

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

На тему РОЗРОБКА ВЕБЗАСТОСУНКУ «ІНФОРМАЦІЙНИЙ ПОРТАЛ
ВІДЕОІГОР»
DEVELOPMENT OF THE VIDEO GAME INFORMATION PORTAL WEB
APPLICATION

Виконав: студент(ка) 4 курсу, групи КНТ-
110сп

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ПАПАЗОВ Д.О.

(прізвище та ініціали)

Керівник ЗЕЛЕНЬОВА І.Я.

(прізвище та ініціали)

Рецензент КАЗУРОВА А.Є.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Факультет ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
 Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
 “ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПАПАЗОВА Дмитра Олександровича
 (прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка вебзастосунку «Інформаційний портал відеоігор» Development of the Video Game Information Portal

Керівник проекту (роботи) ЗЕЛЕНЬОВА Ірина Яківна, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «4» квітня 2023 року № 87

2. Строк подання студентом проекту (роботи) 1 червня 2023 року
 3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Керівництво програміста. 5. Керівництво оператора

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЗЕЛЕНЬОВА І.Я., доцент		
Нормоконтролер	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 4 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Розробка програми	4тиждень	Розділ 3
5	Тестування та експериментальне дослідження програми	5 тиждень	Розділ 4
6	Створення керівництва оператора програми	6 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
8	Захист роботи	8 тиждень	

Студент(ка)

Дмитро ПАПАЗОВ
(підпис) (прізвище та ініціали)

Керівник проекту (роботи)

Ірина ЗЕЛЕНЬОВА
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра:
58 с., 20 рис., 2 дод., 12 джерел., 13 табл.

ПОШУК, ВЕБСАЙТ, РЕЄСТРАЦІЯ, КОМП'ЮТЕРНА ГРА, ФІЛЬТР,
REACT, ТЕГ, КОМЕНТАРІ, РЕЙТИНГ, ОБРАНЕ, НОВИНА, БАЗА ДАНИХ.

Об'єкт дослідження – програмні засоби для отримання інформації що до комп'ютерних ігор.

Предмет дослідження – комп'ютерні програми для перегляду комп'ютерних ігор.

Мета роботи – створення програмної системи для перегляду інформації що до комп'ютерних ігор з додатковим функціоналом.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, технологія та пристрій WebStorm, JetBrains Rider, мови програмування Typescript, Html, CSS, персональний комп'ютер з процесором Ryzen 5 2600x під управлінням операційної системи Microsoft Windows 11.

Результати. Створено програмний web site, використовуючи реактивний підхід за рахунок бібліотеки React.

Висновки. Розроблено програмну систему для перегляду інформації що до комп'ютерних ігор з додатковим функціоналом.

Галузь використання – перегляд актуального ринку комп'ютерних ігор.

Економічна ефективність. При використанні програми збільшена швидкість загрузки даних порівняно з іншими програмами, також готові місця для розміщення реклами.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 58 pp., 20 figures, 2 appendices, 12 sources., 13 table.

SEARCH, WEB SITE,REGISTRATION, COMPUTER GAME, FILTER, REACT, TAG, COMMENTS, RATING, FAVORITE, NEWS, DATABASE.

The object of research is software for obtaining information about computer games.

The subject of research is computer programs for watching computer games.

The purpose of the work is to create a software system for viewing information on computer games with additional functionality.

Materials, methods and technical means: structural and object-oriented programming, WebStorm technology and device, JetBrains Rider, Typescript, Html, CSS programming languages, a personal computer with a Ryzen 5 2600x processor running the Microsoft Windows 11 operating system.

Results. A software web site was created, using a reactive approach at the expense of the React library.

Conclusions. A software system has been developed for viewing information on computer games with additional functionality.

The field of use is a review of the current market of computer games.

Economic efficiency. When using the program, the data download speed is increased compared to other programs, and there are also ready places for advertising.

ЗМІСТ

ВСТУП	9
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	10
1.1 Огляд аналогів	10
1.1.1 Аналоги інших сайтів	10
1.1.2 Аналіз переваг та цікавих вирішень аналогів	10
1.2 Аналіз інструментів для розробки.....	11
1.2.1 Підбір стеку технологій.....	11
1.2.2 Технічна підготовка до завдання.....	11
2. АНАЛІЗ ТЕХНОЛОГІЇ РОЗРОБКИ	12
2.1 Angular.....	12
2.2 TypeScript	12
2.3 Asp net core.....	13
2.4 Entity Framework.....	14
2.5 C#.....	14
3. ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ СИСТЕМИ.....	17
3.1 Розробка бази даних.....	15
3.2 Алгоритм функціонування програми.....	21
3.3 Проектування бази даних.....	22
4. ЕКСПЛУАТАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ.....	25
4.1 Характеристики програми.....	25
4.2 Запуск застосування.....	25
4.3 Загальні відомості про програму.....	26
4.4 Структура програми.....	26
4.5 Тестування програми.....	26

5 КЕРІВНИЦТВО ОПЕРАТОРА.....	27
5.1 Призначення програми.....	27
5.2 Умови виконання програми.....	27
5.3 Експлуатація програми.....	27
Висновки.....	33
Перелік джерел посилань.....	34
Додаток А Текст програми	35
Додаток Б Слайди презентації.....	41

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
IDE	– Integrated Development Environment;
TS	– Typescript
UI	– User Interface;
WEB	– World Wide Web;

ВСТУП

Вебсайти з інформацією про комп'ютерні ігри допомагають гравцям отримати доступ до важливої інформації, такої як огляди, оновлення, новини та поради щодо ігор. Це також може допомогти гравцям знайти нові ігри, які їм можуть сподобатися, а також знайти відповіді на питання, пов'язані з іграми. Вебсайти також можуть допомогти спільноті гравців об'єднатися та обговорювати ігри, обмінюватися порадами та досвідом[1].

Вебпортали з новинами про ігрову індустрію є важливим джерелом інформації для гравців, професіоналів, журналістів та всіх, хто цікавиться ігровою індустрією. Основна мета цих порталів - надати своїм читачам актуальну інформацію про нові релізи ігор, огляди, новини про ігрові консолі та платформи, інтерв'ю з розробниками, інформацію про головні події та турніри в галузі геймінгу тощо.

Крім того, вебпортали з новинами про ігрову індустрію можуть бути корисні для гравців, які шукають інформацію про відповідну гру, що допоможе їм зробити правильний вибір при покупці. Також ці портали можуть надати корисні поради та стратегії для гравців, які хочуть покращити свої навички в іграх.

Також, портали з новинами про ігрову індустрію можуть бути корисні для розробників, видавців та інших фахівців, які працюють в галузі геймінгу. Вони можуть допомогти відстежувати тренди та зміни на ринку ігор, надати інформацію про ігрові консолі, платформи, інструменти розробки, рекламні кампанії та інші аспекти галузі.

Отже, вебпортали з новинами про ігрову індустрію є важливим інформаційним ресурсом для всіх, хто цікавиться галуззю геймінгу, і можуть бути корисні як для ігрових фанатів, так і для фахівців в галузі[2].

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд аналогів

Перегляд аналогів дає можливість для розбору переваг та недоліків сайтів різних розробників та різних пристроїв.

1.1.1 Аналоги інших сайтів

В інтернеті дуже велика кількість сайтів, додатків і віджетів з інформацією про ігри, кожне з них чимось виділяється, але більшість із сайтів не зручні та нудні для перегляду. У деяких додатках і віджетах є адаптивність тим, однак там відсутній вибір того, що хоче бачити користувач.

1.1.2 Аналіз переваг та цікавих вирішень аналогів

Найкраще з побачених аналогів – це віджети для телефонів, де присутня здатність змінювати сайту та фонів картинок – що для мене є найбільш суттєвим та виділяє їх серед усіх. Такі сайти або програми красиві та зручні для перегляду.

Також важливо було дізнатися джерела, звідки вони беруть дані. Більшість з них беруть дані з API-сервісів і дуже мало зі своїх досліджень, які володіють та ведуть власні сайти. Багато з API-сервісів є платними і не мають безкоштовних рішень, але один з найпопулярніших Rawg.io має безкоштовний доступ з варіативними даними інформації що до популярних ігор.

1.2 Аналіз інструментів для розробки

Аналіз включає перегляд інструментів сайтів аналогів та їх вирішення у верстці сторінок. Підбір стеку технологій та технічна підготовка до завдання.

1.2.1 Підбір стеку технологій

Написання сайту на чистому html, css, js зараз не актуальний, програма які перезавантажують сторінку втратили актуальність, так як є жахливою практикою для розробника:

- коли додаток почне розширюватися – подібна техніка додасть проблем майбутнім розробкам;
- повторюваний функціонал для схожих блоків у яких відрізняється лише мала частина;
- велика збитковість ресурсів при багатосторінковому додатку.

З цього стає зрозуміло, що стек технологій потребує сучасних вирішень, а саме вибір фреймворку, що підходить для всіх рішень сайту “StopGame”.

1.2.2 Технічна підготовка до завдання

Сайт “ StopGame ” потребує у таких технологіях:

- динамічні зміни даних та їх централізоване зберігання;
- фреймворк який не вимагає перезавантаження сторінок ;
- технологія відправлення та отримання даних через api;
- можливість перевикористання компонентів або шаблонів.

2 АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ

2.1 Angular

Angular - це фреймворк для розробки вебдодатків, що був розроблений компанією Google. Він використовує мову програмування TypeScript та пропонує широкі можливості для створення високопродуктивних, масштабованих та модульних додатків.

Angular надає структуру для розробки клієнтської частини вебдодатків, дозволяє використовувати компоненти, директиви та сервіси, а також має вбудовану підтримку для створення SPA (односторінкових додатків)[3].

Основні переваги Angular:

- структура та організація коду;
- висока продуктивність;
- підтримка SPA;
- масштабованість та модульність;
- велика кількість інструментів та пакетів підтримки.

2.1 TypeScript

TypeScript — мова програмування, розроблена Microsoft у 2012 році і позиціонується як засіб розробки вебдодатків, для розширення можливостей JavaScript.

TypeScript є сумісним з JavaScript і компілюється в останній. Фактично, після компіляції програму на TypeScript можна використовувати у будь-якому сучасному браузері або використовувати разом із серверною платформою Node.js. Код експериментального компілятора, який транслює TypeScript JavaScript, поширюється під ліцензією Apache. Його технологія застосовується у громадському репозиторії через обслуговування GitHub.

TypeScript відрізняється від JavaScript наявністю статичного призначення типів, підтримкою використання повноцінних класів (як у традиційних об'єктно-орієнтованих мовах), а також підтримкою підключення

модулів, що покликане підвищити швидкість розробки, полегшити читання, рефакторинг та повторне використання коду, він допомагає здійснювати пошук помилок на етапі розробки та компіляції і може, прискорити виконання програм.

Планується, що з повної зворотної сумісності адаптація існуючих додатків новий мову програмування може відбуватися поетапно, шляхом поступового визначення типів.

TypeScript виник через певні недоліки JavaScript у великих додатках як компанії Microsoft, так і в інших користувачів JavaScript. Проблеми з розробкою складних програм на JavaScript призвели до полегшення розробки компонентів мови[3],[4].

2.2 ASP.NET.Core

ASP.NET Core - це фреймворк для розробки вебдодатків, який був розроблений компанією Microsoft. Він є відкритим та безкоштовним інструментом, що підтримує кросплатформенність і може використовуватись для створення вебдодатків на різних платформах, таких як Windows, Linux і macOS.

