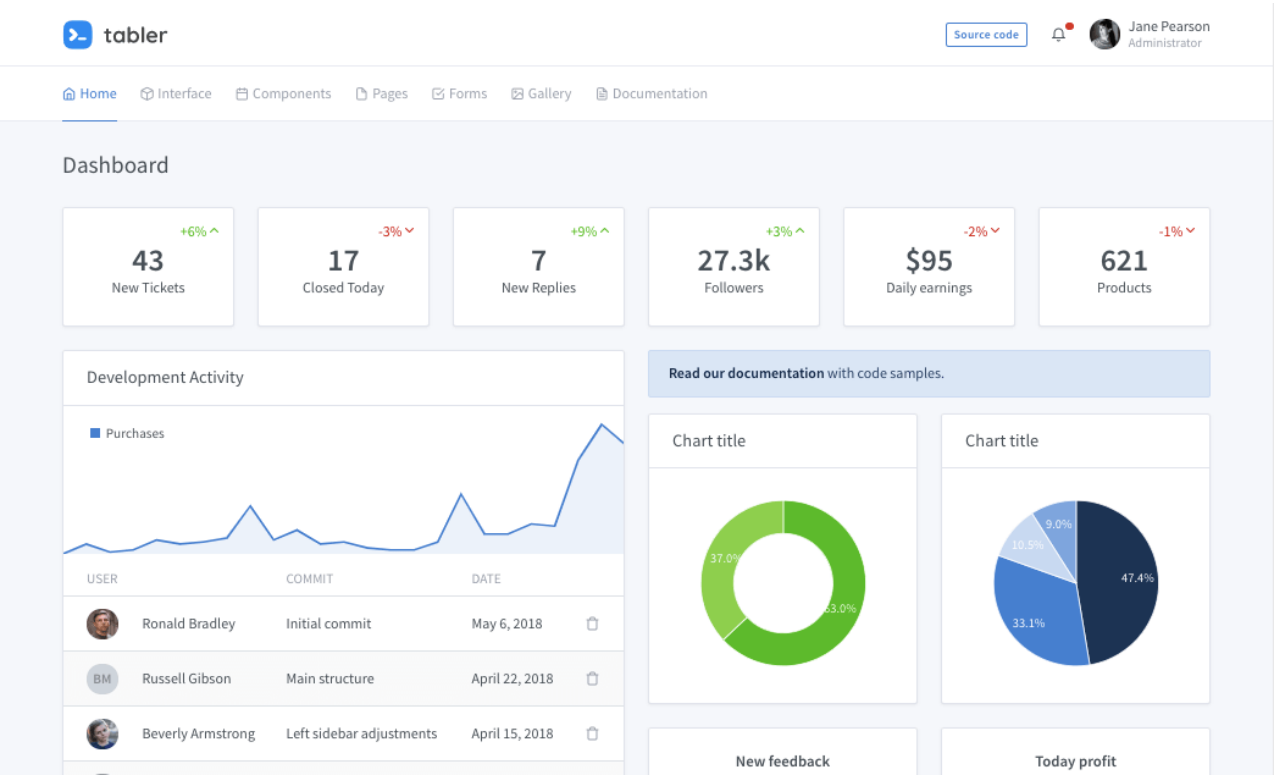

Рисунок 2.1 – Приклад результату сторінки з використання можливостей React

Однією з поширених сфер, з постійним оновлюючим інвентарем є активне використання клієнтських веб-технологій для створення інтерфейсів користувача у вебзастосунках. Наприклад поширення односторінкових застосунків SPA (Single Page Application), а саме React, Angular та Vue.js [2], надають можливості для розробки швидкого інтерфейсу через забезпечення плавної взаємодії користувача із системою, сприяє можливості для зменшення системного навантаження на

серверну частину проєкту [3], та надають можливості для зручного компонування елементів на стороні розробника для подальшої підтримки.

React, бібліотека створена компанією Meta (Facebook), надає можливість створювати швидкі та інтерактивні UI за допомогою використання архітектури, котра базується на використання компонентів. Також великим плюсом є те, що фреймворк має сильно розвинену базу користувачів-розробників, котрі розробляють готові бібліотеки для використання у рішенні різних проблем та питань, котрі виникають під час розробки. Але через це, на початку React-проєкт займає більше потрібного часу на налаштування та імплементацію базового функціоналу. Даний фреймворк підійде під вебзастосунку котрі працюють ‘у реальну годину’ та постійно оновлюють інформацію на сторінці.

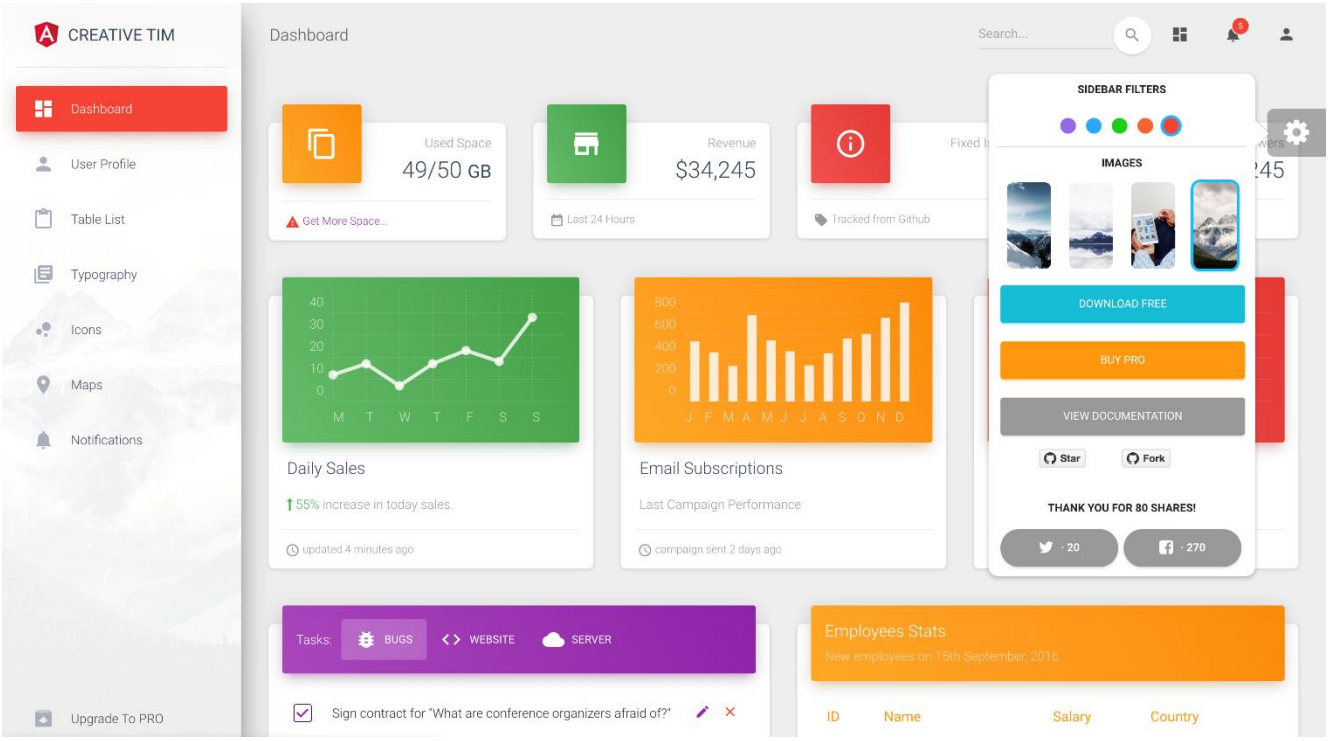

Рисунок 2.2 – Приклад результату сторінки з використання можливостей Angular

Angular, розроблений компанією Google, як повноцінний фреймворк, надає можливості для повної розробки клієнтської частини. На відміну від попередньої бібліотеки, даний фреймворк включає в себе потрібний, для повноцінної розробки UI та взаємодії між клієнт-сервером, інструментарій з самого початку створення проєкту, без налаштування сторонніх бібліотек для маршрутизації чи засоби для

обробки користувацьких форм, що задовольняє потрібності розробника. З іншого боку, це можливо бути і мінусом, якщо дані вбудовані можливості не використовуються, та мають тільки одну мету – займати місце в системі. Даний фреймворк підходить також під дуже великі проєкти з дуже великою командою та дуже багатим функціоналом, котрий потребує вбудовані інструменти для роутингу (routing), роботи з HTTP-клієнтом, валідацією та контролем над дуже складною бізнес логікою [4].

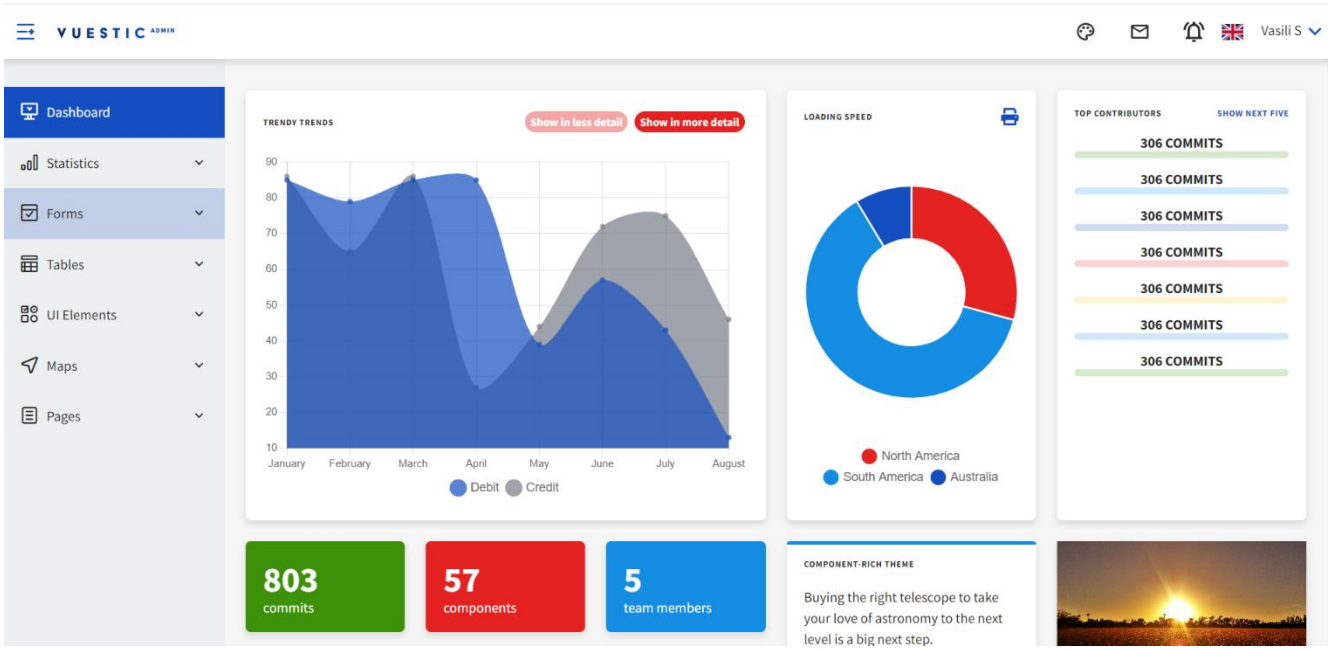

Рисунок 2.3 – Приклад результату сторінки з використання можливостей Vue.js

Vue.js – це фреймворк, котрий базується на принципі відкритого коду, є простим та зрозумілим для початківців у створенні SPA-проєктів. Краще всього проявляється у створенні маленьких чи середніх проєктів, котрі потребують постійних оновлень. Через дуже своєрідну екосистему, що потребує адаптацію під різні браузери та операційні системи, даний фреймворк складно підтримувати на старих системах. Через більш прості можливості, сам проєкт, у порівнянні з розмірами однакового проєкту за функціонал в інших фреймворках, може опинитися дуже легким, що надає перевагу у пошуку Google чи інших, котрі не намагаються показувати більш важкуваті вебсайти на перших сторінках зверху, через дуже довгі завантаження.

Більш важливим аспектом розробки вебзастосунку є вибір технології для серверної частини, котра буде з’єднувати між собою дії з клієнтською частини та даними з бази даних через можливості та функції сервера. Також для можливості краще використовувати Rest API для надання способу викликів стандартних HTTP-методів по-типу GET, REMOVE, POST та інші. Для рішення даного питання можливо виділити платформи Node.js та фреймворк .NET.

Node.js – це платформа, котра базується на рушії V8 JavaScript, функціювання котрого можливий за допомогою мови C++, та дозволяє виконувати саме код мови JavaScript на сервері. Це дозволяє реалізувати швидку мережеву програму для під’єднання між клієнтом та базою даних. Всі процеси в платформі базуються на подіях, через можливості асинхронної моделі, використовуючи один поток центрального процесору, не впливаючи на продуктивність застосунку. Також використовуючи підтримку спільноти, можливо додавати додатковий функціонал через додаткові пакети та бібліотеки, використовуючи можливості npm.

Додатковою особливістю платформи Node.js – це можливість перейти з однієї операційної платформи до іншої, тобто можливість кроссплатформеності (crossplatform), що надає можливість перейти серверу з операційної системи Windows до легкої дистрибуції Linux, що надає додаткові ресурси для роботи сервера. Головним недоліком може бути і факт асинхронності, що може погіршувати можливість розуміння структури і коду дуже складного проєкту, що призводить то складності у підтримки. Rest API можливо реалізувати використовуючи Express.js, що надає можливість додати middleware захист для запитів та реалізувати повноцінно HTTP-методи [5].

Альтернативою можливо визначити платформу .NET. Через свою кроссплатформеність, котра надає можливість запускати різни типи програм на різних операційних системах. Через особливості платформи, можливо реалізувати окремо кожну логіку, що надає ширшу можливість для подальшої підтримки великих проєктів. Платформа надає, через вбудовані функції, інструментарій для забезпечення безпеки в даних проєкта, що особливо потрібно для функцій авторизації та контролю за ролями. Головною проблемою платформи .NET,

особливо порівняно з Node.js, це більше потрібності у ресурсах системи, особливо для більш складною серверної системи [6]. Також .NET використовує мову C#, що може ускладнювати старт проєкту чи написання функцій, через більш великий потрібний об’єм коду для реалізації функціоналу. В ASP.NET Core Rest API встановлений за замовчуванням, тому, на відміну від попереднього прикладу, .NET не потребує окремий бібліотек для створення повноцінного сервера.

Однією з важливих складових всіх проєктів вебзастосунків є зберігання даних, котрі посилаються чи потребуються користувачем, для цього завдання використовуються бази даних, серед котрих можливо виділити Microsoft SQL Server, PostgreSQL, MySQL та MongoDB.

Microsoft SQL Server більш комерційний варіант, але має безкоштовну версію, яка підтримує один процесор, 1 ГБ максимальної пам’яті для використання та максимум 10 ГБ місця системи для використання для однієї бази даних. Противагу цьому складає більш обширний інвентар інструментів та функціоналу. Дана програма надає інструменти для сервісу звітності, аналітики та повну підтримку для JSON-файлів.

PostgreSQL є безкоштовною альтернативою реляційних баз даних, котра підтримує багато різних типів даних. Також основною перевагою даної програми є можливість розширення функціональності, використовуючи власні модулі. Але через додаткові можливості також виникає потрібність у більш глибоких знань для використання у оптимізації адміністрування даних та їх правильної продуктивності [7].

MySQL є однією з найпопулярніших програм з відкритим кодом для роботи з реляційними базами даних. Перевагою даною програми є її простота, що потребує менше ресурсів системи та надає можливості легкої інтеграції з різними мовами програмування. Підходить більше під невеликі чи середні за розмірами проєктами [7].

2.2 Специфікація вимог до ПЗ

1) Призначення та межі проєкту:

1.1) призначення системи (застосунку), для якої розробляється програмне забезпечення;

1.2) було узгоджено, що для створення вебзастосунку буде використовуватись: для клієнтської частини – React, для бази даних – Microsoft Server Management та для серверної частини – NodeServer, мови програмування, котрі будуть використовуватись – JavaScript (бібліотека ReactJS) та TypeScript;

1.3) межі проєкту ПЗ – 31.05.2026 р.

2) Загальний опис:

2.1) сфери застосування:

– авторизація користувачів – відповідає за реєстрацію, авторизацію та аутентифікацію користувачів вебзастосунку та отримання доступу до ролей. Всі ролі мають можливість створювати облікові записи, входити до системи, відновлювати пароль та керувати власним профілем;

– адміністрування вебзастосунку – відповідає за функціонування адміністративної панелі вебзастосунку. Роль адміністратора має повний доступ до керування каталогом ігор, категоріями, користувачами, замовленнями, відгуками, а також здійснювати контроль за роботою системи;

– каталог товарів – відповідає за відображення переліку товарів із можливістю сортування, фільтрації та пошуку за різними параметрами. Користувачі можуть переглядати детальну інформацію про гру, включаючи опис, характеристики, скріншоти та відгуки. Наповнення та редагування каталогу здійснюється продавцями через відповідні панелі та затверджується адміністраторами;

– кошик та оформлення замовлення – надає можливість користувачам додавати обрані ігри до кошика, редагувати кількість товарів та оформлювати замовлення. Модуль забезпечує розрахунок загальної вартості

покупки, вибір способу оплати та підтвердження замовлення. Інформація про замовлення зберігається в системі та доступна для перегляду адміністратором;

– відгуки та рейтинг – надає користувачам можливість переглядати, оцінювати придбані товари та залишати коментарі. Роль адміністраторів отримують можливість модерувати відгуки, редагувати або видаляти некоректний контент;

– персональний кабінет користувача – дозволяє користувачам переглядати особисті дані, історію замовлень, придбані товари та налаштовувати дані профіля для перегляду.

