

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Веб-додаток магазину відеоігор та порталу ігрових
публікацій «Gigashop»**

Виконав студент групи ІПЗ-19-2
ОЛЕКСІЙЧУК Микита Валерійович

Керівник роботи:
ЧИЖМОТРИЯ Олена Геннадіївна

Рецензент:
КРАВЧЕНКО Світлана Миколаївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»
В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: ОЛЕКСІЙЧУК Микита Валерійович

Керівник роботи: ЧИЖМОТРЕ Олена Геннадіївна

Тема роботи: «Веб-додаток магазину відеоігор та порталу ігрових публікацій «Gigashop»

Затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с

Термін здачі студентом закінченої роботи «14» червня 2023 р.

Вихідні дані для роботи (зазначається предмет і об'єкт дослідження):
Методи та засоби розробки веб-додатку за визначеними предметними областями. Моделі, технології, структури, засоби та методи розробки та взаємодії компонентів веб-додатку. Використані необхідні веб-орієнтовані технології.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	16.02.2023р.	16.02.2023р.
2	Чижмотря О.Г.	16.02.2023р.	16.02.2023р.
3	Чижмотря О.Г.	16.02.2023р.	16.02.2023р.

Керівник _____ Олена ЧИЖМОТРЕ

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	16 лютого 2023р – 1 березня 2023р	Виконано
2	Пошук, огляд та аналіз аналогічних розробок	16 лютого 2023р – 1 березня 2023р	Виконано
3	Формулювання технічного завдання	16 лютого 2023р – 1 березня 2023р	Виконано
4	Опрацювання літературних джерел	16 лютого 2023р – 1 березня 2023р	Виконано
5	Моделювання та аналіз програмного забезпечення	20 лютого 2023р – 10 березня 2023р	Виконано
6	Написання програмного коду	24 лютого 2023р – 12 червня 2023р	Виконано
7	Тестування роботи програми	5 квітня 2023р – 10 червня 2023р	Виконано
8	Аналіз одержаних результатів	11 червня 2023р – 12 червня 2023р	Виконано
9	Оформлення пояснювальної записки	13 червня 2023р – 14 червня 2023р	Виконано
10	Попередній захист	26 квітня 2023р	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу веб-додатку магазину відеоігор та порталу ігрових публікацій та пояснювальної записки. Пояснювальна записка до випускної роботи містить текстову частину, викладену на 111 сторінках друкованого тексту, має 132 сторінки додатків. Список використаних джерел містить 21 найменування і займає 3 сторінки. В роботі наведено 115 рисунків та 9 таблиць. Загальний обсяг роботи – 243 сторінки.

Метою роботи є розробка веб-додатку магазину відеоігор з порталом ігрових новин засобами HTML, CSS, TypeScript та технологій на основі засобів, а також налаштування розгортання веб-додатку засобами технології Docker.

У роботі визначено функціональні, нефункціональні та бізнес-вимоги. Також було проаналізовано аналоги програмних продуктів та обрано технології розробки та обґрунтовано інструментальні засоби, що були використані в розробці, обрано архітектуру веб-додатку та його компонентів, проведено аналіз та моделювання програмного забезпечення, проектування системи та описано програмну реалізацію продукту.

КЛЮЧОВІ СЛОВА: БАЗА ДАНИХ, ВЕБ-ДОДАТОК, ВЕБ-САЙТ, ВІДЕОІГРИ, ПУБЛІКАЦІЇ.

					ІПЗ.КР.Б – 121 – 23 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Пояснювальна записка «Веб-додаток магазину відеоігор та порталу ігрових публікацій "Gigashop"»	Літ.	Арк.	Аркушів
Розроб.		М. Олексійчук						
Керівник		О. Чижмотря					4	243
Рецензент		С. Кравченко				Житомирська політехніка група ІПЗ-19-2		
Зав. каф.		А. Морозов						

ABSTRACT

The bachelor's graduate work consists of software package from the web application for video games store and gaming publication portal and documentation. The documentation of the bachelor's work contains a text part, set out on 111 pages of printed text, has 132 pages of appendices. The list of used sources contains 21 names and occupies 3 pages. The paper presents 115 illustrations and 9 tables. The total volume of the work is 243 pages.

The aim of the work is to develop a video games store web application with a gaming news portal using HTML, CSS, TypeScript and asset-based technologies, as well as configuring the deployment of the web application using Docker technology.

The paper identifies functional, non-functional and business requirements. Analogues of software products were also analysed and development technologies were selected, the tools used in the development were substantiated, the architecture of the web application and its components was chosen, software analysis and modelling were carried out, system design and software implementation of the product is described.

KEYWORDS: DATABASE, PUBLICATIONS, VIDEO GAMES, WEB APPLICATION, WEBSITE.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ІІЗ	Арк.
		Чижмоторя О.Г.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	7
ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ВЕБ-ДОДАТКУ МАГАЗИНУ ВІДЕОІГОР ТА ПОРТАЛУ ІГРОВИХ ПУБЛІКАЦІЙ	11
1.1. Постановка задачі	11
1.2. Аналіз аналогів програмного продукту	13
1.3. Вибір архітектури програмного продукту	18
1.4. Обґрунтування вибору інструментальних засобів	19
1.5. Аналіз вимог до програмного продукту	24
Висновки до першого розділу	27
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ, МОДЕЛЮВАННЯ, АНАЛІЗ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	29
2.1. Розробка моделі програмного забезпечення на логічному рівні	29
2.2. Фізична модель та прототип програмного забезпечення	38
2.3. Реалізація серверної частини веб-додатку	44
2.4. Реалізація клієнтської частини веб-додатку	53
Висновки до другого розділу	69
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ДОДАТКОМ МАГАЗИНУ ВІДЕОІГОР ТА ПОРТАЛУ ІГРОВИХ ПУБЛІКАЦІЙ	70
3.1. Порядок встановлення та налаштування параметрів веб-додатку	70
3.2. Структура інтерфейсу та порядок роботи з веб-додатком магазину відеоігор	74
3.3. Тестування роботи програмного забезпечення	83
Висновки до третього розділу	105
ВИСНОВКИ	107
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	109
ДОДАТКИ	112

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – База даних.

ПЗ – Програмне забезпечення.

ORM – Object-Relational Mapping (Об'єктно-реляційна проекція).

UI – User Interface (Інтерфейс користувача).

SPA – Single-Page Application (Односторінковий застосунок).

СКБД – Система керування базами даних.

CRUD – Create, read, update, delete (Створити, читати, оновити, видалити).

JWT – JSON Web Token (JSON веб токен).

CORS – Cross-Origin Resource Sharing (Спільне використання ресурсів з різних джерел).

NPM – Node Package Manager (Менеджер пакунків Node).

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Розвиток інформаційних технологій змінив буденне життя людей, спростивши доступ до інформації завдяки технології Інтернет та створивши окрему сферу розваг у вигляді відеоігор. Розповсюдження відеоігор відбувається як у фізичному вигляді в магазинах відеоігор та/або електронної техніки, так і в електронному через отримання ліцензійних ключів до продуктів після придбання, а разом з розповсюдженням створюється спільнота гравців та фанатів певних серій чи окремих продуктів.

Через це для будь-якого підприємця, що планує бути успішним на ринку відеоігор, створення веб-сайту з продажу даних товарів є обов'язковою умовою планування бізнесу. Сфера електронних розваг тісно пов'язана з веб-технологіями через походження обидвох сфер інформаційних технологій, тож використання веб-сайтів для більшості осіб, зацікавлених у відеоіграх, є звичайною справою.

Велика кількість компаній, які підтримують свої веб-сайти, є направленими на всесвітню спільноту, де основною мовою спілкування є англійська. Через це перед локальними новими користувачами постає проблема мовного бар'єру, яку вирішують через навчання мові для повного та зручного користування сервісами. З цього витікає потреба в локальних сервісах розповсюдження новин, досвіду, думок та оглядів на продукти, що засновані на офіційній мові своєї країни.

З описаної потреби в комбінації з розповсюдженням використання веб-технологій впливає логічний запит локальних користувачів на сервіс, який дозволить спілкуватись та взаємодіяти між собою в рідному мовному середовищі та отримувати ліцензійні легальні копії товарів за доступними регіональними цінами в доступній валюті. Останній фактор також впливає на створення та поширення технологій з керування фінансами та/або оплати товарів у певній країні.

Мета даної роботи полягає в розробці веб-додатку магазину відеоігор та ігрових публікацій, доступний для українських користувачів, досвідчених

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

гравців та новачків у цій сфері, на мові програмування TypeScript із використанням технологій та відкритих розробок на базі даної мови та мови JavaScript, технологій Node.js та React, СКБД PostgreSQL. ПЗ має забезпечувати функціонал для безперебійної роботи веб-сайту та взаємодіяти з комп'ютерними та мобільними платформами. Також метою є поглиблення теоретичних та практичних знань, покращення вмінь та навичок розробки веб-додатків.

Поставлена мета обумовлює необхідність виконання наступних завдань:

- аналіз теоретичних та функціональних потреб проектування та реалізації веб-сайту з продажу відеоігор та створення ігрових публікацій;
- аналіз існуючих веб-сайтів за даною тематикою та визначення унікальності, необхідних функціональних можливостей;
- визначення технологій розробки ПЗ;
- визначення інформаційних потреб предметної області дослідження;
- обрання оптимальної на даний час архітектури функціонування веб-додатку;
- виконання опису способів та методів взаємодії зі сервісом;
- проектування бази даних для збереження інформації сервісу та користувачів;
- розробка адаптивного інтерфейсу веб-сайту;
- розробка алгоритмічної моделі функціонування веб-додатку на основі БД;
- написання серверної частини.

Об'єктом дослідження є методи та засоби розробки веб-сайту за визначеними предметними областями.

Предметом дослідження є моделі, технології, засоби та методи розробки та взаємодії компонентів веб-додатку, а також використані необхідні технології та платформи, програмні засоби.

В роботі було використано методи об'єктно-орієнтованого програмування, веб-розробки та дизайну користувацьких інтерфейсів.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

Розроблена програма може бути оптимізована та дорацьована в додатковому функціоналі, адаптована як практичний веб-сайт фізичного магазину або бути окремим сервісом в мережі Інтернет.

Пояснювальна записка до випускної роботи містить 111 сторінок, 115 ілюстрацій та 9 таблиць., список літературних джерел містить 21 найменування.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ВЕБ-ДОДАТКУ МАГАЗИНУ ВІДЕОІГОР ТА ПОРТАЛУ ІГРОВИХ ПУБЛІКАЦІЙ

1.1. Постановка задачі

Основною метою створення веб-додатку магазину відеоігор та публікацій ігрових новин є розвиток української частини сфери бізнесу електронних розваг та популяризація придбання перевірених ліцензій від офіційних видавців та розробників. Розробка даного програмного продукту дозволить спільноті гравців України мати власну платформу для створення публікацій про сферу відеоігор та обговорень новин спільноти, сфери замість необхідності використання закордонних ресурсів, на заваді чого можуть стояти проблеми мовного бар'єру.

У випускній роботі необхідно розробити веб-додаток магазину з можливістю купівлі товарів, перегляду товарів по жанрам та компаніям, що розробили чи видали товар, та написання публікацій стосовно сфери ігрового бізнесу. Для реалізації поставленого завдання основними етапами є:

1. Виконати моделювання та аналіз вимог до програмного забезпечення:
 - 1.1. Розробити структуру класів даних системи та майбутню структуру БД.
 - 1.2. Змоделювати логіку архітектури та її компонентів, розгортання тощо.
2. Розробити backend-логіку роботи веб-додатку:
 - 2.1. Згідно результату аналізу та моделювання компонентів, розробити CRUD-функціонал роботи з таблицями бази даних.
 - 2.2. Розробити програмні модулі виконання наступного функціоналу в залежності від даних та дій користувачів:
 - автентифікація дій користувачів: купівля, додавання до товарів бажаних/обраних, коментування та створення

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

публікацій автентифікується для швидкого доступу до власних даних та перегляду записів інших користувачів;

- авторизація користувачів: через використання системи ролей, виконується розділення доступу до функціоналу (створення записів даних та надання прав до цього) по ролям: користувач, модератор, адміністратор;
- модуль перегляду та оцінки публікацій усіх користувачів;
- модуль перегляду жанрів товарів;
- модуль перегляду даних про компанії та пов'язані з ними продукти;

3. Розробити дизайн frontend частини веб-додатку:

3.1. Створити попередні прототипи та ідейні нароби до майбутнього дизайну сторінок.

3.2. Розробити frontend частину веб-додатку з урахуванням прототипів та запланованого дизайну, логіки роботи його компонентів.

4. Створити тестові та демонстраційні набори даних контенту до наповнення програмного продукту

5. Виконати детальне тестування веб-додатку, зокрема використовуючи наступні види тестування:

5.1. Функціональне тестування

5.2. Адаптивне тестування

5.3. Тестування користувацького інтерфейсу та зручності використання

Результатом виконання поставленого завдання згідно описаних етапів має бути доконаний веб-додаток магазину відеоігор та порталу ігрових публікацій, що містить серверну частину роботи з даними з імплементацією автентифікації та авторизації та клієнтської частини для зручного виконання дій через графічний інтерфейс користувача на веб-сторінках.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2. Аналіз аналогів програмного продукту

Веб-додаток магазину відеоігор та ігрових публікацій дозволяє зробити більш доступними та популярними придбання ліцензійних копій товарів та гуртування за спільними інтересами в широкій та колоритній сфері бізнесу. На сьогодні існує велика кількість подібних ресурсів, що надають користувачам по всьому світу можливість зручного придбання відеоігор та пов'язаних товарів та/або спілкуватись зі спільнотою інших людей, схожих за інтересами. Для розгляду аналогів за критеріями популярності та позитивного рейтингу вибрані ресурси наведені у таблиці 1.1.

Таблиця 1.1

Порівняння обраних аналогів

Назва	Направленість	Публікації по направленості	Наявність та тип магазину	Мови та валюти
IGN	Відеоігри, молодіжні медіа проекти (комікси, фільми тощо)	Так	Так Колекційні фізичні товари	Англійська, мови країн Азії та Європи (без Української) USD
G2A	Відеоігри	Ні	Так Ліцензійні відеоігри	Англійська, мови країн Європи (без Української) USD, GBP, EUR, JPY тощо
Game Shop	Відеоігри та фізичне приладдя, інструменти для гри	Ні	Так Ігрові консолі, аксесуари ПК, ліцензійні відеоігри до консолей	Українська, російська UAH
Steam	Відеоігри та програмне забезпечення	Так	Так Відеоігри, додатковий завантажувальний контент, програмне забезпечення, ігрове приладдя	Локалізована на більшості мов спілкування Використовує місцеві валюти підтримуваних країн

Опишемо кожний з обраних ресурсів детальніше.

IGN – розважальний веб-ресурс, що орієнтується на популярні течії розваг, такі як відеоігри, фільми, музику, телебачення та інші розважальні

елементи масової культури. Історія ресурсу починається в 1996му році під назвою Imagine Games Network як мережа веб-сайтів, що сфокусовані кожний на різній тематиці. Наразі веб-ресурс є головним майданчиком компанії для створення онлайн-публікацій та обговорень на орієнтовані теми, щорічна нагорода найпопулярнішим продуктам сфери розваг поруч із іншими подібними ресурсами, як Game Awards та BAFTA. Інтернет-магазин дозволяє придбати колекційні товари на тематику популярних відеоігор, серіалів тощо. Даний веб-ресурс орієнтується на світову спільноту та потребує знання англійської мови.

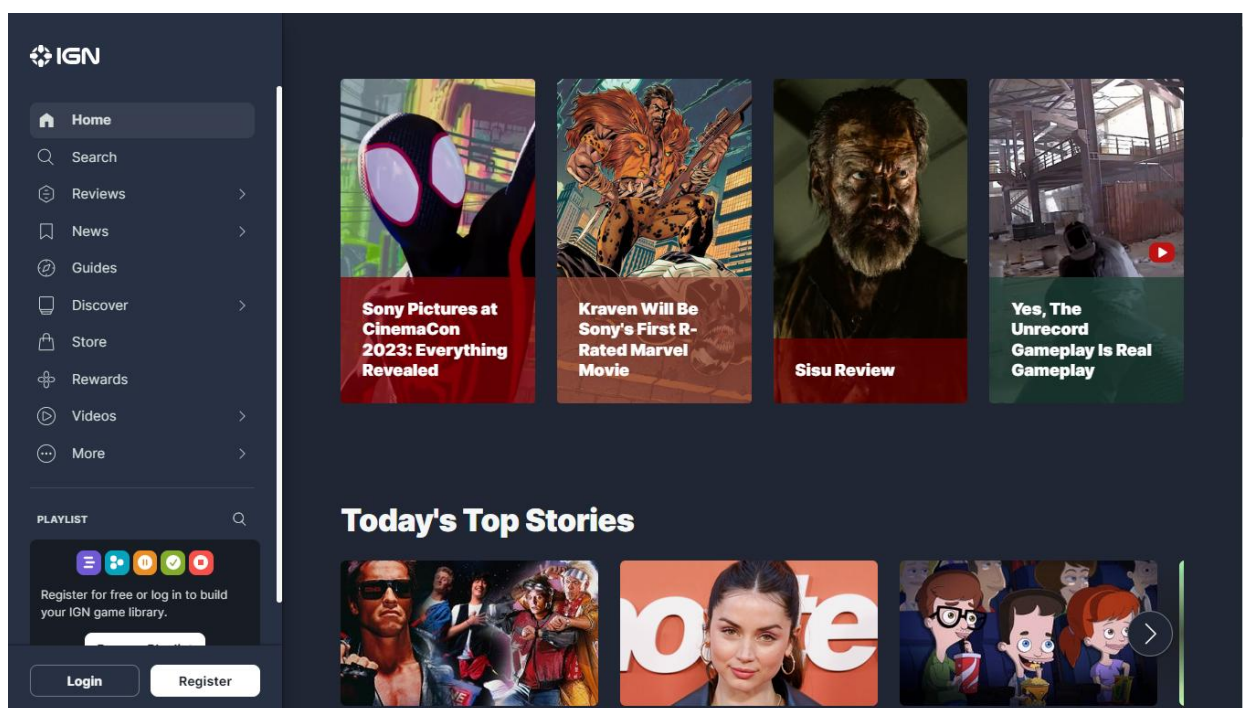


Рис.1.1 – Веб-ресурс IGN

G2A – міжнародний онлайн-маркетплейс зі штаб-квартирою в Нідерландах та офісами в Польщі та Гонконгу. Заснована у 2010му році в Ряшові, Польща, даний веб-магазин станом на 2020й рік має більше ніж 20 мільйонів користувачів. Даний веб-ресурс надає послуги з продажу цифрових ключів відеоігор, активаційних кодів та колекційних товарів. Наявна велика кількість доступних валют та мов до налаштування сайту, проте поки що без української мови та гривні

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

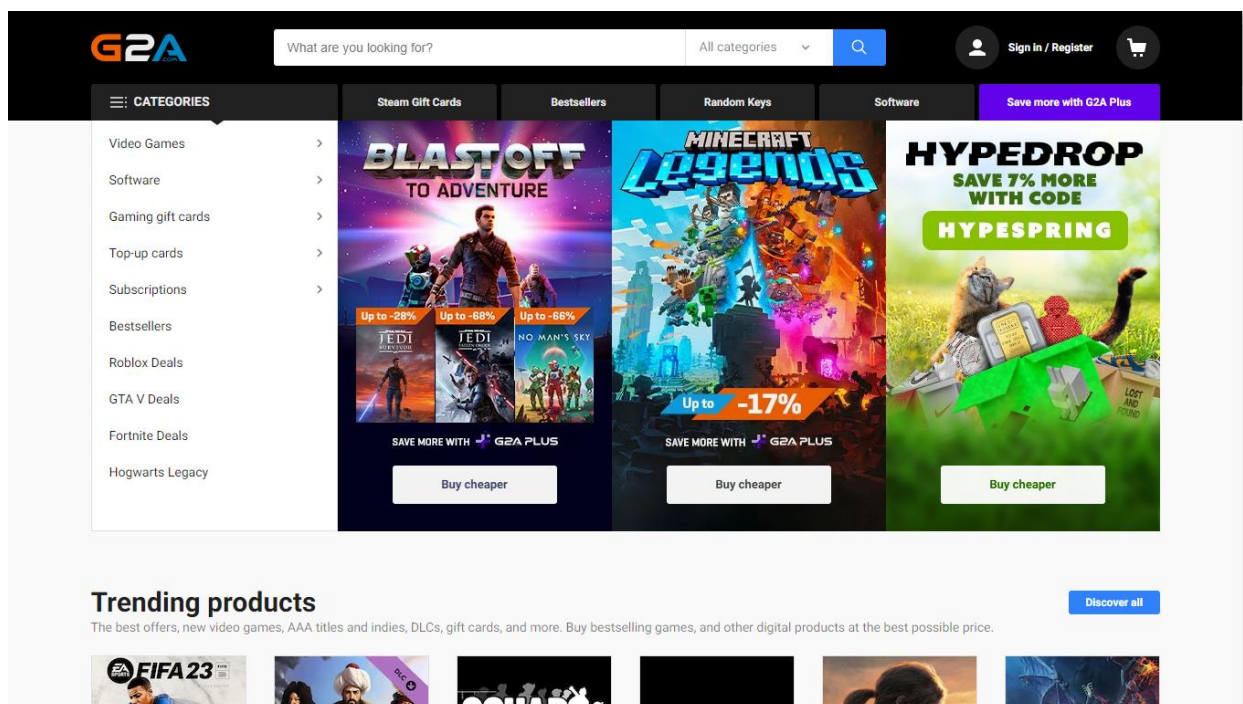


Рис.1.2 – Онлайн-магазин G2A

Game Shop – український сервіс із продажу ігрових консолей та ліцензійних відеоігор до них, аксесуарів та інструментів для гри на ПК, додаткових колекційних товарів. Компанія заснована в 2007му році як магазин з продажу ігрових консолей, наразі має велику кількість інших дочірніх сервісів, такі як оренда консолей, сервісний ремонт апаратури тощо. Даний веб-сервіс пропонує користувачам асортимент з нових ігрових консолей та аксесуарів до них (геймпади), ПК (мишки, клавіатури, навушники тощо) та відеоігор до ПК та консолей. Сервіс працює українською та російською мовами, приймає в оплату гривні.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

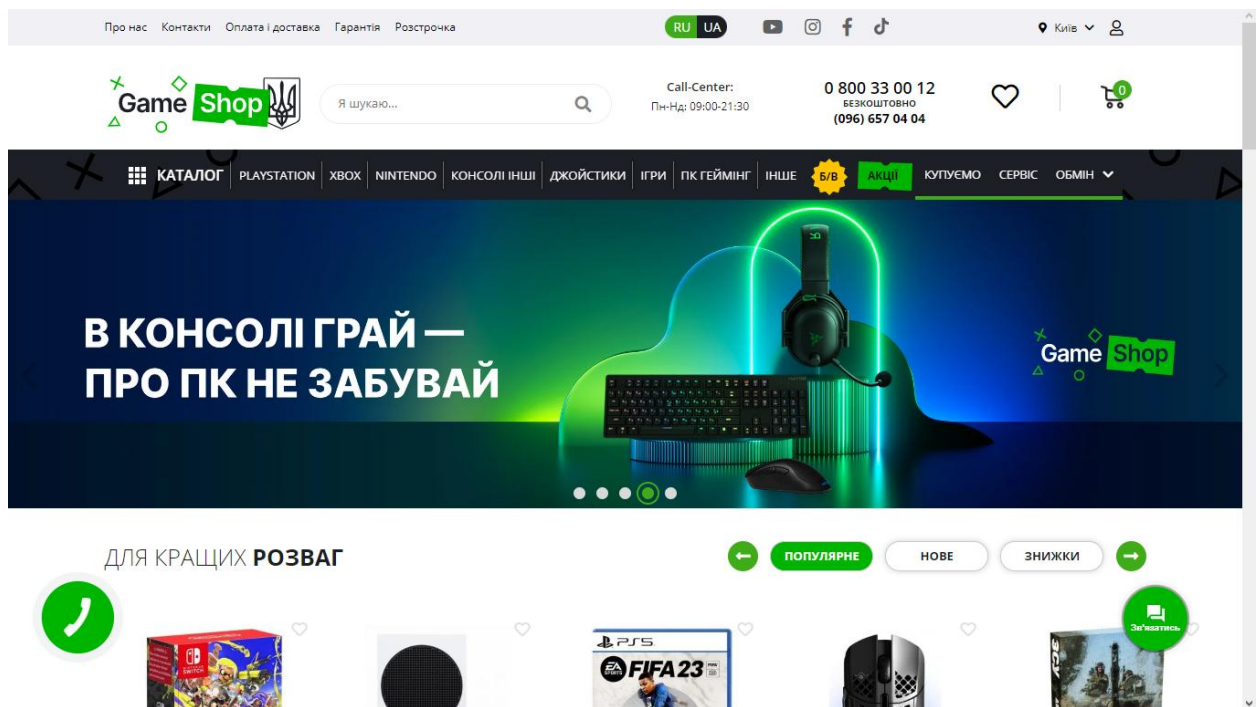


Рис.1.3 – Онлайн-магазин Game Shop

Steam — міжнародний сервіс компанії Valve, відомого розробника відеоігор, вперше представлений громадськості 22 березня 2002 року та замінив сервіс-попередницю WON. Сервіс надає послуги цифрової дистрибуції, багатокористувацьких ігор і спілкування гравців. Steam використовується також для отримання автоматичних оновлень і новин, як для продуктів самої Valve, так і сторонніх розробників, серед яких всесвітньо відомі та невеликі indie-компанії. Крім відеоігор, сервіс пропонує інструменти для розробників ігор, музику, фільми та геймерське обладнання. Не має фізичних магазинів у світі, адже є повністю мережевим веб-сервісом. Станом на червень 2021 року через Steam поширюється понад 50 тис. товарів. У квітні 2019 року кількість акаунтів сервісу досягла 1 млрд. Щодня сервісом у середньому користуються 17 мільйонів людей. Хоча Valve ніколи не висвітлювала особисто показники продажів, підраховано, що їхній сервіс охоплює 70 % ринку цифрової дистрибуції відеоігор.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

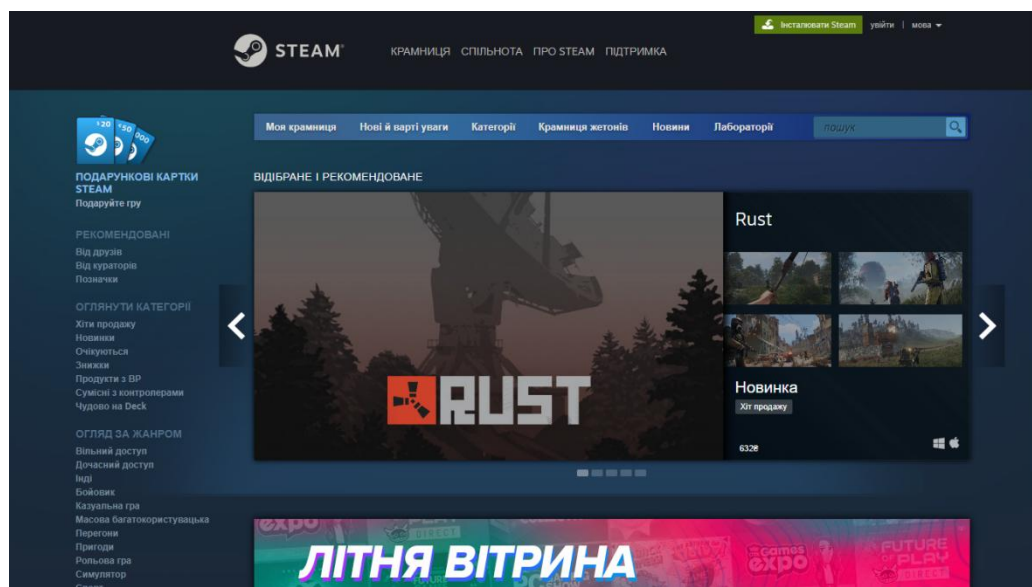


Рис.1.4 – Початкова веб-сторінка крамниці сервісу Steam

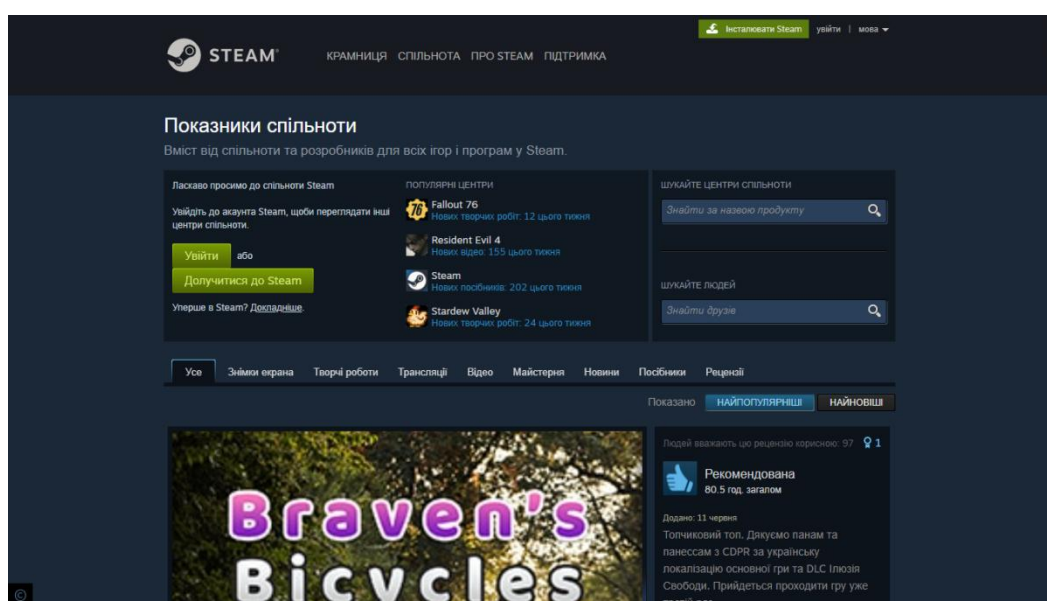


Рис.1.5 – Веб-сторінка спільноти сервісу Steam з новинами, публікаціями та медіа-контентом по певним товарам

Порівнюємо дані ресурси по функціональним можливостям.

Таблиця 1.2

Функціональні можливості описаних ресурсів

Назва	Детальний пошук	Додавання у кошик	Покупка без реєстрації	Використання криптовалюти	Категоризація
IGN	+	+	+	-	+
G2A	+	+	+	+	+
Game Shop	+	+	+	+	+
Steam	+	+	-	-	***

* при купівлі зазначається пошта та можливі розсилки

** категоризація входить у детальний пошук товарів на сторінці пошуку

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Переглянувши список аналогів веб-ресурсів, можна зробити висновки, що всі веб-ресурси мають детальну категоризацію своїх товарів з можливістю детального пошуку по різним характеристикам продуктів та якісні сторінки списків знайдених товарів. Збереження даних у кошику навіть без реєстрації дозволяє одноразове використання веб ресурсу та швидший доступ до оплати обраних товарів.

Таким чином, основними рисами веб-додатку магазину мають бути зручний користувацький інтерфейс на сторінках та можливість детального пошуку, зокрема перегляду даних по категоріям.

1.3. Вибір архітектури програмного продукту

Для реалізації веб-додатку магазину відеоігор та порталу ігрових новин було вирішено використати трирівневу клієнт-серверну архітектуру [1]. Використання даної архітектури дозволяє спростити розділення основних фрагментів розробки та відділити логіку обробки запитів користувача та візуальне відображення сторінок. Трирівнева архітектура передбачає створення бази даних як 3го компонента для зберігання даних окремо від сервера. Кожен з компонентів за коректної розробки має можливість складатись із повністю різних наборів технологій, що також робить веб-додатки більш гнучкими.

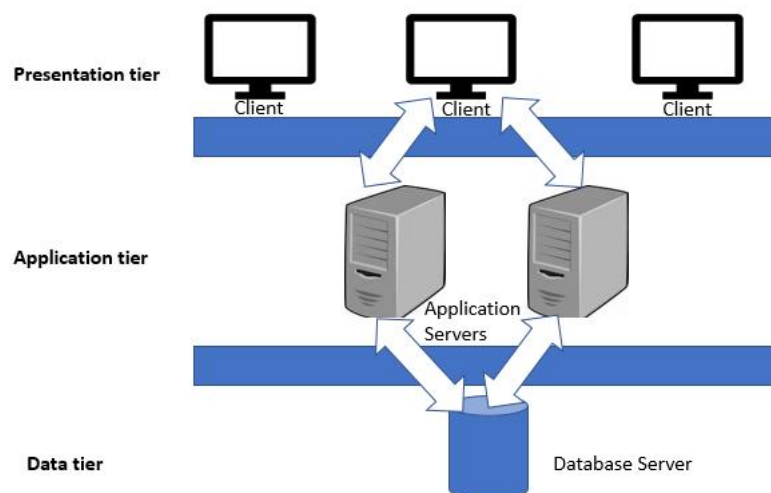


Рис.1.6 – Загальне зображення трирівневої клієнт-серверної архітектури

Особливості даної архітектури напряду залежать від обраних технологій, але в загальному вона працює наступним чином:

- клієнтський додаток (Клієнт) доступний користувачам через мережу Інтернет з використанням веб-браузерів. Його використання супроводжується створенням запитів до сервера;

- серверний додаток (Сервер) приймає запити (HTTP(S)) від клієнтів та обробляє їх, згідно шляху запиту, параметрів та тіла даних виконує логіку з їх обробки та використання зі зміни чи створення, видалення даних на базі даних;

- сервер бази даних (БД) очікує та приймає авторизовані запити від серверу з роботи над даними, згідно яких створює, читає та надає, редагує чи видаляє дані з наявних таблиць. БД може знаходитись на віддаленій апаратурі так само, як і на машині разом зі Сервером. Результат роботи БД надається у відповідь Серверу, а той надає оброблений чи чистий результат Клієнту;

Підтримку роботи додатків можливо забезпечити через хмарні сервіси, що надають послуги не тільки зі запуску цих додатків, але й також із окремої підтримки БД, надання статистики використання, аналізу даних БД та збереження даних на випадок виникнення помилок [2].

Отже, можна зробити висновки, що обраний шаблон архітектури забезпечує всі необхідні можливості до забезпечення виконання вимог до ПЗ та побудови гнучкого веб-додатку. Використання обраної архітектури забезпечує успішну реалізацію проекту та розвиток згідно нових пропозицій та вимог користувачів.

1.4. Обґрунтування вибору інструментальних засобів

Для створення веб-додатку магазину відеоігор та порталу ігрових публікацій було вирішено обрати векторний онлайн-сервіс розробки інтерфейсів та прототипування Figma, інструмент розробки програмного забезпечення та моделювання StarUML, інтегроване середовище розробки для JavaScript, HTML та CSS JetBrains WebStorm, платформу Node.js,

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

програмний каркас розробки серверної частини веб-застосунків Express, відкриту JavaScript бібліотеку для створення інтерфейсів користувача React, інструментарій для управління ізольованими Linux-контейнерами Docker, мову програмування TypeScript із використанням необхідних додаткових модулів та бібліотек типів до оригінальних, що будуть застосовані в розробці, об'єктно-реляційну систему керування базами даних PostgreSQL.

Figma — це спільний веб-додаток для розробки інтерфейсу з додатковими офлайн-функціями, доступними в настільних програмах для macOS і Windows. Набір функцій Figma зосереджений на дизайні інтерфейсу користувача та взаємодії з користувачем, з акцентом на співпрацю в реальному часі, використовуючи різноманітні редактори векторної графіки та інструменти для створення прототипів.

StarUML — це інструмент розробки програмного забезпечення для системного моделювання з використанням уніфікованої мови моделювання, а також мови моделювання систем і класичних нотацій моделювання.

Додаток StarUML є потужним інструментом для роботи з UML-діаграмами, доступний на операційних системах MacOS, Windows 10 та вище, Linux та Fedora. Можливість встановлення плагінів, включно з плагінами кодогенерування для певних мов програмування, є зручною перевагою даного додатку. Редактор підтримує всі стандарти мови моделювання UML, включно з новими. Має зручний та зрозумілий інтерфейс, зберігає створені діаграми в одному файлі формату mdj та надає можливість експорту у форматах PNG, JPEG, SVG.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

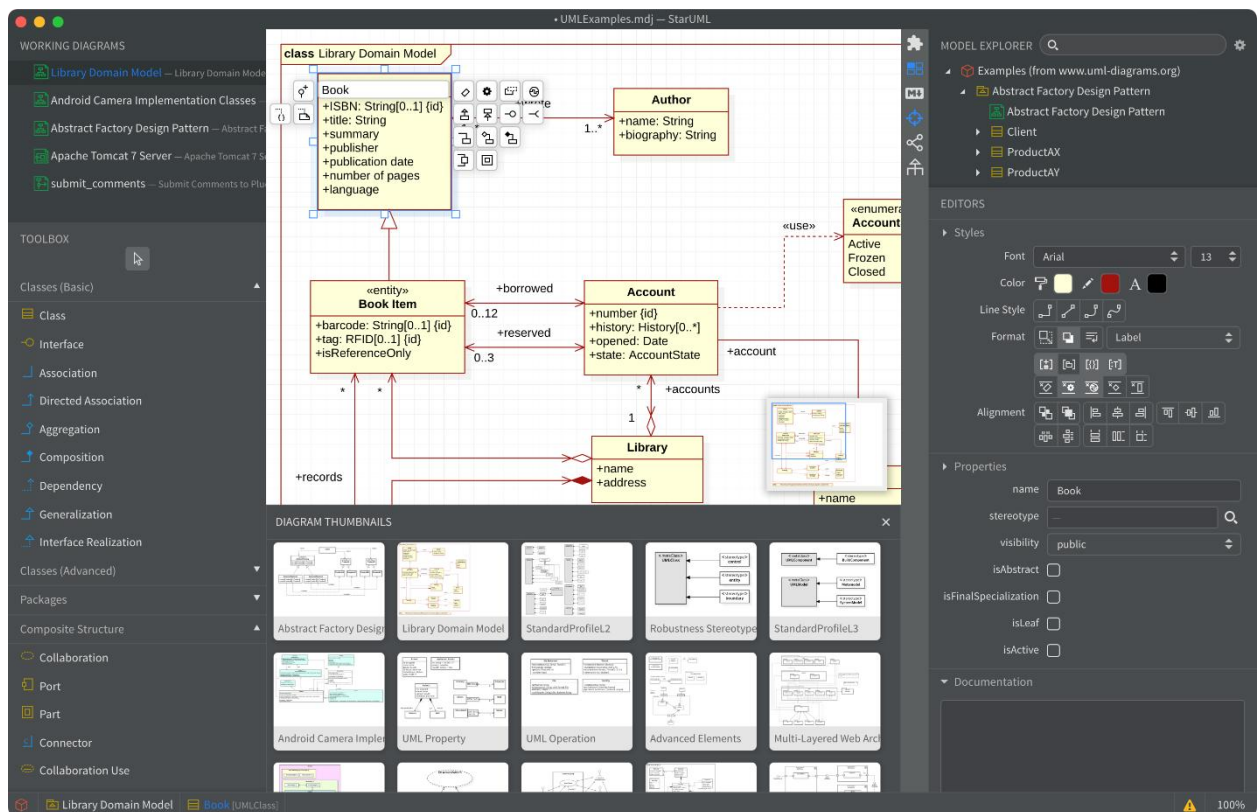


Рис.1.7 – Інтерфейс додатку StarUML

JetBrains WebStorm — інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart.

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних файлах і теках проекту, а також вкладеного в HTML).

Node.js — це міжплатформне серверне середовище з відкритим кодом, яке може працювати в Windows, Linux, Unix, macOS тощо. Node.js — це серверне середовище виконання JavaScript, яке працює на V8 JavaScript Engine і виконує код JavaScript поза веб-браузером.

Express.js, або просто **Express**, — це базова основа веб-додатків для створення RESTful API за допомогою Node.js, випущена як безкоштовне

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижморя О.Г.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

програмне забезпечення з відкритим вихідним кодом за ліцензією MIT. Він призначений для створення веб-додатків і API. Його називають де-факто стандартною серверною структурою для Node.js через його популярність та високу ефективність, простоту розробки з використанням даної технології та мінімалістичності [3].

React (також відомий як **React.js** або **ReactJS**) — це безкоштовна інтерфейсна бібліотека JavaScript із відкритим кодом для створення інтерфейсів користувача на основі компонентів. Сумісна з обраною мовою програмування TypeScript через встановлення бібліотеки типів. На даній бібліотеці базується велика кількість веб-фреймворків, а сама бібліотека завдяки встановленню додаткових пакетів дозволяє розробку реактивних односторінкових застосунків, що не потребують постійного оновлення сторінок.

Docker — це набір продуктів «платформа як послуга» (PaaS), які використовують віртуалізацію на рівні ОС для доставки програмного забезпечення в пакетах, які називаються контейнерами.

Додатки веб-системи контейнеризуються та працюють в окремих середовищах, не впливаючи одне на одного та не зазнаючи впливу від зміни материнської ОС, дозволяється створювати контейнери баз даних (PostgreSQL).

nginx – вільний вебсервер і проксі-сервер, необхідний для клієнтського додатку, поміщеного в контейнер, для зв'язку зі сервером, а саме переадресації запитів по адресі сервера в мережі веб-додатку, створеній через Docker.

TypeScript — це безкоштовна мова програмування високого рівня з відкритим вихідним кодом, розроблена та підтримувана Microsoft. Це суворі синтаксична надбудова JavaScript і додає необов'язкову статичну типізацію до мови. Він розроблений для розробки великих додатків і транспілюється на JavaScript. Оскільки це надбудова JavaScript, існуючі програми мовою JavaScript також є дійсними програмами на TypeScript [4].

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

PostgreSQL — безкоштовна об'єктно-реляційна система керування базами даних з відкритим кодом, яка наголошує на розширюваності та сумісності з SQL.

Дана СКБД має велику кількість можливостей. Функції дозволяють виконувати деякий код безпосередньо сервером бази даних. Індеси наступних типів: B-дерево, хеш, R-дерево, GiST, GIN, та за необхідності можна створити нові типи індексів. PostgreSQL підтримує одночасну модифікацію БД декількома користувачами за допомогою механізму Multiversion Concurrency Control (MVCC). Завдяки цьому виконуються вимоги ACID, і практично відпадає потреба в блокуванні зчитування. СКБД підтримує великий набір вбудованих типів даних:

- числові типи:
 - цілі;
 - з фіксованою крапкою;
 - з нефіксованою крапкою.
- грошовий тип;
- символічні типи довільної довжини;
- двійкові типи (включаючи BLOB);
- типи «дата/час»;
- булевий тип;
- перерахування;
- геометричні примітиви;
- мережеві типи:
 - IP і IPv6-адреси;
 - CIDR-формат;
 - MAC-адреса.
- UUID-ідентифікатор;
- XML-дані;
- JSON-дані;
- масиви;

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

- OID-типи;
- псевдотипи;

Крім того, користувач може самостійно створювати нові необхідні йому типи та програмувати для них механізми індексування за допомогою GiST.

Sequelize – ORM на базі TypeScript та Node.js із підтримкою Oracle, PostgreSQL, MySQL, MariaDB, SQLite, SQL Server та багатьох інших СКБД. Використання даної технології дозволяє в програмному вигляді описувати моделі майбутньої БД з урахуванням потреб валідації, підтримки типів та виконання різноманітних операцій над даними напряму зі сторони БД.

1.5. Аналіз вимог до програмного продукту

Далі буде розглянуто високорівневі вимоги до системи для забезпечення її безперервної роботи та виконання необхідного функціоналу для задоволення поставлених цілей та стандартів, специфікацій тощо. Окрім цього, буде описано загальні умови для роботи користувачів з даним програмним забезпеченням, можливості, обмеження системи.

1. Бізнес-вимоги:

- основні цілі: проект створюється з метою надання послуг офіційної дистрибуції відеоігор та сервісів до них, ведення зацікавленими користувачами публікації на пов'язану тематику;
- можливості: перегляд та придбання товарів (відеоігор та внутрішніх товарів, сервісів до них), перегляд та створення онлайн-публікацій на тематику відеоігор та новітніх графічних, інформаційних технологій;
- представлення проекту: проект буде реалізовано у вигляді веб-додатку зі зручним інтерфейсом та швидким наданням інформації за запитом.

2. Вимоги користувачів:

- зовнішній користувач – авторизований користувач:

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Можливість пошуку товарів за жанрами, компаніями, цінами, назвами тощо.
 2. Можливість перегляду списку компаній, чиї товари наявні для огляду у додатку, та жанрів.
 3. Можливість перегляду даних про певний товар та відгуків користувачів про них, компанії та жанри разом з популярними іграми компаній / жанрів.
 4. Можливість перегляду власного профілю та власних виконаних замовлень за весь час.
 5. Можливість перегляду списку бажань на сайті, поточного кошика та короткого огляду товарів у ньому.
 6. Можливість перегляду створених публікацій та відгуків до них.
 7. Можливість створення власних публікацій та надавання для перевірки перед їх схваленням
- внутрішній користувач – модератор:
1. Можливості авторизованого користувача.
 2. Можливість створення нових товарів та виконання дій над існуючими на окремій адміністративній частині додатку.
 3. Можливість створення публікацій підвищеного рівня з інформацією про останні події на сайті, останні додані товари тощо.
 4. Можливість перевірки створених користувачами публікацій згідно правил користування веб-додатком, їх схвалення чи відхилення та повернення на виправлення.
- внутрішній користувач – адміністратор:
1. Можливості модератора.
 2. Можливість змінювати ролі усіх користувачів веб-додатку, а саме модераторів та користувачів.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Характеристика об'єкту комп'ютеризації:

Користувач матиме можливість придбати ліцензійні копію чи ключ доступу до відеогри, товару в ній тощо із попереднім переглядом даних про товар, відгуків. Окрім цього, користувач матиме можливість перегляду публікацій на сайті, які пов'язані з тематикою відеоігор та подій у них, новин з ІТ, пов'язаних з даною сферою. Модератори через веб-інтерфейс матимуть змогу працювати з базою даних та створювати дані про нові товари, компанії та жанри, редагувати, ховати та видаляти існуючі. Адміністратор через веб-інтерфейс також матиме змогу змінювати ролі доступу користувачів та модераторів згідно зміни їх реального стану відносно веб-додатку: клієнт у працівника та навпаки.

4. Функціональні вимоги:

- підключення сторонніх API для реалізації оплати та авторизації;
- створення записів про товари, компанії, жанри та їх пов'язання, публікацій, відгуків про товари, публікації.

5. Нефункціональні вимоги:

- сприйняття:
 1. Інтерфейс не має бути складним та вимагати додаткової підготовки, повинен бути зручним та звичним для досвідчених, простим для нових користувачів Інтернету.
 2. Схема роботи з веб-додатком не має суттєво відрізнятися від звичної схеми роботи з іншими онлайн-магазинами.
- продуктивність
 1. Система має підтримувати мінімум 50 користувачів, що одночасно працюють, пов'язаних зі спільною БД.
 2. Час відповіді системи на запити не має перевищувати 5-7 секунд для простих запитів та 10 секунд для запитів із великою кількістю даних.

6. Системні вимоги:

- вимоги до середовища виконання (мінімальна комплектація):

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Підтримка браузера Google Chrome версії 45+.
2. Підтримка браузера Mozilla Firefox версії 38+.
3. Підтримка браузера Microsoft Edge версії 12+.
4. Підтримка браузера Opera версії 30+.
5. Підтримка браузера Safari версії 9+.

– вимоги до СКБД та доступу до даних:

У ядрі системи повинна бути представлена СКБД реляційного доступу.

7. Бізнес-правила:

Система повинна відповідати всім стандартам побудови сучасних веб-застосунків, що використовує архітектуру SPA (Single-Page Application) [5].

8. Антивимоги:

- не використовує інформацію, заборонену нормами моралі та законодавством;
- не використовує інформацію, що є недійсною або може зашкодити психіці користувачів;
- не містить реклами.

9. Бізнес-логіка:

Система пропонує придбання ліцензійних товарів за ціною офіційних джерел, безкоштовне створення онлайн-публікацій.

Висновки до першого розділу

Проведений аналіз предметної області дозволив визначити основні напрямки веб-додатку магазину з порталом ігрових новин. Необхідно реалізувати збереження даних про замовлення та власне дані про товари, жанри, компанії, користувачів, а також пошук даних за запитам користувачів. Для відокремлення доступу необхідно створити систему ролей користувачів та розробити систему автентифікації та авторизації.

Огляд наявних аналогів показав, що основними вимогами до таких систем є: зручний користувацький інтерфейс на сторінках, можливість

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

детального пошуку, перегляд даних по категоріям, зручне створення нових даних працівниками тощо.

В роботі обґрунтовано вибір архітектури для реалізації системи:

- мову програмування TypeScript;
- UI-бібліотеку React;
- каркас серверних додатків Express;
- СКБД PostgreSQL.

Будуть використані наступні програмні продукти : векторний редактор Figma, інструментарій моделювання StarUML, інтегроване середовище розробки WebStorm.

Було проведено аналіз функціональних, нефункціональних та бізнес вимог до майбутнього програмного продукту, сформовано бізнес-правила, антивимоги, системні вимоги тощо.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ, МОДЕЛЮВАННЯ, АНАЛІЗ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Розробка моделі програмного забезпечення на логічному рівні

З урахуванням вимог до ПЗ, було сформовано ключові тези, необхідні для виконання моделювання та аналізу ПЗ, проектування системи в цілому. Базуючись на цьому, можна побудувати діаграму використання продукту (UseCase Diagram) [6].

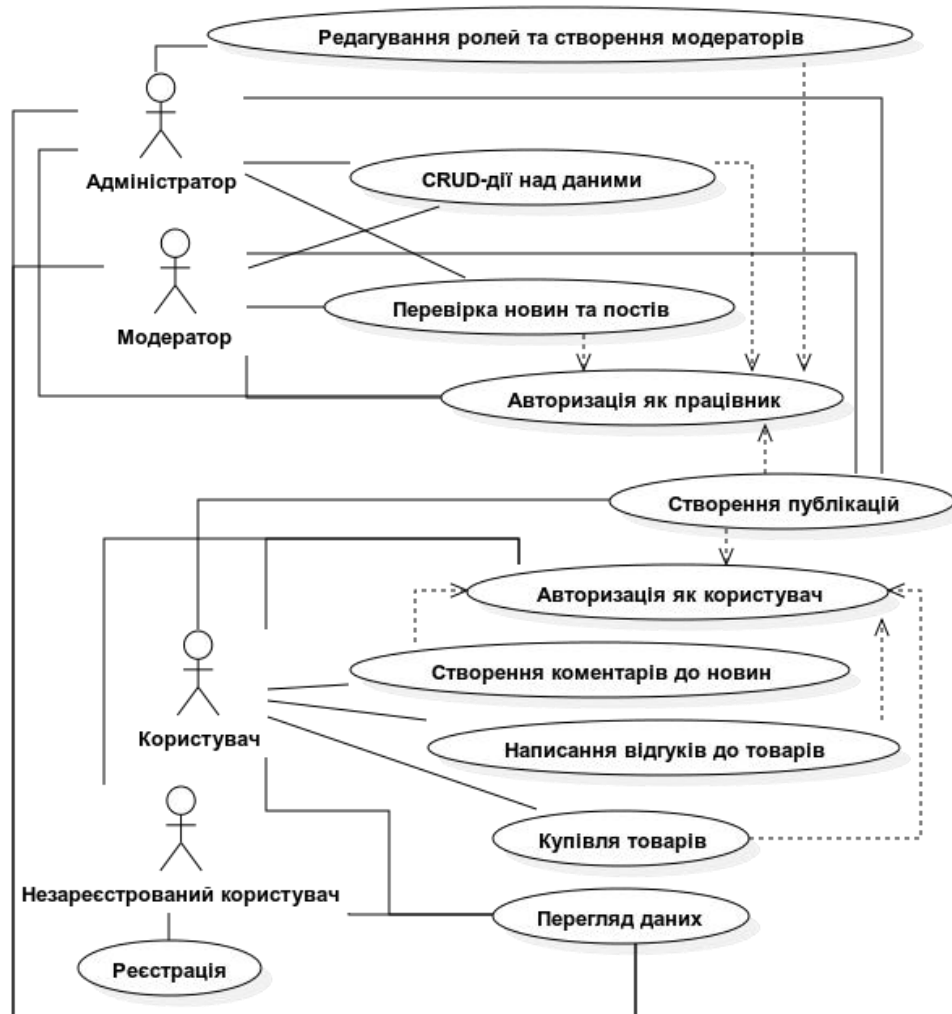


Рис.2.1 – Діаграма варіантів використання

Дана діаграма розділяє користувачів на 4 типи акторів за їх станом автентифікації, авторизації. У нижній частині описані варіанти використання, які можуть виконувати незареєстровані та зареєстровані користувачі (включно з працівниками сайту), у верхній частині – варіанти використання виключно для працівників (модераторів та адміністратора).

Незареєстрований користувач має можливість вільного перегляду будь-яких даних сайту без перешкод. Для виконання інших дій користувач сайту має виконати реєстрацію, якщо ще не був зареєстрований, або авторизацію по даним свого облікового запису.

Авторизований користувач отримує можливість додавання товарів до кошику для подальшого придбання товарів, оцінки товарів веб-сайту та публікацій користувачів, створення та редагування власних публікацій тощо.

Користувачі-працівники сайту з роллю модератора мають доступ до адміністративної частини, де мають можливість виконання CRUD-операцій над даними, а саме: створювати, редагувати та видаляти товари, жанри та компанії, наявні в додатку; змінювати стан порушень публікацій та описування причини їх блокування.

Користувачі-адміністратори мають можливість зміни ролей інших користувачів, таким чином ефективно створюючи облікові записи через реєстрацію та їх зміну для працівників сервісу.

Разом з діаграмою використання, маємо можливість проаналізувати та оформити специфікацію прецедентів, яку наведено у додатку А.

Для опису алгоритму роботи програмного забезпечення ефективними є побудова та використання діаграми діяльності [7]. Дана діаграма виконує функцію опису можливих дій у використанні додатку. Вона дозволяє моделювати виконавчий потік протягом усього сеансу роботи з веб-додатком з точки зору будь-якого типу користувачів. Можливі види дій та діяльності описані асинхронно та з умовами доступу до них, що дозволяє уявити виконання різних сценаріїв одночасно для великої кількості користувачів.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

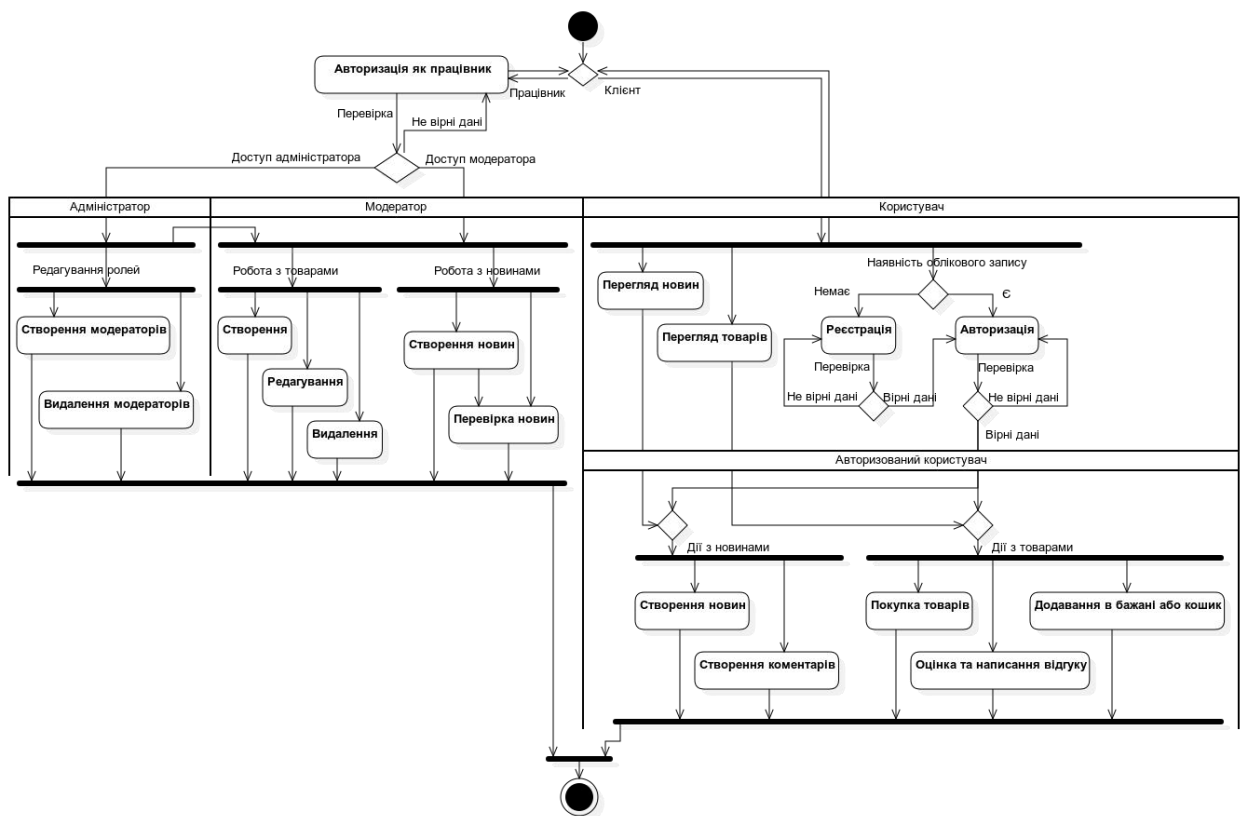


Рис.2.2 – Діаграма діяльності системи

За своєю суттю дана діаграма описує діяльності, доступні користувачам за типами їх стану автентифікації та авторизації. Через перевірки умов та розділення потоків діяльностей зазначені всі можливі дії в додатку, які виконуються паралельно різними користувачами.

Опис діаграми діяльності:

- вибір напряму діяльності: робота з додатком як користувач на користувацькій сторінці або як працівник на адміністративній сторінці (обов'язкова авторизація), з можливістю переходу назад до вибору (зміна сторінки);
- адміністратор:
 - робота з ролями облікових записів (зміна ролей «Користувач» та «Модератор» для фактичного створення чи видалення працівників);
 - діяльність модератора.
- модератор:
 - робота з товарами: створення нових, редагування даних чи доступу до існуючих, їх видалення;

– робота з публікаціями та новинами: перевірка користувацьких публікацій.

