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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка ігрового порталу з використанням технології Nuxt JS, Vue JS та Elastic Search

Виконав(ла): студент(ка) VI курсу, групи СПм-62
спеціальності 121 Програмна інженерія
(шифр і назва спеціальності)

(підпис) (прізвище та ініціали)

Керівник (підпис) (прізвище та ініціали)

Нормоконтроль (підпис) (прізвище та ініціали)

Завідувач кафедри (підпис) (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2023

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

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

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 121 Програмна інженерія
(шифр і назва спеціальності)

студенту _____
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка ігрового порталу з використанням технології Nuxt JS, Vue JS та Elastic Search

Керівник роботи _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 20___ року № _____

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи _____

4. Зміст роботи (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота магістра: 89 сторінок, 64 рисунки, 10 таблиць, 22 літературних джерела.

Мета роботи є розробка ігрового порталу для взаємодії між собою користувачів, що зацікавлені в іграх.

Завдання роботи є розробка ігрового порталу з використанням технології Nuxt JS, Vue JS та Elastic Search.

При розгляді типових рішень детально проаналізовано наступні ігрові портали: «NeoFAQ», «PC Gamer» та «GameFAQ».

Розробка порталу проводилась за допомогою двох JavaScript фреймворків: Vue JS і Nuxt JS, що надає можливість реалізації користувацької частини сайту в поєднанні із ElasticSearch. В якості мов програмування було обрано PHP, для роботи із базою даних, JavaScript для обробки даних на користувацькій частині та HTML і CSS для написання розмітки і стилістики сайту.

Результатом розробки є ігровий портал, що містить все необхідне для користувачів зацікавлених в іграх, надаючи можливості в перегляді різного роду матеріалів та участі в створенні власного. Даний сайт може бути запропонований широкому колу користувачів.

HTML, CSS, JAVASCRIPT, VUE, NUXT, ELASTICSEARCH, MYSQL, PHP, LARAVEL, ПОРТАЛ, ТЕСТУВАННЯ.

ANNOTATION

The aim of the work is to develop a game portal for interaction between users interested in games.

The task is to develop a game portal using Nuxt JS, Vue JS and Elastic Search technologies.

Considering typical solutions, the following forums are analysed in detail "NeoFAQ, PC Gamer and GameFAQ.

The portal was developed using two JavaScript frameworks: Vue JS and Nuxt JS, which allows the user part of the site to be implemented in combination with ElasticSearch. PHP was chosen as the programming language for working with the database, JavaScript for processing data on the user part, and HTML and CSS for writing markup and styling the site.

The result of the development is a games portal that contains everything necessary for users interested in games, providing opportunities to view different types of material and to participate in the creation of their own. This site can be offered to a wide range of users.

HTML, CSS, JAVASCRIPT, VUE, NUXT, ELASTICSEARCH, MYSQL, PHP, LARAVEL, PORTAL, TESTING.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

JS – JavaScript.

HTML – Hypertext Markup Language.

CSS – Cascading Style Sheets.

PHP – Hypertext Preprocessor.

SQL – Structured Query Language.

UI – User Interface.

DOM – Document Object Model.

XML – Extensible Markup Language.

API – Application Programming Interface.

ORM – Object–relational mapping.

MVC – Model View Controller.

IDE – Integrated development environment.

RegExp – Regular expression.

WYSIWYG - What You See Is What You Get.

UX - User Experience.

GUI - Graphical User Interface.

SSR - Server-Side Rendering.

SSG - Static Site Generation.

ВСТУП

Ігрові портали стали не просто платформами для гри, а цілими віртуальними світами, що залучають мільйони гравців з усього світу. Сучасні ігрові портали вже давно вийшли за межі простої гри та стали центрами віртуальної соціальної взаємодії, де гравці об'єднуються за спільними інтересами та цілями.

Ігрові портали – це не лише місце для розваги, це й платформи для обміну досвідом, знаходження нових друзів і навіть для професійного росту. Сучасні геймери зібралися на цих порталах, щоб не лише насолоджуватися ігровим процесом, але й обговорювати нові стратегії, ділитися порадами та враженнями з іншими гравцями.

Кваліфікаційна робота з ігрового порталу передбачає не лише ознайомлення з різноманітними інформаційними джерелами, а й аналіз різних технологій для створення оптимального ігрового середовища. Розглядаючи функціонал обраних технологій, можна підібрати той, що найкраще відповідає потребам створення ігрового порталу.

Завдання на кваліфікаційну роботу є наступними:

– дослідити інформаційні джерела, пов'язані із ігровими порталами.

– проаналізувати різноманітні технології, приділяючи увагу їх можливостям для створення ігрового порталу.

– розглянути функціональні можливості обраних технологій у контексті ігрових порталів.

– спроектувати структуру та функціональність ігрового порталу.

– розробити клієнтську та серверну частини ігрового порталу.

– Провести тестування створеного веб-порталу для гравців.

– Провести аналіз аспектів охорони праці у процесі створення ігрового порталу.

1 ДОСЛІДЖЕННЯ ОБ’ЄКТУ ІНФОРМАТИЗАЦІЇ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Характеристика об’єкту інформатизації

Веб-сайти є не лише набором веб-сторінок, але й вони функціонують як віртуальні платформи для різноманітних потреб. Їхня різноманітність віддзеркалюється у їхніх завданнях: від інформування та освіти до розваг та комерції. Окрім того, веб-сайти часто є ключовими інструментами для спілкування та обміну інформацією між користувачами, створюючи цифрові спільноти зі спільними інтересами та цілями [1].

Ці платформи можуть забезпечувати доступ до широкого спектру ресурсів, починаючи від інтерактивних елементів, таких як відео та зображення, і до більш складних інтерактивних можливостей, що сприяють обговоренню та співпраці між користувачами. Це створює унікальні можливості для навчання, торгівлі, розваг та спілкування, що робить веб-сайти важливими елементами нашого цифрового життя.

Інтернет насичений різними типами веб-сайтів, кожен з яких має свою мету:

1) Блоги є платформами, які зазвичай керує окрема особа або маленька команда, що дозволяє їм ділитися різноманітними темами. Це можуть бути поради з моди, музики, подорожей, фітнесу та багато іншого. У сучасному Інтернеті ведення блогу стало популярним способом заробітку, де професіонали діляться своїми знаннями, привертаючи аудиторію та отримуючи винагороду за це [2].

2) Електронна комерція – це сфера, відома також як інтернет-магазини. Ці онлайн-платформи дозволяють користувачам здійснювати шопінг через Інтернет та оплачувати товари та послуги онлайн. Цей тип магазинів часто існують у формі окремих веб-сайтів [3].

3) Портфоліо – це онлайн-середовище, яке розглядається як розширення резюме фрілансера або спеціаліста. Це дає можливість потенційним клієнтам зручно

переглядати ваші роботи та послуги, а також допомагає вам розширити свій професійний спектр навичок [4].

4) Брошури – це типи веб-сайтів, що зазвичай використовуються невеликими підприємствами. Вони функціонують як цифрові візитні картки, що спрощують відображення контактної інформації та реклами послуг лише на кількох сторінках [5].

5) Новинні та журнальні веб-сайти практично не потребують пояснень: їхня основна мета — інформувати читачів про поточні події. У той час як новинні ресурси надають актуальні новини, журнали спрямовані на розваги та розвиток [6].

6) Соціальні мережі – це не лише майданчики для обміну думками та мультимедійним контентом. Facebook, Twitter, Reddit та інші платформи є справжніми центрами спілкування, які не лише дозволяють обмінюватися інформацією, а й створюють глибокі спільноти з величезним потенціалом. Вони збирають людей за різними інтересами, дозволяють обговорювати важливі теми та розважатися, створюючи неймовірну різноманітність змісту [7].

7) Сайти з освітнім спрямуванням - це платформи, які досить зрозумілі за своєю назвою. Вони створені для представлення інформації через різноманітні медіа-формати, такі як аудіо, відео та зображення [8].

8) Портал – це спеціалізований веб-сайт, що об'єднує багато різноманітних ресурсів та сервісів на певну тематику. Це цифрова платформа, яка надає доступ до різних інструментів, інформаційних ресурсів та можливостей у зручному форматі. Портали забезпечують швидкий доступ до корисної інформації та сервісів, спрямовані на конкретні потреби чи інтереси користувачів [9].

1.2 Обґрунтування доцільності створення ігрового порталу

Ігрові портали – це майданчики для гравців, де вони можуть обговорювати актуальні ігральні теми, обмінюватися порадами та спілкуватися. Інформація на таких порталах зберігається та доступна для перегляду у будь-який момент. Хоча більшість порталів прості у реєстрації, існують ексклюзивні, приватні портали, доступ до яких можливий лише за певними умовами чи оплатою членства.

Ігровий портал, створює спільнота гравців, навіть у випадку, коли обсяг трафіку є обмеженим залежно від тематики. Ці портали привертають користувачів для перевірки статусу обговорень та новин у своєму ігральному світі. Ця повторювальна відвідуваність робить портал «липким».

Гості, а також гравці, зазвичай шукають конкретну інформацію на ігрових порталах, яку їм складно знайти на інших веб-сайтах. Це забезпечує постійний рух та активність на веб-порталі. Успішні та добре керовані ігрові портали можуть стати додатковим авторитетом для веб-сайту.

Ігрові портали розширюють межі спілкування, дозволяючи людям з різних країн об'єднуватися у віртуальних світах, де вони можуть взаємодіяти, спільно грати та обмінюватися досвідом. Це відкриває нові можливості для розвитку спільнот, сприяє формуванню дружніх стосунків, а також сприяє обміну культурними впливами та ідеями. Ці ігрові портали стають місцем зустрічі, в якому гравці різного віку та культурних пристрастей знаходять спільну мову через спільні ігрові переживання та співпрацю.

Висока кількість переглядів сторінок на одного користувача, який зацікавлений у грі та обговореннях на порталі, допомагає підвищити популярність та рейтинг веб-сайту у даній галузі. Авторитетність у ігральному світі може покращити рейтинг сторінок і загальний трафік веб-сайту.

1.3 Аналіз наявних рішень

Нині, практично не можливо створити унікальний сайт, оскільки, ринок веб-індустрії є одним із найпопулярніших напрямків в IT, в цілому, тому при розробці нових сайтів, основною ціллю є надання веб-сайту, пізнаваної особливості.

Веб портали, як такі, стають все більшими і набувають широкої популярності, проте в тематиці геймерства є не так багато дійсно хороших рішень і перше з них «PC Gamer».

Портал PC Gamer був запущений разом із перенесенням сайту PC Gamer, входячи до мережі Computer and Video Games, яка охоплює всі ігрові журнали Future plc. Цей крок викликав певні суперечки, оскільки багато старих користувачів покинули форум через обмежений інтервал нового порталу, рекламу та повільний час завантаження. Впровадження порталу розглядалося як одна з корисних функцій переходу. Після цього портал регулярно оновлювався завдяки внескам багатьох співробітників журналу. Теми, що обговорювалися, варіювалися від спірних питань про насильницькі відеоігри до переваг придбання ПК порівняно з консолями.

У 2010 році PC Gamer провів повторний запуск свого веб-сайту та блогу, об'єднавши онлайн-спільноти журналів США та Великобританії на єдиному веб-сайті. В результаті блог PC Gamer тепер містить внески як з американського, так і з британського журналів, всі розміщені на новому веб-сайті разом із порталами для обох журналів.

Портал сайту PC Gamer має довгу історію і розроблявся та доповнювався спільно з блогом. Наразі він має дуже велику базу матеріалів та активних користувачів.

На рисунку 1.1 можна побачити головну сторінку веб-сайту PC Gamer.

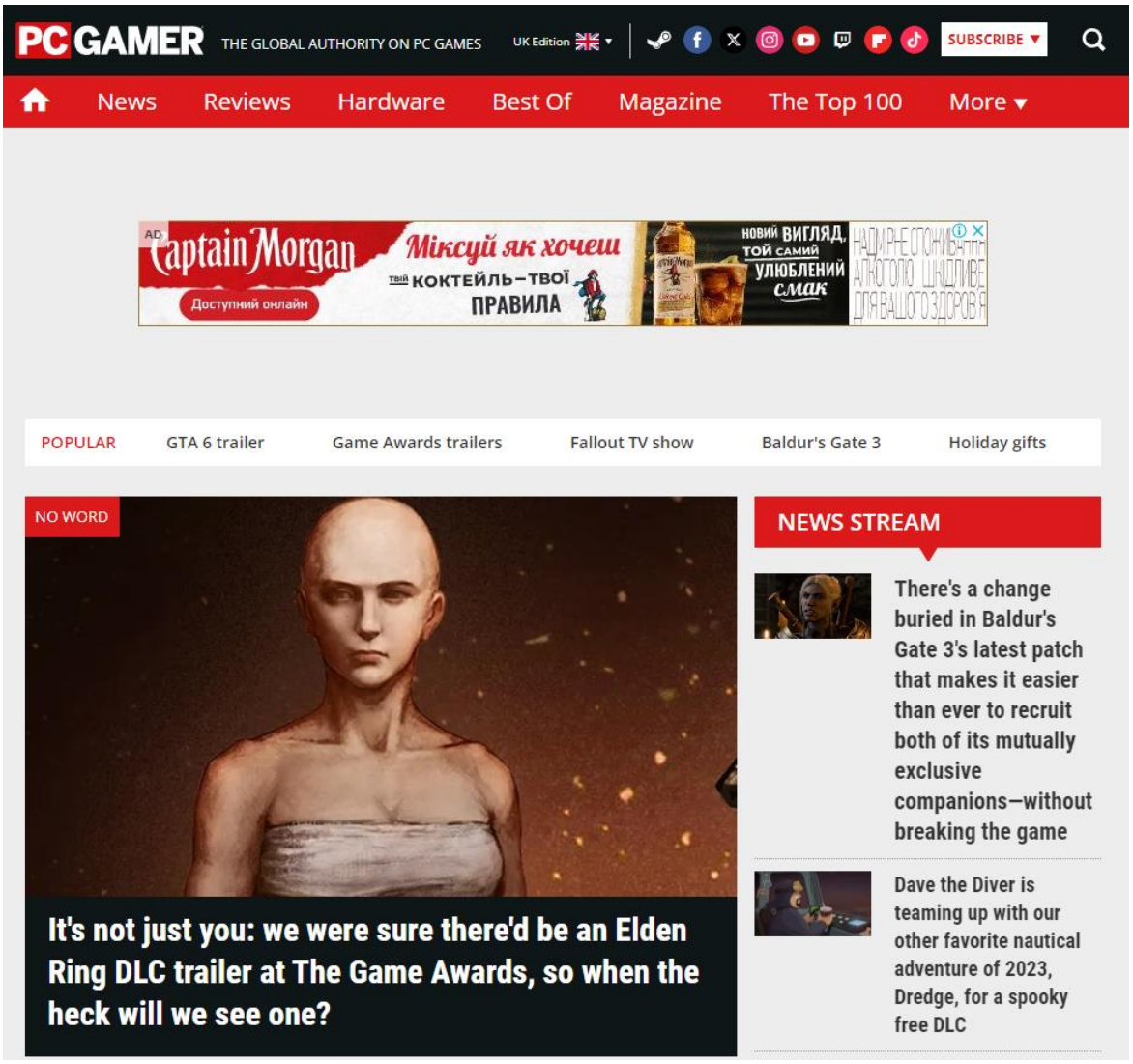

Рисунок 1.1 – Головна сторінка веб-сайту PC Gamer.