2.2) характеристики користувачів:

– гість (незареєстрований користувач): мають можливість переглядати каталог товарів, здійснювати пошук та фільтрацію товарів, переглядати детальну інформацію про товар, та можливість створювати новий обліковий запис. Не мають можливості оформлювати замовлення або залишати відгуки без реєстрації та купленого товару;

– користувач (зареєстрований): мають можливість переглядати каталог товарів, додавати товари до кошика, здійснювати пошук та фільтрацію товарів, переглядати детальну інформацію про товар, оформлювати замовлення, переглядати історію покупок та стан замовлення, редагувати особисті дані на сторінці профілю.

– споживач: мають всі можливості користувача (зареєстрованого) та додатково можуть залишати відгуки та оцінки до придбаних та отриманих товарів;

– продавець: отримують доступ до спеціалізованої панелі керування створення та редагування характеристик товарів, котрі були створені даним продавцем. Також отримують до, створених та затверджених, товарів доступ;

– адміністратор: має повний доступ до функціоналу вебзастосунку. Виконує модерацію каталогу товарів, користувачами, замовленнями, відгуками, ролями доступу та налаштуваннями системи.

2.3) загальна структура і склад системи:

– структура авторизації користувачів: форма для реєстрації нових облікових записів, авторизації, відновлення паролю, видалення облікового запису;

– структура адміністрування: панель керування користувачами системи, панель керування ролями доступу користувачів, вкладка перегляду та обробки всіх поточних замовлень, інструменти статистики продажів, інструменти резервного копіювання та відновлення даних;

– структура каталогу товарів: відображення списку товарів, перегляд детальної інформації про гру, система пошуку за назвою чи словом, фільтрація за жанром, платформою, ціною, рейтингом та іншими параметрами, сортування товарів за популярністю, новизною, ціною, перегляд можливих знижок;

– структура керування товарами продавця: сторінка для додавання нових товарів до каталогу товарів, редагування інформації про наявний товар, запрос на видалення товару з каталогу, можливість обмежити доступ до товару, додавання зображень та медіафайлів до товару;

– структура кошика та оформлення замовлення: можливість додавання товарів до кошика зареєстрованим обліковим записам, редагування вмісту кошика, функція для підрахунка загальної вартості замовлення, сторінка для заповнення даних для замовлення, вибір способу оплати замовлення, підтвердження та оформлення замовлення;

– структура профілю користувача: перегляд стану поточних оформлених замовлень, перегляд історії замовлень виконаних замовлень, перегляд списку отриманих товарів, сторінка для налаштування оформлення профілю користувача;

– структура відгуків та рейтингу товарів: додавання відгуків користувачами з перевіркою на наявність купленного товару, модерація відгуків адміністратором, відображення середнього рейтингу гри на основі відгуків користувачів, отримання рейтингу з сторонніх вебсайтів використовуючи можливості даних API.

2.4) загальні обмеження:

– технічні обмеження: система повинна працювати у сучасних веббраузерах (google chrome, microsoft edge, mozilla firefox тощо), забезпечення адаптивності інтерфейсу для персональних комп’ютерів, планшетів та мобільних пристроїв, обмеження серверних ресурсів (обсяг пам’яті, продуктивність хоста серверів);

– програмні обмеження: використання визначених технологій та фреймворків відповідно до обраного інструментарію, сумісність із системами керування базами даних**,** обмеження можливостей інтеграції сторонніх сервісів (платіжних систем, API);

– безпекові обмеження: забезпечення захисту персональних даних користувачів, використання механізмів аутентифікації та авторизації та шифрування паролів, захист від несанкціонованого доступу та типових загроз;

– організаційні обмеження: обмежені строки виконання проєкту, обмежені фінансові та технічні ресурси для реалізації додаткового функціоналу;

– функціональні обмеження: система не передбачає фізичну доставку товарів у разі орієнтації на цифрові копії, оплата здійснюється лише через інтегровані платіжні сервіси.

3) Функції системи:

3.1) функція перегляду каталогу товарів:

3.1.1) опис функції: дозволяє користувачу переглядати каталог відеоігор, здійснювати пошук та фільтрацію за різними параметрами, а також переглядати детальну інформацію про кожний товар через посилання на іншу сторінку.

3.1.2) вхідна інформація: параметри пошуку, обрані фільтри, параметри сортування, обрана сторінка. Вихідна інформація: перелік товарів відповідно до заданих параметрів пошуку, інформація про товар (назва товару, тип товару, теги, платформа (операційна система), ціновий діапазон (базова ціна та можлива знижка), рейтинг користувачів), номер сторінки, кількість знайдених

товарів за пошуковими запитами чи обраними параметрами фільтрації товарів, контекстні повідомлення.

3.1.3) функціональні вимоги: користувачі повинні мати змогу переглядати каталог товарів, здійснювати пошук за, введеними користувачем, словами, фільтрувати товари за, наданими вебзастосунком, параметрами, переглядати сторінку з детальним описом товару, перейти на інші сторінки вебзастосунку.

3.2) функція реєстрації та авторизації користувачів:

3.2.1) опис функції: забезпечення можливості реєстрації нових облікових записів, авторизації до системи та отримання прав доступу відповідно до ролі облікового запису (користувач, продавець, адміністратор) та відновлення доступу до облікового запису;

3.2.2) вхідна інформація: електронна пошта, пароль. Вихідна інформація: контекстні повідомлення;

3.2.3) функціональні вимоги: користувач повинен мати можливість зареєструватися в системі, можливість авторизуватися, система повинна реєструвати права доступу за ролями, адміністратор повинен мати можливість керувати обліковими записами;

3.3) функція оформлення замовлення:

3.3.1) опис функції: дозволяє користувачу додавати товари до кошика, редагувати вміст та оформлювати замовлення з подальшим вибором способу оплати;

3.3.2) вхідна інформація: обрані користувачем товари; платіжні дані користувача. Вихідна інформація: додані товари, сума замовлення, статус замовлення, заповнена інформація про оплату, контекстні повідомлення;

3.3.3) функціональні вимоги: користувач повинен мати змогу додавати товари до кошика, змінювати кількість товарів, заповнювати потрібні поля, система повинна автоматично розраховувати загальну вартість, зберігати інформацію про замовлення, адміністратор повинен мати змогу змінювати статус замовлення.

3.4) функція керування товарами продавця:

3.4.1) опис функції: надає продавцю можливість додавати нові товари до списку товарів, редагувати інформацію про них та видаляти створені, даним продавцем, товари, розглядати сторінку про товар, отримувати його статистику;

3.4.2) вхідна інформація: назва товару, опис, ціна, жанр, платформа (операційна система), медіаконтент. Вихідна інформація: оновлений список товарів, контекстні повідомлення.

3.4.3) функціональні вимоги: продавець повинен мати можливість додавати нові товари, редагувати вже створені товари, видаляти створені товари, адміністратор повинен мати повний доступ до управління товарами.

3.5) функція рецензій та рейтингу товару:

3.5.1) опис функції: надає користувачу можливість додавати та редагувати рецензії, отримувати, розглядати, оцінювати рецензії інших користувачів. Система має можливість формувати рейтинг товару та гістограми на основі рецензій користувачів, отримувати, зберігати, показувати та оновлювати дані рейтингу з сторонніх сайтів, використовуючи можливості API;

3.5.2) вхідна інформація: коментар користувача, обраний рейтинг товару від користувача (подобається/ не подобається), обраний рейтинг іншого відгуку від користувача (лайк/ дизлайк).

3.5.3) функціональні вимоги: відгук користувача з іменем профіля, коментарем та рейтингом товару, відгуки інших користувачів з іменем профіля, коментарем, рейтингом товару та кількості лайків чи дизлайків з відношенням, гістограма відгуків користувачів, рейтинг товару на основі всіх відгуків користувачів, рейтинг товару на основі отриманих даних з сторонніх API.

4) Вимоги до інформаційного забезпечення:

4.1) джерелами вхідної інформації для вебзастосунку магазину відеоігор є дані, що вводяться користувачами, продавцями та адміністраторами системи, а також дані, що формуються в процесі роботи системи. До вхідної

інформації належать: реєстраційні дані користувачів (електронна пошта, пароль, персональні дані), інформація про товари (назва, опис, жанр, платформа, ціна, медіаконтент), дані про замовлення та оплату, відгуки та рейтинги користувачів, службова інформація (ролі користувачів);

4.2) до нормативно-довідкової інформації належать: класифікатор жанрів відеоігор, довідник платформ (PC, PlayStation, Xbox тощо), довідник ролей користувачів (користувач, продавець, адміністратор), довідник статусів замовлень, довідник способів оплати замовлення;

4.3) вимоги до способів організації, збереження та введення інформації: інформація повинна зберігатися у централізованій базі даних, повинна бути забезпечена цілісність та актуальність даних, персональні дані користувачів повинні зберігатися у захищеному вигляді, повинно бути реалізовано регулярне резервне копіювання даних, доступ до інформації повинен здійснюватися відповідно до ролей користувачів.

5) Вимоги до технічного забезпечення:

Для роботи вебзастосунку необхідно, щоб технічний засіб користувача відповідав мінімальним системним вимогам для використання сучасного браузера, та мав підтримку HTML5, та мав доступ до мережі Інтернет. Вебзастосунок повинен коректно працювати на персональних комп’ютерах, ноутбуках, планшетах та мобільних пристроях.

6) Вимоги до програмного забезпечення:

6.1) архітектура програмної системи складається з трьохрівневної архітектури (клієнт – сервер – база даних) та має такі основні компоненти: серверної частини, яка відповідає за бізнес-логіку та обробку даних, бази даних для зберігання інформації, клієнтської частини (вебінтерфейсу), з котрої відбувається взаємодії користувачів з іншими елементами та отримання чи вивід даних з бази даних через серверну структуру;

6.2) системне програмне забезпечення вебзастосунку реалізовується з використанням можливостей Node.js. У якості системи керування базами даних використовується Microsoft SQL Server Management;

6.3) мережне програмне забезпечення: Система функціонує у мережі Інтернет із використанням протоколів HTTP/HTTPS. Доступ до вебзастосунку здійснюється через інтернет-браузер без необхідності встановлення додаткового програмного забезпечення;

6.4) програмне забезпечення ведення інформаційної бази: Усі операції зі збереження, редагування та видалення даних виконуються через базу даних, доступ до котрої виконується через Microsoft SQL Server Management з використанням серверної логіки вебзастосунку для отримання чи відправки даних. Користувачі взаємодіють з інформаційною базою виключно через інтерфейс системи клієнтської частини;

6.5) Мова і технологія розробки ПЗ: для розробки вебзастосунку використовуються сучасні вебтехнології, зокрема: фреймворки клієнтської частини React (використання ReactJS), технології та мови програмування HTML5, CSS, JavaScript (TypeScript), керування базою даних через мову SQL.

7) Вимоги до зовнішніх інтерфейсів:

7.1) інтерфейс користувача має бути інтуїтивно зрозумілим, зручним у використанні та адаптивним. Кольорова гама та структура сторінок не повинні створювати дискомфорт для очей користувачів та викликати заплутаність;

7.2) апаратний інтерфейс: в якості апаратного інтерфейсу можуть використовуватись будь-які пристрої, що мають доступ до мережі Інтернет та підтримують сучасні інтернет-браузери з сучасними технологіями;

7.3) програмний інтерфейс забезпечує взаємодію між клієнтською та серверною частинами системи, а також інтеграцію з базою даних і сторонніми сервісами (платіжні системи, сторонні API);

7.4) комунікаційний протокол: для передачі даних використовується стандартний протокол HTTP/HTTPS із застосуванням захищеного з’єднання.

8) Властивості програмного забезпечення:

8.1) доступність: програмне забезпечення повинно бути доступним для користувачів цілодобово (24/7), за винятком часу технічного обслуговування або аварійних ситуацій;

8.2) супроводжуваність: система повинна бути зручною для супроводу та подальшого розвитку. Код та структура системи повинні бути зрозумілими для інших розробників;

8.3) переносимість: програмне забезпечення повинно мати можливість розгортання на іншому сервері без суттєвих змін у коді та структурі даних.

8.4) продуктивність: час відгуку системи не повинен перевищувати 2 секунд для стандартних дій користувача. Система повинна коректно працювати при одночасному доступі кількох користувачів;

8.5) надійність: система повинна працювати стабільно протягом тривалого часу. Надійність забезпечується резервним копіюванням даних та обробкою помилок.

8.6) безпека: програмне забезпечення повинно забезпечувати захист та шифрування персональних даних користувачів та запобігати несанкціонованому доступу. Повинні бути реалізовані механізми авторизації, розмежування прав доступу та захисту від типових загроз на стороні клієнту.

9) Інші вимоги:

– програмне забезпечення повинно відповідати чинним стандартам і рекомендаціям щодо розробки вебзастосунків;

– інтерфейс системи має бути адаптивним та коректно відображатися на різних розширеннях екранів;

– система повинна підтримувати можливість подальшого масштабування та розширення чи додавання нового функціоналу.

Висновки до розділу 2

В даному розділі було проведено аналіз сучасного стану інструментарію, моделей та методів. Серед них було визначено структури на які поділяється проєкт: клієнт, сервер та база даних. До даних елементів підібрано та порівняно функціональні можливості та потреби.

Для клієнтською частини оглянуто фреймворки React, Angular та Vue.js. Для реалізації клієнта вебзастосунку обрано React через потребу у створенні SPA-технологій та постійного оновлення вмісту сторінок. Angular, через дуже великий базовий вміст, має дуже високі ресурсні потреби, з іншої сторони Vue.js, навпаки простий та не потребує багато ресурсів системи, але дуже складно підтримується на старих системах.

Для серверної частини розглянуто Node.js та .NET. Для реалізації сервера вебзастосунку обрано Node.js з використанням ExpressJS для реалізації роботи з маршрутами. .NET, має багато функціоналу за замовчуванням, але потребує набагато більше ресурсів системи для роботи.

Для роботи з базою даних розглянуто Microsoft SQL Server Management, MySQL та PostgreSQL. Обрано Microsoft SQL Server Management через додатковий функціонал та кращу підтримку роботи на операційній системі Windows. PostgreSQL потребує більш глибших знань для повного використання даних у системі для оптимізацій, а MySQL, порівняно з можливостями Microsoft SQL Server Management, є дуже легкою програмою з обмеженим функціоналом, але менш важкий для системи.

Також оформлено специфікації вимог до ПЗ та описані призначення та межі проєкту, загальний опис, функції системи, вимоги до інформаційного, технічного, програмного забезпечення, вимоги до зовнішніх інтерфейсів та властивості ПЗ з іншими вимогами.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

3.1 Архітектура програмного забезпечення

Від архітектури програмного забезпечення залежить і як саме користувачі будуть взаємодіяти з вебзастосунком. Загалом проєкт основується на звичайній архітектурі клієнт-сервер (client-server). Це дозволяє декільком клієнтам відправляти запити на сервер, де дані запити будуть оброблятися, та вже готові результати відправлятися назад клієнтам. Якщо одночасно приходе декілька запитів, то з них, сервером, буде формуватися черга запитів, котра сортується за рівнем пріоритетів запитів. Для взаємодії між frontend-частиною проєкту та backend-частиною використовуються правила REST API, через це також всі відправлені чи отримані запити будуть у форматі JSON (.json).

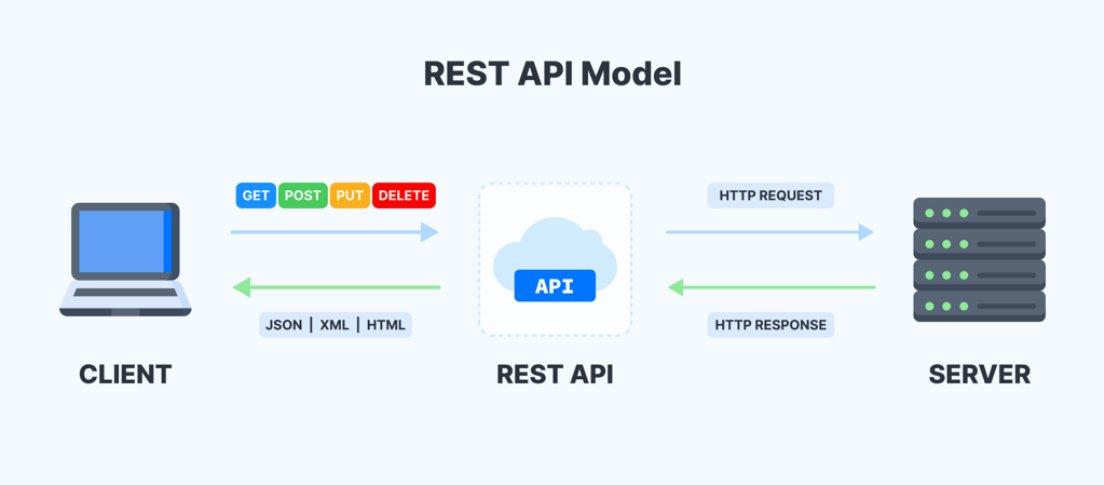

Рисунок 3.1 – Схема роботи правил REST API

Для реалізації даної можливості клієнт (frontend) буде формувати HTTP-запит за такою структурою:

– endpoint, тобто url-адреса сторінки, куди відправляється запит;

– HTTP-метод, вказуємо що саме потрібно зробити (GET, POST, DELETE, PUT);

– заголовок запиту, що зберігає додаткову інформацію типу токена авторизації чи тип контента;

– тіло запиту, що зберігає дані, котрі будуть відправлені (для методів POST, PUT).