– неавторизований користувач:

– перегляд публікацій;

– перегляд товарів;

– реєстрація;

– авторизація (потребує наявності облікового запису).

– авторизований користувач:

– виконання дій з публікаціями: створення власних, створення відгуків до існуючих;

– виконання дій з товарами: додавання у кошик та бажані, купівля з кошику, створення відгуків.

Об'єктно-орієнтована модель зазвичай зображується за допомоги діаграми класів. Дана UML-діаграма дозволяє описати статичну структуру даних у системі, їх взаємодію, відносини одне до одного тощо. Надалі на її основі є можливість створення бази даних SQL з ідентичною структурою через програмний код серверної частини.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

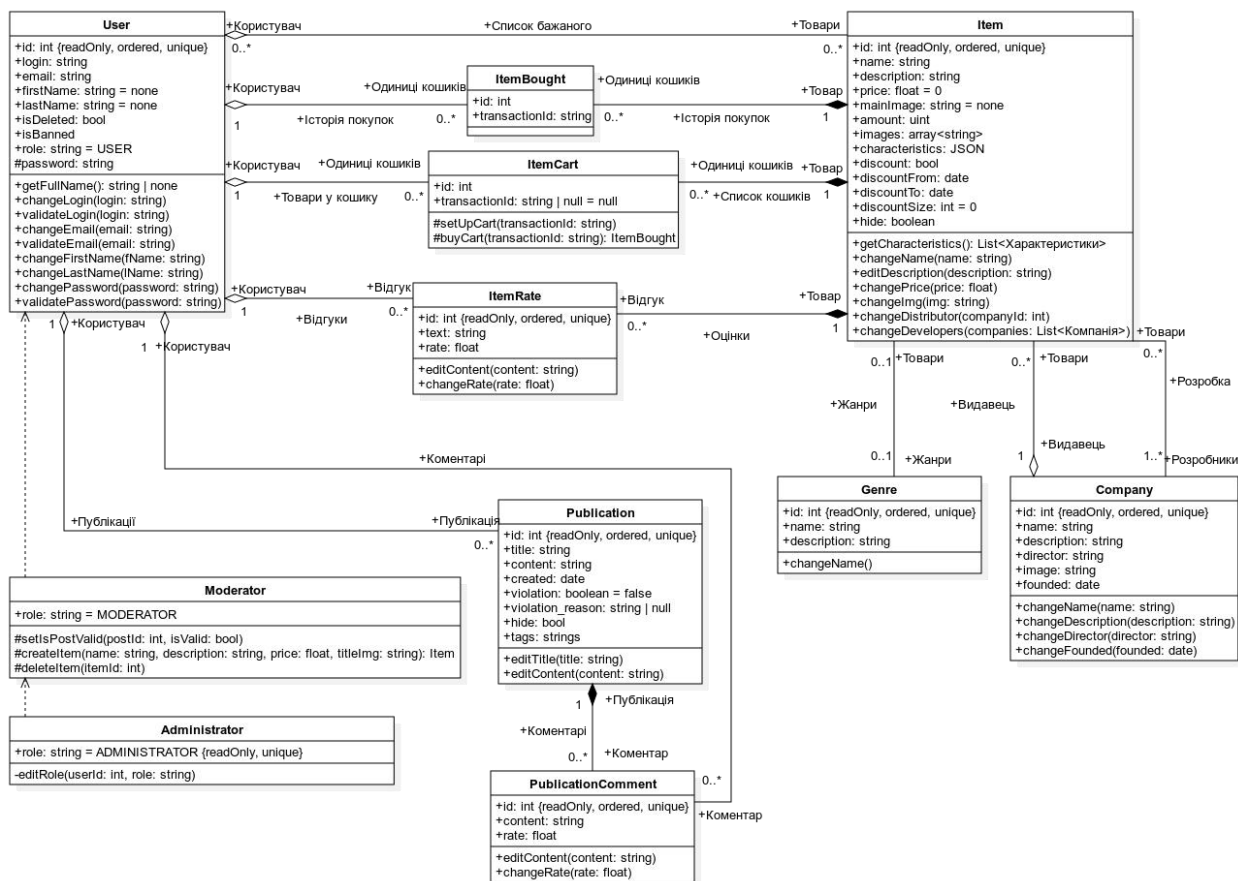


Рис.2.3 – Діаграма класів програмного забезпечення

Опис діаграми класів:

1. **Клас User** є класом облікового запису користувачів системи, містить дані користувача та методи до їх безпечної зміни. Успадковується класами Moderator та Administrator, які додають методи роботи з іншими класами та мають власні значення ролей. Використання даного класу є ключовим для ефективної роботи на сайті через ідентифікацію користувачів.

2. **Клас Item** включає інформацію про певний товар, включно з даними про галерею зображень товару та ID компаній-виробників, компанії-видавника тощо. Детальні характеристики товару (вікова група, мінімальні системні вимоги, особливості товару тощо) описуються через тип даних JSON, який зберігається одним параметром.

3. **Клас Company** містить інформацію про певну компанію та пов'язані з нею товари, включно з коротким описом. Зв'язок з класом товарів дозволяє разом з об'єктами класу компанії отримувати пов'язані дані про

відеоігри, які компанія розробила або видала в реальному житті, спрощуючи ідентифікацію тих чи інших товарів чи компаній.

4. **Клас Genre** містить дані про жанр, включно з описом цього жанру. Зв'язок даного класу з класом товарів дозволяє більш детально описувати дані про той чи інший об'єкт товару, який веб-сайт пропонує до огляду та придбання.

5. **Клас ItemRate** відповідає за збереження даних про відгуки користувачів про товари включно з його оцінкою. При отриманні об'єктів класу товару, до нього додаються всі пов'язані об'єкти оцінок для формування блоку коментарів та розрахунку оцінки товару на сайті.

6. Зв'язок “Список бажаного” описується через службовий клас Wishlist, що має обмежений набір параметрів для максимально оптимального опису взаємозв'язку.

7. **Клас ItemCart** відповідає за збереження користувацьких кошиків. Об'єкти даного класу використовуються об'єднано в один масив при запиті даних користувача, створюються чи видаляються простою дією над товаром зі сторони авторизованого користувача. Описується додатковим параметром transactionId, який необхідний для підготовки товарів кошика до придбання та пошуку необхідних об'єктів.

8. **Клас ItemBought** відповідає за збереження історію придбань користувача. Даний клас повністю ідентичний класу ItemCart та створюється через метод об'єкту кошика, таким чином імплементуючи в коді процес придбання товару.

9. **Клас Publication** містить дані публікацій, новин та постів у веб-додатку, включно з параметрами приховання, перевірки тощо. Даний клас безпосередньо пов'язаний з об'єктом користувача, який було використано для створення публікації, що створює зв'язок авторства типу “один-до-багатьох”.

10. **Клас PublicationComment** є класом відгуків до публікацій разом з її оцінкою. Завдяки об'єктам даного класу з'являється можливість іншим

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачам веб-сайту залишати конструктивну критику чи власне враження від написаної публікації, створюючи зв'язок додатковий «багато-до-багатьох» між об'єктами користувачів та публікації.

За будування бази даних SQL на основі даної діаграми, зв'язки агрегації та композиції матимуть вигляд окремих таблиць для створення зв'язків «багато-до-багатьох» та «багато-до-одного» відповідно [8]. Зв'язок наслідування користувача та модератора у свою чергу для виконання вимоги до наявності авторизації перетвориться на систему перевірки ролі користувача через надання даних.

У сучасному програмуванні та розробці будь-яких програм існує велика кількість допоміжних технологій та технік, усталених стилів розробки тощо, які використовуються для полегшення роботи розробника. Використання цих патернів не тільки дозволяє спростити роботу, але й покращити ефективність роботи додатків через перевірені схеми.

Поширеним патерном проектування, підтримка якого надається «з коробки» використовуваного фреймворку Express.js, є «Middleware» [9]. Даний патерн реалізовує використання проміжних функцій для перевірки запитів, їх попередньої обробки та окремої обробки помилок. Дані функції надаються створеному об'єкту додатку Express або у налаштованих шляхах маршрутизації запитів. Їх використання дозволяє перевіряти за запитом, отриманим від користувачів, роль та можливість виконання дій над даними, попередньо обробляти помилки та надавати вчасну відповідь на них без участі головної частини додатку для економії ресурсів.

Для роботи даного патерну буде використано власні розроблені функції авторизації користувачів та валідації доступу для них на основі технологій JWT [10]. Токени зберігають у собі дані про ID, логін, пошту, роль та головне зображення користувача, що дозволяє авторизувати користувача за його запитом на сервері та зображувати актуальний стан автентифікації на сайті.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

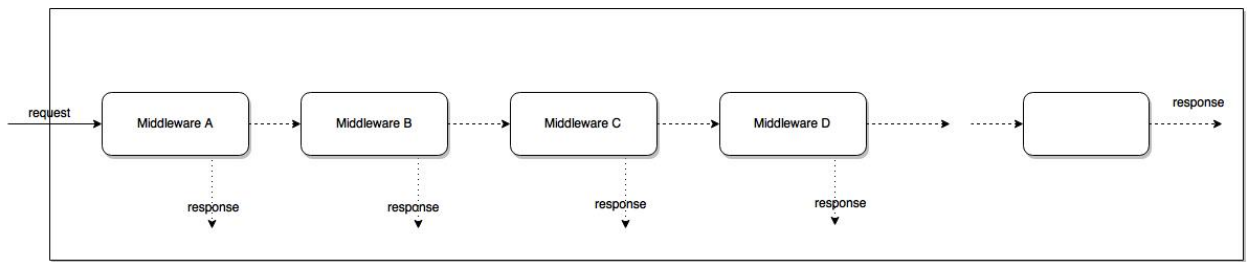


Рис.2.4 – Схема використання патерну «Middleware»

Для зображення взаємодії об'єктів системи мовою UML використовується діаграма послідовності. Вона описує детальні операції та дії, які виконуються акторами та системою у відповідь на дії акторів. Зосередженість на часі у даній діаграмі дозволяє описати загальний процес обробки запитів та дії, які виконуються асинхронно з процесами, відносний час, який проходить від виконання дії до отримання відповіді.

Протягом моделювання та аналізу майбутнього додатку було вирішено розробити діаграми для дій користувачів та працівників відносно їх можливостей. Дії, описані діаграмою, та подальші операції з результатами можуть виконуватись окремо від загальної послідовності або в іншому порядку без змін у внутрішній послідовності дій. Рисунки діаграм послідовності наведені у додатку Б.

Опис діаграми послідовності (рис.Б.1):

1. Користувач переходить по статичним сторінкам, створюючи запити до клієнтської частини, яка повертає незмінний вміст.
2. Користувач переходом на сторінку з даними про певний товар, жанр, компанію, публікацію або їх список ініціює запит до клієнтського додатку, який подає запит до серверу. Сервер відповідає запитаними даними, які Клієнт відмальовує на сторінці користувача
3. Виконання авторизації супроводжується наданням Клієнтом форми та надсиланням запиту на перевірку даних до Сервера. Останній виконує валідацію введених даних та надає помилку або дозвіл, що зберігається у браузері користувача.
4. Перехід на створення публікації перевіряє стан авторизації користувача. У задовільному випадку користувач отримує сторінку для

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

заповнення даних та написання публікації, яка надсилається до Серверу для збереження. Сервер надає повідомлення про помилки або успішність, яке Клієнт відображає та надає користувачу.

5. Додавання товарів у кошик та/або улюблених товарів також перевіряє авторизованість користувача, у задовільному випадку надсилається запит на зміни у БД до Сервера з відповіддю з помилкою або успіхом.

6. Сторінка купівлі товарів, доступна за авторизації, отримує від Серверу даних про товари у кошику та надає користувачу сторінку кошику з можливістю видалення товарів та купівлі залишених. За отриманим запитом Сервер додає до історії купівлі залишені товари та повністю очищає кошик.

Діаграма послідовності Працівників (рис.Б.2) бере до уваги роботу лише з адміністративною частиною веб-додатку та включає перевірку ролей для доступу до функціоналу адміністратора.

Опис діаграми послідовності (див. рис.Б.2):

1. Працівник за запитом статичної сторінки отримує від Клієнту вміст даної сторінки.

2. За запитом сторінки зі списком товарів, жанрів тощо або детальних даних про певний запис Клієнт надсилає запит на отримання необхідних даних з БД до Сервера, який надає відповідь у вигляді даних або помилки.

3. На запит сторінки редагування існуючих даних товарів, жанрів тощо або створення нових Клієнт надає відповідну форму, дані якої потім надсилаються зі запитом до Сервера. Останній у відповідь надає результат роботи (успіх або помилка).

4. Відкриття сторінки перевірки певної статті супроводжується запитом на отримання даних та їх відображення разом з формою підтвердження коректності статті. Результат перевірки у вигляді запиту надається Серверу для зміни статусу статті.

5. Працівник з роллю адміністратора має можливість відкрити сторінку користувачів для редагування ролей. Клієнт за запитом до Сервера отримує список всіх користувачів, які надає у браузер Працівника. Будь-які

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

дії з користувачами з боку Працівника створюють відповідний запит на зміну даних до Сервера, який повідомляє у відповідь про успішність чи певну помилку.

2.2. Фізична модель та прототип програмного забезпечення

Для зображення та опису взаємодію компонентів системи мовою UML застосовується діаграма компонентів [11]. Дана діаграма застосовується для опису роботи системи та моделювання фізичних аспектів систем. На цьому типі діаграм використовуються дані з діаграми класів у контексті компонентів системи та їх взаємодії між собою для створення запланованого функціоналу.

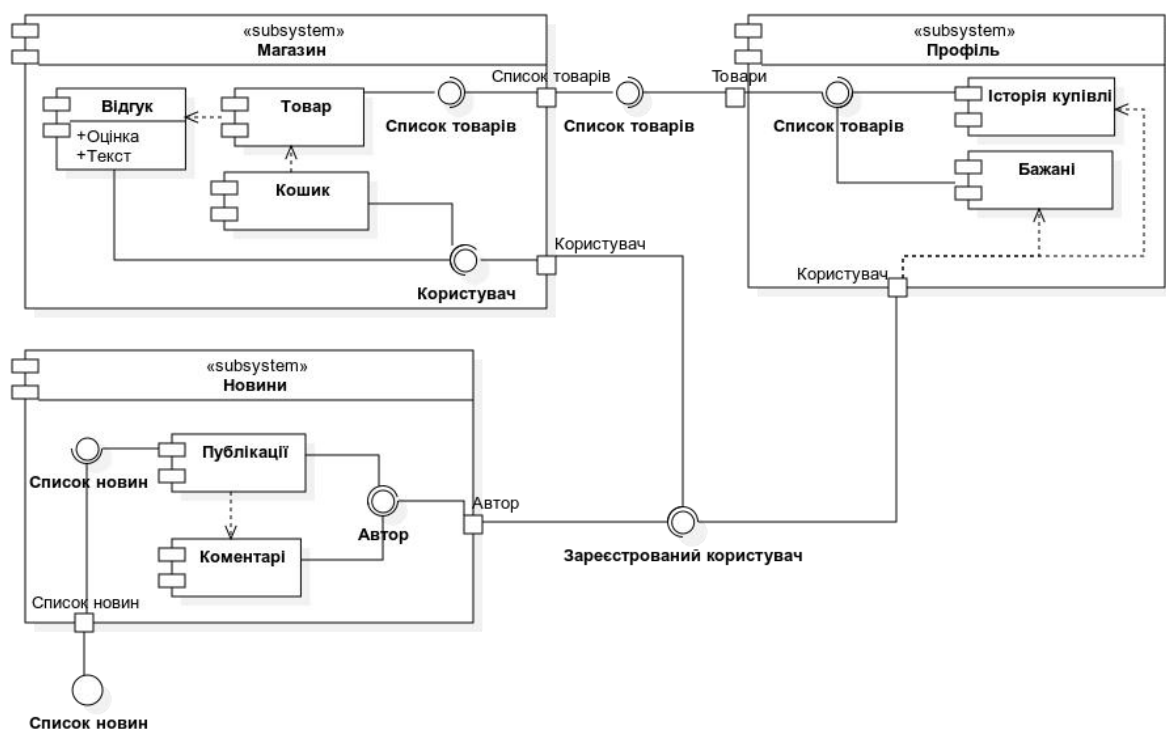


Рис.2.5 – Діаграма компонентів системи

Опис діаграми компонентів:

- система розділена на головні 3 підкомпоненти, які включають основні компоненти системи та взаємопов'язані одне з одним;
- робота з частиною функціоналу компонентів необхідно бути зареєстрованим;
- підсистема «Магазин» представляє сторінку магазину з можливістю огляду товарів та читання, створення відгуків, додавання у кошик;

– підсистема «Профіль» відповідає за збереження та представлення даних певного користувача, а саме куплених та обраних товарів та відгуків до них. Дана підсистема разом з компонентами всередині повністю описують зареєстрованого користувача;

– підсистема «Новини» представляє частину сайту із представленням публікацій користувачів та працівників сайту, відгуків до них.

Протягом виконання проектування та моделювання програмного забезпечення було вирішено використовувати архітектуру односторінкового додатку (Single-Page Application, SPA). Дана архітектура надає користувачу зручний досвід користування програмою, маючи наступні переваги:

1. Швидкість реагування на дії користувача: головною перевагою даної архітектури є швидкодія завантаження необхідних даних без потреби перезавантаження усієї веб-сторінки завдяки завантаженню необхідних даних у форматі JSON та динамічному відображенні лише необхідної частини.

2. Швидкість розробки: швидкість розробки проектів у команді залежить від можливості розробників працювати одночасно без створення перешкод одне одному. За обраної архітектури розробка клієнтської та серверної частини набагато прискорюється через їх розділення, таким чином зміни на frontend не впливають на backend і навпаки.

3. Стабільність та ефективність системи: разом з першим завантаженням сторінки завантажується основна логіка веб-додатку, незмінна протягом роботи з ним. Реагування на дії користувача відбувається із виконанням асинхронних запитів, що дозволяє використовувати зв'язок із меншою пропускнуою здатністю та навантаження на серверне забезпечення.

4. Адаптивність для різних пристроїв: використання даної архітектури є гарним вибором відносно адаптивності додатку через відсутність постійних оновлень веб-сторінок та можливості створення плавних переходів без проблем з налаштуванням чи дублювання основної структури додатку, створення неперервності роботи.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

На основі виконаної роботи з проектування системи та її опису було створено макети вигляду додатку завдяки векторному онлайн-сервісу Figma у вигляді додатку для настільних ПК.

Даний етап передбачає створення первинних дизайнів кількох сторінок сайту, які будуть використані в подальшій розробці клієнтської частини та можуть бути істотно змінені за потреби підлаштуватись під можливості обраних технологій або можливості покращити дизайн.

Загальний дизайн сторінок складається з типових компонентів, які використовуються в сучасній розробці веб-сайтів, а саме блоків header, основного контенту та footer, та використовує ретельно підібрану палітру кольорів для створення атмосфери для сторінок веб-додатку, що відповідає предметній області [12] (рис.2.6).

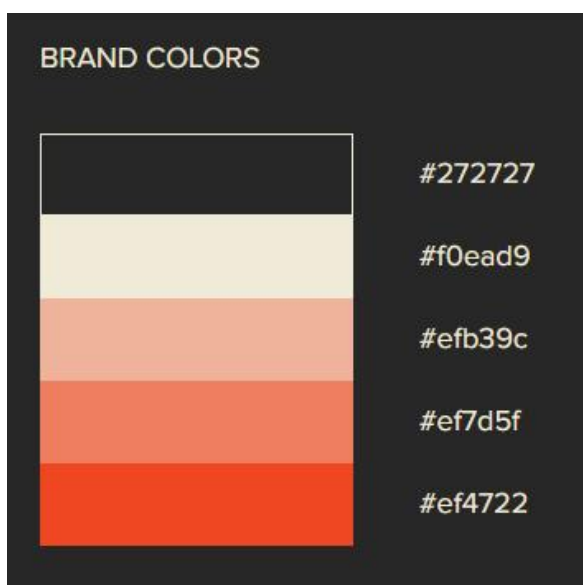


Рис.2.6 – Палітра кольорів дизайну сторінок веб-сайту

Основним кольором даної палітри є чорний #272727, що використовується для більшої частини текстів на білому (ffffff або #f0ead9) фоні та як стандартний фоновий колір елементів. Другорядним кольором встановлено #f0ead9 для текстів, що знаходяться на темному фоні. Кольором акцентів та особливих елементів (посилання, кнопки тощо) використовується #ef4722 з освітленими підвидами #ef7d5f та #efb39c.

Шрифтом заголовків було обрано шрифт Montserrat Alternates, шрифтом текстових блоків – Merriweather.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		40



Рис.2.7 – Прототип дизайну сторінки пошуку товарів

Зображена сторінка пошуку товарів (рис.2.7) складається з:

- поля введення назви товарів;
- блоку фільтрів розширеного пошуку за жанрами, компаніями та середньою оцінкою;
- сітки товарів, які знайдено за результатом пошуку за параметрами;
- елементу пагінації для перемикання сторінок знайдених результатів

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

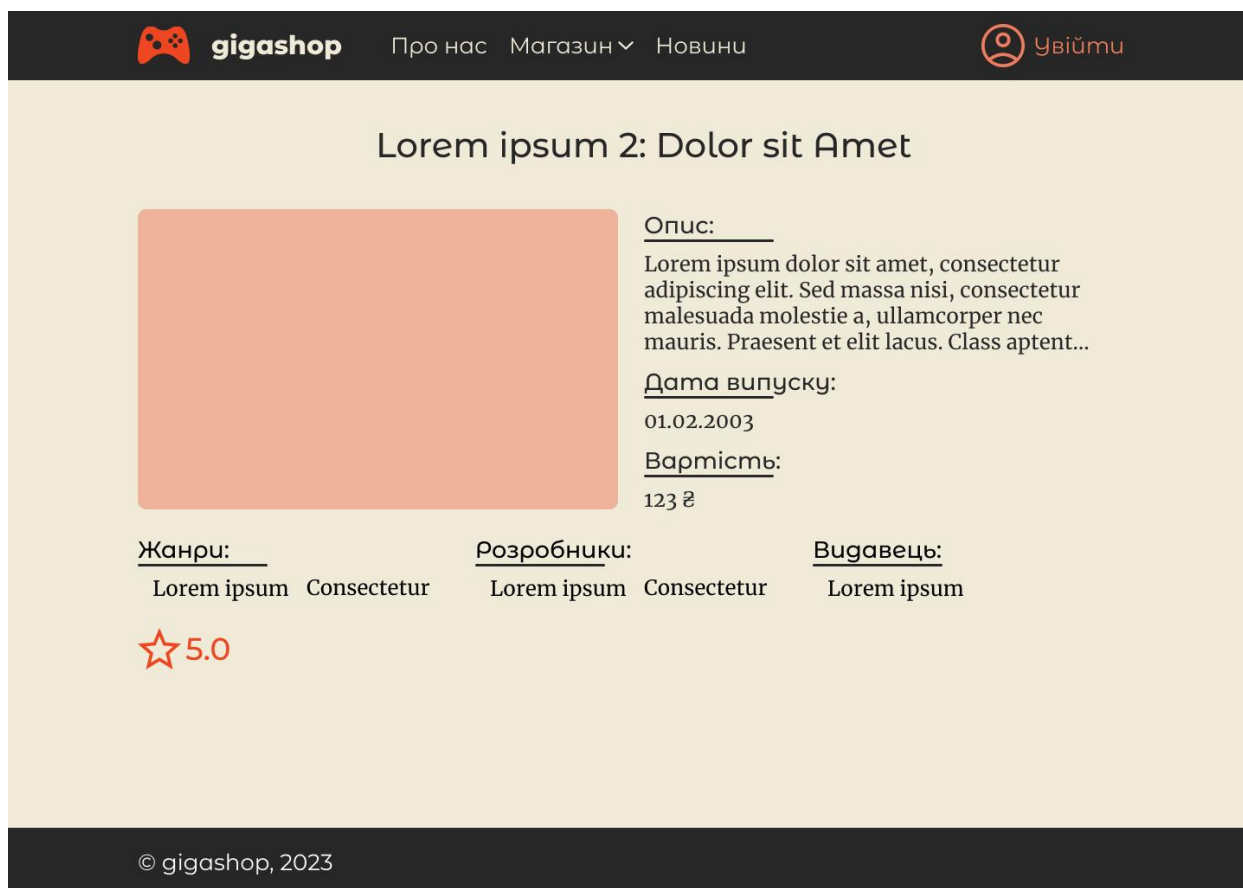


Рис.2.8 – Прототип дизайну сторінки окремого товару

Зображена сторінка окремого товару (рис.2.8) складається з:

- області для галереї зображень товару;
- блоків опису параметрів та деталей товару, а саме:
 - власне опис товару;
 - дата випуску товару;
 - вартість товару;
 - список жанрів;
 - список розробників;
 - видавець.
- оцінка товару;

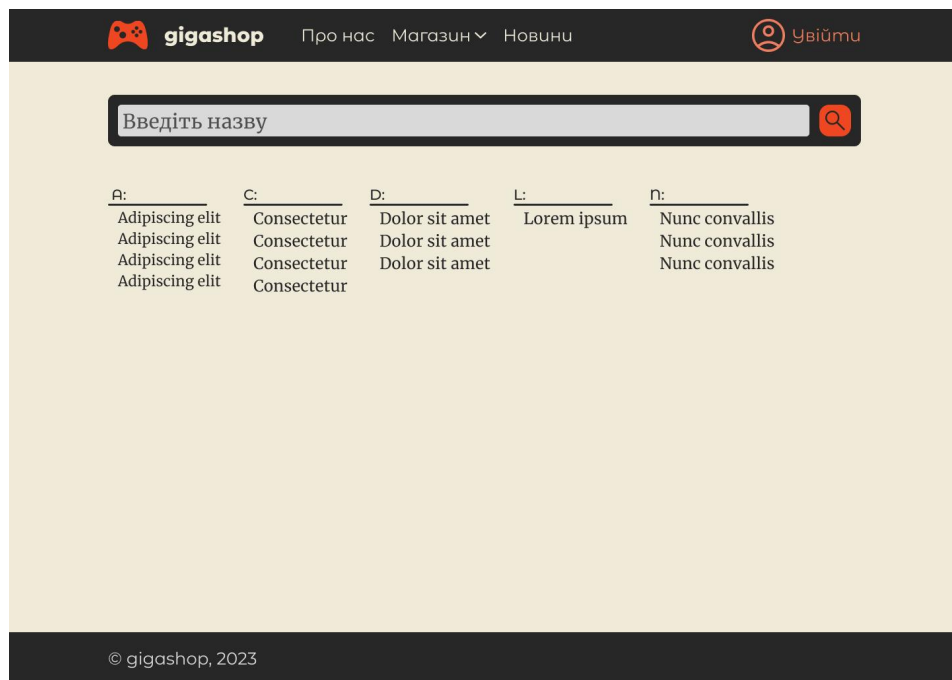


Рис.2.9 – Прототип дизайну сторінки пошуку жанрів

Зображена сторінка пошуку жанрів (рис.2.9) містить поле введення назви жанру та алфавітну сіткою зі списками жанрів літери алфавіту даної одиниці сітки.

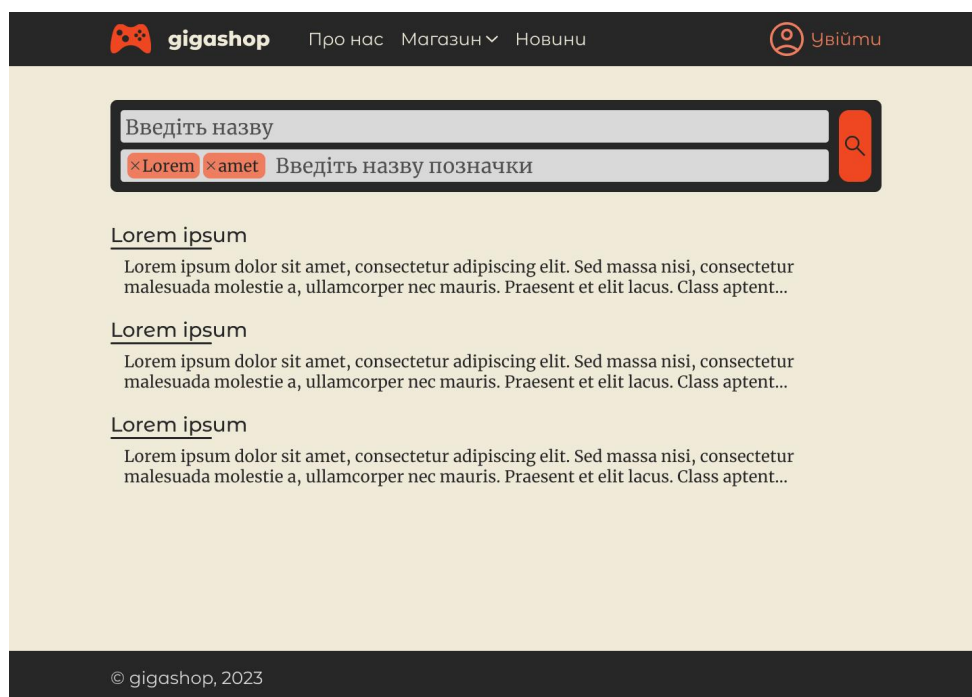


Рис.2.10 – Прототип дизайну сторінки пошуку публікацій

Зображена сторінка пошуку публікацій (рис.2.10) містить поля введення назви (заголовку) публікації, тегів (позначок) публікацій та список публікацій за результатом пошуку.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

2.3. Реалізація серверної частини веб-додатку

Серверний додаток має на меті реалізувати в програмному коді запланований функціонал обробки даних, авторизації, інтерпретувати діаграму класів у діючі моделі даних БД. У розробці було використано серверне середовище Node.js та веб-фреймворк Express.js, мову програмування TypeScript, ORM Sequelize та СКБД PostgreSQL.

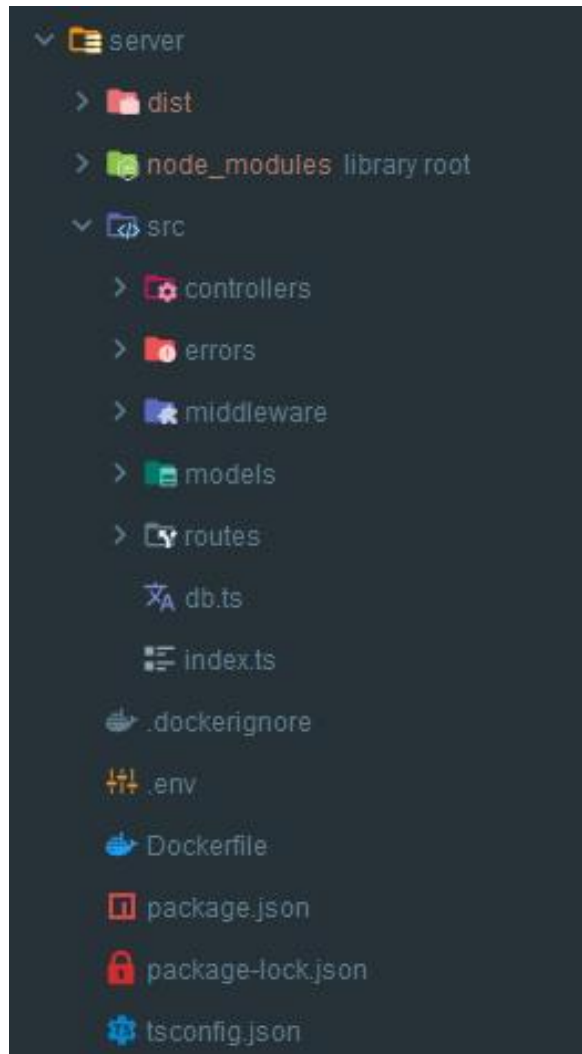


Рис.2.11 – Структура файлів серверного додатку

Через специфіку розробки на мові TypeScript, серверний додаток містить налаштування мови, згідно яких відбувається перевірка типів, якості коду, імпортування зайвого функціоналу, встановлюється папка для перетворення проекту на мову JavaScript та інші налаштування.

Лістинг файлу tsconfig.json серверної частини (сховані коментарями налаштування не вказано):

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
  "compilerOptions": {

    /* Language and Environment */
    "target": "es2016", /* Set the JavaScript language
version for emitted JavaScript and include compatible library declarations. */
    "module": "commonjs", /* Specify what module code is
generated. */
    "rootDir": "./src", /* Specify the root folder within
your source files. */
    "moduleResolution": "node", /* Specify how TypeScript looks up a
file from a given module specifier. */
    "typeRoots": [
      "node_modules/@types",
      "src/@types"
    ], /* Specify multiple folders that act like
    "sourceMap": true, /* Create source map files for emitted
    "outDir": "./dist", /* Specify an output folder for all
emitted files. */
    "removeComments": true, /* Disable emitting comments. */
    "sourceRoot": "./src", /* Specify the root path for
debuggers to find the reference source code. */
    "inlineSources": true, /* Include source code in the
sourcemaps inside the emitted JavaScript. */
    "noEmitOnError": true, /* Disable emitting files if any type
checking errors are reported. */
    "allowSyntheticDefaultImports": true, /* Allow 'import x from y' when a
module doesn't have a default export. */
    "esModuleInterop": true, /* Emit additional JavaScript to
ease support for importing CommonJS modules. This enables
'allowSyntheticDefaultImports' for type compatibility. */
    "forceConsistentCasingInFileNames": true, /* Ensure that casing is correct
in imports. */
    "strict": true, /* Enable all strict type-checking
options. */
    "noImplicitAny": true, /* Enable error reporting for
expressions and declarations with an implied 'any' type. */
    "noUnusedLocals": true, /* Enable error reporting when local
variables aren't read. */
    "noUnusedParameters": true, /* Raise an error when a function
parameter isn't read. */
    "noImplicitReturns": true, /* Enable error reporting for
codepaths that do not explicitly return in a function. */
    "skipLibCheck": true /* Skip type checking all .d.ts
files. */
  },
  "exclude": [
    "node_modules"
  ]
}

```

Лістинг файлу package.json:

```

{
  "name": "server",
  "version": "1.0.4",
  "description": "",
  "main": "dist/index.js",
  "scripts": {
    "dev": "ts-node-dev --poll src/index.ts",
    "build": "rimraf ./dist && tsc",
    "start": "npm run build && node dist/index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Oleksiiichuk Mykyta",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.1.0",
    "clouinary": "^1.37.0",
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    "jsonwebtoken": "^9.0.0",
    "pg": "^8.8.0",
    "pg-hstore": "^2.3.4",
    "sequelize": "^6.28.0",
    "uuid": "^9.0.0"
  },
  "devDependencies": {
    "@types/bcrypt": "^5.0.0",
    "@types/express": "^4.17.15",
    "@types/jsonwebtoken": "^9.0.1",
    "@types/node": "^18.11.18",
    "@types/uuid": "^9.0.2",
    "nodemon": "^2.0.20",
    "rimraf": "^4.1.1",
    "ts-node-dev": "^2.0.0",
    "typescript": "^4.9.4"
  }
}

```

Розгортання серверного додатку відбувається за сценарієм, описаним у локальному Dockerfile, що містить 2 стадії розгортання. Перша стадія виконує повну інсталяцію пакетів та будування готового додатку через транспіляцію в JavaScript, друга стадія встановлює лише необхідні пакети та копіює готовий код з першої стадії [13].

Лістинг файлу Dockerfile серверного додатку:

```

FROM node:18.9.0-alpine3.16 as development
ARG NODE_ENV=development
ENV NODE_ENV=${NODE_ENV}
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

FROM node:18.9.0-alpine3.16 as production
ARG NODE_ENV=production
ENV NODE_ENV=${NODE_ENV}
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install --omit=dev
COPY --from=development /usr/src/app/dist ./dist
CMD ["node", "dist/index.js"]

```

Кореневий виконавчий файл додатку index.ts відповідає за налаштування додатку, використовуючи весь розроблений функціонал.

Лістинг коду файлу index.ts:

```

import type {Express} from "express";

require("dotenv").config();
const express = require('express');
// @ts-ignore
const {connect} = require('./db');

import initModels from "./models";
initModels();
const PORT = process.env.PORT || 5000;
const cors = require('cors');
const routes = require('./routes');
import errorHandler from "./middleware/ErrorHandlerMiddleware";

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

```

declare global {
  namespace Express {
    interface Request {
      user?: any;
    }
  }
}

const app: Express = express();
app.use(cors());
app.use(express.json());
app.use('/api', routes);

// last middleware!
app.use(errorHandler);

(async () => {
  try {
    await connect();
    app.listen(PORT, () =>
      console.log(`\x1b[32m${new Date().toLocaleString()}: Server started on port
${PORT}\x1b[0m`));
  } catch (error) {
    console.error(`\x1b[31m${new Date().toLocaleString()}: Server error:
${error}\x1b[0m`);
  }
})();

```

Бібліотека cors дозволяє активувати однойменний механізм CORS для захисту даних. Для можливості обробки запитів із наявним токеном авторизації, декларується можлива наявність даних користувача зі змінними полями (тип any).

Для зв'язку з базою даних в окремому файлі db.ts створюється налаштування на основі змінних середовища, що в рамках розробки отримуються з файлу .env та надаються ззовні при розгортанні додатку. Для успішного налагодження зв'язку, передбачено використання циклу підключення на випадок ініціалізації бази даних та її попереднього налаштування паралельно запуску сервера.

Лістинг коду файлу db.ts:

```

const {Sequelize} = require('sequelize');

const sequelize_db = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  process.env.DB_PASSWORD,
  {
    dialect: 'postgres',
    host: process.env.DB_HOST || "localhost",
    port: process.env.DB_PORT
  }
)

const TIMEOUT_DEFAULT = 5000;
const RETRIES_DEFAULT = 5;

const connect = async () => {
  let timeout_env = process.env.CONNECT_TIMEOUT;

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

```

let retries_env = process.env.CONNECT_RETRIES;

let timeout = timeout_env ? parseInt(timeout_env) : TIMEOUT_DEFAULT;
let maxRetries = retries_env ? parseInt(retries_env) : RETRIES_DEFAULT;

let retries = 0;
while (retries < maxRetries) {
  try {
    await sequelize_db.authenticate()
      .then(() => {
        console.log(`\x1b[32m${new Date().toLocaleString()}: Database
connected!\x1b[0m`);
      });
    break;
  } catch (e) {
    console.log(`\x1b[31m${new Date().toLocaleString()}: Database connection
error: ${e}\x1b[0m`);
    retries++;
    console.log(`\x1b[31m${new Date().toLocaleString()}: Retrying in
${timeout}ms...\x1b[0m`);
    await new Promise(res => setTimeout(res, timeout));
  }
}

if (retries === maxRetries) {
  throw new Error(`\x1b[31m${new Date().toLocaleString()}: Connection to database
failed!\x1b[0m`);
}

await sequelize_db.sync({force: false})
  .then(() => {
    console.log(`\x1b[32m${new Date().toLocaleString()}: Database
synchronized!\x1b[0m`);
  });
}

module.exports = {
  sequelize_db,
  connect,
}

```

Початковим етапом розробки сервера після створення початкового файлу додатку є налаштування бази даних завдяки обраній ORM. У процесі налаштування на основі діаграми класів було створено 12 моделей, зв'язки між якими налаштовуються завдяки влаштованим методам бібліотеки Sequelize. Типи даних атрибутів моделей відповідають доступним типам даних PostgreSQL (включно з масивами та JSON). Вміст файлів моделей та їх налаштування наявні у додатку В.

Для отримання даних від БД, було створено схему маршрутизації сервера та відповідні класи контролерів з методами обробки запитів користувачів [14]. Дані методи використовують створені моделі та допоміжні функції отримання даних через обробку параметрів фільтрації WHERE.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

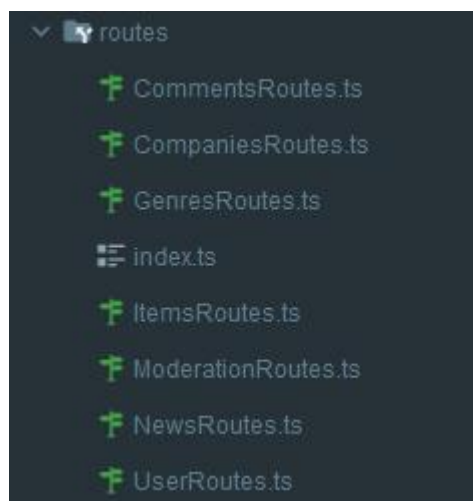


Рис.2.12 – Файли з налаштуваннями маршрутизації серверного додатку в папці routes

Тека routes включає файли-налаштування обробки HTTP-запитів та пов'язання їх з відповідними контролерами, а також застосування проміжного ПЗ (middleware), а саме самостійно розробленого функціоналу з перевірки авторизації та доступу.

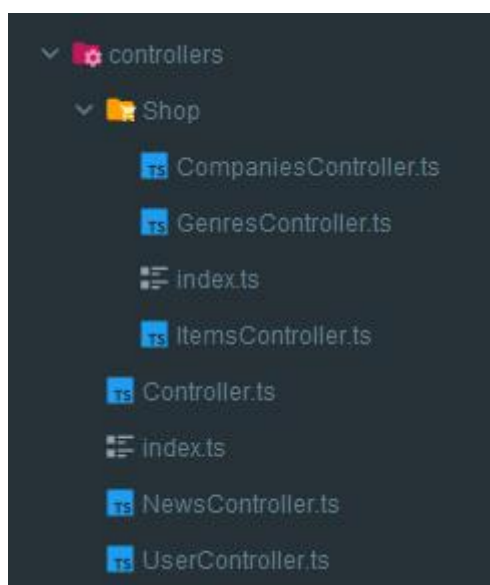


Рис.2.13 – Файли контролерів серверного додатку в папці controllers

Тека controllers включає файли з класами контролерів, які використовуються у маршрутах та для обробки запитів користувачів, надавання даних про помилки у запитах або власне запитані дані.

Код маршрутів та відповідних контролерів наявні у додатку Г.

Для коректної обробки запитів, авторизації та доступів було створено механізм обробки помилок та проміжне ПЗ для перевірки заголовків запитів

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

на наявність токена JWT, його валідність та наявність необхідної ролі у випадку обмеженості доступу до маршрутів та методів контролерів.

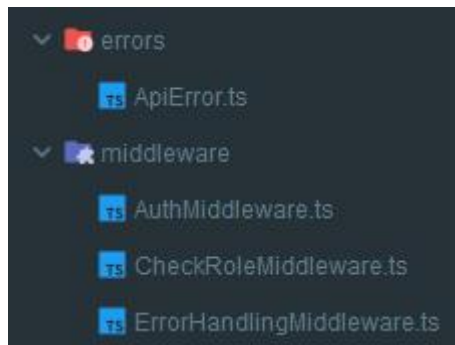


Рис.2.14 – Файли механізму обробки помилок та проміжного ПЗ

Механізм обробки помилок включає у себе створений клас помилок, що успадковує стандартний клас помилок, та розроблене проміжне ПЗ з обробки помилок, які виникають у запитах чи роботі сервера.

Клас помилки `ApiError` додає атрибут числа статусу помилки до існуючих атрибутів класу `Error` та методи зі створення помилки за поширеними HTTP-відповідями на помилки (400+, 500+).

Лістинг коду файлу `ApiError.ts`:

```
class ApiError extends Error {
  status: number;
  errors: object[];

  constructor(status: number, message: string, errors: object[] = []) {
    super(message);
    Object.setPrototypeOf(this, ApiError.prototype);
    this.status = status;
    this.errors = errors;
  }

  static badRequest(message: string = 'Помилка в запиті', errors: any[] = []):
  ApiError {
    return new ApiError(400, message, errors);
  }

  static unauthorized(message: string = 'Неавторизований запит', errors: any[] = []):
  ApiError {
    return new ApiError(401, message, errors);
  }

  static forbidden(message: string = 'Доступ заборонено', errors: any[] = []):
  ApiError {
    return new ApiError(403, message, errors);
  }

  static notFound(message: string = 'Сторінка не знайдена', errors: any[] = []):
  ApiError {
    return new ApiError(404, message, errors);
  }

  static internal(message: string = 'Внутрішня помилка сервера', errors: any[] = []):
  ApiError {
    return new ApiError(500, message, errors);
  }
}
```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    static serviceUnavailable(message: string = 'Сервіс недоступний', errors: any[] = []): ApiError {
        return new ApiError(503, message, errors);
    }
}

export default ApiError;

```

Проміжне ПЗ з обробки помилок є функцією, яка за сигнатурою сприймається веб-фреймворком Express як відповідний middleware з обробки помилок [15]. Дана функція використовується останньою після застосування решти проміжного ПЗ або поза чергою за виникнення та передачі помилки.

Лістинг коду файлу ErrorHandlerMiddleware.ts:

```

import type { Request, Response, NextFunction } from 'express';
import ApiError from "../errors/ApiError";

// error handler is last middleware, hence next() is not used
// @ts-ignore
const errorHandler = function (err: Error | ApiError, req: Request, res: Response, next: NextFunction): Response {
    let status = (err as ApiError).status || 500;
    let errors = (err as ApiError).errors || [];

    let env = process.env.NODE_ENV || 'development';
    if (env.toLowerCase() === 'development') console.error(err);
    return res.status(status).json({ name: err.name, message: err.message,
        errors: errors, stack: env.toLowerCase() === 'development' ? err.stack : undefined });
}

export default errorHandler;

```

Проміжне ПЗ для перевірки авторизації схожі між собою за головною концепцією попередньої обробки запитів. Зі заголовків запиту отримуються дані атрибуту Authorization, який має значення «Bearer TOKEN», де TOKEN – згенерований користувачу JWT. Розшифровка даного токена дозволяє отримати дані про користувача чи виявити спробу підробки.

Для ефективної роботи та полегшення розробки, використання middleware з перевірки авторизації до існуючого типу даних Request з модулю типів для Express було додано атрибут user, у який записуються отримані з токена дані про користувача. У контролерах даний атрибут використовується в подальшому для перевірки власності створених даних, у той час як власне проміжне ПЗ надає помилку у випадку відсутності токена.

Лістинг коду файлу AuthMiddleware.ts:

```

import { NextFunction, Request, Response } from 'express';
import ApiError from "../errors/ApiError";
import jwt from "jsonwebtoken";

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

// @ts-ignore
// res is not used
const auth = (req: Request, res: Response, next: NextFunction) => {
  if (req.method === 'OPTIONS') {
    next();
  }

  try {
    const token = req.headers.authorization?.split(' ')[1]; // Bearer TOKEN

    if (!token) {
      return next(ApiError.unauthorized('Користувач не авторизований'));
    }

    req.user = jwt.verify(token, process.env.SECRET_KEY as string);
    next();
  }
  catch (e: unknown) {
    return next(ApiError.unauthorized('Немає доступу'));
  }
}

export default auth;

```

Проміжне ПЗ перевірки ролей є методом, що використовує паттерн «Фабрика» для створення необхідної функції перевірки згідно отриманих даних про ролі. Отримання та читання токену використовується для порівняння даних про роль власника токену з наданими ролями, яким доступний доступ до запитаного маршруту.

Лістинг коду файлу CheckRoleMiddleware.ts:

```

import type {NextFunction, Request, Response} from 'express';
import ApiError from "../errors/ApiError";
import jwt from "jsonwebtoken";

const checkRoles = (roles: string[]) => {
  // @ts-ignore
  // res is not used
  return (req: Request, res: Response, next: NextFunction) => {
    if (req.method === 'OPTIONS') {
      next();
    }

    try {
      const token = req.headers.authorization?.split(' ')[1]; // Bearer TOKEN

      if (!token) {
        return next(ApiError.unauthorized('Користувач не авторизований'));
      }

      const decoded = jwt.verify(token, process.env.SECRET_KEY as string) as
      jwt.JwtPayload;
      if (!roles.includes(decoded.role)) return next(ApiError.forbidden('Немає
      доступу'));
      req.user = decoded;

      next();
    }
    catch (e: unknown) {
      return next(ApiError.unauthorized('Немає доступу'));
    }
  }
}

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

```
const checkWorker = checkRoles(['ADMIN', 'MODERATOR']);
const checkAdmin = checkRoles(['ADMIN']);

export default checkRoles;
export {
  checkRoles,
  checkWorker,
  checkAdmin
}
```

2.4. Реалізація клієнтської частини веб-додатку

Клієнтський додаток розроблений з використанням бібліотеки React на мові програмування TypeScript та використовує сторонню бібліотеку компонентів з відкритим кодом Material UI. В розробці клієнтського веб-додатку було використано інструмент статичного аналізу коду для виявлення проблемних шаблонів ESLint разом з форматувальником коду Prettier. Оплата товарів на клієнтській частині відбувається завдяки платіжній системі LiqPay та з використанням бібліотеки react-liqpay. Для роботи бібліотеки необхідне використання пакетів web3.js та ether.js, доступ до яких відкривається завдяки перевизначенню конфігурацій [16] та встановленню пакетів завдяки команді `install --legacy-peer-deps` менеджера NPM. Збереження зображень відбувається на хмарному сервісі Cloudinary.

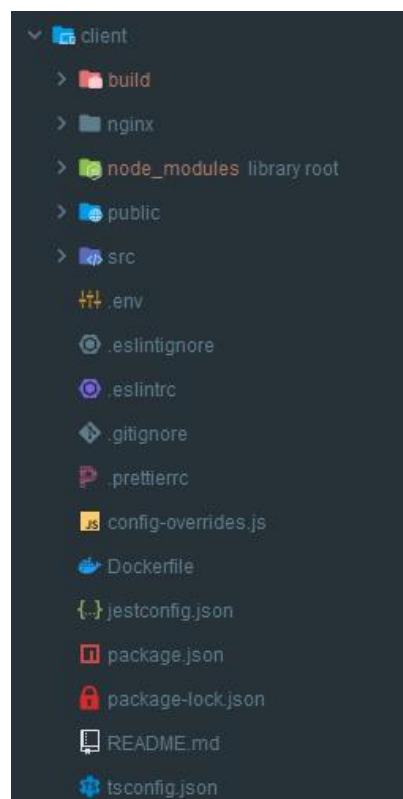


Рис.2.15 – Структура файлів клієнтського додатку

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

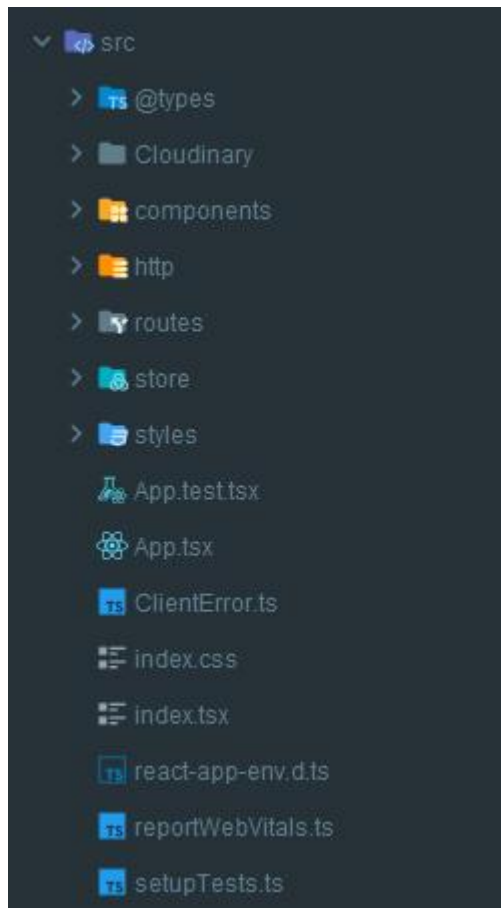


Рис.2.16 – Структура папки з вихідним кодом

Клієнтський додаток, подібно до серверного, використовує налаштування транспіляції та перевірки коду TypeScript, а також налаштування ESLint та Prettier, перевизначення використання пакетів web3.js та ether.js.

Лістинг файлу tsconfig.json клієнтського додатку (сховані коментарями налаштування видалено):

```
{
  "compilerOptions": {
    /* Visit https://aka.ms/tsconfig to read more about this file */

    /* Language and Environment */
    "target": "es6", /* Set the JavaScript language version
for emitted JavaScript and include compatible library declarations. */
    "lib": ["es6", "dom"], /* Specify a set of bundled
library declaration files that describe the target runtime environment. */
    "jsx": "react-jsx", /* Specify what JSX code is generated.
*/
    "jsxImportSource": "@emotion/react", /* Specify module
specifier used to import the JSX factory functions when using 'jsx: react-jsx*'. */

    /* Modules */
    "module": "commonjs", /* Specify what module code is
generated. */
    "rootDir": "./src", /* Specify the root folder within
your source files. */
    "moduleResolution": "node", /* Specify how TypeScript looks up a
```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

```

file from a given module specifier. */
  "typeRoots": [
    "node_modules/@types",
    "@types",
    "src/@types"
  ],
  /* Specify multiple folders that act like
  './node_modules/@types'. */

  /* Emit */
  "sourceMap": true,
  /* Create source map files for emitted
  JavaScript files. */
  "outDir": "./dist/esm",
  /* Specify an output folder for
  all emitted files. */
  "removeComments": true,
  /* Disable emitting comments. */
  "sourceRoot": "./src",
  /* Specify the root path for
  debuggers to find the reference source code. */
  "inlineSources": true,
  /* Include source code in the
  sourcemaps inside the emitted JavaScript. */
  "noEmitOnError": true,
  /* Disable emitting files if any type
  checking errors are reported. */

  /* Interop Constraints */
  "allowSyntheticDefaultImports": true,
  /* Allow 'import x from y' when a
  module doesn't have a default export. */
  "esModuleInterop": true,
  /* Emit additional JavaScript to
  ease support for importing CommonJS
  modules. This enables
  'allowSyntheticDefaultImports' for type compatibility. */
  "forceConsistentCasingInFileNames": true,
  /* Ensure that casing is correct
  in imports. */

  /* Type Checking */
  "strict": true,
  /* Enable all strict type-checking
  options. */
  "noImplicitAny": true,
  /* Enable error reporting for
  expressions and declarations with an implied 'any' type. */
  "strictNullChecks": true,
  /* When type checking, take into
  account 'null' and 'undefined'. */
  "noImplicitThis": true,
  /* Enable error reporting when 'this'
  is given the type 'any'. */
  "noUnusedLocals": true,
  /* Enable error reporting when local
  variables aren't read. */
  "noUnusedParameters": true,
  /* Raise an error when a function
  parameter isn't read. */
  "noImplicitReturns": true,
  /* Enable error reporting for
  codepaths that do not explicitly return in a function. */

  /* Completeness */
  "skipLibCheck": true
  /* Skip type checking all .d.ts
  files. */
},
"exclude": [
  "node_modules"
]
}

```

Лістинг файлу package.json:

```

{
  "name": "client",
  "version": "0.1.0",
  "main": "./dist/cjs/index.js",
  "module": "./dist/esm/index.js",
  "types": "./dist/esm/index.d.ts",
  "repository": {
    "type": "git",
    "url": "git+https://gitlab.com/2019-2023/ipz19-2/oleksiychuk-mykyta/bachelorgradproj_gigashop.git"
  },
  "license": "MIT",
  "private": true,
  "scripts": {

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

```

"start": "react-app-rewired start",
"build": "react-app-rewired build",
"start:build": "serve -s build",
"build:tsc": "npm run build:esm && npm run build:cjs",
"build:esm": "tsc",
"build:cjs": "tsc --module commonjs --outDir dist/cjs",
"eject": "react-scripts eject",
"lint": "eslint \"**/*.*\".{js,ts,jsx,tsx}\"",
"lint:fix": "eslint \"**/*.*\".{js,ts,jsx,tsx}\" --fix",
"prettier": "prettier --write \"src,tests,example/src\"/**/*.{js,ts,jsx,tsx}\"",
"prettify": "prettier --write \"src,tests,example/src\"/**/*.{js,ts,jsx,tsx}\"",
"test": "jest --config jestconfig.json",
"prepublishOnly": "npm test && npm run prettier && npm run lint"
},
"dependencies": {
"@cloudinary/react": "^1.11.2",
"@cloudinary/url-gen": "^1.10.1",
"@emotion/react": "^11.10.6",
"@emotion/styled": "^11.10.6",
"@hookform/resolvers": "^3.1.0",
"@mui/icons-material": "^5.11.16",
"@mui/material": "^5.12.1",
"@tinymce/tinymce-react": "^4.3.0",
"axios": "^1.3.6",
"dompurify": "^3.0.3",
"dotenv": "^16.0.3",
"html-react-parser": "^4.0.0",
"js-cookie": "^3.0.1",
"jwt-decode": "^3.1.2",
"npm": "9.5.0",
"react": "^18.2.0",
"react-dom": "^18.2.0",
"react-error-boundary": "^4.0.4",
"react-hook-form": "^7.43.9",
"react-liqpay": "^1.3.0",
"react-material-ui-carousel": "^3.4.2",
"react-router-dom": "^6.10.0",
"react-scripts": "^5.0.1",
"recoil": "^0.7.7",
"usehooks-ts": "^2.9.1",
"uuid": "^9.0.0",
"web-vitals": "^2.1.4",
"yup": "^1.1.1"
},
"devDependencies": {
"@mui/types": "^7.2.4",
"@testing-library/jest-dom": "^5.16.5",
"@testing-library/react": "^14.0.0",
"@testing-library/user-event": "^14.4.3",
"@types/dompurify": "^3.0.2",
"@types/jest": "^27.5.2",
"@types/js-cookie": "^3.0.3",
"@types/node": "^16.18.24",
"@types/react": "^18.0.37",
"@types/react-dom": "^18.0.11",
"@types/uuid": "^9.0.2",
"@typescript-eslint/eslint-plugin": "^5.59.0",
"@typescript-eslint/parser": "^5.59.0",
"assert": "^2.0.0",
"browserify-zlib": "^0.2.0",
"buffer": "^6.0.3",
"crypto-browserify": "^3.12.0",
"eslint": "^8.39.0",
"eslint-config-prettier": "^8.8.0",
"eslint-plugin-prettier": "^4.2.1",
"eslint-plugin-react": "^7.32.2",
"eslint-plugin-react-hooks": "^4.6.0",
"https-browserify": "^1.0.0",
"jest": "^29.5.0",
"jest-canvas-mock": "^2.5.0",
"jest-environment-jsdom": "^29.5.0",

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    "nodemon": "^2.0.22",
    "os-browserify": "^0.3.0",
    "prettier": "^2.8.8",
    "process": "^0.11.10",
    "react-app-rewired": "^2.2.1",
    "stream-browserify": "^3.0.0",
    "stream-http": "^3.2.0",
    "ts-jest": "^29.1.0",
    "ts-node": "^10.9.1",
    "typescript": "^4.9.5",
    "url": "^0.11.0"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}

```

Лістинг файлу .eslintrc:

```

{
  "root": true,
  "extends": [
    "plugin:prettier/recommended",
    "eslint:recommended",
    "plugin:react/recommended",
    "plugin:@typescript-eslint/eslint-recommended",
    "plugin:@typescript-eslint/recommended",
    "prettier"
  ],
  "rules": {
    "react/react-in-jsx-scope": "off",
    "react-hooks/rules-of-hooks": "error",
    "react-hooks/exhaustive-deps": "warn",
    "@typescript-eslint/no-non-null-assertion": "off",
    "@typescript-eslint/ban-ts-comment": "off",
    "@typescript-eslint/no-explicit-any": "off",
    "react/jsx-filename-extension": [
      1,
      {
        "extensions": [".jsx", ".tsx"]
      }
    ]
  },
  "parser": "@typescript-eslint/parser",
  "plugins": [
    "@typescript-eslint",
    "react",
    "react-hooks",
    "prettier"
  ],
  "settings": {
    "react": {
      "version": "detect"
    }
  },
  "env": {

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    "browser": true,
    "node": true
  },
  "globals": {
    "JSX": true
  }
}

```

Лістинг файлу .prettierrc:

```

{
  "bracketSpacing": true,
  "trailingComma": "es5",
  "tabWidth": 4,
  "semi": true,
  "printWidth": 120,
  "jsxSingleQuote": true,
  "endOfLine": "auto",
  "useTabs": true
}

```

Лістинг коду файлу config-overrides.js:

```

const webpack = require("webpack");

module.exports = function override(config) {
  const fallback = config.resolve.fallback || {};
  Object.assign(fallback, {
    crypto: require.resolve("crypto-browserify"),
    stream: require.resolve("stream-browserify"),
    assert: require.resolve("assert"),
    http: require.resolve("stream-http"),
    https: require.resolve("https-browserify"),
    os: require.resolve("os-browserify"),
    url: require.resolve("url"),
    zlib: require.resolve("browserify-zlib"),
  });
  config.resolve.fallback = fallback;
  config.plugins = (config.plugins || []).concat([
    new webpack.ProvidePlugin({
      process: "process/browser",
      Buffer: ["buffer", "Buffer"],
    }),
  ]);
  config.ignoreWarnings = [/Failed to parse source map/];
  config.module.rules.push({
    test: /\.?(js|mjs|jsx)$/,
    enforce: "pre",
    loader: require.resolve("source-map-loader"),
    resolve: {
      fullySpecified: false,
    },
  });
  return config;
};

```

Розгортання клієнтського додатку відбувається за сценарієм, описаним у локальному Dockerfile, що містить 2 стадії розгортання [13]. Перша стадія виконує повну інсталяцію пакетів та будівлення готового додатку з використанням пакету перевизначених функцій react-app-rewired для підтримки пакетів web3.js та ether.js [16], друга стадія налаштовує новий контейнер на основі образу nginx через файл nginx/nginx.conf та копіює збудований код з папки build [17].