Наступним відомим веб-порталом є NeoGAF, раніше відомий як Gaming-Age Forums, присвячений, головним чином, обговоренню відеоігор. Заснований, як додаток до сайту новин про відеоігри, 4 квітня 2006 року, він змінив назву на NeoGAF і став незалежним хостингом з власним адмініструванням.

У 2017 році власника сайту Тайлера «Evilore» Малку звинуватили у сексуальних домаганнях. Звинувачення призвели до відставки модераторів, масового виходу з

сайту та згодом зміни політики сайту. Колишні учасники та модератори пізніше запустять новий форум ResetEra.

NeoGAF почався як "The Gaming-Age Forums", форум для ігрового веб-сайту Gaming-Age. Оскільки Gaming-Age переріс свій хостинг, IGN взяла на себе розміщення форумів Gaming-Age. Після того, як IGN припинив розміщення GAF у середині 2001 року, GAF перейшов на ezboard , і адміністрація GAF стала більш віддаленою від Gaming Age.

Оскільки персонал Gaming Age поступово відлучався від повсякденної роботи GAF, виникли проблеми з новим хостингом Gamesquad. Помилки програмного забезпечення у vBulletin 2, версія, яку використовував GAF на той час, продовжували погіршуватися, хостинг Gamesquad ставав дедалі більш непрактичним, поки база даних форумів не стала пошкодженою, що змусило перейти на новий хостинг, щоб змінити програмне забезпечення та врятувати те, що залишився з бази даних форумів. Навесні 2004 року був проведений збір коштів для переміщення GAF на новий хостинг. 6 червня 2004 року GAF прийняв нову форму (відомий як NeoGAF для давніх плакатів) і перейшов на новий хостинг і нове програмне забезпечення, vBulletin 3.

На рисунку 1.2 інтерфейс головної сторінки та основні теми, що доступні на сайті NeoGAF.

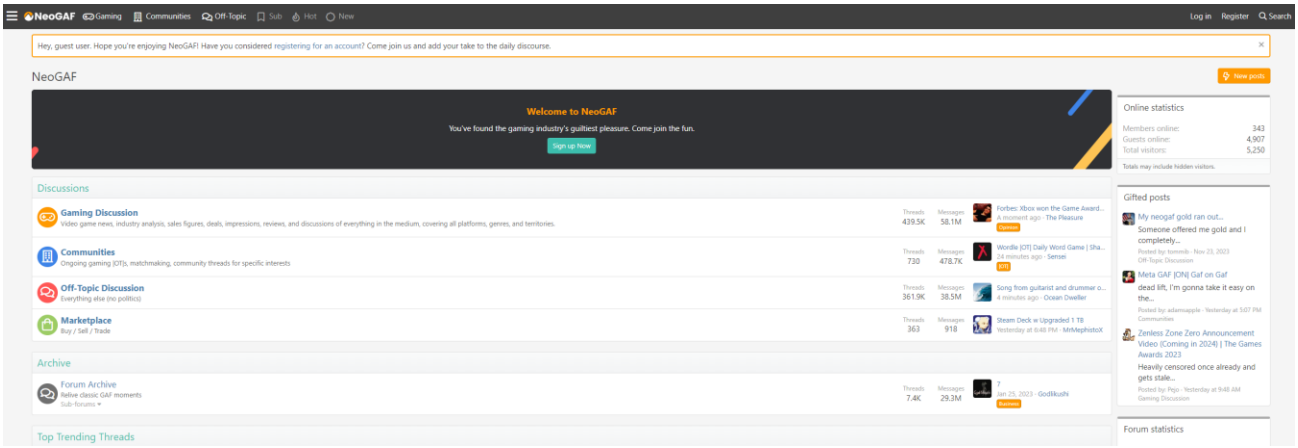

Рисунок 1.2 – Головна сторінка веб-сайту NeoGAF.

Ще одним відомим веб-порталом, спрямованим на ігрову тематику, є GameFAQs – це веб-сайт, який розміщує поширені запитання та покрокові інструкції для відеоігор. Він був створений у листопаді 1995 року Джеффом Візі та був куплений CNET Networks у травні 2003 року. Нині його власником є Red Ventures. Сайт має обширну базу даних інформації про відеоігри, чіт-кодів, оглядів, збережень ігор, зображень у формі обкладинок та знімків екрану, більшість з яких надіслані гравцями. Матеріали на порталі переглядає поточний редактор Аллен «SBAllen» Тайнер.

GameFAQs має активну спільноту веб-порталу, яка включає окремі розділи для обговорення кожної гри в базі даних сайту, а також декілька інших розділів. З 2004 по 2012 рік більшість обговорень ігор велися на GameFAQs та GameSpot, іншому веб-порталі від CBS. Однак 23 березня 2012 року було оголошено, що контент знову буде розділено між цими сайтами. З 7 травня 2012 року спільні розділи обговорень на GameFAQs та GameSpot стали доступні тільки для читання. З 30 листопада 1999 року GameFAQs проводить щоденні опитування спільноти та турніри, а також річні битви персонажів.

Кожна гра, перелічена на GameFAQs, має власний розділ для обговорень, де як новачки, так і досвідчені гравці можуть обговорювати ігрові стратегії та інші теми, пов'язані з іграми. GameFAQs також надає окремі розділи для спілкування на різноманітні теми (такі як аніме, телебачення, музика та професійна боротьба), а також окремі розділи для користувачів з певних регіонів (наприклад, Великобританія, Австралія/Нова Зеландія). На порталі є також розділи для офіційних повідомлень, обговорень учасників, конкурсів, пропозицій та допомоги.

Повідомлення, розміщені на розділах порталу, переважно є текстовими. Для форматування тексту використовується розмітка HTML, включаючи теги для жирного та курсивного виділення. На порталі встановлений фільтр слів, що запобігає використанню певних вульгарних слів, для забезпечення безпеки серед користувачів порталу. На рисунку 1.3 зображено ключові функції сайту та власне головну сторінку.

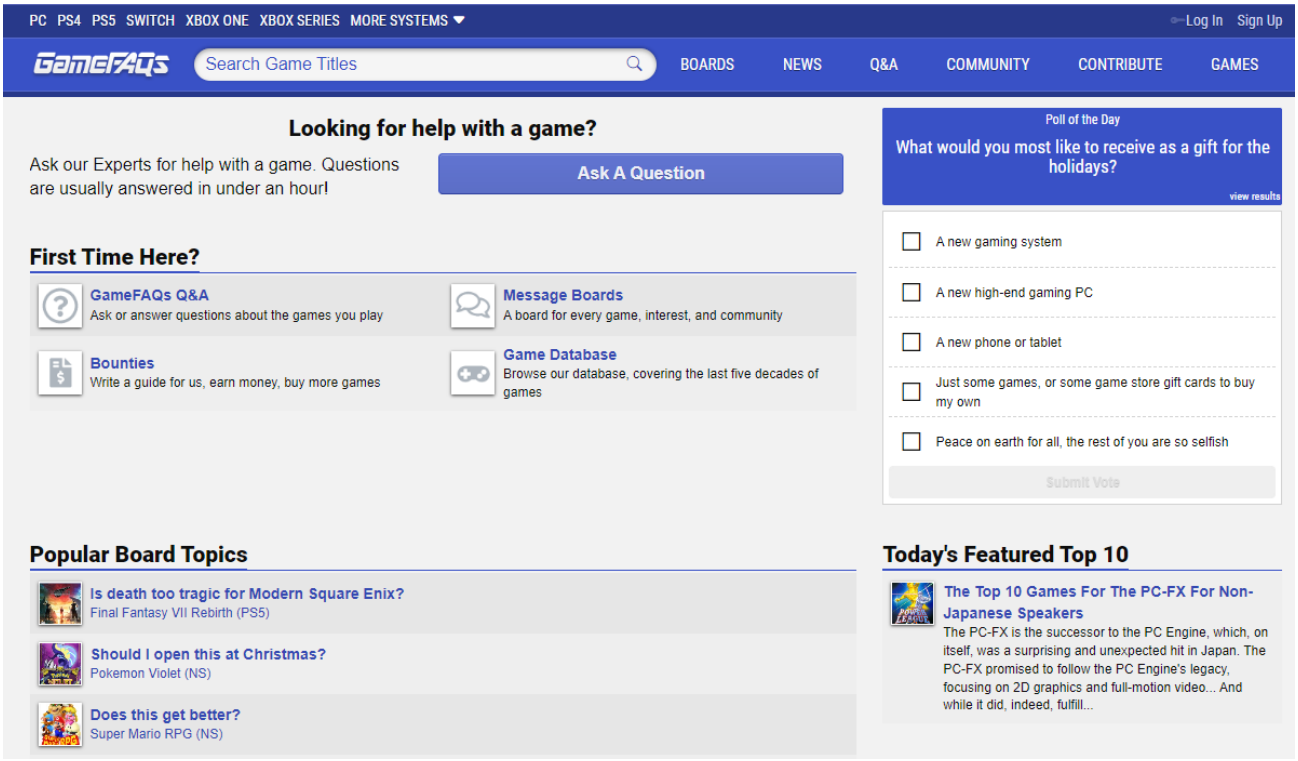

Рисунок 1.3 – Головна сторінка сайту GameFAQs

1.4 Постановка задачі

Під час виконання кваліфікаційної роботи, повинно бути створено ігровий портал, який необхідно наділити наступним функціоналом:

1) Розробити систему ролей, де кожен користувач, зможе виконувати тільки ті дії, які дозволяються його роллю.

2) Сторінка блогу, яка повинна включати питання і відповіді, які може бачити будь-який користувач. Кожне питання, повинно мати власну сторінку, із можливістю залишити відповідь.

3) Сторінка з іграми, яка міститиме підв’язку до інших сутностей, таких як блог чи запитання. Базуючись на цих зв’язках, користувач повинен одразу на сторінці мати змогу, відкрити набір пов’язаних сутностей.

4) Система рейтингу, яка автоматично видає ролі, для користувачів. В залежності від ролі, користувач повинен мати змогу створювати різні сутності – чим вища роль, тим важливіша для порталу сутність.

5) Можливість ставити запитання до блогів і коментувати ігри. Можливість давати відповіді на поставленні запитання, таким чином створюючи тред.

6) Можливість фільтрації сутностей на основі різних фільтрів. Можливість обрання фільтрів при створенні сутності.

7) Добавити WYSIWYG редактор для полів, для того, щоб користувачі, могли із легкістю створювати наповнення для своїх блогів, запитань чи відповідей.

8) Можливість авторизації на сайті за допомогою соціальних мереж (Google і Steam).

9) Розробити адаптивний користувацький інтерфейс для всіх вищевказаних функцій. Адаптивний інтерфейс, повинен підтримувати різні типи пристроїв, такі як смартфон чи то планшет.

2 ПРОЄКТУВАННЯ ІГРОВОГО ПОРТАЛУ

2.1 Опис предметної області

Веб-портал – це спеціально розроблений веб-сайт, який часто слугує єдиною точкою доступу до інформації. Його також можна вважати бібліотекою персоналізованих та категоризованих даних. Веб-портал допомагає в пошуковій навігації, персоналізації, сповіщеннях та інтеграції інформації, а також часто надає такі функції, як управління завданнями, спільна робота, бізнес-аналітика та інтеграція додатків. Веб-портали також відомі просто як портали.

Веб-портали часто надають організаціям і компаніям специфічний вигляд і відчуття, а також контроль доступу та процедури. Вони доступні з різних платформ, таких як персональні комп'ютери, смартфони та інші електронні пристрої. Ключовими характеристиками веб-порталу є доступ до даних, особистий контент, транзакції, безпека, опублікований контент і пошук. Він здатний представляти інформацію, орієнтовану на користувача. Він також може дозволити користувачам добровільно персоналізувати інформацію, представлену на порталі [10].