При формуванні відповіді сервером, до JSON-відповіді додаються коди відповідей (status codes), котрі далі використовуються клієнтом для відладки відповіді та частини коду, що формує її. Бувають три коди відповідей: 200 – це означає що був успіх та проблем немає, 404 – означає що даних не знайдено, 500 – помилка на стороні сервера.

HTTP-методи надають можливість вказувати, що саме хоче клієнт зробити з даними, котрі будуть посилатися на сервер. Дані методи основуються на чотирьох основних CRUD-операціях: CREATE (Створення), READ (Читання), UPDATE (Оновлення) та DELETE (Видалення). Вони надають можливість керувати даними в базі даних по-різному. Всього можливо виділити п’ять різних HTTP-методів, що дозволяють описати дію з даними клієнтом до сервера:

– GET – це метод котрий основується на операції READ та дозволяє отримати дані з бази даних через сервер, сам метод даних не посилає, тому немає тіла;

– POST – це метод котрий основується на операції CREATE та дозволяє створювати нові дані для бази даних через сервер, через це даний метод має тіло для зберігання даних;

– PUT – це метод котрий основується на операції UPDATE та дозволяє оновлювати існуючі дані для бази даних через сервер, тому даний метод має тіло для зберігання даних;

– PATCH – це метод котрий основується на операції UPDATE, схожий на метод PUT, але дозволяє частково оновлювати існуючі дані для бази даних через сервер, через це даний метод має тіло для зберігання даних;

– DELETE – це метод котрий основується на однойменній операції DELETE та дозволяє видалити дані з бази даних через сервер, сам метод даних не посилає, тому немає тіла;

Краще звернути увагу на той факт, що метод GET тільки переглядає дані, тому він ніколи не повинен, для покращення безпеки, якось чіпати дані в базі даних.

Також методи POST, якщо викликати один і той самий декілька разів, буде створювати непотрібні дублікати даних, тому краще робити перевірку на існування за спеціальним, для даного випадку, id.

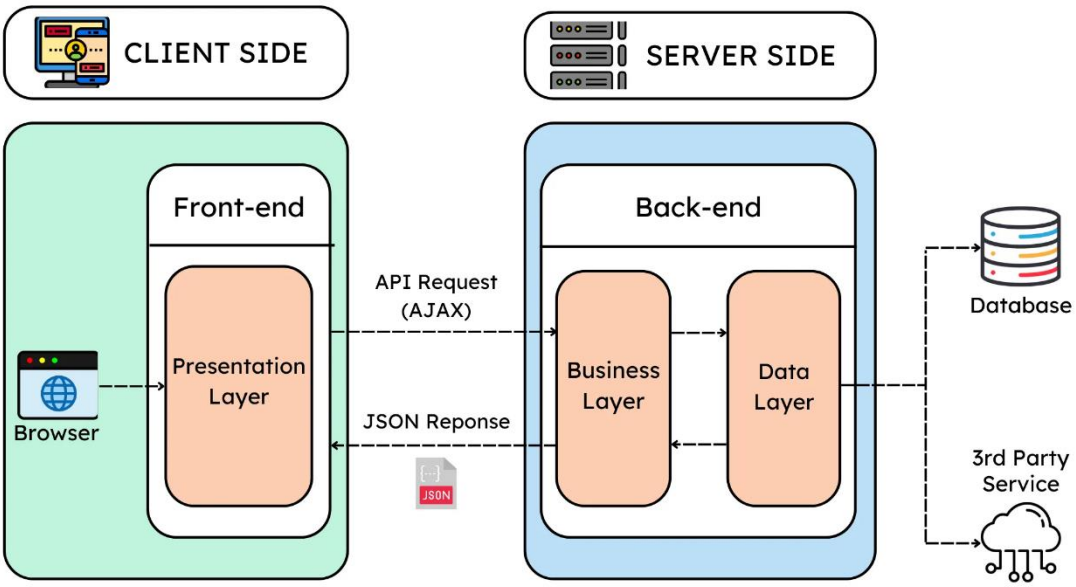

Рисунок 3.2 – Схема роботи SPA-застосунка

В проєкті клієнтська частина вебзастосунка на React створюється як SPA-застосунок. Даний спосіб має перевагу в тому, що всі елементи сторінки відображаються на одній динамічній web-сторінці, що загалом зменшує навантаження на сервер та надає потрібну швидкість користувачеві. Реалізація полягає в тому, що вебзастосунок не буде перезавантажувати сторінку, а оновлювати зміст динамічно, в залежності від дій користувача, та більша частина відповідальності за рендеринг елементів сторінок тепер лежить на клієнтській частині, тобто саме frontend буде вказувати як працювати з перемиканням між екранами, валідацією форм чи як відображати дані. Серверна частина обробляє запити, авторизацію та роботу з базою даних.

3.1.1 Frontend

Клієнтська частина вебзастосунку використовує фреймворк React. Даний фреймворк надає можливість використання компонентного підходу до створення проєкту та зберігання файлів на сторонні розробника. Суть полягає в те, що замість одного файлу, всі потрібні компоненти розділені та знаходяться в різних файлах та

незалежать один від іншого, що дуже спрощує читабельність коду для розробника та надає можливість використати один компонент повторно, а не створювати одну таку ж копію. Для роботи з даними в даних компонентах використовуються state, котрі є даними самого компонента.

Також фреймворк надає бібліотеки для реалізації структури SPA. Для даної цілі існує бібліотека ‘React-router’, що надає елементи <Router>, <Routes> та <Route>, що надають можливість для створення потрібної маршрутизації. За допомогою <Router> можливо обгорнути потрібні елементи для маршрутизації, це надає можливість вебзастосунку бачити елементи для маршрутизації. Компонент <Routes> слугує як контейнер для всіх посилань, котрі знаходяться в ньому. Компонент <Route> описує посилання url, що надає можливість вже вказувати який саме елемент показувати користувачеві в SPA-вебзастосунку.

Для повноцінної роботи в SPA-вебзастосунку на React для переміщення між елементами використовується компонент <Link>, котрий функціонально буде заміняти html-компонент <a>, це надає можливість додавати кнопки користувачеві, що не будуть перезавантажувати сторінку, а змінювати компонент за принципом SPA.

Якщо потрібно зробити щоб блок використовувався як посилання, то це можливо через хук ‘useNavigate()’. Даний хук надає можливість саме коду виконувати перехід між елементами, наприклад після виконання певного коду.

Через правила SPA, більшість сторінок потрібно збирати з запропонованих елементів та отриманих даних через сервер з бази даних, тобто вони будуть динамічними, тому для вирішення цієї задачі можливо використати хук ‘useParams()’, котрий дозволяє отримати параметри URL. Це дозволить створювати повністю динамічні сторінки, записуючи їх так: ‘<Route path=’/game/:id’ element={<ProductPage/>}/>’, та де id буде вказувати саме з якого рядка таблиці в базі даних брати дані для оформлення сторінки.

3.1.2 Backend

Серверна частина вебзастосунку використовує можливості Node.js, але для зручності та додаткових можливостей використовується саме фреймворк

Express.js. Базово Node.js вже самостійно вміє приймати HTTP-запити та повертати відповіді клієнтові, але за допомогою Express.js можливо реалізувати обробку маршрутів, що задовольняє принципи RestAPI.

Для захисту Express.js надає можливість реалізовувати різний middleware. У проєкті використовуються токени JWT (JSON Web Token), що містять дані у json та закодовані у ‘Base64URL’. Використовуються ці токени для функціювання системи користувачів в проєкті та містять в собі закодовані дані про користувачів, котрі можливо використати при авторизації чи інших діях у системі логіну аккаунтів.

За допомогою API в Express.js реалізується робота з базою даних, котра надає можливість кидати запити в базу даних та отримувати дані з неї. Для помітки об’єкта запиту використовується ‘req’ (request/запит). Він може містити в собі різні параметри, від url до cookies чи даних користувача, та потрібен щоб отримати дані саме від клієнта. Як об’єкт відповіді використовується ‘res’ (response/відповідь). Даний об’єкт містить собі як і json-файли, так і html-файли, та потрібен щоб відправити дані до клієнта.

Прямі запити до бази даних зберігаються в окремій папці repositories, а визивання цих запитів та робота з даними відбувається через controllers, потім в routes йде налаштування шляху, через котрий клієнт зможе знайти дані.

3.1.3 База даних

Зберігання та робота з таблицями бази даних через Microsoft SQL Server Management надає перевагу у гнучкості функціоналу у сервері. Підключення до самого SQL Server Management відбувається через можливості Node.js та Express.js. Для підключення до бази даних вказуються: ім’я користувача (user), пароль (password), сервер (server), база даних (database) та, якщо потрібно то додаткові налаштування (options). В даному випадку опція ‘encrypt’ має статус false, через її непотрібність, але опція ‘enableArithAbort’ має статус true. Функція ‘ArithAbort’ надає можливість припинити дію, коли йде переповнення в даних чи відбулась помилка через спробу ділення на 0, та поверне помилку. Самі дані для заповнення та підключення не вказуються прямо в файлі, а зберігаються окремо в спеціальному

.env-файлі. Після цього на сервері, використовуючи функцію sql.connect, з спеціальної бібліотеки ‘sql’, відбувається підключення до бази даних.

В проєкті для зберігання товарів є окрема таблиця, але для зберігання її різних характеристик використовуються окремі таблиці та таблиці, що зв’язують ці дані між собою, через особисті автоікрементуючі id. Для цього на сервері та клієнті зберігаються constant-файли, котрі вказують в собі які існують параметри характеристик товару чи інших елементів, та, якщо потрібно, які саме стилі задаються при певній характеристики.

3.2 Функції системи

Всі функції системи повинні якось отримувати та взаємодіяти з даними. Дані про товар, користувачів, характеристики чи інші дані зберігаються в базі даних, на сервері створюються потрібні запити, котрі зберігаються в repositories, та потім використовуються в controllers, оброблена інформація передається з сервера через посилання, котре створюється в routes. Клієнт для отримання даних переходить на потрібне посилання яке вказане у відповідному сервіс-файлі service, далі через функції сторінки або інших файлів-утиліт йде обробка даних та взаємодія з ними, наприклад вивід даних.

Будь які картинки зберігаються на сервері в папці uploads з спеціальним розширення файлу .webp. Для доступу до них зберігається посилання на точне місце ‘imageUrl’ в таблиці де зберігається інформація про картинки товарів чи користувачів вебзастосунка, де також зберігається ‘ProductId’, котрий буде вказувати до котрого продукту належить картинка. Доступ клієнту до картинки дається такий же, як і до всіх даних, через можливості routes на сервері.

3.2.1 Перегляд каталогу товарів

Базова функція, котра наявна в усіх подібних вебзастосунків. За допомогою даної функції користувачі зможуть дізнатися про вміст вебзастосунка та перейти до сторінки товару, щоб отримати ще більше інформації про продукт. Для виводу каталогу товарів з бази даних потрібно брати id продукту, котрий буде використовуватись в url, назву товару, посилання на картинку, усі види цін, тип

продукту та чи наявний він в таблицях пов’язаних з кошиком чи списком бажаного. Ці дані потрібні саме для виводу коротких даних про товар та стан, чи він в кошику чи в списку бажаного.

Таблиця 2 – Сценарій використання для перегляду каталогу товарів

Usecase section	Comment
Use Case Name	Перегляд каталогу товарів
Scope	Вебзастосунок
Level	Успішно отримати інформацію про товар, вивід правильних стилей
Primary Actor	Користувач
Stakeholders and interests	1) Користувач – перегляд товарів у вебзастосунку.
Preconditions	Клієнт-сервер, node-сервер працюють, підключені до бази даних, інформація існує в базі даних
Success guarantee	1) Клієнт використовує браузер, що підтримує HTML5 та JavaScript. 2) Клієнт має використовувати стабільне підключення до мережі Інтернет. Клієнт вводить правильне посилання вебзастосунку.
Main Success Scenario	1) Клієнт відкриває сторінку; 2) Клієнт знаходить кнопку ‘Список товарів’; 3) Клієнт натискає кнопку; 4) Система відкриває клієнту сторінку з товарами; Інформація про товари отримана, сторінка виводить товари;
Extensions	– Сервер не працює: 1) Клієнт відкриває сторінку; 2) Клієнт отримує помилку браузера про відсутність підключення; – Дані в базі даних відсутні: 1) Клієнт відкриває сторінку; 2) Клієнт знаходить кнопку ‘Список товарів’; 3) Клієнт натискає кнопку; 4) Система відкриває клієнту сторінку з товарами; Товари на сторінці відсутні.
Special Requirements	Швидкість доступу до мережі Інтернет клієнту має бути більшою ніж 60 кб/сек.
Frequency of Occurrence	90%
Miscellaneous	Чи необхідно виводити окреме повідомлення користувачу про відсутність товарів.
Stakeholders and interests	1) Користувач – перегляд товарів у вебзастосунку.
Preconditions	Клієнт-сервер, node-сервер працюють, підключені до бази даних, інформація існує в базі даних

3.2.2 Авторизація користувача

Якщо користувач не увійшов в аккаунт, то дії які потребують інформації про користувача будуть відсилати його на елемент авторизації аккаунту, або користувач сам зможе перейти до нього, через спеціальну кнопку в шапці вебзастосунку. Якщо користувач ще немає аккаунту на вебзастосунку, в нього буде можливість реєстрації через спеціальне посилання на елементі авторизації.

При авторизації користувачу потрібно вказати email та пароль, котрі будуть надіслані для зберігання в спеціальну таблицю в базі даних через сервер. Для паролю додавання перевірки можливо завдяки бібліотеці ‘regex’ на серверній частині. Для перевірки можливо працювати з великими літерами, цифрами чи спеціальні символи. Також можливо вказати мінімальну потрібну кількість символів для затвердження паролю. При надсиланні паролю, за допомогою можливостей бібліотеки ‘bcrypt’, він хешується вказаним методом, та вже зберігається в базі даних захешованим.

Таблиця 3 – Сценарій використання для авторизації користувача

Usecase section	Comment
Use Case Name	Авторизація користувача
Scope	Вебзастосунок
Level	Успішно пройти процес авторизації в вебзастосунку, отримати доступ до власного профілю.
Primary Actor	Користувач
Stakeholders and interests	1) Користувач – замовлення подальших товарів, перегляд власного профілю, зміни налаштувань, перегляд списку замовлень.
Preconditions	Клієнт має бути зареєстрований, його профіль має бути активований та підтвердженим, знаходиться на сторінці вебзастосунку.
Success guarantee	1) Клієнт використовує браузер, що підтримує HTML5 та JavaScript. 2) Клієнт має використовувати стабільне підключення до мережі Інтернет. Клієнт має вводити коректні дані профілю (електронну пошту та пароль).
Main Success Scenario	1) Клієнт знаходить кнопку ‘Login’; 2) Клієнт натискає на кнопку; 3) Система відкриває розділ авторизації; 4) Клієнт вводить пошту та пароль у спеціалізовані поля; 5) Клієнт натискає кнопку вхід; Система відкриває клієнту сторінку з його профілем.

Кінець таблиці 3

Extensions	– Користувач не вводить дані: 1) Клієнт знаходить кнопку ‘Login’; 2) Клієнт натискає на кнопку; 3) Система відкриває розділ авторизації; 4) Клієнт залишає поля вводу даних незаповненими; 5) Клієнт натискає кнопку вхід; 6) Система видає повідомлення про незаповнені поля вводу даних. – Користувач допускає помилку при заповненні даних: 1) Клієнт знаходиться кнопку ‘Login’; 2) Клієнт натискає на кнопку; 3) Система відкриває розділ авторизації; 4) Клієнт неправильно вводить пошту та пароль у спеціалізовані поля; 5) Система повідомляє про невірно задані дані.
Special Requirements	Швидкість доступу до мережі Інтернет клієнту має бути більшою ніж 60 кб/сек.
Frequency of Occurrence	80%
Miscellaneous	Чи необхідно проводити перевірку наявності пошти користувача у базі даних при неправильно введених даних. На котрий час давати доступ до профілю після успішної авторизації.

3.2.3 Оформлення замовлення

Для оформлення замовлення користувачу потрібно буде увійти в аккаунт та додати товари до кошика. Факт зберігання товарів саме в даному кошика даного користувача реалізується використовуючи дві таблиці в базі даних. В першій таблиці вказується кошик з унікальним id та id користувачем, до котрого цей кошик відноситься, цей кошик створюється вперше коли користувач додає хоча б один товар до свого кошика. В другій таблиці зберігається id кошика та id товарів, котрі кошик зберігає в собі, при кожному додаванні чи видаленні з кошика товарів сервер оновлює вміст в базі даних та посилає, що треба оновити вміст на клієнті. Якщо це фізичний товар – то користувач зможе збільшити його кількість в кошику, якщо це інший тип товару – ні.