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг файлу nginx.conf:

```
server {
    listen 3000;

    location / {
        root    /usr/share/nginx/html;
        index   index.html index.htm;
        try_files $uri /index.html;
    }

    location /api {
        proxy_pass http://server:5000;
    }
}
```

Лістинг файлу Dockerfile клієнтського додатку:

```
FROM node:18.9.0-alpine3.16 as development
ARG NODE_ENV=development
ENV NODE_ENV=${NODE_ENV}
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install --legacy-peer-deps
COPY . .
RUN npm run build

FROM nginx:latest as production
ARG NODE_ENV=production
ENV NODE_ENV=${NODE_ENV}
COPY --from=development /usr/src/app/build /usr/share/nginx/html
COPY nginx/nginx.conf /etc/nginx/conf.d/default.conf
WORKDIR /usr/src/app
CMD ["nginx", "-g", "daemon off;"]
```

Клієнтський додаток використовує бібліотеку керування станами Recoil для збереження та швидкого доступу до даних користувача, розшифрованих з наданого при вході чи реєстрації JWT [10]. Налаштування використання бібліотеки відбувається у виконавчому файлі index.tsx.

Лістинг коду файлу index.tsx:

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import { RecoilRoot } from "recoil";
import "./index.css";

const root = ReactDOM.createRoot(document.getElementById("root") as HTMLElement);
root.render(
  <React.StrictMode>
    <RecoilRoot>
      <App />
    </RecoilRoot>
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

Кореневий компонент App налаштовує маршрутизацію клієнтської частини та виконує одноразовий пошук токена для автоматичного встановлення глобального стану користувача.

Лістинг коду файлу App.tsx:

```
import { useRecoilState } from "recoil";
import Router from "../routes";
import Cookies from "js-cookie";
import jwt_decode, { JwtPayload } from "jwt-decode";
import { LogIn, LogOut, UserAtom, userState } from "../store/User";
import { useEffectOnce } from "usehooks-ts";

function App() {
  const [_ , setUser] = useRecoilState(userState);
  const token = Cookies.get("token");

  useEffectOnce(() => {
    if (token) {
      const decoded = jwt_decode<JwtPayload & UserAtom>(token);

      if (decoded.exp && decoded.exp < Date.now() / 1000) {
        LogOut(setUser);
      } else if (decoded.id && decoded.login && decoded.email && decoded.role && decoded.image) {
        LogIn(setUser, token);
      }
    }
  });

  return <Router />;
}

export default App;
```

У розробці клієнтського додатку на мові TypeScript для пакетів оплати та завантаження зображень і для стилізації веб-сайту за створеним прототипом дизайну було створено папку @types з файлами оголошення змін та доповнень до оригінальних типів [18] та/або створень необхідних типів. Дані файли будуть описані разом з функціоналом, до яких їх було написано.

Маршрутизація веб-сайту виконується завдяки використанню функціоналу пакету react-router-dom. Протягом маршрутизації використовуються розроблені компоненти шляхів та методи централізованих запитів до даних зі сервера, що застосовуються разом з функціями-завантажувачів даних.

Повний код файлу routes/index.tsx наведено у додатку Д.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

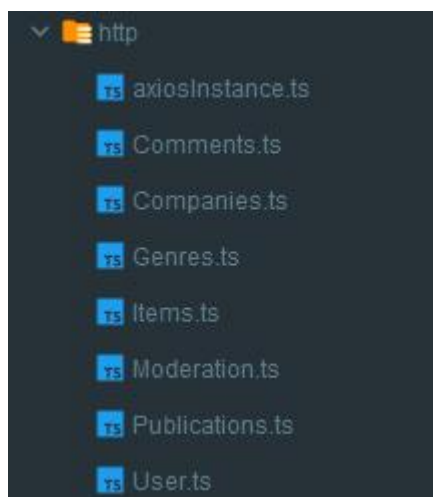
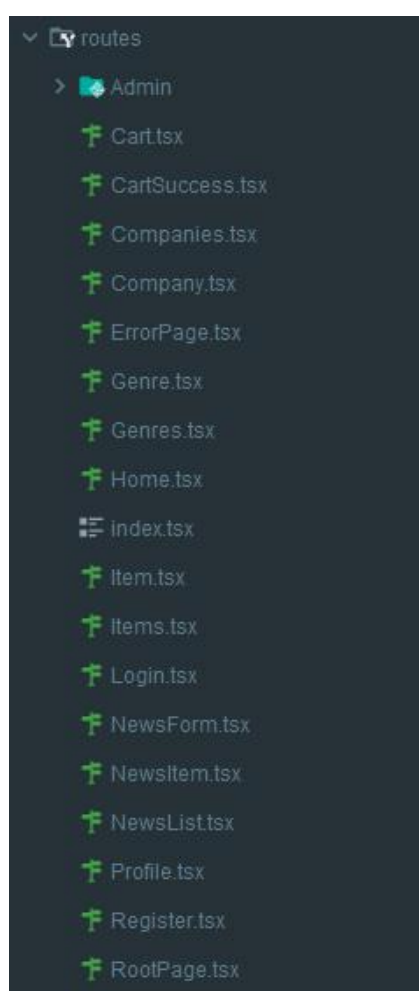
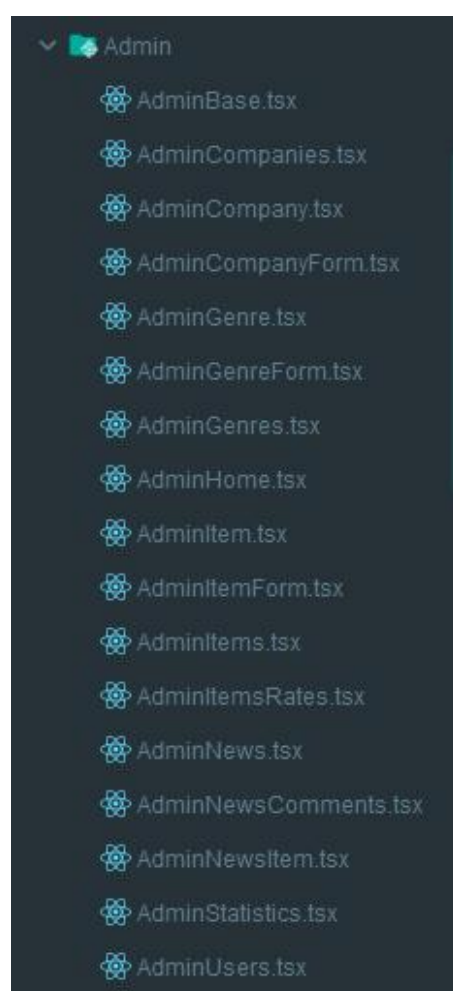


Рис.2.17 – Файлова структура папки НТТР запитів



А) Повний вміст папки



Б) Вміст папки шляхів адміністративної панелі

Рис.2.18 – Файлова структура папки routes

Код файлу index.tsx з налаштуваннями маршрутизації наведено у додатку Г.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

HTTP запити до сервера виконуються з використанням бібліотеки axios та налаштуванням спільного шляху надсилення запитів. Для даного налаштування використовується змінна середовища з посиланням на сервер для спрощення даного етапу розробки та можливого перенесення сервісу до хмарного сервісу.

Лістинг коду файлу axiosInstance.ts:

```
import axios from "axios";

const baseURL = process.env.REACT_APP_API_URL || "http://localhost:5000/api";
const axiosInstance = axios.create({
  baseURL,
});

export default axiosInstance;
```

Лістинг коду файлів виконання запитів наведено у додатку Ж.

Налаштування глобального стану користувача та методи встановлення JWT у cookies-файлах веб-сайту описані у файлі store/User.ts. Для встановлення чи видалення даних користувача за входу/реєстрації чи виходу відповідно необхідно ініціалізувати в компонентах та передавати set-метод пакету керування станами Recoil через неможливість їх ініціалізації глибше, ніж основні компоненти.

Лістинг коду файлу store/User.ts:

```
import { atom, selector, SetterOrUpdater } from "recoil";
import Cookies from "js-cookie";
import jwt_decode, { JwtPayload } from "jwt-decode";

export type UserAtom = {
  id: number;
  login: string;
  email: string;
  role: string;
  image: string;
};

export const userState = atom<UserAtom | undefined>({
  key: "userState",
  default: undefined,
});

export const getUser = selector({
  key: "getUser",
  get: ({ get }) => {
    const user = get(userState);
    if (user) return user;

    const token = Cookies.get("token");
    if (!token) return undefined;

    const decoded = jwt_decode<JwtPayload & UserAtom>(token);
    if (decoded.exp && decoded.exp < Date.now() / 1000) {
      Cookies.remove("token");
    }
  },
});
```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return undefined;
    }

    const data: UserAtom = {
        id: decoded.id,
        login: decoded.login,
        email: decoded.email,
        role: decoded.role,
        image: decoded.image,
    };

    return data;
},
});

export const LogOut = (setUser: SetterOrUpdater<UserAtom | undefined>) => {
    Cookies.remove("token");
    setUser(undefined);

    return true;
};

export const LogIn = (setUser: SetterOrUpdater<UserAtom | undefined>, token: string)
=> {
    const decoded = jwt_decode<JwtPayload & UserAtom>(token);
    if (decoded.exp && decoded.exp < Date.now() / 1000) throw new Error("Token expired");

    Cookies.set("token", token);
    setUser({
        id: decoded.id,
        login: decoded.login,
        email: decoded.email,
        role: decoded.role,
        image: decoded.image,
    });

    return true;
};

```

Для стилізації веб-сайту за створеним прототипом дизайну було налаштовано основні стилі у файлі `styles/index.ts`. Даний файл використовує бібліотеку Material UI та доповненими типами інтерфейсів Theme, Palette, PaletteOptions тощо, що оголошені у файлі `@types/mui.material.d.ts` [18]. Налаштування типів дозволяє оголосити використання кольорів акцентів (помаранчевий #ef4722) у різних компонентах. Стилi встановлюють використання раніше окреслених палітри кольорів та шрифтів текстових блоків компонентів Material UI. Шрифти підключені у файлі `index.html`, що знаходиться у кореневій теці `public`.

Лістинг коду файлу `@types/mui.material.d.ts`:

```

import "@mui/material/styles";
import {
    PaletteColorOptions,
    Palette as BasePalette,
    PaletteOptions as BasePaletteOptions,
    SimplePaletteColorOptions as BaseSimplePaletteColorOptions,
    PropTypes as BasePropTypes,
    Theme as BaseTheme,
} from "@mui/material";

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

declare module "@mui/material/styles" {
  export namespace PropTypes {
    type Color = BasePropTypes.Color | "accent" | "transparent";
  }
  interface Theme extends BaseTheme {
    fonts: {
      body?: string;
      heading?: string;
    };
    colors: {
      primary: string;
      secondary: string;
      accent: string;
      accentLight?: string;
      accentLighter?: string;
      inputPlaceholder?: string;
      inputBackground?: string;
    };
  }
  // allow configuration using `createTheme`
  interface ThemeOptions {
    fonts?: {
      body?: string;
      heading?: string;
    };
    colors?: {
      primary?: string;
      secondary?: string;
      accent?: string;
      accentLight?: string;
      accentLighter?: string;
      inputPlaceholder?: string;
      inputBackground?: string;
    };
  }
  interface SimplePaletteColorOptions extends BaseSimplePaletteColorOptions {
    lighter?: string;
  }
  interface Palette extends BasePalette {
    accent: PaletteColorOptions;
    inputs?: {
      background?: string;
      color?: string;
    };
  }
  interface PaletteOptions extends BasePaletteOptions {
    accent: PaletteColorOptions;
    inputs?: {
      background?: string;
      color?: string;
    };
  }
}

```

Лістинг коду файлу styles/index.ts:

```

import { createTheme } from "@mui/material";

const Black = "#272727";
const White = "#f0ead9";
const Orange = "#ef4722";
const OrangeLight = "#ef7d5f";
const OrangeLighter = "#efb39c";
const InputPlaceholder = "#505050";
const InputBackground = "#D9D9D9";
// const Error = "#ff0000";
// const Warning = "#ffbf00";
// const Info = "#00bfff";
// const Success = "#00ff00";

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

export const BodyFont = "Merriweather, sans-serif";
export const HeadingFont = "Montserrat Alternates, serif";

const BaseTheme = {
  fonts: {
    body: BodyFont,
    heading: HeadingFont,
  },
  colors: {
    inputPlaceholder: InputPlaceholder,
    inputBackground: InputBackground,
  },
  typography: {
    h1: {
      fontFamily: HeadingFont,
      color: Black,
    },
    h2: {
      fontFamily: HeadingFont,
      color: Black,
    },
    h3: {
      fontFamily: HeadingFont,
      color: Black,
    },
    h4: {
      fontFamily: HeadingFont,
      color: Black,
    },
    h5: {
      fontFamily: HeadingFont,
      color: Black,
    },
    h6: {
      fontFamily: HeadingFont,
      color: Black,
    },
    subtitle1: {
      fontFamily: BodyFont,
      color: Black,
    },
    subtitle2: {
      fontFamily: BodyFont,
      color: Black,
    },
    body1: {
      fontFamily: BodyFont,
      color: Black,
    },
    body2: {
      fontFamily: BodyFont,
      color: Black,
    },
    caption: {
      fontFamily: BodyFont,
      color: Black,
    },
    button: {
      fontFamily: BodyFont,
      color: Black,
    },
    overline: {
      fontFamily: BodyFont,
      color: Black,
    },
  },
};

export const RegularTheme = createTheme({
  ...BaseTheme,
  palette: {
    primary: {

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        main: Black,
        contrastText: White,
    },
    secondary: {
        main: White,
        contrastText: Black,
    },
    accent: {
        main: Orange,
        light: OrangeLight,
        lighter: OrangeLighter,
    },
    background: {
        default: White,
    },
    inputs: {
        background: InputBackground,
        color: InputPlaceholder,
    },
    },
    colors: {
        ...BaseTheme.colors,
        primary: Black,
        secondary: White,
        accent: Orange,
        accentLight: OrangeLight,
        accentLighter: OrangeLighter,
    },
});
export const AdminTheme = RegularTheme;

```

На кінцевому етапі тема стилів адміністративної панелі не відрізняється від звичайної, у майбутньому можливе створення окремої палітри кольорів для окремої теми.

Для збереження зображень до хмарного сервісу Cloudinary до файлу index.html було додано підключення скрипту від сервісу та доповнено у файлі @types/window.d.ts інтерфейс Window для зберігання екземплярів віджетів Cloudinary [18]. Дані віджети налаштовуються у двох окремих файлах папки Cloudinary. WidgetMultiple дозволяє завантаження великої кількості зображень одночасно, у той час як WdidgetSingle дозволяє завантаження одного зображення з його обрізанням за необхідності. Обидва віджети використовують назви спільного хмарного середовища та попередніх налаштувань завантажень без використання ключів за типом зображень: користувацькі аватари, логотипи компаній чи описові зображення товарів.

Лістинг коду @types/window.d.ts:

```

declare global {
    interface Window {
        cloudinary: any;
    }
}

```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

```
window.cloudinary = window.cloudinary || {};
```

Лістинг коду файлу Cloudinary/WidgetMultiple.ts:

```
import { Dispatch, SetStateAction, useEffect, useRef } from "react";

export default function WidgetSingle(
  cloudName = "dnqlgypji",
  uploadPreset = "gigashop_general",
  setter?: Dispatch<SetStateAction<string[]>>
) {
  const cloudinaryRef = useRef<any>(null);
  const widgetRef = useRef<any>(null);

  useEffect(() => {
    // @ts-ignore
    cloudinaryRef.current = window.cloudinary;
    widgetRef.current = cloudinaryRef.current.createUploadWidget(
      {
        cloudName: cloudName,
        uploadPreset: uploadPreset,
        multiple: true,
        maxFiles: 10,
      },
      (error: any, result: any) => {
        if (!error && result && result.event === "success") {
          console.log("Done! Here is the image info: ", result.info);

          if (setter && result.info) {
            setter((prev) => [...prev, result.info.secure_url]);
          }
        }
      }
    );
  }, []);

  return { cloudinaryRef, widgetRef };
}
```

Лістинг коду файлу WidgetSingle.ts:

```
import { Dispatch, SetStateAction, useEffect, useRef } from "react";

export default function WidgetSingle(
  cloudName = "dnqlgypji",
  uploadPreset = "gigashop_general",
  setter?: Dispatch<SetStateAction<string | null>>
) {
  const cloudinaryRef = useRef<any>(null);
  const widgetRef = useRef<any>(null);

  useEffect(() => {
    // @ts-ignore
    cloudinaryRef.current = window.cloudinary;
    widgetRef.current = cloudinaryRef.current.createUploadWidget(
      {
        cloudName: cloudName,
        uploadPreset: uploadPreset,
        cropping: true,
        multiple: false,
        maxFiles: 1,
      },
      (error: any, result: any) => {
        if (!error && result && result.event === "success") {
          console.log("Done! Here is the image info: ", result.info);

          if (setter && result.info) {
            setter(result.info.secure_url);
          }
        }
      }
    );
  });
}
```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
  );
}, []);

return { cloudinaryRef, widgetRef };
}

```

Для роботи оплати кошика товарів через сервіс LiqPay було вирішено використати пакет компонентів react-liqpay. Його використання зумовлює встановлення пакетів, що замінюють web3.js та ether.js або доповнюють їх, та перевизначенню конфігурацій [16]. Пакет компонентів оплати не має типізації до TypeScript, через що було створено окремий файл з оголошенням типу необхідного компоненту LiqPayPay [19][20].

Лістинг коду файлу @types/react-liqpay.d.ts:

```

declare module "react-liqpay" {
  import { DetailedReactHTMLElement, InputHTMLAttributes } from "react";

  type LiqPayPayProps = {
    publicKey: string;
    privateKey: string;
    currency: string;
    amount: string;
    description: string;
    orderId: string;
    title?: string;
    style?: {};
    result_url?: string;
    server_url?: string;
    disabled?: boolean;
    extra?: {};
    product_description?: string;
  };
  export function LiqPayPay(
    props: LiqPayPayProps
  ): DetailedReactHTMLElement<InputHTMLAttributes<HTMLInputElement>, HTMLInputElement>;
}

```

Даний компонент отримує зі змінних середовища ключі для налаштування зв'язку зі сервісом; параметри оплати, такі як валюта (прийнятними значеннями є “UAH”, “USD”, “EUR”, “KZT”); кількість (вартість покупки); опис транзакції; ідентифікатор замовлення; підпис до кнопки оплати, що генерується; об'єкт стилів; посилання, на яке повернеться користувач в разі успіху; посилання, на яке прийде запит зі сторони сервера LiqPay; стан кнопки (доступна чи вимкнена); об'єкт компонентів, які відмалюються замість стандартної кнопки; опис товару.

При натисканні на кнопку відбувається формування необхідних хешованих даних, які нададуться клієнту LiqPay для формування сторінки оплати, яка з'являється перед користувачем.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

Висновки до другого розділу

У результаті виконаної роботи у другому розділі було розроблено модель програмного забезпечення на логічному рівні та описано фізичну модель, створено та зображено прототип дизайну клієнтської частини майбутнього програмного продукту, описано процес розробки компонентів системи веб-додатку.

Протягом розробки моделі програмного забезпечення було створено діаграму використання та описано можливі варіанти використання із розділення користувачів по ролям на акторів моделі; створено діаграму діяльності та описано алгоритми роботи зі системою для різних акторів, можливі стани протягом взаємодії акторів зі системою; створено та описано діаграму класів представлення даних у системі, яка стала основою бази даних; спроектовано діаграми послідовностей для опису взаємодії різних компонентів системи між собою в сценаріях виконання запитів двома групами акторів.

Було описано фізичну модель, в рамках якої було представлено діаграму компонентів системи, які беруть участь у взаємодії з користувачами, та діаграму розгортання, що описує майбутню структуру системи веб-додатку в рамках використаної технології розгортання додатку.

Протягом розробки серверної частини було засвоєно нові вміння та навички з розробки серверних додатків з використанням обраного стеку технологій та покращено розуміння роботи HTTP запитів, отримано досвід роботи з обраними технологіями.

Протягом розробки клієнтської частини було покращено існуючі та отримано нові вміння та навички зі створення веб-застосунків на основі бібліотеки React та технологій, пакетів, бібліотек, що розроблені для використання з даною бібліотекою UI, а також технології створення HTTP запитів.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ДОДАТКОМ МАГАЗИНУ ВІДЕОІГОР ТА ПОРТАЛУ ІГРОВИХ ПУБЛІКАЦІЙ

3.1. Порядок встановлення та налаштування параметрів веб-додатку

Клієнтська частина веб-додатку дозволяє зручно переглядати інформацію в системі у вигляді веб-сторінок, з чого випливає можливість використання системи за допомогою браузерів. Для використання сайту потрібен телефон, планшет, ноутбук чи стаціонарний комп'ютер, які відповідають наступним вимогам:

- операційна система Windows 7 і вище, MacOS 10.11 і вище або Android 6 і вище
- 2 ГБ оперативної пам'яті
- наявність графічного дисплею
- наявність веб-браузера

Веб-додаток використовує змінні середовища для зв'язку між компонентами системи, а саме: назва та порт хоста, ім'я та пароль користувача та назва БД, які сервер використовує для зв'язку з контейнером БД; посилання на сервер, яке клієнт передає до клієнту LiqPay для створення запиту про успішну транзакцію. Для зв'язування контейнерів у одну мережу та зручного запуску транспіляції коду застосунків системи, створення виробничих версій додатків та уніфікації місця введення необхідних даних, застосовується функція Compose з налаштуваннями системи, що записані у файл конфігурації YAML.

Для опису розгортання системи використовується діаграма розгортання. Даний тип діаграм є підвидом структурних діаграм та зображає налаштування та взаємозв'язок різних частин однієї системи та проміжних вузлів, їх налаштування та компонентів, які використовуються. Діаграми розгортання часто застосовуються для опису топології програмного забезпечення.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

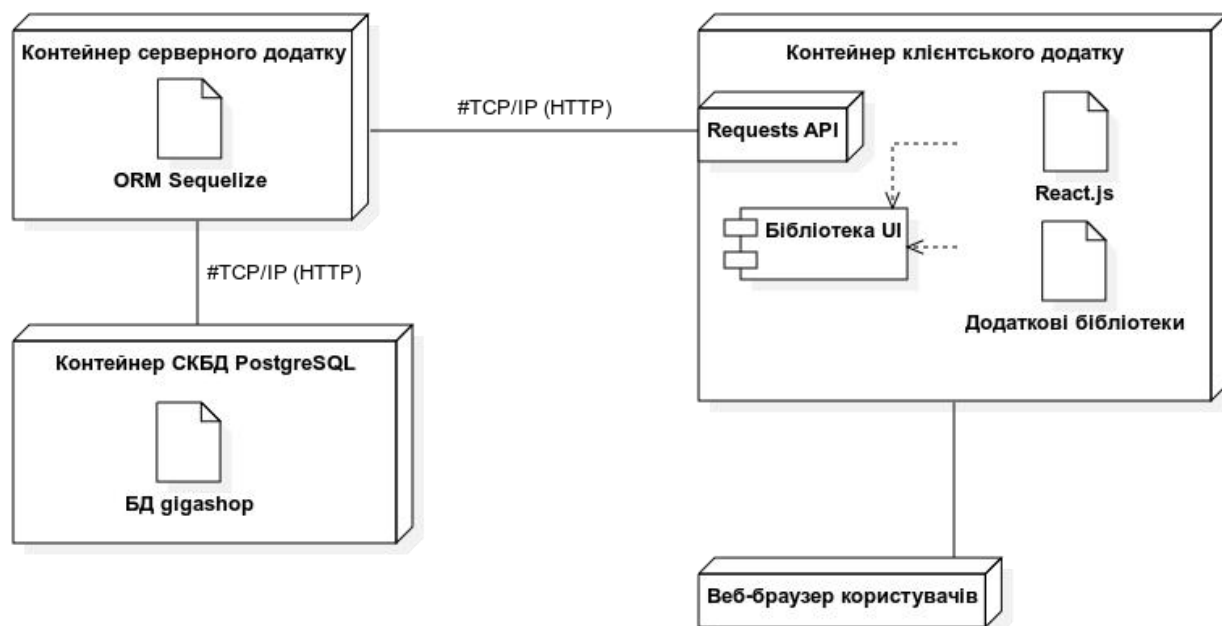


Рис.3.1 – Діаграма розгортання компонентів веб-додатку

Програмне забезпечення веб-додатку передбачає комплекс із 3х окремих Docker контейнерів, пов'язаних спільною мережею [21]. Контейнер клієнтського додатку у збудованому вигляді містить необхідний JavaScript функціонал, який разом з налаштуванням кінцевого nginx-контейнеру додатку створює та адресує запити на сервер по його назві хоста в мережі. Контейнер серверного додатку отримує запит, який проходить обробку в Express та завдяки ORM формує запит до БД в контейнері СКБД по налаштованому зв'язку. Результат обробки запита до БД за необхідності обробляється, результат роботи надається до клієнтського додатку для відображення користувачу.

Лістинг коду файлу docker-compose.yml (дані про посилання на сервер, секретні ключі видалені):

```

version: '1.0'
name: gigashop_prod
services:
  client:
    build:
      context: ./client
      target: production
    container_name: client
    volumes:
      - ./client:/usr/src/app
      - /usr/src/app/node_modules
    ports:
      - 3000:3000
    environment:
      REACT_APP_NGROK_SERVER_URL: https://f547-188-163-19-100.ngrok-free.app
      REACT_APP_NGROK_CLIENT_URL: http://localhost:3000
  
```

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    REACT_APP_API_URL: /api
  depends_on:
    - server

server:
  build:
    context: ./server
    target: production
  container_name: server
  volumes:
    - ./server:/usr/src/app
    - /usr/src/app/node_modules
  ports:
    - 5000:5000
  environment:
    DB_NAME: gigashop
    DB_USER: gigashop
    DB_PASSWORD: root
    DB_PORT: 5432
    DB_HOST: database
    CONNECT_TIMEOUT: 5000
    CONNECT_RETRIES: 10
    CLOUDINARY_SECRET: dxCEJghbMzdX0gtVpZUqn43eEOU
    LIQPAY_TEST_PUBLIC_KEY: sandbox_i27206268381
    LIQPAY_TEST_PRIVATE_KEY: sandbox_3DM1ZMsYJPj9NMgSa73gsG9HEN88pvk14APy9hfu
  depends_on:
    - database

database:
  container_name: database
  image: postgres
  ports:
    - 5001:5432
  environment:
    POSTGRES_DB: gigashop
    POSTGRES_USER: gigashop
    POSTGRES_PASSWORD: root
  volumes:
    - ./data:/var/lib/postgresql/data

```

Контейнер БД використовує для створення офіційний образ від PostgreSQL, у той час як розроблені серверний та клієнтський додатки – образи Node.js та nginx згідно конфігурації їх власних Dockerfile.

Для пришвидшення створення системи, для кожного з контейнерів розроблених додатків встановлюються об'єми пам'яті, які існують та заповнюються контейнерами під час роботи. Таким чином контейнер БД під час першого запуску створює ззовні папку data на кореневому рівні проекту та зберігає там результати роботи, а саме базу даних проекту з усією цілісною інформацією. Це дозволяє перезапускати систему та дороблювати її функціонал (не включаючи схему БД) без втрати створених екземплярів даних.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

При першому запуску конфігурації створюються образи майбутніх контейнерів на основі скрипту Dockerfile та з використанням наявного розробленого коду додатків системи.

```
[+] Running 0/14
- database Pulling 7.4s
- 5b5fe70539cd Downloading [=====] 24.05MB/29.12MB 4.2s
- 254f758dbb83 Download complete 4.2s
- a50b6f601ef0 Download complete 4.2s
- b2a96a17b000 Download complete 4.2s
- b08a8624844f Download complete 4.2s
- 9fb1f8c3deef Download complete 4.2s
- fb374f351a53 Verifying Checksum 4.2s
- 7b3d9f705bf0 Waiting 4.2s
- a85d8c4f987c Waiting 4.2s
- 0543b0f5d286 Waiting 4.2s
- 28bb5ab88d7a Waiting 4.2s
- fdb6f3d907fc Waiting 4.2s
- f06aa6280d55 Waiting 4.2s
```

Рис.3.2 – Початок завантаження даних для створення контейнерів

```
=> [gigashop_prod-client development 2/6] WORKDIR /usr/src/app 4.4s
=> [gigashop_prod-server production 3/5] COPY package*.json ./ 0.4s
=> [gigashop_prod-server development 4/6] RUN npm install 83.0s
=> [gigashop_prod-server production 4/5] RUN npm install --omit=dev 80.1s
=> [gigashop_prod-server development 5/6] COPY . . 0.4s
=> [gigashop_prod-server development 6/6] RUN npm run build 31.2s
=> [gigashop_prod-server production 5/5] COPY --from=development /usr/src/app/dist ./dist 0.8s
=> [gigashop_prod-server] exporting to image 6.2s
=> => exporting layers 6.1s
=> => writing image sha256:1768941a20405523817e6cb48601191a505103f4b16d077038553948f02e0cae 0.0s
=> => naming to docker.io/library/gigashop_prod-server 0.0s
```

Рис.3.3 – Встановлення пакетів та створення серверного додатку

```
=> [development 1/6] FROM docker.io/library/node:18.9.0-alpine3.16@sha256:831d5eca5b7437a8132031a25bd18bdb0399e7415d4e8 0.0s
=> CACHED [production 1/4] FROM docker.io/library/nginx:latest@sha256:593dac25b7733ffb7afe1a72649a43e574778bf025ad60514 0.0s
=> CACHED [development 2/6] WORKDIR /usr/src/app 0.0s
=> CACHED [development 3/6] COPY package*.json ./ 0.0s
=> CACHED [development 4/6] RUN npm install --legacy-peer-deps 0.0s
=> [development 5/6] COPY . . 225.9s
=> [development 6/6] RUN npm run build 313.3s
=> [production 2/4] COPY --from=development /usr/src/app/build /usr/share/nginx/html 0.6s
=> [production 3/4] COPY nginx/nginx.conf /etc/nginx/conf.d/default.conf 0.2s
=> [production 4/4] WORKDIR /usr/src/app 0.2s
=> exporting to image 0.9s
=> => exporting layers 0.8s
=> => writing image sha256:48d6ba3609c0763fbd5e54337544284bdaeddae13a54f71d6c247242cd3e2e8a 0.0s
=> => naming to docker.io/library/gigashop_prod-client 0.0s
```

Рис.3.4 – Встановлення пакетів та створення клієнтського додатку

[+] Running 4/4			
- Network	gigashop_prod_default	Created	1.3s
- Container	database	Started	35.0s
- Container	server	Started	34.1s
- Container	client	Started	12.3s

Рис.3.5 – Результат запуску конфігурації веб-додатку

3.2. Структура інтерфейсу та порядок роботи з веб-додатком магазину відеоігор

В результаті створення контейнерів у додатку Docker можна побачити створені образи контейнерів та отримати детальні відомості про них, а також переглянути активні екземпляри контейнерів та перевірити їх стан.

☐	Name	Tag	Status	Created	Size	Actions
☐	gigashop_prod-client 48d6ba3609c0	latest	In use	1 minute ago	197.39 MB	▶ ⋮ 🗑
☐	gigashop_prod-server 1768941a2040	latest	In use	23 minutes ago	207.39 MB	▶ ⋮ 🗑
☐	postgres 1921dda0e2c5	latest	In use	about 17 hours ago	411.83 MB	▶ ⋮ 🗑

Рис.3.6 – Образи, що використовуються в побудові веб-додатку

☐	Name	Image	Status	Port(s)	Last started	Actions
☒	gigashop_prod	-	Running (3/3)		8 minutes ago	■ ⋮ 🗑
☐	database 933da9085891	postgres	Running	5001:5432	8 minutes ago	■ ⋮ 🗑
☐	server c1a07232a438	gigashop_prod-server	Running	5000:5000	8 minutes ago	■ ⋮ 🗑
☐	client 8706367783c9	gigashop_prod-client	Running	3000:3000	8 minutes ago	■ ⋮ 🗑

Рис.3.7 – Результат побудови у вигляді контейнерів веб-додатку в роботі

З локальної машини маємо можливість переглянути веб-сайт та перевірити відповідність дизайну.

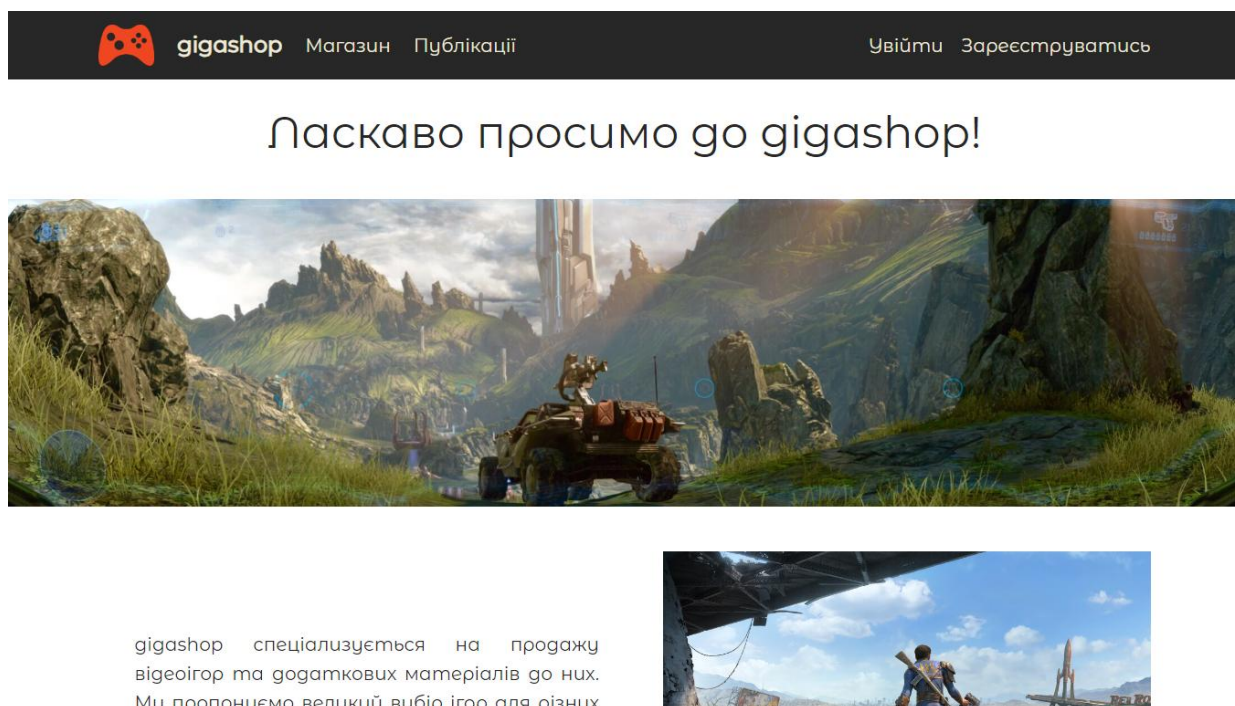


Рис.3.8 – Початкова сторінка веб-сайту

На початковій сторінці веб-сайту маємо вітальне повідомлення та звернення з коротким описом веб-додатку, дій, які доступні на даному сайті.

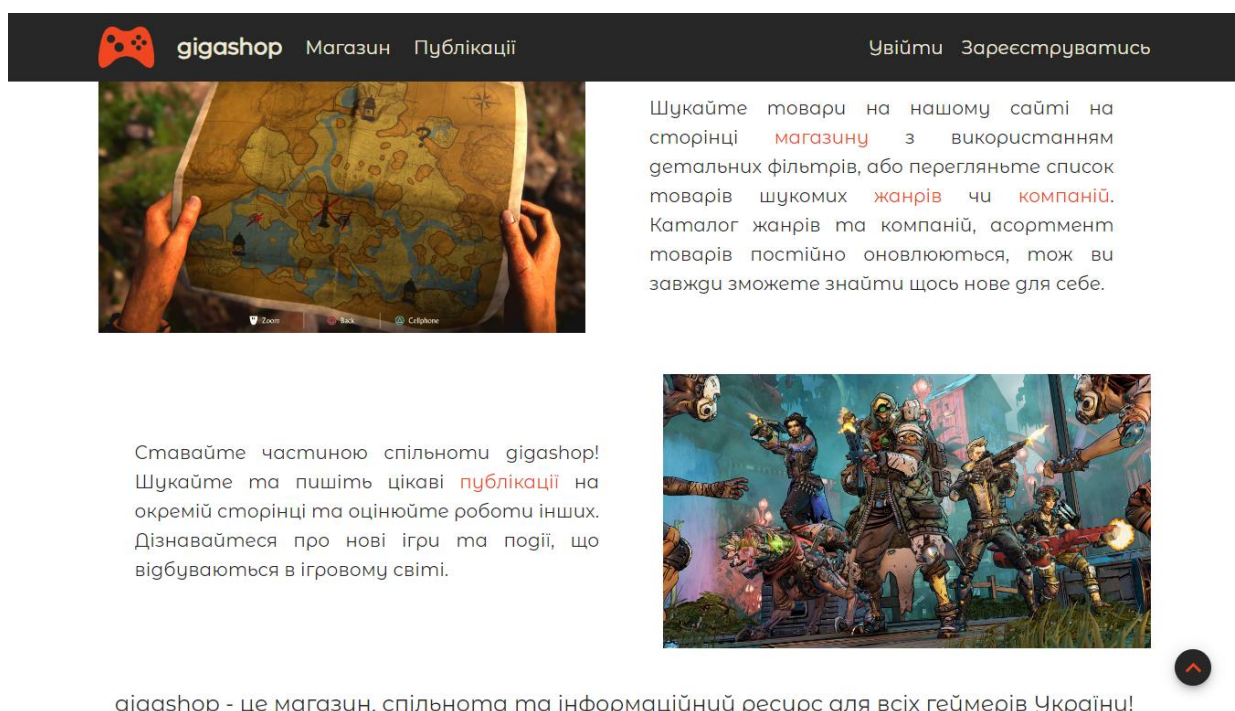


Рис.3.9 – Нижня частина веб-додатку з посиланнями на основні сторінки

При наведенні курсору на надпис “Магазин” з’являється меню з посиланнями до каталогу товарів, списку жанрів та пошуку компаній, які наразі є в системі.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

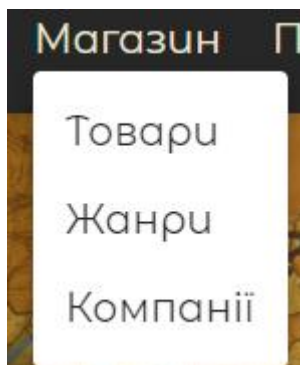


Рис.3.10 – Випадне меню з посиланнями магазину

Для зручності користування веб-сайтом з мобільних пристроїв, було забезпечено адаптивність доступу до панелі посилань хедеру сайту.



Рис.3.11 – Хедер веб-додатку в режимі малих екранів

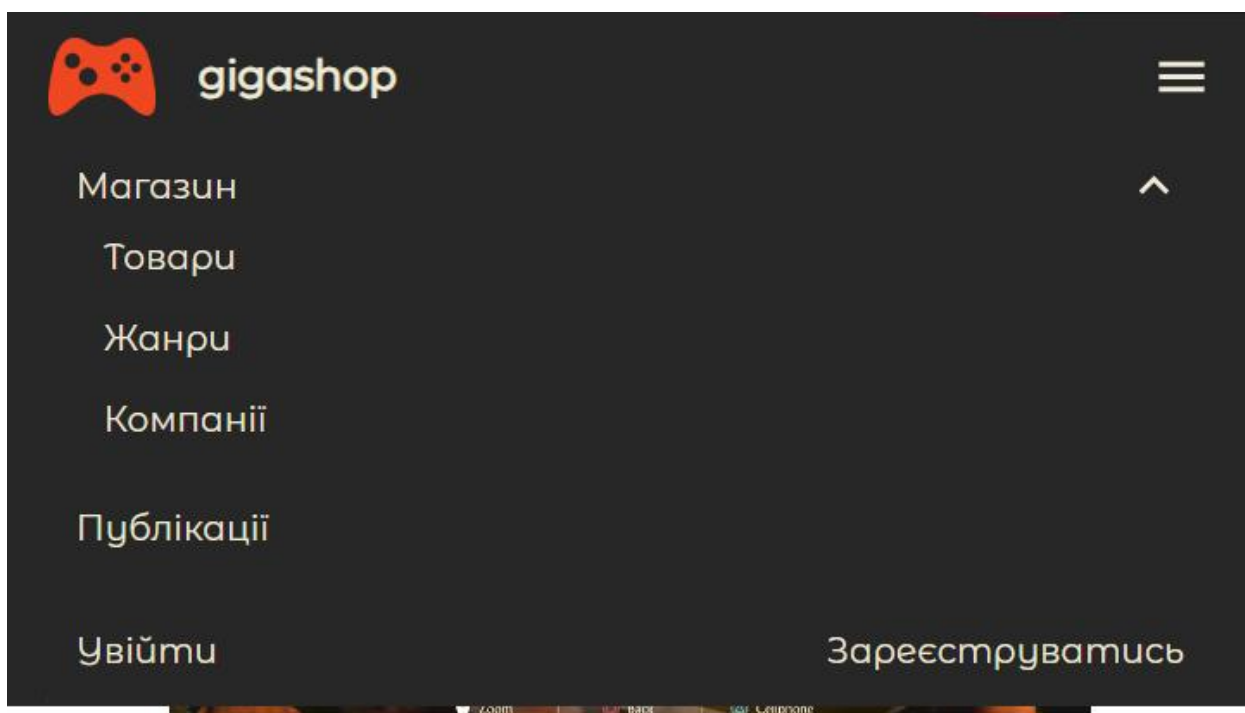


Рис.3.12 – Розгорнуте меню в режимі малих екранів

На сторінці товарів маємо можливість побачити сітку карток товарів з коротким описом та важливими деталями про ці товари, а також боковий компонент фільтрів пошуку та керування пагінацією, зменшенням чи збільшенням кількості зображуваних та шукомих елементів.

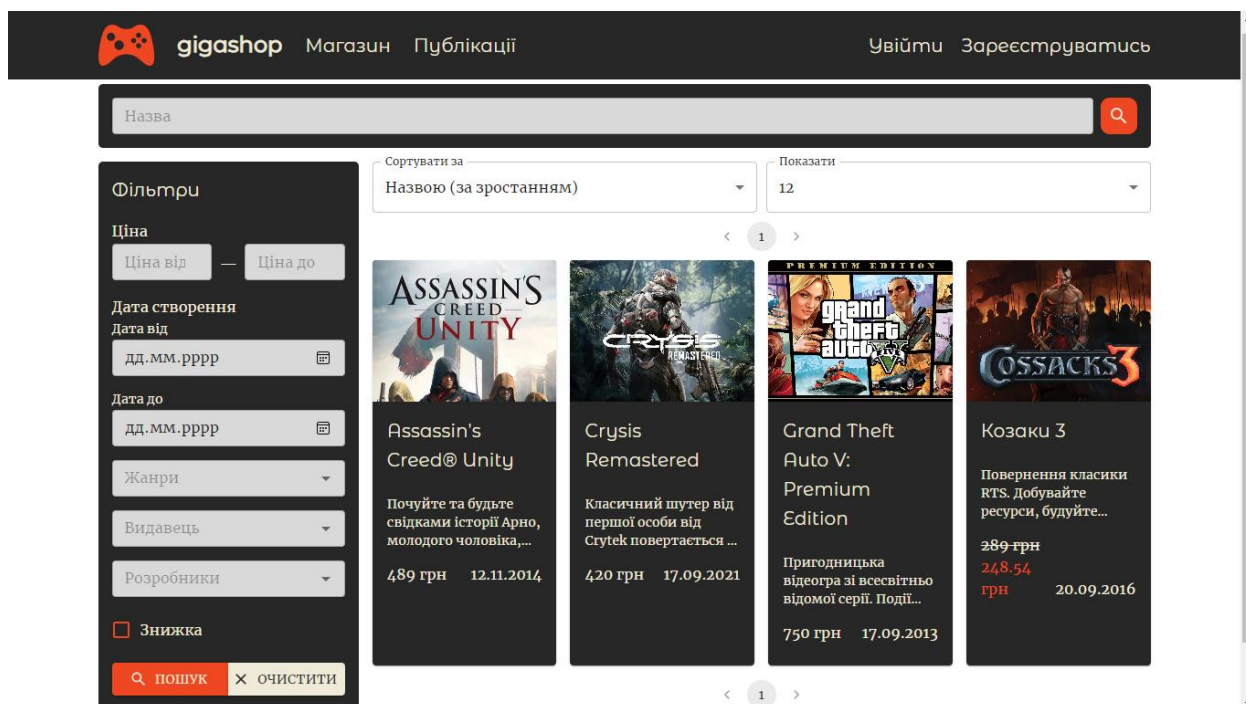


Рис.3.13 – Сторінка пошуку товарів

На сторінці жанрів можемо побачити сітку блоків жанрів, відсортованих за алфавітом. Разом зі сіткою бачимо поле пошуку за назвою жанрів.

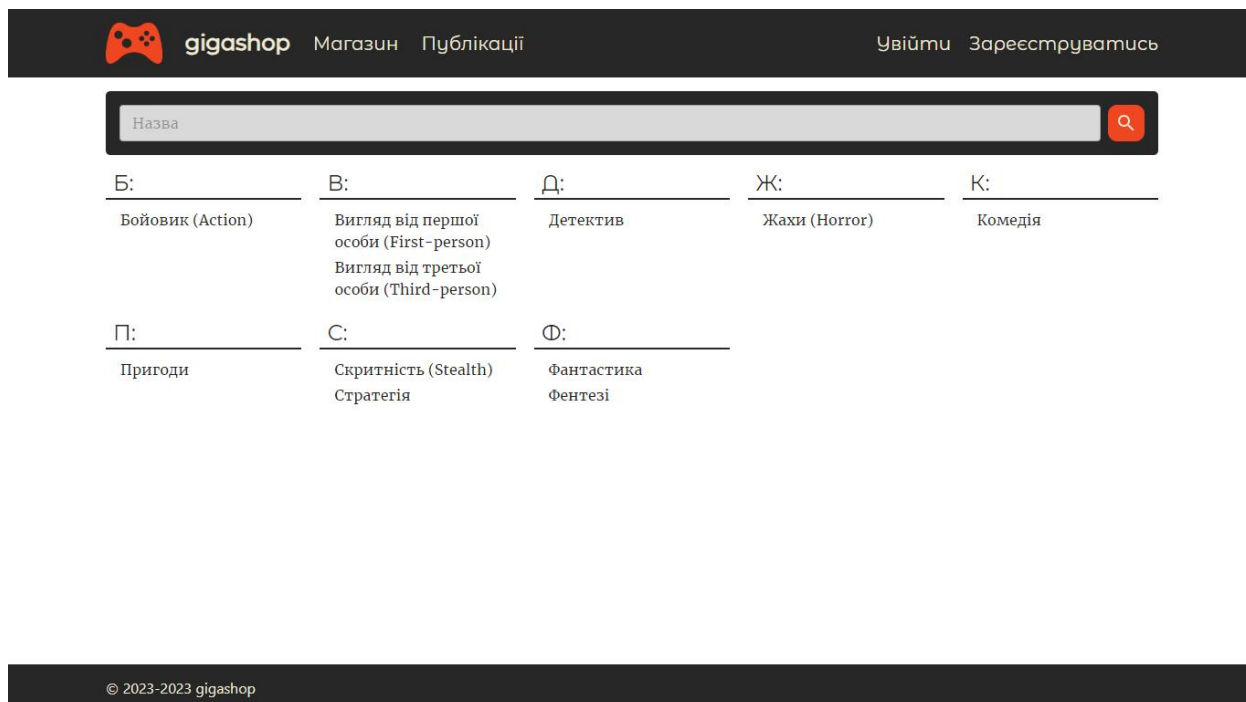


Рис.3.14 – Сторінка пошуку жанрів

На сторінці пошуку публікацій маємо, подібно до сторінки пошуку товарів, сітку з картками компаній та елементи керування сортуванням, пагінації та кількості зображуваних товарів.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

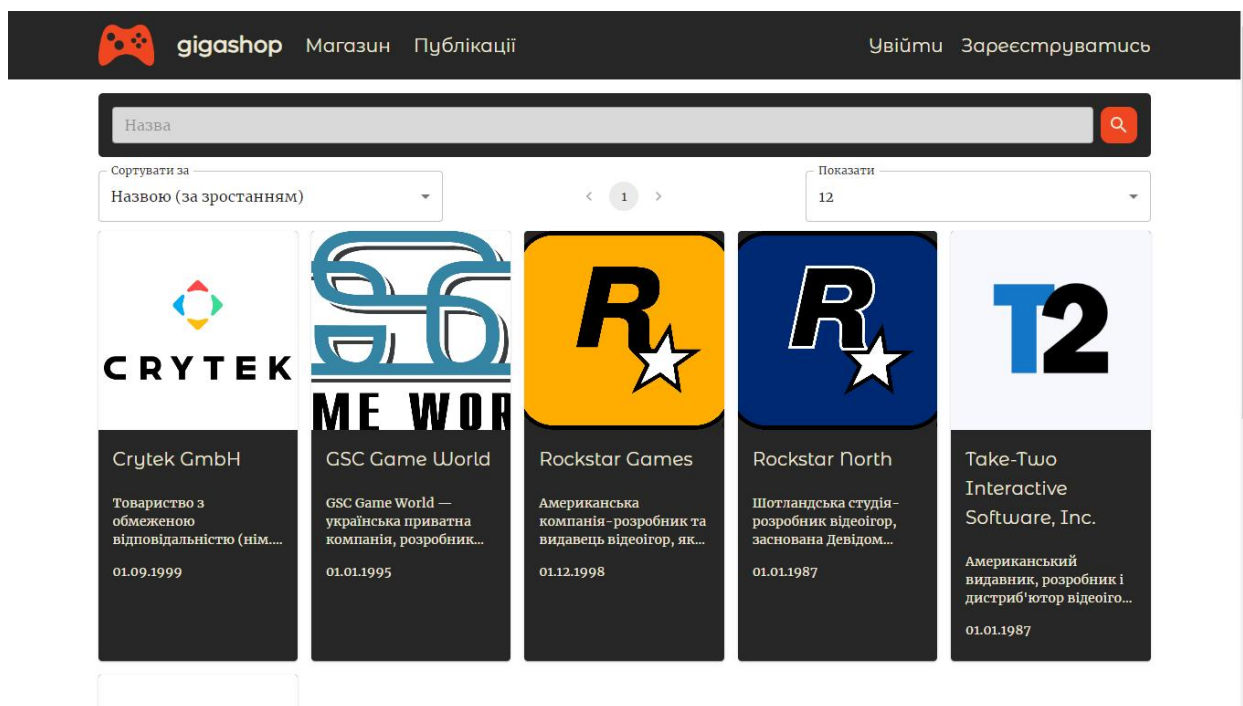


Рис.3.15 – Сторінка пошуку публікацій

При переході до сторінки товарів маємо можливість переглянути дані та характеристики товару, пов'язані з ним жанри та компанії тощо. Одне зі зображень товару під атрибутом `coverImage` використовується для прикрашання сторінки товару над ним, інші разом з головним зображенням використовуються в галереї на основі компоненти каруселі.



Рис.3.16 – Верхня частина сторінки окремого товару

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

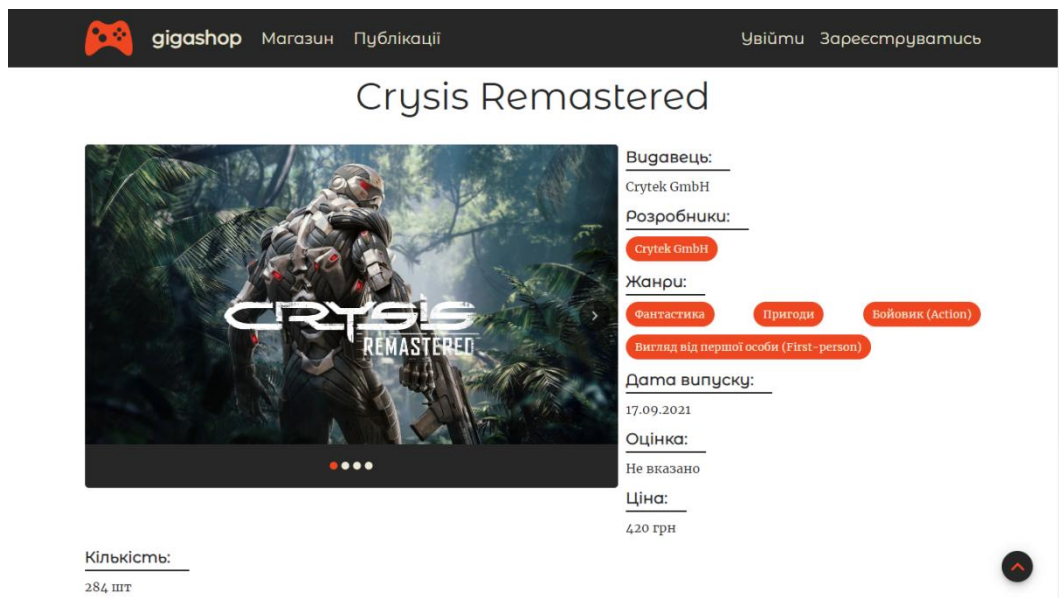


Рис.3.17 – Головна частина сторінки окремого товару

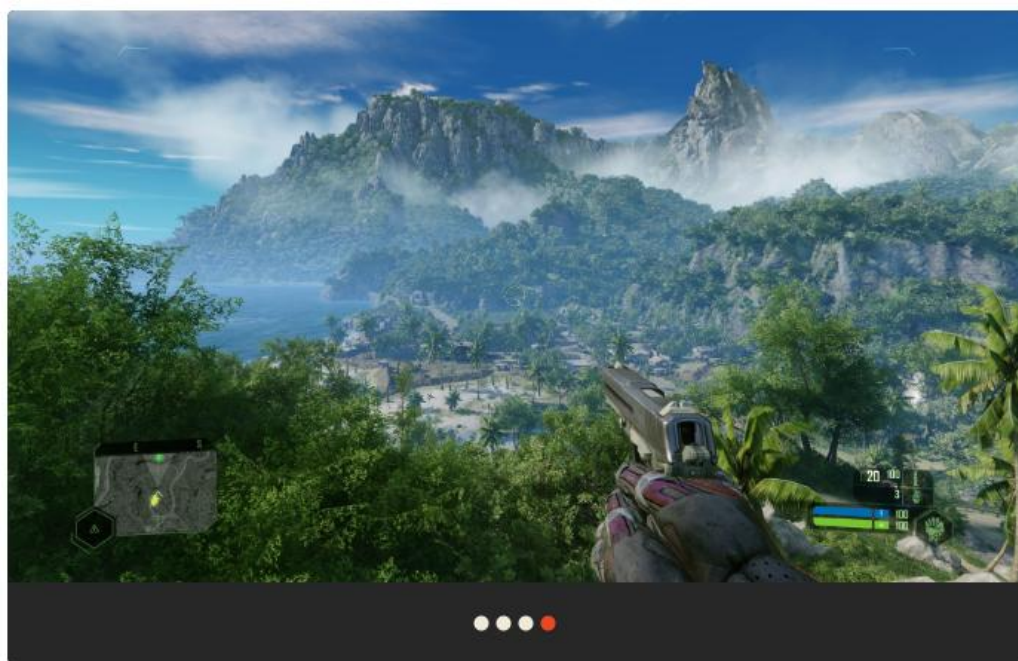


Рис.3.18 – Зміна зображень галереї

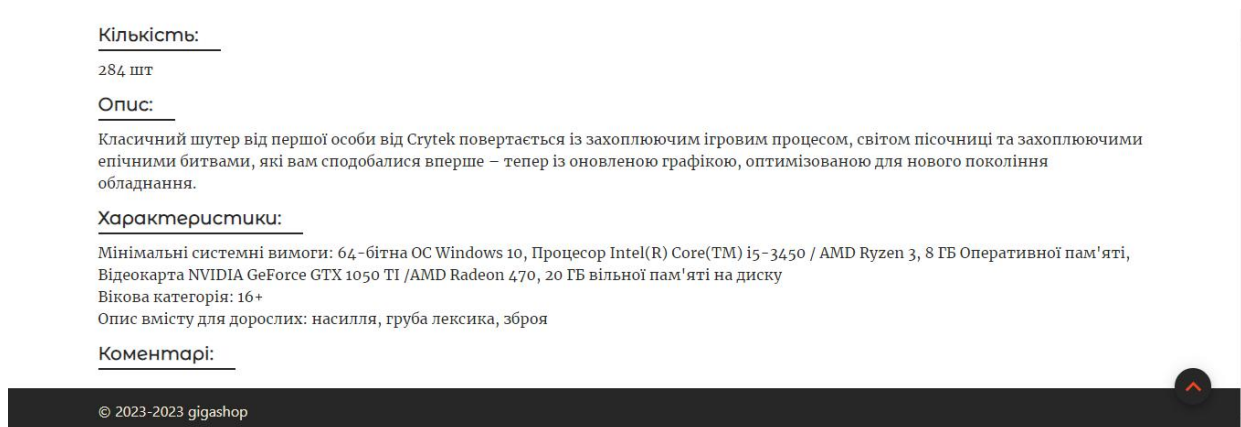


Рис.3.19 – Нижня частина сторінки окремого товару

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

Зі сторінки товарів при натисканні на елементи з назвами жанрів та компаній чи з навігаційного меню в хедері маємо можливість переглянути деталі про ті чи інші жанри чи компанії.

Сторінка жанрів містить назву, опис жанру та сітку товарів виключно цього жанру зі звичними елементами керування сортуванням, пагінацією та лімітом товарів.

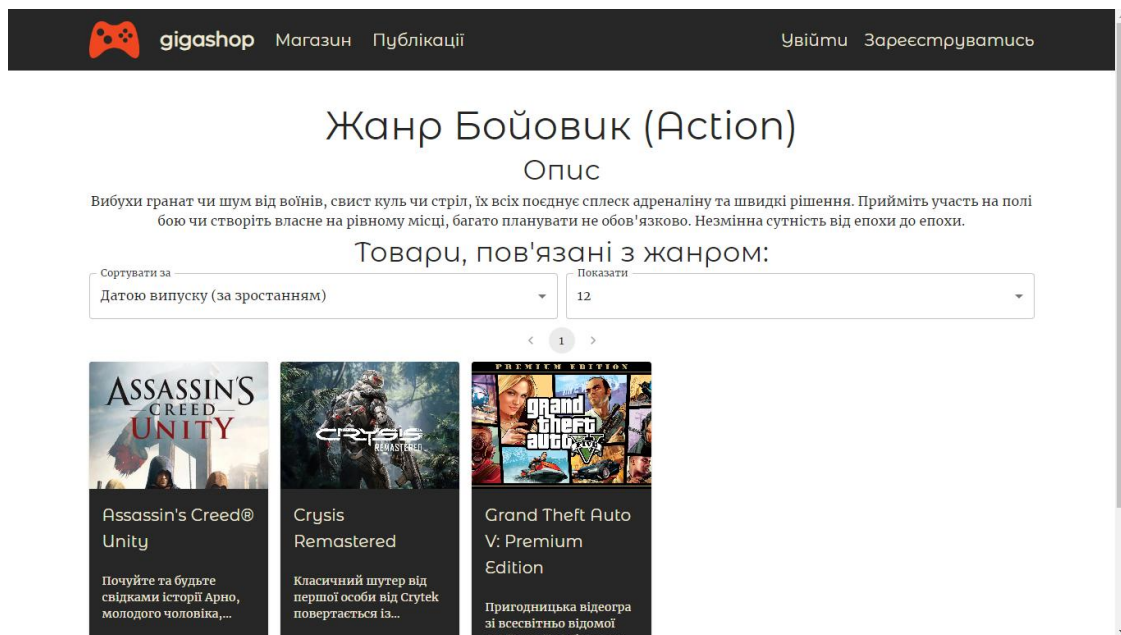


Рис.3.20 – Сторінка окремого жанру

На сторінці окремої компанії маємо можливість дізнатись більше про дану компанію, включно зі списками тих товарів, з якими вони пов'язані, а саме всі випущені та розроблені продукти окремо за вибором елементів меню



Рис.3.21 – Сторінка окремої компанії

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк. 80
		Чижмотря О.Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		

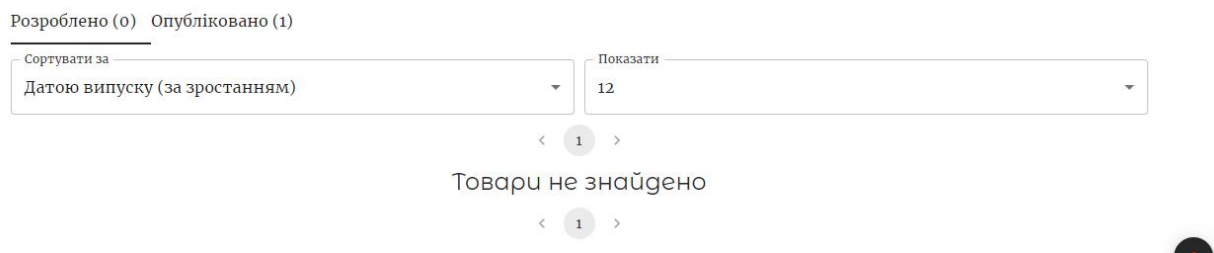


Рис.3.22 – Пустий список Розроблених товарів для обраної компанії

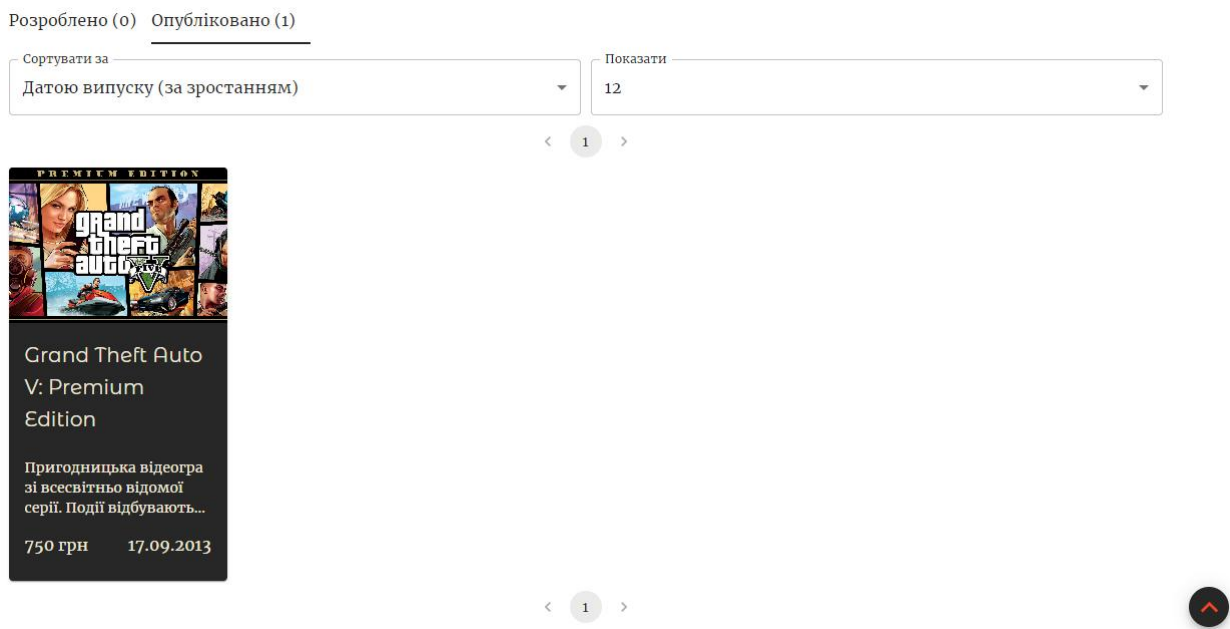


Рис.3.23 – Список випущених відеоігор обраною компанією

При переході на сторінку публікацій та статей веб-додатку отримуємо список публікацій до перегляду та елементи керування, поле пошуку за назвою та випадне меню фільтрів.

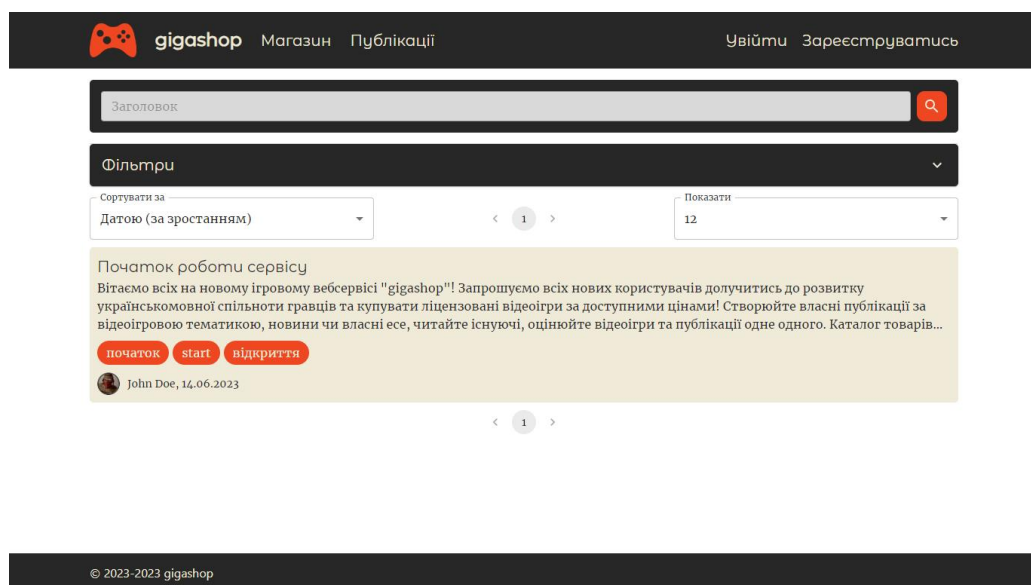


Рис.3.24 – Сторінка пошуку публікацій

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

Фільтри

Дата створення

дд.мм.рррр

Дата від

дд.мм.рррр

Дата до

Теги

Автори

пошук

ОЧИСТИТИ

Рис.3.25 – Відкрите меню фільтрів пошуку публікацій

При перегляді публікації на сторінці з’являються дані про автора, включно з його роллю, заголовком, дату створення, теги даної статті та вміст статті. Вміст статті зберігається у базі даних у вигляді HTML, яке при друку на екрані перетворюється на JSX компоненти.



Рис.3.26 – Верхня частина сторінки публікації

Вітаємо всіх на новому ігровому вебсервісі "gigashop"!

Запрошуємо всіх нових користувачів долучитись до розвитку українськомовної спільноти гравців та купувати ліцензовані відеоігри за доступними цінами! Створюйте власні публікації за відеоігровою тематикою, новини чи власні есе, читайте існуючі, оцінюйте відеоігри та публікації одне одного.

Каталог товарів оновлюється регулярно для якнайшвидшого доступу користувачів до найновіших відеоігор та товарів до них! Купуючи наші товари, ви підтримуєте український бізнес та поширюєте наш сервіс для доступу до нього українським новачкам світу відеоігор.

Разом ми не переможемо!

Рис.3.27 – Змістова частина сторінки публікації

Оцінка:

★★★★★

5

Коментарі:



Jane Doe

Дякую за вашу роботу! Слава Україні!

★★★★★

Рис.3.28 – Блок коментарів публікації

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінки реєстрації та входу (логіну) містять текстові поля для введення даних облікового запису до створення чи виконання перевірки, авторизації відповідно до сторінки.

Рис.3.29 – Поля сторінки логіну

Рис.3.30 – Форма реєстрації

Інші компоненти системи, що включають в себе сторінки профілю, кошику та адміністративні сторінки використовують подібний дизайн з елементами керування та кольоровою палітрою. Їх огляд відбудеться в наступному підрозділі під час тестування функціоналу системи.

3.3. Тестування роботи програмного забезпечення

Тестування роботи розроблених додатків буде розділено на 2 частини, а саме перевірка роботи сервера за допомоги платформи Postman через створення до нього запитів з корисним навантаженням та тестування дизайну та коректної зв'язку зі сервером, роботи функціоналу.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

Платформа Postman дозволяє тестувати розроблені API на етапі їх розробки без необхідності створювати графічний дизайн, який викликав би необхідний для тестування функціонал. Дана платформа зберігає створені API запити та дозволяє каталогізувати їх, експортувати в різних форматах та імпортувати назад тощо. Протягом циклу розробки серверної частини та під час кінцевого налагодження у Postman було організовано велику кількість запитів та зібрано в каталоги по основним цілям та напрямкам виконання запитів.

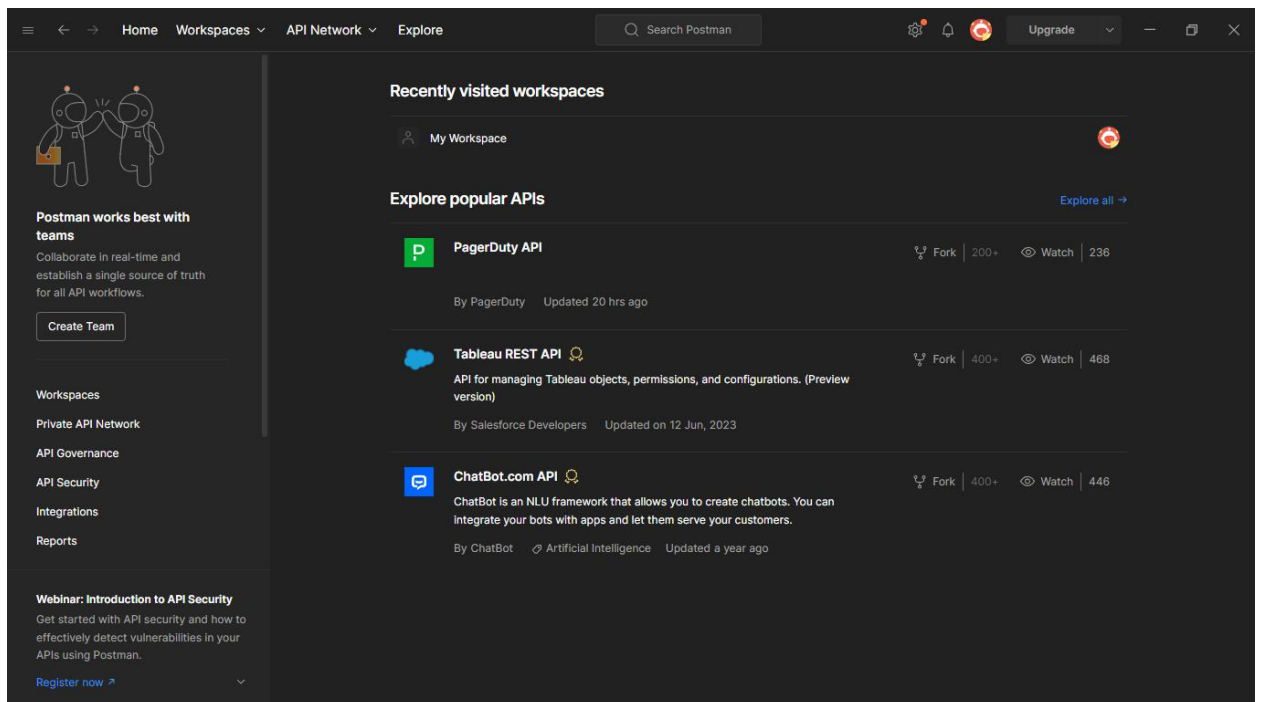


Рис.3.31 – Графічний інтерфейс платформи Postman

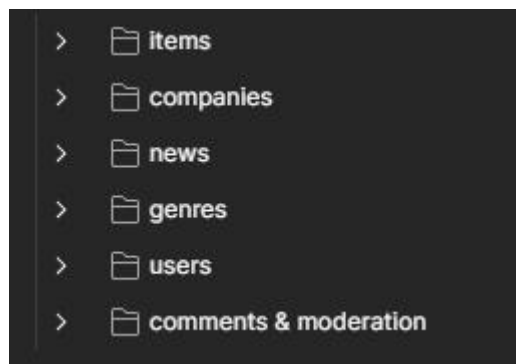


Рис.3.32 – Зібрані колекції запитів для тестування

Експортований JSON об'єкт колекції створених запитів для тестування наведено у додатку К.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

Кожний тестовий запит, створений на платформі Postman, відповідає певному маршруту, що налаштовано на серверному додатку. Відповідниками Create, Read, Update та Delete операцій є запити POST, GET, PATCH, DELETE. Запити GET на загальну та точкову адресу з використанням ID запису даних не потребують авторизації, у той час як решта запитів відбуваються з наданим заголовком Authorization, значення якого є “Bearer TOKEN”, де TOKEN – JWT з cookies-файлів чи у прямому розпорядженні користувача. Кожен токен має термін придатності в 24 години, тому після вичерпання строку необхідно оновлювати авторизацію через логін в системі. Наразі перевіримо GET запити, що не потребують авторизації.

Типово групові GET запити отримують в якості параметрів значення для фільтрації через SQL конструкцію WHERE. Окрім цього, всі запити на читання даних можуть отримувати параметри пагінації, тип сортування та назву атрибуту, по якому проводиться сортування. Окрім цього, для отримання прихованих даних можливе використання параметру hidden типу boolean, за замовченням значення false. В результаті отримуємо масив об’єктів БД за параметрами запиту та кількість всіх об’єктів запитаної таблиці, включно з прихованими за значення hidden = true.

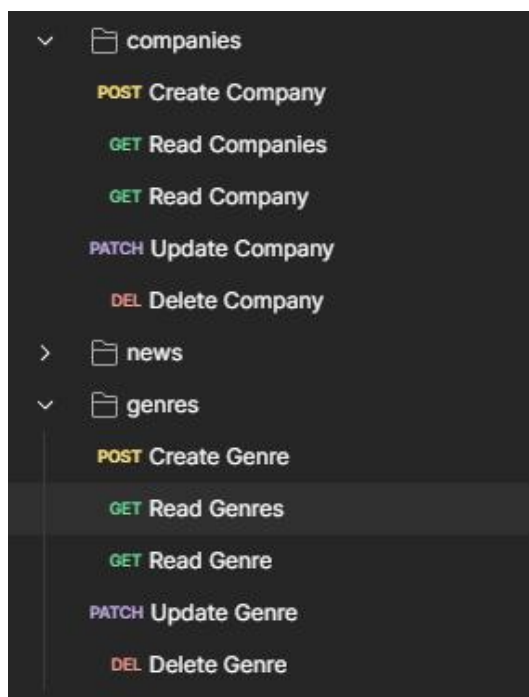


Рис.3.33 – Типові запити на виконання CRUD операцій

GET

http://localhost:5000/api/shop/genres

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

	Key	Value
☐	name	det
☐	description	w
☐	hidden	true
☐	limit	3
☐	page	1
☐	sortBy	id
☐	desc	true
	Key	Value

Рис.3.34 – Запит на отримання всіх жанрів та параметри запиту

Pretty	Raw	Preview	Visualize	JSON	⌵	⌵
1	"genres": [					
2	{					
3	"id": 11,					
4	"name": "Бойовик (Action)",					
5	"description": "Вибухи гранат чи шум від воїнів,					
6	та швидкі рішення. Прийміть участь на полі бою,					
7	не обов'язково. Незмінна сутність від епохи,					
8	"hide": false,					
9	"createdAt": "2023-06-14T07:23:08.029Z",					
10	"updatedAt": "2023-06-14T07:23:08.029Z",					
11	"deletedAt": null					
12	},					
13	{					
14	"id": 5,					
15	"name": "Вигляд від першої особи (First-person)",					
16	"description": "Відчуйте через свій екран вогонь					
17	Різноманітні відеоігри, пов'язані любов'ю					
18	оточення. Співробітьте реалізмом, та стримуйте					
19	"hide": false,					
20	"createdAt": "2023-06-13T21:49:54.106Z",					
21	"updatedAt": "2023-06-14T07:24:48.120Z",					
22	"deletedAt": null					
23	},					

Рис.3.35 – Результат пошуку без використання параметрів

GET

http://localhost:5000/api/shop/genres?name=Вир&sortBy=id&desc=true

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Body

Cookies

Headers (8)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON

```
4      "id": 8,
5      "name": "Вигляд від третьої особи (Third-person)",
6      "description": "Змініть акцент гри для іншого типу занурення в ігровий світ. Ви будете очима головного героя, але ніби в кіно. Станьте особливим гравцем, який переживає події, що розгортаються всюди та одночасно.",
7      "hide": false,
8      "createdAt": "2023-06-14T06:45:43.348Z",
9      "updatedAt": "2023-06-14T07:26:28.063Z",
10     "deletedAt": null
11   },
12   {
13     "id": 5,
14     "name": "Вигляд від першої особи (First-person)",
15     "description": "Відчуйте через свій екран вогонь зі зброєю чи подіями. Різноманітні відеоігри, пов'язані любов'ю до реалістичного оточення. Співробітьте реалізмом, та стримуйте бажання самим гравцем.",
16     "hide": false,
17     "createdAt": "2023-06-13T21:49:54.106Z",
18     "updatedAt": "2023-06-14T07:24:48.120Z",
19     "deletedAt": null
20   }
21 ],
22 "totalCount": 11
23 ]
```

Рис.3.36 – Результат запиту на жанри зі встановленням параметрів пошуку

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

Запити по різні дані мають різні параметри з огляду різних моделей БД. Для таких таблиць, як Компанія чи Товар можливий пошук за точними параметрами чи значенням в певному проміжку.

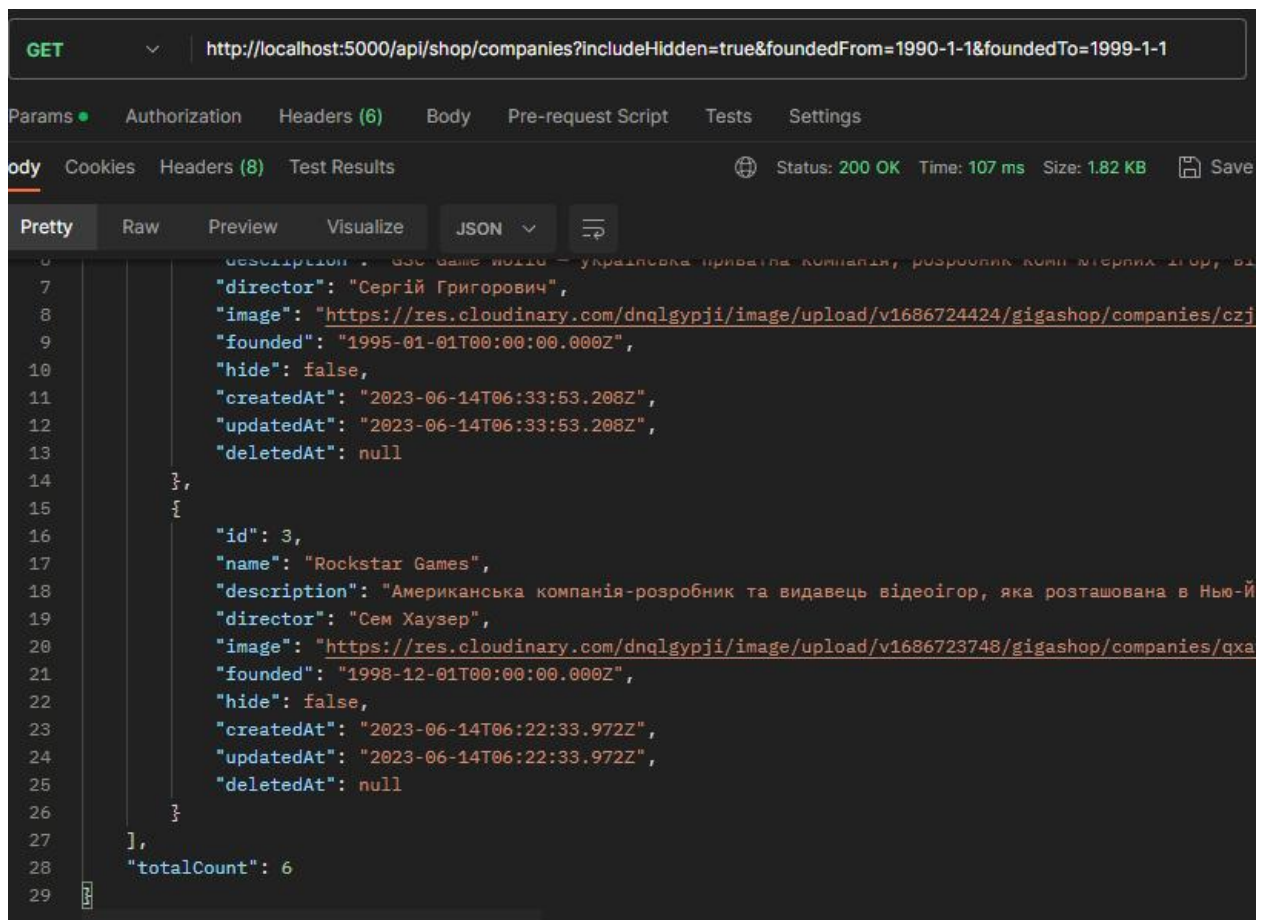


Рис.3.37 – Запит на дані компаній з проміжком значень до дати заснування

Отримання окремих записів відбувається за тим самим шляхом запити з додаванням ID необхідної одиниці даних. При запиті окремих одиниць разом з ними можна отримати масиви пов'язаних записів з інших таблиць. Наприклад, разом з товаром надаються дані про його жанри, розробників та видавця. Окрім цього, за допомоги параметру includeHidden можливо отримати схований запис таблиці, що використовується в запитах веб-сайту для адміністративної частини.

Отриманий токен дозволяє ідентифікувати особу в системі та виконувати операції над власними оцінками, коментарями та публікаціями. Встановлення даного токена в браузері виконується автоматично.

Params		Authorization	Headers (9)	Body	Pre-request Script	Tests	Settings
Headers		8 hidden					
		Key		Value			
		☒ Authorization		Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ...			
		Key		Value			

Рис.3.40 – Встановлений заголовок авторизації з токеном доступу

POST

http://localhost:5000/ap/news/

Send

Params

Authorization

Headers (9)

Body •

Pre-request Script

Tests

Settings

Cookies

Body

Cookies

Headers (8)

Test Results

Status: 200 OK

Time: 139 ms

Size: 2.27 KB

Save as Example

Pretty

Raw

Preview

Visualize

JSON

4

5

6

7

8

9

10

"title": "News1",

"content": "<p>Lorem ipsum sagittis: enim porta vivamus lorem ligula ut, mauris odio molestie sapien non proin. Eget at enim cursus eget amet magna eros, arcu, enim justo, lorem enim vivamus gravida sapien. Elementum eget tempus fusce auctor morbi lorem gravida — sapien lectus risus ornare: quisque, nulla. Eu ipsum sodales nec morbi porta arcu lectus non in vivamus cursus: ipsum proin vulputate: mattis tempus metus orci rutrum, tempus quisque sodales nam.</p><p>Porttitor mattis vivamus odio magna non — in quam porttitor lectus pellentesque. Ipsum et justo vitae molestie eu at nulla vivamus porta, orci quam rutrum amet. Auctor eget ut integer commodo enim ornare quam bibendum ipsum lorem ipsum sed duis commodo. Odio leo et sem arcu amet elementum justo, nibh integer bibendum donec, arcu elementum molestie a nec curabitur. Massa mattis tellus ipsum, ornare, malesuada orci bibendum cursus quisque amet, et sodales, vivamus —— malesuada sodales.</p><p>Elementum quam pellentesque eros metus maecenas tempus orci adipiscing et nibh. Rutrum quam sem magna ornare nibh ultricies bibendum tempus massa, leo donec pellentesque duis proin. Elementum sagittis tellus duis amet ultricies et cursus donec porttitor eu non pharetra proin magna porta eu porttitor sagittis leo pellentesque, sem diam. Sit nam maecenas mattis lectus proin ornare nulla, ipsum ligula bibendum porttitor sodales arcu pellentesque rutrum. In cursus orci at curabitur malesuada pellentesque, sem fusce porta sodales, porttitor. Morbi diam eget, eros amet duis ultricies eros odio adipiscing morbi leo maecenas justo a quisque integer tempus malesuada enim maecenas eros. Magna tempus massa et non pharetra sapien commodo sapien tempus lectus sem quisque rutrum nibh ut, justo mauris eu non, quam at ultricies.</p>>

"hide": false,

"userId": 3,

"tags": [

"Fantasy",

"Fantastic",

Протягом використання веб-сайту можливе одночасне тестування дизайну, розробленого функціоналу зі створення запитів та серверний функціонал з їх обробки, тож далі тестування буде проходити через клієнтський додаток системи.

Форма реєстрації дозволяє створити обліковий запис згідно реальних особистих даних та відкриває можливість ефективного використання сайту в якості користувача. Дана форма використовує віджет завантаження єдиного зображення, який викликає відповідний компонент у вікні браузера.

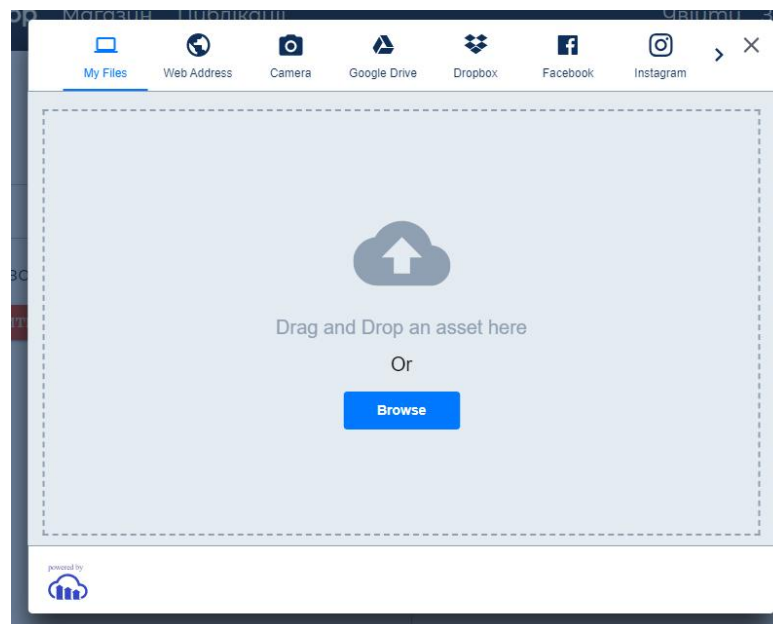


Рис.3.43 – Вікно завантаження зображення в початковому стані

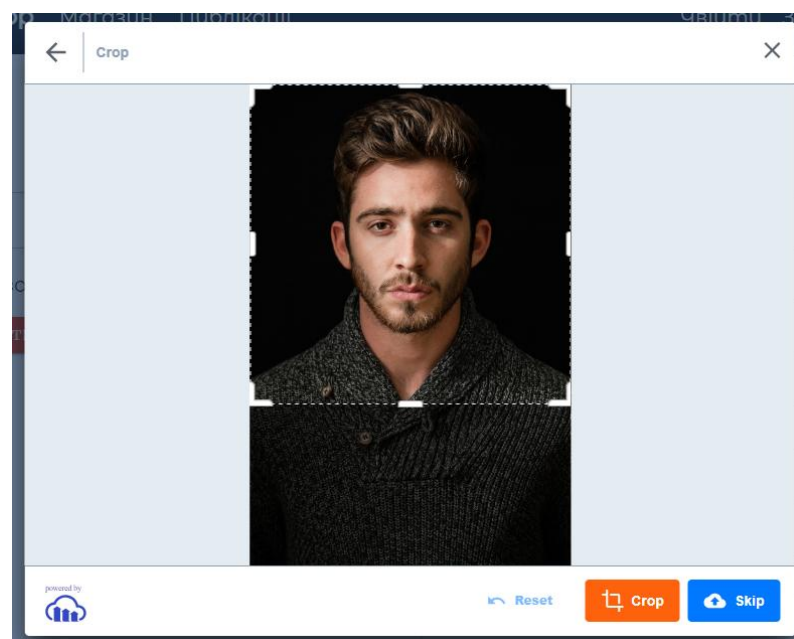


Рис.3.44 – Можливість обрізки зображення

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

Ім'я	Прізвище
Larry	Ipsum
Завантажити зображення	Пошта
ЗАВАНТАЖИТИ ОЧИСТИТИ	larry@mail.com
	Логін
	larry
	Пароль
	
	Повторіть пароль
	

Рис.3.45 – Заповнена форма реєстрації

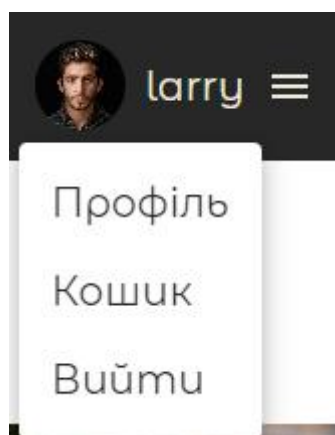


Рис.3.46 – Результат авторизації та доступ до меню користувача

Після авторизації стає можливим створення коментарів та оцінок, які впливають на рейтинг публікацій та товарів відповідно. Окрім цього, можливо скористатись формою створення публікацій через меню пошуку.

Коментарі:

Ваш коментар:

Оцінка:



Рис.3.47 – Спільна форма для створення коментарів та оцінок

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

Ваш коментар:



Larry Ipsum

Сподіваюсь, в мене потягне залізо на таке!



Рис.3.48 – Відображення створеного коментаря

Створити новину

Фільтри

Рис.3.49 – Поява посилання до форми створення публікації

Заголовок

Теги

☐ Приховати?

Контент

File Edit View Insert Format Tools Table Help

↶ ↷ Paragraph

B *I*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean vitae tincidunt ligula. Nullam at arcu venenatis nibh lobortis ultrices sit amet et nibh. Curabitur sit amet interdum ante, consectetur lacinia nisi. Praesent non ligula porta, sollicitudin erat eu, lacinia massa. Quisque finibus, nibh et sagittis feugiat, diam turpis feugiat augue, ut faucibus nulla ex eu nisi. Aenean non nunc vitae urna vestibulum ornare laoreet non ex. Sed ornare sem quam, et finibus nulla mattis in. Proin eleifend libero facilisis, vestibulum velit ac, accumsan enim. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec egestas neque a tincidunt consequat. Suspendisse potenti. Mauris pretium arcu in justo consectetur, nec varius felis elementum. Maecenas et malesuada libero. Duis mollis ultrices dolor in porta. In erat turpis, eleifend quis aliquam ut, lacinia vel massa. Nullam pretium lobortis aliquam.

Sed gravida sed erat at lacinia. Vestibulum mollis nunc vel metus porta, nec volutpat nisi dictum. Nullam turpis urna, iaculis sit amet faucibus et, scelerisque eu metus. Sed at ex ligula. Morbi egestas nulla a dolor euismod fermentum. Donec orci elit, dignissim ac aliquam non, luctus vel elit. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nulla facilisi. Sed quis lorem eros. Donec ultrices enim mi.

Рис.3.50 – Форма створення публікації

News1

Lorem ipsum sagittis: enim porta vivamus lorem ligula ut, mauris odio molestie sapien non proin. Eget at enim cursus eget amet magna eros, arcu, enim justo, lorem enim vivamus gravida sapien. Elementum eget tempus fusce auctor morbi lorem gravida — sapien lectus risus ornare: quisque, nulla. Eu ipsum sodales nec morbi porta arcu lectus non in vivamus cursus: ipsum proin vulputate: mattis tempus metus...

Fantasy

Fantastic

Detective

Jane Doe, 15.06.2023

Моя перша новина

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean vitae tincidunt ligula. Nullam at arcu venenatis nibh lobortis ultrices sit amet et nibh. Curabitur sit amet interdum ante, consectetur lacinia nisi. Praesent non ligula porta, sollicitudin erat eu, lacinia massa. Quisque finibus, nibh et sagittis feugiat, diam turpis feugiat augue, ut faucibus nulla ex eu nisi. Aenean non nunc vitae urna vestibulum...

початок

відеоігри

Larry Ipsum, 15.06.2023

Рис.3.51 – Нові публікації, створені при тестуванні

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТИ ДО СПИСКУ БАЖАНЬ

ДОДАТИ ДО КОШИКА

Успіх!
Товар успішно додано до списку бажань

Успіх!
Товар успішно додано до кошика

Рис.3.53 – Повідомлення про успішну взаємодію з товаром

ВИДАЛИТИ ЗІ СПИСКУ БАЖАНЬ

ВИДАЛИТИ З КОШИКА



МагазинПублікації



larry



Larry Ipsum
larry
larry@mail.com

СПИСОК БАЖАНЬ

ПУБЛІКАЦІЇ

КОМЕНТАРІ ДО ПУБЛІКАЦІЙ

ВІДГУКИ ДО ТОВАРІВ

ІСТОРІЯ ПОКУПОК

Assassin's Creed® Unity

Почуйте та будьте свідками історії Арно, молодого чоловіка, який вирушає в надзвичайну подорож, щоб викрити справжні сили, що стоять за Французькою революцією. У абсолютно новому кооперативному режимі ви та ваші друзі також опинитесь у центрі нещадної боротьби за долю нації.

Козаки 3

Повернення класики RTS. Добувайте ресурси, будуйте міста та досліджуйте технології, керуйте колосальними арміями та знищуйте війська супротивників у грандіозних історичних баталіях на суші та в морі!

© 2023-2023 gigashop

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ІЗ	Арк.
		Чижевотря О.Г.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

Моя перша новина

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean vitae tincidunt ligula. Nullam at arcu venenatis nibh lobortis ultrices sit amet et nibh. Curabitur sit amet interdum ante, consectetur lacinia nisi. Praesent non ligula porta, sollicitudin erat eu, lacinia massa. Quisque finibus, nibh et sagittis feugiat, diam turpis feugiat augue, ut faucibus nulla ex eu nisi. Aenean non nunc vitae urna vestibulum...

Рис.3.56 – Панель створених користувачем публікацій

СПИСОК БАЖАНЬ ПУБЛІКАЦІЇ КОМЕНТАРІ ДО ПУБЛІКАЦІЙ ВІДГУКИ ДО ТОВАРІВ ІСТОРІЯ ПОКУПОК



Larry Ipsum
Сподіваюсь, в мене потягне залізо на таке!
★★★★★

Рис.3.57 – Приклад відображення коментарів та відгуків користувача

Для доступу до кошика необхідно бути авторизованим до системи, наявність товарів у кошику не обов’язкова. При перегляді в залежності від товарів у кошику буде відображено повідомлення про відсутність товарів або їх список та кінцева ціна до оплати. Вартість кошика користувача враховує стани знижки на товари, та відображається на сторінці.

Кошик користувача larry

Кошик порожній

Рис.3.58 – Приклад пустого кошику

Кошик користувача larry

Crysis Remastered

Класичний шутер від першої особи від Crytek повертається із захоплюючим ігровим процесом, світом пісочниці та захоплюючими епічними битвами, які вам сподобалися вперше – тепер із оновленою графікою, оптимізованою для нового покоління обладнання.

420 грн

×

Козаки 3

Повернення класики RTS. Добувайте ресурси, будуйте міста та досліджуйте технології, керуйте колосальними арміями та знищуйте війська супротивників у грандіозних історичних баталіях на суші та в морі!

289 грн

248.54 грн

×

Загальна вартість: 668.54 грн

ПІДГОТУВАТИ КОШИК ДО ОПЛАТИ

ОЧИСТИТИ КОШИК

Рис.3.59 – Приклад кошика зі списком товарів до оплати

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

Для підготування кошику до оплати, а саме встановлення товарам ID транзакції для їх ідентифікації в системі, при натисненні кнопки відбувається запит до БД від імені користувача за його токеном для виконання налаштування. В результаті всі одиниці кошику отримують ідентифікатор за алгоритмом UUID V4, що повертається до веб-сайту для створення кнопки оплати.

	id [PK] integer	transactionId character varying (255)	createdAt timestamp with time zone	updatedAt timestamp with time zone	userId integer	itemId integer
1	4	[null]	2023-06-16 07:42:12.38+00	2023-06-16 07:42:12.38+00	4	4
2	5	[null]	2023-06-16 07:42:19.949+00	2023-06-16 07:42:19.949+00	4	2

Рис.3.60 – Стан кошику користувача до налаштування

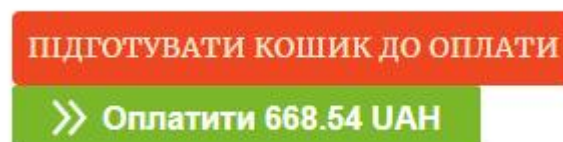


Рис.3.61 – Поява кнопки оплати та придбання товарів

	id [PK] integer	transactionId character varying (255)	createdAt timestamp with time zone	updatedAt timestamp with time zone	userId integer	itemId integer
1	4	fa04e5bc-d996-4471-8e42-494ccd729c9b	2023-06-16 07:42:12.38+00	2023-06-16 07:46:09.636+00	4	4
2	5	fa04e5bc-d996-4471-8e42-494ccd729c9b	2023-06-16 07:42:19.949+00	2023-06-16 07:46:09.636+00	4	2

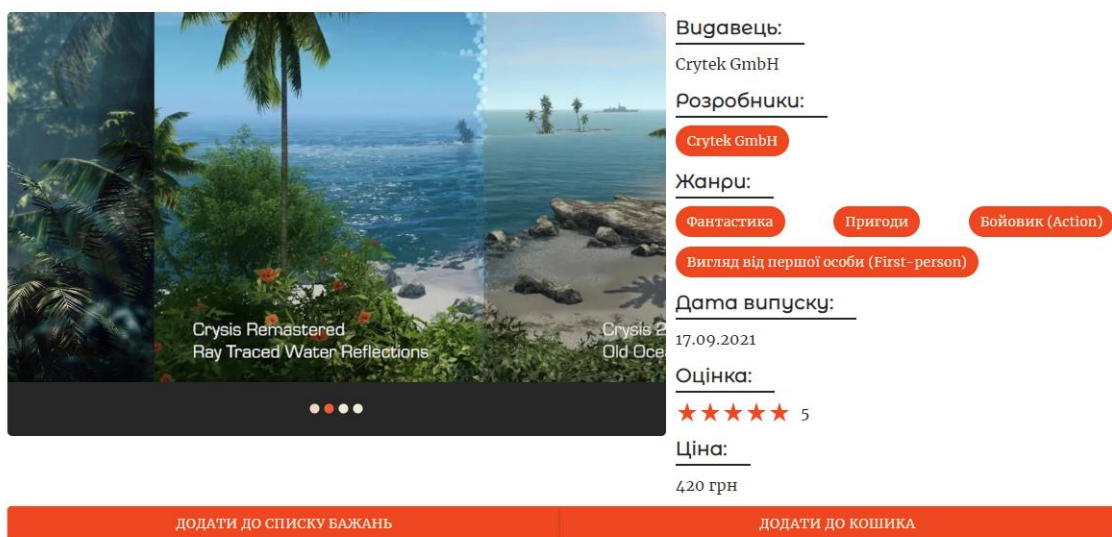
Рис.3.62 – Результат налаштування кошика

На сторінці кошика можливе видалення окремих товарів або очищення всього кошика однією кнопкою, що виконує запит на видалення пов'язаних записів з таблиці БД.

	id [PK] integer	transactionId character varying (255)	createdAt timestamp with time zone	updatedAt timestamp with time zone	userId integer	itemId integer
--	--------------------	--	---------------------------------------	---------------------------------------	-------------------	-------------------

Рис.3.63 – Результат повного очищення кошика

Crysis Remastered



Видавець:
Crytek GmbH

Розробники:
Crytek GmbH

Жанри:
Фантастика Пригоди Войовик (Action)

Вигляд від першої особи (First-person)

Дата випуску:
17.09.2021

Оцінка:
★★★★★ 5

Ціна:
420 грн

ДОДАТИ ДО СПИСКУ БАЖАНЬ ДОДАТИ ДО КОШИКА

Рис.3.64 – Стан товару при очищенні кошика

При виконанні оплати виконується перенаправлення на сторінку оплати LiqPay, що отримує необхідні зашифровані дані про вартість оплати. В результаті оплати користувач має можливість повернутись на сайт, а сервер отримує запит із ідентифікатором транзакції для перенесення товарів до придбаних.

Тестовий режим

До сплати: 668.54 UAH
Оплата товарів

Сплатити через Приват24

24 Pay

Pay VISA

Картка Інший спосіб

Номер картки
**** * * * *

Термін дії CV2
MM/YY * * *

Натискаючи на кнопку «Сплатити», ви підтверджуєте що ознайомлені з переліком інформації про послугу та приймаєте умови публічного договору

Сплатити

Скасувати оплату

Рис.3.65 – Сторінка сплати вартості товарів

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				96
Змн.	Арк.	№ докум.	Підпис	Дата		

1	SELECT * FROM public.item_carts
2	ORDER BY id ASC

Data Output	Messages	Notifications									
<table> <tr> <th></th><th>id [PK] integer</th><th>transactionId character varying (255)</th></tr> <tr> <td>1</td><td>6</td><td>a239c066-c7f9-44ba-a413-f4d7ccb498f3</td></tr> <tr> <td>2</td><td>7</td><td>a239c066-c7f9-44ba-a413-f4d7ccb498f3</td></tr> </table>		id [PK] integer	transactionId character varying (255)	1	6	a239c066-c7f9-44ba-a413-f4d7ccb498f3	2	7	a239c066-c7f9-44ba-a413-f4d7ccb498f3		
	id [PK] integer	transactionId character varying (255)									
1	6	a239c066-c7f9-44ba-a413-f4d7ccb498f3									
2	7	a239c066-c7f9-44ba-a413-f4d7ccb498f3									

Рис.3.66 – Налаштовані товари кошику перед придбанням

Тестовий режим

Дякуємо! Платіж успішно відправлено.

ID: 2327512685

Відправити квитанцію

Підтримати українську армію!

Допомогти

Повернутися до сайту

Допомога онлайн

3700 безкоштовно з мобільних

+38-073-716-11-31 для дзвінків з-за кордону

беремо і робимо

ID Check

VISA

Powered by LIQPAY»

Рис.3.67 – Повідомлення про успіх оплати

1

SELECT * FROM public.item_boughts

2

WHERE "transactionId" like 'a239c066-c7f9-44ba-a413-f4d7ccb498f3'

3

ORDER BY id ASC

Data Output

Messages

Notifications

	id [PK] integer	itemId integer	transactionId character varying (255)	createdAt timestamp w	updatedAt timestamp w	deletedAt timestamp w	userId integer
1	3	2	a239c066-c7f9-44ba-a413-f4d7ccb498f3	2023-06-...	2023-06-...	[null]	4
2	4	4	a239c066-c7f9-44ba-a413-f4d7ccb498f3	2023-06-...	2023-06-...	[null]	4

Рис.3.68 – Записи про результат придбання товарів у БД

Crysis Remastered

Класичний шутер від першої особи від Crytek повертається із захоплюючим ігровим процесом, світом пісочниці та захоплюючими епічними битвами, які вам сподобалися вперше – тепер із оновленою графікою, оптимізованою для нового покоління обладнання.

Придбано: 16.06.2023

Козаки 3

Повернення класики RTS. Добувайте ресурси, будуйте міста та досліджуйте технології, керуйте колосальними арміями та знищуйте війська супротивників у грандіозних історичних баталіях на суші та в морі!

Придбано: 16.06.2023

Рис.3.69 – Панель придбаних товарів на сторінці профілю користувача

Для створення нових записів до БД необхідно мати доступ працівника через роль адміністратора або модератора. Адміністративна панель має бічну навігаційну панель для доступу до основних сторінок панелі, що включають в себе сторінки списків даних, з яких можливо оглянути детально окремі записи, та посилання на форми для створення нових записів. Редагування записів відбувається з використанням компонент відповідних форм з наданням об'єкту запису, що необхідно відредагувати.

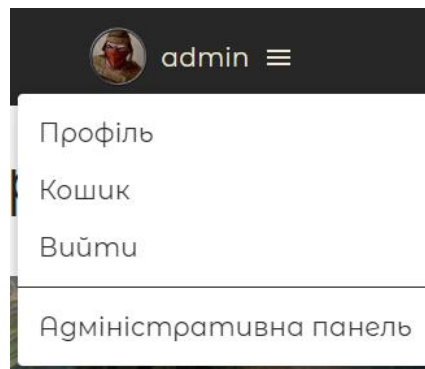


Рис.3.70 – Нове посилання в меню дій користувача для адміністратора.

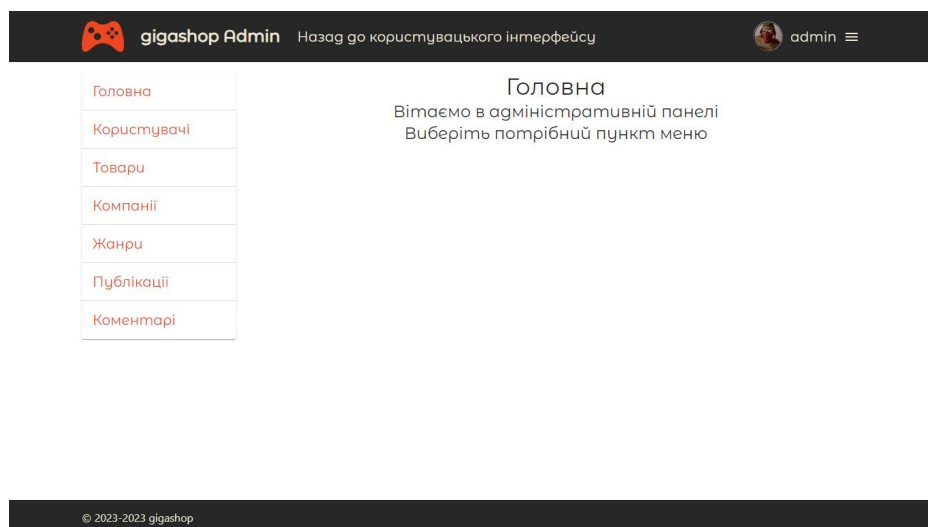


Рис.3.71 – Початкова сторінка адміністративної панелі

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				98
Змн.	Арк.	№ докум.	Підпис	Дата		

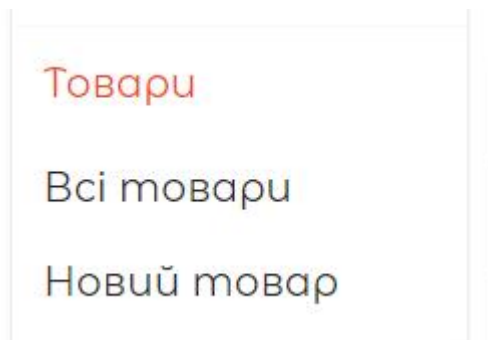


Рис.3.72 – Приклад розгортання елемента навігаційної панелі

Список товарів

Сортувати за
Назвою (за зростанням) ▼

Показати
12 ▼

< 1 >

Assassin's Creed® Unity (№3)	▼
Crysis Remastered (№2)	▼
Grand Theft Auto V: Premium Edition (№1)	▼
Козаки 3 (№4)	▼

< 1 >

Рис.3.73 – Сторінка всіх товарів як приклад відображення списків даних

При створенні одиниць записів БД використовуються різноманітні елементи форм, як для записування текстових даних, так і для введення дати до необхідного параметра об'єкту, разом із віджетами зображень.

Створення жанру

Назва

Опис

Відчуйте міць сталі в своїх руках, прорубайте собі шлях через орди ворогів та відчуйте смак крові тих, хто наважився стати перед Вами та Вашою метою!

ЗБЕРЕГТИ

СКИНУТИ

Рис.3.74 – Приклад створення жанру

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				99
Змн.	Арк.	№ докум.	Підпис	Дата		

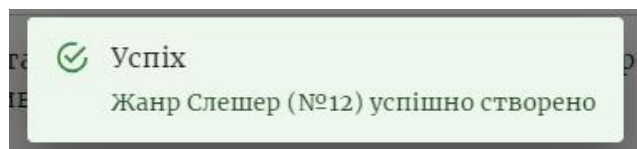


Рис.3.75 – Повідомлення про успішне створення жанру

Список жанрів

Сортувати за: Назвою (за зростанням) Показати: 12

< 1 >

Бойовик (Action)	👁️	✎	🗑️
Вигляд від першої особи (First-person)	👁️	✎	🗑️
Вигляд від третьої особи (Third-person)	👁️	✎	🗑️
Детектив	👁️	✎	🗑️
Жахи (Horror)	👁️	✎	🗑️
Комедія	👁️	✎	🗑️
Пригоди	👁️	✎	🗑️
Скритність (Stealth)	👁️	✎	🗑️
Слешер	👁️	✎	🗑️
Стратегія	👁️	✎	🗑️
Фантастика	👁️	✎	🗑️
Фентезі	👁️	✎	🗑️

Рис.3.76 – Сторінка всіх жанрів у адміністративній панелі

Сортувати за: Назвою (за спаданням) Показати: 3

< 1 2 3 4 >

Слешер	👁️	✎	🗑️
Скритність (Stealth)	👁️	✎	🗑️
Пригоди	👁️	✎	🗑️

< 1 2 3 4 >

Рис.3.77 – Приклад роботи пагінації та сортування

Редагування жанру Слешер (№12)

Назва

Слешер

Опис

Відчуйте міць сталі в своїх руках, прорубайте собі шлях через орди ворогів та відчуйте смак крові тих, хто наважився стати перед Вами та Вашою метою! Динамічний геймплей та любов до холодної зброї, епічні спецефекти та постійні виклики чекають на Вас!

ЗБЕРЕГТИ СКИНУТИ

Рис.3.78 – Редагування жанру

Створення компанії

Назва
Platinum Games

Директор
Ацуші Інаба

Дата заснування
01-10-2007

Опис
Японська компанія - розробник відеоігор, що була спершу створена під назвою Seeds, Inc. колишніми ключовими працівниками Studio Clover.

ЗАВАНТАЖИТИ ЗОБРАЖЕННЯ ВИДАЛИТИ ЗОБРАЖЕННЯ



ЗБЕРЕГТИ СКИНУТИ

Рис.3.79 – Створення компанії

Створення товару

Назва
Metal Gear Rising: Revengeance

Опис
компанії Desperado Enforcement. Ігровий процес фокусується на битвах з ворогами використовуючи меч та інше озброєння для здійснення комбо та контратак. Райдзен може за допомогою «Blade Mode» розрізати кіборгів в уповільненому русі та діставати спеціальні частини з їх тіл. Стедс, властивий серії тут є, але опціональний.

Дата випуску
19-02-2013

Ціна
890

Кількість
590

Жанри
Скритність (Stealth) Бойовик (Action) Вигляд від третьої особи (Third-person) Спешер Фантастика Жанри

Розробники
Platinum Games Розробники

Видавець
Konami

Рис.3.80 – Верхня частина форми створення товару

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

Характеристики

оперативної пам'яті, Відеокарта NVIDIA Geforce GTS 450, 25 ГБ вільного дискового простору,
DirectX 9.0с+;
Вікова категорія: 18+;
Дорослий вміст: груба лексика, насилля;

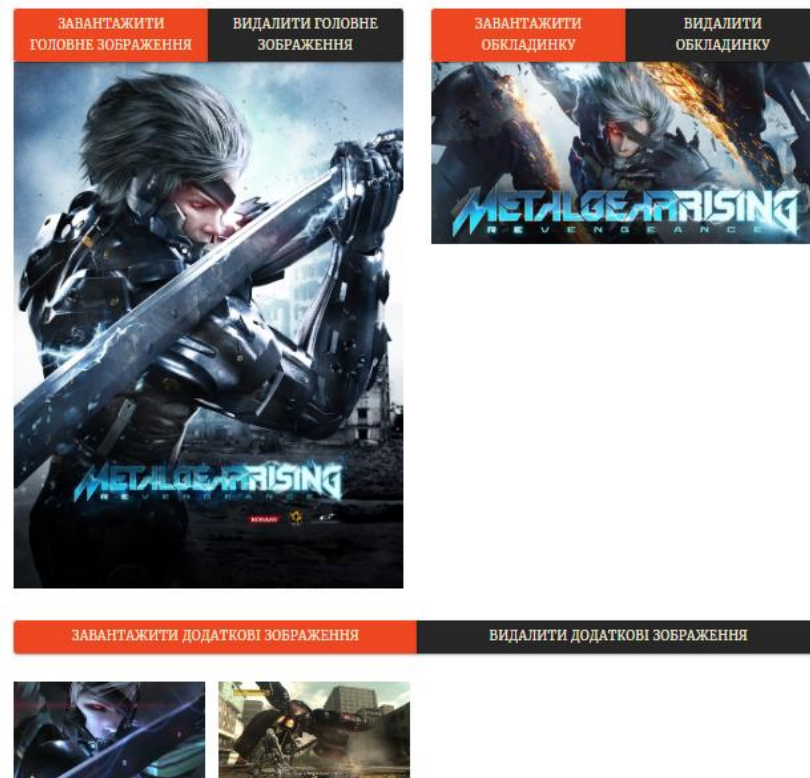


Рис.3.81 – Нижня частина форми створення товару

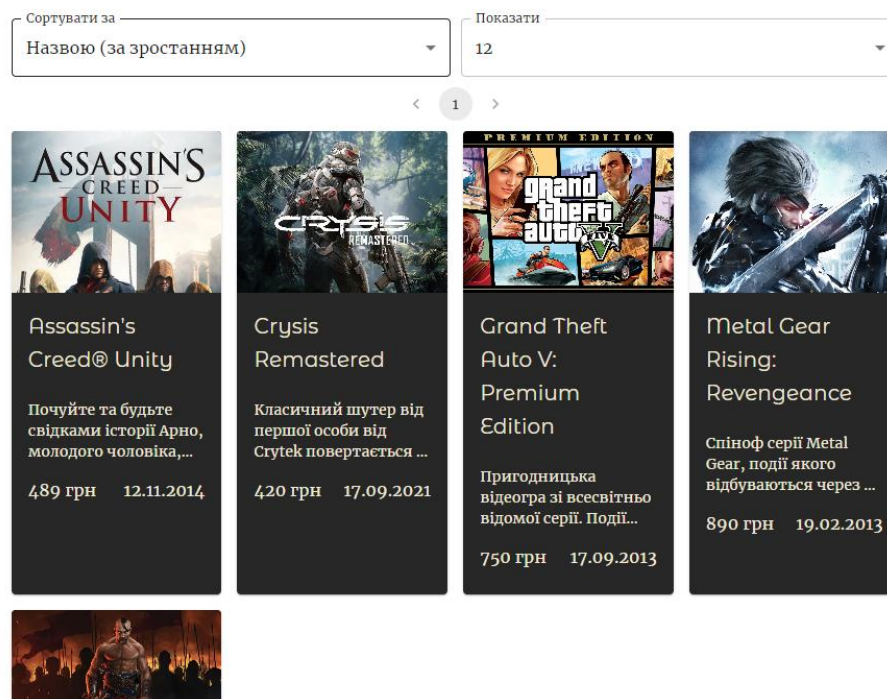


Рис.3.82 – Список товарів користувацької частини

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

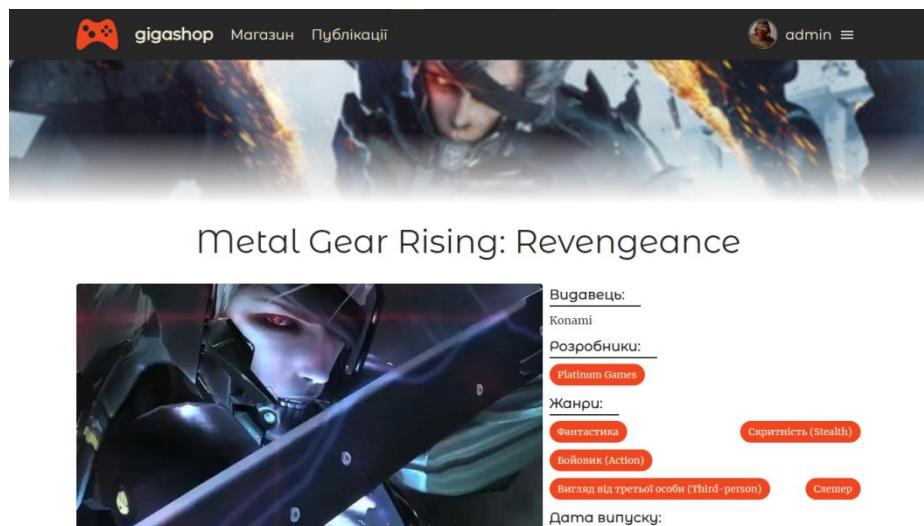


Рис.3.83 – Сторінка створеного товару

Адміністратори мають можливість зміни доступів для користувачів на сторінці перегляду всіх користувачів. Завдяки можливості зміни ролей сайт дозволяє імплементувати функціонально наймання чи звільнення працівників для подальшого керування контентом веб-додатку.

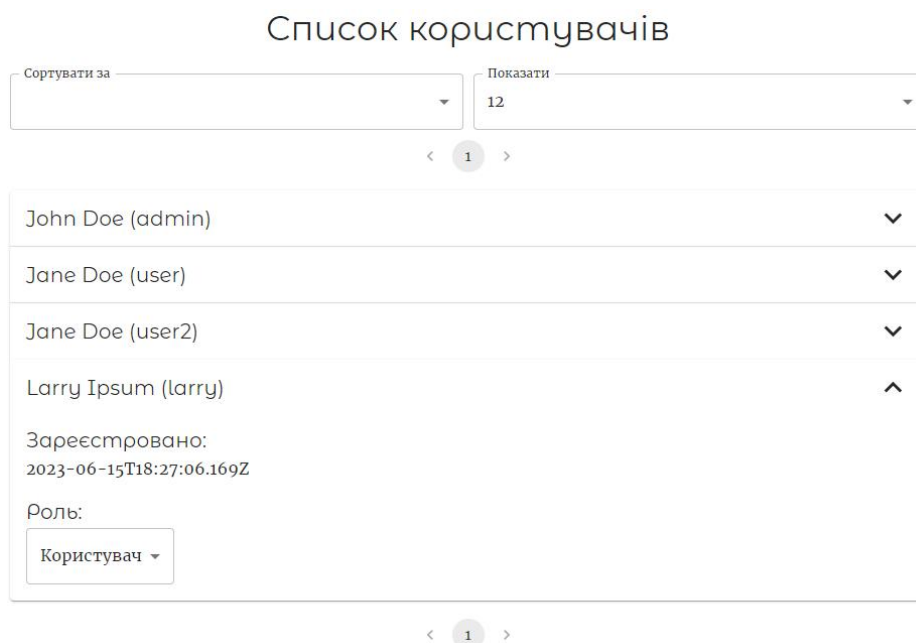


Рис.3.84 – Сторінка користувачів та дані певного користувача

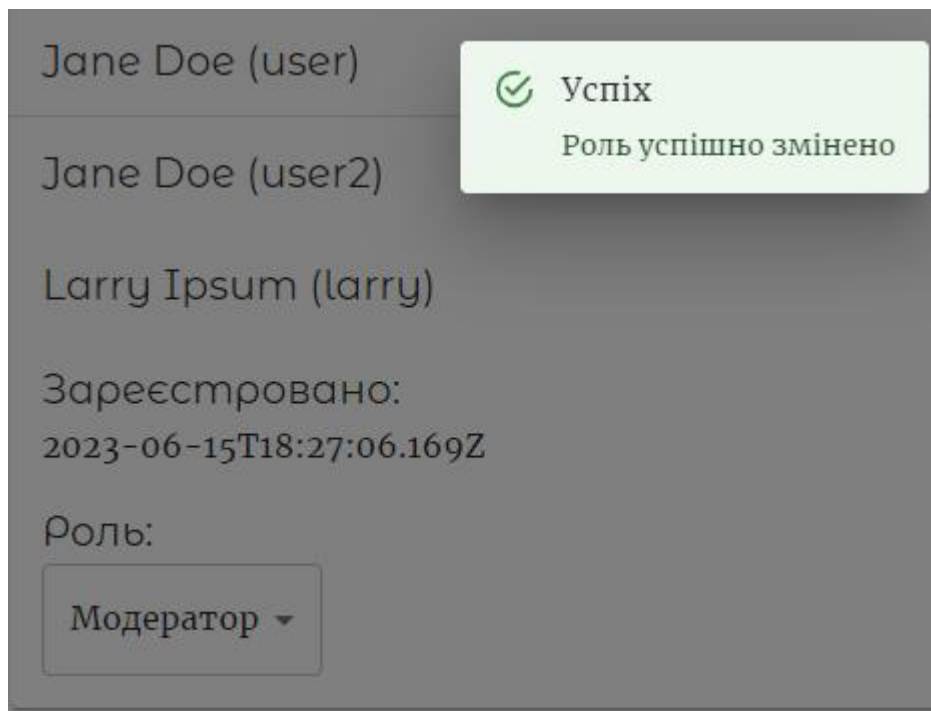


Рис.3.85 – Успішне редагування ролі користувача



Larry Ipsum
MODERATOR

Моя перша новина

15.06.2023

Теги: початок, відеоігри

Рис.3.86 – Наочне відображення зміни ролі користувача

Окрім створення та керування даними, адміністративна панель дозволяє переглянути статистику сайту за останній через відкриття панелі “Головна” та перехід до сторінки “Статистика”. Вміст сторінки на даному етапі розвитку веб-додатку включає лише текстове відображення даних про всі записи БД, в подальшому можливе покращення відображення зі застосуванням діаграм, графіків та інших об’єктів.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				104
Змн.	Арк.	№ докум.	Підпис	Дата		

Статистика даних вебсайту

Користувачі	▼
Товари	▼
Публікації	▼
Компанії	▼
Жанри	▼

Рис.3.87 – Початковий стан сторінки статистики

Публікації	^
Всього публікацій: 3	
За цей тиждень: 3	
За цей місяць: 3	
За цей рік: 3	
Всього прихованих: 0	
Всього з порушеннями: 0	
Всього авторів: 3	
Всього коментарів: 1	
За цей тиждень: 1	
За цей місяць: 1	
Всього з порушеннями: 0	

Рис.3.88 – Приклад панелі статистики по категорії

Висновки до третього розділу

У результаті виконаної роботи протягом третього розділу було виконано детальний опис механізму розгортання веб-додатку до робочого стану, виконано тестування дизайну створених сторінок клієнтського додатку системи та функціоналу компонентів веб-додатку.

Опис механізму розгортання веб-додатку до робочого стану включає графічне зображення рисунку діаграми розгортання, що включає всі компоненти системи та описує зв'язок з користувачами завдяки їх веб-браузерам. Під час тестування розгортання системи помилок в автоматизованому процесі за YAML конфігурацією не знайдено, система розгорнута успішно.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				105
Змн.	Арк.	№ докум.	Підпис	Дата		

Протягом аналізу дизайну та тестування сумісності з пристроями малого розміру було досягнуто висновку про задовільну адаптацію веб-сторінок для використання користувачами на більшій частині платформ.

Тестування функціоналу, що включало використання API платформи Postman та практичне використання сторінок клієнтського додатку, дозволило впевнитись у готовності додатку до презентації та подальшого використання, покращення та можливої оптимізації за необхідності.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				106
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час виконання випускної роботи бакалавра було спроектовано та реалізовано веб-додаток магазину відеоігор та порталу ігрових публікацій, для чого було вирішено наступні завдання:

1. Виконано аналіз конкурентів із визначенням головного необхідного функціоналу веб-додатку за предметною областю, а саме:

- наявність сторінок пошуку зі зручним способом введення даних фільтрації товарів магазину;
- категоризація товарів за жанрами та компаніями, які пов'язані з певним товаром;
- можливість детального огляду товарів та зображення на сторінці всіх деталей запитаних об'єктів магазину в зручному поданні;
- доступ до створення публікацій за предметною областю для створення спільноти веб-сайту та популяризації продукту.

2. Виконано аналіз функціональних, нефункціональних та бізнес-вимог до майбутнього програмного забезпечення. Визначено необхідний стек технологій та обґрунтовано вибір інструментарію для розробки.

3. Виконано проектування та аналіз, моделювання програмного забезпечення. За результатами моделювання та проектування було описано модель програмного продукту через зображення та опис створених діаграм.

4. Виконано програмну реалізацію компонентів програмного забезпечення згідно поставлених вимог до функціоналу, розробленого прототипу дизайну та отриманої моделі від аналізу та проектування програмного забезпечення.

5. Виконано опис процесу та тестування розгортання веб-додатку через використання запланованих технологій, протестовано створений дизайн веб-сторінок додатку на відповідність дизайну та вимогам, розроблений функціонал головних компонентів системи та їх взаємозв'язок.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				107
Змн.	Арк.	№ докум.	Підпис	Дата		

Реалізований програмний продукт відповідає поставленим до нього вимогам та реалізує весь функціонал. За час роботи над проектом було виконано поставлені задачі та створено програмне забезпечення за обраною тематикою.

Протягом розробки веб-додатку було покращено наявні та отримано нові знання, вміння та навички з розробки веб-додатків, а саме отримано навички та вміння розробки серверних додатків веб-систем та знання з реалізації зв'язку між клієнтськими та серверними додатками систем, оплати товарів завдяки доступним сервісам тощо.

В подальшому дана система може бути суттєво покращена для відповідності до можливих нових вимог до програмного забезпечення, включно з функціональними, адаптовано до використання в мережі Інтернет, використано при побудові подібних веб-додатків у вигляді практичного прикладу та теоретичного досвіду, що отримано за час виконання роботи.

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмотря О.Г.				108
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Annieahujaweb2020. Three-Tier Client Server Architecture in Distributed System – GeeksforGeeks [Електронний ресурс] / Annieahujaweb2020. – 24 лютого 2022р. – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/three-tier-client-server-architecture-in-distributed-system>
2. Cloud computing for business – Business Queensland [Електронний ресурс] – 24 листопада 2022р. – Режим доступу до ресурсу: <https://www.business.qld.gov.au/running-business/digital-business/online-risk-security/cloud>
3. What is Express.js? | Why should use Express.js? | Features of Express.js – Besant Technologies [Електронний ресурс] – Режим доступу до ресурсу: <https://www.besanttechnologies.com/what-is-expressjs>
4. Deniz Colak. What are the Advantages of TypeScript over JavaScript? – Altogic [Електронний ресурс] / Deniz Colak. – 5 лютого 2023р. – Режим доступу до ресурсу: <https://www.altogic.com/blog/what-are-the-advantages-of-typescript-over-javascript>
5. Katie Lawson. What Is a Single Page Application? – Bloomreach [Електронний ресурс] / Katie Lawson. – 16 вересня 2022р. – Режим доступу до ресурсу: https://www.bloomreach.com/en/blog/2018/what-is-a-single-page-application?spz=learn_orig
6. What is Use Case Diagram? – Visual Paradigm [Електронний ресурс] – 2022р. – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>
7. What is Activity Diagram? – Visual Paradigm [Електронний ресурс] – 2022р. – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>

8. Nishadha. UML Class Diagram Relationships Explained with Examples – createely [Електронний ресурс] / Nishanda. – 25 листопада 2022р. – Режим доступу до ресурсу: <https://createely.com/guides/class-diagram-relationships/>

9. What is middleware? – Red Hat [Електронний ресурс] – 16 грудня 2022р. – Режим доступу до ресурсу: <https://www.redhat.com/en/topics/middleware/what-is-middleware>

10. Introduction to JSON Web Token – JWT [Електронний ресурс] – Режим доступу до ресурсу: <https://jwt.io/introduction>

11. What is Component Diagram? – Visual Paradigm [Електронний ресурс] – 2022р. – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/>

12. Why your brand's color palette matters – Pinckney Harmon [Електронний ресурс] – 2017р. – Режим доступу до ресурсу: <https://pinckneyharmon.com/brand-color-palette/>

13. Crafting multi-stage builds with Docker in Node.js – Cloudnweb [Електронний ресурс] – 5 жовтня 2019р. – Режим доступу до ресурсу: <https://www.cloudnweb.dev/2019/10/crafting-multi-stage-builds-with-docker-in-node-js/>

14. Devaadi. How to separate routers and controllers in Node.js? – GeeksforGeeks [Електронний ресурс] / Devaadi. – 6 жовтня 2022р. – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/how-to-separate-routers-and-controllers-in-node-js/>

15. Error Handling – Express [Електронний ресурс] – 2017р. – Режим доступу до ресурсу: <https://expressjs.com/en/guide/error-handling.html>

16. Bundler Polyfill Issues - Webpack 5 – Web3Auth [Електронний ресурс] – 2022р. – Режим доступу до ресурсу: <https://web3auth.io/docs/troubleshooting/webpack-issues#react>

17. Bhargav Bachina. How To Serve React Application With NGINX and Docker – Medium [Електронний ресурс] / Bhargav Bachina. – 30 грудня 2019р.

– Режим доступу до ресурсу: <https://medium.com/bb-tutorials-and-thoughts/how-to-serve-react-application-with-nginx-and-docker-9c51ac2c50ba>

18. Tim Mouskhelichvili. How To Extend An Interface In TypeScript? – Tim Mousk [Електронний ресурс] / Tim Mouskhelichvili. – 21 листопада 2022р.

– Режим доступу до ресурсу: <https://timmousk.com/blog/typescript-extend-interface/>

19. Payment widget – LiqPay Documentation [Електронний ресурс] – Режим доступу до ресурсу:

<https://www.liqpay.ua/en/documentation/api/acquiring/widget/doc>

20. Alexneo2003. react-liqpay – NPM [Електронний ресурс] / Alexneo2003. – 5 квітня 2020р. – Режим доступу до ресурсу:

<https://www.npmjs.com/package/react-liqpay>

21. Joseph Gefroh. Simplify web application deployment with Docker Compose – Medium [Електронний ресурс] / Joseph Gefroh. – 25 вересня 2017р.

– Режим доступу до ресурсу: <https://jgefroh.medium.com/simplify-web-application-deployment-with-docker-compose-194bba8ac5a7>

		Олексійчук М.В.			ІПЗ.КР.Б – 121 – 23 ПЗ	Арк.
		Чижмоторя О.Г.				111
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

Специфікація прецедентів

Таблиця А.1

Ім'я варіанту використання: Пошук та/або перегляд списку товарів, жанрів, компаній або публікацій у додатку		Рівень важливості: високий
Основний актор: користувач		
Короткий опис:		Перегляд наявних у системі товарів, ігрових жанрів та відеоігрових компаній, публікацій
Передумови:		Ініціалізація у системі
Пост-умови:		
Відносини:		
	Асоціація:	
	Включення:	Ініціалізація
	Розширення:	
	Узагальнення:	
Основний потік:		Користувач отримує на сторінці в залежності від запиту популярні та нові товари, списки жанрів або компаній та нові публікації з можливістю сортування та фільтрації даних. У випадку помилкового запиту користувача буде повідомлено на сторінці, а за помилки системи – надано контактні дані для зв'язку з адміністрацією.
Альтернативні/виключні потоки:		Поява спливаючого повідомлення про помилку та контактів адміністратора.

Таблиця А.2

Ім'я варіанту використання: Перегляд даних про запитані товар, жанр, компанію, публікацію		Рівень важливості: високий
Основний актор: користувач		
Короткий опис:		Перегляд детальних відомостей про товар, жанр, компанію або змісту публікації за запитом.
Передумови:		Ініціалізація у системі
Пост-умови:		
Відносини:		
	Асоціація:	
	Включення:	Ініціалізація
	Розширення:	
	Узагальнення:	
Основний потік:		Користувач отримує на сторінці дані про товар з можливістю додавання у кошик, жанр чи компанію зі списком пов'язаних товарів чи змісту публікації, відгуків до товару чи публікації. У випадку помилкового запиту користувача буде повідомлено на сторінці, а за помилки системи – надано контактні дані для зв'язку з адміністрацією.
Альтернативні/виключні потоки:		Поява спливаючого повідомлення про помилку та контактів адміністратора.

Таблиця А.3

Ім'я варіанту використання: Купівля товарів у кошику		Рівень важливості: високий
Основний актор: користувач		
Короткий опис:		Перегляд товарів у кошику та виконання дій над ним, включно із купівлею
Передумови:		Ініціалізація у системі
Пост-умови:		
Відносини:		
	Асоціація:	
	Включення:	Ініціалізація
	Розширення:	
	Узагальнення:	
Основний потік:		Користувач отримує на сторінці список товарів у власному кошику та можливість видалення з кошика та перегляду коротких відомостей про товар, купівлі залишених товарів. У випадку помилкового запиту користувача буде повідомлено на сторінці, а за помилки системи – надано контактні дані для зв'язку з адміністрацією.
Альтернативні/виключні потоки:		Поява спливаючого повідомлення про помилку та контактів адміністратора.

Ім'я варіанту використання: Створення відгуку до товару або публікації		Рівень важливості: високий
Основний актор: користувач		
Короткий опис:		Написання відгуку з оцінкою на сторінці товару/публікації.
Передумови:		Ініціалізація у системі
Пост-умови:		
Відносини:		
	Асоціація:	
	Включення:	Ініціалізація
	Розширення:	
	Узагальнення:	
Основний потік:		Користувач отримує змогу написання відгуку з певною оцінкою на сторінці запитаних товару чи публікації. Дані зберігаються та впливають на рейтинг товару чи публікації. У випадку помилкового запиту користувача буде повідомлено на сторінці, а за помилки системи – надано контактні дані для зв'язку з адміністрацією.
Альтернативні/виключні потоки:		Поява спливаючого повідомлення про помилку та контактів адміністратора.

Ім'я варіанту використання: Ховання товарів та публікацій, видалення даних		Рівень важливості: високий
Основний актор: модератор. адміністратор		
Короткий опис:		Керування існуючими даними товарів, жанрів, компаній або публікацій.
Передумови:		Ініціалізація у системі
Пост-умови:		
Відносини:		
	Асоціація:	
	Включення:	Ініціалізація
	Розширення:	
	Узагальнення:	
Основний потік:		Працівник отримує на адміністративній сторінці список даних за запитом з можливістю приховання, видалення або переходу до редагування. У випадку помилкового запиту користувача буде повідомлено на сторінці, за помилки системи – надано контактні дані для зв'язку з розробниками.
Альтернативні/виключні потоки:		Поява спливаючого повідомлення про помилку та контактів адміністратора.

Таблиця А.6

Ім'я варіанту використання: Створення чи редагування записів товарів, жанрів, компаній, публікацій у системі		Рівень важливості: високий
Основний актор: модератор. адміністратор		
Короткий опис:		Створення нових товарів, публікацій, редагування існуючих.
Передумови:		Ініціалізація у системі
Пост-умови:		
Відносини:		
	Асоціація:	
	Включення:	Ініціалізація
	Розширення:	
	Узагальнення:	
Основний потік:		Працівник отримує на адміністративній сторінці форму для заповнення або редагування даних за запитом. У випадку помилкового запиту працівника буде повідомлено на сторінці, за помилки системи – надано контактні дані для зв'язку з розробниками.
Альтернативні/виключні потоки:		Поява спливаючого повідомлення про помилку та контактів адміністратора.

Ім'я варіанту використання: Редагування ролей інших акторів		Рівень важливості: високий
Основний актор: адміністратор		
Короткий опис:		Зміна ролей у існуючих облікових записах.
Передумови:		Ініціалізація у системі
Пост-умови:		
Відносини:		
	Асоціація:	
	Включення:	Ініціалізація
	Розширення:	
	Узагальнення:	
Основний потік:		Адміністратор отримує список облікових записів з можливістю пошуку потрібного та зміни ролей між «користувач» та «модератор». У випадку помилкового запиту адміністратора буде повідомлено на сторінці, а за помилки системи – надано контактні дані для зв'язку з розробниками.
Альтернативні/виключні потоки:		Поява спливаючого повідомлення про помилку та контактів розробників.

Рисунки діаграм послідовності

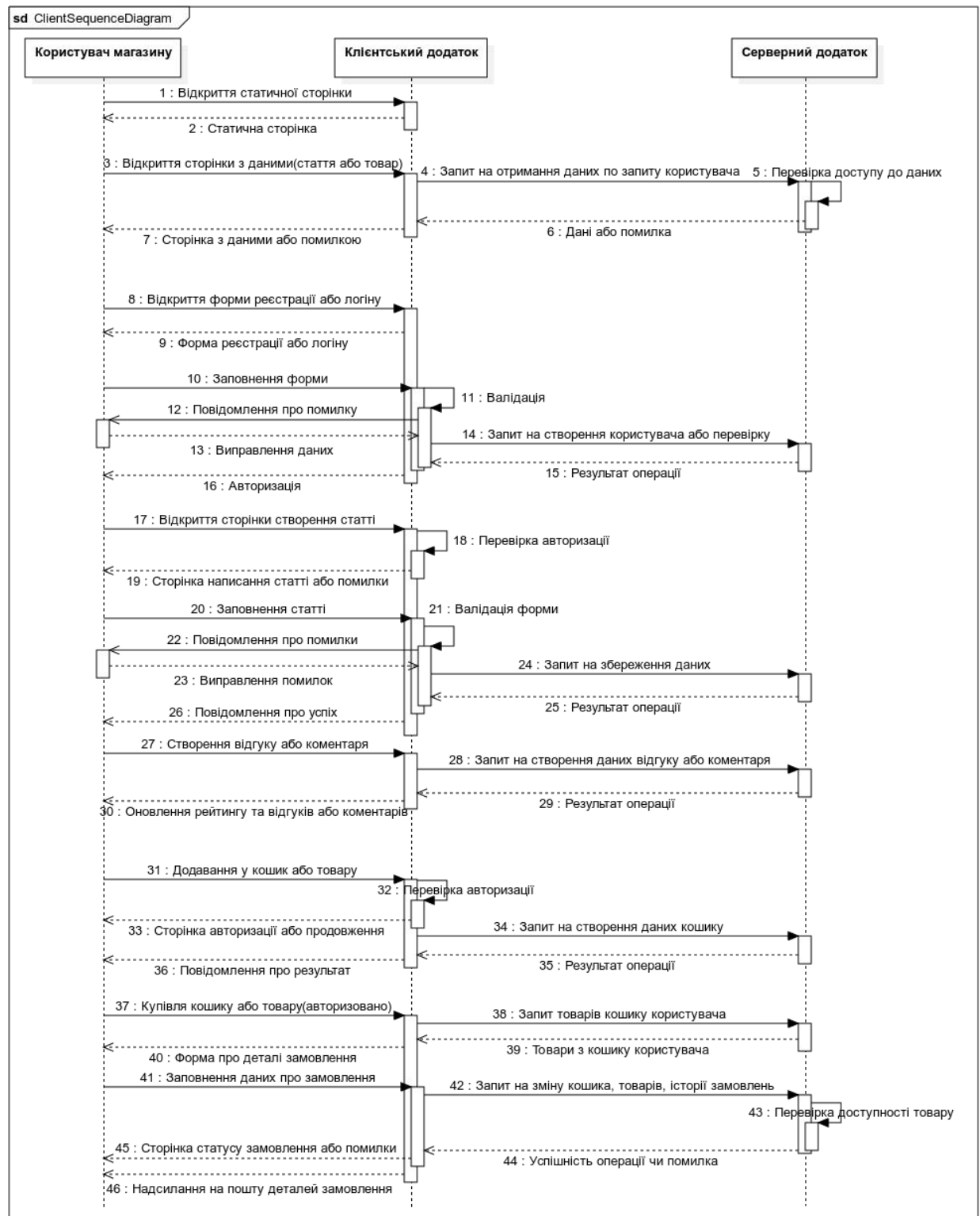


Рис.Б.1 – Діаграма послідовності дій звичайних Користувачів

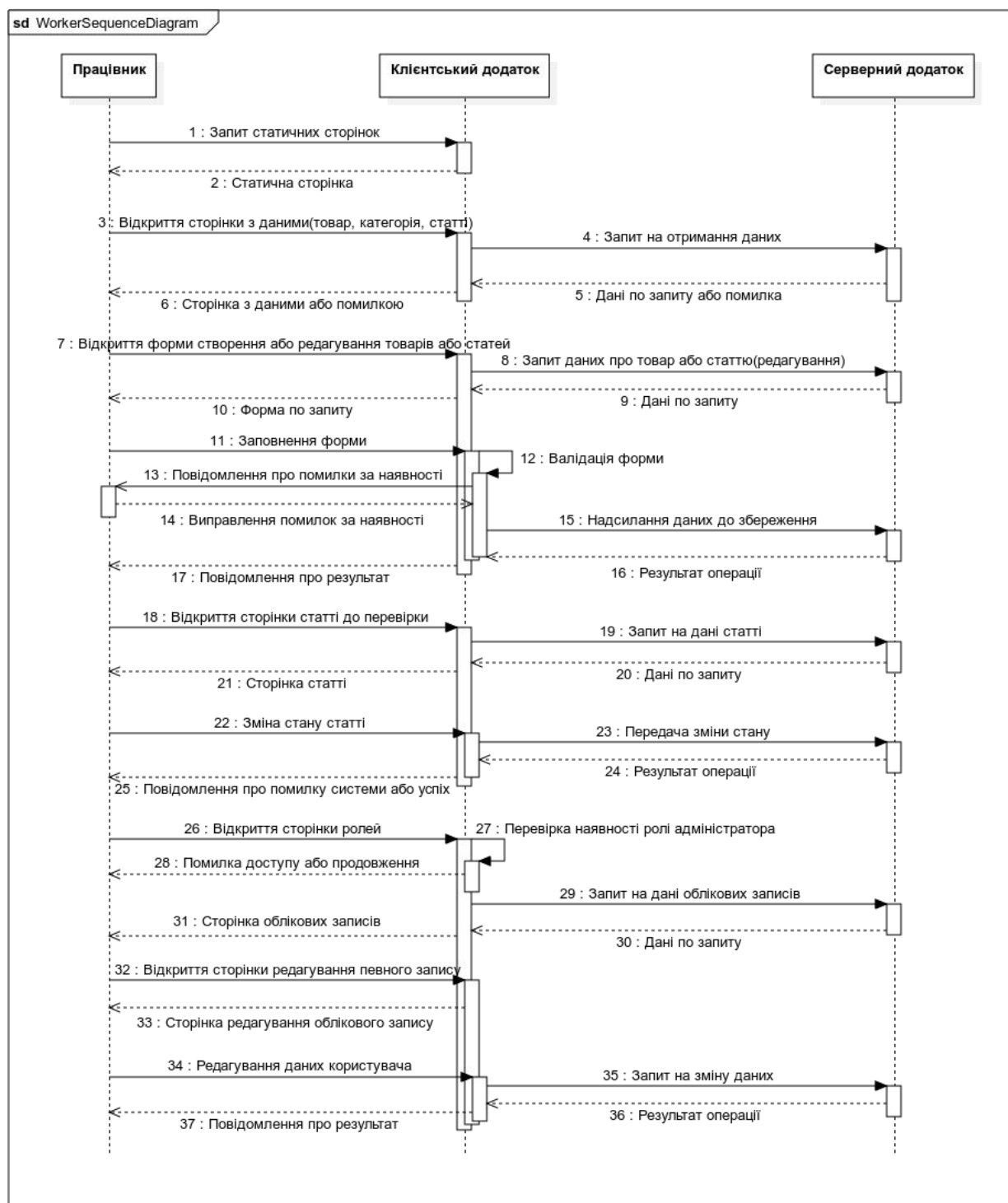


Рис.Б.2 – Діаграма послідовності дій Користувачів-працівників

Лістинг файлів моделей Sequelize та ініціалізації таблиць БД у теці models

Лістинг коду файлу Company.ts:

```

import Item from "../Item";

const {sequelize_db} = require('../db');
import {DataTypes, Op} from 'sequelize';

const Company = sequelize_db.define('company', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  name: {
    type: DataTypes.STRING,
    allowNull: false
  },
  description: {
    type: DataTypes.TEXT,
  },
  director: {
    type: DataTypes.STRING,
    allowNull: false,
    validate: {
      len: {args: [2, 20], msg: 'Поле "Директор" повинно бути довжиною від 2 до 20 символів'},
      is: {args: /^[A-Z][a-zA-Z\-' ]+ ?)*$|^([A-ЯЄІІГ][a-яА-ЯЄіііІг\-' ]+ ?)*$/,
        msg: 'Поле "Директор" повинно містити лише літери'}
    }
  },
  image: {
    type: DataTypes.STRING,
    allowNull: false,
    defaultValue: 'default.jpg'
  },
  founded: {
    type: DataTypes.DATE,
    allowNull: false
  },
  hide: {
    type: DataTypes.BOOLEAN,
    allowNull: false,
    defaultValue: false
  },
}, {
  paranoid: true,
});

const _whereHandler = (name?: string, description?: string,
  director?: string, founded?: Date, foundedFrom?: Date,
  foundedTo?: Date, includeHidden?: boolean) => {
  let where: {name?: {}, description?: {}, director?: {}, founded?: {}, hide?: {}} =
  {};
  if (name) {
    where.name = {
      [Op.iLike]: `>${name}%`
    }
  }
  if (description) {
    where.description = {
      [Op.iLike]: `>${description}%`
    }
  }
  if (director) {
    where.director = {
      [Op.iLike]: `>${director}%`
    }
  }

```

```

    }
  }
  if (founded) {
    where.founded = founded;
  }
  if (foundedFrom) {
    where.founded = {
      ...where.founded,
      [Op.gte]: foundedFrom
    }
  }
  if (foundedTo) {
    where.founded = {
      ...where.founded,
      [Op.lte]: foundedTo
    }
  }
  if (!includeHidden) {
    where.hide = false;
  }

  return where;
}

const _includeHandler = (includeItemsDeveloped: boolean, includeItemsPublished:
boolean, includeHidden: boolean) => {
  let include: {}[] = [];
  const where: {hide?: {}} = {};
  if (!includeHidden) {
    where.hide = false;
  }

  if (includeItemsDeveloped) {
    include.push({
      model: Item,
      as: 'ItemsDeveloped',
      through: {attributes: []}, // exclude junction table from result
      where,
      required: false
    });
  }
  if (includeItemsPublished) {
    include.push({
      model: Item,
      as: 'ItemsPublished',
      where,
      required: false
    });
  }

  return include;
}

type getAllCompaniesParams = {
  name?: string,
  description?: string,
  director?: string,
  founded?: Date,
  foundedFrom?: Date,
  foundedTo?: Date,
  descending?: boolean,
  limit?: number,
  page?: number,
  sortBy?: string,
  includeItemsDeveloped?: boolean,
  includeItemsPublished?: boolean,
  includeHidden?: boolean
}

const getCompanies = async ({name, description, director, founded, foundedFrom,
foundedTo, descending = false,
                                limit = 10, page = 0, sortBy = 'name', includeItemsDeveloped
= false,
                                includeItemsPublished = false, includeHidden = false}:

```

```

getAllCompaniesParams) => {
  const where = _whereHandler(name, description, director, founded, foundedFrom,
foundedTo, includeHidden);
  const include = _includeHandler(includeItemsDeveloped, includeItemsPublished,
includeHidden);
  const totalCount = await Company.count({ where: includeHidden ? {} : { hide:
false } } );
  const companies = await Company.findAll({
    where,
    limit,
    offset: page * limit,
    order: [[sortBy, descending ? "DESC" : "ASC"]],
    include,
  });

  return {
    companies,
    totalCount
  };
};

type getOneCompanyParams = {
  id: number,
  includeItemsDeveloped?: boolean,
  includeItemsPublished?: boolean,
  includeHidden?: boolean
}

const getCompany = async ({id, includeItemsDeveloped = true, includeItemsPublished =
true, includeHidden = false}: getOneCompanyParams) => {
  const where = {
    id,
    hide: {
      [Op.in]: [includeHidden, false]
    }
  }
  const include = _includeHandler(includeItemsDeveloped, includeItemsPublished,
includeHidden);
  return Company.findOne({where, include});
}

export default Company;
export {
  Company,
  getCompanies,
  getCompany
};

```

Лістинг коду файлу Genre.ts:

```

import Item from "../Item";

const {sequelize_db} = require('../db');
import {DataTypes, Op} from 'sequelize';

const Genre = sequelize_db.define('genre', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  name: {
    type: DataTypes.STRING,
    allowNull: false
  },
  description: {
    type: DataTypes.TEXT,
    allowNull: true
  },
  hide: {
    type: DataTypes.BOOLEAN,
    allowNull: false,
    defaultValue: false
  }
});

```

```

    },
  }, {
    paranoid: true,
  });

const _whereHandler = (includeHidden: boolean, name?: string, description?: string) => {
  let where: {name?: {}, description?: {}, hide?: {}} = {};
  if (name) {
    where.name = {
      [Op.iLike]: `%${name}%`
    }
  }
  if (description) {
    where.description = {
      [Op.iLike]: `%${description}%`
    }
  }
  if (!includeHidden) {
    where.hide = false;
  }

  return where;
}

const _includeHandler = (includeItems: boolean, includeHidden: boolean) => {
  let include: {}[] = [];
  const where: {hide?: {}} = {};
  if (!includeHidden) {
    where.hide = false;
  }

  if (includeItems) {
    include.push({
      model: Item,
      as: 'Items',
      through: {attributes: []},
      where,
      required: false
    });
  }

  return include;
}

type getAllGenresParams = {
  name?: string,
  description?: string,
  descending?: boolean,
  limit?: number,
  page?: number,
  sortBy?: string,
  includeItems?: boolean,
  includeHidden?: boolean
}

const getGenres = async ({name, description, descending = false, limit = 10, page = 0,
  sortBy = 'name', includeItems = false, includeHidden = false}: getAllGenresParams) => {
  const where = _whereHandler(includeHidden, name, description);
  const include = _includeHandler(includeItems, includeHidden);
  const totalCount = await Genre.count({ where: includeHidden ? {} : { hide: false } });
  const genres = await Genre.findAll({
    where, limit, offset: page * limit, order: [[sortBy, descending ? "DESC" : "ASC"]], include
  });

  return {
    genres,
    totalCount,
  };
}

type getOneGenreParams = {

```

```

    id: number | string,
    includeItems?: boolean,
    includeHidden?: boolean
  }
  const getGenre = async ({id, includeItems = true, includeHidden = false}:
  getOneGenreParams) => {
    const where: {id: number | string, hide?: boolean} = {id};
    if (!includeHidden) {
      where.hide = false;
    }

    const include = _includeHandler(includeItems, includeHidden);
    return Genre.findOne({ where, include });
  }

export default Genre;
export {
  Genre,
  getGenres,
  getGenre
}

```

Лістинг коду файлу Item.ts:

```

import { Company, Genre, ItemRate, User } from "../index";
import { DataTypes, Op } from 'sequelize';
import ApiError from "../errors/ApiError";
const {sequelize_db} = require('../db');

const Item = sequelize_db.define('item', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  name: {
    type: DataTypes.STRING,
    allowNull: false
  },
  description: {
    type: DataTypes.TEXT,
  },
  releaseDate: {
    type: DataTypes.DATEONLY,
    allowNull: false
  },
  price: {
    type: DataTypes.FLOAT,
    allowNull: false
  },
  amount: {
    type: DataTypes.INTEGER,
    allowNull: false,
    defaultValue: 0,
    validate: {
      min: {args: [0], msg: 'Поле "Кількість" повинно бути більше 0'}
    }
  },
  discount: {
    type: DataTypes.BOOLEAN,
    allowNull: false,
    defaultValue: false
  },
  discountFrom: {
    type: DataTypes.DATEONLY,
    allowNull: true
  },
  discountTo: {
    type: DataTypes.DATEONLY,
    allowNull: true,
  },
},

```

```

discountSize: {
  type: DataTypes.FLOAT,
  allowNull: true
},
mainImage: {
  type: DataTypes.STRING,
  allowNull: false,
  defaultValue: 'default.png'
},
images: {
  type: DataTypes.ARRAY(DataTypes.STRING),
  allowNull: false,
  defaultValue: []
},
coverImage: {
  type: DataTypes.STRING,
  allowNull: true,
},
characteristics: {
  type: DataTypes.JSON,
  allowNull: true,
  defaultValue: {}
},
hide: {
  type: DataTypes.BOOLEAN,
  allowNull: false,
  defaultValue: false
},
}, {
  paranoid: true,
});

const _whereHandler = (name?: string, description?: string,
  releaseDate?: Date, releaseDateFrom?: Date, releaseDateTo?: Date,
  price?: number, priceFrom?: number, priceTo?: number,
  amount?: number, amountFrom?: number, amountTo?: number,
  discount?: boolean, discountFrom?: Date, discountTo?: Date,
  discountSize?: number, discountSizeFrom?: number, discountSizeTo?:
number,
  hide = true) => {
  let where: {name?: {}, description?: {}, releaseDate?: {}, releaseDateFrom?: {},
releaseDateTo?: {},
  price?: {}, discount?: {}, discountFrom?: {}, discountTo?: {},
  amount?: {}, amountFrom?: {}, amountTo?: {},
  discountSize?: {}, company_publisherId?: {}, hide?: {}} = {};

  if (name) {
    where.name = {
      [Op.iLike]: `%${name}%`
    }
  }
  if (description) {
    where.description = {
      [Op.iLike]: `%${description}%`
    }
  }

  if (releaseDate) {
    where.releaseDate = releaseDate;
  }
  if (releaseDateFrom) {
    where.releaseDate = {
      ...where.releaseDate,
      [Op.gte]: releaseDateFrom
    };
  }
  if (releaseDateTo) {
    where.releaseDate = {
      ...where.releaseDate,
      [Op.lte]: releaseDateTo
    };
  }
};

```

```

    }

    if (price) {
      if (price < 0) throw ApiError.badRequest('Ціна не може бути меншою за 0');
      where.price = {
        [Op.eq]: price
      }
    }

    if (priceFrom) {
      if (priceFrom < 0) throw ApiError.badRequest('Ціна не може бути меншою за 0');
      where.price = {
        ...where.price,
        [Op.gte]: priceFrom
      };
    }

    if (priceTo) {
      if (priceTo < 0) throw ApiError.badRequest('Ціна не може бути меншою за 0');
      where.price = {
        ...where.price,
        [Op.lte]: priceTo
      };
    }

    if (amount) {
      if (amount < 0) throw ApiError.badRequest('Кількість не може бути меншою за 0');
      where.amount = amount;
    }

    if (amountFrom) {
      if (amountFrom < 0) throw ApiError.badRequest('Кількість не може бути меншою за
0');
      where.amount = {
        ...where.amount,
        [Op.gte]: amountFrom
      };
    }

    if (amountTo) {
      if (amountTo < 0) throw ApiError.badRequest('Кількість не може бути меншою за
0');
      where.amount = {
        ...where.amount,
        [Op.lte]: amountTo
      };
    }

    if (discount !== undefined) {
      where.discount = discount
    }

    if (discountFrom) {
      where.discountFrom = {
        [Op.gte]: discountFrom
      };
    }

    if (discountTo) {
      where.discountTo = {
        [Op.lte]: discountTo
      };
    }

    if (discountSize) {
      if (discountSize < 0 || discountSize > 100)
        throw ApiError.badRequest('Розмір знижки не може бути меншим за 0 або більшим
за 100');
      where.discountSize = discountSize;
    }

    if (discountSizeFrom) {
      if (discountSizeFrom < 0 || discountSizeFrom > 100)
        throw ApiError.badRequest('Розмір знижки не може бути меншим за 0 або більшим
за 100');
      where.discountSize = {
        ...where.discountSize,
        [Op.gte]: discountSizeFrom
      };
    }

```

```

    };
  }
  if (discountSizeTo) {
    if (discountSizeTo < 0 || discountSizeTo > 100)
      throw ApiError.badRequest('Розмір знижки не може бути меншим за 0 або більшим
за 100');
    where.discountSize = {
      ...where.discountSize,
      [Op.lte]: discountSizeTo
    };
  }
  where.hide = {
    [Op.or]: [false, hide]
  }

  return where;
}
let _includeHandler = (includePublisher: boolean, includeGenres: boolean,
includeDevelopers: boolean,
includeWishlisted: boolean, includeInCart: boolean,
includeRated: boolean, includeHidden: boolean,
publisherId?: number, genresIds?: number[], developersIds?: number[])
=> {
  let include: {}[] = [];

  if (includePublisher) {
    let where = {};

    if (!includeHidden) {
      where = {
        hide: false
      }
    }
    if (publisherId) {
      where = {
        ...where,
        id: publisherId
      }
    }

    include.push({
      model: Company,
      as: 'Publisher',
      where,
      required: !!publisherId
    });
  }

  if (includeGenres) {
    let where = {};
    let required = false;

    where = {
      hide: {
        [Op.or]: [false, includeHidden]
      }
    }
    if (genresIds) {
      where = {
        ...where,
        id: {[Op.in]: genresIds}
      }
      required = genresIds.length > 0
    }

    include.push({
      model: Genre,
      as: 'Genres',
      where,
      through: {
        attributes: []
      }
    });
  }
}

```

```

        },
        required
    ));
}

if (includeDevelopers) {
    let where = {};
    let required = false;

    if (!includeHidden) {
        where = {
            hide: false
        }
    }
    if (developersIds) {
        where = {
            ...where,
            id: {[Op.in]: developersIds}
        }
        required = developersIds.length > 0
    }

    include.push({
        model: Company,
        as: 'Developers',
        where,
        through: {
            attributes: []
        },
        required
    });
}

if (includeWishlisted) {
    include.push({
        model: User,
        attributes: {exclude: ['password']},
        as: 'WishlistedUsers',
        through: { attributes: [] },
        required: false
    });
}

if (includeInCart) {
    include.push({
        model: User,
        as: 'CartedUsers',
        attributes: {exclude: ['password']},
        through: {
            attributes: []
        },
        required: false
    });
}

if (includeRated) {
    include.push({
        model: ItemRate,
        as: 'Rates',
        include: [{
            model: User,
            as: 'User',
            attributes: {exclude: ['password']},
        }],
        where: {
            hide: {
                [Op.or]: [false, includeHidden]
            },
            violation: {
                [Op.or]: [false, includeHidden]
            }
        }
    });
}

```

```

        },
        required: false
    });
}

return include;
}
type getAllItemsParams = {
    name?: string,
    description?: string,
    releaseDate?: Date,
    releaseDateFrom?: Date,
    releaseDateTo?: Date,
    price?: number,
    priceFrom?: number,
    priceTo?: number,
    amount?: number,
    amountFrom?: number,
    amountTo?: number,
    discount?: boolean,
    discountFrom?: Date,
    discountTo?: Date,
    discountSize?: number,
    discountSizeFrom?: number,
    discountSizeTo?: number,
    descending?: boolean,
    limit?: number,
    page?: number,
    sortBy?: string,
    includePublisher?: boolean,
    publisherId?: number,
    includeGenres?: boolean,
    genresIds?: number[],
    includeDevelopers?: boolean,
    developersIds?: number[],
    includeWishlisted?: boolean,
    includeInCart?: boolean,
    includeRated?: boolean,
    includeHidden?: boolean
}
const getItems = async ({name, description, releaseDate, releaseDateFrom,
releaseDateTo,
                        price, priceFrom, priceTo, amount, amountFrom, amountTo,
                        discount, discountFrom, discountTo, discountSize,
discountSizeFrom, discountSizeTo,
                        descending = false, limit = 10, page = 0, sortBy = 'id',
                        includePublisher = false, publisherId,
                        includeGenres = false, genresIds,
                        includeDevelopers = false, developersIds,
                        includeWishlisted = false, includeInCart = false,
                        includeRated = true, includeHidden = false}: getAllItemsParams)
=> {
    const where = _whereHandler(name, description, releaseDate, releaseDateFrom,
releaseDateTo,
                        price, priceFrom, priceTo, amount, amountFrom, amountTo,
                        discount, discountFrom, discountTo, discountSize, discountSizeFrom,
discountSizeTo, includeHidden)
    const include = _includeHandler(includePublisher, includeGenres, includeDevelopers,
includeWishlisted, includeInCart, includeRated, includeHidden,
publisherId, genresIds, developersIds);
    const totalCount = await Item.count({where: includeHidden ? {} : { hide: false }});
    const items = await Item.findAll({
        where, limit, offset: limit * page, order: [[sortBy, descending ? "DESC" :
"ASC"]], include
    });

    console.log({
        where, limit, offset: limit * page, order: [[sortBy, descending ? "DESC" :
"ASC"]], include
    })
}

```

```

    return {
      items,
      totalCount,
    };
  }
}
type getOneItemParams = {
  id: number,
  includePublisher?: boolean,
  includeGenres?: boolean,
  includeDevelopers?: boolean,
  includeWishlisted?: boolean,
  includeInCart?: boolean,
  includeRated?: boolean,
  includeHidden?: boolean
}
const getItem = async ({id, includePublisher = true, includeGenres = true,
  includeDevelopers = true,
    includeWishlisted = false, includeInCart = false, includeRated =
false, includeHidden = false}: getOneItemParams) => {
  const where = {
    id,
    hide: {
      [Op.or]: [includeHidden, false]
    }
  }

  const include = _includeHandler(includePublisher, includeGenres, includeDevelopers,
    includeWishlisted, includeInCart, includeRated, includeHidden);

  return Item.findOne({where, include});
}

const getDiscountItems = async () => {
  const date = Date.now();
  return Item.findAll({
    where: {
      discount: true,
      discountFrom: {
        [Op.lte]: date
      },
      discountTo: {
        [Op.gte]: date
      }
    }
  });
}

export default Item;
export {
  Item,
  getItems,
  getItem,
  getDiscountItems
}

```

Лістинг коду файлу ItemBought.ts:

```

const {sequelize_db} = require('../db');
import {DataTypes} from 'sequelize';

const ItemBought = sequelize_db.define('item_bought', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
  itemId: {type: DataTypes.INTEGER, allowNull: false},
  transactionId: {type: DataTypes.STRING, allowNull: false},
}, {
  paranoid: true,
});

const getItemsBought = async (userId?: number, itemId?: number) => {
  let where: {userId?: {}, itemId?: {}} = {};

```

```

    if (userId) {
      where.userId = userId;
    }
    if (itemId) {
      where.itemId = itemId;
    }

    return ItemBought.findAll({where});
  }

export default ItemBought;
export {
  ItemBought,
  getItemBought
}

```

Лістинг коду файлу ItemCart.ts:

```

const {sequelize_db} = require('../db');
import {DataTypes} from 'sequelize';

const ItemCart = sequelize_db.define('item_cart', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
  transactionId: {type: DataTypes.STRING, allowNull: true},
});

const getCart = async (userId?: number, itemId?: number) => {
  let where: {userId?: {}, itemId?: {}} = {};

  if (userId) {
    where.userId = userId;
  }
  if (itemId) {
    where.itemId = itemId;
  }

  return ItemCart.findAll({where});
}

export default ItemCart;
export {
  ItemCart,
  getCart
}

```

Лістинг коду файлу ItemDevelopers.ts:

```

const {sequelize_db} = require('../db');
import {DataTypes} from 'sequelize';

const ItemDevelopers = sequelize_db.define('item_developer', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
});

const getItemDevelopers = async (itemId?: number, developerId?: number) => {
  let where: {itemId?: {}, developerId?: {}} = {};

  if (itemId) {
    where.itemId = itemId;
  }
  if (developerId) {
    where.developerId = developerId;
  }

  return ItemDevelopers.findAll({where});
}

export default ItemDevelopers;
export {
  ItemDevelopers,
  getItemDevelopers
}

```

Лістинг коду файлу ItemGenre.ts:

```
const {sequelize_db} = require('../db');
import {DataTypes} from 'sequelize';

const ItemGenre = sequelize_db.define('item_genre', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
});
const getItemGenres = async (itemId?: number, genreId?: number) => {
  let where: {itemId?: {}, genreId?: {}} = {};

  if (itemId) {
    where.itemId = itemId;
  }
  if (genreId) {
    where.genreId = genreId;
  }

  return ItemGenre.findAll({where});
}

export default ItemGenre;
export {
  ItemGenre,
  getItemGenres
}
```

Лістинг коду файлу ItemRate.ts:

```
import ApiError from "../errors/ApiError";

const {sequelize_db} = require('../db');
import {DataTypes, Op} from 'sequelize';
import Item from "../Item";
import User from "../User";

const ItemRate = sequelize_db.define('item_rate', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  content: {
    type: DataTypes.TEXT,
    allowNull: true
  },
  rate: {
    type: DataTypes.INTEGER,
    allowNull: false
  },
  hide: {
    type: DataTypes.BOOLEAN,
    allowNull: false,
    defaultValue: false
  },
  violation: {
    type: DataTypes.BOOLEAN,
    allowNull: false,
    defaultValue: false
  },
  violation_reason: {
    type: DataTypes.TEXT,
    allowNull: true
  },
});
type getAllItemRatesParams = {
  itemId?: number,
  content?: string,
  rate?: number,
  rateFrom?: number,
  rateTo?: number,
```

```

    hide?: boolean,
    violation?: boolean,
    violationReason?: string
  }
  const getItemRates = async ({itemId, content, rate, rateFrom, rateTo, hide, violation,
    violationReason}: getAllItemRatesParams) => {
    let where: {itemId?: {}, content?: {}, rate?: {}, hide?: {}, violation?: {},
    violationReason?: {}} = {};

    if (itemId) {
      where.itemId = itemId;
    }
    if (content) {
      where.content = {
        [Op.like]: `%${content}%`
      };
    }
    if (rate) {
      if (rate < 1 || rate > 5) throw ApiError.badRequest('Оцінка повинна бути від 1
до 5');
      where.rate = rate;
    }
    if (rateFrom) {
      if (rateFrom < 1 || rateFrom > 5) throw ApiError.badRequest('Оцінка повинна бути
від 1 до 5');
      where.rate = {
        ...where.rate,
        [Op.gte]: rateFrom
      };
    }
    if (rateTo) {
      if (rateTo < 1 || rateTo > 5) throw ApiError.badRequest('Оцінка повинна бути від
1 до 5');
      where.rate = {
        ...where.rate,
        [Op.lte]: rateTo
      };
    }
    where.hide = hide;
    where.violation = violation;
    if (violation && violationReason) {
      where.violationReason = {
        [Op.like]: `%${violationReason}%`
      };
    }

    return ItemRate.findAll({where});
  }

  type getAllRatesParams = {
    descending?: boolean,
    limit?: number,
    page?: number,
    sortBy?: string,
  }
  export const getAllRates = async ({descending = true, limit = 12, page = 1, sortBy =
"createdAt": getAllRatesParams) => {
    if (page < 1) throw ApiError.badRequest('Сторінка повинна бути більша за 0');
    if (limit < 1) throw ApiError.badRequest('Ліміт повинен бути більший за 0');

    const totalCount = await ItemRate.count();
    const rates = await ItemRate.findAll({
      order: [[sortBy, descending ? 'DESC' : 'ASC']],
      limit,
      offset: (page - 1) * limit,
      include: [
        {
          model: Item,
          as: 'Item'
        },
      ],
    });
  }

```

```

        model: User,
        as: 'User',
        attributes: {exclude: ['password']}
      }
    ]
  });

  return {
    totalCount,
    rates,
  }
};

export default ItemRate;
export {
  ItemRate,
  getItemRates
}

```

Лістинг коду файлу Publication.ts:

```

const {sequelize_db} = require('../db');
import {DataTypes, Op} from 'sequelize';
import {PublicationComment, User} from "../index";

const Publication = sequelize_db.define('publication', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  title: {
    type: DataTypes.STRING,
    allowNull: false
  },
  content: {
    type: DataTypes.TEXT,
    allowNull: false
  },
  hide: {
    type: DataTypes.BOOLEAN,
    allowNull: false,
    defaultValue: false
  },
  tags: {
    type: DataTypes.ARRAY(DataTypes.STRING),
    allowNull: false,
    defaultValue: []
  },
  violation: {
    type: DataTypes.BOOLEAN,
    allowNull: false,
    defaultValue: false
  },
  violation_reason: {
    type: DataTypes.TEXT,
    allowNull: true
  },
}, {
  paranoid: true,
});

const _whereHandler = (title?: string, content?: string,
  createdAt?: Date, createdFrom?: Date, createdTo?: Date,
  includeHidden = false, tags?: string[],
  violation = false, violationReason?: string) => {
  let where: {title?: {}, content?: {}, createdAt?: {}, hide?: {}, violation?: {},
  violationReason?: {}, tags?: {}} = {};

  console.log({
    title,

```

```

        content,
        createdAt,
        createdFrom,
        createdTo,
        includeHidden,
        violation,
        violationReason,
        tags,
    })

    if (title) {
        where.title = {
            [Op.iLike]: `%${title}%`
        };
    }
    if (content) {
        where.content = {
            [Op.iLike]: `%${content}%`
        };
    }
    if (!includeHidden) {
        where.hide = includeHidden;
    }
    where.violation = violation;
    if (violation && violationReason) {
        where.violationReason = {
            [Op.like]: `%${violationReason}%`
        };
    }
    if (createdAt) {
        where.createdAt =
            sequelize_db.where(sequelize_db.fn('date',
sequelize_db.col('createdAt')), createdAt);
    }
    if (createdFrom) {
        where.createdAt = {
            ...where.createdAt,
            [Op.gte]: createdFrom
        };
    }
    if (createdTo) {
        createdTo.setDate(createdTo.getDate() + 1);
        where.createdAt = {
            ...where.createdAt,
            [Op.lte]: createdTo
        };
    }
    if (tags && tags.length > 0) {
        console.log(tags)
        where.tags = {
            [Op.contains]: tags
        };
    }

    return where;
}

const _includeHandler = (includeComments: boolean, includeViolations: boolean,
    includeHidden: boolean, authorsIds?: number[]) => {
    let include: {}[] = [];

    include.push({
        model: User,
        as: 'AuthoredUser',
        attributes: {exclude: ['password']},
    })

    if (includeComments) {
        let where = {};

        if (!includeHidden) {
            where = {
                hide: false
            }
        }
    }

```

```

    };
  }
  if (!includeViolations) {
    where = {
      ...where,
      violation: false,
    };
  }

  include.push({
    model: PublicationComment,
    as: 'CommentsList',
    include: [{
      model: User,
      as: 'User',
      attributes: {exclude: ['password']},
    }],
    required: false,
    where
  });
}

if (authorsIds && authorsIds.length > 0) {
  include.push({
    model: User,
    as: 'AuthoredUser',
    attributes: {exclude: ['password']},
    where: {
      id: {
        [Op.in]: authorsIds
      }
    }
  });
}

return include;
}
type getAllPublicationsParams = {
  title?: string,
  content?: string,
  createdAt?: Date,
  createdFrom?: Date,
  createdTo?: Date,
  descending?: boolean,
  limit?: number,
  page?: number,
  sortBy?: string,
  tags?: string[],
  authorsIds?: number[],
  includeComments?: boolean,
  includeViolations?: boolean,
  includeHidden?: boolean,
  violationReason?: string
}
const getPublications = async ({title, content, createdAt, createdFrom, createdTo,
  descending = false,
  limit = 10, page = 0, sortBy = 'createdAt', tags = [],
  authorsIds = [],
  includeComments = true, includeViolations = false,
  includeHidden = false,
  violationReason}: getAllPublicationsParams) => {
  console.log({
    title,
    content,
    createdAt,
    createdFrom,
    createdTo,
    descending,
    limit,
    page,
    sortBy,

```