ASP.NET Core пропонує багато інструментів для розробки вебдодатків, таких як вбудований механізм маршрутизації, підтримка середовища Docker та вбудовані шаблони проектів[5].

Основні переваги ASP.NET Core:

- відкритість та безкоштовність;
- підтримка кросплатформеності;
- висока продуктивність;
- інтеграція з Azure та іншими хмарними платформами;
- підтримка середовища Docker.

2.4 Entity Framework

Entity Framework - це об'єктно-орієнтований фреймворк для роботи з базами даних в середовищі .NET. Він дозволяє розробникам працювати з базами даних за допомогою об'єктно-орієнтованого підходу, використовуючи мову C# або VB.NET.

Entity Framework дозволяє розробникам використовувати LINQ (Language-Integrated Query) для отримання даних з бази даних та підтримує такі функції, як міграція даних, кешування та кешування другого рівня[6].

Основні переваги Entity Framework:

- об'єктно-орієнтований підхід до роботи з базами даних;
- використання LINQ для отримання даних;
- підтримка міграції даних;
- підтримка кешування та кешування другого рівня.

2.5 C#

C# - це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона стала однією з основних мов програмування для розробки програмного забезпечення в середовищі .NET.

C# дозволяє розробникам створювати різноманітні програми, від додатків для робочого столу до вебдодатків та мобільних додатків. Вона має багато функцій, таких як вбудовану підтримку многопоточності, генеріки, делегати та лямбда-вирази[7].

Основні переваги C#:

- строга типізація та безпека типів;
- об'єктно-орієнтований підхід до програмування;
- вбудована підтримка многопоточності;
- підтримка LINQ;
- висока продуктивність.

2.6 Вибір засобів розробки

Розробку вебзастосунку заплановано на основі мови TypeScript[4] для фронтенду через її придатність для вирішення широкого кола задач і зокрема створення вебзастосунків, які за рахунок використання вебфреймворку придатні для використання у бізнесах різного рівня, хоча на рівень впливає і вибір фреймворка. Для бекенду було обрано мову C#[7] через її швидкодійність, можливість працювати в декілька потоків, гнучкість при використанні шаблонів проектування, та орієнтацію на високонавантажені застосунки. Тому тепер потрібно порівняти (табл. 2.1) два найпопулярніші фреймворки веброботи для TypeScript: Angular[3], React[8]. Також необхідно порівняти (табл. 2.2) дві найпопулярніші технології мови C#: ASP.Net.Core[9], а саме Web Api та Blazor[10].

Таблиця 2.1 – Вибір вебфреймворку розробки мовою TypeScript

Показник	Angular	React
Високорівневий API	+	—
Висока оптимізація	+	—
Кросплатформеність	—	+
Формування динамічних вебсторінок	+	+
Використовувана мова програмування	TypeScript	TypeScript

Таблиця 2.2 – Вибір технології розробки мовою C#

Показник	ASP.Net.CoreWeb API	ASP.Net.CoreBlazor
Висока оптимізація	+	+
Кросплатформеність	+	+
Використання стороннього фреймворку для фронтенду	+	—
Використовувана мова програмування	C#	C#

Вибір у підсумку фреймворку Angular обґрунтований його основними перевагами, що включають високорівневий API, завдяки чому спрощено початок розробки вебзастосунку через відсутність необхідності самостійного встановлення великої кількості бібліотек. Високий рівень оптимізації також є важливим показником для розробки високо навантаженої системи. Враховуючи вибір цього фреймворку, за основу було взято компонентну архітектуру [3] для фронтенду.

Вибір у підсумку «чистого» ASP.Net.Core обґрунтований в основному можливістю використовувати сторонній фреймворк для розробки фронтенду.

Враховуючи вибір цих технологій, за основу було взято шаблон проектування Model-View-Controller (MVC)[11].

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ СИСТЕМИ

3.1 Розробка бази даних

Розроблювана БД має містити в собі інформацію про користувача, новини, скріншоти, коментарі, тегі, платформи та улюблені ігри користувача. Тобто ми можемо виділити наступні сутності, які будуть зберігатися у БД:

- Games;
- Comments;
- News;
- Screenshots;
- Users;
- UserGames;
- Tags;
- Platforms.

Таблиця Games призначена для зберігання інформації що до ігор (табл. 3.1).

Таблиця 3.1 – Атрибути таблиці Games

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
id	bigint	+	Bigint max	Первинний ключ
Name	text	+	255	
Description	text	+	255	
Rating	Double precision	+	Bigint max	
ImgPath	text	+	255	
TrailerPath	text	+	255	

Таблиця Comments містить у собі список та зміст коментаря (табл. 3.2) з коментарями.

Таблиця 3.2 – Атрибути таблиці Comments

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
Id	bigint	+	Bigint max	Первинний ключ
CommentText	text	+	255	
GameId	bigint	+	Bigint max	Зовнішній ключ
UserId	bigint	+	Bigint max	Зовнішній ключ

Таблиця News містить список новин що до ігор та новин з ігрової індустрії (табл. 3.3).

Таблиця 3.3 – Атрибути таблиці News

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
Id	bigint	+	Bigint max	Первинний ключ
Title	text	+	255	
Description	text	+	255	
ImgPath	text	+	255	
GameId	bigint	+	Bigint max	Зовнішній ключ

У таблиці Screenshots зберігаються фото с іграми для більш детального ознайомлення з грою (табл. 3.4).

Таблиця 3.4 – Атрибути таблиці Screenshots

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
id	bigint	+	Bigint max	Первинний ключ
ScreenShotPath	text	+	255	
GameId	bigint	+	Bigint max	Зовнішній ключ

Таблиця User (табл. 3.5) буде містити в собі головну інформацію про користувача.

Таблиця 3.5 – Атрибути таблиці Users

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
Id	bigint	+	Bigint max	Первинний ключ
Email	text	+	255	
Password	text	+	255	

Інформація про улюблені ігри користувачем буде міститися у таблиці UserGames (табл. 3.6).

Таблиця 3.6 – Атрибути таблиці UserGames

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
Id	bigint	+	Bigint max	Первинний ключ
GameId	bigint	+	Bigint max	Зовнішній ключ
UserRating	integer	+	max	
UserDescription	text	+	255	
UserId	bigint	+	Bigint max	Зовнішній ключ

Таблиця Tags (табл. 3.7) та Platforms (табл. 3.8) буде містити в собі список тегів та платформ які використовуються при пошуку гри та її класифікації,

Таблиця 3.7 – Атрибути таблиці Tags

Атрибут	Тип даних	Обов'язковість	Припустиме значення	Ключовий атрибут
Id	bigint	+	Bigint max	Первинний ключ
Name	text	+	255	

Таблиця 3.8 – Атрибути таблиці Platforms

Атрибут	Тип даних	Обов’язковість	Припустиме значення	Ключовий атрибут
Id	bigint	+	Bigint max	Первинний ключ
Name	text	+	255	

Розроблювана БД була спроектована за допомогою PostGress EntityFramework (Рис. 3.1).

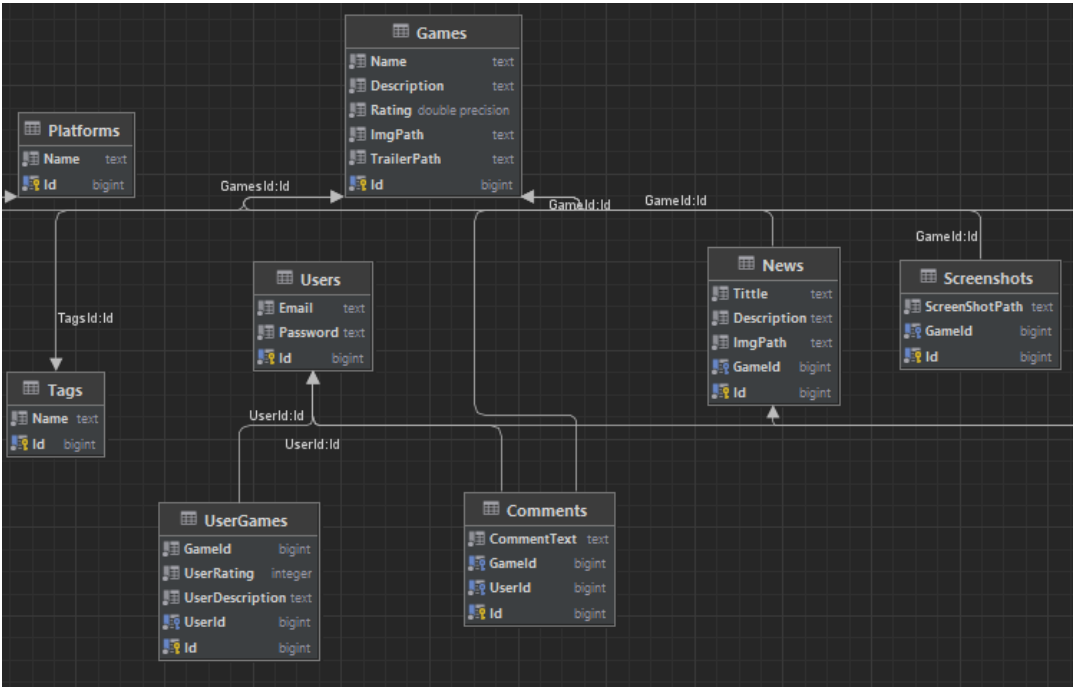


Рисунок 3.1 – Макет створеної БД.

3.2 Алгоритм функціонування програми

Процес роботи користувача виконано таким чином щоб йому було просто та зручно користуватись програмою. Інтерфейс був створений таким чином щоб кожний користувач міг інтуїтивно зрозуміти яка панель за що відповідає. Сайт складається з декількох сторінок, а саме:

- головна сторінка;
- сторінка реєстрації;
- сторінка входу;
- сторінка з грою;
- сторінка з новинами;
- сторінка з більш детальною інформацією що до новини.