Коли користувач додає товар до кошика то відбувається заморозка ціни на короткий час. Інформація про заморожену ціну зберігається в другій таблиці, а саме буде додаватись інформація яка сама ціна заморожена та до котрого часу.

Заморозка ціни надає можливість користувачу купити товар за ‘старою ціною’ при кінці знижках.

Таблиця 4 – Сценарій використання для оформлення замовлення

Usecase section	Comment
Use Case Name	Оформлення замовлення
Scope	Вебзастосунок
Level	Успішно пройти процес оформлення замовлення в вебзастосунку.
Primary Actor	Користувач
Stakeholders and interests	1) Користувач – оформлення нового замовлення з товарами.
Preconditions	Клієнт має бути зареєстрований, його профіль має бути активований та підтвердженим, знаходиться на сторінці вебзастосунку, товари наявні у вебзастосунку.
Success guarantee	1) Клієнт використовує браузер, що підтримує HTML5 та JavaScript. 2) Клієнт має використовувати стабільне підключення до мережі Інтернет. Кошик клієнта не пустий, містить хоча б один товар.
Main Success Scenario	1) Клієнт знаходить кнопку кошика; 2) Клієнт натискає на кнопку; 3) Система відкриває розділ кошика; 4) Клієнт знаходить кнопку для оформлення замовлення; 5) Клієнт натискає на кнопку; 6) Система відкриває розділ оформлення замовлення; 7) Клієнт заповнює потрібні дані; Клієнт натискає на кнопку оформити замовлення.
Extensions	– Користувач не вводить дані: 1) Клієнт знаходить кнопку кошика; 2) Клієнт натискає на кнопку; 3) Система відкриває розділ кошика; 4) Клієнт знаходить кнопку для оформлення замовлення; 5) Клієнт натискає на кнопку; 6) Система відкриває розділ оформлення замовлення; 7) Клієнт не вводить дані; 8) Клієнт натискає на кнопку оформити замовлення. 9) Система видає повідомлення про пусті поля. – Кошик користувача пустий: 1) Клієнт знаходить кнопку кошика; 2) Клієнт натискає на кнопку; 3) Система відкриває розділ кошика; 4) Клієнт знаходить кнопку для оформлення замовлення; 5) Клієнт натискає на кнопку; Система видає повідомлення про відсутність товарів в кошику;

Кінець таблиці 4

Special Requirements	Швидкість доступу до мережі Інтернет клієнту має бути більшою ніж 60 кб/сек.
Frequency of Occurrence	80%
Miscellaneous	Якщо товар був доданий в кошик перед закінченням знижки, у користувача надається вікно у п’ять хвилин для використання знижкової ціни для товару після закінчення даної знижки

3.2.4 Керування товарами

Доступ до елементів даної функції та до самої функції відбувається через перевірку наявності ролі на аккаунті. Кожного разу коли продавець робить дію, що потребує отримання даних чи надсилання нових чи оновлених, відбувається перевірка на роль у користувача. Всі дії доступні усім користувачам за роллю продавця або, якщо в їх ролі є батьківська роль продавець. Також для отримання доступу до редагування своїх товарів, йде перевірка у таблицях базі даних чи належить товар даному користувачу.

Доступ до керування з’являється через спеціальну сторінку на шапці вебзастосунку. До кожного свого товару, продавець зможе побачити спеціальні статуси товару та деякі дії: редагувати інформацію про товар, видалити товар, побачити історію товару чи статистику.

Таблиця 5 – Сценарій використання для керування товарами

Usecase section	**Comment**
Use Case Name	Керування товарами
Scope	Вебзастосунок
Level	Успішно отримати доступ до сторінки керування товарами.
Primary Actor	Продавець
Stakeholders and interests	1) Продавець – доступ до керування своїх товарів.
Preconditions	Продавець має бути зареєстрований, його профіль має бути активований та підтвердженим, знаходиться на сторінці вебзастосунку, немає обмежень щодо можливостей керування товарами, має роль продавця.

Кінець таблиці 5

Success guarantee	1) Клієнт використовує браузер, що підтримує HTML5 та JavaScript. 2) Клієнт має використовувати стабільне підключення до мережі Інтернет. 3) Клієнт має роль продавця. 4) Клієнт немає обмежень щодо сторінки керування товарами.
Main Success Scenario	1) Клієнт знаходить кнопку для панелі продавця; 2) Клієнт натискає на панель; 3) Система відкриває панель продавця; 4) Клієнт знаходить кнопку для сторінки керування товарами; 5) Клієнт натискає на кнопку; 6) Система відкриває розділ керування товарами;
Extensions	– Клієнт немає ролі продавця: 1) Клієнт не знаходить кнопку для панелі; – Клієнт має обмеження на можливість керування товарами: 1) Клієнт знаходить кнопку для панелі; 2) Клієнт натискає на панель; 3) Система відкриває панель продавця; 4) Клієнт знаходить кнопку для сторінки керування товарами; 5) Клієнт натискає на кнопку; 6) Система виводить повідомлення про відсутність прав доступу до системи керування товарами.
Special Requirements	Швидкість доступу до мережі Інтернет клієнту має бути більшою ніж 60 кб/сек.
Frequency of Occurrence	80%
Miscellaneous	Якщо товар був доданий в кошик перед закінченням знижки, у користувача надається вікно у п’ять хвилин для використання знижкової ціни для товару після закінчення даної знижки

3.2.5 Рецензії товару

Після того, як користувач зробив успішне замовлення з товаром, йде запис що цей товар наявний у користувача. Це надає можливість користувачеві залишати відгуки про товар.

Відгуки записуються через сервер в базу даних та зберігає в собі який користувач залишив відгук, коментар відгуку, рейтинг (1-5), рекомендація та до якого товару рецензія відноситься. Коли користувач залишає відгук є варіант залишити оцінку для системи оцінки від 1 до 5 та коментар.

Оцінки на відгуках всіх користувачів формує середній бал товару. Це використовується для можливості фільтрації при пошуку продукту на сторінках вебзастосунку. Також на сторінці товару вказується медіана оцінок та загальна кількість залишених відгуків на товар.

Таблиця 6 – Сценарій використання для рецензій товару

Usecase section	**Comment**
Use Case Name	Рецензії товару
Scope	Вебзастосунок
Level	Успішно пройти процес оформлення замовлення в вебзастосунку.
Primary Actor	Користувач
Stakeholders and interests	1) Користувач – оформлення нової рецензії до товару.
Preconditions	Клієнт має бути зареєстрований, його профіль має бути активований та підтвердженим, знаходиться на сторінці вебзастосунку товару, товари наявні у вебзастосунку та клієнт придбав товар.
Success guarantee	1) Клієнт використовує браузер, що підтримує HTML5 та JavaScript. 2) Клієнт має використовувати стабільне підключення до мережі Інтернет. 3) Клієнт придбав товар. 4) Товар наявний в базі даних. 5) Клієнт має доступ до оформлення рецензій.
Main Success Scenario	1) Клієнт знаходить форму для оформлення рецензії на придбаний товар; 2) Клієнт заповнює та обирає дані; 3) Клієнт натискає на кнопку для додавання рецензії; 4) Система виводить створену рецензію на сторінці товару та надає клієнтові форму для редагування рецензії;
Extensions	– Користувач не вводить дані: 1) Клієнт знаходить форму для оформлення рецензії на придбаний товар; 2) Клієнт не заповнює дані; 3) Клієнт натискає на кнопку для додавання рецензії; 4) Система виводить повідомлення про пусті поля; – Користувач не придбав товар: 1) Клієнт знаходить форму для оформлення рецензії на придбаний товар; 2) Система виводить повідомлення про відсутність можливість залишення рецензій на не придбаний товар;
Special Requirements	Швидкість доступу до мережі Інтернет клієнту має бути більшою ніж 60 кб/сек.

Кінець таблиці 6

Frequency of Occurrence	70%
Miscellaneous	-

3.3 Моделювання системи

3.3.1 Модель діаграми класів

Діаграма класів, наведена на рисунку А.1 в додатку А, відображає логічну структуру системи вебзастосунку. На даній діаграмі відображаються класи, їх атрибути та операції, що показують пов’язану дію, та відношення між класами.

Дані в таблицях типу CheckoutSessions, UserDeniedPermissions, Wishlist, Notifications, ReleaseSubscriptions не можуть існувати без користувача в User, тому вони мають відношення композиції. Також таблиці ProductTypes, ProductImages мають відношення агрегації через доповнення таблиці Games.

Головними таблицями є Games та User, бо вокруг них існують багато інших таблиць для взаємодії та прямої взаємодії між даними таблицями немає. Таблиці типу Categories, Platforms, InputSupports, Tags зберігають, відповідно до назви, інформацію, але не напряму взаємодіють з таблицею Games, а через відповідних посередників GamesCategories, GamePlatforms, ProductInputSupports, ProductTags.

Таблиця SaleCampaigns описує діючі знижки та SaleCampaignGames існує щоб описати які саме входять товари до якої знижки, через це між собою вони мають відношення композиції. Теж саме відбувається і з таблицями Orders та OrderItems, що відповідно описують замовлення та вміст замовлення, та таблицями Carts та CartItems, що відповідно описують кошик та вміст кошика користувача. Reviews описує написані користувачем рецензії до товару, тому між даною таблицею та User відношення композиції, бо видаляючи користувача видаляються пов’язані до нього рецензії.

Ролі та взаємодії користувача можуть існувати без самого користувача, але запретні дії саме для даного користувача в таблиці UserDeniedPermissions не можуть існувати окремо від користувача, видаляються разом з користувачем, тому мають відношення композиції.

Кожен користувач може зробити підписку до товару, що ще не вийшов через таблицю ReleaseSubscriptions, але якщо видалити користувача то сама підписка також видаляється, тому відношення композиції.

3.3.2 Модель діаграми взаємодії

Діаграма взаємодії, наведена на рисунку А.2 в додатку А, відображає взаємодію логічних елементів вебзастосунку між собою в процесі виконання певного варіанту використання, що запускається користувачем. На діаграмі взаємодії зображується увесь процес взаємодій елементів клієнту, серверу та бази даних у процесі реєстрації нового користувача, з умовою, що пошта не використана ще в базі даних.

Користувач посилає на клієнт пошту та пароль, де на сторінці відразу відбувається перевірка на виконання кожної умови для затвердження даних. Після перевірки сторінка за допомогою функції register посилає дані в authService.js. Для посилання даних на node-сервер створюється посилання в api.js за допомогою бібліотеки axios, та задається тип повідомлення POST, для відправки даних. Axios передає дані на сервер через HTTP-запит, де отримується в файлі authRotes.js за допомогою можливостей Express.js router. У відповідній функції посилаються далі дані в AuthController.js, де на початку викликається перевірка користувача за поштою через функцію findUserByEmail(email). Функція надсилає на базу даних запит ‘SELECT User BY Email’ через можливості бібліотеки mssql, та з попереднім підключенням до серверів бази даних Microsoft SQL Server Management. Успішним кроком вважається коли база даних у відповідь надсилає пустоту – null, що означає що дана пошта ще не використовувалась для створення аккаунту в вебзастосунку. Після даної процедури файл AuthController.js отримує відповідь про успіх та починає роботу з паролем: перевірка складності пароль через функцію isStrongPassword(password), хешування пароль за допомогою функції bcrypt.Hash(password). Після цього на репозиторії викликається запит для створення користувача з поштою, хешованим паролем та токеном для пошти, що викликає запит ‘INSERT INTO Users’. При отриманні у відповідь успіх, контролер надсилає на пошту користувача лист, використовуючи можливості бібліотеки

nodemailer. Для реалізації можливості підтвердження через пошту, створюється спеціальне посилання за допомогою токена для пошти, по котрому користувачу потрібно перейти для підтвердження даних. Використовуючи функції transporter.sendMail(mail), де в mail вказується від якої пошти прийде посилання, до якої пошти прийде посилання (користувача), об’єкт посилання, та рядок html – що вказує на вигляд вмісту посилання в пошті через елементи html. При переході користувача на пошту та натисканні кнопки, де знаходиться посилання на сторінку в клієнті, пересилається на спеціальну сторінку, котра створюється в результаті спеціального посилання за допомогою токена для пошти. Факт переходу користувача на дану сторінку запускає функцію useSearchParams.get(token), що шукає в посиланні сторінки токен та надсилає обратно на сервер. Після отримання токену в authRoutes, викликається verifyEmail(req, res) для пересилання токену в AuthController.js, де потім відбувається перевірка токена з клієнта з токеном в базі даних. В базі даних йде пошук за токеном з клієнта через запит – ‘SELECT * FROM Users WHERE EmailToken = token’, успіхом вважається знаходження користувача за токеном. При успішному знаходженні користувача за токеном, база даних відправляє повідомлення про успіх та користувача на сервер, що передаються через AuthRepository.js до AuthController.js, який після отримання повідомлення про успіх, викликає функцію verifyUser(user.Id), що підтеджує реєстрування користувача шляхом запитом ‘UPDATE Users SET IsVerified = 1, EmailToken = NULL WHERE Id = userId’, де IsVerified – відповідає чи підтверджений користувач, де 0 – це ні, 1 – так, EmailToken – відповідав за токен, котрий передається на клієнт та отримується назад через сервер для відповідності користувача. Після успішних внесених змін в базу даних, відправляється повідомлення про успіх на клієнт та надається можливість використати нові дані для авторизації в акаунт.

3.3.3 Модель вебзастосунку

Усі вебзастосунки можуть ділитися на складові, що взаємодіють між собою та надають можливість повноцінної роботи ідеї вебзастосунку. Вебзастосунок інтернет-магазину складається з трьох основних складових: web browser, web server та database server.

Web-browser складається з React клієнт-застосунку та socket.io. Клієнт-застосунок надає можливість усім користувачам взаємодіяти з елементами клієнту, надавати та отримувати дані через веб-сервер до бази даних та навпаки. При переключені елементів, елемент робить запит на сервер, де вже той робить запит на бази даних для отримання чи додавання з оновленням даних. Через React можливо використовувати можливості SPA для швидкої дії та не навантажувати сервер через факт однієї сторінки. Більшість посилань на web-server відбувається через HTTP-запити.

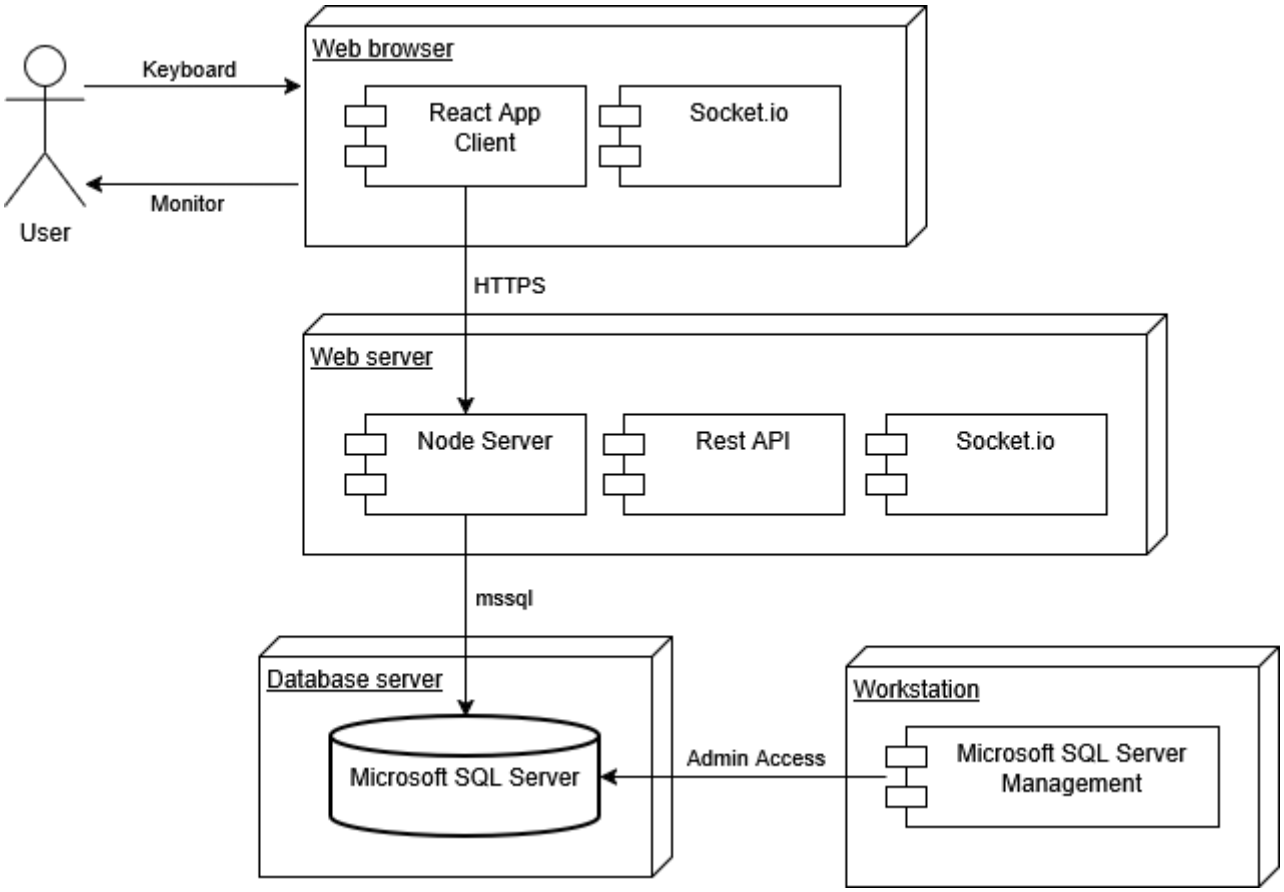

Рисунок 3.3 – Діаграма розгортання вебзастосунку

Звичайна взаємодія клієнта та сервера відбувається через посилання htpp-запитів, але за допомогою socket.io можливо створювати постійне з’єднання, котре працює в реальному часі, клієнта з сервером. Для роботи socket.io потрібен на клієнті та сервері. Це надає можливість отримувати та оновлювати дані на виводі сторінки не перезавантажуючи її. Прикладом результату socket.io є онлайн-статуси аккаунтів чи повідомлення в реальному часі. Також вебзастосунок використовує

можливості socket.io для отримання даних про кошик та список бажаного аккаунту в реальному часі, що також надає можливість оновлювати показники в шапці вебзастосунку в реальному часі. Socket.io працює на основі подій (events), де як і клієнт так і сервер можуть відправляти (emit) чи слухати (on) подію.