```

        tags,
        authorsIds,
    });
    const where = _whereHandler(title, content, createdAt, createdFrom, createdTo,
includeHidden, tags, includeViolations, violationReason);
    const include = _includeHandler(includeComments, includeViolations, includeHidden,
authorsIds);
    const totalCount = await Publication.count({where: includeHidden ? {} : { hide:
false }});
    const publications = await Publication.findAll({
        where, limit, offset: page * limit, order: [[sortBy, descending ? 'DESC' :
'ASC']], include
    });

    return {
        publications,
        totalCount,
    };
}
type getOnePublicationParams = {
    id: number,
    includeComments?: boolean,
    includeViolations?: boolean,
    includeHidden?: boolean
}
const getPublication = async ({id, includeComments = true, includeViolations = false,
includeHidden = false}: getOnePublicationParams) => {
    const include = _includeHandler(includeComments, includeViolations, includeHidden);
    return await Publication.findByPk(id, {include});
}

export default Publication;
export {
    Publication,
    getPublications,
    getPublication,
}

```

Лістинг коду файлу PublicationComment.ts:

```

import ApiError from "../errors/ApiError";

const {sequelize_db} = require('../db');
import {DataTypes, Op} from 'sequelize';
import { Publication } from "../index";
import User from "../User";

const PublicationComment = sequelize_db.define('publication_comment', {
    id: {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true
    },
    content: {
        type: DataTypes.TEXT,
        allowNull: false
    },
    rate: {
        type: DataTypes.INTEGER,
        allowNull: false
    },
    hide: {
        type: DataTypes.BOOLEAN,
        allowNull: false,
        defaultValue: false
    },
    violation: {
        type: DataTypes.BOOLEAN,
        allowNull: false,
        defaultValue: false
    },
},

```

```

    violation_reason: {
      type: DataTypes.TEXT,
      allowNull: true
    },
  });
type getAllPublicationCommentsParams = {
  publicationId?: number,
  content?: string,
  rate?: number,
  rateFrom?: number,
  rateTo?: number,
  hide?: boolean,
  violation?: boolean,
  violationReason?: string
}
const getPublicationComments = async ({publicationId, content, rate, rateFrom, rateTo,
hide, violation, violationReason}: getAllPublicationCommentsParams) => {
  let where: {publicationId?: {}, content?: {}, rate?: {}, hide?: {},
    violation?: {}, violationReason?: {}} = {};

  if (publicationId) {
    where.publicationId = publicationId;
  }
  if (content) {
    where.content = {
      [Op.like]: `%${content}%`
    };
  }
  if (rate) {
    if (rate < 1 || rate > 5) throw ApiError.badRequest('Оцінка повинна бути від 1
до 5');
    where.rate = rate;
  }
  if (rateFrom) {
    if (rateFrom < 1 || rateFrom > 5) throw ApiError.badRequest('Оцінка повинна бути
від 1 до 5');
    where.rate = {
      ...where.rate,
      [Op.gte]: rateFrom
    };
  }
  if (rateTo) {
    if (rateTo < 1 || rateTo > 5) throw ApiError.badRequest('Оцінка повинна бути від
1 до 5');
    where.rate = {
      ...where.rate,
      [Op.lte]: rateTo
    };
  }
  where.hide = hide;
  where.violation = violation;
  if (violation && violationReason) {
    where.violationReason = {
      [Op.like]: `%${violationReason}%`
    };
  }
}

return PublicationComment.findAll({where});
}

type getAllCommentsParams = {
  descending?: boolean,
  limit?: number,
  page?: number,
  sortBy?: string,
}
export const getAllComments = async ({descending = true, limit = 12, page = 1, sortBy
= "createdAt": getAllCommentsParams) => {
  if (page < 1) throw ApiError.badRequest('Сторінка повинна бути більша за 0');
  if (limit < 1) throw ApiError.badRequest('Ліміт повинен бути більший за 0');

```

```

const totalCount = await PublicationComment.count();
const comments = await PublicationComment.findAll({
  order: [[sortBy, descending ? 'DESC' : 'ASC']],
  limit,
  offset: (page - 1) * limit,
  include: [
    {
      model: Publication,
      as: 'Publication',
    },
    {
      model: User,
      as: 'User',
      attributes: {exclude: ['password']}
    }
  ]
});

return {
  totalCount,
  comments,
}
};

export default PublicationComment;
export {
  PublicationComment,
  getPublicationComments
}

```

Лістинг коду файлу User.ts:

```

import ItemRate from "../ItemRate";
import PublicationComment from "../PublicationComment";
import Publication from "../Publication";
import Item from "../Item";

const {sequelize_db} = require('../db');
import {DataTypes, Op} from 'sequelize';
import { ItemBought } from "../index";

const User = sequelize_db.define('user', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  login: {
    type: DataTypes.STRING,
    unique: true,
    allowNull: false,
    validate: {
      len: {args: [4, 20], msg: 'Поле "Логін" повинно бути довжиною від 4 до 20 символів'},
      isAlphanumeric: {msg: 'Поле "Логін" повинно містити лише літери латинського алфавіту та цифри'},
    }
  },
  email: {
    type: DataTypes.STRING,
    unique: true,
    allowNull: false,
    validate: {
      isEmail: {msg: 'Поле "Email" повинно бути валідним email-адресом'},
    }
  },
  image: {
    type: DataTypes.STRING,
    allowNull: false,
    defaultValue:
'https://res.cloudinary.com/dnqlgypji/image/upload/v1686310061/gigashop/users/r7mocr34

```

```

rjlgfissqkaf.jpg'
},
firstName: {
  type: DataTypes.STRING,
  allowNull: false,
  validate: {
    len: {args: [2, 20], msg: 'Поле "Ім\''я" повинно бути довжиною від 2 до 20 символів'},
    is: {args: /^[A-Z][a-zA-Z\-' ]+$|^[A-ЯЄІІІГ][a-яА-ЯЄііііІг\-' ]+$/, msg: 'Поле "Ім\''я" повинно містити лише літери'}
  }
},
lastName: {
  type: DataTypes.STRING,
  allowNull: false,
  validate: {
    len: {args: [2, 20], msg: 'Поле "Прізвище" повинно бути довжиною від 2 до 20 символів'},
    is: {args: /^[A-Z][a-zA-Z\-' ]+$|^[A-ЯЄІІІГ][a-яА-ЯЄііііІг\-' ]+$/, msg: 'Поле "Прізвище" повинно містити лише літери'}
  }
},
role: {
  type: DataTypes.ENUM('USER', 'MODERATOR', 'ADMIN'),
  defaultValue: 'USER',
  allowNull: false
},
password: {
  type: DataTypes.STRING,
  allowNull: false
},
isBlocked: {
  type: DataTypes.BOOLEAN,
  defaultValue: false
},
}, {
  paranoid: true,
});

const _whereHandler = (login?: string, email?: string, firstName?: string, lastName?: string,
                      role?: string, isBlocked = false) => {
  let where: {login?: {}, email?: {}, firstName?: {}, lastName?: {}, role?: {}, isBlocked?: {}} = {};

  if (login) {
    where.login = {
      [Op.like]: `%${login}%`
    }
  }
  if (email) {
    where.email = {
      [Op.like]: `%${email}%`
    }
  }
  if (firstName) {
    where.firstName = {
      [Op.like]: `%${firstName}%`
    }
  }
  if (lastName) {
    where.lastName = {
      [Op.like]: `%${lastName}%`
    }
  }
  if (role) {
    where.role = {
      [Op.like]: `%${role}%`
    }
  }
  if (isBlocked) {

```

```

        where.isBlocked = {
            [Op.or]: [isBlocked, false]
        };
    }

    return where;
}

const __includeHandler = (includeBoughtItems: boolean, includeCart: boolean,
includeItemsRates: boolean,
                        includeWishlist: boolean, includePublications: boolean,
includePublicationComments: boolean) => {
    let include: {}[] = [];

    if (includeCart) {
        include.push({
            model: Item,
            as: 'Cart',
            through: {attributes: ["transactionId"]},
            required: false
        });
    }

    console.log({ includeBoughtItems });
    if (includeBoughtItems) {
        include.push({
            model: ItemBought,
            as: 'Bought',
            required: false
        });
    }

    if (includeItemsRates) {
        include.push({
            model: ItemRate,
            as: 'Rates',
            include: [{
                model: Item,
                as: 'Item',
            }],
            required: false
        });
    }

    if (includeWishlist) {
        include.push({
            model: Item,
            as: 'Wishlist',
            through: {attributes: []},
            required: false
        });
    }

    if (includePublications) {
        include.push({
            model: Publication,
            as: 'Publications',
            required: false
        });
    }

    if (includePublicationComments) {
        include.push({
            model: PublicationComment,
            as: 'CommentsList',
            required: false
        });
    }

    return include;
}

```

```

type getUsersParams = {
  login?: string,
  email?: string,
  firstName?: string,
  lastName?: string,
  role?: string,
  isBlocked?: boolean,
  sortBy?: string,
  descending?: boolean,
  page?: number,
  limit?: number
}

const getUsers = async ({login, email, firstName, lastName, role, isBlocked = true,
limit = 10, page = 1, sortBy = 'login', descending = false}: getUsersParams) => {
  const where = _whereHandler(login, email, firstName, lastName, role, isBlocked);
  const totalCount = await User.count();
  const users = await User.findAll({
    where,
    limit,
    offset: (page - 1) * limit,
    order: [[sortBy, descending ? 'DESC' : 'ASC']],
    attributes: {exclude: ['password']}
  });

  return {
    users,
    totalCount,
  };
}

type getUserParams = {
  id: number,
  includeBoughtItems?: boolean,
  includeCart?: boolean,
  includeItemsRates?: boolean,
  includeWishlist?: boolean,
  includePublications?: boolean,
  includePublicationComments?: boolean
}

const getUser = async ({ id, includeBoughtItems = false, includeCart = false,
includeItemsRates = false,
includeWishlist = false, includePublications = false,
includePublicationComments = false }: getUserParams) => {
  let include = _includeHandler(includeBoughtItems, includeCart, includeItemsRates,
includeWishlist,
includePublications, includePublicationComments);

  const user = await User.findPk(id, { include, attributes: {exclude:
['password']} })
  console.log({ bought: user.Bought });
  const ids = user.Bought ? user.Bought.map((item: any) => item.itemId) : [];

  const boughtItems = ids.length > 0 ? await Item.findAll({
    where: {
      id: {
        [Op.in]: ids
      }
    }
  }) : [];

  return { user, boughtItems };
}

export default User;
export {
  User,
  getUsers,
  getUser
}

```

Лістинг коду файлу Wishlist.ts:

```
const {sequelize_db} = require('../db');
import {DataTypes} from 'sequelize';

const Wishlist = sequelize_db.define('item_favorite', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
});
const getWishlists = async (userId?: number | number[], itemId?: number | number[]) =>
{
  let where: {userId?: {}, itemId?: {}} = {};

  if (userId) {
    where.userId = userId;
  }
  if (itemId) {
    where.itemId = itemId;
  }

  return Wishlist.findAll({where});
}

export default Wishlist;
export {
  Wishlist,
  getWishlists
}
```

Лістинг коду файлу index.ts:

```
import User from "./User";
import Item from "./Item";
import ItemRate from "./ItemRate";
import ItemGenre from "./ItemGenre";
import ItemBought from "./ItemBought";
import Wishlist from "./Wishlist";
import ItemCart from "./ItemCart";
import ItemDevelopers from "./ItemDevelopers";
import Company from "./Company";
import Genre from "./Genre";
import Publication from "./Publication";
import PublicationComment from "./PublicationComment";

function initModels() {
  User.belongsToMany(Item, {through: ItemCart, foreignKey: 'userId', as: 'Cart',
onDelete: 'CASCADE'});
  Item.belongsToMany(User, {through: ItemCart, foreignKey: 'itemId', as: 'CartedUsers',
onDelete: 'CASCADE'});

  User.belongsToMany(Item, {through: Wishlist, foreignKey: 'userId', as: 'Wishlist',
onDelete: 'CASCADE'});
  Item.belongsToMany(User, {through: Wishlist, foreignKey: 'itemId', as:
'WishlistedUsers', onDelete: 'CASCADE'});

  User.hasMany(ItemBought, {foreignKey: 'userId', as: 'Bought', onDelete: 'CASCADE'});
  ItemBought.belongsTo(User, {foreignKey: 'userId', as: 'User', onDelete: 'CASCADE'});

  Company.hasMany(Item, {foreignKey: 'company_publisherId', as: 'ItemsPublished',
onDelete: 'CASCADE'});
  Item.belongsTo(Company, {foreignKey: 'company_publisherId', as: 'Publisher',
onDelete: 'CASCADE'});

  Company.belongsToMany(Item, {through: ItemDevelopers, as: 'ItemsDeveloped', onDelete:
'CASCADE'});
  Item.belongsToMany(Company, {through: ItemDevelopers, as: 'Developers', onDelete:
'CASCADE'});

  Genre.belongsToMany(Item, {through: ItemGenre, as: 'Items', onDelete: 'CASCADE'});
  Item.belongsToMany(Genre, {through: ItemGenre, as: 'Genres', onDelete: 'CASCADE'});

  ItemRate.belongsTo(User, {foreignKey: 'userId', as: 'User', onDelete: 'CASCADE'});
```

```

    User.hasMany(ItemRate, {foreignKey: 'userId', as: 'Rates', onDelete: 'CASCADE'});
    ItemRate.belongsTo(Item, {foreignKey: 'itemId', as: 'Item', onDelete: 'CASCADE'});
    Item.hasMany(ItemRate, {foreignKey: 'itemId', as: 'Rates', onDelete: 'CASCADE'});

    PublicationComment.belongsTo(User, {foreignKey: 'userId', as: 'User', onDelete:
'CASCADE'});
    User.hasMany(PublicationComment, {foreignKey: 'userId', as: 'CommentsList', onDelete:
'CASCADE'});
    PublicationComment.belongsTo(Publication, {foreignKey: 'publicationId', as:
'Publication', onDelete: 'CASCADE'});
    Publication.hasMany(PublicationComment, {foreignKey: 'publicationId', as:
'CommentsList', onDelete: 'CASCADE'});

    User.hasMany(Publication, {foreignKey: 'userId', as: 'Publications'});
    Publication.belongsTo(User, {foreignKey: 'userId', as: 'AuthoredUser'});
}

export default initModels;
export {
  User,
  Item,
  ItemRate,
  ItemGenre,
  ItemBought,
  Wishlist,
  ItemDevelopers,
  Company,
  Genre,
  Publication,
  PublicationComment,
};

```

Лістинг файлів маршрутизації папки routes та відповідних контролерів папки controllers серверного додатку

Лістинг коду Controller.ts:

```
import type { NextFunction, Request, Response } from 'express';
import { ValidationError as SequelizeValidationError } from "sequelize";
import ApiError from "../errors/ApiError";
import path from "path";
import { Company, Genre, Item, ItemRate, Publication, PublicationComment, User,
Wishlist } from "../models";

export default class Controller {
  protected exceptionHandle(error: unknown): ApiError {
    if (error instanceof SequelizeValidationError) {
      console.log(error);
      if (error.name === 'SequelizeUniqueConstraintError') {
        return ApiError.badRequest("Запис із даними значеннями вже існує",
error.errors);
      }
      return ApiError.badRequest(error.name, error.errors);
    }
    if (error instanceof ApiError) {
      console.log({ return: "ApiError" });
      return error;
    }
    if (error instanceof Error) {
      console.log({ error, isError: error instanceof Error });
      return ApiError.internal(error.message);
    }
    return ApiError.internal("Помилка обробки запиту");
  }

  // ToDo: change file deletion from local static to cloud storage
  // Delete file from static folder
  protected deleteFile(dir: string, file: string) {
    const fs = require('fs');

    if(!fs.existsSync(path.resolve('./static', dir, file))) return;

    fs.unlink(path.resolve('./static', dir, file), (err: unknown) => {
      if (err) {
        console.error(err);
      }
    });
  }

  protected parseDate(date: string | undefined): Date | undefined {
    if (!date) return undefined;
    return new Date(date);
  }

  protected parseNumber(number: string | undefined): number | undefined {
    if (number === undefined) return undefined;
    return Number(number);
  }

  protected parseBoolean(boolean?: boolean | string | number): boolean | undefined {
    if (boolean === undefined) return undefined;
    if (!boolean) return false;
    return [true, 'true', 'True', 'on', 'yes', '1', 1].includes(boolean);
  }

  protected parsePagination(desc: string | boolean | number | undefined,
    descending: string | boolean | number | undefined,
    limit: string | undefined, page: string | undefined):
```

```

    { descending: boolean | undefined, limit: number | undefined, page: number |
undefined }
    {
        let controller = new Controller();

        let descendingParsed: boolean | undefined;
        if (desc) descendingParsed = controller.parseBoolean(desc as boolean | string |
number | undefined);
        else descendingParsed = controller.parseBoolean(descending as boolean | string |
number | undefined);

        let limitParsed = controller.parseNumber(limit as string | undefined);
        let pageParsed = Math.max(controller.parseNumber(page as string | undefined) ||
1, 1);

        return { descending: descendingParsed, limit: limitParsed, page: pageParsed -
1 };
    }

    async statistics(_: Request, res: Response, next: NextFunction) {
        try {
            const Users = await User.findAll();
            const Items = await Item.findAll({
                include: [
                    {
                        model: Company,
                        as: 'Developers',
                        through: { attributes: [] },
                    },
                    {
                        model: Company,
                        as: 'Publisher',
                    },
                    {
                        model: Genre,
                        as: 'Genres',
                        through: { attributes: [] },
                    },
                ],
            });
            const Publications = await Publication.findAll();
            const Companies = await Company.findAll();
            const Genres = await Genre.findAll();
            const ItemsRates = await ItemRate.findAll();
            const PublicationsComments = await PublicationComment.findAll();
            const Wishlists = await Wishlist.findAll();

            return res.json({
                Users,
                Items,
                Publications,
                Companies,
                Genres,
                ItemsRates,
                PublicationsComments,
                Wishlists
            });
        }
        catch (e) {
            return next(this.exceptionHandle(e));
        }
    }
}

const controller = new Controller();

export {
    controller
}

```

Лістинг коду файлу CommentsRoutes.ts:

```
import {Router} from 'express';
import ItemsController from "../controllers/Shop/ItemsController";
import NewsController from "../controllers/NewsController";

const commentsRouter = Router();

commentsRouter.use('/items', ItemsController.getAllRates);
commentsRouter.use('/news', NewsController.getAllComments);

module.exports = commentsRouter;
```

Лістинг коду файлу ModerationRoutes.ts:

```
import {Router} from 'express';
import { checkWorker } from "../middleware/CheckRoleMiddleware";
import ItemsController from "../controllers/Shop/ItemsController";
import NewsController from "../controllers/NewsController";
import { controller } from "../controllers/Controller";

const moderationRouter = Router();

moderationRouter.patch('/news/:id', checkWorker, NewsController.toggleViolation);
moderationRouter.patch('/comments/:id', checkWorker, NewsController.toggleCommentViolation);

moderationRouter.patch('/rates/:id', checkWorker, ItemsController.toggleViolation);

moderationRouter.get('/statistics', checkWorker, controller.statistics);

module.exports = moderationRouter;
```

Лістинг коду файлу CompaniesRoutes.ts:

```
import {Router} from 'express';
import CompaniesController from "../controllers/Shop/CompaniesController";
import {checkWorker} from "../middleware/CheckRoleMiddleware";

const companiesRouter = Router()

companiesRouter.post('/', checkWorker, CompaniesController.create);
companiesRouter.patch('/:id', checkWorker, CompaniesController.update);
companiesRouter.delete('/:id', checkWorker, CompaniesController.delete);

companiesRouter.get('/', CompaniesController.getAll);
companiesRouter.get('/:id', CompaniesController.getOne);

module.exports = companiesRouter
```

Лістинг коду файлу CompaniesController.ts:

```
import type {NextFunction, Request, Response} from 'express';
import ApiError from "../../errors/ApiError";
import Controller from "../Controller";
import {Company, getCompanies, getCompany} from "../../models/Company";

class CompaniesController extends Controller {
  async create(req: Request, res: Response, next: NextFunction) {
    try {
      const {name, description, director, image} = req.body;
      const founded = super.parseDate(req.body.founded);
      const hide = super.parseBoolean(req.body.hide) || false;

      const company = await Company
        .create({ name, description, director, image, founded, hide })
        .catch((e: unknown) => {
          // super.deleteFile(COMPANIES_DIR, imageName);
          return next(super.exceptionHandle(e));
        });
    }
  }
}
```

```

        if (!company) {
            // super.deleteFile(COMPANIES_DIR, imageName);
            return next(ApiError.badRequest('Компанію не створено'));
        }

        return res.json(company);
    }
    catch (e: unknown) {
        return next(super.exceptionHandle(e));
    }
}

async getAll(req: Request, res: Response, next: NextFunction) {
    let {
        name,
        description,
        director,
        desc,
        descending,
        limit,
        page,
        sortBy,
        includeHidden } = req.query;

    const founded = super.parseDate(req.query.founded as string | undefined);
    const foundedFrom = super.parseDate(req.query.foundedFrom as string | undefined);
    const foundedTo = super.parseDate(req.query.foundedTo as string | undefined);
    const hideParsed = super.parseBoolean(includeHidden as string | undefined) ||
false;

    let {descending: descendingParsed, limit: limitParsed, page: pageParsed} =
super.parsePagination(desc as string | undefined, descending as string |
undefined,
        limit as string | undefined, page as string | undefined);

    const result = await getCompanies({
        name: name as string | undefined, description: description as string |
undefined,
        director: director as string | undefined, founded, foundedFrom, foundedTo,
        descending: descendingParsed, limit: limitParsed, page: pageParsed, sortBy:
sortBy as string | undefined,
        includeHidden: hideParsed
    })
    .catch((e: unknown) => {
        return next(super.exceptionHandle(e));
    });
    if (!result) return next(ApiError.badRequest('Компаній не знайдено'));

    const { companies, totalCount } = result;

    if (!companies) return next(ApiError.badRequest('Компаній не знайдено'));
    return res.json({ companies, totalCount });
}

async getOne(req: Request, res: Response, next: NextFunction) {
    const { id } = req.params;
    const includeItemsDeveloped = super.parseBoolean(req.query.includeDeveloped as
string | undefined);
    const includeItemsPublished = super.parseBoolean(req.query.includePublished as
string | undefined);
    const includeHidden = super.parseBoolean(req.query.includeHidden as string |
undefined) || false;

    const company = await getCompany({
        id: +id, includeItemsDeveloped, includeItemsPublished, includeHidden
    });

    if (!company) return next(ApiError.badRequest('Компанію не знайдено'));
    return res.json(company);
}

```

```

    async update(req: Request, res: Response, next: NextFunction) {
      const { id } = req.params;
      const { name, description, director, founded, hide, image } = req.body;

      const company = await Company.findById(id)
        .catch((e: unknown) => {
          return next(super.exceptionHandle(e));
        });
      if (!company) return next(ApiError.badRequest('Компанію не знайдено'));

      // let oldImage = company.image;

      let hideParsed = super.parseBoolean(hide as string | undefined);
      let foundedParsed = super.parseDate(founded);

      if (name) company.name = name;
      if (description) company.description = description;
      if (director) company.director = director;
      if (founded) company.founded = foundedParsed;
      if (hide !== undefined) company.hide = hideParsed;
      console.log({ hide, hideParsed });
      if (image) company.image = image;

      let result = await company.save()
        .catch((e: unknown) => {
          // if (image) super.deleteFile(COMPANIES_DIR, image);
          return next(super.exceptionHandle(e));
        });

      if (!result) {
        // if (image) super.deleteFile(COMPANIES_DIR, image);
        return next(ApiError.badRequest('Компанію не оновлено'));
      }

      // ToDo: delete files
      //
      // if (image) super.deleteFile(COMPANIES_DIR, oldImage);
      return res.json(company);
    }

    async delete(req: Request, res: Response, next: NextFunction) {
      const { id } = req.params;

      const company = await Company
        .findById(id)
        .catch((e: unknown) => {
          return next(super.exceptionHandle(e));
        });
      // const { image } = company;

      const result = await company.destroy()
        .catch((e: unknown) => {
          return next(super.exceptionHandle(e));
        });

      if (!result) return next(ApiError.badRequest('Компанію не видалено'));

      // ToDo: delete files
      //
      // super.deleteFile(COMPANIES_DIR, image);
      return res.json({ ok: true });
    }

    async test(req: Request, res: Response) {
      return res.json({message: `Companies route works!`, request: {body: req.body,
        query: req.query}})
    }
  }
  export default new CompaniesController();

```

Лістинг коду файлу GenresRoutes.ts:

```
import {Router} from 'express';
import GenresController from "../controllers/Shop/GenresController";
import {checkWorker} from "../middleware/CheckRoleMiddleware";

const genresRouter = Router()

genresRouter.post('/', checkWorker, GenresController.create);
genresRouter.patch('/:id', checkWorker, GenresController.update);
genresRouter.delete('/:id', checkWorker, GenresController.delete);

genresRouter.get('/', GenresController.getAll);
genresRouter.get('/:id', GenresController.getOne);

module.exports = genresRouter
```

Лістинг коду файлу GenresController.ts:

```
import type { Request, Response, NextFunction } from 'express';
import Controller from '../Controller';
import {Genre} from "../../models";
import {getGenre, getGenres} from "../../models/Genre";
import ApiError from "../../errors/ApiError";

class GenresController extends Controller {
  async create(req: Request, res: Response, next: NextFunction) {
    try {
      const {name, description} = req.body;
      const hide = super.parseBoolean(req.body.hide) || false;

      const genre = await Genre
        .create({name, description, hide})
        .catch((e: unknown) => {
          return next(super.exceptionHandle(e));
        })
    };

    if (!genre) {
      return next(ApiError.badRequest('Жанр не створено'));
    }
    return res.json(genre);
  }
  catch (e: unknown) {
    return next(super.exceptionHandle(e));
  }
}

async getAll(req: Request, res: Response, next: NextFunction) {
  let { name,
    description,
    desc,
    descending,
    limit,
    page,
    sortBy,
    hidden } = req.query;

  const hideParsed = super.parseBoolean(hidden as string | undefined) || false;
  let {descending: descendingParsed, limit: limitParsed, page: pageParsed} =
    super.parsePagination(desc as string | undefined, descending as string |
undefined,
      limit as string | undefined, page as string | undefined);

  const result = await getGenres({
    name: name as string | undefined, description: description as string |
undefined,
    descending: descendingParsed, limit: limitParsed, page: pageParsed, sortBy:
sortBy as string | undefined,
    includeHidden: hideParsed
  })
}
```

```

        .catch((e: unknown) => {
            return next(super.exceptionHandle(e));
        })
    );
    if (!result) return next(ApiError.badRequest('Жанрів не знайдено'));

    const { genres, totalCount } = result;

    if (!genres) return next(ApiError.badRequest('Жанрів не знайдено'));
    return res.json({ genres, totalCount });
}

async getOne(req: Request, res: Response, next: NextFunction) {
    const { id } = req.params;

    const includeItems = super.parseBoolean(req.query.includeItems as string | undefined);
    const hideParsed = super.parseBoolean(req.query.includeHidden as string | undefined);

    const genre = await getGenre({ id: id, includeItems, includeHidden: hideParsed });
    if (!genre) return next(ApiError.badRequest('Жанр не знайдено'));
    return res.json(genre);
}

async update(req: Request, res: Response, next: NextFunction) {
    const { id } = req.params;
    const { name, description, hide } = req.body;

    let hideParsed = super.parseBoolean(hide as string | undefined);

    const genre = await Genre.findById(id);
    if (!genre) return next(ApiError.badRequest('Жанр не знайдено'));

    if (name) genre.name = name;
    if (description) genre.description = description;
    if (hideParsed !== undefined) genre.hide = hideParsed;

    const result = await genre
        .save()
        .catch((e: unknown) => {
            return next(super.exceptionHandle(e));
        });

    if (!result) return next(ApiError.badRequest('Жанр не оновлено'));
    return res.json(genre);
}

async delete(req: Request, res: Response, next: NextFunction) {
    const { id } = req.params;

    const genre = await Genre
        .destroy({where: {id}})
        .catch((e: unknown) => {
            return next(super.exceptionHandle(e));
        })
    );
    if (!genre) return next(ApiError.badRequest('Жанр не видалено'));
    return res.json({ ok: true });
}

async test(req: Request, res: Response) {
    return res.json({message: `Genres route works!`, request: {body: req.body, query: req.query}})
}

export default new GenresController();

```

Лістинг коду файлу ItemsRoutes.ts:

```
import {Router} from 'express';
import ItemsController from "../controllers/Shop/ItemsController";
import {checkWorker} from "../middleware/CheckRoleMiddleware";
import auth from "../middleware/AuthMiddleware";

const itemsRouter = Router()

itemsRouter.post('/', checkWorker, ItemsController.create);
itemsRouter.patch('/:id', checkWorker, ItemsController.update);
itemsRouter.delete('/:id', checkWorker, ItemsController.delete);

itemsRouter.post('/:id/toggleWishlist', auth, ItemsController.toggleWishlist);
itemsRouter.post('/:id/toggleCart', auth, ItemsController.toggleCart);

itemsRouter.post('/:id/rate', auth, ItemsController.setRate);
itemsRouter.delete('/:id/rate', auth, ItemsController.removeRate);

itemsRouter.post('/:id/developers', checkWorker, ItemsController.addDevelopers);
itemsRouter.delete('/:id/developers', checkWorker, ItemsController.removeDevelopers);
itemsRouter.post('/:id/genres', checkWorker, ItemsController.addGenres);
itemsRouter.delete('/:id/genres', checkWorker, ItemsController.removeGenres);

itemsRouter.get('/', ItemsController.getAll);
itemsRouter.get('/:id', ItemsController.getOne);

module.exports = itemsRouter
```

Лістинг коду файлу ItemsController.ts:

```
import type {NextFunction, Request, Response} from 'express';
import Controller from '../Controller';
import Item, {getItem, getItems} from "../../models/Item";
import ApiError from "../../errors/ApiError";
import { User, Wishlist, ItemDevelopers, Genre, ItemGenre, Company } from
"../../models";
import ItemCart from "../../models/ItemCart";
import ItemRate, { getAllRates } from "../../models/ItemRate";
import { Op } from "sequelize";

type parseOutput = {
  name: string | undefined, description: string | undefined, sortBy: string |
undefined,
  price: number | undefined, priceFrom: number | undefined, priceTo: number |
undefined,
  releaseDate: Date | undefined, releaseDateFrom: Date | undefined, releaseDateTo:
Date | undefined,
  amount: number | undefined, amountFrom: number | undefined, amountTo: number |
undefined,
  discount: boolean | undefined, discountFrom: Date | undefined, discountTo: Date |
undefined,
  discountSize: number | undefined, discountSizeFrom: number | undefined,
discountSizeTo: number | undefined,
  descending: boolean | undefined, limit: number | undefined, page: number | undefined,
includePublisher: boolean | undefined, publisherId: number | undefined,
includeGenres: boolean | undefined, genresIds: number[] | undefined,
includeDevelopers: boolean | undefined, developersIds: number[] | undefined,
includeWishlisted: boolean | undefined, includeInCart: boolean | undefined,
includeRated: boolean | undefined,
includeHidden: boolean | undefined, hide: boolean | undefined, hidden: boolean |
undefined
}

class ItemsController extends Controller {
  static parseData({name, description, sortBy,
    price, priceFrom, priceTo,
    releaseDate, releaseDateFrom, releaseDateTo,
    amount, amountFrom, amountTo,
    discount, discountFrom, discountTo,
    discountSize, discountSizeFrom, discountSizeTo,
```

```

        desc, descending, limit, page,
        includePublisher, publisherId,
        includeGenres, genresIds,
        includeDevelopers, developersIds,
        includeWishlisted, includeInCart,
        includeRated, hidden}: any): parseOutput {
let Controller = new ItemsController();

price = Controller.parseNumber(price as string | undefined);
priceFrom = Controller.parseNumber(priceFrom as string | undefined);
priceTo = Controller.parseNumber(priceTo as string | undefined);

releaseDate = Controller.parseDate(releaseDate as string | undefined);
releaseDateFrom = Controller.parseDate(releaseDateFrom as string | undefined);
releaseDateTo = Controller.parseDate(releaseDateTo as string | undefined);

amount = Controller.parseNumber(amount as string | undefined);
amountFrom = Controller.parseNumber(amountFrom as string | undefined);
amountTo = Controller.parseNumber(amountTo as string | undefined);

discount = Controller.parseBoolean(discount as boolean | string | number |
undefined);
if (discount) {
    discountFrom = Controller.parseDate(discountFrom as string | undefined);
    discountTo = Controller.parseDate(discountTo as string | undefined);
    discountSize = Controller.parseNumber(discountSize as string | undefined);
    discountSizeFrom = Controller.parseNumber(discountSizeFrom as string |
undefined);
    discountSizeTo = Controller.parseNumber(discountSizeTo as string | undefined);
}

let {descending: descendingParsed, limit: limitParsed, page: pageParsed} =
    Controller.parsePagination(desc as string | undefined, descending as string |
undefined,
        limit as string | undefined, page as string | undefined);

hidden = Controller.parseBoolean(hidden as boolean | string | number |
undefined);

if (includePublisher) {
    includePublisher = Controller.parseBoolean(includePublisher as boolean |
string | number | undefined);
}
if (includeGenres) {
    includeGenres = Controller.parseBoolean(includeGenres as boolean | string |
number | undefined);
}
if (includeDevelopers) {
    includeDevelopers = Controller.parseBoolean(includeDevelopers as boolean |
string | number | undefined);
}

if (includeWishlisted) {
    includeWishlisted = Controller.parseBoolean(includeWishlisted as boolean |
string | number | undefined);
}
if (includeInCart) {
    includeInCart = Controller.parseBoolean(includeInCart as boolean | string |
number | undefined);
}
if (includeRated) {
    includeRated = Controller.parseBoolean(includeRated as boolean | string |
number | undefined);
}

publisherId = Controller.parseNumber(publisherId as string | undefined);
if (typeof developersIds === 'string') {
    developersIds = [developersIds];
}
if (typeof genresIds === 'string') {
    genresIds = [genresIds];
}

```

```

    }

    return {
      name, description, sortBy,
      price, priceFrom, priceTo,
      releaseDate, releaseDateFrom, releaseDateTo,
      amount, amountFrom, amountTo,
      discount, discountFrom, discountTo,
      discountSize, discountSizeFrom, discountSizeTo,
      descending: descendingParsed, limit: limitParsed, page: pageParsed,
      includePublisher, publisherId,
      includeGenres, genresIds,
      includeDevelopers, developersIds,
      includeWishlisted, includeInCart,
      includeRated,
      includeHidden: hidden, hide: hidden, hidden
    }
  }
}

private static async _checkDevelopers(id?: number | string, developersIds?: number |
string | (number | string)[]) {
  if (!id) throw ApiError.badRequest('Неправильний id товару');
  const item = await Item.findByPk(id);
  if (!item) throw ApiError.notFound('Товар не знайдено');

  if (!developersIds) throw ApiError.badRequest('Неправильні дані розробників');
  let developers: unknown[] | unknown | undefined;

  if (developersIds instanceof Array) developers = await Company.findAll({where:
{id: developersIds}});
  else developers = await Company.findByPk(developersIds);

  if (!developers) throw ApiError.notFound('Розробника(-ів) не знайдено');
}

static async addDevelopers(id: number | string, developersIds: number | string |
(number | string)[]) {
  let candidates = await ItemDevelopers.findAll({where: {itemId: id, companyId:
developersIds}});
  if (candidates.length > 0) throw ApiError.badRequest('Товар вже має такого(-их)
розробника(-ів)', candidates);

  await this._checkDevelopers(id, developersIds)
    .catch((e: unknown) => {
      throw e;
    });

  if (developersIds instanceof Array) {
    let data = developersIds.map((devId: number | string) => {
      return {itemId: id, companyId: devId};
    });
    return await ItemDevelopers.bulkCreate(data)
      .catch((e: unknown) => {
        throw e;
      });
  }

  return await ItemDevelopers.create({itemId: id, companyId: developersIds})
    .catch((e: unknown) => {
      throw e;
    });
}

static async removeDevelopers(id: number | string, developersIds?: number | string |
(number | string)[]) {
  let where: {itemId: number | string, companyId?: number | string | (number |
string)[]} = {itemId: id};
  if (developersIds) where.companyId = developersIds;

  return await ItemDevelopers.destroy({where})
    .catch((e: unknown) => {
      throw e;
    });
}

```

```

    }
    static async setDevelopers(id: number | string, developersIds: number | string |
(number | string)[]) {
        await this._checkDevelopers(id, developersIds)
            .catch((e: unknown) => {
                throw e;
            });

        await ItemDevelopers.destroy({where: {itemId: id}})
            .catch((e: unknown) => {
                throw e;
            });

        if (developersIds instanceof Array) {
            let data = developersIds.map((devId: number | string) => {
                return {itemId: id, companyId: devId};
            });

            return await ItemDevelopers.bulkCreate(data)
                .catch((e: unknown) => {
                    throw e;
                });
        }

        return await ItemDevelopers.create({itemId: id, companyId: developersIds})
            .catch((e: unknown) => {
                throw e;
            });
    }

    private static async _checkGenres(id?: number | string, genresIds?: number | string
| (number | string)[]) {
        if (!id) throw ApiError.badRequest('Неправильний id товару');
        const item = await Item.findByPk(id);
        if (!item) throw ApiError.notFound('Товар не знайдено');

        if (!genresIds) throw ApiError.badRequest('Неправильні дані жанрів');
        let genres: unknown[] | unknown | undefined;

        if (genresIds instanceof Array) genres = await Genre.findAll({where: {id:
genresIds}});
        else genres = await Genre.findByPk(genresIds);

        if (!genres) throw ApiError.notFound('Жанр(-и) не знайдено');
    }

    static async addGenres(id: number | string, genresIds: number | string | (number |
string)[]) {
        let candidates = await ItemGenre.findAll({where: {itemId: id, genreId:
genresIds}});
        if (candidates.length > 0) throw ApiError.badRequest('Товар вже має такий(-и)
жанр(и)', candidates);

        await this._checkGenres(id, genresIds)
            .catch((e: ApiError | Error | unknown) => {
                throw e;
            });

        if (genresIds instanceof Array) {
            let data = genresIds.map((genreId: number | string) => {
                return {itemId: id, genreId: genreId};
            });

            return await ItemGenre.bulkCreate(data)
                .catch((e: unknown) => {
                    throw e;
                });
        }

        return await ItemGenre.create({itemId: id, genreId: genresIds})
            .catch((e: unknown) => {
                throw e;
            });
    }

```

```

    });
  }
  static async removeGenres(id: number | string, genresIds?: number | string | (number | string)[]) {
    let where: {itemId: number | string, genreId?: number | string | (number | string)[]} = {itemId: id};
    if (genresIds) where.genreId = genresIds;

    return await ItemGenre.destroy({where})
      .catch((e: unknown) => {
        throw e;
      });
  }

  static async setGenres(id: number | string, genresIds: number | string | (number | string)[]) {
    await this._checkGenres(id, genresIds)
      .catch((e: unknown) => {
        throw e;
      });

    await ItemGenre.destroy({where: {itemId: id}})
      .catch((e: unknown) => {
        throw e;
      });

    if (genresIds instanceof Array) {
      let data = genresIds.map((genreId: number | string) => {
        return {itemId: id, genreId: genreId};
      });
      return await ItemGenre.bulkCreate(data)
        .catch((e: unknown) => {
          throw e;
        });
    }

    return await ItemGenre.create({itemId: id, genreId: genresIds})
      .catch((e: unknown) => {
        throw e;
      });
  }
}

async create(req: Request, res: Response, next: NextFunction) {
  try {
    const { characteristics, developersIds, genresIds, mainImage, coverImage } = req.body;
    let { images } = req.body;
    if (images && !Array.isArray(images)) images = [images];

    let {name, description,
      releaseDate, publisherId: company_publisherId,
      discount, discountFrom, discountTo,
      discountSize,
      price, amount, hide } = ItemsController.parseData(req.body);

    let characteristicsParsed: Object | undefined = undefined;
    if (characteristics) {
      characteristicsParsed = JSON.parse(characteristics);
    }

    const item = await Item
      .create({name, description, price, releaseDate, amount,
        discount, discountFrom, discountTo, discountSize, mainImage,
        coverImage, images, characteristics: characteristicsParsed, company_publisherId,
        hide})
      .catch((e: unknown) => {
        // ToDo: delete files
        //
        // if (mainImage) super.deleteFile(ITEMS_DIR, mainImage);
        // if (coverImage) super.deleteFile(ITEMS_DIR, coverImage);
      });
  }
}

```

```

        // if (images) {
        //     images.forEach((name: string) => {
        //         super.deleteFile(ITEMS_DIR, name);
        //     });
        // }
        return next(super.exceptionHandle(e));
    });

    if (!item) {
        // ToDo: delete files
        //
        // if (mainImage) super.deleteFile(ITEMS_DIR, mainImage);
        // if (coverImage) super.deleteFile(ITEMS_DIR, coverImage);
        // if (images) {
        //     images.forEach((name: string) => {
        //         super.deleteFile(ITEMS_DIR, name);
        //     });
        // }
        return next(ApiError.badRequest('Не удалось створити товар'));
    }

    if (!developersIds && !genresIds) {
        return res.json(item);
    }

    let err: unknown;
    developersIds ? await ItemsController.addDevelopers(item.id, developersIds)
        .catch((e: unknown) => {
            err = e;
        }) : undefined;
    if (err) return next(super.exceptionHandle(err));

    genresIds ? await ItemsController.addGenres(item.id, genresIds)
        .catch((e: unknown) => {
            err = e;
        }) : undefined;
    if (err) return next(super.exceptionHandle(err));

    return res.json(item);
}

catch (e: unknown) {
    // ToDo: delete files
    //
    // if (mainImage) super.deleteFile(ITEMS_DIR, mainImage);
    // if (images) {
    //     images.forEach((name: string) => {
    //         super.deleteFile(ITEMS_DIR, name);
    //     });
    // }
    return next(super.exceptionHandle(e));
}

}

async getAll(req: Request, res: Response, next: NextFunction) {
    const { name, description,
        releaseDate, releaseDateFrom, releaseDateTo,
        price, priceFrom, priceTo,
        amount, amountFrom, amountTo,
        discount, discountFrom, discountTo,
        discountSize, discountSizeFrom, discountSizeTo,
        descending, limit, page, sortBy,
        includePublisher, publisherId,
        includeGenres, genresIds,
        includeDevelopers, developersIds,
        includeWishlisted, includeInCart,
        includeRated, includeHidden
    } = ItemsController.parseData(req.query);

    const result = await getItems({
        name, description,
        releaseDate, releaseDateFrom, releaseDateTo,

```

```

        price, priceFrom, priceTo,
        amount, amountFrom, amountTo,
        discount, discountFrom, discountTo,
        discountSize, discountSizeFrom, discountSizeTo,
        descending, limit, page, sortBy,
        includePublisher, publisherId,
        includeGenres, genresIds,
        includeDevelopers, developersIds,
        includeWishlisted, includeInCart,
        includeRated, includeHidden
    ))
    .catch((e: unknown) => {
        return next(super.exceptionHandle(e));
    });
    if (!result) return next(ApiError.notFound('Товари не найдено'));

    const { items, totalCount } = result;

    let where = {};
    if (includeHidden) where = {
        hide: {
            [Op.or]: [false, includeHidden]
        }
    }

    const companies = await Company.findAll({ where });

    const genres = await Genre.findAll({ where });

    if (!items) return next(ApiError.notFound('Товари не найдено'));
    return res.json({ items, totalCount, companies, genres });
}

async getOne(req: Request, res: Response, next: NextFunction) {
    const id = (new ItemsController()).parseNumber(req.params.id as string |
undefined);
    if (!id) return next(ApiError.badRequest('Неправильный id товара'));

    const { includePublisher,
        includeGenres,
        includeDevelopers,
        includeWishlisted,
        includeInCart,
        includeRated
    } = ItemsController.parseData(req.query);
    const includeHidden = super.parseBoolean(req.query.includeHidden as string |
undefined) || false;

    const item = await getItem({
        id, includePublisher, includeGenres,
        includeDevelopers, includeWishlisted, includeInCart,
        includeRated, includeHidden
    })
    .catch((e: unknown) => {
        return next(super.exceptionHandle(e));
    });
    if (!item) return next(ApiError.notFound('Товар не найдено'));
    return res.json(item);
}

async update(req: Request, res: Response, next: NextFunction) {
    try {
        const { id } = req.params;

        const { name, description, characteristics, developersIds, genresIds,
mainImage, coverImage } = req.body;
        const { releaseDate, price,
            discount, discountFrom, discountTo,
            discountSize, publisherId, amount,
            hide } = ItemsController.parseData(req.body);

```

```

let { images } = req.body;
if (images && !Array.isArray(images)) images = [images];

let characteristicsParsed: Object | null | undefined;
if (characteristics !== undefined) {
  characteristicsParsed = JSON.parse(characteristics);
}
else if (characteristics === null) {
  characteristicsParsed = null;
}
console.log({ characteristics, characteristicsParsed });

const item = await Item.findByPk(id)
  .catch((e: unknown) => {
    return next(super.exceptionHandle(e));
  });
if (!item) return next(ApiError.notFound('Товар не найдено'));

// let oldMainName: string | undefined = item.mainImage;
// let oldImages: string[] | undefined = item.images;
// let oldCoverImage: string | undefined = item.coverImage;

if (name) item.name = name;
if (description) item.description = description;
if (releaseDate) item.releaseDate = releaseDate;
if (price !== undefined && price >= 0) item.price = price;
if (amount !== undefined && amount >= 0) {
  item.amount = amount;
}
if (mainImage) item.mainImage = mainImage;
if (images) item.images = images;
if (coverImage) item.coverImage = coverImage;

if (discount !== undefined) {
  item.discount = discount;

  if (discountFrom) item.discountFrom = discount ? discountFrom : null;
  if (discountTo) item.discountTo = discount ? discountTo : null;
  if (discountSize !== undefined) item.discountSize = discount ?
discountSize : null;
}
if (characteristicsParsed !== undefined) item.characteristics =
characteristicsParsed;
if (hide !== undefined) item.hide = hide;
if (publisherId && await Company.findByPk(publisherId))
item.company_publisherId = publisherId;

let err: unknown;
const result = await item.save()
  .catch((e: unknown) => {
    // ToDo: delete files
    //
    // if(mainImage) super.deleteFile(ITEMS_DIR, mainImage);
    // if(coverImage) super.deleteFile(ITEMS_DIR, coverImage);
    // if(images) {
    //   images.forEach((name: string) => {
    //     super.deleteFile(ITEMS_DIR, name);
    //   });
    // }
    err = e;
  });
if (err) return next(super.exceptionHandle(err));

if (!result) {
  // ToDo: delete files
  //
  // if (mainImage) super.deleteFile(ITEMS_DIR, mainImage);
  // if (coverImage) super.deleteFile(ITEMS_DIR, coverImage);
  // if (images) {
  //   images.forEach((name: string) => {
  //     super.deleteFile(ITEMS_DIR, name);
  //   });
  // }

```

```

        //    });
        // }
        return next(ApiError.badRequest('Не вдалося оновити товар'));
    }

    if (!developersIds && !genresIds) {
        return res.json(result);
    }

    developersIds ? await ItemsController.setDevelopers(id, developersIds)
        .catch((e: unknown) => {
            err = e;
        }) : undefined;
    if (err) return next(super.exceptionHandle(err));

    genresIds ? await ItemsController.setGenres(id, genresIds)
        .catch((e: unknown) => {
            err = e;
        }) : undefined;
    if (err) return next(super.exceptionHandle(err));

    // ToDo: delete files
    //
    // if(oldMainName && mainImage) super.deleteFile(ITEMS_DIR, oldMainName);
    // if(oldCoverImage && coverImage) super.deleteFile(ITEMS_DIR, oldCoverImage);
    // if(oldImages && images) {
    //     oldImages.forEach((name: string) => {
    //         super.deleteFile(ITEMS_DIR, name);
    //     });
    // }

    return res.json(item);
}
catch (e: unknown) {
    return next(super.exceptionHandle(e));
}
}

async delete(req: Request, res: Response, next: NextFunction) {
    try {
        const {id} = req.params;

        const oldItem = await Item.findByPk(id);
        // let oldImageName: string | undefined = oldItem?.mainImage;
        // let oldImagesNames: string[] | undefined = oldItem?.images;
        // let oldCoverImage: string | undefined = oldItem?.coverImage;

        const result = await oldItem?.destroy()
            .catch((e: unknown) => {
                return next(super.exceptionHandle(e));
            });

        if (!result) return next(ApiError.badRequest('Не вдалося видалити товар'));

        // ToDo: delete files
        //
        // if(oldImageName) super.deleteFile(ITEMS_DIR, oldImageName);
        // if(oldCoverImage) super.deleteFile(ITEMS_DIR, oldCoverImage);
        // if(oldImagesNames) {
        //     oldImagesNames.forEach((name: string) => {
        //         super.deleteFile(ITEMS_DIR, name);
        //     });
        // }

        await ItemsController.removeDevelopers(id)
            .catch((e: unknown) => {
                return next(super.exceptionHandle(e));
            });

        await ItemsController.removeGenres(id)
            .catch((e: unknown) => {

```

```

        return next(super.exceptionHandle(e));
    });

    return res.json({ ok: true });
}
catch (e: unknown) {
    return next(super.exceptionHandle(e));
}
}

async addGenres(req: Request, res: Response, next: NextFunction) {
    const { id } = req.params;
    const { genresIds } = req.body;

    let err: unknown;
    const result = await ItemsController.addGenres(id, genresIds)
        .catch((e: unknown) => {
            err = super.exceptionHandle(e);
        });
    if (err) return next(err);

    return res.json({message: "Жанр(-и) успішно додано товару", result});
}

async removeGenres(req: Request, res: Response, next: NextFunction) {
    const { id } = req.params;
    const { genresIds } = req.body;

    let err: unknown;
    const result = await ItemsController.removeGenres(id, genresIds)
        .catch((e: unknown) => {
            err = super.exceptionHandle(e);
        });
    if (err) return next(err);

    return res.json({message: "Жанр успішно видалено з товару", result});
}

async addDevelopers(req: Request, res: Response, next: NextFunction) {
    const { id } = req.params;
    const { developersIds } = req.body;

    let err: unknown;
    const result = await ItemsController.addDevelopers(id, developersIds)
        .catch((e: unknown) => {
            err = super.exceptionHandle(e);
        });
    if (err) return next(err);

    return res.json({message: "Розробника(-ів) успішно додано товару", result});
}

async removeDevelopers(req: Request, res: Response, next: NextFunction) {
    const { id } = req.params;
    const { developersIds } = req.body;

    let err: unknown;
    const result = await ItemsController.removeDevelopers(id, developersIds)
        .catch((e: unknown) => {
            err = super.exceptionHandle(e);
        });
    if (err) return next(err);

    return res.json({message: "Розробника(-ів) успішно видалено з товару", result});
}

async addToWishList(req: Request, res: Response, next: NextFunction) {
    try {
        const {id} = req.query;
        const request_user = req.user;
    }

```

```

const item = await Item.findByPk(id);
if (!item) return next(ApiError.notFound('Товар не знайдено'));

const user = await User.findByPk(request_user.Id);
if (!user) return next(ApiError.notFound('Користувача не знайдено'));

const wishList = await Wishlist.findAll({where: {userId: user.id, itemId:
id}});

if (wishList.length > 0)
    return next(ApiError.badRequest('Товар вже є в списку бажань'));

const wishListItem = await Wishlist.create({userId: user.id, itemId: id});

return res.json(wishListItem);
}
catch (e: unknown) {
    return next(super.exceptionHandle(e));
}
}

async removeFromWishList(req: Request, res: Response, next: NextFunction) {
    try {
        const {id} = req.query;
        const request_user = req.user;

        const item = await Item.findByPk(id);
        if (!item) return next(ApiError.notFound('Товар не знайдено'));

        const user = await User.findByPk(request_user.Id);
        if (!user) return next(ApiError.notFound('Користувача не знайдено'));

        const wishList = await Wishlist.findOne({where: {userId: user.id, itemId:
id}});

        if (!wishList)
            return next(ApiError.badRequest('Товару немає в списку бажань'));

        await Wishlist.destroy({where: {userId: user.id, itemId: id}});
        return res.json({ ok: true });
    }
    catch (e: unknown) {
        return next(super.exceptionHandle(e));
    }
}

async toggleWishList(req: Request, res: Response, next: NextFunction) {
    try {
        const {id} = req.params;
        const request_user = req.user;

        const item = await Item.findByPk(id);
        if (!item) return next(ApiError.notFound('Товар не знайдено'));
        if (item.hide) return next(ApiError.badRequest('Товар не доступний'));

        const user = await User.findByPk(request_user.id);
        if (!user) return next(ApiError.notFound('Користувача не знайдено'));

        const wishList = await Wishlist.findOne({where: {userId: user.id, itemId:
id}});

        if (wishList) {
            await Wishlist.destroy({where: {userId: user.id, itemId: id}});
            return res.json({message: 'Товар успішно видалено зі списку бажань', ok:
true});
        }
        await Wishlist.create({userId: user.id, itemId: id});
        return res.json({message: 'Товар успішно додано до списку бажань', ok: true});
    }
    catch (e: unknown) {
        return next(super.exceptionHandle(e));
    }
}

```

```

    }

    async toggleCart(req: Request, res: Response, next: NextFunction) {
      try {
        const {id} = req.params;
        const request_user = req.user;

        const item = await Item.findByPk(id);
        if (!item) return next(ApiError.notFound('Товар не знайдено'));
        if (item.hide) return next(ApiError.badRequest('Товар не доступний для покупки'));

        const user = await User.findByPk(request_user.id);
        if (!user) return next(ApiError.notFound('Користувача не знайдено'));

        const itemCart = await ItemCart.findOne({where: {userId: user.id, itemId: id}});
        if (itemCart) {
          await ItemCart.destroy({where: {userId: user.id, itemId: id}});
          return res.json({message: 'Товар успішно видалено з кошика', ok: true});
        }

        if (item.amount <= 0) return next(ApiError.badRequest('Товар скінчився в наявності'));
        await ItemCart.create({userId: user.id, itemId: id});
        return res.json({message: 'Товар успішно додано до кошика', ok: true});
      } catch (e: unknown) {
        return next(super.exceptionHandle(e));
      }
    }

    async getAllRates(req: Request, res: Response, next: NextFunction) {
      try {
        const {
          sortBy
        } = req.query;
        let descending = super.parseBoolean(req.query.desc as string);
        if (descending === undefined) descending = super.parseBoolean(req.query.descending as string) || false;
        const limit = super.parseNumber(req.query.limit as string) || 12;
        const page = super.parseNumber(req.query.page as string) || 1;

        const result = await getAllRates({sortBy: sortBy as string, descending, limit, page});
        if (!result) return next(ApiError.notFound('Коментарів не знайдено'));

        const { totalCount, rates } = result;

        if(!rates) return next(ApiError.notFound('Коментарів не знайдено'));
        return res.json({totalCount, rates});
      } catch (e: unknown) {
        return next(super.exceptionHandle(e));
      }
    }

    async setRate(req: Request, res: Response, next: NextFunction) {
      try {
        const { id } = req.params;
        const { content } = req.body;
        const rate = super.parseNumber(req.body.rate as string);

        if (!req.user) return next(ApiError.unauthorized('Необхідна авторизація'));
        const user = await User.findByPk(req.user.id);
        if (!user) return next(ApiError.notFound('Користувача не знайдено'));

        const item = await Item.findByPk(id);
        if (!item) return next(ApiError.notFound('Товар не знайдено'));
        if (item.hide) return next(ApiError.badRequest('Товар не доступний'));

```

```

        const itemRate = await ItemRate.findOne({where: {userId: req.user.id, itemId:
id}}});

        if (!itemRate) {
            if (!rate) return next(ApiError.badRequest('Необхідно вказати оцінку'));
            const newRate = await ItemRate.create({userId: req.user.id, itemId: id,
content, rate});
            return res.json({message: 'Відгук успішно додано', rate: newRate, user });
        }

        if (content) itemRate.content = content;
        if (rate) itemRate.rate = rate;
        await itemRate.save();

        return res.json({message: 'Відгук успішно оновлено', rate: itemRate, user });
    }
    catch (e) {
        return next(super.exceptionHandle(e));
    }
}

async removeRate(req: Request, res: Response, next: NextFunction) {
    try {
        const { id } = req.params;

        if (!req.user) return next(ApiError.unauthorized('Необхідна авторизація'));
        const user = await User.findByPk(req.user.id);
        if (!user) return next(ApiError.notFound('Користувача не знайдено'));

        const itemRate = await ItemRate.findOne({where: {userId: req.user.id, itemId:
id}}});
        if (!itemRate) return next(ApiError.notFound('Відгук не знайдено'));

        await ItemRate.destroy({where: {userId: req.user.id, itemId: id}});
        return res.json({message: 'Відгук успішно видалено', ok: true });
    }
    catch (e) {
        return next(super.exceptionHandle(e));
    }
}

async toggleViolation(req: Request, res: Response, next: NextFunction) {
    try {
        const { id } = req.params;
        const { violation, violation_reason } = req.body;

        let violationParsed = super.parseBoolean(violation as string | undefined) ||
false;

        const rate = await ItemRate.findByPk(id);
        if (!rate) return next(ApiError.notFound('Товар не знайдено'));

        if (!violationParsed) {
            rate.violation_reason = null;
            rate.hide = false;
            rate.violation = false;
        }
        else {
            rate.violation_reason = violation_reason;
            rate.hide = true;
            rate.violation = true;
        }

        const result = await rate.save();

        return res.json({message: 'Статус порушення успішно оновлено', result });
    }
    catch (e) {
        return next(super.exceptionHandle(e));
    }
}
}

```

```

    async test(req: Request, res: Response) {
        return res.json({message: `Items route works!`, request: {body: req.body, query: req.query}})
    }
}

export default new ItemsController();

```

Лістинг коду файлу NewsRouter.ts:

```

import {Router} from 'express';
import NewsController from "../controllers/NewsController";
import auth from "../middleware/AuthMiddleware";

const newsRouter = Router()

newsRouter.post('/', auth, NewsController.create);
newsRouter.patch('/:id', auth, NewsController.update);
newsRouter.delete('/:id', auth, NewsController.delete);
newsRouter.get('/', NewsController.getAll);
newsRouter.get('/:id', NewsController.getOne);
newsRouter.get('/test', NewsController.test);

newsRouter.post('/:id/comment', auth, NewsController.setComment);
newsRouter.delete('/:id/comment', auth, NewsController.removeComment);
newsRouter.get('/:id/comments', NewsController.getComments);

module.exports = newsRouter

```

Лістинг коду файлу NewsController.ts:

```

import type {NextFunction, Request, Response} from 'express';
import Controller from './Controller';
import Publication, { getPublication, getPublications } from "../models/Publication";
import ApiError from "../errors/ApiError";
import PublicationComment, { getAllComments } from "../models/PublicationComment";
import User from "../models/User";

class NewsController extends Controller {
    private static async _checkPublication(id?: number | string) {
        if (!id) throw ApiError.badRequest('Неправильний id публікації');
        const news = await Publication.findPk(id);
        if (!news) throw ApiError.badRequest('Публікацію не знайдено');
    }

    async create(req: Request, res: Response, next: NextFunction) {
        try {
            const { title, content, hide } = req.body;
            let { tags } = req.body;

            const hideParsed = super.parseBoolean(hide);
            const userId = req.user.id;
            if (tags && !Array.isArray(tags)) tags = [tags];

            const publication = await Publication
                .create({ title, content, hide: hideParsed, userId, tags })
                .catch((e: unknown) => {
                    return next(super.exceptionHandle(e));
                });

            if (!publication) {
                return next(ApiError.badRequest('Публікацію не створено'));
            }

            if (!tags) return res.json(publication);

            return res.json(publication);
        }
    }
}

```

```

        catch (e: unknown) {
            return next(super.exceptionHandle(e));
        }
    }

    async getAll(req: Request, res: Response, next: NextFunction) {
        try {
            const { title, content,
                createdAt, createdFrom, createdTo,
                desc, descending, limit,
                page, sortBy, hidden
            } = req.query;

            let { tags } = req.query;
            if (tags && typeof tags === 'string') tags = [tags];

            let { authorsIds } = req.query;
            if (authorsIds && typeof authorsIds === 'string') authorsIds = [authorsIds];

            const createdAtParsed = super.parseDate(createdAt as string | undefined);
            const createdFromParsed = super.parseDate(createdFrom as string | undefined);
            const createdToParsed = super.parseDate(createdTo as string | undefined);
            const hideParsed = super.parseBoolean(hidden as string | undefined) || false;

            let {descending: descendingParsed, limit: limitParsed, page: pageParsed} =
                super.parsePagination(desc as string | undefined, descending as string |
undefined,
                    limit as string | undefined, page as string | undefined);

            const result = await getPublications({
                title: title as string | undefined, content: content as string | undefined,
                createdAt: createdAtParsed, createdFrom: createdFromParsed, createdTo:
createdToParsed,
                descending: descendingParsed, limit: limitParsed, page: pageParsed, sortBy:
sortBy as string | undefined,
                includeHidden: hideParsed, tags: tags as string[] | undefined, authorsIds:
authorsIds as number[] | undefined
            })
                .catch((e: unknown) => {
                    return next(super.exceptionHandle(e));
                });
            if (!result) return next(ApiError.badRequest('Публікацій не знайдено'));

            const { publications, totalCount } = result;

            if (!publications) return next(ApiError.badRequest('Публікацій не знайдено'));

            return res.json({ publications, totalCount });
        }
        catch (e: unknown) {
            return next(super.exceptionHandle(e));
        }
    }

    async getOne(req: Request, res: Response, next: NextFunction) {
        try {
            const id = super.parseNumber(req.params.id);
            if (!id) return next(ApiError.badRequest('Неправильний id публікації'));
            const includeComments = super.parseBoolean(req.query.includeComments as
string | undefined) || true;
            const includeViolations = super.parseBoolean(req.query.includeViolations as
string | undefined) || false;
            const includeHidden = super.parseBoolean(req.query.includeHidden as string |
undefined) || false;

            const publication = await getPublication({
                id,
                includeComments,
                includeViolations,
                includeHidden
            })
        }
    }

```

```

        .catch((e: unknown) => {
            return next(super.exceptionHandle(e));
        });

        if (!publication) return next(ApiError.badRequest('Публікацію не знайдено'));

        return res.json(publication);
    }
    catch (e: unknown) {
        console.log(e);
        return next(super.exceptionHandle(e));
    }
}

async update(req: Request, res: Response, next: NextFunction) {
    try {
        const { id } = req.params;
        const { title, content, hide } = req.body;
        let { tags } = req.body;
        if (tags && !Array.isArray(tags)) tags = [tags];

        let hideParsed = super.parseBoolean(hide as string | undefined);

        let publication = await Publication.findPk(id);
        if (!publication) return next(ApiError.badRequest('Публікацію не знайдено'));

        if (title) publication.title = title;
        if (content) publication.content = content;
        if (publication.violation) hideParsed = true;
        if (hide) publication.hide = hideParsed;
        if (tags) publication.tags = tags;

        let err: unknown;
        const result = await publication.save()
            .catch((e: unknown) => {
                err = e;
            });
        if (err) return next(super.exceptionHandle(err));

        if (!result) return next(ApiError.badRequest('Публікацію не оновлено'));

        return res.json(publication);
    }
    catch (e: unknown) {
        return next(super.exceptionHandle(e));
    }
}

async toggleViolation (req: Request, res: Response, next: NextFunction) {
    try {
        const { id } = req.params;
        const { violation, violation_reason } = req.body;

        let violationParsed = super.parseBoolean(violation as string | undefined) ||
false;

        const publication = await Publication.findPk(id);
        if (!publication) return next(ApiError.badRequest('Публікацію не знайдено'));

        if (!violationParsed) {
            publication.violation = false;
            publication.violation_reason = null;
        }
        else {
            publication.violation = true;
            publication.hide = true;
            publication.violation_reason = violation_reason;
        }

        const result = await publication.save()
            .catch((e: unknown) => {

```

```

        return next(super.exceptionHandle(e));
    });

    return res.json({ message: 'Статус порушення оновлено', result });
}
catch (e: unknown) {
    console.log(e);
    return next(super.exceptionHandle(e));
}
}

async toggleCommentViolation(req: Request, res: Response, next: NextFunction) {
    try {
        const { id } = req.params;
        const { violation, violation_reason } = req.body;

        let violationParsed = super.parseBoolean(violation as string | undefined);

        const comment = await PublicationComment.findPk(id);
        if (!comment) return next(ApiError.badRequest('Коментар не знайдено'));

        if (!violationParsed) {
            comment.violation = false;
            comment.hide = false;
            comment.violation_reason = null;
        }
        else {
            comment.violation = violationParsed;
            comment.hide = true;
            comment.violation_reason = violation_reason;
        }

        const result = await comment.save();

        return res.json({ message: "Статус порушення оновлено", result });
    }
    catch (e: unknown) {
        console.log(e);
        return next(super.exceptionHandle(e));
    }
}

async delete(req: Request, res: Response, next: NextFunction) {
    try {
        const { id } = req.params;

        const publication = await Publication.findPk(id);
        if (!publication) return next(ApiError.badRequest('Публікацію не знайдено'));

        const author = await User.findPk(publication.userId);
        if (req.user.id !== author.id && author.role.toLowerCase() === "user") {
            return next(ApiError.badRequest("Ви не можете видалити чужу публікацію"));
        }

        await Publication
            .destroy({ where: { id } })
            .catch((e: unknown) => {
                return next(super.exceptionHandle(e));
            });

        return res.json({ ok: true });
    }
    catch (e: unknown) {
        return next(super.exceptionHandle(e));
    }
}

async getComments(req: Request, res: Response, next: NextFunction) {
    try {
        const { id } = req.params;

```

```

    await NewsController._checkPublication(id)
    .catch((e: unknown) => {
        return next(super.exceptionHandle(e));
    });

    const comments = await PublicationComment.findAll({
        where: { publicationId: id },
        attributes: ['content', 'rate', 'createdAt', 'updatedAt'],
        include: [
            {
                model: User,
                as: 'User',
                attributes: ['id', 'firstName', 'lastName', 'login', 'image'],
            },
        ],
    });

    return res.json({ message: "Коментарі успішно отримано", comments });
}
catch (e: unknown) {
    return next(super.exceptionHandle(e));
}
}

async getAllComments(req: Request, res: Response, next: NextFunction) {
    try {
        const {
            sortBy
        } = req.query;
        let descending = super.parseBoolean(req.query.desc as string);
        if (descending === undefined) descending =
super.parseBoolean(req.query.descending as string) || false;
        const limit = super.parseNumber(req.query.limit as string) || 12;
        const page = super.parseNumber(req.query.page as string) || 1;

        const result = getAllComments({sortBy: sortBy as string, descending, limit,
page});
        if (!result) return next(ApiError.badRequest('Невірні параметри запиту'));

        const { totalCount, comments } = await result;

        if (!comments) return next(ApiError.badRequest('Коментарі не знайдено'));
        return res.json({ totalCount, comments });
    }
    catch (e: unknown) {
        return next(super.exceptionHandle(e));
    }
}

async setComment(req: Request, res: Response, next: NextFunction) {
    try {
        const { id } = req.params;
        const { content } = req.body;
        const rate = super.parseNumber(req.body.rate as string);

        if (!req.user) return next(ApiError.unauthorized('Необхідна авторизація'));
        const user = await User.findByPk(req.user.id);
        if (!user) return next(ApiError.notFound('Користувача не знайдено'));

        const publication = await Publication.findByPk(id);
        if (!publication) return next(ApiError.notFound('Публікацію не знайдено'));
        if (publication.hide) return next(ApiError.forbidden('Немає доступу'));
        if (publication.userId === req.user.id) return next(ApiError.forbidden('Не
можна оцінювати власну публікацію'));

        const publicationComment = await PublicationComment.findOne({ where:
{ publicationId: id, userId: req.user.id } });

        if (!publicationComment) {
            if (!rate) return next(ApiError.badRequest('Невірна оцінка'));
            const newComment = await PublicationComment.create({ content, rate,

```

```

publicationId: id, userId: req.user.id });
    return res.json({ message: "Коментар успішно додано", comment: newComment,
user });
  }

  if (content) publicationComment.content = content;
  if (rate) publicationComment.rate = rate;

  await publicationComment.save();
  return res.json({ message: "Коментар успішно оновлено", comment:
publicationComment, user });
}
catch (e) {
  return next(super.exceptionHandle(e));
}
}

async removeComment(req: Request, res: Response, next: NextFunction) {
  try {
    const { id } = req.params;

    if (!req.user) return next(ApiError.unauthorized('Необхідна авторизація'));
    const user = await User.findByPk(req.user.id);
    if (!user) return next(ApiError.notFound('Користувача не знайдено'));

    const publicationComment = await PublicationComment.findOne({ where:
{ publicationId: id, userId: req.user.id } });
    if (!publicationComment) return next(ApiError.badRequest('Коментар не
знайдено'));

    await publicationComment.destroy();
    return res.json({ message: "Коментар успішно видалено", ok: true });
  }
  catch (e) {
    return next(super.exceptionHandle(e));
  }
}

async test(req: Request, res: Response) {
  return res.json({message: `News route works!`, request: {body: req.body, query:
req.query}})
}
}

export default new NewsController();

```

Лістинг коду файлу UserRoutes.ts:

```

import {Router} from 'express';
import UserController from "../controllers/UserController";
import { checkAdmin, checkWorker } from "../middleware/CheckRoleMiddleware";
import auth from "../middleware/AuthMiddleware";

const userRouter = Router()

userRouter.get('/test', UserController.test);
userRouter.post('/register', UserController.register);
userRouter.post('/createModerator', checkAdmin, UserController.createModerator);
userRouter.post('/toggleModerator/:id', checkAdmin, UserController.toggleModerator);
userRouter.patch('/setRole/:id', checkAdmin, UserController.setRole);

userRouter.post('/login', UserController.login);
userRouter.patch('/update', auth, UserController.update);
userRouter.get('/profile', auth, UserController.profile);
userRouter.get('/auth', auth, UserController.check);

userRouter.get('/all', checkWorker, UserController.allUsers);

userRouter.delete('/cart/', auth, UserController.clearCart);
userRouter.post('/cart/setup', auth, UserController.setUpCart);
userRouter.post('/cart/success/:transactionId', UserController.buyCart);

```

```
module.exports = userRouter
```

Лістинг коду файлу UserController.ts:

```
import type { NextFunction, Request, Response } from 'express';
import ApiError from "../errors/ApiError";
import Controller from "../Controller";
import bcrypt from "bcrypt";
import jwt from "jsonwebtoken";
import User, { getUser, getUsers } from "../models/User";
import { Op } from "sequelize";
import ItemCart from "../models/ItemCart";
import { Item, ItemBought } from "../models";
import { v4 as uuidv4 } from 'uuid';

const USER = 'USER'
const MODERATOR = 'MODERATOR';
const ADMIN = 'ADMIN';

const SALT_ROUNDS = 5;

class UserController extends Controller {
  static generateJwt(id: number, login: string, email: string, role: string, image: string) {
    return jwt.sign({
      id, login, email, role, image,
    }, process.env.SECRET_KEY as string, { expiresIn: '24h' });
  }

  private static async _createUser(email: string, login: string, firstName: string, lastName: string, password: string, imageName: string | undefined, role: string) {
    const hashPasswordValid = await UserController._validateData(email, login, firstName, lastName, password)
      .catch((e: ApiError) => {
        throw e;
      });

    const user = await User.create({
      email, login, firstName, lastName, password: hashPasswordValid, image: imageName, role
    });

    return UserController.generateJwt(user.id, user.login, user.email, user.role, user.image);
  }

  private static async _validateData(email: string, login: string, firstName: string, lastName: string, password: string): Promise<string> {
    if (!email || !login || !firstName || !lastName || !password) {
      throw ApiError.badRequest('Не всі поля заповнені');
    }

    await UserController._checkCandidate(email, login).catch((e: ApiError) => {
      throw e
    });

    await UserController._checkPassword(password).catch((e: ApiError) => {
      throw e
    });

    return await bcrypt.hash(password, SALT_ROUNDS);
  }

  private static async _checkCandidate(email = '', login = '') {
    const candidate = await User.findOne({ where: {
      [Op.or]: [
        { email },
        { login }
      ]
    }});
  }
}
```

```

    }}
  });
  if (candidate) {
    throw ApiError.badRequest('Користувач з таким email або логіном вже існує');
  }
}

private static async _checkPassword(password: string) {
  if (password.length < 6) {
    throw ApiError.badRequest('Пароль має бути не менше 6 символів');
  }

  let passwordRegex = new RegExp(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).+$/);
  if (!passwordRegex.test(password)) {
    throw ApiError.badRequest('Пароль повинен містити хоча б одну велику літеру,
' +
    'одну малу літеру та одну цифру');
  }
}

async register(req: Request, res: Response, next: NextFunction) {
  const { image } = req.body;
  try {
    const { email, login, firstName, lastName, password } = req.body;

    let role = req.body.role?.toUpperCase() || USER;

    const admin = await User.findOne({where: {role: ADMIN}});
    if (!admin) role = ADMIN;
    else if ([ADMIN, MODERATOR].includes(role)) {
      // TODO: delete image
      //
      // if (image) super.deleteFile(USERS_DIR, image);
      return next(ApiError.badRequest('Тільки адміністратор може створювати ' +
        'адміністраторів та модераторів'));
    }

    let error: unknown;
    let token = await UserController._createUser(email, login, firstName,
lastName,
    password, image, role)
      .catch((e: ApiError | Error | unknown) => {
        // TODO: delete image
        //
        // if (image) super.deleteFile(USERS_DIR, image);
        error = e;
      });
    if (error) return next(super.exceptionHandle(error));

    return res.json({message: "Реєстрацію пройдено успішно", token});
  }
  catch (e: unknown) {
    // TODO: delete image
    // if (image) super.deleteFile(USERS_DIR, image);
    return next(super.exceptionHandle(e));
  }
}

async createModerator(req: Request, res: Response, next: NextFunction) {
  const { image } = req.body;
  try {
    const { email, login, firstName, lastName, password } = req.body;

    const token = await UserController._createUser(email, login, firstName,
lastName,
    password, image, MODERATOR)
      .catch((e: ApiError | Error | unknown) => {
        // TODO: delete image
        // if (image) super.deleteFile(USERS_DIR, image);
        if (e instanceof ApiError) return next(e);
        return next(super.exceptionHandle(e));
      });
  }
}

```

```

    });

    return res.json({message: "Успішно створено модератора", token});
  }
  catch (e: unknown) {
    // ToDo: delete image
    // if (image) super.deleteFile(USERS_DIR, image);
    return next(super.exceptionHandle(e));
  }
}

async login(req: Request, res: Response, next: NextFunction) {
  const { credentials, password } = req.body;

  if (!credentials || !password) {
    return next(ApiError.badRequest('Некоректні дані'));
  }

  const user = await User.findOne({ where: {
    [Op.or]: [
      { email: credentials },
      { login: credentials }
    ]
  }});
  if (!user) {
    return next(ApiError.badRequest('Користувача з таким email або логіном не існує'));
  }

  let comparePassword = bcrypt.compareSync(password, user.password);
  if (!comparePassword) {
    return next(ApiError.badRequest('Невірний пароль'));
  }

  const token = UserController.generateJwt(user.id, user.login, user.email, user.role, user.image);
  return res.json({message: "Авторизацію пройдено успішно", token});
}

async check(req: Request, res: Response) {
  const token = UserController.generateJwt(req.user.id, req.user.login, req.user.email, req.user.role, req.user.image);
  return res.json({token});
}

async update(req: Request, res: Response, next: NextFunction) {
  const { image } = req.body;
  let error: unknown;
  try {
    const user = await User.findPk(req.user.id);
    if (!user) {
      // ToDo: delete image
      // if (image) super.deleteFile(USERS_DIR, image);
      return next(ApiError.internal('Користувача не знайдено'));
    }

    const { email, login, firstName, lastName, password, newPassword } = req.body;

    if (!password) {
      // ToDo: delete image
      // if (image) super.deleteFile(USERS_DIR, image);
      return next(ApiError.badRequest('Введіть пароль'));
    }

    await UserController._checkCandidate(email, login).catch((e: unknown) => {
      // ToDo: delete image
      // if (image) super.deleteFile(USERS_DIR, image);
      error = e;
    });
    if (error) return next(super.exceptionHandle(error));
  }
}

```

```

    if (newPassword) {
      await UserController._checkPassword(newPassword).catch((e: unknown) => {
        // ToDo: delete image
        // if (image) super.deleteFile(USERS_DIR, image);
        error = e;
      });
      if (error) return next(super.exceptionHandle(error));
    }

    let comparePassword = bcrypt.compareSync(password, user.password);
    if (!comparePassword) {
      // ToDo: delete image
      // if (image) super.deleteFile(USERS_DIR, image);
      return next(ApiError.badRequest('Невірний пароль'));
    }

    let oldImageName: string | undefined = user.image;
    if (oldImageName === 'default.png') oldImageName = undefined;

    if (email) user.email = email;
    if (login) user.login = login;
    if (firstName) user.firstName = firstName;
    if (lastName) user.lastName = lastName;
    if (newPassword) user.password = bcrypt.hashSync(newPassword, SALT_ROUNDS);
    if (image) user.image = image;

    await user.save().catch((e: unknown) => {
      // ToDo: delete image
      // if (image) super.deleteFile(USERS_DIR, image);
      error = e;
    });
    if (error) return next(super.exceptionHandle(error));

    // ToDo: delete image
    // if (image && oldImageName) super.deleteFile(USERS_DIR, oldImageName);

    const token = UserController.generateJwt(user.id, user.login, user.email,
user.role, user.image);

    return res.json({message: "Користувача відредаговано успішно", token});
  }
  catch (e: unknown) {
    // ToDo: delete image
    // if (image) super.deleteFile(USERS_DIR, image);
    return next(super.exceptionHandle(e));
  }
}

async profile(req: Request, res: Response, next: NextFunction) {
  try {
    let { includeBought, includeCart,
      includeRates,
      includeWishlist, includePublications,
      includePublicationComments } = req.query;

    let includeBoughtItemsParsed = super.parseBoolean(includeBought as string) ||
false;
    let includeCartParsed = super.parseBoolean(includeCart as string) || false;
    let includeItemsRatesParsed = super.parseBoolean(includeRates as string) ||
false;
    let includeWishlistParsed = super.parseBoolean(includeWishlist as string) ||
false;
    let includePublicationsParsed = super.parseBoolean(includePublications as
string) || false;
    let
      includePublicationCommentsParsed
      =
super.parseBoolean(includePublicationComments as string) || false;

    const { user, boughtItems } = await getUser({
      id: req.user.id,
      includeBoughtItems: includeBoughtItemsParsed,
      includeCart: includeCartParsed,

```

```

        includeItemsRates: includeItemsRatesParsed,
        includeWishlist: includeWishlistParsed,
        includePublications: includePublicationsParsed,
        includePublicationComments: includePublicationCommentsParsed
    });
    if (!user) {
        return next(ApiError.internal('Користувача не знайдено'));
    }

    const data = boughtItems ? boughtItems.map((item: any) => item.dataValues) :
[];

    return res.json({ ...user.dataValues, BoughtItems: data });
}
catch (e: unknown) {
    return next(super.exceptionHandle(e));
}
}

async setUpCart(req: Request, res: Response, next: NextFunction) {
    try {
        const reqUser = req.user;
        const transactionId = uuidv4();

        const cart = await ItemCart.findAll({ where: { userId: reqUser.id }});
        if (!cart || cart.length === 0) return next(ApiError.badRequest('Кошик
порожній'));

        const setupResult = ItemCart.update({ transactionId }, { where: { userId:
reqUser.id }});
        if (!setupResult) return next(ApiError.internal('Помилка при встановленні
транзакції'));

        return res.json({message: "Транзакцію встановлено успішно", transactionId });

        // const itemsIds = cart.map((item: typeof ItemCart) => item.itemId);
        // const items = await Item.findAll({ where: { id: itemsIds }});
        // const type = "pay";
        // const description = "Оплата товарів";
        // const amount = items.reduce((sum: number, item: typeof Item) => sum +
item.price, 0);
        // const currency = "UAH";
        //
        // const client_url = "http://localhost:3000/cart/success";
        // const server_url = (process.env.NGROK_SERVER_URL ||
'http://localhost:5000') + `/api/user/cart/success/${transactionId}`;
    }
    catch (e: unknown) {
        return next(super.exceptionHandle(e));
    }
}

async buyCart(req: Request, res: Response, next: NextFunction) {
    try {
        const { transactionId } = req.params;

        const alreadyBought = await ItemBought.findOne({ where: { transactionId }});
        if (alreadyBought) return next(ApiError.badRequest('Товари вже придбані'));

        const cart = await ItemCart.findAll({ where: { transactionId }});
        if (!cart || cart.length === 0) return next(ApiError.badRequest('Кошик
порожній'));

        const userId = cart[0].userId;
        const ids = cart.map((item: typeof ItemCart) => item.itemId);

        const data = ids.map((id: number) => {
            return {
                itemId: id,
                userId,
                transactionId
            }
        })
    }
    catch (e: unknown) {
        return next(super.exceptionHandle(e));
    }
}

```

```

    });
  });

  const bought = await ItemBought.bulkCreate(data);
  await ItemCart.destroy({where: { transactionId }});
  await Item.decrement('amount', { where: { id: ids }});

  return res.json({message: 'Товари успішно придбані', bought, ok: true});
}
catch (e: unknown) {
  return next(super.exceptionHandle(e));
}
}

async clearCart(req: Request, res: Response, next: NextFunction) {
  try {
    const reqUser = req.user;
    await ItemCart.destroy({where: { userId: reqUser.id }});

    return res.json({message: 'Кошик очищено', ok: true});
  }
  catch (e: unknown) {
    return next(super.exceptionHandle(e));
  }
}

async allUsers(req: Request, res: Response, next: NextFunction) {
  const {
    login,
    email,
    firstName,
    lastName,
    role,
    sortBy,
    descending, desc,
  } = req.query;
  const descendingParsed = super.parseBoolean(descending as string) ||
super.parseBoolean(desc as string) || false;
  const page = Math.max(super.parseNumber(req.query.page as string) || 1, 1);
  const limit = Math.max(super.parseNumber(req.query.limit as string) || 10, 1);

  const isBlocked = super.parseBoolean(req.query.isBlocked as string);

  const result = await getUsers({
    login: login as string,
    email: email as string,
    firstName: firstName as string,
    lastName: lastName as string,
    role: role as string,
    isBlocked,
    sortBy: sortBy as string,
    descending: descendingParsed,
    page,
    limit
  })
  .catch((e: unknown) => {
    return next(super.exceptionHandle(e));
  });
  if (!result) return next(ApiError.internal('Помилка отримання користувачів'));

  const { users, totalCount } = result;

  if (!users) return next(ApiError.internal('Користувачів не знайдено'));

  return res.json({users, totalCount});
}

async toggleModerator(req: Request, res: Response, next: NextFunction) {
  try {
    const { id } = req.params;

```

```

        const user = await User.findOne({ where: { id }, attributes: { exclude:
['password'] }});
        if (!user) return next(ApiError.badRequest('Користувача не знайдено'));

        if (user.role.toUpperCase() === ADMIN) return
next(ApiError.badRequest('Неможливо змінити роль адміністратора'));

        if (user.role.toUpperCase() === MODERATOR) {
            user.role = USER;
        }
        else {
            user.role = MODERATOR;
        }

        const result = await user.save();
        if (!result) return next(ApiError.internal('Помилка при зміні ролі'));

        return res.json({message: 'Роль успішно змінено', result });
    }
    catch (e: unknown) {
        return next(super.exceptionHandle(e));
    }
}

async setRole(req: Request, res: Response, next: NextFunction) {
    try {
        const { id } = req.params;
        let { role } = req.body;

        const user = await User.findOne({ where: { id }, attributes: { exclude:
['password'] }});
        if (!user) return next(ApiError.badRequest('Користувача не знайдено'));

        if (user.role.toUpperCase() === ADMIN) return
next(ApiError.badRequest('Неможливо змінити роль адміністратора'));

        if (![USER, MODERATOR].includes(role.toUpperCase())) role = USER;
        user.role = role;

        const result = await user.save();
        if (!result) return next(ApiError.internal('Помилка при зміні ролі'));

        return res.json({message: 'Роль успішно змінено', result });
    }
    catch (e: unknown) {
        return next(super.exceptionHandle(e));
    }
}

async test(req: Request, res: Response) {
    res.json({message: `User route works!!!!`, request: {body: req.body, query:
req.query}})
}

export default new UserController();

```

Лістинг файлу налаштування маршрутизації клієнтського додатку з папки

routes

Лістинг файлу index.tsx:

```

import { createBrowserRouter, Params, RouterProvider } from "react-router-dom";
import RootPage from "../RootPage";
import Items, { SortSwitch as ItemsSortSwitch } from "../Items";
import Item from "../Item";
import Genres from "../Genres";
import Genre from "../Genre";
import Companies, { SortSwitch as CompaniesSortSwitch } from "../Companies";
import Company from "../Company";
import NewsList, { SortSwitch as PublicationsSortSwitch } from "../NewsList";
import NewsItem from "../NewsItem";
import Login from "../Login";
import Register from "../Register";
import AdminBase from "../Admin/AdminBase";
import AdminHome from "../Admin/AdminHome";
import AdminItems from "../Admin/AdminItems";
import AdminItem from "../Admin/AdminItem";
import AdminItemForm from "../Admin/AdminItemForm";
import NewsForm from "../NewsForm";
import AdminNews from "../Admin/AdminNews";
import AdminNewsItem from "../Admin/AdminNewsItem";
import AdminGenres, { SortSwitch as GenresSortSwitch } from "../Admin/AdminGenres";
import AdminGenre from "../Admin/AdminGenre";
import Cart from "../Cart";
import ClientError from "../ClientError";
import AdminGenreForm from "../Admin/AdminGenreForm";
import AdminCompanies from "../Admin/AdminCompanies";
import AdminCompanyForm from "../Admin/AdminCompanyForm";
import AdminNewsComments, { SortSwitch as PublicationsCommentsSortSwitch } from
"./Admin/AdminNewsComments";
import AdminItemsRates, { SortSwitch as ItemsRatesSortSwitch } from
"./Admin/AdminItemsRates";
import AdminUsers, { SortSwitch as UsersSortSwitch } from "../Admin/AdminUsers";
import Home from "../Home";
import AdminCompany from "../Admin/AdminCompany";
import Profile from "../Profile";
import AdminStatistics from "../Admin/AdminStatistics";
import CartSuccess from "../CartSuccess";
import { GetAllUsersRequest, ProfileRequest } from "../http/User";
import Cookies from "js-cookie";
import { GetAllItemsParams, GetAllItemsRequest, GetItemRequest } from "../http/Items";
import { AxiosError } from "axios";
import { GetAllGenresRequest, GetGenreRequest } from "../http/Genres";
import { GetAllPublicationsRequest, GetPublicationRequest } from
"./http/Publications";
import { GetAllCompaniesRequest, GetCompanyRequest } from "../http/Companies";
import { GetAllItemsRatesRequest, GetAllPublicationsCommentsRequest } from
"./http/Comments";
import { GetStatisticsDataRequest } from "../http/Moderation";
import { useRecoilState } from "recoil";
import { UserAtom, userState } from "../store/User";

const SearchItem = async (params: Params, admin?: boolean) => {
  try {
    const { id } = params;
    if (!id) return { error: new ClientError(400, "Не вказано ID товару") };

    const parsed = parseInt(id);
    if (isNaN(parsed)) return { error: new ClientError(400, "ID товару не є
числом") };

    const item = await GetItemRequest(parsed, admin).catch((e) => {

```

```

        if (e instanceof ClientError) return e;
        if (e instanceof Error) return new ClientError(500, "Помилка сервера: " +
e.message);
        if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
        return new ClientError(500, "Помилка сервера");
    });
    if (item instanceof ClientError) return { error: item };

    return { item };
} catch (error) {
    return { error: new ClientError(500, "Помилка сервера") };
}
};

const SearchGenre = async (params: Params, admin?: boolean) => {
    try {
        const { id } = params;
        if (!id) return { error: new ClientError(400, "Не вказано ID жанру") };

        const parsed = parseInt(id);
        if (isNaN(parsed)) return { error: new ClientError(400, "ID жанру не є числом") };

        const result = await GetGenreRequest(parsed, admin).catch((e) => {
            if (e instanceof ClientError) return e;
            if (e instanceof Error) return new ClientError(500, "Помилка сервера: " +
e.message);
            if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
            return new ClientError(500, "Помилка сервера");
        });
        if (result instanceof ClientError) return { error: result };

        const genre = result;

        return { genre };
    } catch (error) {
        return { error: new ClientError(500, "Помилка сервера") };
    }
};

const SearchPublication = async (params: Params, user?: UserAtom) => {
    try {
        const { id } = params;
        if (!id) return { error: new ClientError(400, "Не вказано ID публікації") };

        const parsed = parseInt(id);
        if (isNaN(parsed)) return { error: new ClientError(400, "ID публікації не є
числом") };

        const admin = user && user.role.toLowerCase() === "admin";
        const result = await GetPublicationRequest(parsed, admin).catch((e) => {
            if (e instanceof ClientError) return e;
            if (e instanceof Error) return new ClientError(500, "Помилка сервера: " +
e.message);
            if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
            return new ClientError(500, "Помилка сервера");
        });
        if (result instanceof ClientError) return { error: result };

        if (user && result && (result.userId !== user.id || user.role.toLowerCase() ===
"user")) {
            return { error: new ClientError(403, "Ви не можете переглянути цю
публікацію") };
        }

        const publication = result;

        return { publication };
    } catch (error) {
        return { error: new ClientError(500, "Помилка сервера") };
    }
};

```

```

const SearchCompany = async (params: Params, admin?: boolean) => {
  try {
    const { id } = params;
    if (!id) return { error: new ClientError(400, "Не вказано ID компанії") };

    const parsed = parseInt(id);
    if (isNaN(parsed)) return { error: new ClientError(400, "ID компанії не є числом") };

    const result = await GetCompanyRequest(parsed, admin).catch((e) => {
      if (e instanceof ClientError) return e;
      if (e instanceof Error) return new ClientError(500, "Помилка сервера: " + e.message);
      if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
      return new ClientError(500, "Помилка сервера");
    });
    if (result instanceof ClientError) return { error: result };

    const company = result;

    return {
      company,
    };
  } catch (error) {
    return { error: new ClientError(500, "Помилка сервера") };
  }
};

const SearchProfile = async () => {
  const token = Cookies.get("token");
  if (!token) return { error: new ClientError(401, "Необхідно авторизуватися") };

  const result = await ProfileRequest(token).catch((e) => {
    if (e instanceof ClientError) {
      if (e.status === 401) return new ClientError(401, "Необхідно авторизуватися");
      if (e.status === 403) return new ClientError(403, "Необхідно авторизуватися");
    }
    if (e instanceof Error) return new ClientError(500, "Помилка сервера: " + e.message);
    return new ClientError(500, "Помилка сервера");
  });
  if (result instanceof ClientError) return { error: result };
  const user = result;

  const cart = user.Cart;
  const boughtData = user.Bought;
  const boughtItems = user.BoughtItems;
  const wishlist = user.Wishlist;
  const publications = user.Publications;
  const publicationsComments = user.CommentsList;
  const itemsRates = user.Rates;

  return { user, cart, boughtData, boughtItems, wishlist, publications, publicationsComments, itemsRates };
};

type GetItemsParams = {
  admin?: boolean;
  limit?: number;
  page?: number;
  sortBy?: string;
  searchParams?: {
    name?: string;
    priceFrom?: number;
    priceTo?: number;
    dateFrom?: Date;
    dateTo?: Date;
    genres?: {
      id: number;
      name: string;
    };
  };
};

```

```

    }[];
    developers?: {
      id: number;
      name: string;
    }[];
    publisher?: {
      id: number;
      name: string;
    };
    discount?: boolean;
  };
};

export const GetItems = async (params: GetItemsParams) => {
  const { admin = false, limit = 12, page = 1, sortBy = "nameAsc", searchParams =
undefined } = params;
  const { sortBy: specificSortBy, descending } = ItemsSortSwitch(sortBy);

  const requestParams: GetAllItemsParams = {
    sortBy: specificSortBy,
    desc: descending,
    page,
    limit,
    hidden: admin,

    name: searchParams?.name,
    priceFrom: searchParams?.priceFrom,
    priceTo: searchParams?.priceTo,
    releaseDateFrom: searchParams?.dateFrom?.toISOString().slice(0, 10),
    releaseDateTo: searchParams?.dateTo?.toISOString().slice(0, 10),
    discount: searchParams?.discount,
    includeGenres: !!searchParams?.genres,
    genresIds: searchParams?.genres?.map((genre) => genre.id),
    includeDevelopers: !!searchParams?.developers,
    developersIds: searchParams?.developers?.map((developer) => developer.id),
    includePublisher: !!searchParams?.publisher,
    publisherId: searchParams?.publisher?.id,
  };
  const result = await GetAllItemsRequest(requestParams);
  if (!result) return { error: new ClientError(500, "Помилка сервера") };

  const data = result.items;
  const totalCount = result.totalCount;
  const companies = result.companies;
  const genres = result.genres;

  return { data, totalCount, limit, page, sortBy, companies, genres };
};

type GetGenresParams = {
  admin?: boolean;
  limit?: number;
  page?: number;
  sortBy?: string;
  name?: string;
};

export const GetGenres = async (params: GetGenresParams) => {
  const { admin = false, limit = 12, page = 1, sortBy = "nameAsc", name = "" } =
params;
  const { descending } = GenresSortSwitch(sortBy);

  const data = await GetAllGenresRequest({
    sortBy: "name",
    desc: descending,
    page,
    limit,
    hidden: admin,
    admin,
    name,
  }).catch((e) => {
    if (e instanceof ClientError) return e;
    if (e instanceof Error) return new ClientError(500, "Помилка сервера: " +

```

```

e.message);
    if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
    return new ClientError(500, "Помилка сервера");
  });
  if (data instanceof ClientError) return { error: data };

  const genres = data.genres;
  const totalCount = data.totalCount;

  return { data: genres, totalCount, limit, page, sortBy };
};

type GetCompaniesParams = {
  admin?: boolean;
  limit?: number;
  page?: number;
  sortBy?: string;
  descending?: boolean;
  name?: string;
};

export const GetCompanies = async (params: GetCompaniesParams) => {
  const { admin = false, limit = 12, page = 1, sortBy = "nameAsc", name = "" } =
    params;
  const { sortBy: specificSortBy, descending } = CompaniesSortSwitch(sortBy);

  const result = await GetAllCompaniesRequest({
    name,
    sortBy: specificSortBy,
    desc: descending,
    page,
    limit,
    hidden: admin,
  }).catch((e) => {
    if (e instanceof ClientError) return e;
    if (e instanceof Error) return new ClientError(500, "Помилка сервера: " +
e.message);
    if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
    return new ClientError(500, "Помилка сервера");
  });
  if (result instanceof ClientError) return { error: result };

  const data = result.companies;
  const totalCount = result.totalCount;

  return { data, totalCount, limit, page, sortBy };
};

type GetPublicationsParams = {
  admin?: boolean;
  limit?: number;
  page?: number;
  sortBy?: string;
  searchParams?: {
    title?: string;
    createdFrom?: Date;
    createdTo?: Date;
    tags?: string[];
    authors?: {
      id: number;
      login: string;
      firstName: string;
      lastName: string;
    }[];
  };
};

export const GetPublications = async (params: GetPublicationsParams) => {
  const { admin = false, limit = 12, page = 1, sortBy = "createdAtAsc", searchParams =
undefined } = params;
  const { sortBy: specificSortBy, descending } = PublicationsSortSwitch(sortBy);

  let authorsIds: number[] = [];

```

```

    if (searchParams && searchParams.authors) {
      authorsIds = searchParams.authors.map((author) => author.id);
    }

    const result = await GetAllPublicationsRequest({
      admin,
      limit,
      page,
      sortBy: specificSortBy,
      desc: descending,

      title: searchParams ? searchParams.title : "",
      createdFrom: searchParams ? searchParams.createdFrom : undefined,
      createdTo: searchParams ? searchParams.createdTo : undefined,
      tags: searchParams ? searchParams.tags : undefined,
      authorsIds: authorsIds,
    }).catch((e) => {
      if (e instanceof ClientError) return e;
      if (e instanceof Error) return new ClientError(500, "Помилка сервера: " +
e.message);
      if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
      return new ClientError(500, "Помилка сервера");
    });
    if (result instanceof ClientError) return { error: result };

    const data = result.publications;
    const totalCount = result.totalCount;

    return { data, totalCount, limit, page, sortBy };
  };

  type GetUsersParams = {
    limit?: number;
    page?: number;
    sortBy?: string;
  };

  export const GetUsers = async (params: GetUsersParams) => {
    const token = Cookies.get("token");
    if (!token) return { error: new ClientError(401, "Необхідно авторизуватися") };

    const { limit = 12, page = 1, sortBy = "createdAtAsc" } = params;
    const { sortBy: specificSortBy, descending } = UsersSortSwitch(sortBy);

    const result = await GetAllUsersRequest(token, limit, page, specificSortBy,
descending).catch((e) => {
      if (e instanceof ClientError) return e;
      if (e instanceof Error) return new ClientError(500, "Помилка сервера: " +
e.message);
      if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
      return new ClientError(500, "Помилка сервера");
    });
    if (result instanceof ClientError) return { error: result };

    if (!result.data.users) return { error: new ClientError(500, "Не вдалося отримати
користувачів") };

    const users = result.data.users;
    const totalCount = result.data.totalCount;

    return { data: users, totalCount, limit, page, sortBy };
  };

  type GetPublicationsCommentsParams = {
    limit?: number;
    page?: number;
    sortBy?: string;
  };

  export const GetPublicationsComments = async (params: GetPublicationsCommentsParams)
=> {
    const { limit = 12, page = 1, sortBy = "createdAtAsc" } = params;
    const { descending } = PublicationsCommentsSortSwitch(sortBy);

```

```

const result = await GetAllPublicationsCommentsRequest({
  limit,
  page,
  sortBy: "createdAt",
  descending,
}).catch((e) => {
  if (e instanceof ClientError) return e;
  if (e instanceof Error) return new ClientError(500, "Помилка сервера: " +
e.message);
  if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
  return new ClientError(500, "Помилка сервера");
});
if (result instanceof ClientError) return { error: result };

const comments = result.comments;
const totalCount = result.totalCount;

return { data: comments, totalCount, limit, page, sortBy };
};

type GetItemsRatesParams = {
  limit?: number;
  page?: number;
  sortBy?: string;
};

export const GetItemsRates = async (params: GetItemsRatesParams) => {
  const { limit = 12, page = 1, sortBy = "createdAtAsc" } = params;
  const { descending } = ItemsRatesSortSwitch(sortBy);

  const result = await GetAllItemsRatesRequest({
    limit,
    page,
    sortBy: "createdAt",
    descending,
  }).catch((e) => {
    if (e instanceof ClientError) return e;
    if (e instanceof Error) return new ClientError(500, "Помилка сервера: " +
e.message);
    if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
    return new ClientError(500, "Помилка сервера");
  });
  if (result instanceof ClientError) return { error: result };

  const data = result.rates;
  const totalCount = result.totalCount;

  return { data, totalCount, limit, page, sortBy };
};

const GetStatistics = async () => {
  const token = Cookies.get("token");
  if (!token) return { error: new ClientError(401, "Необхідно авторизуватися") };

  const result = await GetStatisticsDataRequest(token).catch((e) => {
    if (e instanceof ClientError) return e;
    if (e instanceof Error) return new ClientError(500, "Помилка сервера: " +
e.message);
    if (e instanceof AxiosError) return new ClientError(e.code || "500", e.message);
    return new ClientError(500, "Помилка сервера");
  });
  if (result instanceof ClientError) return { error: result };

  return result;
};

export default function Router() {
  const [user, _] = useRecoilState(userState);
  const router = createBrowserRouter([
    {
      path: "",

```

```

element: <RootPage />,
children: [
  {
    path: "/",
    children: [
      {
        path: "/",
        element: <Home />,
      },
      {
        path: "/login",
        element: <Login />,
      },
      {
        path: "/register",
        element: <Register />,
      },
      {
        path: "/profile",
        element: <Profile />,
        loader: async () => {
          return await SearchProfile();
        },
      },
      {
        path: "/cart",
        element: <Cart />,
        loader: async () => {
          const userData = await SearchProfile();
          if (userData.error) return { error: userData.error };

          const { user, cart } = userData;

          return { user, cart };
        },
      },
      {
        path: "/cart/success",
        element: <CartSuccess />,
        loader: async () => {
          return await SearchProfile();
        },
      },
      {
        path: "/shop/items",
        element: <Items />,
        loader: async () => {
          const { data, totalCount, limit, page, sortBy, error, companies,
genres } =
            await GetItems({});

          return {
            data,
            totalCount,
            initLimit: limit,
            initPage: page,
            initSortBy: sortBy,
            error,
            companies,
            genres,
          };
        },
      },
      {
        path: "/shop/items/:id",
        element: <Item />,
        loader: ({ params }) => {
          return SearchItem(params);
        },
      },
    ],
  },
]

```

```

        path: "/shop/genres",
        element: <Genres />,
        loader: async () => {
            const { data, error } = await GetGenres({});
            return { data, error };
        },
    },
    {
        path: "/shop/genres/:id",
        element: <Genre />,
        loader: ({ params }) => {
            return SearchGenre(params);
        },
    },
    {
        path: "/shop/companies",
        element: <Companies />,
        loader: async () => {
            const { data, totalCount, limit, page, sortBy, error } = await
GetCompanies({});
            return {
                data,
                totalCount,
                initLimit: limit,
                initPage: page,
                initSortBy: sortBy,
                error,
            };
        },
    },
    {
        path: "/shop/companies/:id",
        element: <Company />,
        loader: ({ params }) => {
            return SearchCompany(params);
        },
    },
    {
        path: "/news",
        element: <NewsList />,
        loader: async () => {
            const { data, totalCount, limit, page, sortBy } = await
GetPublications({});
            return {
                data,
                totalCount,
                initLimit: limit,
                initPage: page,
                initSortBy: sortBy,
            };
        },
    },
    {
        path: "/news/:id",
        element: <NewsItem />,
        loader: ({ params }) => {
            return SearchPublication(params);
        },
    },
    {
        path: "/news/create",
        element: <NewsForm />,
        loader: () => {
            return { publication: undefined };
        },
    },
    {
        path: "/news/:id/edit",
        element: <NewsForm />,
        loader: ({ params }) => {
            return SearchPublication(params, user);
        },
    },

```

```

    },
  ],
},
{
  path: "/admin",
  element: <AdminBase />,
  children: [
    {
      path: "/admin",
      element: <AdminHome />,
    },
    {
      path: "/admin/statistics",
      element: <AdminStatistics />,
      loader: async () => {
        return await GetStatistics();
      },
    },
    {
      path: "/admin/users",
      element: <AdminUsers />,
      loader: async () => {
        const { data, totalCount, limit, page, sortBy, error } = await
GetUsers({});

        return {
          data,
          totalCount,
          initLimit: limit,
          initPage: page,
          initSortBy: sortBy,
          error,
        };
      },
    },
    {
      path: "/admin/items",
      element: <AdminItems />,
      loader: async () => {
        const { data, totalCount, limit, page, sortBy, error } = await
GetItems({
          admin: true,
        });

        return {
          data,
          totalCount,
          initLimit: limit,
          initPage: page,
          initSortBy: sortBy,
          error,
        };
      },
    },
    {
      path: "/admin/items/:id",
      element: <AdminItem />,
      loader: ({ params }) => {
        return SearchItem(params, true);
      },
    },
    {
      path: "/admin/items/create",
      element: <AdminItemForm />,
      loader: async () => {
        const genres = await GetGenres({ admin: true });
        if (genres.error) return { error: genres.error };

        const companies = await GetCompanies({ admin: true });
        if (companies.error) return { error: companies.error };
      },
    },
  ],
},

```

```

        return { item: undefined, genres: genres.data, companies:
companies.data };
    },
  },
  {
    path: "/admin/items/:id/edit",
    element: <AdminItemForm />,
    loader: async ({ params }) => {
      const itemData = await SearchItem(params, true);
      if (itemData.error) return { error: itemData.error };

      const genres = await GetGenres({ admin: true });
      if (genres.error) return { error: genres.error };

      const companies = await GetCompanies({ admin: true });
      if (companies.error) return { error: companies.error };

      return { item: itemData.item, genres: genres.data, companies:
companies.data };
    },
  },
  {
    path: "/admin/genres",
    element: <AdminGenres />,
    loader: async () => {
      const { data, totalCount, sortBy, limit, page, error } = await
GetGenres({
        admin: true,
      });
      return {
        data,
        totalCount,
        initSortBy: sortBy,
        initLimit: limit,
        initPage: page,
        error,
      };
    },
  },
  {
    path: "/admin/genres/:id",
    element: <AdminGenre />,
    loader: ({ params }) => {
      return SearchGenre(params, true);
    },
  },
  {
    path: "/admin/genres/create",
    element: <AdminGenreForm />,
    loader: () => {
      return { genre: undefined };
    },
  },
  {
    path: "/admin/genres/:id/edit",
    element: <AdminGenreForm />,
    loader: ({ params }) => {
      return SearchGenre(params);
    },
  },
  {
    path: "/admin/companies",
    element: <AdminCompanies />,
    loader: async () => {
      const { data, totalCount, sortBy, limit, page, error } = await
GetCompanies({
        admin: true,
      });
      return {
        data,
        totalCount,

```

```

        initSortBy: sortBy,
        initLimit: limit,
        initPage: page,
        error,
    };
},
{
    path: "/admin/companies/:id",
    element: <AdminCompany />,
    loader: ({ params }) => {
        return SearchCompany(params, true);
    },
},
{
    path: "/admin/companies/create",
    element: <AdminCompanyForm />,
    loader: () => {
        return { company: undefined };
    },
},
{
    path: "/admin/companies/:id/edit",
    element: <AdminCompanyForm />,
    loader: ({ params }) => {
        return SearchCompany(params);
    },
},
{
    path: "/admin/news",
    element: <AdminNews />,
    loader: async () => {
        const { data, totalCount, sortBy, limit, page, error } = await
GetPublications({
        admin: true,
    });
    return {
        data,
        totalCount,
        initSortBy: sortBy,
        initLimit: limit,
        initPage: page,
        error,
    };
},
{
    path: "/admin/news/:id",
    element: <AdminNewsItem />,
    loader: ({ params }) => {
        return SearchPublication(params, user);
    },
},
{
    path: "/admin/comments/news",
    element: <AdminNewsComments />,
    loader: async () => {
        const { data, totalCount, limit, page, sortBy, error } = await
GetPublicationsComments(
        {}
    );
    return {
        data,
        totalCount,
        initLimit: limit,
        initPage: page,
        initSortBy: sortBy,
        error,
    };
},
},

```

```

        {
            path: "/admin/comments/items",
            element: <AdminItemsRates />,
            loader: async () => {
                const { data, totalCount, limit, page, sortBy, error } = await
GetItemsRates({});
                return {
                    data,
                    totalCount,
                    initLimit: limit,
                    initPage: page,
                    initSortBy: sortBy,
                    error,
                };
            },
        },
    ],
},
],
},
]);

return <RouterProvider router={router} />;
}

```

Лістинг файлів клієнтського додатку з налаштування виконання HTTP запитів з типізацією результатів запитів в папці http

Лістинг коду файлу Comments.ts:

```
import { ItemRate, Item } from "../Items";
import { Publication, Comment } from "../Publications";
import { User } from "../User";
import axiosInstance from "../axiosInstance";

type GetAllCommentsParams = {
  sortBy?: string;
  descending?: boolean;
  limit?: number;
  page?: number;
};

type GetAllCommentsReturn = {
  comments: (Comment & {
    User: User;
    Publication: Publication;
  })[];
  totalCount: number;
};

export const GetAllPublicationsCommentsRequest = async (params: GetAllCommentsParams) => {
  return await axiosInstance
    .get<GetAllCommentsReturn>(`/comments/news`, {
      params: {
        sortBy: params.sortBy,
        descending: params.descending,
        limit: params.limit,
        page: params.page,
      },
    })
    .then((res) => {
      return res.data;
    })
    .catch((err) => {
      throw new Error(err.message);
    });
};

type GetAllRatesParams = {
  sortBy?: string;
  descending?: boolean;
  limit?: number;
  page?: number;
};

type GetAllRatesReturn = {
  rates: (ItemRate & {
    Item: Item;
    User: User;
  })[];
  totalCount: number;
};

export const GetAllItemsRatesRequest = async (params: GetAllRatesParams) => {
  return await axiosInstance
    .get<GetAllRatesReturn>(`/comments/items`, {
      params: {
        sortBy: params.sortBy,
        descending: params.descending,
        limit: params.limit,
        page: params.page,
      },
    })
    .then((res) => {
```

```

        return res.data;
    })
    .catch((err) => {
        throw new Error(err.message);
    });
};

```

Лістинг коду файлу Companies.ts:

```

import { AxiosError } from "axios";
import { Item } from "../Items";
import axiosInstance from "../axiosInstance";

export type Company = {
    id: number;
    name: string;
    description: string;
    director: string;
    image: string;
    founded: string;
    hide: boolean;
    createdAt: string;
    updatedAt: string | null;
    deletedAt: string | null;
};

export type GetAllCompaniesParams = {
    name?: string;
    desc?: boolean;
    sortBy?: string;
    limit: number;
    page: number;
    hidden?: boolean;
};

type GetAllCompaniesReturn = {
    companies: Company[];
    totalCount: number;
};

export const GetAllCompaniesRequest = async (params: GetAllCompaniesParams) => {
    return await axiosInstance
        .get<GetAllCompaniesReturn>(`/shop/companies`, {
            params: params,
        })
        .then((res) => {
            return res.data;
        })
        .catch((err: AxiosError) => {
            throw new Error(err.message);
        });
};

export type GetCompanyReturn = Company & {
    ItemsDeveloped: Item[];
    ItemsPublished: Item[];
};

export const GetCompanyRequest = async (id: number, admin?: boolean) => {
    return await axiosInstance
        .get<GetCompanyReturn>(`/shop/companies/${id}`, {
            params: {
                includeDeveloped: true,
                includePublished: true,
                includeHidden: admin,
            },
        })
        .then((res) => {
            return res.data;
        })
        .catch((err: AxiosError) => {
            throw new Error(err.message);
        });
};

```

```

type CreateCompanyParams = {
  name: string;
  description: string;
  director: string;
  image: string;
  founded: string;
  hide: boolean;
};
type CreateCompanyReturn = Company;
export const CreateCompanyRequest = async (token: string, params: CreateCompanyParams)
=> {
  return await axiosInstance
    .post<CreateCompanyReturn>(`/shop/companies`, params, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
    .then((res) => {
      return res.data;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};

type UpdateCompanyParams = {
  name?: string;
  description?: string;
  director?: string;
  image?: string;
  founded?: string;
  hide?: boolean;
};
type UpdateCompanyReturn = Company;
export const UpdateCompanyRequest = async (token: string, id: number, params:
UpdateCompanyParams) => {
  return await axiosInstance
    .patch<UpdateCompanyReturn>(`/shop/companies/${id}`, params, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
    .then((res) => {
      return res.data;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};

export const DeleteCompanyRequest = async (token: string, id: number) => {
  return await axiosInstance
    .delete<{ ok: boolean }>(`/shop/companies/${id}`, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
    .then((res) => {
      return res.data.ok;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};

```

Лістинг коду файлу Genres.ts:

```

import { AxiosError } from "axios";
import { Item } from "../Items";
import axiosInstance from "../axiosInstance";

export type Genre = {
  id: number;
  description?: string;
  name: string;
  createdAt: string;
  updatedAt: string | null;
  hide: boolean;
};

type GetAllGenresParams = {
  name?: string;
  desc?: boolean;
  sortBy?: string;
  limit: number;
  page: number;
  hidden?: boolean;
  admin?: boolean;
};

type GetAllGenresReturn = {
  genres: (Genre & {
    Items: Item[];
  })[];
  totalCount: number;
};

export const GetAllGenresRequest = async (params: GetAllGenresParams) => {
  const requestParams = params.admin
    ? {
        desc: params.desc,
        sortBy: params.sortBy,
      }
    : {};

  return await axiosInstance
    .get<GetAllGenresReturn>(`/shop/genres`, {
      params: {
        ...requestParams,
        name: params.name,
        limit: params.admin ? params.limit : 9999,
        page: params.admin ? params.page : 1,
      },
    })
    .then((res) => {
      return res.data;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};

type GetGenreReturn = Genre & {
  Items?: Item[];
};

export const GetGenreRequest = async (id: number, admin?: boolean) => {
  return await axiosInstance
    .get<GetGenreReturn>(`/shop/genres/${id}`, {
      params: {
        includeItems: true,
        includeHidden: admin,
      },
    })
    .then((res) => {
      return res.data;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};

```

```

type CreateGenreParams = {
  name: string;
  description: string;
  hide?: boolean;
};
type CreateGenreReturn = Genre;
export const CreateGenreRequest = async (token: string, params: CreateGenreParams) => {
  return await axiosInstance
    .post<CreateGenreReturn>(`/shop/genres`, params, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
    .then((res) => {
      return res.data;
    })
    .catch((err: AxiosError | Error) => {
      throw new Error(err.message);
    });
};

type UpdateGenreParams = {
  name?: string;
  description?: string;
  hide?: boolean;
};
type UpdateGenreReturn = Genre;
export const UpdateGenreRequest = async (token: string, id: number, params:
UpdateGenreParams) => {
  return await axiosInstance
    .patch<UpdateGenreReturn>(`/shop/genres/${id}`, params, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
    .then((res) => {
      return res.data;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};

export const DeleteGenreRequest = async (token: string, id: number) => {
  return await axiosInstance
    .delete<{ ok: boolean }>(`/shop/genres/${id}`, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
    .then((res) => {
      return res.data.ok;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};

```

Лістинг коду файлу Items.ts:

```

import { type AxiosError } from "axios";
import { Genre } from "../Genres";
import { Company } from "../Companies";
import { User } from "../User";
import axiosInstance from "../axiosInstance";

export type Item = {
  id: number;
  name: string;

```

```

    description: string;
    releaseDate: string;
    price: number;
    amount: number;
    discount: boolean;
    discountFrom: string | null;
    discountTo: string | null;
    discountSize: number | null;
    mainImage: string;
    images: string[];
    coverImage: string | null;
    characteristics: { [key: string]: string | number | boolean };
    hide: boolean;
    createdAt: string;
    updatedAt: string | null;
    deletedAt: string | null;
    company_publisherId: number | null;
};

export type ItemRate = {
    id: number;
    content: string;
    rate: number;
    hide: boolean;
    violation: boolean;
    violation_reason: string;
    createdAt: string;
    updatedAt: string | null;
    userId: number;
    itemId: number;
};

export type ItemCart = {
    id: number;
    userId: number;
    itemId: number;
    transactionId?: string | null;
    createdAt: string;
    updatedAt: string | null;
};

export type ItemBought = {
    id: number;
    itemId: number;
    transactionId: string;
    createdAt: string;
    userId: number;
};

export type Wishlist = {
    id: number;
    userId: number;
    itemId: number;
    createdAt: string;
    updatedAt: string | null;
};

export type GetAllItemsParams = {
    name?: string;
    description?: string;
    price?: number;
    priceFrom?: number;
    priceTo?: number;

    amount?: number;
    amountFrom?: number;
    amountTo?: number;

    releaseDate?: string;
    releaseDateFrom?: string;
    releaseDateTo?: string;

    discount?: boolean;
    discountFrom?: string;
    discountTo?: string;
};

```

```

discountSize?: number;
discountSizeFrom?: number;
discountSizeTo?: number;

includePublisher?: boolean;
publisherId?: number;

includeDevelopers?: boolean;
developersIds?: number[];

includeGenres?: boolean;
genresIds?: number[];

includeRated?: boolean;
includeWishlisted?: boolean;

sortBy?: string;
desc?: boolean;
page?: number;
limit?: number;
hidden?: boolean;
};
type GetAllReturn = {
  items: (Item & {
    Publisher: Company | null;
    Developers: Company[];
    Genres: Genre[];
    WishlistedUsers: User[];
    Rates: (ItemRate & {
      User: User;
    })[];
  })[];
  companies: Company[];
  genres: Genre[];
  totalCount: number;
};
export const GetAllItemsRequest = async (params: GetAllItemsParams) => {
  console.log({ params });
  return await axiosInstance
    .get<GetAllReturn>(`/shop/items`, {
      params: params,
    })
    .then((res) => {
      return res.data;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};

type GetItemReturn = Item & {
  Publisher: Company | null;
  Developers: Company[];
  Genres: Genre[];
  Rates: (ItemRate & {
    User: User;
  })[];
  WishlistedUsers: User[];
  CartedUsers: User[];
};
export const GetItemRequest = async (id: number, admin?: boolean) => {
  return await axiosInstance
    .get<GetItemReturn>(`/shop/items/${id}`, {
      params: {
        includeHidden: admin,
        includePublisher: true,
        includeGenres: true,
        includeDevelopers: true,
        includeRated: true,
        includeInCart: true,
        includeWishlisted: true,

```