Нижче показано схему функціонування програми (рис 3.2):

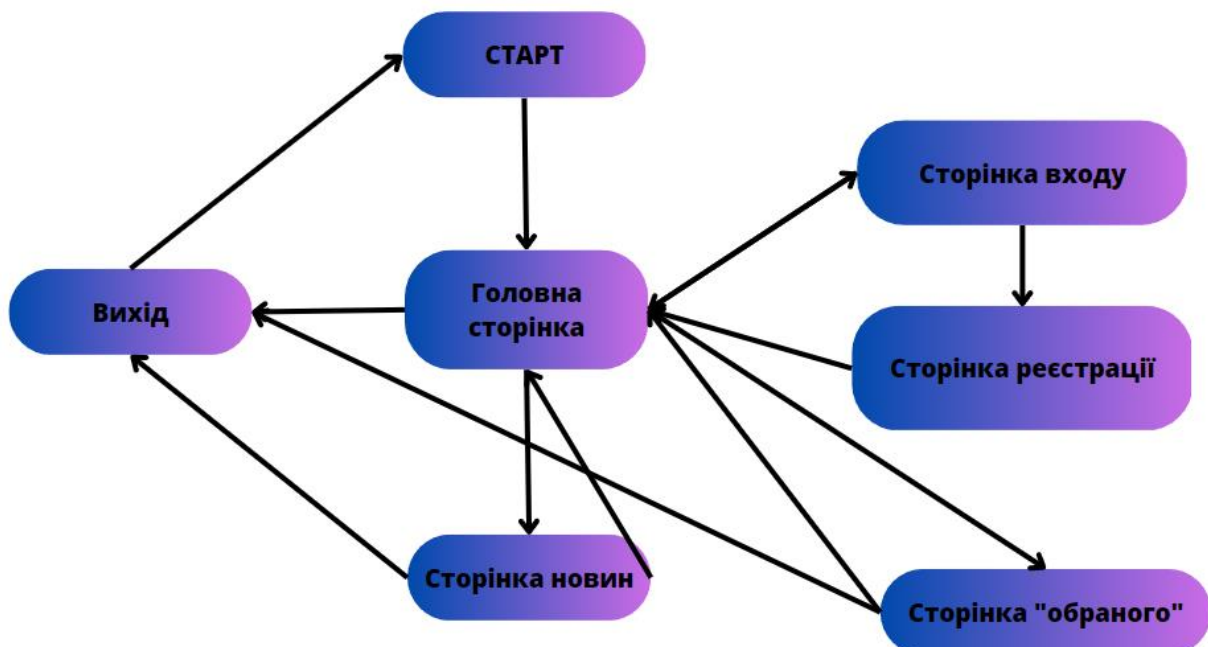


Рисунок 3.2 – Схема функціонування програми

3.3 Проектування бази даних

Проектування БД є важливою частиною проектування застосунку. Для створення БД було обрано EntityFramework[6] (EF) на базі технології .Net. EF має декілька підходів[10] для проектування БД:

- Code-first[12] – вперше з'явився в EntityFramework 4.1, зазвичай використовується, коли у вас вже є існуюча програма, що містить модель даних. Ця модель зазвичай описується за допомогою декількох класів і коду взаємодії між цими класами;

- Model-First – вперше з'явився у версії EntityFramework 4, застосовується розробниками, які не хочуть використовувати інструменти СУБД для створення та управління базами даних, а також не хочуть вручну налаштовувати класи моделі EDM. Фактично це найпростіший підхід під час роботи з EntityFramework. Проектування моделі відбувається у графічному дизайнері EDM середовища VisualStudio;

- Database-First – робочий процес створення моделі починається зі створення та проектування бази даних. Після створення сутнісних класів моделі з існуючої бази даних, робота з EntityFramework аналогічна підходам Code-First і Model-First. Це означає створення об'єкта класу контексту та використання цього об'єкта для виконання необхідних завдань.

Для розробки було обрано підхід Code-first через відсутність необхідності роботи зі сторонніми засобами окрім середовища розробки та кращу сумісність створеної БД з кодом, що було написано через відсутність конфліктів типів даних.

Нижче приведені список полів у БД та що вони виконують. Таблиця Games містить поля:

- Name – назва гри;
- Description – опис гри;
- Rating – рейтинг гри;

- ImgPath – фото гри;
- TrailerPath - трейлер гри;
- Id – ідентифікатор гри.

Таблиця Platforms містить поля:

- Name – назва платформи;
- Id – ідентифікатор платформи.

Таблиця Tags містить поля:

- Name – назва тегу;
- Id – ідентифікатор тегу.

Таблиця Users містить поля:

- Email - пошта користувача;
- Password – пароль користувача;
- Id – ідентифікатор користувача.

Таблиця UserGames містить поля:

- gameId – ідентифікатор гри;
- UserRating – рейтинг гри який виставив користувач;
- UserDescription – опис гри який додав собі користувач;
- UserId – ідентифікатор користувача;
- Id – ідентифікатор «обраного».

Таблиця Comments містить поля:

- CommentText – текст коментаря від користувача;
- gameId – ідентифікатор гри;
- UserId – ідентифікатор користувача;
- Id – ідентифікатор коментаря.

Таблиця News містить поля:

- Title – заголовок новини;
- Description – опис новини;
- ImgPath – фото новини;
- gameId – ідентифікатор гри на яку додана новина;

4 ЕКСПЛУАТАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Характеристики програми

У розробленому вебзастосунку було реалізовано усі потрібні складові для коректної роботи сервісу. Основні серед них є наступні:

- реєстрація та авторизація користувачів;
- можливість додавання коментарів ;
- перегляд інформації о новинах та списку ігор;
- додавання інформації до списку обраного;
- додавання новин;
- пошук інформації.

4.2 Запуск застосування

Запуск програми відбувається на будь-якій операційній системі, що має підтримку бібліотеки пакетів dot-net. Перед запуском необхідно використати файл prelaunch.bat, що автоматично вставить усі необхідні для роботи програми бібліотеки, та налаштує середовище перед запуском.

Назви та версії усіх необхідних бібліотек описані у файлі package.json в обох проектах.

Щоб запустити програму потрібно окремо запустити проекти Assets.WebApi та Frontend. Проект Assets.WebApi запускається за допомогою консольної команди dot-netstart. Запуск проекту Frontend відбувається за допомогою команди ng serve.

4.3 Загальні відомості про програму

Програма призначена для спрощення пошуку інформації що до ігрової індустрії

Додаток було створено за допомогою середовищ розробки JetBrains Rider, JetBrains WebStorm та JetBrains IntelliJ IDEA.

Базу даних було створено за допомогою EntityFramework.

Підключення таблиць бази даних на форму проекту було виконано за допомогою EntityFramework та HTTP.

Підключення до бази даних було організоване за допомогою строки з'єднання.

4.4 Структура програми

Програма складається з двох проектів:

- Assets.WebAPI – проект бекенду на базі Asp.Net Core Web API, що містить в собі основні контролери та функції програми, створення БД, міграції та допоміжні бібліотеки;
- Frontend – проект на базі Angular 14, що містить в собі основні компоненти програми з відображення даних.

Зв'язок між проектами Assets.WebAPI та Frontend відбувається за допомогою HTTP протоколу.

4.5 Тестування програми

Тестування програми дозволило перевірити роботу програми щодо виконання функціональних можливостей, які передбачались у процесі розробки. Тестування спочатку виконувалось під час реалізації цих можливостей і дозволяло виявляти проблеми і вирішувати їх.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Загалом призначенням програми є створення інструменту для пошуку та ознайомлення ресурсів ігрової індустрії з відстеженням новин про неї.

Інформаційний сайт про відеоігри часто публікує огляди та рецензії на нові ігри. Це дозволяє гравцям отримати об'єктивну інформацію про гру, її якість, геймплей, графіку та інші аспекти перед придбанням або вирішенням, чи варто грати в неї.

Поради та стратегії сайтів про відеоігри надає поради, стратегії та вказівки для гравців. Це допомагає гравцям поліпшити свої навички, зрозуміти складні механіки гри і отримати перевагу над іншими гравцями.

5.2 Умови виконання програми

Мінімальний рівень технічних засобів для забезпечення роботи програмної системи включає процесор не менш ніж 4 ядра і частотою 2,9 ГГц, оперативну пам'ять у 16 Гб, жорсткий диск у 120Гб.

5.3 Експлуатація програми

Робота користувача з програмою є достатньо простою. Кожен користувач має абсолютно однаковий інтерфейс. Для того щоб отримати доступ до системи, йому потрібно авторизуватися.

Одним із перших і головних вікон програми є головне вікно зі списком ігор та навігаційна панель (рис. 5.1).

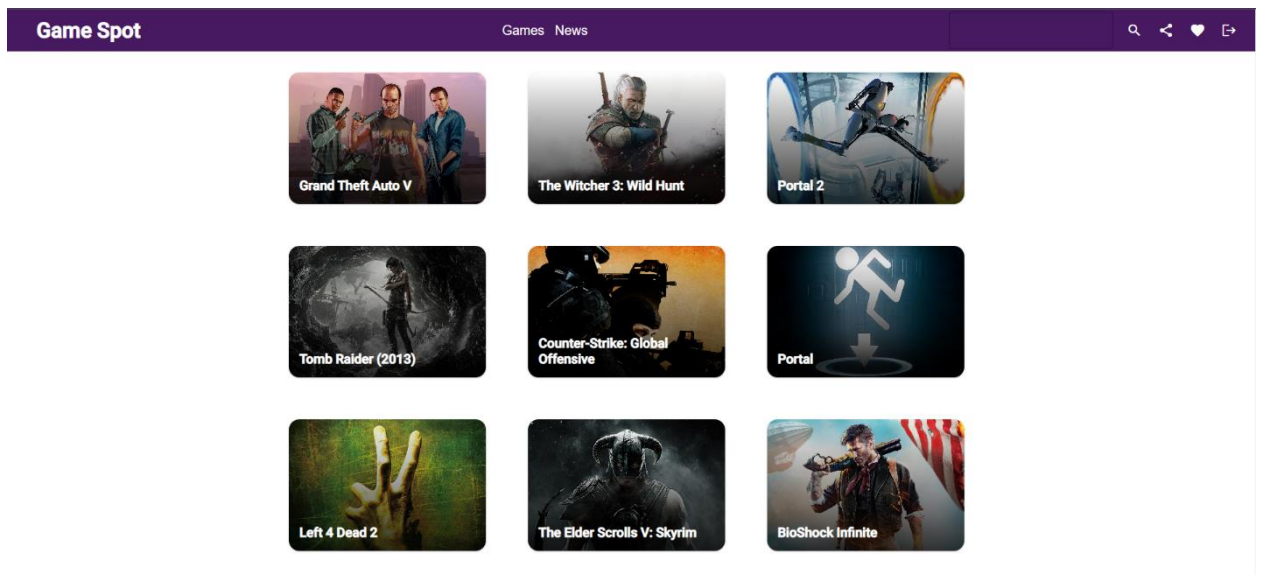


Рисунок 5.1 – Вікно головної сторінки

На будь який елемент можна натиснути та переглянути інформацію, будь це гра новини обране чи пошук або реєстрація (рис. 5.2).

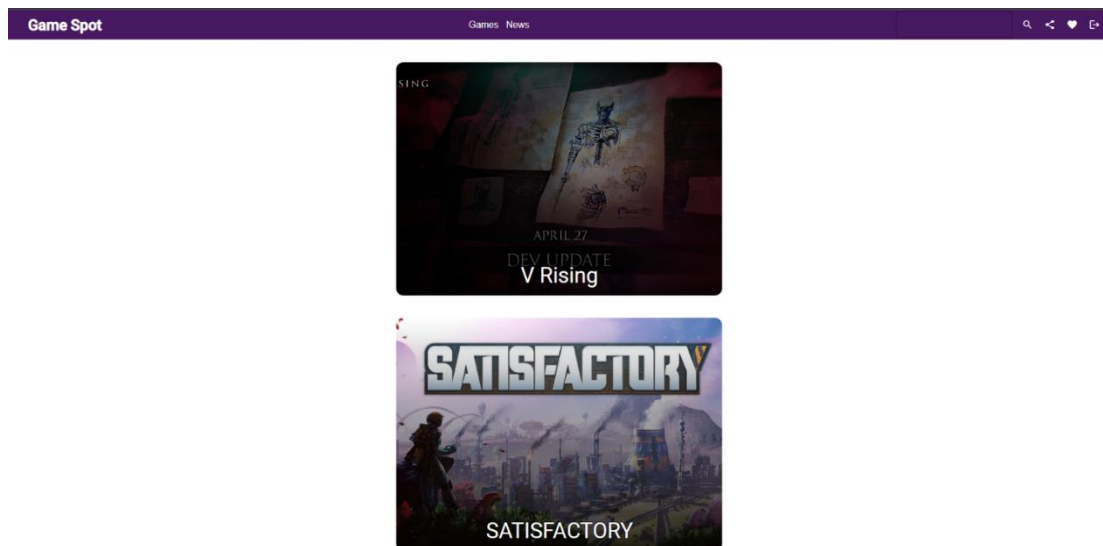


Рисунок 5.2 – Вікно сторінки новин

При натисканні на кнопку News можна побачити список новин про ігрову індустрію та додати ще новину (рис. 5.3).