Web-server слугує мостом між web-browser та database server. Даний сервер отримує дані чи посилає їх до клієнта або бази даних. Також на даному сервері відбуваються додаткові перевірки перед пересилання даних на інші сервера. Сервер слугує також і місцем, де зберігаються всі картинки, що використовуються на клієнті, та log-файли дій чи снапшоти версій продуктів. Роботи (jobs) на сервері дозволяють оновлювати дані, котрі мають час дій. Викликаються роботи у заданий час через можливості бібліотеки ‘node-cron’. Посилання запитів на database server чи отримання даних, відбувається через можливості бібліотеки ‘mssql’, що надає можливість підключитись до бази даних Microsoft SQL Server Management та вставляти ці запити через дану програму. Web-server посилає дані на web-browser, окрім можливостей socket.io, через створення шляхів, до котрого потім підключається клієнт та отримує дані.

Також наявний блок ‘Workstation’, що описує Microsoft SQL Server Management, котрий є програмою доступу до бази даних розробниками вебзастосунку. Всі користувачі можуть взаємодіяти з вебзастосунком, використовуючи звичайні для даної потреби інструменти, як-от клавіатура, та отримувати візуалізацію через звичайні пристрої з монітором, як от монітор комп’ютера чи екран телефону.

3.4 Макети інтерфейсу користувача

Header та footer буде з’являтися на кожної сторінці вебзастосунку, що надає можливість переміщуватися між елементами клієнта та відображати деяку інформація, наприклад кількість товарів в кошику. У footer зберігається не дуже важлива інформація звичайному користувачеві, наприклад юридична інформація чи угода підписника, але деяка інформація, наприклад посилання на підтримку, знадобиться користувачеві.

Для всіх фонів сторінок та елементів використовують темні кольори, а для виділення тексту чи елементів для взаємодії, використовуються виділяючи палітри. Також за допомогою кольорів можливо виділити стан об’єкта, наприклад якщо товар можливо додати в кошик – кнопка зелена, якщо ні – то кнопка сіра.

На сторінці товарів, самі продукти зображаються у виглядів окремих блоків, що збираються з отриманою з бази даних інформації про товар, з короткою інформацією: картинкою, назвою, платформою, сформованим рейтингом, розробником та цінник з кнопкою купити, що додає товар до кошика. Це надає можливість отримувати головну користувачеві інформацію про товар – що це за товар, про що даний товар, що відображається або в назві, або через картинку, хто це зробив, рейтинг товару та ціна.

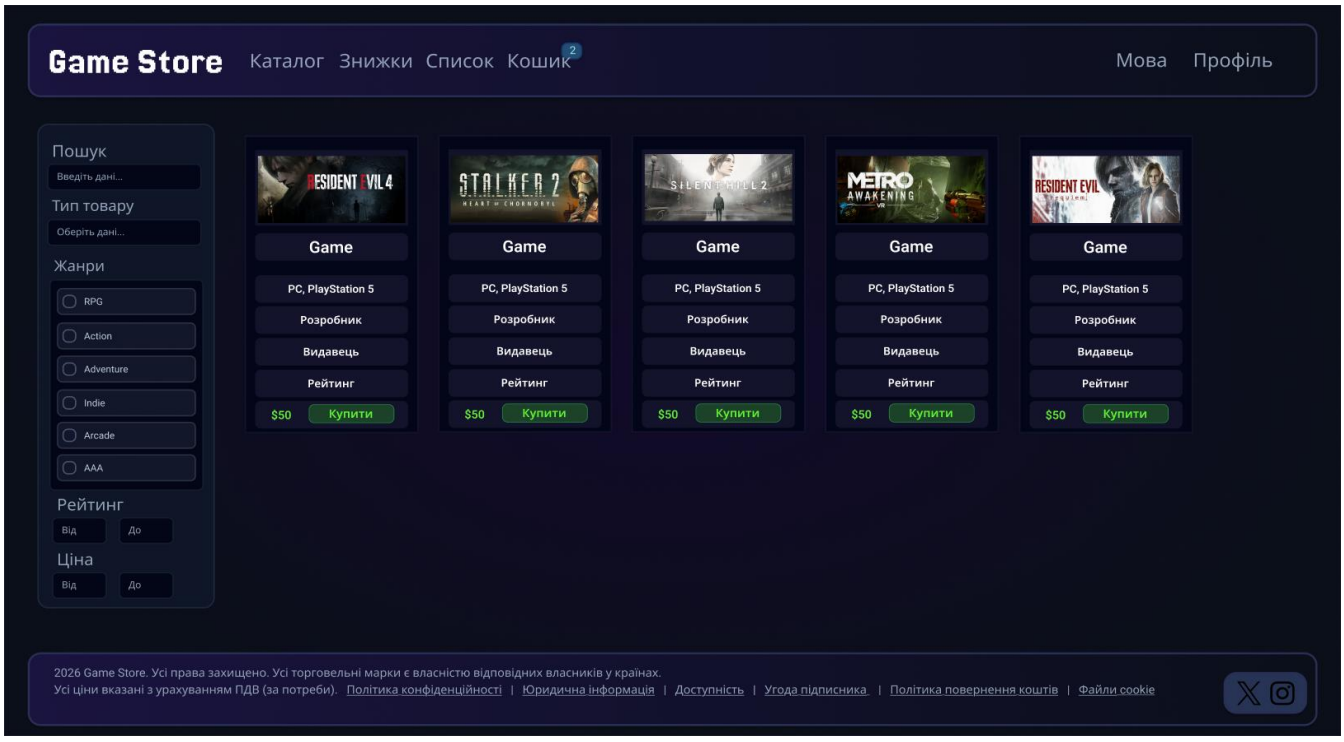

Рисунок 3.4 – Макет сторінки списку товарів

Зліва знаходиться панель, що надає можливість користувачеві робити більш точний пошук використовуючи поле для пошуку по словам в назві чи в описі, або запропонованими способами через різні характеристики товару: за типом, за жанром, за рейтингом, за ціною та іншими. Це дозволяє користувачу не шукати потрібний товар на різних сторінках, а відразу знайти потрібний за допомогою

пошука чи зацікавитись схожими товарами використовуючи можливості фільтрів. Кнопки виділяються більш яскравими кольорами для показу можливості взаємодії з ними. Також для інформації котра потребує особливої увагу бажано використовувати більш яскраві та виразні кольори, порівняно з іншими елементами сторінки, для привертання особливої увагу користувача.

На сторінці товару відображається більше інформації про сам товар: більше медіа-галереї, опис товару, мінімальні та рекомендовані системні вимоги, пов’язані до опису товару теги, рейтинг зі сторонніх довірених вебсайтів, що надають рейтинг критиків, рейтинг гравців вебсайту та рецензії залишені користувачами з описом.

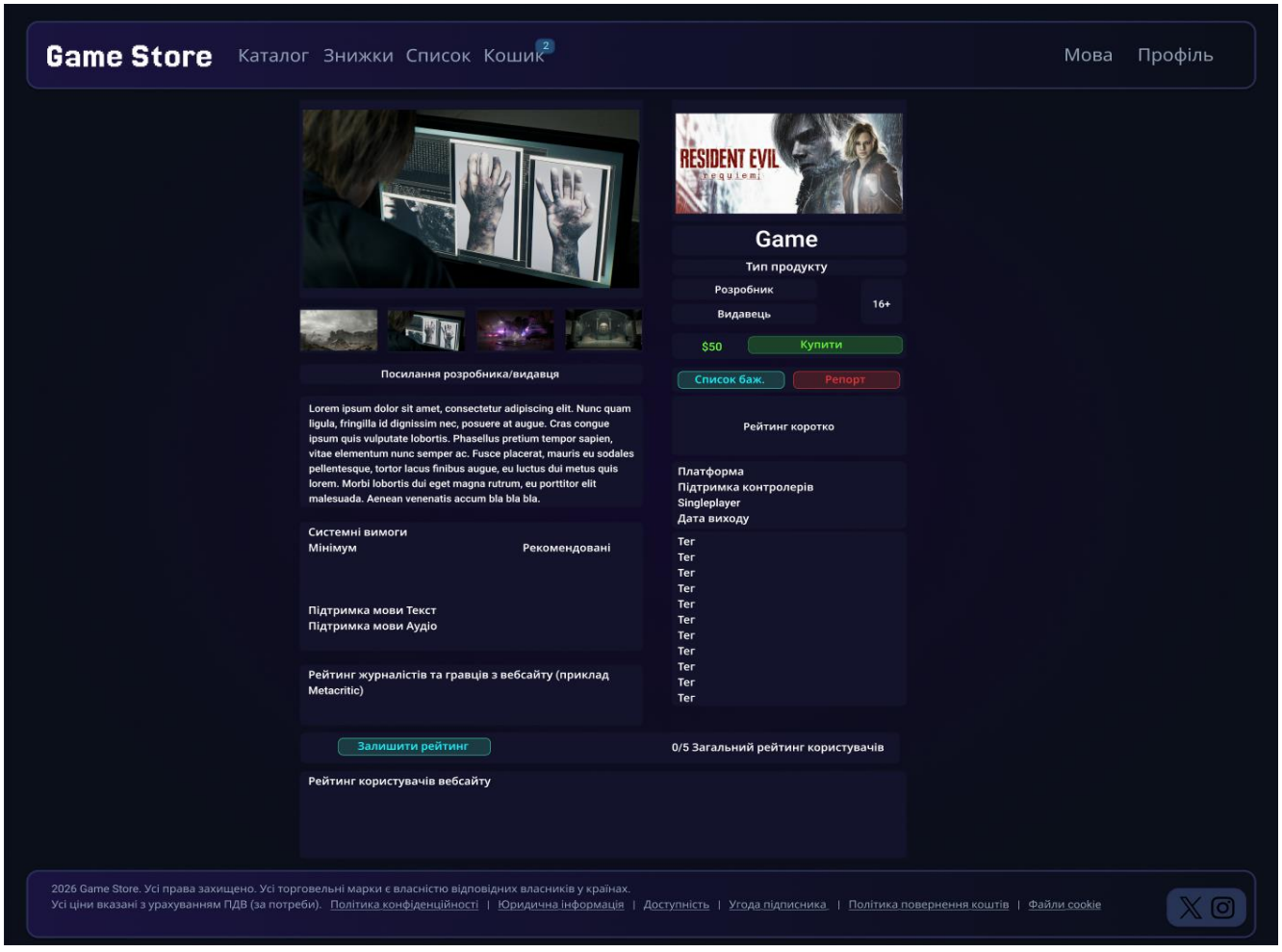

Рисунок 3.5 – Макет сторінки товару

Додатково продавці можуть залишати посилання на корисні вебсайти чи на свої сторінки в соціальних мережах. Додатково у користувача також є можливість

додаткової взаємодії з продуктом: додати до списку бажаного для відслідковування, пожалуватися на товар, бо він має несумісний вміст чи має інші проблеми, або, якщо товар був придбаний користувачем, то надається можливість залишити рецензію на товар, поставити оцінку, рекомендацію та залишити опис, також інші користувачі зможуть оцінити рецензію поставивши лайк чи дизлайк та рецензія з найбільшою кількості лайків буде вважатися найкориснішою. Кнопки на сторінці продукту виділяються більш яскравими кольорами для показу можливості взаємодії.

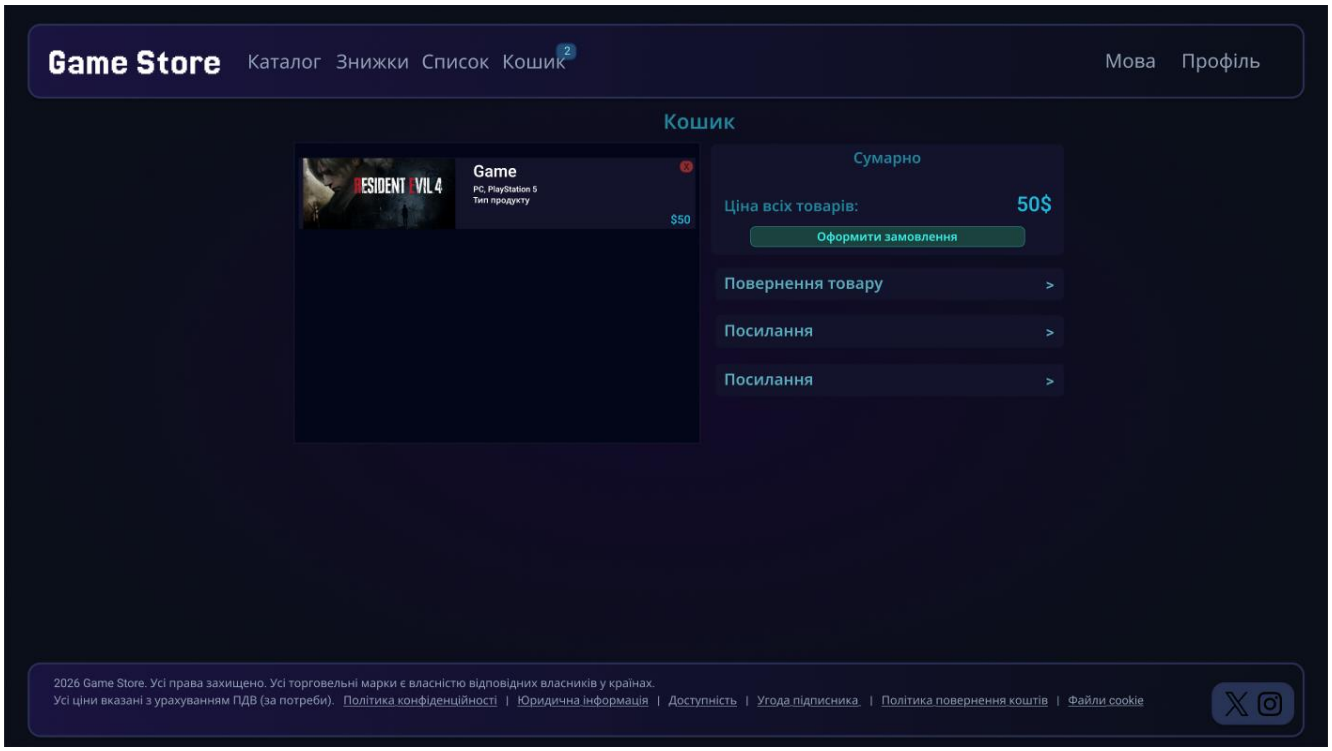

Рисунок 3.6 – Макет сторінки кошика

Сторінка кошика зображає усі додані товари, з можливістю видалення, та відображає картинку, тип товару, платформу, назву та ціну товару. Також якщо наявно декілька товарів то клієнтська частина зможе порахувати ціну всіх товарів та вивести її користувачеві. Кнопка ‘Оформити замовлення’ надає можливість користувачеві перейти, з вмістом кошика, на відповідну сторінку. Наявні корисні посилання зможуть допомогти користувачу зорієнтуватися в питаннях щодо повернення товару або чим ліцензія товару відрізняється фізичною копією для інтернет-магазину чи в інших подібних питаннях. При знижках на товарі

відображається нова ціна та стара буде закреслена, зменшена, та використовувати сірий колір. Також можливо однією кнопкою вичистити увесь вміст кошика, не потребуючи видаляти кожний товар окремо, через кнопку, котра знаходиться у кожного товару. Нижче на сторінці виводяться подібні продукти до вмісту кошика користувача для зацікавлення в подальшої купівлі даних товарів.

Сторінка для оформлення замовлення надає можливість користувачу заповнити потрібні дані для замовлення, якщо наявний фізичний товар – то вказати локацію куди потрібно доставити та чи можливо це виконати. При всіх видів товарів потрібно вказати базову інформацію: ім’я, прізвисько, номер телефону, країна, адреса та обрати спосіб оплати, що надає можливість оплатити вказану ціну всіх товарів.