Веб-портал може обробляти як структуровану, так і неструктуровану інформацію. Для користувачів він забезпечує простоту навігації, а для організацій – значну економію коштів, підвищення продуктивності та спосіб побудови довгострокових відносин з користувачами. Веб-портал може полегшити сповіщення та багатоканальну узгодженість. Він забезпечує єдиний вхід та інтеграцію з іншими додатками і системами за потреби. Він також може інтегрувати і підтримувати певний тип додатків, наприклад, підтримку електронної комерції, бізнес-аналітику або додаток постачальника послуг.

Ідея веб-порталу полягає в тому, щоб запропонувати якомога більше інтернет-послуг, щоб залучити й утримати якомога більше користувачів. Існує два типи веб-порталів:

- горизонтальні портали надають користувачам можливість переглядати та збирати дані з декількох додатків в одному вікні. Горизонтальні веб-портали надають інформацію для всієї мережі, а не забезпечують персоналізований доступ для конкретної аудиторії. Такі портали, як портали державних послуг і портали управління знаннями, є прикладами горизонтальних веб-порталів. Також сюди входять і ігрові портали.
- вертикальні – в основному, дані портали зосереджені на конкретних бізнес-функціях або додатках, таких як бухгалтерський облік і фінанси, CRM, управління послугами або корпоративне управління. Вертикальні веб-портали дозволяють користувачам з обох сторін учасникам і відвідувачам – переглядати, редагувати і робити свій внесок у процеси в рамках конкретної програми. Прикладами вертикальних веб-порталів є студентські портали, портали для працівників або портали для пацієнтів.

Веб-портали мають п’ять ключових переваг, завдяки яким, багато компаній розробляють такі типи веб-сайтів:

1. Охоплення ширшої аудиторії шляхом створення позитивного іміджу бренду. Дуже часто головна сторінка порталу, містить велику кількість корисної інформації, адже, такі типи сайтів мають багато інформації, тому головна сторінка доволі насичена. Також вона визнає тип порталу, наприклад на порталі коледжу або школи користувачі очікують побачити освітні ресурси, в той час як ігровому порталі повинні висвітлюватись ігрові новини, гарячі ігрові дискусії та очікувані ігри.

2. Доступ з будь-якого місця для підвищення продуктивності бізнесу. Портал компанії забезпечує можливість доступу до необхідної інформації з будь-якого місця та пристрою, що сприяє підвищенню продуктивності бізнесу. Його універсальність і доступність незалежно від місця розташування користувача. Цей

зручний доступ до ресурсів компанії стимулює оперативність та ефективність у вирішенні завдань, сприяючи усвідомленню та розвитку бізнесу.

3. Покращена комунікація може призвести до збільшення бізнесу. Ефективна комунікація виступає основною складовою успішного управління бізнесом. Здатність взаємодіяти з індивідами як всередині, так і поза межами організації, є ключовим аспектом для досягнення оптимального функціонування підприємства.

4. Зручний інтерфейс завдяки унікальному UX. Організація великого обсягу інформації та послуг є критичним фактором у сучасному світі, однак у цьому контексті часто залишається поза увагою аспект, що стосується користувацького досвіду. Підтримка високоякісного користувацького досвіду полягає у наданні інформації, що є корисною, доступною, цінною та легкою для пошуку в контексті, що відповідає потребам та очікуванням кінцевих користувачів.

5. Покращення обслуговування клієнтів за допомогою клієнтського порталу. Послуги, які надаються через клієнтські портали, спрощують весь процес взаємодії з клієнтами. Ці платформи розроблені з метою поліпшення якості обслуговування клієнтів.

2.2 Функціональна структура сайту

Функціональна структура веб-проєкту обумовлює певні аспекти створення та подальшого проєктування веб-сторінок та загальних зв’язків між ними. Уся структура позначає чітку взаємодію усіх наявних на сайті елементів, незважаючи на їх розташування.

На сайті, користувачі можуть виконувати різні дії, переглядати різні сторінки та взаємодіяти із різними об’єктами, маючи різні права доступу. Для того, аби зрозуміло

описати всі можливі варіанти дій на сайті, було спроєктовано діаграму варіантів використання. На рисунку 2.1 можна переглянути діаграму прецедентів.

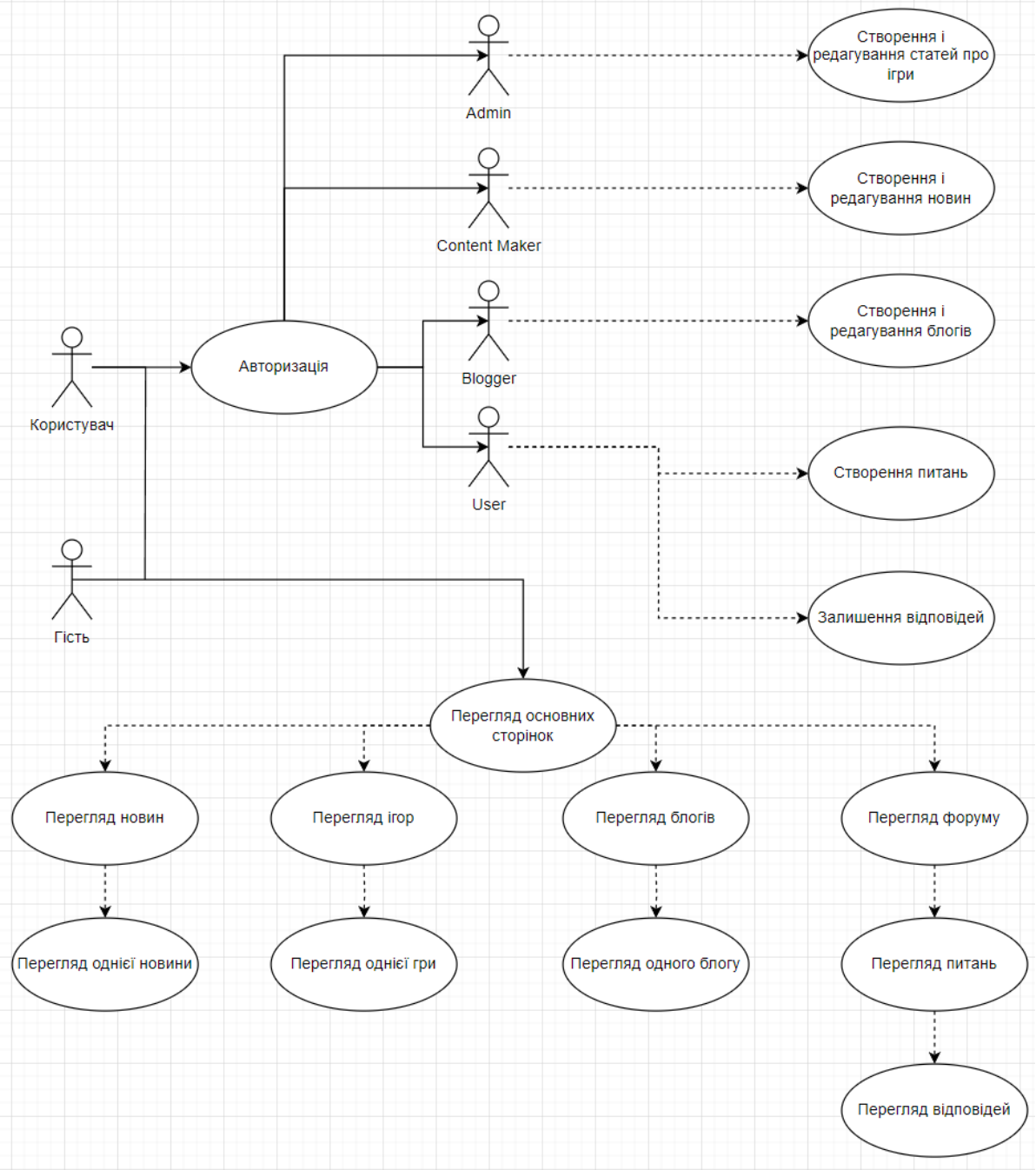

Рисунок 2.1 – Діаграма варіантів використання

Нижче буде наведена діаграма класів (рисунок 2.2), що повністю позначає увесь процес роботи веб-сайту та усіх його аспектів. Ця діаграма складається з окремих

класів, кожен з яких виконує те чи інше завдання, проте усі класи пов’язані між собою, адже, кожна сутність пов’язана із користувачами та іграми.

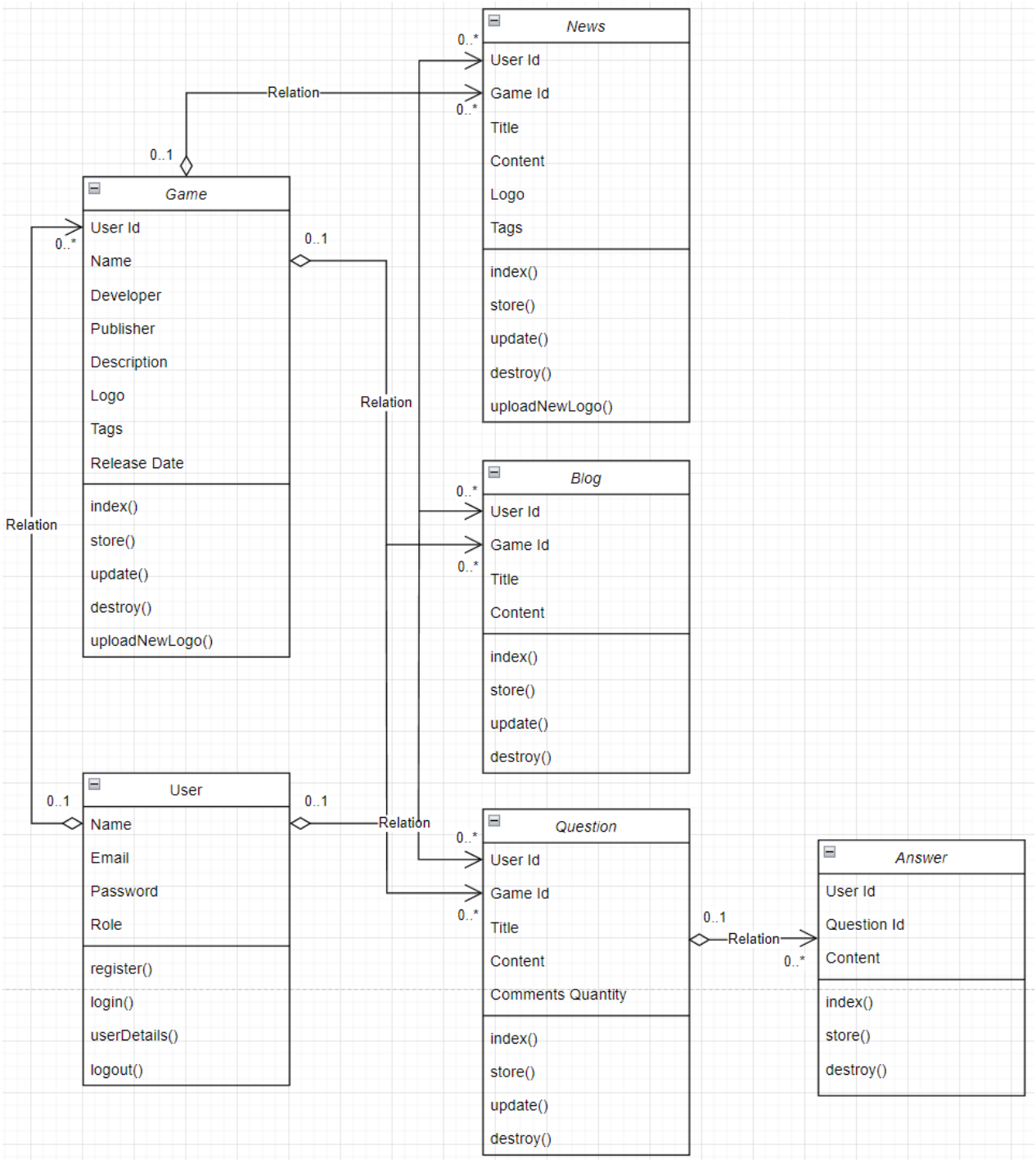

Рисунок 2.2 – Діаграма класів

2.3 Проєктування користувацького інтерфейсу та дизайну

Процес проєктування користувацького інтерфейсу (UI) представляє собою етап, коли фахівці в області дизайну зосереджують свою увагу на розробці інтерфейсу для програмного забезпечення або комп'ютерних пристроїв, наголошуючи на аспектах зовнішнього вигляду і стилю. Основною метою дизайнерів є створення інтерфейсу, який максимально спрощений у використанні та зрозумілий для пересічних користувачів. Концепція інтерфейсу користувача включає в себе графічні інтерфейси користувача (GUI) та інші форми інтерфейсів, такі як голосове керування.

У цьому випадку, було спроектовано графічний інтерфейс користувача (GUI), який дає можливість користувачам взаємодіяти з візуальними елементами на цифрових панелях управління (комп'ютер, смартфон і т. д.).

Ігровий портал був спроєктований таким чином, щоб звичайні користувачі, могли знайти для себе корисну інформацію на доступних сторінках. Якщо ж на сторінках ігор, блогів чи в новинах немає необхідної інформації, користувач можем звернутись до форуму із конкретним запитанням, що його цікавить.

Вперше відвідавши сайт, користувач звичайно ж, натрапить на головну сторінку, де можна знайти багато цікавої інформації. З цієї сторінки, користувач може рухатись до будь яких інших сторінок порталу, для пошуку затребуваної інформації. Також на головній сторінці виводиться найактуальніші блоги, новини та ігрові огляди. На рисунку 2.3 зображено сторінку головну порталу.