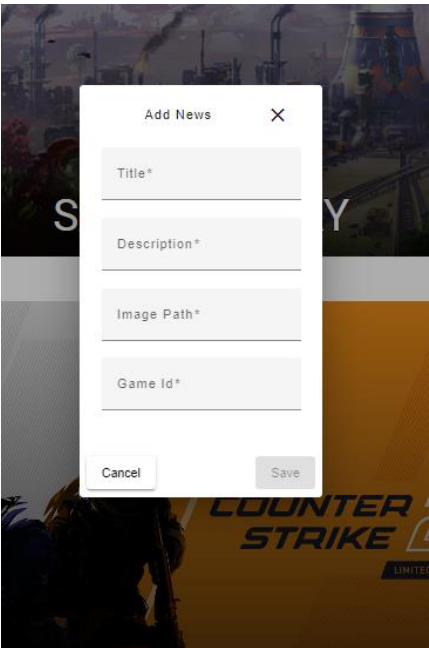


Рисунок 5.3 – Діалогове вікно додавання новини

При натисканні на гру можна побачити детальну інформацію про неї. (рис. 5.4).

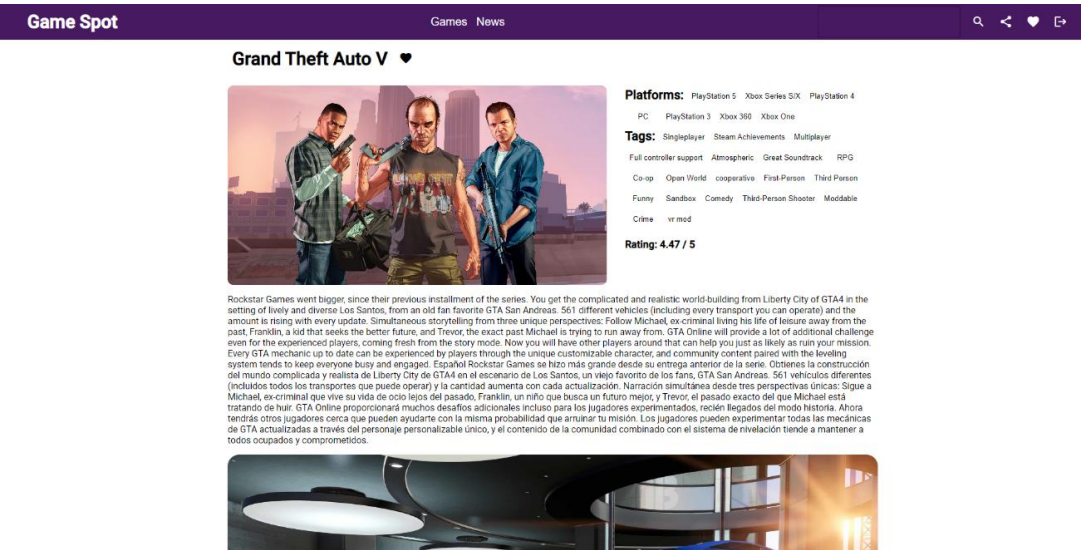


Рисунок 5.4 – Вікно інформації що до гри

Під кожною новиною або грою можна додати коментар (рис. 5.5).

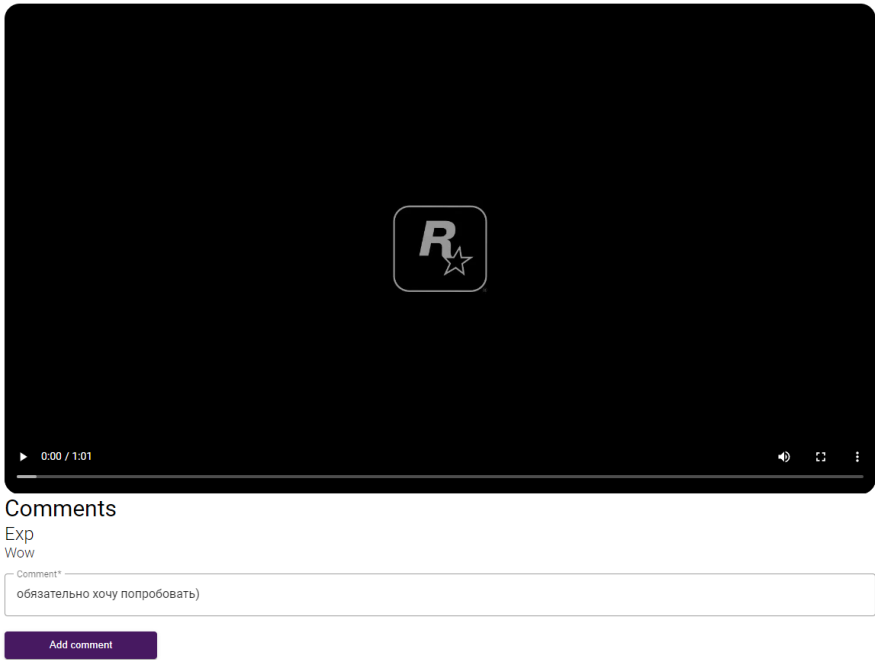


Рисунок 5.5 – Вікно коментарів гри

При натисканні на блок новини можна побачити інформацію про гру та розділ коментарів (рис. 5.6).

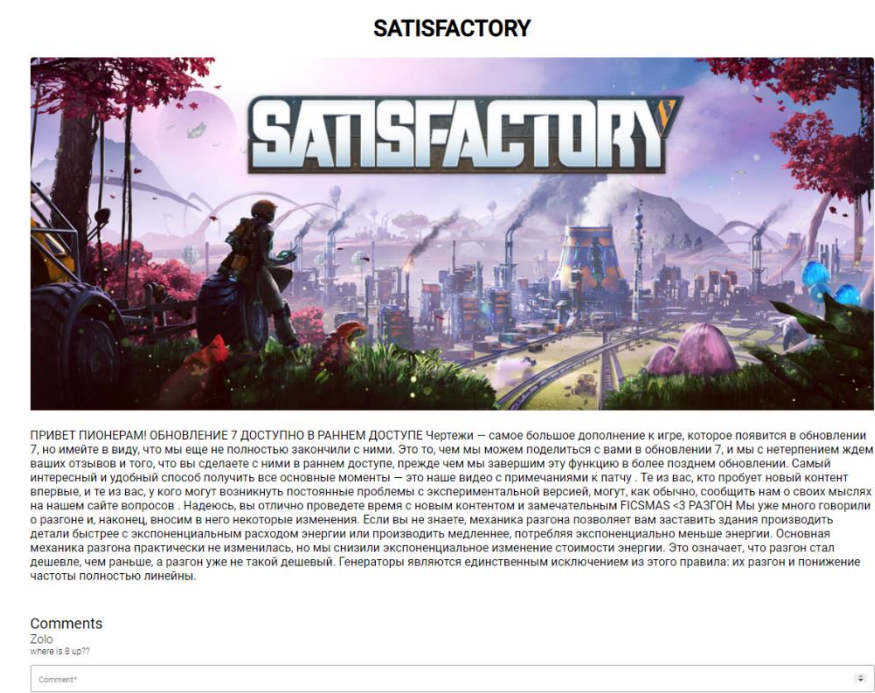


Рисунок 5.6 – Вікно новин гри

На сайті також можна фільтрувати пошук ігор за назвою (рис. 5.7).

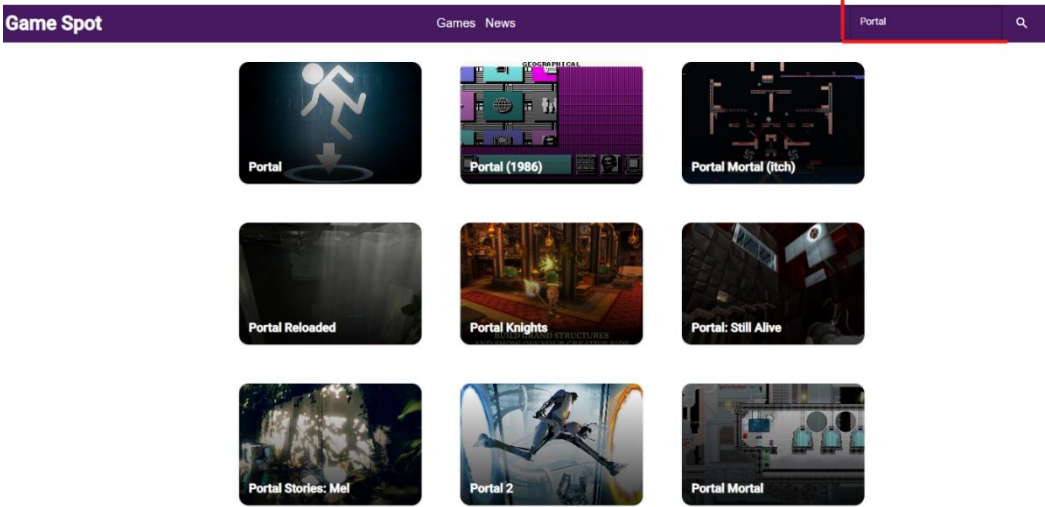


Рисунок 5.7 – Вікно пошуку

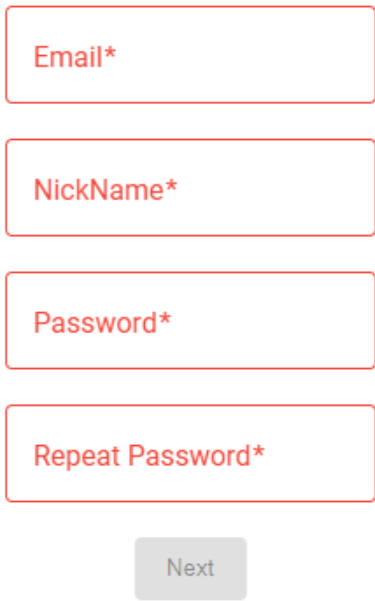
При натисканні на кнопку логіну можна перейти на сторінку входу до аккаунту або реєстрації якщо ви не маєте аккаунту (рис. 5.8).

Login

Don't have an account? [Sign up](#)

Рисунок 5.8 – Вікно входу

При натисканні на текст Sign up відкриється вікно реєстрації (рис. 5.9).