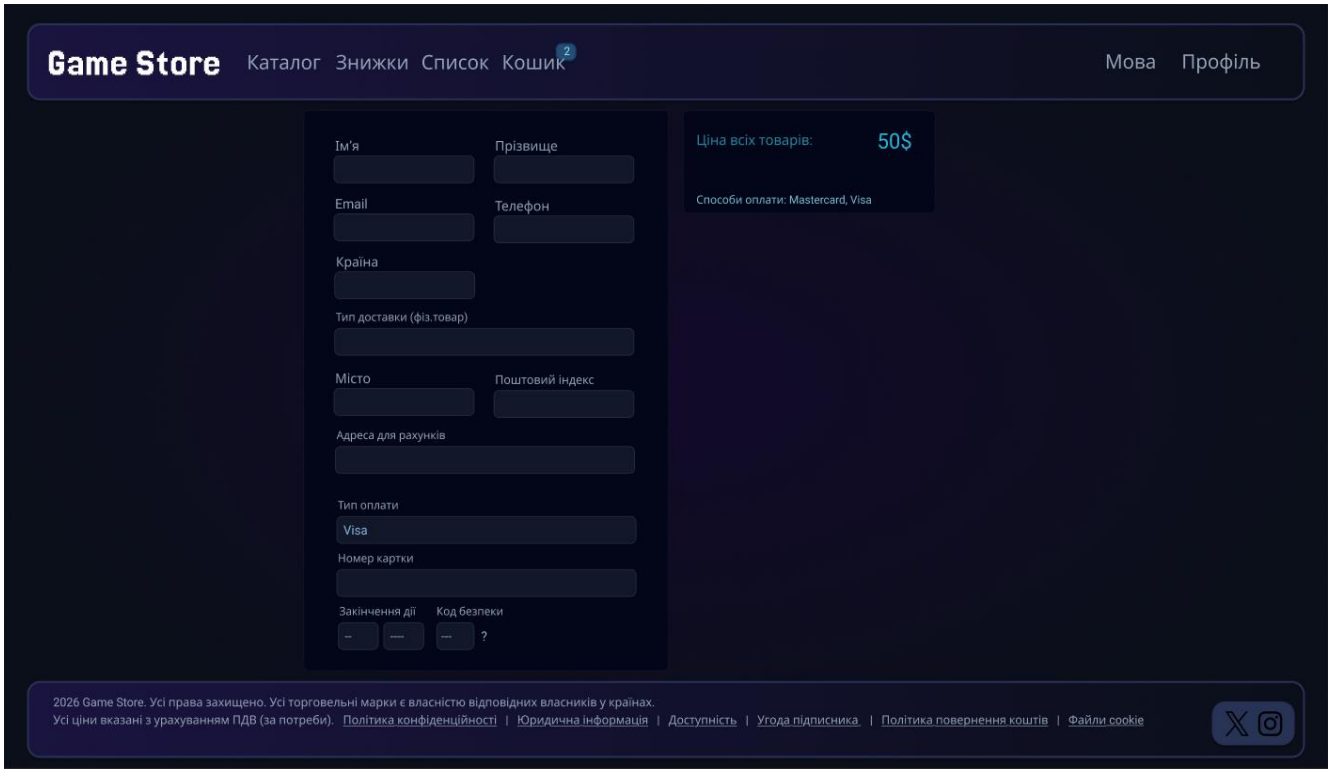

Рисунок 3.7 – Макет сторінки оформлення замовлення

Сама сторінка є простою в суті, дозволяє користувачу заповнити потрібні поля, котрі виділяються стилями .css, вказується ціна всіх товарів в кошику користувача та наявні способи оплати з можливістю зробити покупку за допомогою спеціальною кнопки, та при умові що всі дані були заповнені правильно.

Висновки до розділу 3

В даному розділі було проведено аналіз архітектури, моделювання та проєктування програмного забезпечення. Можливо визначити головні елементи при дальшій реалізації вебзастосунку та приблизний вигляд функціоналу та вигляду.

В архітектурі клієнтської частини використовується React, для роботи кошика та списку бажаного між клієнтом та сервером використовуються можливості Socket.io. Для реалізації принципу SPA використовуються можливості бібліотеки ‘React-router’, що надає можливість перемикати елементи в межі однієї сторінки. Для посилання даних між клієнтом та сервером використовується також принцип REST API, що надає клієнтові, через сервер, отримувати чи посилати дані на базу даних.

Для серверної частини використовується Express.js та можливості для додаткової захисту у створенні middleware для токенів. Для підключення до Microsoft SQL Server Management використовується бібліотеки ‘mssql’ та дані для підключення зберігаються в іншому файлі.

Серед моделюванні вебзастосунку виділено розроблено діаграму вебзастосунку та діаграми класів. Діаграма вебзастосунку описує складові та деякі елементи для взаємодії між ними. Діаграма класів описує класи, що наявні в вебзастосунку та відношення між ними.

Створені макети сторінок надають приблизний вигляд ідеї розташування елементів та вигляду сторінки. Макет сторінки списку товарів зображує список товарів та можливість до взаємодії з ними, приблизну інформацію та можливості до пошуку чи фільтрації товарів. Макет товару зображує більшу частину інформацію про товар користувачеві та надає можливість до більшого взаємодії з товаром. Макет сторінки кошика описує вигляд вмісту кошика, можливості до взаємодії з товарами, при наявності, та інформація, що подається користувачу. Макет сторінки оформлення замовлення описує потрібну інформацію для заповнення користувачем.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Реалізація програмного забезпечення

4.1.1 Реалізація клієнтської частини програмного забезпечення

Для створення клієнтської частини обрано фреймворк React, проєкт створюється за допомогою команди ‘npx create-react-app client’, що створить дуже базовий проєкт на даному фреймворку. Для реалізації маршрутизації встановлено бібліотеку ‘React-router’ використовуючи команду ‘npm install react-router-dom’. В головному файлі App.js всі частини обгорнуть в <Router> та в <Routes>, посилання на інші сторінки відбуваються за допомогою елементу <Route>, де в параметрі path вказується адреса та в elements вказується який елемент сторінки саме використати.

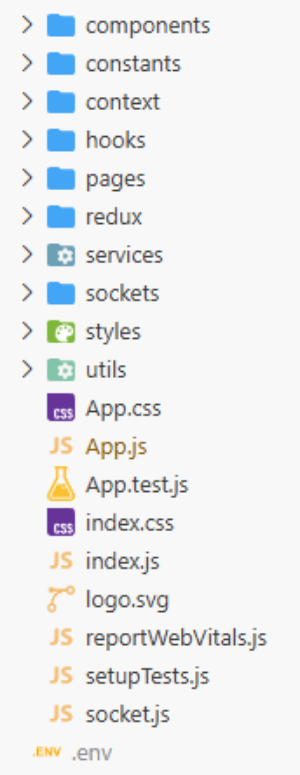

Рисунок 4.1 – Приклад структури клієнту на React та розділення на елементи

Всі елементи сторінок можливо розділити на різні файли: pages – вказує на зміст сторінки, components – окремі компоненти котрі використовуються декілька разів, constants – файли-константи що вказують стилі при певних параметрах, context – функціональна частина клієнту, hooks – спеціальні хуки, services – отримання даних з сервера шляхом переходу на спеціальне посилання, styles –

спеціальні види класів стилів, що використовуються декілька разів в різних елементах, utils – спеціальні утиліти винесені в окремий файл, наприклад для скорочення шляху до картинки чи нормалізація дати клієнта та бази даних.

Для постійного зв'язку між сервером та клієнтом використовується socket.io, для підключення використовується однойменна бібліотека на обох серверах. Socket.io потрібен для безперебійної роботи кошика та списку бажаного в реальному часі. Це дозволяє відразу отримувати дані про зміни у вмісті відповідних таблиць та реагувати на ці зміні на стороні клієнта не потребуючи оновлення сторінки.

```
let socketRef;
const cleanup = onSocketReady((socket) => {
  socketRef = socket;
  const handler = (rawUpdate) => {···
  };
  socket.on("cart:price-updated", handler);
  return () => socket.off("cart:price-updated", handler);
});
```

Рисунок 4.2 – Приклад відправки подій на сервер через клієнт

Для посилання звичайних HTTP-запитів на сервер використовуються можливості бібліотеки 'axios'. Функція 'interceptor', котрий автоматично виконується перед кожним запитом (request) та після кожної відповіді (response) надає можливість додавати токени у кожен запит, не записуючи його кожного разу. Для роботи використовуються функції get() для отримання даних чи post() для посилання даних. В функціях записується путь, по котрому будуть знаходитись дані, та які саме дані отримуються з сервера чи посилаються на сервер.

Бібліотека 'dayjs' використовується для роботи з датами на клієнті, особливо з часовими поясами та форматами дати. Дана бібліотека надає можливість для форматування дати в звичайну зрозуміли структуру 'DD.MM.YYYY HH:mm'. Також надає можливість для роботи з датами через використання спеціальних функцій по-типу add() чи subtract() що дозволяє змінювати значення в конкретному часу та на конкретне значення. Також за допомогою функціонал функцій toDate(), toString(), toJSON() надається можливість записувати дані як дату, рядок чи json-

файл. Функції isAfter() та isSame() використовуються для порівняння дат між собою, відповідно чи дата пізніша за інші або дати схожі.

Для валідацій форм використовуються можливості бібліотеки ‘Yup’, що надає можливість створити мінімальні потреби для рядків у вигляді функції .min(), що задає мінімальну кількість символів, та .email(), що перевіряє чи є рядок поштою, все це можливо зробити обов’язковим використовуючи функції .required(). Для всіх функцій ‘yup’ можливо задати повідомлення при невиконанні умов всередині функції. Для додаткової реалізації форм використовується бібліотека ‘react-hook-form’, що спрощує форматування коду для форм та надає можливість для додаткової валідації. При деяких потрібних діях, користувачу потрібно виводити деяку інформацію, для цієї задачі підходить бібліотека ‘react-toastify’, що надає можливість виводити маленькі повідомлення з короткою інформацією.

4.1.2 Реалізація серверної частини програмного забезпечення

Для створення серверної частини обрано Express.js, проєкт Node.js створюється за допомогою команди ‘npm init -y’, вже після створення даного проєкту на ньому створюється Express-проєкт за допомогою команди ‘npm install express cors’. Для повноцінного включення веб-сервера для інших серверів вказується порт значення та об’єкт веб-сервера слухає його через функцію listen().

```
const app = express();
const PORT = process.env.PORT || 5000;
server.listen(PORT, () => {
  log(`Server running with socket.io on port ${PORT}`);
});
```

Рисунок 4.3 – Повне підключення веб-сервера

В головному файлі потрібно вказати express(), що створює об’єкт веб-сервера. Після цього використання бібліотек потребує використання в об’єкті через use(). Для роботи з даними використовуються різні команди get(), use(), delete() та всередині задаються, якщо потрібно, повідомлення при роботі з даними.

Підключення до Microsoft SQL Server Management відбувається за допомогою бібліотеки ‘mssql’, де обов’язково потрібно вказати деякі параметри при підключенні: user, password, server, database. Після цього можливо підключитись до бази даних через функцію connect() та всередині вказавши правильні параметри. Для надсилання запитів до бази даних використовується функція query(), де всередині записується повний потрібний запит. Після отримання даних з запиту, можливо записати їх у json за допомогою функції Express.js json() та відправити на клієнт.

```
router.put(
  "/users/:id",
  authenticateToken,
  requirePermission("admin.users.edit"),
  adminController.updateUser
);
```

Рисунок 4.4 – Приклад надсилання даних на клієнт, де вказується путь до даних, додатковий middleware та функція, котра посилає дані

Відправка на клієнт відбувається за допомогою Express.router(). За допомогою даної функції можливо створювати окремі міні-маршрути, ділячи їх по файлах. Маршрути задаються через операції, що вказують тип дії, котра відбувається з інформацією, тобто посилається на читання, зміни параметрів чи інше. Всередині вказується путь, через який клієнт зможе отримати доступ до даних, що передаються, додаткову перевірку middleware, якщо реалізована, та функцію, яка сама отримала з бази даних, обробила та передає інформацію до клієнта.

Деяка інформація в базі даних потребує оновлення через деякий час, даний процес реалізується використовуючи бібліотеку ‘node-cron’, що дозволяє запускати задачі за заданим розкладом. Даний вид розкладу працює за принципом cron на операційній системі Linux. За розкладом можливо реалізувати перевірку на застарілі знижки, порівнюючи в базі даних дату в даний час та записану дату закінчення знижки, що дозволить порівняти та оновити за потрібності вміст деяких

рядків в базі даних. Для вказування розкладу використовуються п’ять позицій: minute (хвилина), hour (година), day of month (день місяця), month (місяць), day of week (день неділі). Якщо якийсь з елементів не потрібен, замість значення записується символ – ‘*’.

4.1.3 Реалізація бази даних програмного забезпечення

В базі даних зберігаються інформації про товари, користувачів, знижки та окремо, якщо це потрібно, характеристики. Більшість таблиць мають ‘primary key’, головний ключ, котрий надається унікальному ідентифікатору таблиці – ‘id’. Також даному ідентифікатору видається можливість автоінкрементації.

Для зв’язків рядків таблиць використовуються ключі ‘foreign key’. Даний ключ в таблиці вказується в рядку таблиці, та робить посилання на інший рядок вже іншої таблиці, наприклад таблиця ‘Games’ для рядка ‘AgeRatingId’ має ключ ‘FK_Games_AgeRatings’, що надає можливість даному рядку посилатися на рядок ‘Id’ в таблиці ‘AgeRatings’, зв’язуючи їх. Даний ключ не дозволить рядкам ‘AgeRatingId’ мати значення, котрого немає в таблиці ‘AgeRatings’.

Деякі дані при створенні нових рядків повинні мати відразу значення за замовчуванням. Для даної задачі використовуються default constraints, що задають значення за замовчуванням, в залежності від типу рядків. Наприклад в таблиці рядок ‘IsDeleted’ перевіряє, чи видалений товар, та має тип bit, для цієї задачі краще всього за замовчуванням всім товарам давати значення ‘0’, тобто false, бо всі товари за замовчуванням не видалені. Або існує рядок ‘Status’, що вказує тип реалізації товару в магазині. Всі нові створенні товари не відразу попадають в магазин, а знаходяться на етапі макету, що вказується в рядку ‘Status’ через значення ‘Draft’, тому за замовчуванням краще вказати всім новим продуктам значення ‘Draft’, котре пізніше потрібно буде замінити за потребою та виконання умов.

Для деяких випадків потрібно не допускати факт існування в одному рядку однакових значень пар полів. Для цієї задачі використовуються ‘Unique constraints’, котра дозволяє існувати тільки унікальним полям. В таблиці ‘Wishlist’ зберігаються рядки для ‘UserId’, що описує ідентифікатор користувача, котрий додав товар до свого списку бажаного, та ‘GameId’, що описує доданий товар до списку бажаного,

але один користувач не зможе додати такий же товар до свого списку бажаного ще раз, бо використовується unique constraint для пари даних полів в таблиці.

4.2 Тестування програмного забезпечення

Кожний новий користувач має пароль в базі даних, але той не зберігається з таким же видом, яким і записує та подає користувач. Пароль на сервері проходить обробку хешування за допомогою bcrypt, де додається значення salt, котрий є випадковим, та повторює даний алгоритм 2 в n-степені разів. Після хешування паролю важливо перевірити можливість повторного використання вже захешованого пароля та звичайного введеного пароля при спробі авторизації в акаунт.

Таблиця 7 – Тест-кейс реєстрації

№ T1.1: Реєстрація нового користувача на вебзастосунку за допомогою форми реєстрації та пошти користувача		
Мета: Перевірити функцію реєстрації незареєстрованого користувача через форму реєстрації у вебзастосунку		
Тип: Functional		
Приорітет: High	Час на виконання: 1 хв.	
Дата: 22.05.2026	Власник: Матвієнко О. П.	
Передумови: 1. Відкрита головна сторінка вебзастосунку 2. Користувач не зареєстрований у системі 3. Пошта користувача не використовується для іншого аккаунту у вебзастосунку		
Дії	**Очікуваний результат**	**Наявний результат**
1. Натиснути кнопку ‘Login’ у верхньому правому куті шапки	1. Відкрилась форма авторизації	Успішно
2. Натиснути ‘Register’	2. Відкрилась форма реєстрації	Успішно
3. Ввести пошту та пароль, виконуючи умови	3. Пошта та пароль були прийняті, на пошту відправлено повідомлення	Успішно
4. Відкрити повідомлення	4. Вивід повідомлення	Успішно

5. Натиснути ‘Далі’	5. Система підтвердила пошту, аккаунт був доданий	Успішно
Післяумови: Користувач успішно авторізований, кнопка ‘Login’ замінена на кнопку профіля, котра знаходиться у правому верхньому куті		
Результат: Успішно **Час виконання:** 1 хв **Дата виконання: 22.05.2026**		

При спробі увійти в аккаунт користувач повинен спочатку ввести правильну пошту, якщо такової пошти немає в бази даних – виводиться ‘User not found’, що сігналізує про відсутність такого користувача в базі даних вебзастосунка.