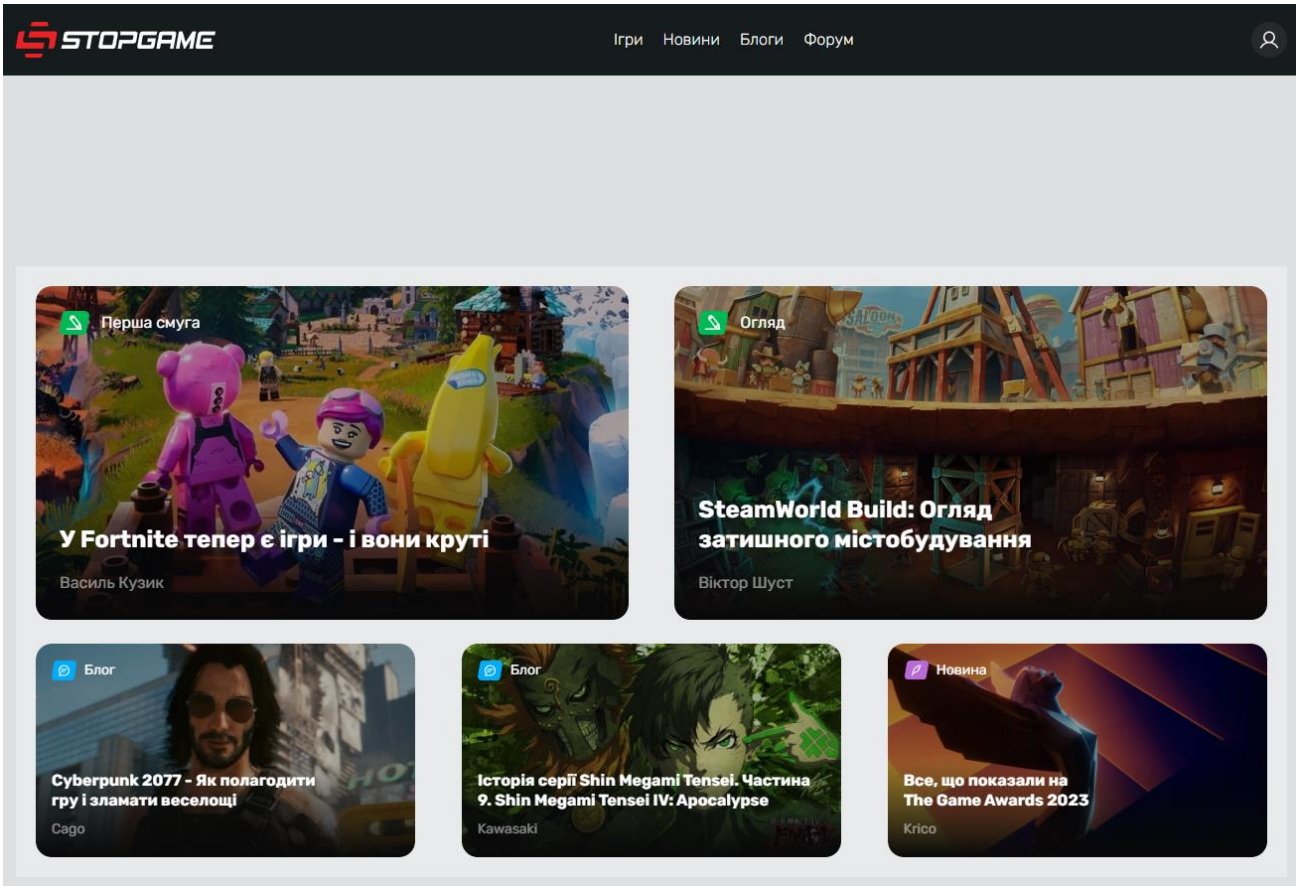

Рисунок 2.3 – Головна сторінка порталу

Рухаючись з головної сторінки, найцікавішою для пересічного користувача буде сторінка ігор, адже, на ній можна знайти інформації про всі ігри, що можуть зацікавити. За замовчуванням ігри відображаються таким чином, що показуються найкращі за 30 днів. Також на цій сторінці є каталог із описами всіх ігор. Також на сторінці доступні фільтри, за якими користувачам може відбирати ігри, відповідно до своїх інтересів. На рисунку 2.4 зображено сторінку списку ігор.

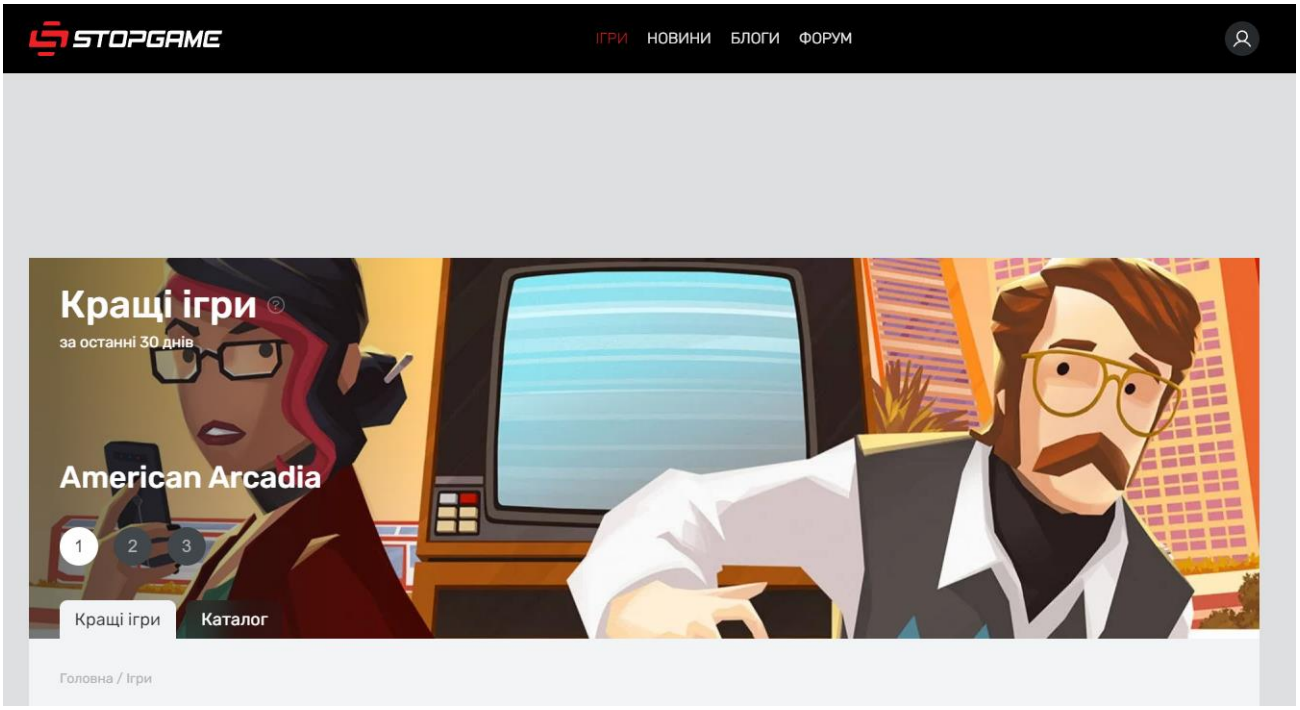

Рисунок 2.4 – Сторінка зі списком ігор

Обравши цікаву для себе гру, користувач може відкрити сторінку із описом цієї гри та великою кількість додаткової інформації, такої як, дата виходу гри, хто її розробив, опублікував та на яких платформах вона може запускатись, також ознайомитись, під критерії яких жанрів вона підпадає. На сторінці є окремі вкладки із додаткової відібраними новинами, блогами та запитаннями пов’язаними саме із цією грою. Ці вкладки є фактично фільтром, за яким користувач може обрати те, що його цікавить та рухатись до наступних сторінок порталу. На рисунку 2.5 можна побачити сторінку однієї гри із доступними вкладками.

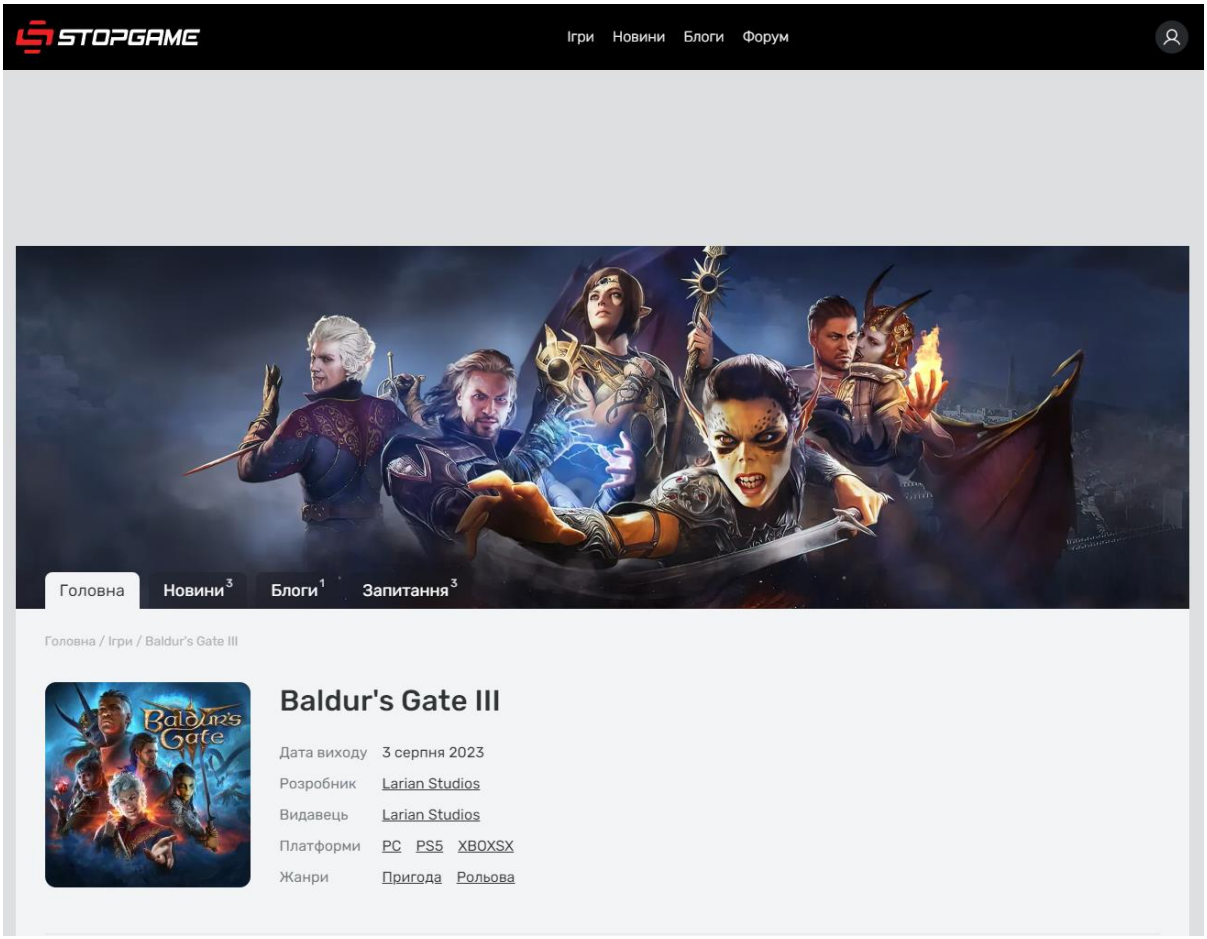

Рисунок 2.5 – Сторінка гри

Велика кількість користувачів, безумовно, також мають бажання прийняти участь в житті порталу, адже, практично весь вміст буде заповнятись пересічними користувачами. Хтось хоче відвідати форум та залишити запитання, хтось бажає писати довгі та інформативні блоги, а хтось створювати новинні інсайди та залучати увагу людей. Тільки зареєстровані користувачі можуть створювати контент сайту, в залежності від їхньої ролі. Аби здійснити ці дії, відвідувач сайту, може скористатись модальним вікном, яке одночасно дозволяє як авторизуватись, так і зареєструватись в системі, до того ж це можна зробити за допомогою двох доступних соцмереж Google – який має практично кожен і Steam – більшість зацікавлених в іграх людей мають і його теж. Дане вікно зображене на рисунку 2.6.