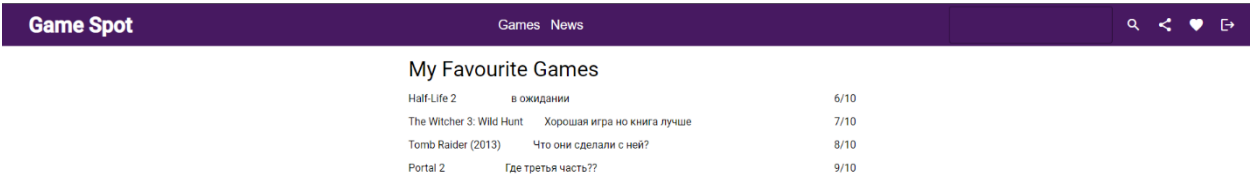


Registration form fields:

- Email*
- NickName*
- Password*
- Repeat Password*
- Next

Рисунок 5.9 – Вікно реєстрації

При натискані на кнопку «серце» можна побачити список обраних ігор, ваш коментар та рейтинг який ви указали (рис. 5.10).



Game Spot			Games	News		Search	Share	Heart	Exit
My Favourite Games									
Half-Life 2	в очікуванні								6/10
The Witcher 3: Wild Hunt	Хорошая игра но книга лучше								7/10
Tomb Raider (2013)	Что они сделали с ней?								8/10
Portal 2	Где третья часть??								9/10

Рисунок 5.10 – Вікно «обраного»

ВИСНОВКИ

В результаті було розробки було створено програмне забезпечення вебсайту з інформацією про комп'ютерні ігри які допомагають гравцям отримати доступ до важливої інформації, такої як огляди, оновлення, новини та поради щодо ігор. За результатами аналізу проблеми виконано аналіз аналогів і їх порівняння з розробкою, визначено її особливості.

За результатами проектування представлено результати порівняння фреймворків для розробки вебзастосунків мовою TypeScript та C#, обрано фреймворк Angular у цій якості, що визначило структуру програми на основі використання шаблону проектування MVC, представлено функціональні вимоги до програми, діаграму варіантів використання, схему бази даних і описано її відповідні таблиці і зв'язки між ними.

За результатами реалізації програми подано схему її функціонування, детально описано створені одиниці програми, включаючи класи моделей і уявлень, та загальну логіку роботи користувача. Під час виконання розробки та за результатами її виконання реалізовано тестування програми, що дозволило отримати результат, який відповідає запланованим для реалізації у технічному завданні вимогам.

Розроблена програма дозволила в результаті створити інструмент для зручного та швидкого огляду інформації що до комп'ютерних ігор та ігрової індустрії у цілому, реалізовано пошук та додавання новин та коментарів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. GameDev documentanion [Electronic resource]. – Access mode: <https://hsbi.hse.ru/articles/igrovaya-industriya-geymdev/>
2. Review of the game documentanion [Electronic resource]. – Access mode: https://komiinform.ru/page/novosti_i_trendy
3. Angular Офіційна документація [Electronic resource]. – Access mode: <https://angular.io/>
4. Стаття "Введение в TypeScript". [Electronic resource]. – Access mode: <https://metanit.com/web/typescript/1.1.php>
5. Офіційна документація ASP.NET Core. [Electronic resource]. – Access mode: <https://metanit.com/sharp/aspnet5/1.1.php>
6. Entity Framework Офіційна документація [Electronic resource]. – Access mode: https://skillbox.ru/media/code/entity_framework/
7. C# Офіційна документація [Electronic resource]. – Access mode: <https://docs.microsoft.com/en-us/dotnet/csharp/>
8. Офіційна документація React [Electronic resource]. – Access mode: <https://www.udemy.com/course/react-np/>
9. ASP.Net Core documentation [Electronic resource]. – Access mode: <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0>
10. Blazor for ASP.NET Core documentation [Electronic resource]. – Access mode: <https://docs.microsoft.com/en-us/aspnet/core/blazor>
11. MVC Pattern documentation [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/Model-View-Controller>
12. Code First to a New Database [Electronic resource]. – Access mode: <https://docs.microsoft.com/en-us/ef/ef6/modeling/codefirst/workflows/new-database>

ДОДАТОК А
Текст програми

```

using GameStop.WebAPI.Persistence.Entities;
using GameStop.WebAPI.Persistence.Repository;
using Imdb.WebApi.Dtos;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace GameStop.WebAPI.Controllers;

[ApiController]
[Route("[controller]")]
public class AuthController : ControllerBase
{
    private readonly IGameStopRepository _repository;

    public AuthController(
        ILogger<AuthController> logger,
        IGameStopRepository repository,
        IConfiguration configuration)
    {
        _repository = repository;
    }

    [HttpPost("login")]
    public async Task<IActionResult> Login(LoginRequestDto requestDto)
    {
        var user = await _repository.DbContext.Users.FirstOrDefaultAsync(x => x.Email
== requestDto.Email);
        if (user == null)
        {
            return BadRequest("User not found");
        }
        if (user.Password == requestDto.Password)
        {
            return Ok(new { user.Email, userId = user.Id, user.NickName });
        }

        return BadRequest("Email or password not correct");
    }

    [HttpPost("signup")]
    public async Task<IActionResult> SignUp(SignupRequestDto requestDto)
    {
        var user = await _repository.DbContext.Users.FirstOrDefaultAsync(x => x.Email
== requestDto.Email);
        if (user != null)
        {
            return BadRequest("User already exist");
        }
        var created = await _repository.CreateAsync(new User()
        {
            Email = requestDto.Email,
            Password = requestDto.Password,

```

```

        NickName = requestDto.NickName
    });
    return Ok(created);
}
}
using GameStop.WebAPI.Persistence.Entities;
using GameStop.WebAPI.Persistence.Repository;
using Imdb.WebApi.Dtos;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace GameStop.WebAPI.Controllers;

[ApiController]
[Route("[controller]")]
public class GameController : ControllerBase
{
    private readonly IGameStopRepository _repository;

    public GameController(
        ILogger<GameController> logger,
        IGameStopRepository repository,
        IConfiguration configuration)
    {
        _repository = repository;
    }

    [HttpPost("addToFavourite")]
    public async Task<IActionResult> AddToFavourite(AddToFavouriteRequestDto
requestDto, [FromQuery] long userId)
    {
        var movie = await _repository.DbContext.UserGames
            .FirstOrDefaultAsync(x => x.GameId == requestDto.GameId && x.UserId ==
userId);

        if (movie != null)
        {
            return BadRequest("already in fav");
        }

        var user = await _repository.DbContext.Users.FirstOrDefaultAsync(x => x.Id ==
userId);

        if (user != null)
        {
            await _repository.CreateAsync(new UserGame()
            {
                UserId = userId,
                GameId = requestDto.GameId,
                UserRating = requestDto.Rating,
                GameName = requestDto.GameName,
                UserDescription = requestDto.Description,
                User = user
            });
        }
    }
}

```

```

        return Ok("ooh yeah");
    }

    return BadRequest("cannot find user");
}

[HttpGet("getFavourites")]
public async Task<IActionResult> GetFavourites([FromQuery] long userId)
{
    var movies = await _repository.DbContext.UserGames.Where(x => x.UserId ==
userId).ToListAsync();
    return Ok(movies);
}

[HttpPost("createGameComment")]
public async Task<IActionResult> CreateGameComment([FromQuery] long userId,
[FromBody] CreateCommentRequestDto dto)
{
    var user = await _repository.DbContext.Users.FirstOrDefaultAsync(x => x.Id ==
userId);
    if (user != null)
    {
        var comment = new Comment()
        {
            User = user,
            UserId = user.Id,
            CommentText = dto.CommentText,
            GameId = dto.TargetId
        };
        var created = await _repository.CreateAsync(comment);
        return Ok(created);
    }

    return NoContent();
}

[HttpPost("createNewsComment")]
public async Task<IActionResult> CreateNewsComment([FromQuery] long userId,
[FromBody] CreateCommentRequestDto dto)
{
    var user = await _repository.DbContext.Users.FirstOrDefaultAsync(x => x.Id ==
userId);
    if (user == null)
    {
        return BadRequest();
    }

    var news = await _repository.DbContext.News.FirstOrDefaultAsync(x => x.Id ==
dto.TargetId);
    if (news == null)
    {
        return BadRequest();
    }

```

```

    }

    var comment = new NewsComment()
    {
        User = user,
        UserId = user.Id,
        Comment = dto.CommentText,
        NewsId = news.Id,
        News = news
    };
    var created = await _repository.CreateAsync(comment);

    return Ok(created);
}

[HttpPost("getGameComments")]
public async Task<IActionResult> GetGameComments([FromBody] long movieId)
{
    var comments = await _repository.DbContext.Comments.Include(x =>
x.User).Where(x => x.GameId == movieId).ToListAsync();
    return Ok(comments.Select(x => new CommentResponse()
    {
        Comment = x.CommentText,
        NickName = x.User.NickName
    }));
}

[HttpPost("getNewsComments")]
public async Task<IActionResult> GetNewsComments([FromBody] long newsId)
{
    var comments = await _repository.DbContext.NewsComments.Include(x =>
x.User).Where(x => x.NewsId == newsId).ToListAsync();
    return Ok(comments.Select(x => new CommentResponse()
    {
        Comment = x.Comment,
        NickName = x.User.NickName
    }));
}

[HttpPost("createNews")]
public async Task<IActionResult> CreateNews([FromBody]
CreateNewsRequestDto dto)
{
    var news = new News()
    {
        Title = dto.Title,
        Description = dto.Description,
        ImgPath = dto.ImgPath,
        GameId = dto.GameId,
    };
    var created = await _repository.CreateAsync(news);

```

```

        return Ok(created);
    }

    [HttpGet("getNews")]
    public async Task<IActionResult> GetNews()
    {
        var news = await _repository.DbContext.News.Include(x =>
x.NewsComments).ThenInclude(x => x.User).ToListAsync();
        return Ok(news.Select(x => new NewsResponseDto()
        {
            Id = x.Id,
            Description = x.Description,
            Tittle = x.Tittle,
            ImgPath = x.ImgPath,
            GameId = x.GameId,
            Comments = x.NewsComments.Select(z => new CommentResponse()
            {
                Comment = z.Comment,
                NickName = z.User.NickName
            }).ToList()
        }));
    }

    [HttpPost("getNewsDetails")]
    public async Task<IActionResult> GetNews([FromBody] long newsId)
    {
        var news = await _repository.DbContext.News.Include(x =>
x.NewsComments).ThenInclude(x => x.User).FirstOrDefaultAsync(x => x.Id == newsId);
        if (news == null)
        {
            return NoContent();
        }
        return Ok(new NewsResponseDto()
        {
            Id = news.Id,
            Description = news.Description,
            Tittle = news.Tittle,
            ImgPath = news.ImgPath,
            GameId = news.GameId,
            Comments = news.NewsComments.Select(z => new CommentResponse()
            {
                Comment = z.Comment,
                NickName = z.User.NickName
            }).ToList()
        }));
    }
}

using Microsoft.AspNetCore.Mvc;

namespace GameStop.WebAPI.Controllers;

[ApiController]