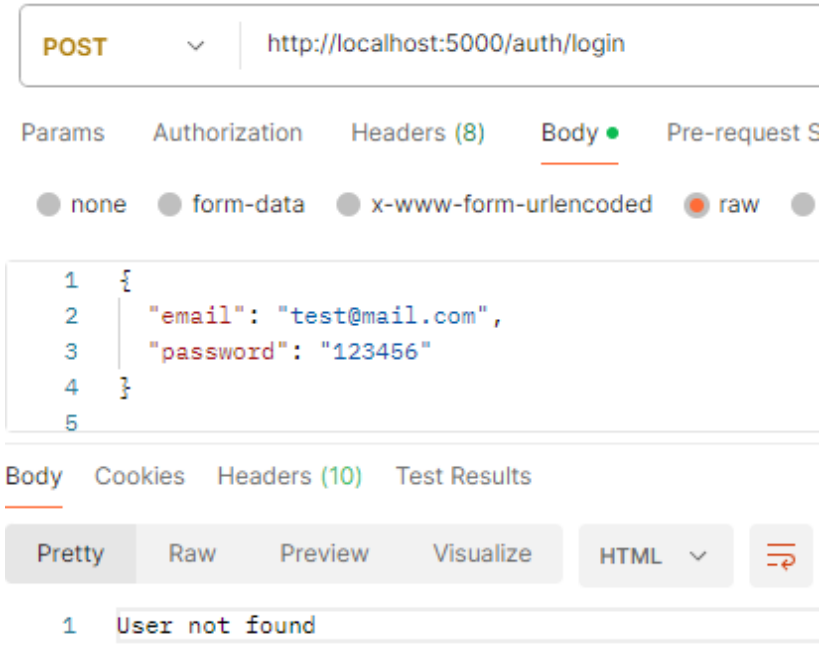

Рисунок 4.5 – Тестування авторизації з неправильною поштою

При введення правильної пошти вебзастосунок повинен перевірити пароль, порівняння відбувається за допомогою функції bcrypt.compare(). Значення salt та його кількість зберігається в самому хешованому рядку, тому при порівнянні з звичайним паролем, він пропускається через той самий salt стільки ж разів, що в результаті повинно видати той хешований варіант паролю.

Таблиця 8 – Тест-кейс кошика

№ Т1.2: Додавання товару до кошика користувача		
Мета: Перевірити функцію кошика користувача та оновлення вмісту шляхом додавання та видалення товару		
Тип: Functional		
Приорітет: High	Час на виконання: 1 хв.	
Дата: 22.05.2026	Власник: Матвієнко О. П.	
Передумови: 1. Відкрита головна сторінка вебзастосунку 2. Користувач зареєстрований у системі 3. Користувач увійшов в аккаунт успішно 4. Користувач немає обмеження на використання кошика 5. Товар не був доданий до кошика		
Дії	**Очікуваний результат**	**Наявний результат**
1. Натиснути кнопку ‘Cart’ у блоці потрібного товару	1. Кнопка замінилась на ‘In cart’, лічильник в шапці біля кошика змінив значення	Успішно
2. Знайти кнопку кошика користувача	2. Кнопка користувача змінила стиль	Успішно
3. Клікнути на кнопку кошика в шапці товару	3. Відкрилась сторінка кошика	Успішно
4. Перегляд вмісту кошика	4. Доданий товар відображається на сторінці кошика	Успішно
5.Видалення товару з кошика	5. Оновлений вміст кошика, оновлений лічильник на кнопці кошика в шапці	Успішно
6.Клікнути на кнопка товари в шапці	6. Відкрилась сторінка товарів магазину	Успішно
7.Перевірити кнопку товару	7. Товар знову можливо додати до кошика	Успішно
Післяумови: Користувач успішно додав та видалив товар у своєму кошику		
Результат: Успішно **Час виконання:** 1 хв **Дата виконання: 22.05.2026**		

Для отримання більшості даних через GET, вебзастосунок потребує вхід користувача, бо йде обмеження прав доступу для незареєстрованих гостей. Для авторизації користувача потрібно в адресі http://localhost:5000/auth/login з типом POST, та з правильно введеними email та password. Якщо авторизація успішна – виводиться токен, котрий використовується для сеансу користувача.

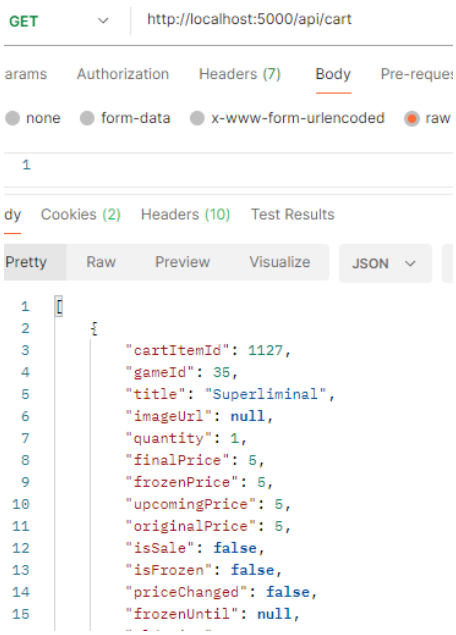

Рисунок 4.6 – Отримання даних в Postman для вмісту кошика користувача

При спробі використання GET та правильно введеної адреси, наприклад для отримання вмісту кошика користувача, на аккаунт котрого авторизувались, вводиться адреса http://localhost:5000/api/cart, що видає у стилі json всі дані. Для формування правильної адреси функції вебзастосунку розбиті на головні функції: авторизація – auth, кошик – cart, список бажаного – wishlist, знижкова компанія – campaign і т. д.. Для правильного запису потрібно спочатку брати головну функцію вебзастосунку потім з неї брати потрібну саму функцію.

Таблиця 9 – Тест-кейс додавання нового товару до вебзастосунку

№ Т1.3: Додавання та затвердження нового товару	
Мета: Перевірити функцію створення та додавання нового товару продавцем та затвердження товару адміністратором	
Тип: Functional	
Приорітет: High	Час на виконання: 6 хв.
Дата: 22.05.2026	Власник: Матвієнко О. П.

Кінець таблиці 9

Передумови: 1. Відкрита сторінка товарів продавця 2. Продавець увійшов в аккаунт 3. Аккаунт продавця має роль продавця 4. Аккаунт продавця немає обмеження на додавання нових товарів 5. Адміністратор увійшов в аккаунт 6. Адміністратор має роль адміністратора 7. Адміністратор немає обмеження на затвердження нових товарів до магазину 8. Адміністратор має батьківську роль продавця		
Дії	**Очікуваний результат**	**Наявний результат**
1. Натиснути кнопку ‘Додати товар’	1. Відкрилась форма додавання товару	Успішно
2. Заповнення всіх потрібних полів	2. Всі поля заповнені	Успішно
3. Натиснути на кнопку ‘Додати товар’	3. Товар був створений з типом ‘Draft’	Успішно
4. Натиснути на кнопку ‘Подати на модерацію’	4. Товар змінює статус на ‘Pending’	Успішно
5. Адміністратор відкриває панель модерації	5. З’являються посилання на сторінки модерації	Успішно
6. Адміністратор обирає ‘Вхідні товари’	6. Адміністратор переходить на сторінку вхідних товарів	Успішно
7. Адміністратор клікає по вхідному товару на кнопку ‘Змінити статус’	7. Відкривається панель з можливість обрати: ‘Approve’ чи ‘Dissaprove’	Успішно
8. Адміністратор обирає статус ‘Approve’ та натискає кнопку ‘Змінити’	8. Статус товару змінюється на ‘Approved’	
Післяумови: Товар успішно доданий до вебзастосунку та відображається на сторінках		
Результат: Успішно **Час виконання:** 6 хв **Дата виконання: 22.05.2026**		

4.3 Результати роботи програмного забезпечення

На шапці (header) клієнта вебзастосунку відображаються елементи для взаємодії для користувача, де можливо перейти по складовим магазина чи сторінки для знаходження нових товарів, також можливо відкрити кошик чи список бажаного або побачити кількість товарів в них, повідомлення користувача, календар майбутніх релізів, на котрі користувач підписався, та якщо користувач не увійшов в аккаунт – користувачу буде показувати кнопка ‘Login’ для авторизації, якщо увійшов аккаунт – то кнопка буде посилатися на профіль користувача та виводити ім’я аккаунту.

У нижній частині сторінки (footer) клієнта вебзастосунку знаходяться корисні посилання по-типу різні соціальні сторінки, деякі посилання з шапки, інформація про угоди підписника та інше.

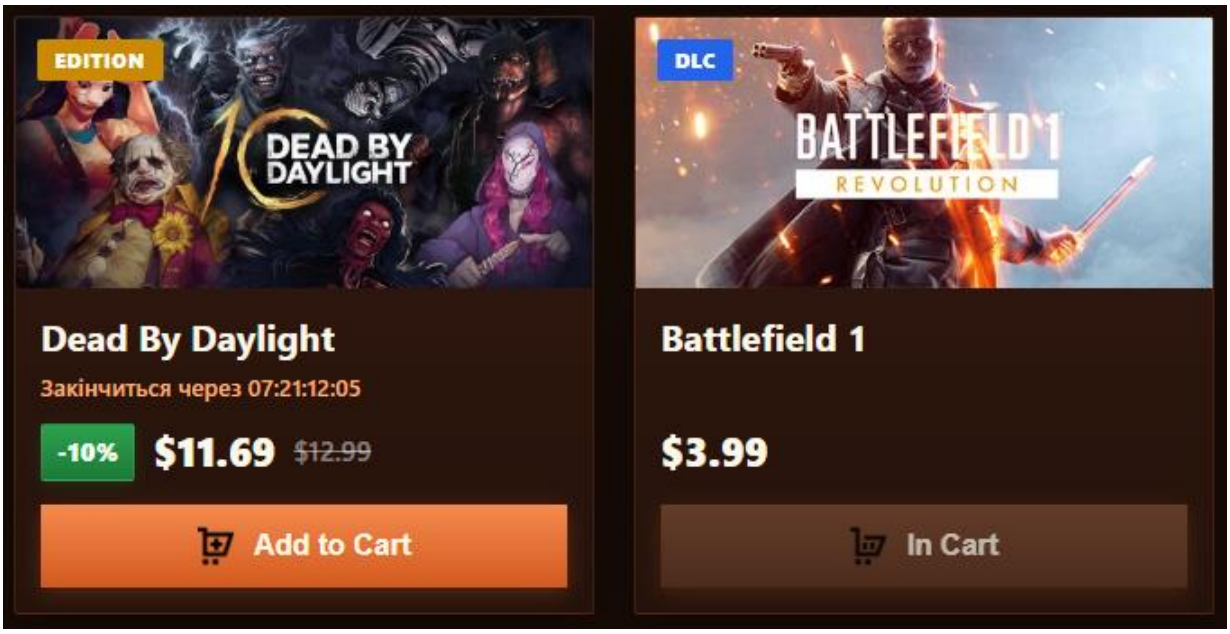

Рисунок 4.7 – Блок товару з короткою інформацією

На головній сторінці клієнта вебзастосунка відображаються різні типи товарів. У всіх товарів наявний банер-картинка, назва товару, тип товару та статуси. Якщо цей товар вийшов (released), то в нього наявна кнопка для додавання в кошик, котра змінює вміст, колір та відключає можливості бути натиснутою при додаванні товару в кошик, якщо знижки немає – то звичайний цінник, при

наявності знижки стара ціна перекреслюється, заміняється новою ціною в результаті знижки та відображається процент знижки, вище ціни додається таймер котрий вказує через скільки закінчиться знижка. Якщо товар має статус ‘preorder’ – то йде відображення ціни, таймер коли відбудиться вихід товару та можливість додати товар до списку бажаного, майже так само і в статусі ‘announced’, але ціна не відображається.

Рисунок 4.8 – Товар у списку бажаного

Коли товар додається в кошик або в список бажаного то він записується в базу даних та оновлюється лічильник у відповідному посиланні через socket. Вміст кошика та списку бажаного схожий, але де-як відрізняється: у кошика присутня можливість побачити суму ціну товарів та перейти до формування замовлення, в списку бажаного наявна кнопка додавання в кошик. Після переходу до оформлення замовлення користувач повинен заповнити всі потрібні поля та обрати спосіб оплати. Якщо у користувача присутній фізичний товар, то додатково потрібно заповнити поля для доставки, щоб перевірити чи можливо доставити фізичний товар користувачеві. Після надсилання даних, ті зберігаються в базі даних зі статусом замовлення для фізичного товару. Перевірити статус замовлення користувач зможе у себе в профілі.

4.4 Керівництво користувача

Вебзастосунок складається з трьох складових: React-клієнт, Node-сервер на Express.js та база даних в Microsoft SQL Server Management. Для встановлення

React-клієнт та Node-сервера на Express.js потрібно завантажити Node.js, інсталятор котрого знаходиться на офіційному вебсайті Node: https://nodejs.org/uk. В проєкті використовується Node версії 24.11.1.

Для створення клієнтського проєкту React необхідно виконати команду ‘npx create-react-app client’. Після цього для запуску проєкту потрібно перейти в папку проєкту та запускати сервер за допомогою команди ‘npm start’. В вебзастосунку використовується React версії 19.2.4 та бібліотеки, котрі підходять під дану версію фреймворка.

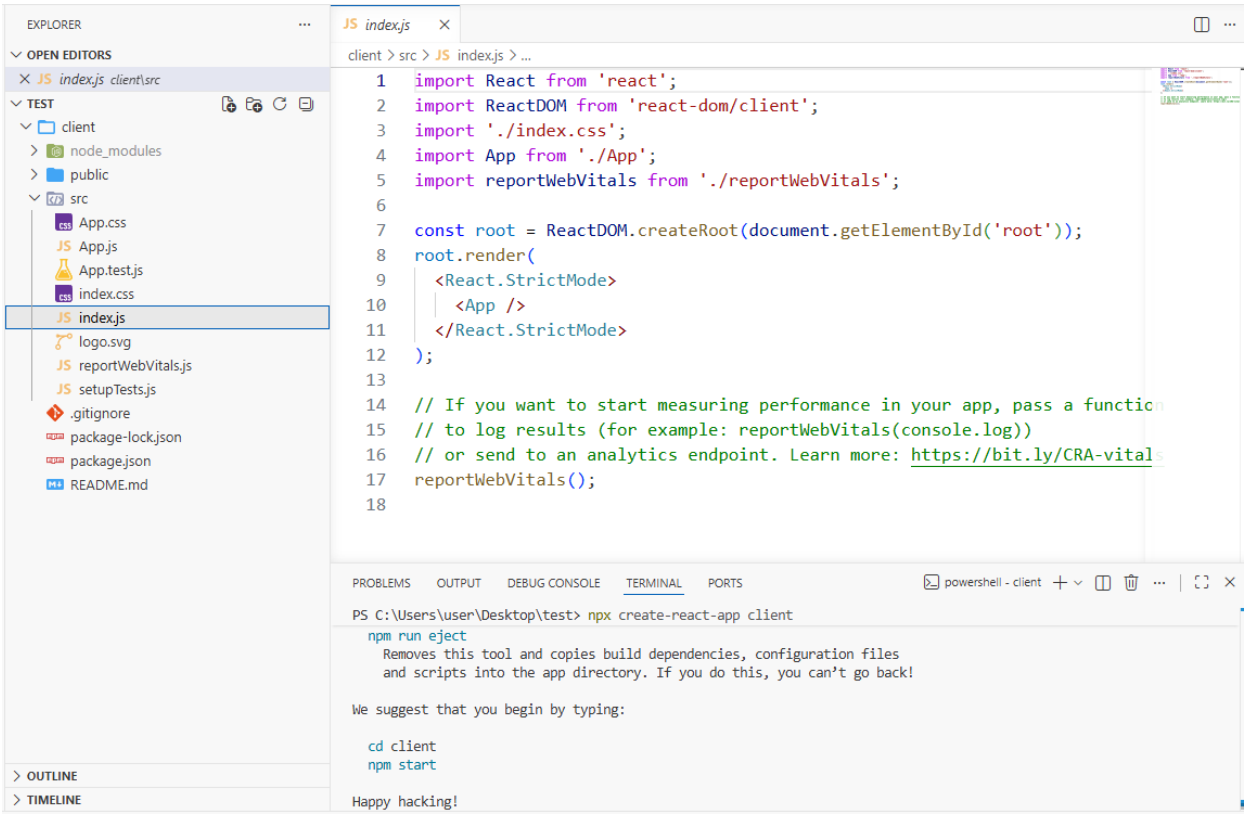

Рисунок 4.9 – Результат виконання команди ‘npx create-react-app client’

Для маршрутизації на клієнті використовується бібліотека react-router версією 7.13.2, що встановлюється командою ‘npm install react-router-dom’. Для безперервного з’єднання з сервером використовується socket.io версії 4.8.3, котрий встановлюється командою ‘npm install socket.io-client’. Для посилання HTTP-запитів на сервер використовується бібліотека axios версії 1.14 та встановлюється командою ‘npm install axios’. Для роботи з часом та датами використовується бібліотека dayjs версії 1.11.20, що встановлюється командою ‘npm install dayjs’. Для роботи з валідацією даних використовується бібліотека yup версії 1.7.1, що

встановлюється командою ‘npm install yup’. Для показу складності пароля використовується бібліотека react password strength bar версії 0.4.1 та встановлюється за командою ‘npm install react-password-strength-bar’, для оцінки складності пароля використовується бібліотека zxcvbn версії 4.4.2, що встановлюється командою ‘npm install zxcvbn’. Бібліотека react-redux версії 9.2.0 надає можливість працювати з глобальними станами в клієнті та встановлюється командою ‘npm install react-redux’. Для додавання повідомлень використовується бібліотека react-toastify версії 11.0.5 та встановлюється командою ‘npm install react-toastify’. Для додавання таймерів чи звичайного зворотнього відліку в проєкті використовується бібліотека react-countdown версії 2.3.6 та котра встановлюється командою ‘npm install react-countdown’.

```
"dependencies": {
  "axios": "^1.14.0",
  "dayjs": "^1.11.20",
  "dompurify": "^3.4.5",
  "react": "^19.2.4",
  "react-countdown": "^2.3.6",
  "react-dom": "^19.2.4",
  "react-hook-form": "^7.72.0",
  "react-password-strength-bar": "^0.4.1",
  "react-redux": "^9.2.0",
  "react-router-dom": "^7.13.2",
  "react-scripts": "5.0.1",
  "react-toastify": "^11.0.5",
  "redux-persist": "^6.0.0",
  "socket.io": "^4.8.3",
  "socket.io-client": "^4.8.3",
  "web-vitals": "^2.1.4",
  "yup": "^1.7.1",
  "zxcvbn": "^4.4.2"
},
```

Рисунок 4.10 – Список всіх додаткових встановлених бібліотек в клієнті та записаних в package.json

Для створення Node-сервера на Express.js потрібно встановити Node.js. Після встановлення на систему, для створення звичайного node-сервера використовується команда: ‘npm init -y’. Для додавання Express.js до проєкту

використовується команда ‘npm install express’. Запуск node-сервера відбувається за допомогою команди ‘node app.js’. У проєкті використовується Express.js версією 5.2.1 та бібліотеки, котрі підходять під дану версію Node.