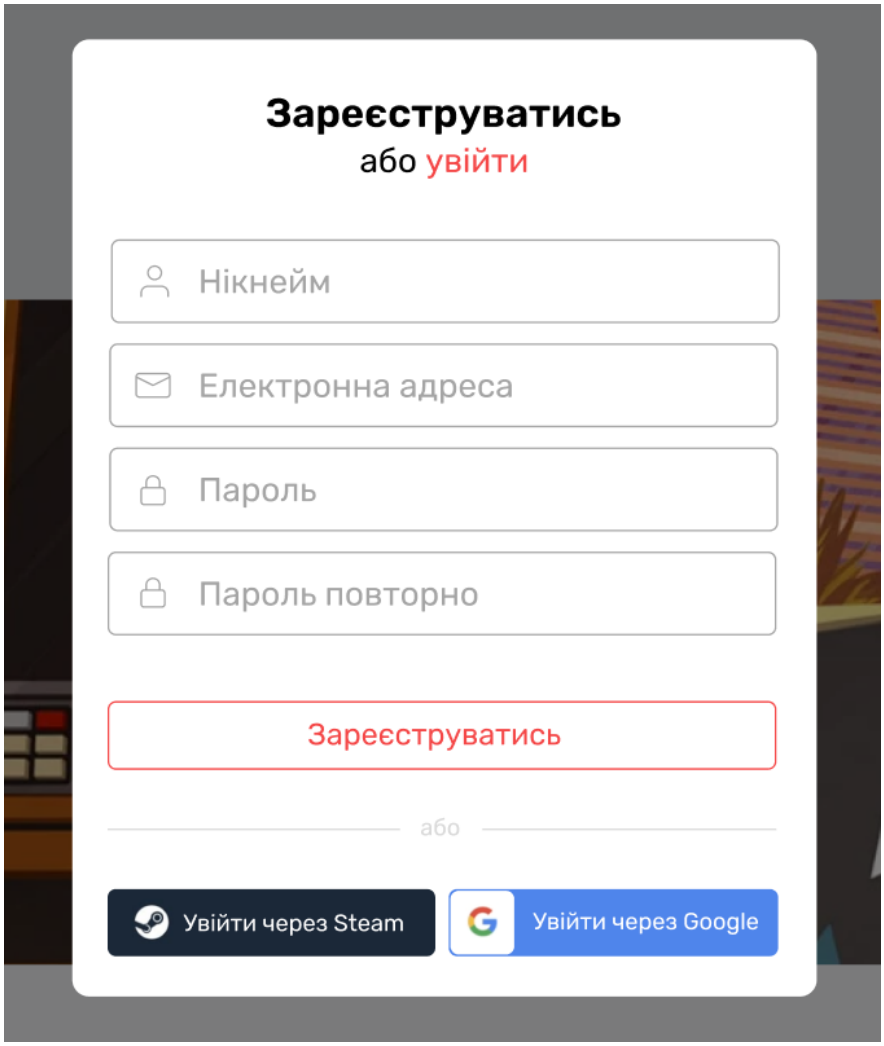

Рисунок 2.6 – Форма реєстрації та авторизації

Також, неабиякий інтерес користувачів, викликають блоги, що розповідають про ті речі, які турбують звичайних користувачів, адже, це роблять користувачі для користувачів. Побувши певний час активним на форумі, користувач може отримати доступ до створення блогів, ділячись інформації, порадами чи просто своїми думками щодо ігор чи останніх ігрових новин. Блоги можуть бути, як прив’язані до певної гри, так і бути окремими сутностями. На рисунку 2.7 можна побачити зображення сторінку одного блогу.

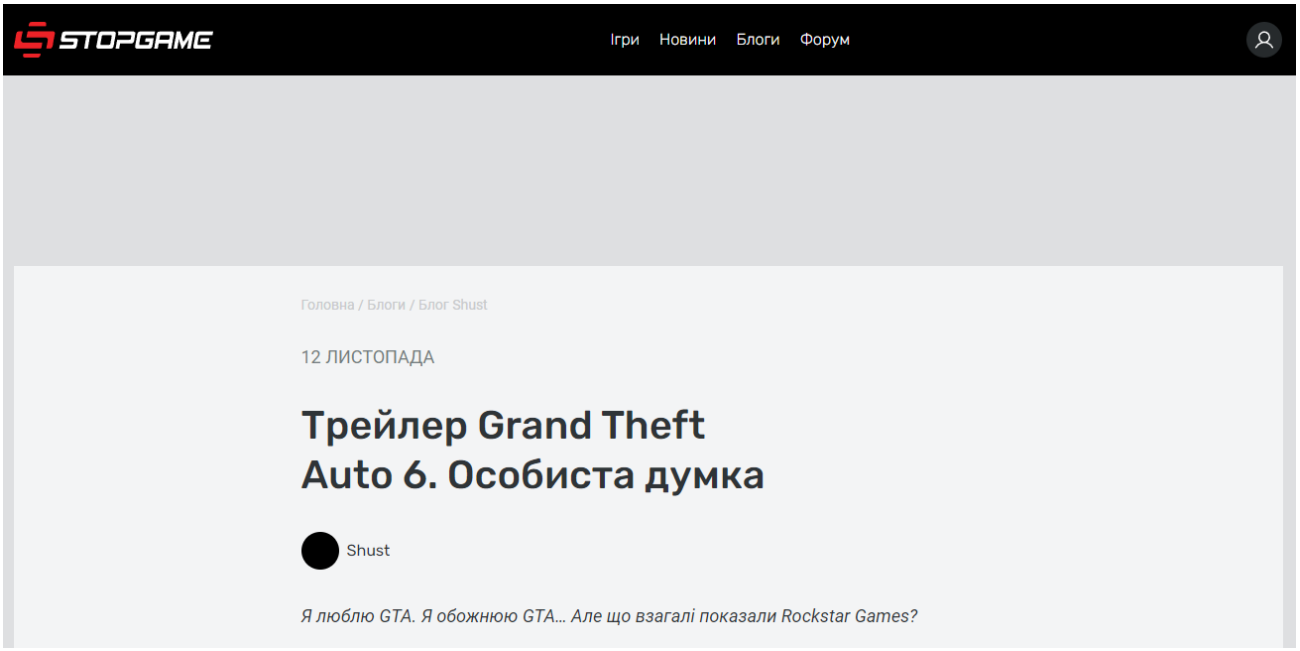

Рисунок 2.7 – Сторінка одного блогу

Всі ці основні сторінки, що були спроєктовані, також мають підкріплятись, великою кількістю додаткового функціоналу, який є обов'язковим для функціонування сайту та підтримки роботи, вище представлених сторінок. Наприклад, такими функціями є: форми додавання сутностей (новини, запитання), спливаючі повідомлення, адаптація наявних сторінок під різні пристрої та багато інших, мілких деталей.

2.4 Обґрунтування технологій та засобів реалізації

У процесі розробки веб-сайтів, використовуються різноманітні підходи та технології для їх створення. Деякі сайти реалізуються виключно за допомогою певних мов програмування, в той час як інші використовують фреймворки як для клієнтської, так і для серверної частини. Також для збереження та пошуку даних, можна обирати велику кількість різноманітних пошукових рушіїв.

Для клієнтської частини, для обробки даних на якій використовуються JS, також є ряд відомих фреймворків, таких як:

- Vue JS;
- Angular;
- React.

Також на клієнтській частині сайту, є серверна загрузка даних, що виконується на за допомогою JS, проте це відбувається, ще до ініціалізації базових елементів JS. Це такі фреймворки, як:

- Nuxt JS;
- Next JS;
- Express.

Серед пошукових рушіїв, також існує велике різноманіття створених рішень, проте до уваги варто брати лише найкращі з них, що дійсно дають очікувані результати, а саме це:

- ElasticSearch;
- Algolia;
- Solr.

Перед початком розробки сайту, необхідно обрати фреймворки для клієнтської частини сайту. Базовою технологією, яка відповідає за відображення даних та взаємодію з користувачем є Vue JS, який і був обраний поміж інших фрейморків.

Vue JS представляє собою прогресивний фреймворк для JavaScript, що використовується для розробки веб-інтерфейсів та односторінкових застосунків. Цей фреймворк знайшов застосування не лише у веб-інтерфейсах, але й у створенні настільних та мобільних додатків за допомогою Electron framework. Завдяки розширенню HTML та основам JS, Vue швидко набув популярності серед розробників інтерфейсів. Його активно використовують такі великі компанії, як Adobe, Gitlab і Xiaomi, що свідчить про його популярність і ефективність [11].

Vue JS має 5 ключових переваг:

1. Vue JS, крім прогресивного підходу, відомий як "поступово адаптований". Це означає, що як сам фреймворк, так і програми, які його використовують, розробляються від початку. Однією з переваг цього підходу є легкість початку роботи. Основна бібліотека Vue JS базується на CSS, HTML і JavaScript - основних складових будь-якого веб-проєкту веб-розробки.

2. Функціональні можливості, які можуть знадобитись у Vue JS для додаткового функціоналу, вже охоплені офіційними бібліотеками фреймворку. Після встановлення основ, ймовірно, буде необхідно налаштувати маршрутизацію на сайті і управління даними. Для досягнення цих цілей використовуються дві бібліотеки, які майже завжди присутні на сайтах, що використовують Vue JS: Vue Router і Vuex.

3. Легкий пакет Vue JS, розмір якого складає лише 21Кб, дозволяє йому працювати швидше, ніж його конкуренти. Це можливо завдяки його віртуальному DOM, який ефективно прискорює процес відображення сайту. Об’єктна модель документа (DOM) є інтерфейсом програмування, що моделює структуру тексту у HTML і XML. Віртуальний DOM, як реальне відображення, але із можливістю синхронізації змін, дозволяє оптимізувати роботу з ним.

4. Оволодіння Vue JS не потребує глибоких знань в області бібліотек та варіантів JavaScript. Тут ідеально підходять класичні CSS, HTML і JavaScript.

5. Vue JS, як і більшість мов та фреймворків з відкритим вихідним кодом, користується активною підтримкою та відданою спільнотою. Понад 132 000 учасників на GitHub використовують Vue JS.

React представляє собою бібліотеку для розробки інтерфейсу користувача, засновану на мові програмування JavaScript. Управління цією бібліотекою здійснюється компанією Facebook спільно з відкритою спільнотою розробників. Незважаючи на те, що React - це бібліотека, а не мова програмування, вона має широке застосування у веб-розробці. Перший випуск цієї бібліотеки відбувся у травні 2013

року, і на сьогоднішній день вона є однією з найбільш популярних інтерфейсних бібліотек для розробки веб-додатків.

React надає різноманітні розширення для підтримки всієї архітектури додатків, таких як Flux і React Native, що виходять за межі базового функціоналу інтерфейсу [12].

React, на рівні із Vue JS, також має свої переваги, а саме:

1. Створення динамічних веб-додатків з використанням React виявляється менш трудомістким завдяки зменшенню обсягу коду та наданню більш широкого функціоналу. У порівнянні з чистим JavaScript, де розробка може швидко ускладнюватися, React пропонує більші можливості, зберігаючи простоту програмування.

2. React використовує віртуальний DOM, що призводить до підвищення продуктивності веб-додатків. Цей механізм сприяє прискоренню роботи програм, оскільки дозволяє ефективніше взаємодіяти з реальним DOM, що в свою чергу сприяє покращенню швидкодії веб-додатків.

3. Компоненти в React є фундаментальними структурними елементами, які складають будь-яку програму. Зазвичай програма у React складається з кількох таких компонентів, кожен з яких має свою внутрішню логіку та елементи керування. Ця концепція дозволяє повторно використовувати ці компоненти в різних частинах додатку, що сприяє значному скороченню часу розробки програми.

Всі перераховані переваги React в повній мірі також притаманні Vue JS. Однак Vue JS вирізняється більш простим порогом входження і написання коду, що дозволяє працювати з ним легше, порівняно з React. Підхід до компонентів у Vue JS видається більш очевидним, а сполучення компонентів між собою краще організоване на рівні самого фреймворку.

Angular – це інтерфейсний фреймворк з відкритим вихідним кодом, побудований на JavaScript, і використовується для створення користувацьких програм, що базуються на HTML, CSS і Typescript. Цей фреймворк спрямований на

полегшення розробки динамічних веб-додатків шляхом застосування архітектурного шаблону MVC (Model-View-Controller). AngularJS зараз функціонує як основний інструмент для розробки, хоча не отримує нових оновлень та розробок [13].

Angular має кілька власних переваг:

1. Надійність платформи: Angular отримує підтримку від Google, що надає впевненість у стабільності та довгостроковій підтримці.
2. Використання віртуальної машини JavaScript: Angular використовує потужний механізм, що перетворює шаблони в оптимізований код, що сприяє швидкому завантаженню.
3. Двостороннє прив'язування даних: Модулі та компоненти Angular мають двостороннє прив'язування даних, що робить код зрозумілим для модульного тестування та сприяє послідовності його структури.
4. Підтримка PWA: При застосуванні Angular для розробки прогресивних веб-додатків (PWA), можна створювати програми, що працюють на різних платформах, включаючи мобільні пристрої, що розширює його можливості в різноманітних сценаріях використання.

На базі обраного Vue JS, як фреймворку для клієнтської частини, необхідно обрати фреймворк для серверної частини клієнтської сторони сайту, який відповідатиме за роботу за загрузку даних та попередній рендер таких даних на сторінці. Основним кандидатом на таку роль, був Nuxt JS.

Nuxt.js представляє собою високорівневий фреймворк, заснований на Vue JS, який надає розширені можливості рендерингу на стороні сервера. Цей інструмент поєднує потужність Vue JS з функціоналом серверного рендерингу, що зробило його популярним серед розробників. Nuxt.js забезпечує можливість створювати різноманітні види додатків, включаючи клієнтські, статичні та монолітні за допомогою Vue JS.

Цей серверний фреймворк рендерингу спрощує складні налаштування, пов'язані з управлінням асинхронними даними, проміжним програмним забезпеченням та

маршрутизацією, забезпечуючи абстракцію від цих складнощів. Він сприяє структуруванню додатків Vue JS за допомогою галузевої архітектури, що робить можливим розробку як простих, так і корпоративних додатків на Vue JS [14].

Виділяють наступні переваги Nuxt JS:

1. Nuxt.js, як потужний фреймворк Vue JS, пропонує спрощену розробку додатків шляхом надання попередньо налаштованих параметрів. Його вбудовані можливості, такі як серверний рендеринг, маршрутизація та управління станами, дозволяють розробникам зосередитися на логіці додатку, уникнувши витрат часу на складні конфігурації.

2. Механізм серверного рендерингу (SSR) у Nuxt.js сприяє поліпшенню пошукової оптимізації (SEO), створюючи початковий HTML на сервері для кожної сторінки перед його відправленням клієнту. Це сприяє більш ефективному скануванню та індексації контенту пошуковими системами.

3. Фреймворк реалізує автоматичне розділення коду, завантажуючи лише необхідний JavaScript для кожної сторінки, що сприяє покращенню продуктивності та зменшенню часу завантаження.