```

```

[Route("[controller]")]
public class WeatherForecastController : ControllerBase
{
    private static readonly string[] Summaries =
    {
        "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot",
        "Sweltering", "Scorching"
    };

    private readonly ILogger<WeatherForecastController> _logger;

    public WeatherForecastController(ILogger<WeatherForecastController> logger)
    {
        _logger = logger;
    }

    [HttpGet(Name = "GetWeatherForecast")]
    public IEnumerable<WeatherForecast> Get()
    {
        return Enumerable.Range(1, 5).Select(index => new WeatherForecast
            {
                Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
                TemperatureC = Random.Shared.Next(-20, 55),
                Summary = Summaries[Random.Shared.Next(Summaries.Length)]
            })
            .ToArray();
    }
}

using System.Text.Json.Serialization;
using GameStop.WebAPI.Persistence;
using GameStop.WebAPI.Persistence.Repository;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers().AddJsonOptions(x =>
x.JsonSerializerOptions.ReferenceHandler = ReferenceHandler.IgnoreCycles);
// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddCors();
builder.Services.AddDbContext<GameStopDbContext>(opts =>
{
    opts.UseNpgsql("Host=localhost;Port=5432;Database=gamestop;Username=postgres;Password
=ak1941658");
});
builder.Services.AddScoped<IGameStopRepository, GameStopRepository>();
var app = builder.Build();

```

```

app.UseCors(x =>
{
    x.AllowAnyHeader();
    x.AllowAnyMethod();
    x.AllowAnyOrigin();
});
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

import {Component, OnInit} from '@angular/core';
import {GameService} from "../../services/game-service";
import {ActivatedRoute, Router} from "@angular/router";
import {Comment, CommentType, GameListRequest, Root, SearchTypes} from
"./../utilities/game-types";
import {AddToFavoriteDialogComponent} from "../add-to-favorite-dialog/add-to-
favorite-dialog.component";
import {MatDialog} from "@angular/material/dialog";
import {map} from "rxjs";
import {GamestopService} from "../../services/gamestop-service";

@Component({
    selector: 'app-game-page',
    templateUrl: './game-page.component.html',
    styleUrls: ['./game-page.component.css']
})
export class GamePageComponent implements OnInit{
    public game: Root;
    public images: any;
    public video: any;
    public currentIndex = 0;
    comments: Comment[] = []
    public id: number = 0;
    commentTypes = CommentType

    constructor(private readonly gameService: GameService,
        private readonly gamestopService: GamestopService,
        private readonly router: Router,
        private readonly route: ActivatedRoute,
        private readonly dialog: MatDialog) {
        const routeId = this.route.snapshot.paramMap.get('id');

```

```

    if (routeId) {
      this.id = +routeId; // Use the + operator to convert the string to a number
      console.log(this.id);
    }
  }

  ngOnInit() {
    this.getGame();
    this.getScreenshots();
    this.getTrailers();
    this.initComments();
  }

  initComments(): any {
    this.gamestopService.getGameComments(this.id).subscribe(x => {
      console.log(x);
      this.comments = x;
    })
  }

  getGame() {
    this.gameService.getGameDetails(this.id).subscribe(x => this.game = x )
  }

  stripHtmlTags(inputString: string): string {
    return inputString.replace(/<[^\>]*>/g, "");
  }

  getObjectEntries() {
    return Object.entries(this.game).map(([key, value]) => ({ key, value }));
  }

  searchByPlatform(platform: number): void {
    console.log('Searching for:', platform);
    this.router.navigate(['/games'], { queryParams: { page: 1, page_size: 15, search:
`${platform}`, type: SearchTypes.platform } as GameListRequest });
  }

  searchByTag(tag: number): void {
    console.log('Searching for:', tag);
    this.router.navigate(['/games'], { queryParams: { page: 1, page_size: 15, search:
`${tag}`, type: SearchTypes.tag } as GameListRequest });
  }

  getScreenshots(): void {
    this.gameService.getGameScreenshots(this.id).subscribe(x => {
      this.images = x.results
    })
  }

  prevSlide(): void {
    this.currentIndex--;
  }

```

```

    if (this.currentIndex < 0) {
      this.currentIndex = this.images.length - 1;
    }
  }

  nextSlide(): void {
    this.currentIndex++;
    if (this.currentIndex >= this.images.length) {
      this.currentIndex = 0;
    }
  }

  getTrailers(): void {
    this.gameService.getGameVideos(this.id).subscribe(x => {
      this.video = x.results[0]
    })
  }

  addToFavourite() {
    this.dialog.open(AddToFavouriteDialogComponent, {
      data: {
        gameId: this.game.id,
        gameName: this.game.name
      }
    })
    .afterClosed()
    .pipe(map(x => x as any))
    .subscribe({
      next: dialogResult => {
        if (dialogResult) {
          this.gamestopService.addGameToFavourite({
            gameId: this.game.id,
            description: dialogResult.description,
            rating: dialogResult.rating,
            gameName: this.game.name
          })
          .subscribe(result => {
            console.log(result)
          }, error => console.log(error.message));
        }
        console.log(dialogResult);
      },
      error: x => {
        console.log(x)
      }
    })
  }

  reloadComments(): void {
    this.initComments();
  }

```

```

}

import {Component, OnChanges, OnInit, SimpleChanges} from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {GameService} from "../../services/game-service";
import {ActivatedRoute, Router} from "@angular/router";
import {GameListRequest} from "../../utilities/game-types";

@Component({
  selector: 'app-games-list',
  templateUrl: './games-list.component.html',
  styleUrls: ['./games-list.component.css']
})
export class GamesListComponent implements OnInit {
  public games: any
  public page: number = 2

  constructor(private readonly http: HttpClient,
               private readonly gameService: GameService,
               private readonly router: Router,
               private readonly route: ActivatedRoute) {

  }

  ngOnInit() {
    this.route.queryParams.subscribe((params) => {
      const search = params as GameListRequest;
      if (search) {
        this.getGames({page: 1, page_size: 15, search: search.search, type: search.type});
      } else {
        this.getGames();
      }
    });
  }

  goToGame(id: number): void {
    this.router.navigate(['game', id])
  }

  public getGames(request?: GameListRequest): void {
    if (request) {
      this.gameService.getGames(request).subscribe(x => {
        // @ts-ignore
        this.games = x.results;
      })
      return;
    }
    this.gameService.getGames({page: 1, page_size: 15}).subscribe(x => {
      // @ts-ignore
      this.games = x.results;
    })
  }
}

```

```

public loadMore(): void {
  this.page++
  this.gameService.getGames({page: this.page, page_size: 15}).subscribe(x => {
    // @ts-ignore
    this.games = this.games.concat(x.results);
  })
}
}

```

```

import {Component, OnInit} from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {GameService} from "../services/game-service";

```

```

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

```

```

  ngOnInit() {
    console.log('redirected to homepage')
  }
}

```

```

import {Component, OnInit} from '@angular/core';
import {FormControl, Validators} from "@angular/forms";
import {Router} from "@angular/router";
import {AuthService} from "../services/AuthService";

```

```

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

```

```

  public email = new FormControl("", [Validators.required]);
  public password = new FormControl("", [Validators.required]);

```

```

  constructor(
    private router: Router,
    private readonly authService: AuthService,
  ) {
  }

```

```

  ngOnInit(): void {
  }

```

```

  login() {

```

```

        this.authService.login(this.email.value, this.password.value).subscribe(x => {
            this.router.navigate(['/games'])
        }, error => {
            this.email.setValue("");
            this.password.setValue("");
            console.log(error.message);
        });
    }

    loginValid(): boolean {
        return this.password.valid && this.email.valid
    }

    goToSignup() {
        return this.router.navigate(['/signup']);
    }

}

import {Component, OnInit} from '@angular/core';
import {AbstractControl, FormControl, ValidationErrors, ValidatorFn, Validators}
from "@angular/forms";
import {Router} from "@angular/router";
import {AuthService} from "../../services/AuthService";

export function passwordMatchValidator(passwordControl: AbstractControl):
ValidatorFn {
    return (repeatPasswordControl: AbstractControl): ValidationErrors | null => {
        if (!passwordControl || !repeatPasswordControl) return null;
        if (passwordControl.value !== repeatPasswordControl.value) return
        {passwordMismatch: true};
        return null;
    };
}

@Component({
    selector: 'app-register',
    templateUrl: './register.component.html',
    styleUrls: ['./register.component.css']
})
export class RegisterComponent implements OnInit {
    public email = new FormControl("", [Validators.required])
    public nickname = new FormControl("", [Validators.required])
    public password = new FormControl("", [Validators.required])
    public repeatPassword = new FormControl("", [Validators.required,
passwordMatchValidator(this.password)])

    constructor(private readonly router: Router, private readonly authService:
AuthService) {
    }

```

```

ngOnInit(): void {
  this.email.markAsTouched();
  this.nickname.markAsTouched();
  this.password.markAsTouched();
  this.repeatPassword.markAsTouched();
}

signup() {
  this.authService.signup(this.email.value, this.password.value,
this.nickname.value).subscribe(x => {
    this.router.navigate(['/reg-successful'])
  }, error => {
    console.log(error.message);
    this.password.setValue("");
    this.repeatPassword.setValue("");
    this.email.setValue("");
    this.nickname.setValue("");
  });
}

passwordValid(): boolean {
  return this.password.value == this.repeatPassword.value && this.password.valid &&
this.repeatPassword.valid
}
}

import { Component } from '@angular/core';
import { Router } from "@angular/router";
import { GameListRequest, SearchTypes } from "../utilities/game-types";
import { AuthService } from "../services/AuthService";

@Component({
  selector: 'app-toolbar',
  templateUrl: './toolbar.component.html',
  styleUrls: ['./toolbar.component.css']
})
export class ToolbarComponent {
  searchText: string;
  constructor(private readonly router: Router, private readonly authService:
AuthService) {
  }

  performSearch() {
    console.log('Searching for:', this.searchText);
    this.router.navigate(['/games'], { queryParams: {page: 1, page_size: 15, search:
this.searchText, type: SearchTypes.name } as GameListRequest });
  }

  goToGames() {
    this.router.navigate(['/games'],{ queryParams: { search: null } })
  }
}

```

```

    }

    goToHome() {
      this.router.navigate(['/games'])
    }

    logout() {
      this.authService.logout()
      this.router.navigate(['/login']);
    }

    goToNews() {
      this.router.navigate(['/news'], { queryParams: { search: null } })
      console.log('USER: ', this.authService.getUser())
    }

    goToFavourite() {
      this.router.navigate(['/favs'])
    }
  }
}

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));

<div class="main-container">
  <ng-container *ngIf="game">
    <div class="name-container">
      {{ game.name }}      <button (click)="addToFavourite()" mat-icon-button>
        <mat-icon>favorite</mat-icon>
      </button>
    </div>
    <div class="top-container">
      <div class="image-name-rating">

        <div class="image-container">
          
        </div>

      </div>
      <div class="platforms-tags-container">
        <div class="platforms-container">
          Platforms:
          <ng-container *ngFor="let platform of game.platforms">
            <button (click)="searchByPlatform(platform.platform.id)" mat-
button>{{ platform.platform.name }}</button>
          </ng-container>
        </div>
      </div>
    </div>
  </ng-container>
</div>