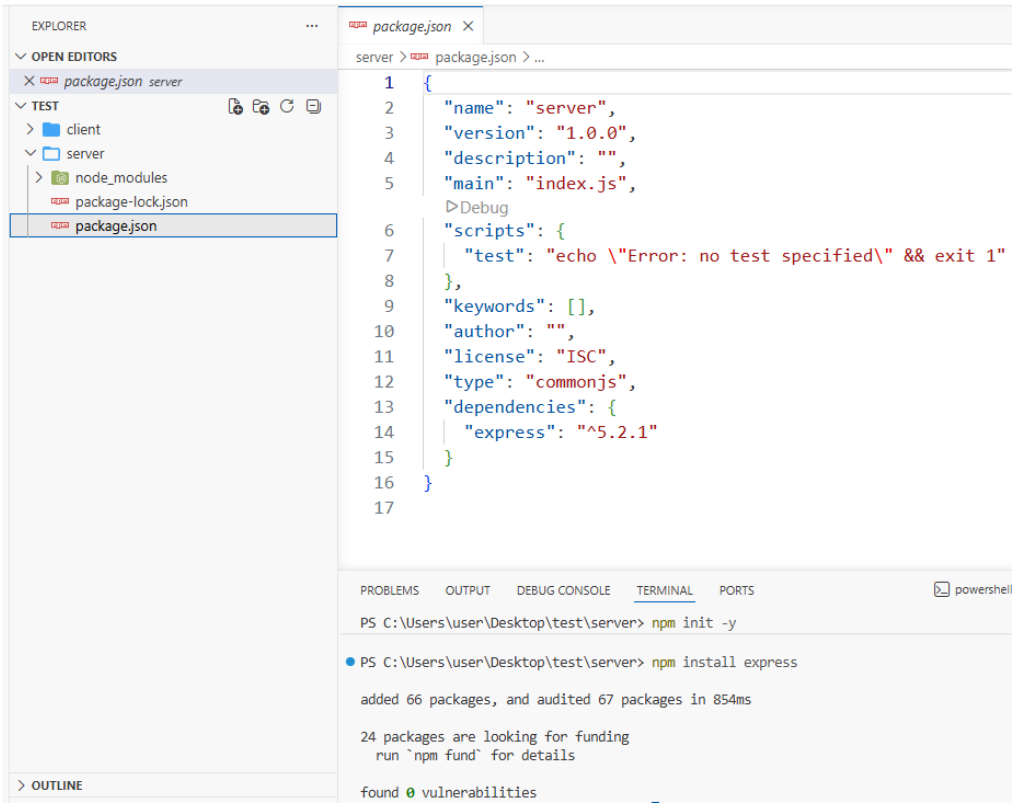

Рисунок 4.11 – Результат виконання команди ‘npx create-react-app client’

Для отримання доступу до бази даних використовується бібліотека mssql версією 12.2.1, котра встановлюється командою ‘npm install mssql’. Для безперервного з’єднання з клієнтом використовується socket.io версії 4.8.3, котрий встановлюється командою ‘npm install socket.io’. Для посилання HTTP-запитів на клієнт використовується бібліотека axios версії 1.14 та встановлюється командою ‘npm install axios’. Для хешування паролів та подальшої їх перевірки використовується бібліотека bcrypt версії 6.0.0, котра встановлюється командою ‘npm install bcrypt’. Для створення та опрацювання web-токенів використовуються можливості бібліотеки jsonwebtoken версії 9.0.3, що встановлюється командою ‘npm install jsonwebtoken’. Для роботи Express.js з cookies використовується бібліотека cookie-parser версії 1.4.7, котра встановлюється командою ‘npm install cookie-parser’. Для одночасної роботи сервера та клієнта на різних портах використовується бібліотека CORS версії 2.8.6 та встановлюється командою ‘npm

install cors’. Для посилання повідомлень на пошту користувачам використовується бібліотека nodemailer версії 8.0.4 та встановлюється через команду ‘npm install nodemailer’ та для покращення візуалізації повідомлень в пошті використовується бібліотека MJML версії 4.18.0 та встановлюється через команду ‘npm install mjml’. В сервері діє планувальник задач за розкладом, котрий працює за допомогою бібліотеки node-cron версії 4.2.1 та встановлюється через команду ‘npm install node-cron’.

```
"dependencies": {
  "axios": "^1.14.0",
  "bcrypt": "^6.0.0",
  "cheerio": "^1.2.0",
  "cookie-parser": "^1.4.7",
  "cors": "^2.8.6",
  "dotenv": "^17.3.1",
  "express": "^5.2.1",
  "handlebars": "^4.7.9",
  "howlongtobeat": "^1.8.0",
  "jsonwebtoken": "^9.0.3",
  "mjml": "^4.18.0",
  "mssql": "^12.2.1",
  "node-cron": "^4.2.1",
  "nodemailer": "^8.0.4",
  "socket.io": "^4.8.3",
  "socket.io-client": "^4.8.3"
}
```

Рисунок 4.12 – Список всіх додаткових встановлених бібліотек на сервері та записаних в package.json

База даних працює через програму Microsoft SQL Server Management. Всі бази даних, таблиці, рядки та дані зберігаються на sql-сервері. Сервер отримує дані через запити до підключеної бази даних та якщо у користувача бази даних є права доступу до перегляду та передачі даних. Більшість таблиці, мають рядок для особливої ідентифікації вмісту, підключено автоінкремент та primary key. Якщо таблиця зв’язує між собою дві інші – то вона не має особливої ідентифікації. В

деяких бази даних присутні додаткові ключі, що дають можливість посилатися одному рядку таблиці на інший, або робити правила унікальності даних.

Висновки до розділу 4

В розділі описано програмну реалізацію вебзастосунку та його елементів: клієнт-сервер на React, node-сервер використовуючи бібліотеку Express.js та база даних в програмі Microsoft SQL Server Management. В React використовують додаткові бібліотеки для посилання http-запитів на сервер або встановлення безперервного з'єднання в реальному часі. Також використовуються додаткові бібліотеки для покращення функціональності клієнта, наприклад для роботи з датами та годинами, валідацією рядків та форм, маршрутизації для реалізації принципів SPA, та додаткові повідомлення користувачеві.

На node-сервері з використанням Express.js використовується можливості бібліотеки mssql для підключення до бази даних та надсилання запитів та отримання, з подальшою обробкою, даних. Для можливості роботи і сервера і клієнта на одній системі використовується додаткові бібліотеки. Для взаємодії з поштою користувача використовуються додаткові бібліотеки і для покращення вигляду повідомлень. Для можливості додавання токенів та роботи з ними використовується jsonwebtoken, що дає можливість додавати сеанси для користувача та додатковий шар безпеки.

Описано способи тестування та їх результати. Тестування вийшло успішним та програмне забезпечення видало очікуваний результат. В результаті роботи описано очікувані результати при виклику деяких умов та як саме дійти до них. Зміни при виклику функцій та їх результати дій, оновлення даних та зміни стилей у елементів клієнта.

В керівництві користувача описані всі бібліотеки до кожного елементу проєкту, що вони роблять та поточні версії в проєкті. Також описано як саме встановити та завантажити основні частини для проєкту, як от Node.js, чи додати бібліотеку Express.js.

ВИСНОВКИ

Створене програмне рішення можливо використати для повноцінної реалізації мети у вигляді продажу та покупки цифрового товару у вигляді відеоігор. Даний вебзастосунок виконує всі потрібні дії та функції для інтернет-магазину. Зберігання акаунтів, інформація про користувача та про товари відбувається у базі даних. Фінальний результат схожий за макетами при проєктуванні, але дещо відрізняється в оформленні сторінок та реалізації деяких функцій та ідей.

Порівнюючи з існуючими аналогами, багато стильових та функціональних ідей було запозичено, але також були принесені додаткові ідеї – продаж, разом з цифровими товарами, фізичних товарів. Це розширює круг можливих покупців, що надає перевагу серед інших аналогів. Також при проєктуванні вебзастосунку були виделені і недоліки у аналогів, які потрібно було уникнути при реалізації, наприклад перевантажений інтерфейс чи навпаки – дуже мінімалістичний, показ не всієї інформації чи обмеження можливостей покупців в магазині.

Для повного впровадження в оборот в даному вебзастосунку потрібно реалізувати повноцінну оплату товарів та взаємодію з платіжними системами. Також для функціювання деяких елементів, наприклад вебзастосунку потрібні фізичні сервера чи для зберігання фізичних копій товарів – локація.

При створенні проєкту були поставленні такі завдання: провести аналіз сучасних вебзастосунків для продажу відеоігор, дослідити сучасні технології та засоби розробки вебзастосунків, зробити проєктування та моделювання вебзастосунку, розробити структуру базу даних для зберігання інформації про користувачів, товару та замовлення, реалізувати функціональні можливості для додавання, створення та видалення зареєстрованих користувачів, реалізувати функціональні можливості для створення каталогу товарів, систему кошика та взаємодії з нею та оформлення замовлення зареєстрованих користувачів, реалізувати та забезпечити адаптивний інтерфейс сторінок вебзастосунку для користування користувачем та зробити тестування функціоналу вебзастосунка. Всі поставлені задачі були повністю виконані.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Swacha J., Kulpa A. Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks. 2023. URL: https://doi.org/10.3390/electronics12173563 (Accessed: 29.04.2026).

2. Most Popular Web Frameworks among Developers Worldwide: website 2022. URL: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/ (Accessed: 29.04.2026).

3. Khan M. Web application development using Angular and Node.js. 2022. 56 p. URL: https://www.theseus.fi/handle/10024/490108 (Accessed: 27.04.2026).

4. Richards M., Ford N. Fundamentals of Software Architecture. An Engineering Approach. O'Reilly Media, 2025. 546 p.

5. Herron D. Node.js Web Development: Server-side web development made easy with Node 14 using practical examples. Packt Publishing Ltd. fifth edition. 2020. 760 p.

6. De Sanctis V. Building WEB APIs with ASP.NET Core. Packt Publishing Ltd. 2023. 472 p.

7. Vilas Salunke S., Ouda A. A Performance Benchmark for the PostgreSQL and MySQL Databases. 2024. URL: https://doi.org/10.3390/fi16100382 (Accessed: 29.04.2026).

8. React Reference Overview: website. 2026. URL: https://react.dev/reference/react (Accessed: 22.05.2026).

9. Node.js v26.2.0 documentation: website. 2026. URL: https://nodejs.org/docs/latest/api/ (Accessed: 22.05.2026).

10. Routing. Express.js: website. 2026. URL: https://expressjs.com/en/5x/guide/routing/ (Accessed: 22.05.2026).

11. Components and features in SQL Server Management Studio: website. 2026. URL: https://learn.microsoft.com/uk-ua/ssms/components-features (Accessed: 22.05.2026).

12. Introduction. Socket.io: website. 2026. URL: https://socket.io/docs/v4/ (Accessed 22.05.2026).

13. Axios: website. 2026. URL: https://www.npmjs.com/package/axios (Accessed: 22.05.2026).

14. React Router Home: website. 2026. URL: https://reactrouter.com/home (Accessed: 22.05.2026).

15. Cron for Node.js: website. 2026. URL: https://www.npmjs.com/package/cron (Accessed: 22.05.2026).

16. Microsoft SQL Server client for Node.js: website. 2026. URL: https://www.npmjs.com/package/mssql (Accessed: 22.05.2026).

17. CORS: website. 2026. URL: https://www.npmjs.com/package/cors (Accessed: 22.05.2026).

18. Jsonwebtoken: website. 2026. URL: https://www.npmjs.com/package/jsonwebtoken (Accessed: 22.05.2026).

19. Nodemailer: website. 2026. URL: https://nodemailer.com/ (Accessed: 22.05.2026).

20. Yup: website. 2026. URL: https://www.npmjs.com/package/yup (Accessed: 22.05.2026).

ДОДАТОК А

Діаграми

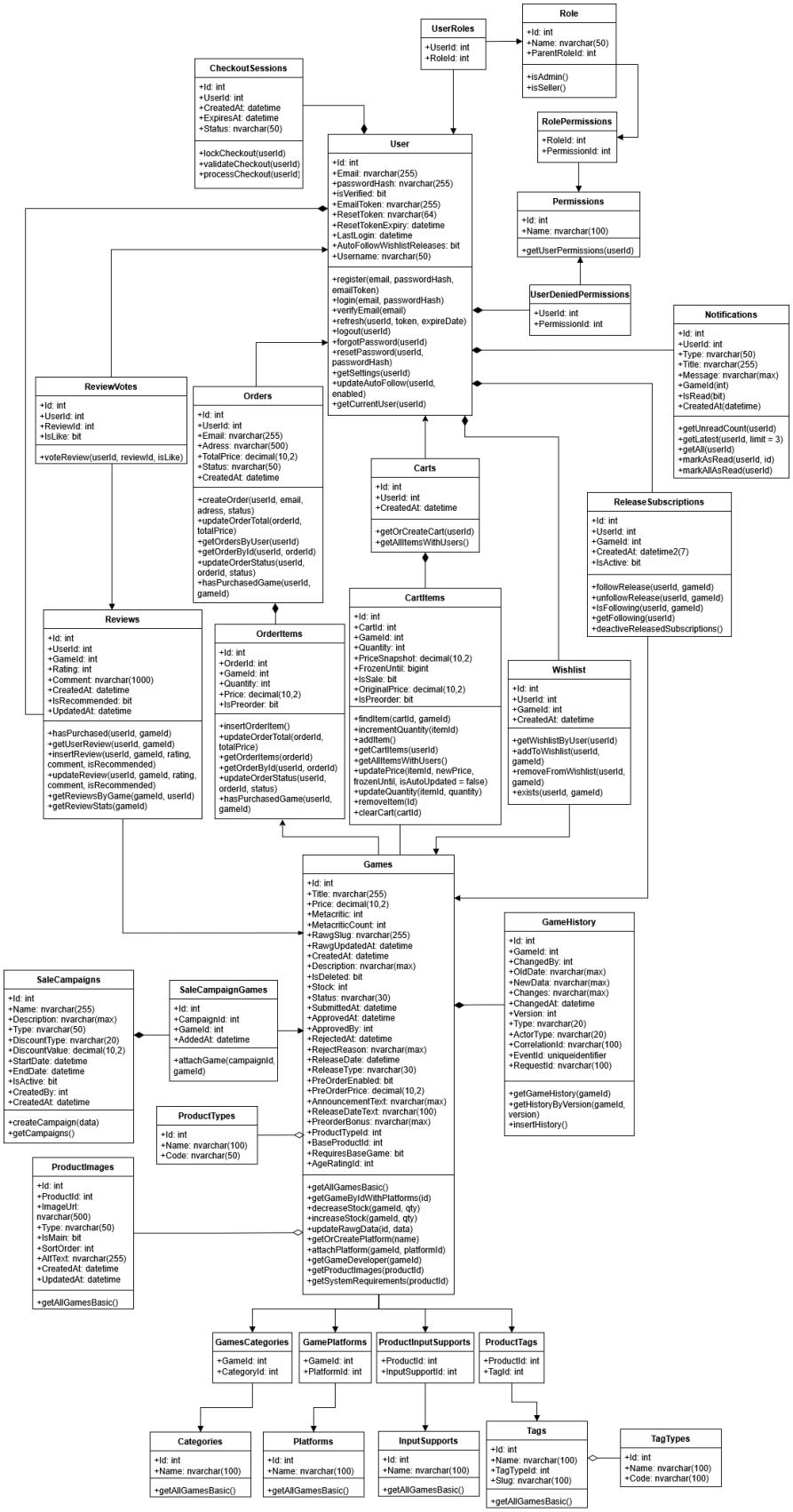

Рисунок А.1 – Діаграма класів

Рисунок А.2 – Діаграма взаємодії