4. Nuxt.js легко інтегрується з екосистемою Vue JS, підтримуючи інструменти та бібліотеки, такі як Vuex для керування станом та Vue Router для маршрутизації, спрощуючи роботу розробників.

5. Фреймворк надає універсальні можливості конфігурації, дозволяючи розробникам налаштовувати його відповідно до вимог проекту, незалежно від налаштувань веб-пакетів, проміжного програмного забезпечення чи керування макетами.

Next.js – це фреймворк, спеціалізований на розробці комплексних веб-додатків на базі React. Використовуючи React-компоненти для створення користувацьких інтерфейсів, Next.js додає додаткові можливості та оптимізації, розширюючи функціонал.

Усередині Next.js автоматично абстрагуються та налаштовуються інструменти, необхідні для ефективної роботи з React, такі як збірка, компіляція та інші. Це спрощує процес розробки, дозволяючи зосередитися на створенні додатку, уникнувши витрат часу на технічну конфігурацію [15].

Next.js має потенціал сприяти як індивідуальним розробникам, так і великим командам, допомагаючи створювати інтерактивні, динамічні та ефективні React-додатки.

До ключових переваг Next JS, відносять:

1. Рендеринг на стороні сервера (SSR): Next.js використовує SSR з метою генерації та доставки повністю відформатованого HTML безпосередньо до браузера клієнта. Це сприяє прискоренню завантаження сторінок та поліпшенню взаємодії з користувачем, особливо важливо для ресурсоємних додатків, таких як платформи електронної комерції.

2. SEO-переваги: Використання Next.js сприяє покращенню SEO, оскільки допомагає створювати оптимізовані для пошукових систем сторінки. Це призводить до покращення індексації та видимості у результатах пошуку. Такі компанії, як Netflix та Airbnb, спостерігали позитивні зміни в органічному трафіку та позиціях після використання Next.js.

3. Інтегрована генерація статичних сайтів (SSG): Механізм SSG у Next.js виконує попередній рендеринг сторінок під час їх створення, що дозволяє зменшити навантаження на сервер та покращити швидкість обробки. Це особливо корисно для сайтів з великим обсягом даних або з рідкісними оновленнями контенту, наприклад, блогів або сторінок опису продуктів.

4. Покращений досвід розробників: Next.js надає розробникам зручне середовище для роботи, з такими функціями, як автоматичне розділення коду, гаряче перезавантаження для оновлення коду в режимі реального часу та прості варіанти розгортання. Це сприяє спрощенню процесу розробки та збереженню часу та зусиль розробників.

5. Більша гнучкість та кастомізація: Next.js дозволяє розробникам створювати власні маршрути, інтегруватися з різними бібліотеками та технологіями, реалізовувати маршрути API для безсерверних функцій та налаштовувати поведінку сервера. Це надає розробникам можливість налагоджувати рішення відповідно до вимог проекту.

Express – це ще один лаконічний веб-фреймворк, який виступає чудовою альтернативою Nuxt JS, просто для його інтеграція із Vue JS, є більш складною, адже, він побудований як шар поверх Node.js, призначений для забезпечення набору надійних функцій для ефективного управління серверами та маршрутами. Його основною концепцією є послідовне виконання функцій проміжного програмного забезпечення, кожна з яких відповідає за певний функціонал. Express не обмежується одним конкретним шаблоном дизайну або структурою папок, що дозволяє розробникам використовувати його у різних сценаріях розробки з відмінними підходами [16].

Серед його основних переваг можна чітко визначити наступні:

1. Express є вільним із відкритим вихідним кодом, пристосованим до ліцензії MIT, що робить його доступним для розробників безкоштовно. Це забезпечує економію фінансових ресурсів для інструментів та стимулює глобальну співпрацю у розвитку, прикладами якої є проекти, такі як SaaS Boilerplate та Builder Book.

2. Основою Express є мінімалістичний фреймворк, який сприяє високій продуктивності веб-додатків, що є критичним для забезпечення найкращого користувацького досвіду. Його експертно написаний код спрощує операції та надає можливість подальшої оптимізації, такої як оптимізація зображень для поліпшення продуктивності та швидкості завантаження.

3. Виділяється його мінімалістичний та простий у використанні підхід, подібний до Node.js, який пропонує простоту та ефективність, роблячи процес розробки більш зручним та ефективним, що веде до заощадження часу розробників.

4. Express також відомий своєю здатністю ефективно створювати API, що важливо для розробників серверної частини, навіть тих, хто не володіє великим досвідом у JavaScript. Це забезпечує швидке нарощування та ефективну розробку серверної частини.

5. Express прискорює розробку додатків, надаючи готові функціональні можливості, бібліотеки та проміжне програмне забезпечення. Це допомагає у виконанні рутинних завдань, зменшує складність кодування та надає сучасні можливості для створення потужних, адаптивних додатків. Підтримка спільноти розробників надає безліч ресурсів для навчання та вирішення проблем у процесі розробки.

Надзвичайно важливим для сучасних сайтів є швидкість їхньої роботи, загрузки даних, опрацювання та вивід даних для користувача. Надсилання запитів до серверної частини через REST API виконується занадто довго, тому існують різні пошукові рушії, які спрямовані на пришвидшений пошук даних. Дані в собі вони містять за принципом нерелятивних баз даних.

Першим пошуковим рушієм, що і був обраний, як основа є ElasticSearch, який представляє собою систему розподіленого пошуку та аналітики з відкритим вихідним кодом, побудовану на основі Apache Lucene та написану на мові програмування Java. Починаючи як розширення відкритого фреймворку пошуку Lucene, ElasticSearch швидко розвинувся, додавши можливість горизонтального масштабування індексів Lucene.

Ця система здатна зберігати, шукати та аналізувати величезні обсяги даних швидко та практично в реальному часі, надаючи відповіді на запити майже миттєво. ElasticSearch досягає високої продуктивності завдяки використанню індексування замість прямого пошуку у тексті. Він використовує структуру, базовану на документах, а не на таблицях та схемах, і наділяється розширеним набором REST API для зберігання та пошуку даних. Фактично, ElasticSearch можна уявити як сервер, який опрацьовує запити у форматі JSON та повертає відповідні JSON-дані [17].

Особливості та переваги ElasticSearch включають в себе:

1. Висока продуктивність: Elasticsearch вражає швидкістю завдяки розподіленим інвертованим індексам і кешуванню запитів. Пошуки, які в традиційних базах даних SQL забирають понад 10 секунд, виконуються менш ніж за 10 мілісекунд, надаючи ефективне та швидке отримання інформації.

2. Легка масштабованість: Розподілена архітектура Elasticsearch дозволяє легко розширювати її для підтримки тисяч серверів та зберігання величезних обсягів даних. Незалежно від масштабу проекту, процес масштабування відбувається майже автоматично, що зменшує необхідність у великому плануванні та робить розширення безперервним.

3. Розподілена архітектура: Крім пошукових операцій, Elasticsearch ефективно керує обробкою великих обсягів даних завдяки поділу індексів на шарди та створенню декількох реплік. Це забезпечує надійність обробки інформації, а автоматичні операції маршрутизації та ребалансування підвищують ефективність.

4. Документно-орієнтована база даних: Elasticsearch використовує JSON для серіалізації документів, що робить його ідеальним для зберігання складних об'єктів. Сумісність з JSON та простота сприяють підвищенню продуктивності та зручності використання.

5. Відсутність схеми: Elasticsearch не потребує строго визначених властивостей даних та впорядкованих схем. Він автоматично визначає типи даних, індексує записи і залишається гнучким. При додаванні нових властивостей до документів вони автоматично включаються у визначення, забезпечуючи гнучкість та адаптивність без суворих вимог до схеми.

Основним конкурентом для ElasticSearch виступає Algolia, яка має дещо ширший функціонал, проте є платною, через що, є найменш привабливим рішенням для малих проєктів.

Algolia – це платформа для пошуку, яка надає хостингові послуги для інтеграції пошукової функціональності у веб-сайти та додатки за допомогою API. Вона

спрямована на забезпечення швидкого та релевантного пошуку відразу з моменту впровадження, оскільки це ключові характеристики ефективної роботи пошукової системи для кінцевих користувачів [18].

Algolia містить велику кількість функціоналу, проте необхідно зазначити наступні преваги:

1. Algolia – це пошукова система, спроектована для досягнення високої продуктивності та масштабованості, що забезпечує швидкість обробки запитів і надійність. Інфраструктура та оптимізації, застосовані в Algolia, сприяють оперативному виконанню пошукових запитів, що позитивно впливає на загальний досвід користувачів.

2. Algolia вирізняється простотою інтеграції з різними платформами і фреймворками, що спрощує впровадження пошукових функцій у вже існуючі веб-сайти та додатки. Наявність інструментів для розробників, широка документація, SDK та плагіни полегшують процес інтеграції та налаштування.

3. Гнучкість і масштабованість є іншими ключовими перевагами Algolia. Її інфраструктура легко масштабується для відповіді на зростаючі потреби у пошуку, навіть для великих корпоративних додатків, не втрачаючи продуктивності.

4. Послуга також пропонує розширені можливості та налаштування, такі як толерантність до помилок у написанні, обробка синонімів та персоналізовані рекомендації, що дозволяє підприємствам отримувати більш релевантні результати пошуку, хоча це може вплинути на вартість сервісу.

5. Algolia акцентує увагу на безпеці та надійності, забезпечуючи захист даних та стабільну інфраструктуру, що допомагає уникнути простоїв і витоків інформації, що є важливим для підприємств.

Apache Solr – це безкоштовна система пошуку з відкритим вихідним кодом, побудована на основі знаменитої бібліотеки Apache Lucene. Lucene, визнана по всьому світу, виступає основою для створення Solr. Проте, Solr переходить за межі звичайної пошукової системи, надаючи різноманітні можливості.

Система Solr використовується як база даних NoSQL, що працює на основі документів, і може обробляти транзакції. Це робить Solr відмінним вибором для зберігання даних, а також як сховище ключ-значення. Його гнучкість і різноманітність застосування роблять його корисним і для інших завдань, крім пошуку, розширюючи сферу його застосування в інформаційному пошуку та обробці даних [19].

Оскільки, Solr є дещо ширшим за інші, то ряд його переваг є наступним:

1. Apache Solr - це потужна система повнотекстового пошуку, яка пропонує ряд розширених можливостей для ефективного пошуку різних типів даних. Він забезпечує багатофункціональний пошук, включаючи пошук за полями, булеві та фразові запити, а також можливості роботи з перестановкою слів та правописом. Такий широкий спектр можливостей робить його ідеальним для різноманітних завдань пошуку та фільтрації даних.

2. Система Solr має високу масштабованість та гнучкість, що дозволяє легко масштабувати та налаштовувати його для різних конфігурацій та потреб. За допомогою Apache ZooKeeper, вона автоматизує процеси реплікації, балансування навантаження та відновлення, що робить її ефективною та надійною для великих обсягів даних.

3. Розширена архітектура плагінів дозволяє легко інтегрувати додаткові функціональності для оптимізації пошукових процесів та розширення можливостей системи.

4. Solr активно забезпечує безпеку даних, надаючи можливості шифрування для трафіку, різні методи аутентифікації та авторизації, що дозволяють визначати доступ користувачів, ролей і дозволів.

5. Щодо моніторингу, Solr надає засоби для вимірювання продуктивності через JMX MBeans і HTTP API, а також підтримку комерційних інструментів з відкритим вихідним кодом для глибшого розуміння метрик, спрощуючи процес моніторингу в реальному часі.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІГРОВОГО ПОРТАЛУ

3.1 Реалізація функціоналу клієнтської частини сайту

Рушійною силою веб-сайтів завжди була функціональна частина та алгоритми, які надають користувачам можливість використовувати різноманітні функції сайту. У цьому дослідженні реалізовано значну кількість можливостей, що розширюють функціонал веб-сайту, роблячи його більш повноцінним. Це створює нелінійні шляхи для користувачів у пошуку необхідної інформації на створеному ігровому порталі.

Веб-сайт організовано як комплексна система, складові якої взаємодіють за допомогою спільних алгоритмів, що сприяє формуванню інтегрованого функціоналу. Його клієнтська частина базується на основному файлі App.vue, що містить основну верстку, яка є фундаментальною для всього проєкту.

Структура веб-сайту розділена на різні секції, де деякі з них мають статичний характер і доступні користувачеві на всіх сторінках. Ці секції включають в себе шапку (header), основний контент (main) з рядом сторінок, підвал (footer) та додаткові компоненти, такі як модальні вікна та сповіщення. Розмітка цих компонентів наведена у лістингу на рисунку 3.1.1.

```
<template>
    <div class="wrapper">
        <!-- Header section -->
        <Header />
        <!-- -->
        <!-- Pages -->
        <main class="main">
            <router-view></router-view>
        </main>
        <!-- -->
        <!-- Footer -->
        <Footer />
        <!-- -->
        <!-- Other components -->
        <Modal>
            <Auth v-if="Object.keys(user).length === 0" />
            <Create v-else />
        </Modal>
        <Notify />
        <Burger />
        <!-- -->
    </div>
</template>
```

Рисунок 3.1.1 – Верстка первинного компонента

Роль шапки (Header) полягає у навігації, забезпечуючи можливість переходу між сторінками користувачеві. Окрім основного навігаційного меню, цей компонент містить розділ користувача, де доступний виклик модального вікна для реєстрації або авторизації. Виклик вікна ініціює процес створення динамічного компонента, який відповідає за поточне вікно, що показується користувачеві. Функція заміни поточного динамічного компоненту іншим, представлена у лістингу на рисунку 3.1.2. Цей алгоритм ґрунтується на формуванні поточного вікна з локально створеного масиву об'єктів, використовуючи параметри, передані для створення самого компонента. Після цього відбувається заміна попереднього активного значення на поточне для відображення бажаного вікна.

```
// Change data in list with authorization components
changeAuthComponent (id) {
    let current = this.components.find(el => el.id === id)
    this.components.find(el => el.is_active === true).is_active = false // set false value for previous component
    current.is_active = true
    this.current = current.component // replace previous component with
}
```

Рисунок 3.1.2 – Алгоритм обробки для вибірки динамічного компоненту у модальному вікні

Користувач, викликавши модальне вікно, має опції створення нового облікового запису або входу у вже існуючий.