```

```

<div class="tags-container">
  Tags:
  <ng-container *ngFor="let tag of game.tags">
    <button (click)="searchByTag(tag.id)" mat-button> {{tag.name}}</button>
  </ng-container>
</div>
<div class="rating-container">
  {{'Rating: '}}{{game.rating.toFixed(2)}}{{' / 5'}}
</div>
</div>
</div>
<div class="desc-container">
  <div>
    {{stripHtmlTags(game.description)}}
  </div>
</div>

<div class="slider-container">
  <ng-container *ngIf="images">
    <mat-card style="border-radius: 20px;" class="image-slide">
      <img style="border-radius: 20px;" [src]="images[currentIndex].image"
alt="Image slide" mat-card-image>
    </mat-card>
    <button (click)="prevSlide()" class="prev-button" mat-mini-fab>
      <mat-icon>chevron_left</mat-icon>
    </button>
    <button (click)="nextSlide()" class="next-button" mat-mini-fab>
      <mat-icon>chevron_right</mat-icon>
    </button>
  </ng-container>
</div>
<div class="video-container">
  <ng-container *ngIf="video">
    <video style="border-radius: 20px;" [poster]="video.preview" controls>
      <source [src]="video.data.max" type="video/mp4">
      Your browser does not support the video tag.
    </video>
  </ng-container>
</div>
<div class="comments-container">
  <div class="comments-tittle">
    Comments
  </div>
  <ng-container *ngFor="let comment of comments">
    <app-comment [comment]="comment">
    </app-comment>
  </ng-container>
  <app-create-comment
[commentType]="commentTypes.Games"
(targetId)="id"></app-create-comment>
</div>
</ng-container>

```

```

</div>

<div class="main-container">
  <ng-container *ngIf="games">
    <div class="games-block" >
      <ng-container *ngFor="let game of games">
        <div class="game-block">
          <mat-card (click)="goToGame(game.id)" class="game-card" [ngStyle]="{
            'background-image': 'linear-gradient(to top, rgba(0,0,0,0.8) 0%, rgba(0,0,0,0)
100%), url(' + game.background_image + ')',
            'background-size': 'cover',
            'background-position': 'center center'}">
            <mat-card-content class="game-name">
              <div >{{ game.name }}</div>
            </mat-card-content>
          </mat-card>
        </div>
      </ng-container>
    </div>
    <button mat-raised-button (click)="loadMore()" >Load more</button>
  </ng-container>
</div>
<div class="login-container">
  <mat-form-field appearance="outline">
    <mat-label>Email</mat-label>
    <input [formControl]="email" matInput type="email"/>
  </mat-form-field>
  <mat-form-field appearance="outline">
    <mat-label>Password</mat-label>
    <input [formControl]="password" matInput type="password"/>
  </mat-form-field>
  <button [disabled]="!loginValid()" mat-raised-button color="primary"
(click)="login()">Login</button>
  <p class="signup-prompt">
    Don't have an account? <a (click)="goToSignup()">Sign up</a>
  </p>
</div>
<div class="reg-container">
  <mat-form-field appearance="outline" class="full-width">
    <mat-label>Email</mat-label>
    <input autocomplete="off" [formControl]="email" [value]="email.value" matInput
type="email">
  </mat-form-field>
  <mat-form-field appearance="outline" class="full-width">
    <mat-label>NickName</mat-label>
    <input autocomplete="off" [formControl]="nickname" [value]="nickname.value"
matInput type="text">
  </mat-form-field>
  <mat-form-field appearance="outline" class="full-width">
    <mat-label>Password</mat-label>
    <input autocomplete="off" [formControl]="password" [value]="password.value"
matInput type="password">

```

```

    </mat-form-field>
    <mat-form-field appearance="outline" class="full-width">
      <mat-label>Repeat Password</mat-label>
      <input
        autocomplete="off"
        [formControl]="repeatPassword"
        [value]="repeatPassword.value" matInput type="password">
    </mat-form-field>
    <mat-error
      *ngIf="repeatPassword.errors
      repeatPassword.errors['passwordMismatch']">
      Passwords do not match
    </mat-error>
    <div class="submit-button">
      <button [disabled]="!passwordValid()" (click)="signup()" color="primary" mat-
        raised-button>Next</button>
    </div>
  </div>
  <mat-toolbar class="toolbar" color="primary">
    <div class="side-menu" style="padding-left: 32px">
      <span (click)="goToHome()" style="cursor: pointer; font-size: 32px; font-weight:
        bold">Game Spot</span>
    </div>

    <div class="navigation">
      <button [routerLink]="['/games']" mat-button style="font-size: 20px">
        Games
      </button>
      <button [routerLink]="['/news']" mat-button style="font-size: 20px">
        News
      </button>
    </div>
    <div class="right-side">
      <div class="search" >
        <mat-form-field class="search-form-field" appearance="outline">
          <input style="color: white" matInput type="text" [(ngModel)]="searchText"
            (keyup.enter)="performSearch()" />
        </mat-form-field>
        <button mat-icon-button (click)="performSearch()">
          <mat-icon>search</mat-icon>
        </button>
      </div>
      <div class="share">
        <button mat-icon-button>
          <mat-icon>share</mat-icon>
        </button>
      </div>
    </div>
  </div>

```

Додаток Б
Слайди презентації



Рисунок Б.1 – Слайд № 1

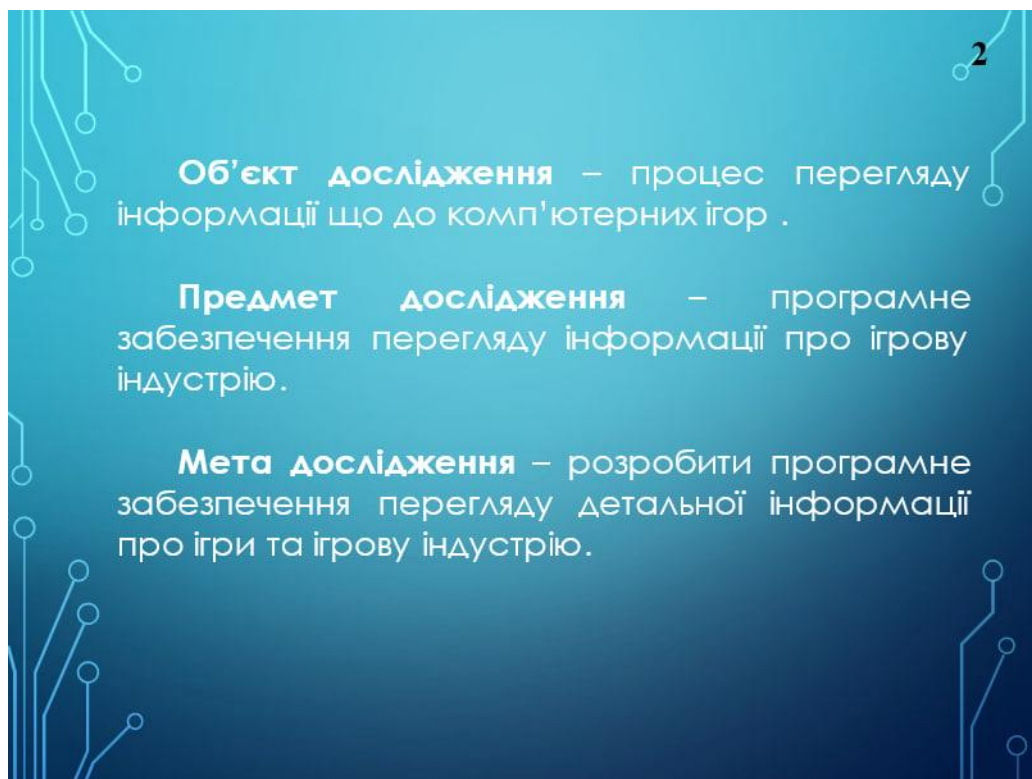


Рисунок Б.2 – Слайд № 2

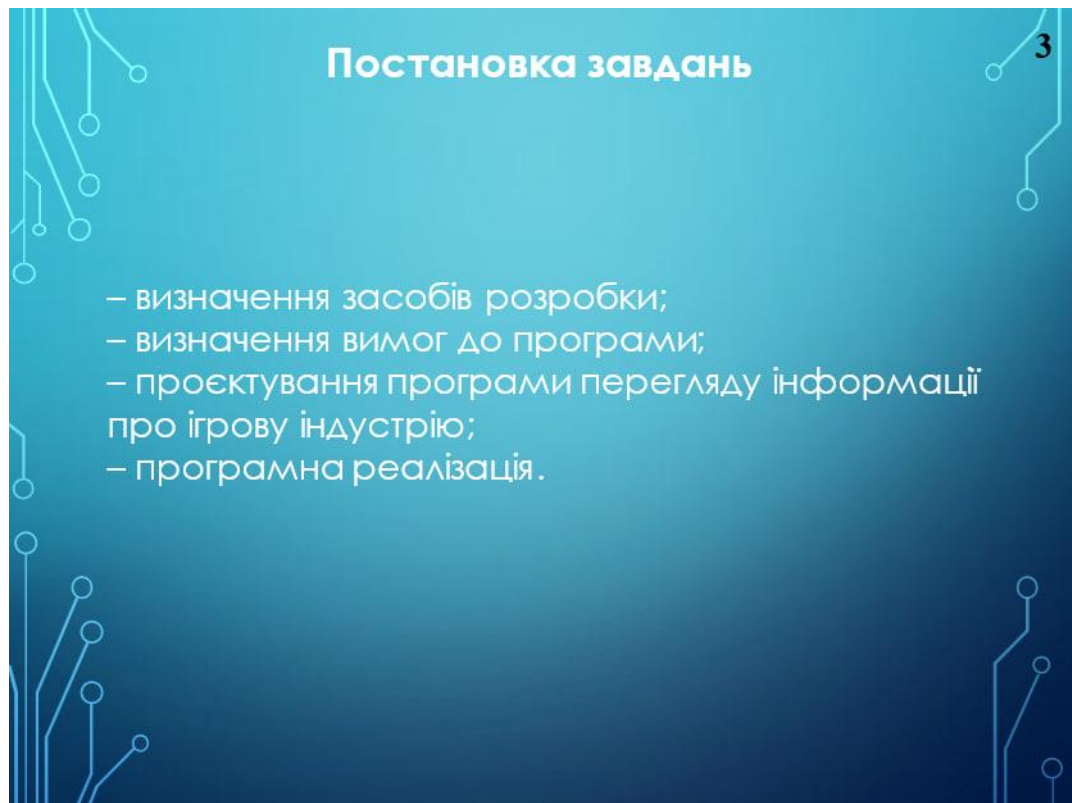


Рисунок Б.3 – Слайд № 3

Вибір веб-фреймворку розробки мовою TypeScript

Показник	Angular	React
Високорівневий API	+	–
Висока оптимізація	+	–
Крос-платформеність	–	+
Формування динамічних веб-сторінок	+	+
Використовувана мова програмування	TypeScript	TypeScript