```

        },
    })
    .then((res) => {
        return res.data;
    })
    .catch((err: AxiosError) => {
        throw new Error(err.message);
    });
};

type CreateItemParams = {
    name: string;
    description: string;
    releaseDate: string;
    price: number;
    amount: number;
    discount: boolean;
    discountFrom: string | null;
    discountTo: string | null;
    discountSize: number | null;
    mainImage: string;
    images: string[];
    coverImage: string | null;
    characteristics: { [key: string]: string | number | boolean };
    hide: boolean;
    publisherId: number | null;
    developersIds: number[];
    genresIds: number[];
};

export const CreateItemRequest = async (token: string, params: CreateItemParams) => {
    return await axiosInstance
        .post<Item>(
            `/shop/items`,
            {
                ...params,
                releaseDate: new Date(params.releaseDate).toISOString().slice(0, 10),
                discountFrom: params.discountFrom ? new Date(params.discountFrom).toISOString().slice(0, 10) : null,
                discountTo: params.discountTo ? new Date(params.discountTo).toISOString().slice(0, 10) : null,
                characteristics: JSON.stringify(params.characteristics),
            },
            {
                headers: {
                    Authorization: `Bearer ${token}`,
                },
            }
        )
        .then((res) => {
            return res.data;
        })
        .catch((err: AxiosError) => {
            throw new Error(err.message);
        });
};

type UpdateItemParams = {
    name?: string;
    description?: string;
    releaseDate?: string;
    price?: number;
    amount?: number;
    discount?: boolean;
    discountFrom?: string | null;
    discountTo?: string | null;
    discountSize?: number | null;
    mainImage?: string;
    images?: string[];
    coverImage?: string | null;
    characteristics?: { [key: string]: string | number | boolean };
    hide?: boolean;
};

```

```

    publisherId?: number | null;
    developersIds?: number[];
    genresIds?: number[];
  };
  export const UpdateItemRequest = async (token: string, id: number, params: UpdateItemParams) => {
    return await axiosInstance
      .patch<Item>(`shop/items/${id}`,
        {
          ...params,
          releaseDate: params.releaseDate ? new Date(params.releaseDate).toISOString().slice(0, 10) : undefined,
          discountFrom: params.discountFrom ? new Date(params.discountFrom).toISOString().slice(0, 10) : undefined,
          discountTo: params.discountTo ? new Date(params.discountTo).toISOString().slice(0, 10) : undefined,
          characteristics: params.characteristics ? JSON.stringify(params.characteristics) : null,
        },
        {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        }
      )
      .then((res) => {
        return res.data;
      })
      .catch((err: AxiosError) => {
        throw new Error(err.message);
      });
  };

  export const DeleteItemRequest = async (token: string, id: number) => {
    return await axiosInstance
      .delete<{ ok: true }>(`shop/items/${id}`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      })
      .then((res) => {
        return res.data.ok;
      })
      .catch((err: AxiosError) => {
        throw new Error(err.message);
      });
  };

  export const calculateDiscount = (item: Item) => {
    const date = new Date();
    let isDiscount = item.discount;
    if (item.discountFrom) isDiscount = isDiscount && date.getTime() >= new Date(item.discountFrom).getTime();
    if (item.discountTo) isDiscount = isDiscount && date.getTime() <= new Date(item.discountTo).getTime();

    const finalPrice =
      isDiscount && item.discountSize ? item.price - (item.price * item.discountSize) / 100 : item.price;
    return { isDiscount, finalPrice };
  };

  export const SetItemRateRequest = async (token: string, itemId: number, rate: number, content: string) => {
    return await axiosInstance
      .post<{ message: string; rate: ItemRate; user: User }>(`shop/items/${itemId}/rate`,
        {
          rate,

```

```

        content,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      },
    )
  ).then((res) => {
    return res.data;
  })
  .catch((err: AxiosError) => {
    throw new Error(err.message);
  });
};

export const DeleteItemRateRequest = async (token: string, itemId: number) => {
  return await axiosInstance
    .delete<{ ok: boolean }>(`/shop/items/${itemId}/rate`, {
      headers: {
        Authorization: `Bearer ${token}`,
      },
    })
    .then((res) => {
      return res.data.ok;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};

export const ToggleCartItemRequest = async (token: string, itemId: number) => {
  return await axiosInstance
    .post<{ message: string; ok: boolean }>(
      `/shop/items/${itemId}/toggleCart`,
      {},
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      },
    )
    .then((res) => {
      return res.data;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};

export const ToggleWishlistItemRequest = async (token: string, itemId: number) => {
  return await axiosInstance
    .post<{ message: string; ok: boolean }>(
      `/shop/items/${itemId}/toggleWishlist`,
      {},
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      },
    )
    .then((res) => {
      return res.data;
    })
    .catch((err: AxiosError) => {
      throw new Error(err.message);
    });
};
};

```

Лістинг коду файлу Moderation.ts:

```

import { Publication, Comment } from "../Publications";
import { Item, ItemRate, Wishlist } from "../Items";
import { User } from "../User";
import { Company } from "../Companies";
import { Genre } from "../Genres";
import axiosInstance from "../axiosInstance";

type SetViolationParams = {
  violation: boolean;
  violation_reason?: string;
};

export const SetPublicationViolationRequest = async (
  token: string,
  id: number,
  { violation, violation_reason }: SetViolationParams
) => {
  return await axiosInstance
    .patch<{ message: string; result: Publication }>(
      `/moderation/news/${id}`,
      {
        violation,
        violation_reason,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    )
    .then((res) => {
      return res.data;
    })
    .catch((err) => {
      throw new Error(err.message);
    });
};

export const SetCommentViolationRequest = async (
  token: string,
  id: number,
  { violation, violation_reason }: SetViolationParams
) => {
  return await axiosInstance
    .patch<{ message: string; result: Comment }>(
      `/moderation/comments/${id}`,
      {
        violation,
        violation_reason,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    )
    .then((res) => {
      return res.data;
    })
    .catch((err) => {
      throw new Error(err.message);
    });
};

export const SetRateViolationRequest = async (
  token: string,
  id: number,
  { violation, violation_reason }: SetViolationParams
) => {
  return await axiosInstance
    .patch<{ message: string; result: ItemRate }>(
      `/moderation/rates/${id}`,

```

```

        {
            violation,
            violation_reason,
        },
        {
            headers: {
                Authorization: `Bearer ${token}`,
            },
        }
    )
    .then((res) => {
        return res.data;
    })
    .catch((err) => {
        throw new Error(err.message);
    });
};

type GetStatisticsDataResponse = {
    Users: User[];
    Items: (Item & {
        Developers: Company[];
        Publisher: Company;
        Genres: Genre[];
    })[];
    Publications: Publication[];
    Companies: Company[];
    Genres: Genre[];
    ItemsRates: ItemRate[];
    PublicationsComments: Comment[];
    Wishlists: Wishlist[];
};

export const GetStatisticsDataRequest = async (token: string) => {
    return await axiosInstance
        .get<GetStatisticsDataResponse>(`/moderation/statistics`, {
            headers: {
                Authorization: `Bearer ${token}`,
            },
        })
        .then((res) => {
            return res.data;
        })
        .catch((err) => {
            throw new Error(err.message);
        });
};

export const ToggleRoleRequest = async (token: string, id: number) => {
    return await axiosInstance
        .patch<{ message: string; result: User }>(
            `/user/toggleModerator/${id}`,
            {},
            {
                headers: {
                    Authorization: `Bearer ${token}`,
                },
            }
        )
        .then((res) => {
            return res.data;
        })
        .catch((err) => {
            throw new Error(err.message);
        });
};

export const SetRoleRequest = async (token: string, id: number, role: string) => {
    return await axiosInstance
        .patch<{ message: string; result: User }>(
            `/user/setRole/${id}`,
            {

```

```

        role,
      },
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    )
    .then((res) => {
      return res.data;
    })
    .catch((err) => {
      throw new Error(err.message);
    });
};

```

Лістинг коду файлу Publications.ts:

```

import { AxiosError } from "axios";
import { User } from "../User";
import axiosInstance from "../axiosInstance";

export type Publication = {
  id: number;
  title: string;
  content: string;
  hide: boolean;
  tags: string[];
  violation: boolean;
  violation_reason: string;
  userId: number;
  createdAt: string;
  updatedAt: string | null;
  deletedAt: string | null;
};

export type Comment = {
  id: number;
  content: string;
  rate: number;
  createdAt: string;
  updatedAt: string | null;
  violation: boolean;
  violation_reason: string;
  hide: boolean;
  userId: number;
  publicationId: number;
};

type GetAllPublicationsParams = {
  admin?: boolean;
  limit?: number;
  page?: number;
  sortBy?: string;
  desc?: boolean;

  title?: string;
  createdFrom?: Date;
  createdTo?: Date;
  tags?: string[];
  authorsIds?: number[];
};

type GetAllPublicationsReturn = {
  publications: (Publication & {
    AuthoredUser: User;
  })[];
  totalCount: number;
};

export const GetAllPublicationsRequest = async (params: GetAllPublicationsParams) => {
  return await axiosInstance
    .get<GetAllPublicationsReturn>(`/news`, {

```

```

        params: {
            title: params.title,
            createdFrom: params.createdFrom?.toISOString(),
            createdTo: params.createdTo?.toISOString(),
            limit: params.limit,
            page: params.page,
            sortBy: params.sortBy,
            hidden: params.admin,
            desc: params.desc,
            tags: params.tags,
            authorsIds: params.authorsIds,
        },
    ))
    .then((res) => {
        return res.data;
    })
    .catch((err: AxiosError) => {
        throw new Error(err.message);
    });
};

type GetPublicationReturn = Publication & {
    CommentsList: (Comment & {
        User: User;
    })[];
    AuthoredUser: User;
};

export const GetPublicationRequest = async (id: number, admin?: boolean) => {
    return await axiosInstance
        .get<GetPublicationReturn>(`/news/${id}`, {
            params: {
                includeComments: true,
                includeHidden: admin,
                includeViolations: admin,
            },
        })
        .then((res) => {
            return res.data;
        })
        .catch((err: AxiosError) => {
            throw new Error(err.message);
        });
};

type CreatePublicationParams = {
    title: string;
    content: string;
    tags: string[];
    hide: boolean;
};

type CreatePublicationReturn = Publication;
export const CreatePublicationRequest = async (token: string, params:
CreatePublicationParams) => {
    return await axiosInstance
        .post<CreatePublicationReturn>(`/news`, params, {
            headers: {
                Authorization: `Bearer ${token}`,
            },
        })
        .then((res) => {
            return res.data;
        })
        .catch((err: AxiosError) => {
            throw new Error(err.message);
        });
};

type UpdatePublicationParams = {
    title?: string;
    content?: string;
    tags?: string[];
};

```

```

    hide?: boolean;
  };
  type UpdatePublicationReturn = Publication;
  export const UpdatePublicationRequest = async (token: string, id: number, params:
UpdatePublicationParams) => {
    return await axiosInstance
      .patch<UpdatePublicationReturn>(`/news/${id}`, params, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      })
      .then((res) => {
        return res.data;
      })
      .catch((err: AxiosError) => {
        throw new Error(err.message);
      });
  };

  export const DeletePublicationRequest = async (token: string, id: number) => {
    return await axiosInstance
      .delete<{ ok: boolean }>(`/news/${id}`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      })
      .then((res) => {
        return res.data.ok;
      })
      .catch((err: AxiosError) => {
        throw new Error(err.message);
      });
  };

  type SetCommentParams = {
    content: string;
    rate: number;
  };
  export const SetCommentRequest = async (token: string, id: number, params:
SetCommentParams) => {
    return await axiosInstance
      .post<{ message: string; comment: Comment; user: User }>(`/news/${id}/comment`,
params, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      })
      .then((res) => {
        return res.data;
      })
      .catch((err: AxiosError) => {
        throw new Error(err.message);
      });
  };

  export const DeleteCommentRequest = async (token: string, id: number) => {
    return await axiosInstance
      .delete<{ message: string; ok: boolean }>(`/news/${id}/comment`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      })
      .then((res) => {
        return res.data;
      })
      .catch((err: AxiosError) => {
        throw new Error(err.message);
      });
  };
};

```

Лістинг коду файлу User.ts:

```

import { type AxiosError } from "axios";
import { ItemRate, ItemBought, Item } from "./Items";
import { Comment, Publication } from "./Publications";
import axiosInstance from "./axiosInstance";

export type LoginResults = {
  message?: string;
  token: string;
};

export const LogInRequest = async (credentials: string, password: string) => {
  return await axiosInstance
    .post<LoginResults>(`/user/login`, {
      credentials,
      password,
    })
    .then((res) => {
      return res.data;
    });
};

export type RegisterParams = {
  email: string;
  login: string;
  firstName: string;
  lastName: string;
  password: string;
  image: string;
};

export const RegisterRequest = async ({ login, email, password, firstName, lastName,
image }: RegisterParams) => {
  try {
    const res = await axiosInstance
      .post<{ message: string; token: string }>(`/user/register`, {
        email,
        login,
        firstName,
        lastName,
        password,
        image,
      })
      .then((res) => {
        return res.data;
      });

    console.log({ res });
    return res;
  } catch (err) {
    console.log({ err });
    if (err instanceof Error) throw err;
    throw new Error();
  }
};

export type User = {
  id: number;
  login: string;
  email: string;
  image: string;
  firstName: string;
  lastName: string;
  role: string;
  isBlocked: boolean;
  createdAt: string;
  updatedAt: string | null;
  deletedAt: string | null;
};

export type Profile = User & {
  Cart: Item[];
  Bought: ItemBought[];
  Rates: (ItemRate & {
    Item: Item;
  })[];
};

```

```

    ))[];
    Wishlist: Item[];
    Publications: Publication[];
    CommentsList: Comment[];
    BoughtItems: Item[];
  };
  export const ProfileRequest = async (token: string) => {
    return await axiosInstance
      .get<Profile>(`/user/profile`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
        params: {
          includeRates: true,
          includeWishlist: true,
          includePublications: true,
          includePublicationComments: true,
          includeBought: true,
          includeCart: true,
        },
      })
      .then((res) => {
        return res.data;
      })
      .catch((err: AxiosError) => {
        throw new Error(err.message);
      });
  };
  export const GetAllUsersRequest = async (token: string, limit: number, page: number,
  sortBy: string, desc: boolean) => {
    return await axiosInstance
      .get<{ users: User[]; totalCount: number; message?: string }>(`/user/all`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
        params: {
          limit,
          page,
          sortBy,
          desc,
        },
      })
      .catch((err: AxiosError) => {
        throw new Error(err.message);
      });
  };
  export const ClearCartRequest = async (token: string) => {
    return await axiosInstance
      .delete<{ message: string; ok: boolean }>(`/user/cart`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      })
      .then((res) => {
        return res.data;
      })
      .catch((err: AxiosError) => {
        throw new Error(err.message);
      });
  };
  export const SetUpCartRequest = async (token: string) => {
    return await axiosInstance
      .post<{ message: string; transactionId: string }>(
        `/user/cart/setup`,
        {},
        {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        },
      )
  }
}

```

```
.then((res) => {  
    return res.data;  
})  
.catch((err: AxiosError) => {  
    throw new Error(err.message);  
});  
};
```

Експортований JSON об'єкт налаштованих запитів платформи Postman

Лістинг файлу gigashop.postman_collection.json:

```

{
  "info": {
    "postman_id": "51140125-ab66-4fb5-868d-f5479fbc7f6a",
    "name": "gigashop",
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",
    "_exporter_id": "18877953"
  },
  "item": [
    {
      "name": "items",
      "item": [
        {
          "name": "Create Item",
          "request": {
            "method": "POST",
            "header": [
              {
                "key": "Authorization",
                "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoieWVWZmVhZC5jb20iLCJyb2x1IjoieWVWZmVhZC5jb20iLCJpbnVnZSI6Imh0dHBzOi8vcmlnaW50LmNsb3VkaW5hcnkuY29tL2RucWxneXBqaS9pbWFnZS91cGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NTg3NDg5LCJleHAiOjE2ODY2NzQ0ODl9.Maq5HIgFd9p8UD_VOYFFV2X_J7OtFN5vE2USvWr6hTI",
                "type": "text"
              }
            ],
            "body": {
              "mode": "raw",
              "raw": "{\r\n  \"name\": \"Test Item\", \r\n  \"description\": \"Play me!\", \r\n  \"releaseDate\": \"2000-1-4\", \r\n  \"price\": 250, \r\n  \"amount\": 123, \r\n  \"discount\": true, \r\n  \"discountFrom\": \"2000-1-4\", \r\n  \"discountTo\": \"2000-1-14\", \r\n  \"discountSize\": 10, \r\n  \"mainImage\": \"https://m.media-amazon.com/images/M/MV5BMGU4ODVkdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NTg3NDg5LCJleHAiOjE2ODY2NzQ0ODl9.Maq5HIgFd9p8UD_VOYFFV2X_J7OtFN5vE2USvWr6hTI.jpg\", \r\n  \"https://m.media-amazon.com/images/M/MV5BMGU4ODVkdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NTg3NDg5LCJleHAiOjE2ODY2NzQ0ODl9.Maq5HIgFd9p8UD_VOYFFV2X_J7OtFN5vE2USvWr6hTI.jpg\", \r\n  \"https://static.wikia.nocookie.net/youtube/images/1/18/Cyberpunk_2077.jpg/revision/latest?cb=20220923163511\", \r\n  \"coverImage\": \"https://www.cyberpunk.net/build/images/social-thumbnail-en-ddcf4d23.jpg\", \r\n  \"characteristics\": \"{\\\"data1\\\": false, \\\"data2\\\": 11, \\\"data3\\\": 3}\", \r\n  \"hide\": false\r\n}\",
            "options": {
              "raw": {
                "language": "json"
              }
            }
          },
          "url": {
            "raw": "http://localhost:5000/api/shop/items/",
            "protocol": "http",
            "host": [
              "localhost"
            ],
            "port": "5000",
            "path": [
              "api",
              "shop",
              "items",
              ""
            ]
          }
        }
      ]
    }
  ]
}

```

```

    ]
  },
  "response": []
},
{
  "name": "Read Items",
  "protocolProfileBehavior": {
    "disableBodyPruning": true
  },
  "request": {
    "method": "GET",
    "header": [],
    "body": {
      "mode": "raw",
      "raw": "",
      "options": {
        "raw": {
          "language": "json"
        }
      }
    },
    "url": {
      "raw":
"http://localhost:5000/api/shop/items?hidden=true",
      "protocol": "http",
      "host": [
        "localhost"
      ],
      "port": "5000",
      "path": [
        "api",
        "shop",
        "items"
      ],
      "query": [
        {
          "key": "name",
          "value": "test",
          "disabled": true
        },
        {
          "key": "description",
          "value": "play",
          "disabled": true
        },
        {
          "key": "price",
          "value": "250",
          "disabled": true
        },
        {
          "key": "priceFrom",
          "value": "200",
          "disabled": true
        },
        {
          "key": "priceTo",
          "value": "300",
          "disabled": true
        },
        {
          "key": "amount",
          "value": "123",
          "disabled": true
        },
        {
          "key": "amountFrom",
          "value": "120",
          "disabled": true
        }
      ]
    }
  }
}

```

```

{
    "key": "amountTo",
    "value": "130",
    "disabled": true
},
{
    "key": "releaseDate",
    "value": "2000-01-4",
    "disabled": true
},
{
    "key": "releaseDateFrom",
    "value": "2000-01-03",
    "disabled": true
},
{
    "key": "releaseDateTo",
    "value": "2000-01-05",
    "disabled": true
},
{
    "key": "discount",
    "value": "true",
    "disabled": true
},
{
    "key": "discountFrom",
    "value": "2000-01-03",
    "disabled": true
},
{
    "key": "discountTo",
    "value": "2000-01-15",
    "disabled": true
},
{
    "key": "discountSize",
    "value": "10",
    "disabled": true
},
{
    "key": "discountSizeFrom",
    "value": "9",
    "disabled": true
},
{
    "key": "discountSizeTo",
    "value": "11",
    "disabled": true
},
{
    "key": "includePublisher",
    "value": "true",
    "disabled": true
},
{
    "key": "publisherId",
    "value": "1",
    "disabled": true
},
{
    "key": "includeDevelopers",
    "value": "true",
    "disabled": true
},
{
    "key": "developersIds",
    "value": "1",
    "disabled": true
},
{

```

```

        "key": "developersIds",
        "value": "3",
        "disabled": true
    },
    {
        "key": "includeGenres",
        "value": "true",
        "disabled": true
    },
    {
        "key": "genresIds",
        "value": "2",
        "disabled": true
    },
    {
        "key": "includeWishlisted",
        "value": "true",
        "disabled": true
    },
    {
        "key": "includeRated",
        "value": "true",
        "disabled": true
    },
    {
        "key": "hidden",
        "value": "true"
    },
    {
        "key": "sortBy",
        "value": "name",
        "disabled": true
    },
    {
        "key": "desc",
        "value": "true",
        "disabled": true
    },
    {
        "key": "limit",
        "value": "1",
        "disabled": true
    },
    {
        "key": "page",
        "value": "2",
        "disabled": true
    }
    ]
}
},
"response": []
},
{
    "name": "Read Item",
    "request": {
        "method": "GET",
        "header": [],
        "url": {
            "raw":
"http://localhost:5000/api/shop/items/2?includePublisher=true&includeGenres=true&inclu
deDevelopers=true&includeRated=true&includeInCart=true&includeWishlisted=true",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "shop",
                "items",

```

```

        "2"
    ],
    "query": [
        {
            "key": "includeHidden",
            "value": "true",
            "disabled": true
        },
        {
            "key": "includePublisher",
            "value": "true"
        },
        {
            "key": "includeGenres",
            "value": "true"
        },
        {
            "key": "includeDevelopers",
            "value": "true"
        },
        {
            "key": "includeRated",
            "value": "true"
        },
        {
            "key": "includeInCart",
            "value": "true"
        },
        {
            "key": "includeWishlisted",
            "value": "true"
        }
    ]
},
"response": []
},
{
    "name": "Update Item",
    "request": {
        "method": "PATCH",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJBZGlubiIsImVtYWlsIjoIYWRTaW5AbWFpbC5jb20iLCJyb2xlIjoIQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW5kaW5hcnkuY29tL2RucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODZyMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NTgzODkzLCJleHAiOiJlE2ODY2NzAyOTN9.UMJEC6d6snSjFyHoguHS2rK7wN3hZRml1v9UAvSDMbs",
                "type": "text"
            }
        ],
        "body": {
            "mode": "raw",
            "raw": "{\r\n        \"name\": \"Test Item\", \r\n        \"description\": \"Play me!\", \r\n        \"releaseDate\": \"2000-1-4\", \r\n        \"price\": 250, \r\n        \"amount\": 100, \r\n        \"discount\": true, \r\n        \"discountFrom\": \"2000-1-4\", \r\n        \"discountTo\": \"2000-1-14\", \r\n        \"discountSize\": 10, \r\n        \"mainImage\": \"image.png\", \r\n        \"images\": [\r\n            \"image_1\", \r\n            \"image_2\" \r\n        ], \r\n        \"coverImage\": \"image.png\", \r\n        \"characteristics\": {\r\n            \"data1\": false, \r\n            \"data2\": 11, \r\n            \"data3\": 3 \r\n        }, \r\n        \"hide\": false, \r\n        \"publisherId\": 1, \r\n        \"developersIds\": [\r\n            1, \r\n            2 \r\n        ], \r\n        \"genresIds\": [\r\n            1 \r\n        ] \r\n    }",
            "options": {
                "raw": {
                    "language": "json"
                }
            }
        }
    },
    "url": {

```

```

"raw":
"http://localhost:5000/api/shop/items/1",
"protocol": "http",
"host": [
    "localhost"
],
"port": "5000",
"path": [
    "api",
    "shop",
    "items",
    "1"
]
},
"response": []
},
{
    "name": "Delete Item",
    "request": {
        "method": "DELETE",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiYWRTa
W5AbWFpbC5jb20iLCJyb2x1IjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5bW5kaW5hcnkuY29tL2R
ucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc
3FrYWYuanBnIiwiaWF0IjoxNjg2NDY1ODI4LCJleHAiOiJlODY1NTIyMjh9.I0NsX5dq5gqs8IjK5yVl2vPShF
mgeUNUXPXg7_FgbTg",
                "type": "text"
            }
        ],
        "url": {
            "raw":
"http://localhost:5000/api/shop/items/8",
"protocol": "http",
"host": [
    "localhost"
],
"port": "5000",
"path": [
    "api",
    "shop",
    "items",
    "8"
]
},
"response": []
},
{
    "name": "Add Genre(s)",
    "request": {
        "method": "POST",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiYWRTa
W5AbWFpbC5jb20iLCJyb2x1IjoiQURNSU4iLCJpYXQiOiJlODY1NTIyMjh9.I0NsX5dq5gqs8IjK5yVl2vPShF
MsZUfzrvIa9Jcd_tRvvTIIxaot6b-3rxdX6Dk4Ek",
                "type": "text"
            }
        ],
        "body": {
            "mode": "raw",
            "raw": "4,5\r\n    \r\n\"\\\"genresIds\\\": [\r\n
4,5\r\n    ]\r\n\"",
            "options": {
                "raw": {

```

```

        "language": "json"
    }
    },
    "url": {
        "raw":
"http://localhost:5000/api/shop/items/11/genres",
        "protocol": "http",
        "host": [
            "localhost"
        ],
        "port": "5000",
        "path": [
            "api",
            "shop",
            "items",
            "11",
            "genres"
        ]
    }
},
"response": []
},
{
    "name": "Delete Genre(s)",
    "request": {
        "method": "DELETE",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiYWRTa
W5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpYXQiOiJlODQ3MTAxMDQsImV4cCI6MTY4MDE5NjUwNH0.J3-
MsZUfzrvIa9Jcd_tRvvTIIxaot6b-3rxdX6Dk4Ek",
                "type": "text"
            }
        ],
        "body": {
            "mode": "raw",
            "raw": "{\r\n    \"genresIds\": [\r\n
1,\r\n        2\r\n    ]\r\n}",
            "options": {
                "raw": {
                    "language": "json"
                }
            }
        },
        "url": {
            "raw":
"http://localhost:5000/api/shop/items/11/genres",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "shop",
                "items",
                "11",
                "genres"
            ]
        }
    },
    "response": []
},
{
    "name": "Add Developer(s)",
    "request": {
        "method": "POST",
        "header": [

```



```

        "url": {
            "raw":
"http://localhost:5000/api/shop/items/2/rate",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "shop",
                "items",
                "2",
                "rate"
            ]
        },
        "response": []
    },
    {
        "name": "Toggle Cart",
        "request": {
            "method": "POST",
            "header": [
                {
                    "key": "Authorization",
                    "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiYWRTa
W5AbWFpbC5jb20iLCJyb2xlIjoibG9naW4iLCJpbnVnZSI6Imh0dHBzOi8vcmlnaW5kaW5hcnkuY29tL2R
ucWxneXBqaS9pbWFnZS91cGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc
3FrYWYuanBnIiwiaWF0IjoxNjg2NTg4NDgzLCJleHAiOjE2ODY2NzQ4ODN9.yZhVwYYQ-
waMVxWtctmX4JJyamRDlrlvX4Ec3UFWxdw",
                    "type": "text"
                }
            ],
            "url": {
                "raw":
"http://localhost:5000/api/shop/items/1/toggleCart",
                "protocol": "http",
                "host": [
                    "localhost"
                ],
                "port": "5000",
                "path": [
                    "api",
                    "shop",
                    "items",
                    "1",
                    "toggleCart"
                ]
            }
        },
        "response": []
    },
    {
        "name": "Toggle Wishlist",
        "request": {
            "method": "POST",
            "header": [
                {
                    "key": "Authorization",
                    "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiYWRTa
W5AbWFpbC5jb20iLCJyb2xlIjoibG9naW4iLCJpbnVnZSI6Imh0dHBzOi8vcmlnaW5kaW5hcnkuY29tL2R
ucWxneXBqaS9pbWFnZS91cGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc
3FrYWYuanBnIiwiaWF0IjoxNjg2NDk5MDE3LCJleHAiOjE2ODY1ODU0MTd9.n1T-
soiglynr72RFJ5Xqr7vn5MoGBTksXdg1pUyb4wY",
                    "type": "text"
                }
            ],
            "url": {

```

```

        "raw":
"http://localhost:5000/api/shop/items/2/toggleWishlist",
        "protocol": "http",
        "host": [
            "localhost"
        ],
        "port": "5000",
        "path": [
            "api",
            "shop",
            "items",
            "2",
            "toggleWishlist"
        ]
    },
    "response": []
},
{
    "name": "companies",
    "item": [
        {
            "name": "Create Company",
            "request": {
                "method": "POST",
                "header": [
                    {
                        "key": "Authorization",
                        "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlpciIsImVtYWlsIjoiYWRTa
W5AbWFpbC5jb20iLCJyb2xlIjoiaURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbmVkaW5hcnkuY29tL2R
ucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc
3FrYWYuanBnIiwiaWF0IjoxNjg2NTg3NDg5LCJleHAiOiJlODY2NzY0ODl9.Maq5HIgFd9p8UD_VOYFFV2X_J7
OtFN5vE2USvWr6hTI",
                        "type": "text"
                    }
                ],
                "body": {
                    "mode": "raw",
                    "raw": "{\r\n        \"name\": \"ABC
Games\", \r\n        \"description\": \"We make alphabet games!\", \r\n        \"director\":
\"John Connor\", \r\n        \"image\": \"image.png\", \r\n        \"founded\": \"2000-1-1\", \r\n
        \"hide\": false\r\n}",
                    "options": {
                        "raw": {
                            "language": "json"
                        }
                    }
                },
                "url": {
                    "raw":
"http://localhost:5000/api/shop/companies",
                    "protocol": "http",
                    "host": [
                        "localhost"
                    ],
                    "port": "5000",
                    "path": [
                        "api",
                        "shop",
                        "companies"
                    ]
                }
            },
            "response": []
        },
        {
            "name": "Read Companies",
            "request": {

```

```

        "method": "GET",
        "header": [],
        "url": {
            "raw":
"http://localhost:5000/api/shop/companies?includeHidden=true",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "shop",
                "companies"
            ],
            "query": [
                {
                    "key": "limit",
                    "value": "2",
                    "disabled": true
                },
                {
                    "key": "page",
                    "value": "2",
                    "disabled": true
                },
                {
                    "key": "sortBy",
                    "value": "id",
                    "disabled": true
                },
                {
                    "key": "desc",
                    "value": "true",
                    "disabled": true
                },
                {
                    "key": "includeHidden",
                    "value": "true"
                },
                {
                    "key": "name",
                    "value": "ABD",
                    "disabled": true
                },
                {
                    "key": "description",
                    "value": "You",
                    "disabled": true
                },
                {
                    "key": "director",
                    "value": "Wayne",
                    "disabled": true
                },
                {
                    "key": "founded",
                    "value": "2000-1-1",
                    "disabled": true
                },
                {
                    "key": "foundedFrom",
                    "value": "1999-12-30",
                    "disabled": true
                },
                {
                    "key": "foundedTo",
                    "value": "2000-1-2",
                    "disabled": true
                }
            ]
        }
    ]

```

```

    },
    "response": []
  },
  {
    "name": "Read Company",
    "request": {
      "method": "GET",
      "header": [],
      "url": {
        "raw":
"http://localhost:5000/api/shop/companies/5?includeHidden=true&includeDeveloped=true&includePublished=true",
        "protocol": "http",
        "host": [
          "localhost"
        ],
        "port": "5000",
        "path": [
          "api",
          "shop",
          "companies",
          "5"
        ],
        "query": [
          {
            "key": "includeHidden",
            "value": "true"
          },
          {
            "key": "includeDeveloped",
            "value": "true"
          },
          {
            "key": "includePublished",
            "value": "true"
          }
        ]
      }
    },
    "response": []
  },
  {
    "name": "Update Company",
    "request": {
      "method": "PATCH",
      "header": [
        {
          "key": "Authorization",
          "value":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiYWRTaW5AbWFpbiC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW5kaW5hcnkuY29tL2RucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NTg3NDg5LCJleHAiOjE2ODY2NzM4ODl9.Maq5HIgFd9p8UD_VOYFFV2X_J7OtFN5vE2USvWr6hTI",
          "type": "text"
        }
      ],
      "body": {
        "mode": "raw",
        "raw": "{\r\n  \"hide\": false\r\n}",
        "options": {
          "raw": {
            "language": "json"
          }
        }
      },
      "url": {
        "raw":
"http://localhost:5000/api/shop/companies/4",
        "protocol": "http",

```

```

        "host": [
            "localhost"
        ],
        "port": "5000",
        "path": [
            "api",
            "shop",
            "companies",
            "4"
        ]
    },
    "response": []
},
{
    "name": "Delete Company",
    "request": {
        "method": "DELETE",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiiYWRtaW5AbWFpbC5jb20iLCJyb2xlIjoiiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5bW5hcnkuY29tL2RucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NDY1ODI4LCJleHAiOiJlODY1NTIyMjh9.I0NsX5dq5gqs8IjK5yVl2vPShFmgeUNUXPXg7_FgbTg",
                "type": "text"
            }
        ],
        "url": {
            "raw": "http://localhost:5000/api/shop/companies/6",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "shop",
                "companies",
                "6"
            ]
        }
    },
    "response": []
}
],
{
    "name": "news",
    "item": [
        {
            "name": "Create",
            "request": {
                "method": "POST",
                "header": [
                    {
                        "key": "Authorization",
                        "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiiYWRtaW5AbWFpbC5jb20iLCJyb2xlIjoiiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5bW5hcnkuY29tL2RucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NDk5MDE3LCJleHAiOiJlODY1ODU0MTd9.n1T-soiglynr72RFJ5Xqr7vn5MoGBTkSXdg1pUyb4wY",
                        "type": "text"
                    }
                ],
                "body": {
                    "mode": "raw",

```



```

        "disabled": true
    },
    {
        "key": "page",
        "value": "3",
        "disabled": true
    }
]
},
"response": []
},
{
    "name": "Get One",
    "request": {
        "method": "GET",
        "header": [],
        "url": {
            "raw":
"http://localhost:5000/api/news/2",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "news",
                "2"
            ]
        }
    },
    "response": []
},
{
    "name": "Update",
    "request": {
        "method": "PATCH",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiYm9naW4iOiJlc2VyIiwiaWZlhaWwiOiJlc2VyQ
GlhaWwY29tIiwicm9sZSI6IlVTRVliLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5bW5kaW5hcnkuY29tL2RucWx
neXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrY
WYuanBnIiwiaWF0IjoxNjg2NDc0MjA2LCJleHAiOiJlY2ODY1NjA2MDZ9.96RTrUelu_Dw4haUJS1QMHSsP9mukr
p72zI17kEf5Co",
                "type": "text"
            }
        ],
        "body": {
            "mode": "raw",
            "raw": "{\r\n
\"Kekek\", \r\n
\"content\": \"Kekek\", \r\n
\"tags\": [\r\n
\"Fantastic\", \r\n
\"Detective\" \r\n
], \r\n
\"hide\": false \r\n }",
            "options": {
                "raw": {
                    "language": "json"
                }
            }
        },
        "url": {
            "raw":
"http://localhost:5000/api/news/5",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",

```

```

        "news",
        "5"
    ]
},
"response": []
},
{
    "name": "Delete",
    "request": {
        "method": "DELETE",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJBZGlubiIsImVtYWlsIjoiiYWRTaW5AbWFpbC5jb20iLCJyb2xlIjoiiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5b3VkaW5hcnkuY29tL2RucWxneXBqaS9pbWFnZS9lcGxvYXQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NDk5MDE3LCJleHAiOiJlODY1ODU0MTd9.n1T-soiglynr72RFJ5Xqr7vn5MoGBTkSXdg1pUyb4wY",
                "type": "text"
            }
        ],
        "url": {
            "raw": "http://localhost:5000/api/news/6",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "news",
                "6"
            ]
        }
    },
    "response": []
},
{
    "name": "Read Comments",
    "request": {
        "method": "GET",
        "header": [],
        "url": {
            "raw": "http://localhost:5000/api/news/comments/5",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "news",
                "comments",
                "5"
            ]
        }
    },
    "response": []
},
{
    "name": "Set Comment",
    "request": {
        "method": "POST",
        "header": [
            {
                "key": "Authorization",

```

```

        "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwibG9naW4iOiJlc2VyIiwiaWwiOiJlc2VyQ
GlhaWwuY29tIiwicm9sZSI6I1VTRVliLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW50dHkiOiJlc2VyQ
neXBqaS9pbWFnZS91cGxvYQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrY
WYuanBnIiwiaWF0IjoxNjg2NDk5NTQ5LCJleHAiOjE2ODY1ODU5NDI5.hltI9NJF-v9VnRzy7uNLT-
23lj9wvpVE7dEh5DegaE",
        "type": "text"
    },
    "body": {
        "mode": "raw",
        "raw": "{\r\n    \"content\": \"Lorem
ipsum\", \r\n    \"rate\": 5\r\n}\"",
        "options": {
            "raw": {
                "language": "json"
            }
        }
    },
    "url": {
        "raw":
"http://localhost:5000/api/news/1/comment",
        "protocol": "http",
        "host": [
            "localhost"
        ],
        "port": "5000",
        "path": [
            "api",
            "news",
            "1",
            "comment"
        ]
    },
    "response": []
},
{
    "name": "Remove Comment",
    "request": {
        "method": "DELETE",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwibG9naW4iOiJlc2VyIiwiaWwiOiJlc2VyQ
GlhaWwuY29tIiwicm9sZSI6I1VTRVliLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW50dHkiOiJlc2VyQ
neXBqaS9pbWFnZS91cGxvYQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrY
WYuanBnIiwiaWF0IjoxNjg2NDk5NTQ5LCJleHAiOjE2ODY1ODU5NDI5.hltI9NJF-v9VnRzy7uNLT-
p72zI17kEf5Co",
                "type": "text"
            }
        ],
        "url": {
            "raw":
"http://localhost:5000/api/news/1/comment",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "news",
                "1",
                "comment"
            ]
        }
    },
    "response": []
}
}

```

```

    ],
    {
      "name": "genres",
      "item": [
        {
          "name": "Create Genre",
          "request": {
            "method": "POST",
            "header": [
              {
                "key": "Authorization",
                "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiYWRTaW5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW5kaW5hcnkuY29tL2RucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NTg3NDg5LCJleHAiOjE2ODY2NzQ0ODl9.Maq5HIgFd9p8UD_VOYFFV2X_J7OtFN5vE2USvWr6hTI",
                "type": "text"
              }
            ],
            "body": {
              "mode": "raw",
              "raw": "{\r\n  \"name\": \"Sci-fi 3\", \r\n  \"description\": \"Pew-pew!\", \r\n  \"hide\": false\r\n}",
              "options": {
                "raw": {
                  "language": "json"
                }
              }
            },
            "url": {
              "raw": "http://localhost:5000/api/shop/genres/",
              "protocol": "http",
              "host": [
                "localhost"
              ],
              "port": "5000",
              "path": [
                "api",
                "shop",
                "genres",
                ""
              ]
            }
          },
          "response": []
        },
        {
          "name": "Read Genres",
          "protocolProfileBehavior": {
            "disableBodyPruning": true
          },
          "request": {
            "method": "GET",
            "header": [],
            "body": {
              "mode": "raw",
              "raw": "",
              "options": {
                "raw": {
                  "language": "json"
                }
              }
            },
            "url": {
              "raw": "http://localhost:5000/api/shop/genres?hidden=true",
              "protocol": "http",
              "host": [

```

```

        "localhost"
    ],
    "port": "5000",
    "path": [
        "api",
        "shop",
        "genres"
    ],
    "query": [
        {
            "key": "name",
            "value": "det",
            "disabled": true
        },
        {
            "key": "description",
            "value": "w",
            "disabled": true
        },
        {
            "key": "hidden",
            "value": "true"
        },
        {
            "key": "limit",
            "value": "3",
            "disabled": true
        },
        {
            "key": "page",
            "value": "1",
            "disabled": true
        },
        {
            "key": "sortBy",
            "value": "id",
            "disabled": true
        },
        {
            "key": "desc",
            "value": "true",
            "disabled": true
        }
    ]
},
    "response": []
},
{
    "name": "Read Genre",
    "protocolProfileBehavior": {
        "disableBodyPruning": true
    },
    "request": {
        "method": "GET",
        "header": [],
        "body": {
            "mode": "raw",
            "raw": "",
            "options": {
                "raw": {
                    "language": "json"
                }
            }
        },
        "url": {
            "raw":
"http://localhost:5000/api/shop/genres/6?includeHidden=true&includeItems=true",
            "protocol": "http",
            "host": [
                "localhost"
            ]
        }
    }
}

```

```

    ],
    "port": "5000",
    "path": [
        "api",
        "shop",
        "genres",
        "6"
    ],
    "query": [
        {
            "key": "includeHidden",
            "value": "true"
        },
        {
            "key": "includeItems",
            "value": "true"
        }
    ]
    },
    "response": []
},
{
    "name": "Update Genre",
    "request": {
        "method": "PATCH",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlpciIsImVtYWlsIjoiYWRTaW5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW50LmNsY3VkaW5hcnkuY29tL2RucWxneXBqaS9pbWFnZS91cGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXN1cnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NDY1ODI4LCJleHAiOiJlODY1NTIyMjh9.I0NsX5dq5gqs8IjK5yVl2vPShFmgeUNUXPXg7_FgbTg",
                "type": "text"
            }
        ],
        "body": {
            "mode": "raw",
            "raw": "{\r\n    \"name\": \"DC Fan Games\", \r\n    \"description\": \"We have DC!\" \r\n}",
            "options": {
                "raw": {
                    "language": "json"
                }
            }
        },
        "url": {
            "raw": "http://localhost:5000/api/shop/genres/6",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "shop",
                "genres",
                "6"
            ]
        }
    },
    "response": []
},
{
    "name": "Delete Genre",
    "request": {
        "method": "DELETE",
        "header": [

```

```

        {
            "key": "Authorization",
            "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiYWRTaW5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW5kaW5hcnkuY29tL2RucWxneXBqaS9pbWFnZS91cGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NDY1ODI4LCJleHAiOjE2ODY1NTIyMjh9.I0NsX5dq5gqs8IjK5yVl2vPShFmgeUNUXPXg7_FgbTg",
            "type": "text"
        },
        {
            "url": {
                "raw": "http://localhost:5000/api/shop/genres/7",
                "protocol": "http",
                "host": [
                    "localhost"
                ],
                "port": "5000",
                "path": [
                    "api",
                    "shop",
                    "genres",
                    "7"
                ]
            },
            "response": []
        }
    ],
    {
        "name": "users",
        "item": [
            {
                "name": "Register",
                "request": {
                    "method": "POST",
                    "header": [],
                    "body": {
                        "mode": "raw",
                        "raw": "{\r\n    \"email\": \"user2@mail.com\", \r\n    \"login\": \"user2\", \r\n    \"firstName\": \"Jane\", \r\n    \"lastName\": \"Doe\", \r\n    \"password\": \"Qwerty123456\", \r\n    \"image\": \"https://res.cloudinary.com/dnqlgypji/image/upload/v1686310061/gigashop/users/r7mocr34rjlgfissqkaf.jpg\" \r\n}",
                        "options": {
                            "raw": {
                                "language": "json"
                            }
                        }
                    },
                    "url": {
                        "raw": "http://localhost:5000/api/user/register",
                        "protocol": "http",
                        "host": [
                            "localhost"
                        ],
                        "port": "5000",
                        "path": [
                            "api",
                            "user",
                            "register"
                        ]
                    },
                    "response": []
                },
                {
                    "name": "Register Moderator",

```

```

        "request": {
            "method": "POST",
            "header": [
                {
                    "key": "Authorization",
                    "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZG1pbiIsImVtYWlsIjoiYWRTaW5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpYXQiOiJE2ODAwOTQ0MDcsImV4cCI6MTY4MDE4MDgwN30.DqC4Jq6-CmBJUKpbyMzoiCsuAgsevGRXfAKpwBbYC5o",
                    "type": "text"
                }
            ],
            "body": {
                "mode": "formdata",
                "formdata": [
                    {
                        "key": "email",
                        "value": "moder3@mail.com",
                        "type": "text"
                    },
                    {
                        "key": "login",
                        "value": "фывфывфыв",
                        "type": "text"
                    },
                    {
                        "key": "firstName",
                        "value": "John",
                        "type": "text"
                    },
                    {
                        "key": "lastName",
                        "value": "Smith",
                        "type": "text"
                    },
                    {
                        "key": "password",
                        "value": "Qwerty12345",
                        "type": "text"
                    },
                    {
                        "key": "image",
                        "type": "file",
                        "src":
"/D:/N/dnd/Numbers/N_7.png"
                    },
                    {
                        "key": "role",
                        "value": "moderator",
                        "type": "text",
                        "disabled": true
                    }
                ]
            },
            "url": {
                "raw":
"http://localhost:5000/api/user/createModerator",
                "protocol": "http",
                "host": [
                    "localhost"
                ],
                "port": "5000",
                "path": [
                    "api",
                    "user",
                    "createModerator"
                ]
            }
        },
        "response": []
    },

```



```

        "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwibG9naW4iOiJlc2VyIiwiaWwiOiJlc2VyQ
GlhaWwuY29tIiwicm9sZSI6IiVTRVliLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW5kaW5hcnkuY29tL2RucWx
neXBqaS9pbWFnZS91cGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyamxnZmlzc3FrY
WYuanBnIiwiaWF0IjoxNjg2NDc0MjA2LCJleHAiOjE2ODY1NjA2MDZ9.96RTrUelu_Dw4haUJS1QMHSsP9mukr
p72zI17kEf5Co",
        "type": "text"
    },
    "body": {
        "mode": "raw",
        "raw": "{\r\n
        \"user1@mail.com\", \r\n
        \"login\": \"user1\", \r\n
        \"firstName\": \"Jane\", \r\n
        \"lastName\": \"Doe\", \r\n
        \"password\": \"Qwerty123456\", \r\n
        \"image\": \"https://res.cloudinary.com/dnqlgypji/image/upload/v1686310061/gigashop/users/r7mocr3
4rjlgfissqkaf.jpg\" \r\n}",
        "options": {
            "raw": {
                "language": "json"
            }
        },
        "url": {
            "raw":
"http://localhost:5000/api/user/update",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "user",
                "update"
            ]
        }
    },
    "response": [
        {
            "name": "Profile",
            "request": {
                "method": "GET",
                "header": [
                    {
                        "key": "Authorization",
                        "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwibG9naW4iOiJhZGlpb2VlImVtYWlsIjoiYWRta
W5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW5kaW5hcnkuY29tL2R
ucWxneXBqaS9pbWFnZS91cGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyamxnZmlzc
3FrYWYuanBnIiwiaWF0IjoxNjg2NTg5NjI4LCJleHAiOjE2ODY2NzYwMjh9.121fUyyDdYPkS66QKmpjbbmSuC
1u4_LWbwaKyWBLCPg",
                        "type": "text"
                    }
                ],
                "url": {
                    "raw":
"http://localhost:5000/api/user/profile?includeBought=true&includeCart=true&includeRat
es=true&includePublications=true&includePublicationComments=true&includeWishlist=true",
                    "protocol": "http",
                    "host": [
                        "localhost"
                    ],
                    "port": "5000",
                    "path": [
                        "api",
                        "user",
                        "profile"
                    ],
                    "query": [

```

```

        "key": "includeBought",
        "value": "true"
    },
    {
        "key": "includeCart",
        "value": "true"
    },
    {
        "key": "includeRates",
        "value": "true"
    },
    {
        "key":
            "includePublications",
        "value": "true"
    },
    {
        "key":
            "includePublicationComments",
        "value": "true"
    },
    {
        "key": "includeWishlist",
        "value": "true"
    }
]
    },
    "response": []
},
{
    "name": "Get All Users",
    "request": {
        "method": "GET",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlpbGlzImVtYWlsIjoiYWRTaW5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5bW5hcnkuY29tL2RucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NTg4NDgzLCJleHAiOjE2ODY2NzQ4ODN9.yZhVwYYQ-waMVxWtctmX4JJyamRDlrlvX4Ec3UFWxdw",
                "description": "As worker",
                "type": "text"
            }
        ],
        "url": {
            "raw":
                "http://localhost:5000/api/user/all",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "user",
                "all"
            ]
        }
    },
    "response": []
},
{
    "name": "Set Up Cart",
    "request": {
        "method": "POST",
        "header": [
            {
                "key": "Authorization",

```

```

        "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlpciIsImVtYWlsIjoiYWRTa
W5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5b29tL2RucWx
ucWxneXBqaS9pbWFnZS91cGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRy
amxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NTg4NDgzLCJleHAiOjE2ODY2NzQ0ODN9.yZhVwYYQ-
waMVxWtctmX4JJyamRDlrlvX4Ec3UFWxdw",
        "type": "text"
    },
    {
        "url": {
            "raw":
"http://localhost:5000/api/user/cart/setup",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "user",
                "cart",
                "setup"
            ]
        },
        "response": []
    },
    {
        "name": "Buy Cart",
        "request": {
            "method": "POST",
            "header": [
                {
                    "key": "Authorization",
                    "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiwibG9naW4iOiJlc2VyTiwiZWlhaWwiOiJlc2VyQ
GlhaWwuY29tIiwicm9sZSI6I1VTRVliLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5b29tL2RucWx
neXBqaS9pbWFnZS91cGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRy
amxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NTgzNDI1LCJleHAiOjE2ODY2Njk4MjV9.19qfNQQQoOZzES-
84SgCaufQi9LwnAjWpP76IyUHVmA",
                    "type": "text"
                }
            ],
            "url": {
                "raw":
"http://localhost:5000/api/user/cart/success/f68269fe-5dac-4127-a234-e20a894a2292",
                "protocol": "http",
                "host": [
                    "localhost"
                ],
                "port": "5000",
                "path": [
                    "api",
                    "user",
                    "cart",
                    "success",
                    "f68269fe-5dac-4127-a234-
e20a894a2292"
                ]
            }
        },
        "response": []
    },
    {
        "name": "Toggle Moderator",
        "request": {
            "method": "POST",
            "header": [
                {
                    "key": "Authorization",

```



```

{
  "name": "Clear Cart",
  "request": {
    "auth": {
      "type": "bearer"
    },
    "method": "DELETE",
    "header": [
      {
        "key": "Authorization",
        "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiYWRTaW5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5b29tL2RucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NTg5NjI4LCJleHAiOiJlE2ODY2NzYwMjh9.121fUyyDdYPkS66QKmpjbbmSuC1u4_LWbwaKyWBLCPg",
        "type": "text"
      }
    ],
    "url": {
      "raw": "http://localhost:5000/api/user/cart",
      "protocol": "http",
      "host": [
        "localhost"
      ],
      "port": "5000",
      "path": [
        "api",
        "user",
        "cart"
      ]
    }
  },
  "response": []
}
],
{
  "name": "comments & moderation",
  "item": [
    {
      "name": "Read Items Rates",
      "protocolProfileBehavior": {
        "disableBodyPruning": true
      },
      "request": {
        "method": "GET",
        "header": [
          {
            "key": "Authorization",
            "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJhZGlubiIsImVtYWlsIjoiYWRTaW5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5b29tL2RucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NDk5MDE3LCJleHAiOiJlE2ODY1ODU0MTd9.n1T-soiglynr72RFJ5Xqr7vn5MoGBTkSXdg1pUyb4wY",
            "type": "text",
            "disabled": true
          }
        ],
        "body": {
          "mode": "raw",
          "raw": "",
          "options": {
            "raw": {
              "language": "json"
            }
          }
        }
      },
      "url": {

```

```

"raw":
"protocol": "http",
"host": [
  "localhost"
],
"port": "5000",
"path": [
  "api",
  "comments",
  "items"
],
"query": [
  {
    "key": "sortBy",
    "value": "createdAt",
    "disabled": true
  },
  {
    "key": "desc",
    "value": "true",
    "disabled": true
  },
  {
    "key": "page",
    "value": "1",
    "disabled": true
  },
  {
    "key": "limit",
    "value": "10",
    "disabled": true
  }
]
},
"response": []
},
{
  "name": "Read Publications Comments",
  "request": {
    "method": "GET",
    "header": [
      {
        "key": "Authorization",
        "value": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJBZGlpbiIsImVtYWlsIjoiYWRTaW5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5bWVudjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc3FrYWYuanBnIiwiaWF0IjoxNjg2NDk5MDE3LCJleHAiOjE2ODY1ODU0MTd9.n1T-soiglynr72RFJ5Xqr7vn5MoGBTkSXdg1pUyb4wY",
        "type": "text",
        "disabled": true
      }
    ],
    "url": {
      "raw":
"http://localhost:5000/api/comments/news?sortBy=id&desc=false&limit=12&page=1",
      "protocol": "http",
      "host": [
        "localhost"
      ],
      "port": "5000",
      "path": [
        "api",
        "comments",
        "news"
      ],
      "query": [
        {
          "key": "sortBy",

```

```

        "value": "id"
    },
    {
        "key": "desc",
        "value": "false"
    },
    {
        "key": "limit",
        "value": "12"
    },
    {
        "key": "page",
        "value": "1"
    }
]
    },
    "response": []
},
{
    "name": "Set Publication Violation",
    "request": {
        "method": "PATCH",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJBZGlpciIsImVtYWlsIjoiYWRTa
W5AbWFpbC5jb20iLCJyb2xlIjoIQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW50LmNsb3VkaW5hcnkuY29tL2R
ucWxneXBqaS9pbWFnZS91cGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyamxnZmlzc
3FrYWYuanBnIiwiaWF0IjoxNjg2NDk5MDE3LCJleHAiOjE2ODY1ODU0MTd9.nlT-
soiglynr72RFJ5Xqr7vn5MoGBTkSXdglpUyb4wY",
                "type": "text"
            }
        ],
        "body": {
            "mode": "raw",
            "raw": "{\r\n                \"violation\":
false,\r\n                \"violation_reason\": \"Lorem ipsum\"\r\n            }",
            "options": {
                "raw": {
                    "language": "json"
                }
            }
        },
        "url": {
            "raw":
"http://localhost:5000/api/moderation/news/2",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "moderation",
                "news",
                "2"
            ]
        }
    },
    "response": []
},
{
    "name": "Set Comment Violation",
    "request": {
        "method": "PATCH",
        "header": [
            {
                "key": "Authorization",

```

```

        "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJBZGlubiIsImVtYWlsIjoIYWRTa
W5AbWFpbC5jb20iLCJyb2xlIjoIQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW5kaW5hcnkuY29tL2R
ucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyamxnZmlzc
3FrYWYuanBnIiwiaWF0IjoxNjg2NDk5MDE3LCJleHAiOjE2ODY1ODU0MTd9.n1T-
soiglynr72RFJ5Xqr7vn5MoGBTkSXdg1pUyb4wY",
        "type": "text"
    },
    ],
    "body": {
        "mode": "raw",
        "raw": "{\r\n                \"violation\":
false,\r\n    \"violation_reason\": \"Lorem ipsum\"\r\n}",
        "options": {
            "raw": {
                "language": "json"
            }
        }
    },
    "url": {
        "raw":
"http://localhost:5000/api/moderation/comments/2",
        "protocol": "http",
        "host": [
            "localhost"
        ],
        "port": "5000",
        "path": [
            "api",
            "moderation",
            "comments",
            "2"
        ]
    },
    },
    "response": []
},
{
    "name": "Set Rate Violation",
    "request": {
        "method": "PATCH",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJBZGlubiIsImVtYWlsIjoIYWRTa
W5AbWFpbC5jb20iLCJyb2xlIjoIQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlnaW5kaW5hcnkuY29tL2R
ucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyamxnZmlzc
3FrYWYuanBnIiwiaWF0IjoxNjg2NDk5MDE3LCJleHAiOjE2ODY1ODU0MTd9.n1T-
soiglynr72RFJ5Xqr7vn5MoGBTkSXdg1pUyb4wY",
                "type": "text"
            }
        ],
        "body": {
            "mode": "raw",
            "raw": "{\r\n                \"violation\":
true,\r\n    \"violation_reason\": \"Lorem ipsum\"\r\n}",
            "options": {
                "raw": {
                    "language": "json"
                }
            }
        },
        "url": {
            "raw":
"http://localhost:5000/api/moderation/rates/2",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",

```

```

        "path": [
            "api",
            "moderation",
            "rates",
            "2"
        ]
    },
    "response": []
},
{
    "name": "Statistics",
    "request": {
        "method": "GET",
        "header": [
            {
                "key": "Authorization",
                "value": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibG9naW4iOiJBZGlpciIsImVtYWlsIjoiYWRTa
W5AbWFpbC5jb20iLCJyb2xlIjoiQURNSU4iLCJpbWFnZSI6Imh0dHBzOi8vcmlhbnR5b3VkaW5hcnkuY29tL2R
ucWxneXBqaS9pbWFnZS9lcGxvYWQvdjE2ODYzMTAwNjEvZ2lnYXNob3AvdXNlcnMvcjdtb2NyMzRyYamxnZmlzc
3FrYWYuanBnIiwiaWF0IjoxNjg2NDk5MDE3LCJleHAiOiJlODY1ODU0MTd9.n1T-
soiglynr72RFJ5Xqr7vn5MoGBTkSXdg1pUyb4wY",
                "type": "text"
            }
        ],
        "url": {
            "raw":
"http://localhost:5000/api/moderation/statistics",
            "protocol": "http",
            "host": [
                "localhost"
            ],
            "port": "5000",
            "path": [
                "api",
                "moderation",
                "statistics"
            ]
        }
    },
    "response": []
}
]
}
}

```