Компоненти авторизації та реєстрації виконують схожі функції, за винятком кроків надсилання запиту та перевірки введених даних. Валідація полів у компоненті реєстрації відбувається на кількох етапах: перевірка на пусті поля, подальша детальна перевірка відповідно до regex-шаблонів та порівняння введених паролів на ідентичність. У лістингу на рисунку 3.1.3 наведено повний алгоритм валідації набору даних, що вводить користувач для реєстрації облікового запису.

```
// Data validation
if (userData.filter(el => el.value).length === 4) {
  let errorData = userData.find(el => !this.verifyField(el.value, el.format))
    if (errorData === undefined) {
        if (this.password === this.re_password) {
            // User data from fields
            let payload = {
                name: this.nickname,
                email: this.email,
                password: this.password
            }
            // Clear error message section
            this.error = ''
        } else {
            this.error = 'Passwords don\'t match.'
        }
    } else {
        this.error = errorData.error
    }
} else {
    this.error = 'Fill the empty fields.'
}
```

Рисунок 3.1.3 – Алгоритм обробки набору даних для їх валідації

На етапі виконання валідації настає процес надсилання запиту для додавання нового користувача у базу даних, за яким наступає оновлення списку існуючих облікових записів та можливість авторизації. Відправлення запиту супроводжується викликом компоненту сповіщення для інформування користувача. У лістингу на рисунку 3.1.4 наведено процес виклику запиту та наступні дії з використанням сторонніх компонентів.

```
this.register(payload)
  .then(response => {
     this.clean()
     bus.$emit('change-auth-component', 'login')
     bus.$emit('show-notify', { msg:response.data.message, type: true })
   })
   .catch(error => {
      this.clean()
      bus.$emit('show-notify', { msg: error.response.data.message, type: false })
  })
```

Рисунок 3.1.4 – Відправка запиту для реєстрації користувача

На етапі обробки запиту за допомогою Axios спрощується автоматизація конвертації json-даних, а отримана відповідь (response) передається для занесення даних у змінну Vuex. Оскільки алгоритм перевірки та надсилання запиту до action.js майже однаковий для функцій реєстрації та авторизації, крім перевірки паролів на сумісність, у лістингу на рисунку 3.1.5 демонструються лише запити відправлення даних на серверну частину. Під час виконання запиту на авторизацію, сервер повертає спеціальний токен користувача, який подальше заноситься до локального сховища даних для отримання інформації про користувача.

```
// (Request) Register new user
async register({ commit }, payload) {
  return axios.post('/api/register', payload)
      .then(response => response)
},
// (Request) Login user
async login({ commit, dispatch }, payload) {
  return axios.post('/api/login', payload)
      .then(response => {
          localStorage.setItem('access_token', response.data.token) // set user access token
          dispatch('getUserData')
          return response
      })
},
// (Request) Get user data
async getUserData({ commit }) {
  let accessToken = localStorage.getItem('access_token') // get user access token
  return axios.post('/api/user-details', {}, {
      headers: {
          'Accept': 'application/json',
          'Authorization': `Bearer ${ accessToken }`
      }
  }).then(response => {
      commit('setUserData', response.data.data)
  })
},
```

Рисунок 3.1.5 – Відправка запиту для авторизації користувача

Також, авторизований користувач отримує набір особистих даних, зокрема: ім'я, електронну пошту та токен. У будь-який момент є можливість вийти з особистого облікового запису, процес якого представлений у лістингу на рисунку 3.1.6. Ця операція призводить до видалення даних з Vuex та локального сховища.

```
// (Request) Logout user
async logout({ commit }) {
  let accessToken = localStorage.getItem('access_token') // get user access token
  return axios.get('/api/logout', {
    headers: {
      'Authorization': `Bearer ${ accessToken }`
    }
  }).then(response => {
    commit('deleteUserData')
    localStorage.removeItem('access_token') // remove user access token
  })
}
```

Рисунок 3.1.6 – Відправка запиту для виходу з аккаунту

Усього в компоненті є 5 сторінок: головна, ігри, новини, блоги та форум. Оскільки головна сторінка є просто готовим компонентом із обраними для показу даними, то це є цікавим для розбору алгоритмом, зовсім інакше із сутностями, дані яких грузяться із ElasticSeacrh. Функція із загрузкою усіх новин на сторінку представлена в лістингу на рисунку 3.1.7.

```
// Load data from ES
const actions = {
  async fetchData({ commit }, query) {
    try {
      const response = await axios.post('elasticsearch:9200/games/_search', {
        query: {
           match_all: {}
        },
      });
      commit('setSearchData', response.data);
    } catch (error) {
      console.error('Error fetching data from Elasticsearch:', error);
    }
  },
};
```

Рисунок 3.1.7 – Відправка запиту для завантаження усіх ігор

Також, перейшовши на одну конкретну гру, вона буде отримана із ElasticSearch, проте для цього в запит, буде додано конкретний фільтр. Так само фільтри добавляються при обранні тегів на сторінці ігор. В лістингу на рисунку 3.1.8 можна побачити запит до ElasticSearch з доданим фільтром.

```
// Load one game from ES
async fetchDataById({ commit }, gameId) {
  try {
    const response = await axios.post('elasticsearch:9200/games/_search', {
      query: {
        term: {
          id: gameId
        },
      },
    });
    commit('setSearchData', response.data);
  } catch (error) {
    console.error('Error fetching data by ID from Elasticsearch:', error);
  }
},
```

Рисунок 3.1.8 – Відправка запиту для завантаження ігор по фільтру

У деяких сутностей є свої фотографія. Вони обираються при створенні такої сутності в окремих модальних вікнах, доступ до яких мають лише користувачі з певними ролями. Фотографія завантажується у виділеній формі та надсилається запит на її додавання. Алгоритм запиту показано у лістингу на рисунку 3.1.9.

```
const form = new FormData()
form.append('logo', this.file, this.file.name)

async uploadFile ({ commit }, payload) {
  let accessToken = localStorage.getItem('access_token')
  return axios.post(`/api/{payload.entity}/upload-image`, payload, {
      headers: {
          'Content-Type': 'multipart/form-data',
          'Authorization': `Bearer ${ accessToken }`
      }
  })
  .then(response => response)
}
```

Рисунок 3.1.9 – Відправка запиту для додавання фотографії

Як тільки фотографія буде завантажена, надсилається запит на створення нової сутності із прив’язкою до фото. Візьмемо для прикладу створення сутності гра. У вікні створення можна ввести наступні дані: назву гри, опис гри, обрати дату випуску, обрати видавця, обрати розробника, загрузити логотип та обрати набір тегів для подальшої фільтрації. Відправка даних для створення нового запису про гру, продемонстровано у лістингу на рисунку 3.1.10.

```
async createGame ({ commit }, payload) {
  let accessToken = localStorage.getItem('access_token')
  return axios.post('/api/game', payload, {
    headers: {
      'Authorization': `Bearer ${ accessToken }`
    }
  })
  .then(response => {
    commit('createGame', response.data)
    return response
  })
}
```

Рисунок 3.1.10 – Відправка запиту на створення гри

Дещо відрізняється сторінку форуму, адже, там є додатковий функціонал, а саме пряма комунікація із іншими користувачами. На цій сторінці є можливість власноруч створити запитання та поділитися ним з іншими користувачами. Для цього необхідно авторизуватися та натиснути кнопку "Задати питання". Після цього у вікні можна ввести заголовок та текст питання. Якщо введені дані успішно пройдуть валідацію, система відправить запит на додавання нового питання до загального списку запитань. Алгоритм представлено в лістингу на рисунку 3.1.11.

```
// (Request) Create new question
async createQuestion ({ commit }, payload) {
  let accessToken = localStorage.getItem('access_token')
  return axios.post('/api/question', payload, {
    headers: {
      'Accept': 'application/json',
      'Authorization': `Bearer ${ accessToken }`
    }
  })
  .then(response => {
    commit('createQuestion', response.data)
    return response
  })
}
```

Рисунок 3.1.11 – Відправка запиту на створення запитання на сторінці форуму

У разі переходу до певного запитання, відбудеться завантаження сторінки цього запитання з усією відповідною інформацією та коментарями до нього. Ця операція схожа на попереднє завантаження сторінки гри і виконується за аналогічним алгоритмом, що представлено в лістингу на рисунку 3.1.12.

```
const actions = {
  async loadQuestionArticle({ commit }, questionId) {
    try {
      const response = await axios.post('elasticsearch:9200/questions/_search', {
        query: {
          term: {
            id: questionId,
          },
        },
      });
      commit('setQuestionArticle', response.data);
      return response.data;
    } catch (error) {
      console.error('Error loading question article from Elasticsearch:', error);
      throw error;
    }
  },

  async loadAnswerArticle({ commit }, questionId) {
    try {
      const response = await axios.post('elasticsearch:9200/answers/_search', {
        query: {
          term: {
            question_id: questionId,
          },
        },
      });
      commit('setAnswerArticle', response.data);
      return response.data;
    } catch (error) {
      console.error('Error loading answer article from Elasticsearch:', error);
      throw error;
    }
  },
};
```

Рисунок 3.1.12 – Завантаження інформації про запитання та коментарів до нього

На даній сторінці ви можете видалити своє власне запитання. Це здійснюється через кнопку "Видалити", яка розташована поруч із інформацією про автора запитання. Операцію видалення можна побачити у лістингу на рисунку 3.1.13.

```
// (Request) Delete question
async deleteQuestion ({ commit }, payload) {
  let accessToken = localStorage.getItem('access_token') // get user access token
  return axios.delete(`/api/question/${ payload }`, {
    headers: {
        'Authorization': `Bearer ${ accessToken }`
    }
  })
  .then(response => {
      commit('deleteQuestion', payload)
      return response
  })
}
```

Рисунок 3.1.13 – Видалення запитання з форуму

На цій сторінці можна додати власний коментар, але для цього необхідно авторизуватися. Якщо користувач не увійшов у систему, функція поверне сповіщення про відсутність токена у сховищі. У випадку авторизації, користувач може написати коментар та опублікувати його нижче, як це показано у лістингу на рисунку 3.1.14.

```
// (Request) Add new comment
async addComment ({ commit }, payload) {
  let accessToken = localStorage.getItem('access_token')
  return axios.post('/api/answer', payload, {
    headers: {
      'Accept': 'application/json',
      'Authorization': `Bearer ${ accessToken }`
    }
  })
  .then(response => {
    commit('addComment', response.data)
    return response
  })
}
```

Рисунок 3.1.14 – Створення коментаря

Реалізація видалення власних коментарів передбачає можливість прибрати коментар, який не влаштовує вас або який було додано помилково. Цей процес показано у лістингу на рисунку 3.1.15. Видалення здійснюється за допомогою алгоритму, який знаходить потрібний індекс коментаря у загальному масиві коментарів та вилучає його.

```
async deleteComment ({ commit }, payload) {
  let accessToken = localStorage.getItem('access_token')
  return axios.delete(`/api/answer/${ payload }`, {
    headers: {'Authorization': `Bearer ${ accessToken}`}
  })
  .then(response => {
      commit('deleteComment', payload)
      return response
  })
}
```

Рисунок 3.1.15 – Видалення коментаря

Згідно з роботою, допоміжні компоненти відіграють суттєву роль у системі. Ці елементи, такі як плейсхолдер, помилка, поле, допоміжна іконка для поля, модальне вікно та сповіщення, призначені для повторного використання. На кожній сторінці сайту, де передбачається завантаження даних з сервера, присутній плейсхолдер. Розроблено шість шаблонів цього компонента для натурального відображення сторінки з відсутністю актуальних даних. Анімаційні ефекти плейсхолдерів створені за допомогою @keyframes, інструкція до яких подана у лістингу на рисунку 3.1.16.

```
// --- Animation gradient block --- //
.activity {
    width: 45%;
    height: 100%;
    position: absolute;
    left: -45%;
    background-image: linear-gradient(to left, rgba(251,251,251, .05), rgba(251,251,251, .3),
                      rgba(251,251,251, .6), rgba(251,251,251, .3), rgba(251,251,251, .05));
    animation: loading 1s infinite;
    z-index: 45;
}
// --- Animation --- //
@keyframes loading {
    0% { left: -45%; }
    100% { left: 100%; }
}
```

Рисунок 3.1.16 – Анімація плейсхолдеру в CSS

Повідомлення про помилку використовувалось у всіх областях, де користувач міг взаємодіяти з різними елементами. Компонент повідомлення передавав значення через тег slot.

Окремий компонент поля був розроблений для забезпечення більшої гнучкості в передачі даних та уникнення повторного копіювання стилів. Цей компонент приймає кілька атрибутів, таких як тип, текст, максимальна довжина введеного значення та стиль іконки. Розмітка компонента поля представлена у лістингу на рисунку 3.1.17.

```
<div class="field">
  <!-- Field -->
  <input :type="data.type"
         :placeholder="$t(`${ data.placeholder }`)"
         :maxlength="data.maxlength"
         :style="{ 'background-image': `url(${ require(`@/assets/icons/${ data.icon }.png`).default })` }"
         :value="value"
         @input="$emit('input', $event.target.value)"
  >
  <!-- -->

  <!-- Help section -->
  <Help :text="data.msg" v-if="data.msg" />
  <!-- -->
</div>
```

Рисунок 3.1.17 – Компонент field.vue

Допоміжна іконка для поля є складовою частиною компонента поля і використовується для більш детального пояснення даних, які потрібно ввести у відповідне поле. Підказка з'являється при наведенні на цей компонент.