Рисунок Б.4 – Слайд № 4

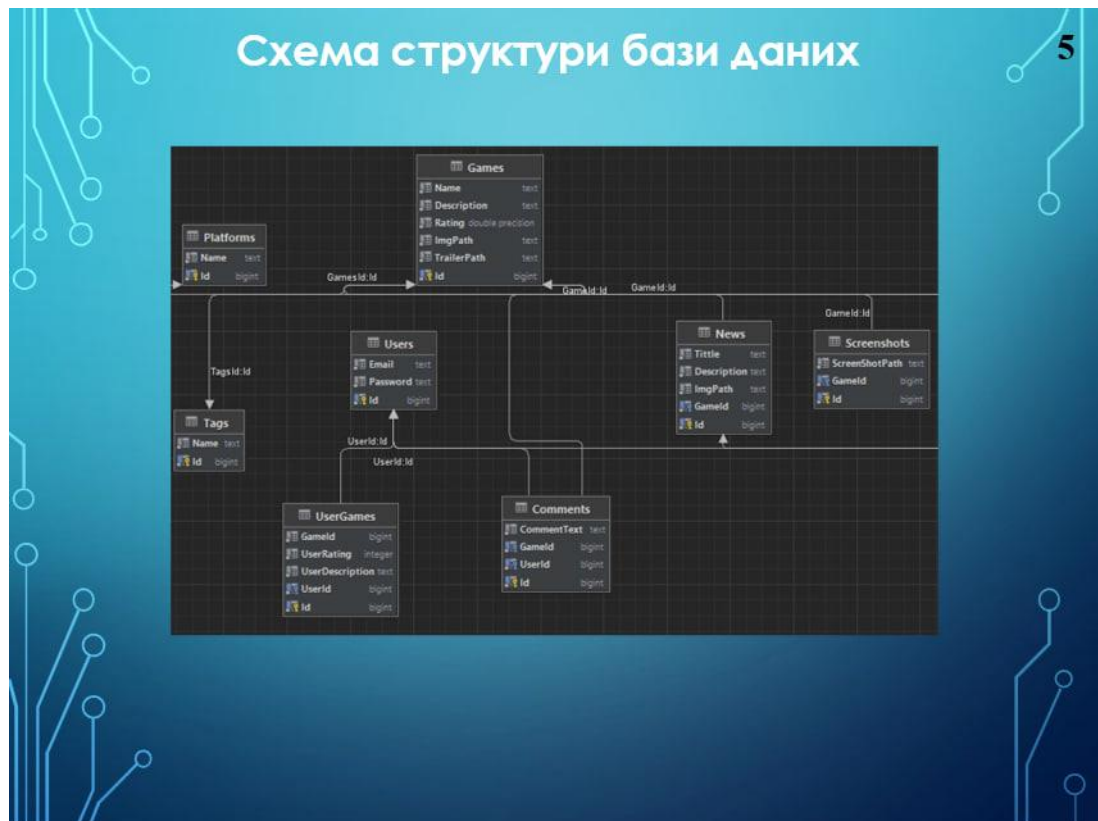


Рисунок Б.5 – Слайд № 5



Рисунок Б.6 – Слайд № 6

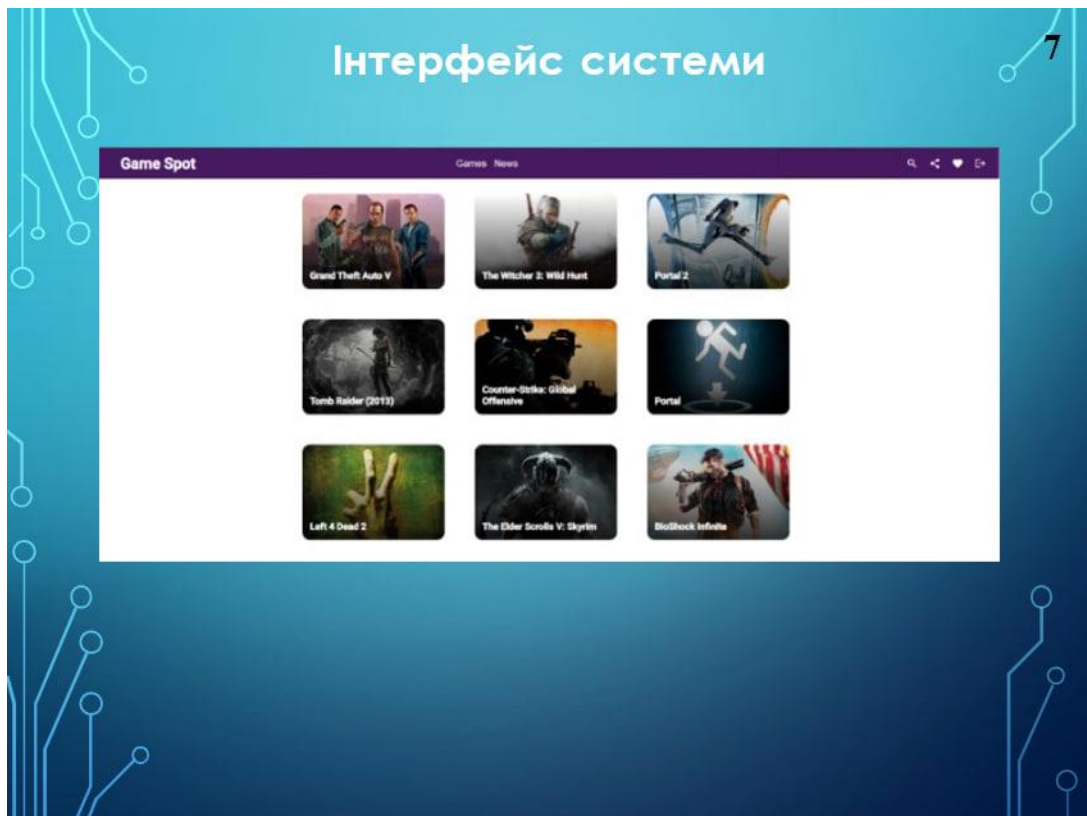


Рисунок Б.7 – Слайд № 7

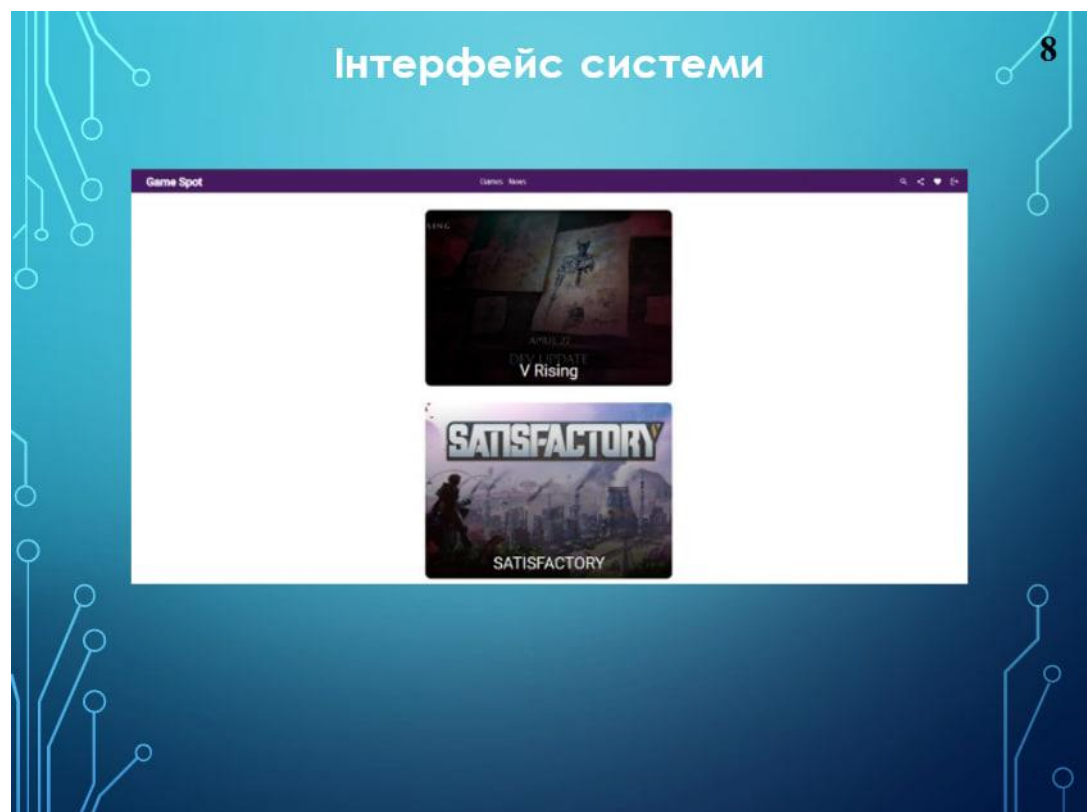


Рисунок Б.8 – Слайд № 8

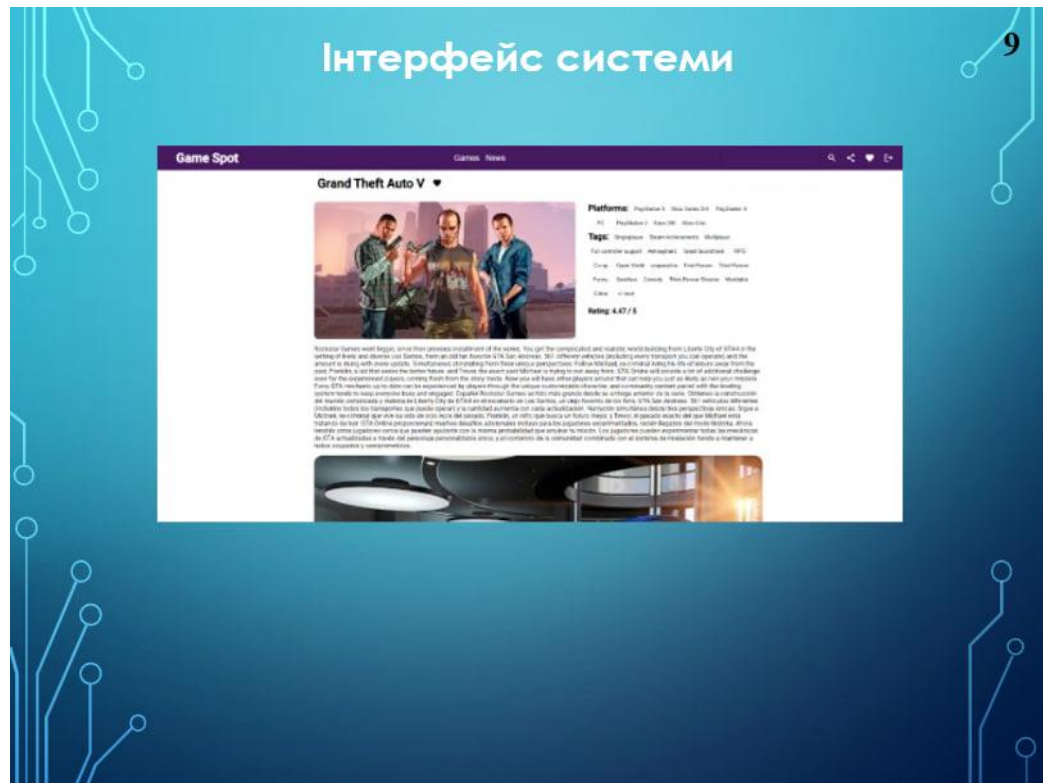


Рисунок Б.9 – Слайд № 9

Висновки

10

За результатами аналізу проблеми виконано аналіз аналогів і їх порівняння з розробкою, визначено її особливості, що стосуються перегляду інформації щодо комп'ютерних ігор та ігрової індустрії.

За результатами проєктування представлено результати порівняння фреймворків для розробки вебзастосунків мовою TypeScript та C#, обрано фреймворк Angular у цій якості, що визначило структуру програми на основі використання шаблону проєктування MVC, представлено функціональні вимоги до програми, діаграму варіантів використання, схему бази даних і описано її відповідні таблиці і зв'язки між ними.

За результатами реалізації програми подано схему її функціонування, детально описано створені одиниці програми. Під час виконання розробки та за результатами її виконання реалізовано тестування програми, що дозволило отримати результат, який відповідає запланованим для реалізації у технічному завданні вимогам.

Рисунок Б.10 – Слайд № 10