Модальне вікно було розроблене з метою виділення окремої області контенту, де користувачеві потрібно взаємодіяти з веб-сайтом, вводячи певні дані. Це вікно має вбудовані функції для появи та зникнення, а також динамічну область для вмісту. Опис функціональності модального вікна наведено у лістингу на рисунку 3.1.18.

```
// Show modal window
show () {
  this.isBgCreated = this.isModalCreated = true
  this.timer(100)
    .then(() => { this.isBgActivated = true })
    .then(() => { this.isModalActivated = true })
},
// Hide modal window
hide () {
  this.isBgActivated = this.isModalActivated = false
  this.timer(100).then(() => { this.isBgCreated = this.isModalCreated = false })
}
```

Рисунок 3.1.18 – Анімації модального вікна

Веб-компонент "сповіщення" було створено з метою надання важливої інформації користувачеві. Його робочий алгоритм, порівняно з попереднім компонентом, відрізняється наявністю внутрішнього таймеру, який відповідає за автоматичне закриття сповіщення після певного часу. Описаний алгоритм функціонування компоненту сповіщення можна знайти у лістингу на рисунку 3.1.19.

```
// Show notify
show () {
  this.isNotifyCreated = true
  this.timer(100).then(() => { this.isNotifyActivated = true })
  this.timer(3000).then(() => { this.hide() })
},
// Hide notify
hide () {
  this.isNotifyActivated = false
  this.timer(100).then(() => {
    this.isNotifyCreated = false
    this.msg = ''
    this.type = false
  })
}
```

Рисунок 3.1.19 – Анімації вікна сповіщень

3.2 Реалізація функціоналу серверної частини сайту

Реалізація серверної частини – це тип розробки, що виконуються на сервері, відіграють ключову роль. Розробники, які працюють на серверній стороні, фокусуються на внутрішніх механізмах інфраструктури, і цей аспект також називають внутрішньою розробкою. Ця сфера програмування має велике значення, оскільки веб-браузери або клієнти взаємодіють з веб-серверами для отримання інформації. Основні завдання на серверній стороні включають:

- розробку динамічних веб-сайтів;
- створення веб-додатків;
- підключення веб-сайтів до баз даних.

В першу чергу, на серверній частині сайту була реалізована система авторизації, що дозволяє звичайним відвідувачам ігрового порталу створювати свої власні облікові записи та користуватися ними для створення запитань або відповідей.

Реєстрація становить перший етап, який користувач повинен пройти, щоб мати можливість користуватися власним обліковим записом у майбутньому. Для досягнення цієї мети був розроблений контролер AuthController.php, де присутня функція реєстрації, представлена в лістингу на рисунку 3.2.1.

```php
public function register(Request $request)
{
  $userEmail = User::where('email', $request->email)->value('email');
  if ($userEmail && $userEmail != '') {
    return response()->json([
      'success' => false,
      'message' => 'User with requested email already exists.'
    ], 400);
  }
  $request->validate([
    'name' => 'string|max:255',
    'email' => 'required|string|email|max:255|unique:users',
    'password' => 'required|string|min:6'
  ]);
  $user = User::create([
    'name' => $request->name,
    'email' => $request->email,
    'password' => Hash::make($request->password)
  ]);
  $user->assignRole(env('START_USER_ROLE'));
  return response()->json([
    'success' => true,
    'message' => 'User registered successfully, use the login page to log into the account.'
  ]);
}
```

Рисунок 3.2.1 – Функція реєстрації користувача

На наступному етапі після реєстрації користувача відбувається процес авторизації. При цьому клієнтська частина сайту надсилає форму авторизації, а обробка цих даних виконується згідно представленого коду в лістингу на рисунку 3.2.2.

```php
public function login(Request $request)
{
  $input = $request->only(['email', 'password']);
  $request->validate([
    'email' => 'required|email',
    'password' => 'required|min:8',
  ]);
  if (auth()->attempt($input)) {
    $token = auth()->user()->createToken('passport_token')->accessToken;
    return response()->json([
        'success' => true,
        'message' => 'User login successfully.',
        'token' => $token
    ]);
  } else {
    return response()->json([
      'success' => false,
      'message' => 'User authentication failed.'
    ], 401);
  }
}
```

Рисунок 3.2.2 – Функція авторизації користувача

На сервері відбувається авторизація користувача, після чого надходить запит щодо отримання його особистих даних на підставі виданого токена. Функція, що представлена в лістингу на рисунку 3.2.3, відповідає за повернення інформації про користувача та його роль, базуючись на отриманому токені.

```php
public function userDetails()
{
    $user = auth()->user();
    $user->role = current($user->getRoles());
    unset($user->roles);

    return response()->json([
      'success' => true,
      'message' => 'Data fetched successfully.',
      'data' => $user
    ]);
}
```

Рисунок 3.2.3 – Функція повернення даних користувача

На серверній частині реалізована можливість виходу користувача з аккаунту шляхом відкликання його токену. Це призводить до припинення активного стану токена, унеможливлюючи його подальше використання. Логіка цієї операції описана в лістингу на рисунку 3.2.4.

```
public function logout()
{
    $access_token = auth()->user()->token();

    // logout from only the current device
    $tokenRepository = app(TokenRepository::class);
    $tokenRepository->revokeAccessToken($access_token->id);

    return response()->json([
      'success' => true,
      'message' => 'User logout successfully.'
    ]);
}
```

Рисунок 3.2.4 – Функція виходу з аккаунту

На серверній частині сайту присутня логіка API для створення ролей, яка відповідальна за збереження даних у відповідній таблиці. Такий функціонал надано у лістингу на рисунку 3.2.5.

```
public function store(Request $request): \Illuminate\Http\JsonResponse
{
    $data = $request->validate([
      'name' => 'required|string',
      'slug' => 'required|string',
      'description' => 'required|string'
    ]);

    $role = new Role();
    $newRole = $role::create($data);

    return response()->json([
      'success' => true,
      'message' => 'Role ' . $newRole->name . ' created successfully.'
    ]);
}
```

Рисунок 3.2.5 – Функція створення ролі

На серверній частині сайту реалізовано API для створення дозволів для певної ролі. Ця логіка дозволяє зберігати дозволи у вигляді масиву значень для конкретної ролі. Цей функціонал представлено в лістингу на рисунку 3.2.6.

```
public function store(Request $request): \Illuminate\Http\JsonResponse
{
    $data = $request->validate([
      'name' => 'required|string',
      'slug' => 'required',
      'description' => 'required|string',
      'role' => 'required|string'
    ]);
    $permission = new Permission();
    $newPermission = $permission::create($data);
    $role = Role::where('slug', $data['role'])->first();
    $role->assignPermission($newPermission);
    return response()->json([
      'success' => true,
      'message' => 'Permission ' . $newPermission->name . ' created and assigned to ' . $data['role'] . '.'
    ]);
}
```

Рисунок 3.2.6 – Функція зберігання дозволів

Також на серверній частині сайту було реалізовано контролер, який відповідає за присвоєння певної ролі користувачеві. Цей функціонал показано в лістингу на рисунку 3.2.7.

```
public function store(Request $request): \Illuminate\Http\JsonResponse
{
    $data = $request->validate([
      'user_id' => 'required',
      'role' => 'required|string'
    ]);

    $user = User::find($data['user_id']);
    $user->assignRole($data['role']);

    return response()->json([
      'success' => true,
      'message' => 'User ' . $user->name . ' got the next role: ' . $data['role'] . '.'
    ]);
}
```

Рисунок 3.2.7 – Функція присвоєння ролі користувачеві

Основною перевагою реалізованого порталу є швидка загрузка даних, яка була забезпечена за допомогою зберігання та отримання статичних даних з ElasticSearch. Для того, аби ці дані точно було там, створено два базових механізми. Першим було збереження даних одразу при створенні сутності, іншим ж було додано команду збереження усіх або ж обраних індексів з даними до рушія, в випадках коли всі дані мають бути завантажені одномоментно. Функція для збереження одного індексу до ElasticSearch представлена у лістингу на рисунку 3.2.8.

```
private function reindexOneIndex(array $indexData): void
{
  $baseClass = $indexData['base'];
  $esClass = $indexData['es'];

  $records = $baseClass::all();
  foreach ($records as $record) {
    $record->_id = $record->id;
    $record->author = User::getAuthor($record->user_id);
    try {
      $esClass::updateOrCreate($record->toArray());
      if (php_sapi_name() === 'cli') {
        $this->output->writeln("<info>Record with $record->id is saved.</info>");
      }
    } catch (\Exception $e) {
      if (php_sapi_name() === 'cli') {
        $this->output->writeln("<error>An error during saving data into ElasticSeacrh for record: $record->id. Entity is $baseClass. An error: {$e->getMessage()}</error>");
      } else {
        $this->logger->error("An error during saving data into ElasticSeacrh for record: $record->id. Entity is $baseClass. An error: {$e->getMessage()}");
      }
    }
  }
}
```

Рисунок 3.2.8 – Функція збереження одного індексу

Також, дані до ElasticSearch потрапляють при створенні чи оновленні певної сутності. Розібравши на прикладі сутності game, можна побачити, як дані зберігаються до основної бази даних та до ElasticSearch. Функція збереження сутності гри представлена в лістингу на рисунку 3.2.9.

```
public function store(Request $request)
{
    $data = $request->validate([
      'name' => 'required|string',
      'description' => 'required|string',
    ]);
    $data['user_id'] = auth()->id();
    $data['comments_qty'] = 0;

    $game = Game::create($data);
    $game->author = User::getAuthor($game->user_id);
    $game->_id = $question->id;
    EsGame::updateOrCreate($game->toArray());

    return $game;
}
```

Рисунок 3.2.9 – Функція збереження сутності гри

Звичайно ж, створену сутність, в даному випадку гру, можна редагувати, при цьому дані в ElasticSearch також оновлюються разом із цим. Функція редагування даних про гру представлена в лістингу на рисунку 3.2.10.

```
public function update(Request $request, $gameId)
{
    $game = Game::find($gameId);
    if (!$game || !$game->id) {
      return response()->json(['error' => 'Requested news does not exist'], 404);
    }
    $data = $request->validate([
      'title' => 'required|string',
      'content' => 'required|string',
    ]);
    try {
      $game->update($data);
      $game->author = User::getAuthor($game->user_id);
      $game->_id = $game->id;
      EsGame::updateOrCreate($game->toArray());

      return $game;
    } catch (\Exception $e) {
      report($e);
      return response()->json(['error' => 'Something went wrong: ' . $e->getMessage()]);
    }
}
```

Рисунок 3.2.10 – Функція редагування сутності гри

Відповідно, останньою функцією, для роботи із будь якою сутністю є її видалення, дана логіка показана в лістингу на рисунку 3.2.11.

```
public function destroy($gameId)
{
    try {
      $game = Game::find($gameId);
      if ($game) {
        $game->delete();
        EsGame::delete(gameId);
        return response()->json(null, 204);
      }
      return response()->json(['error' => 'Requested game does not exist'], 404);
    } catch (\Exception $e) {
      report($e);
      return response()->json(['error' => 'Something went wrong: ' . $e->getMessage()]);
    }
}
```

Рисунок 3.2.11 – Функція видалення сутності гри

Цікавим та важливим функціоналом, що потребує описування є логіка коментування для усіх сутностей, адже, це окремий механізм, який є представлений на сторінках кожної окремої гри, новини, блогу чи запитання. Відповідно було створено універсальну логіку, яка зберігає коментар із прив’язкою до його сутності, також рахуючи кількість відповідей для певної сутності та зберігає ці дані до ElasticSearch. В лістингу на рисунку 3.2.12 показано логіку збереження коментаря.

```php
public function store(Request $request)
{
    $data = $request->validate([
        'entity_id' => 'required|integer',
        'entity_type' => 'required|string',
        'content' => 'required|string',
    ]);
    $data['user_id'] = auth()->id()
    $comment = Comment::create($data);
    $comment->_id = $comment->id;
    EsComment::updateOrCreate($comment->toArray());
    if ($comment) {
        $entity = null;
        // Switch case logic here
        if ($entity) {
            $entity->comments_qty = ++$entity->comments_qty;
            $entity->save();
            $entity->_id = $entity->id;
            $esEntity->updateOrCreate($entity->toArray());

            $comment->author = User::getAuthor($data['user_id']);

            return $comment;
        }
    }

    return response()->json(['error' => 'Comment was not saved'], 400);
}
```

Рисунок 3.2.12 – Функція збереження коментаря

Окремо хочеться винести логіку, яка показує алгоритм, за яким для коментаря відбирається сутність до якої власне цей коментар і залишається. Це реалізовано через switch case і показано в лістингу на рисунку 3.2.13.

```php
switch ($data['entity_type']) {
    case 'game':
        $entity = Game::find($data['entity_id']);
        $esEntity = new EsGame();
        break;
    case 'news':
        $entity = News::find($data['entity_id']);
        $esEntity = new EsNews();
        break;
    case 'blog':
        $entity = Blog::find($data['entity_id']);
        $esEntity = new EsBlog();
        break;
    case 'question':
        $entity = Question::find($data['entity_id']);
        $esEntity = new EsQuestion();
        break;
    default:
        return response()->json(['error' => 'Unsupported entity type'], 400);
}
```

Рисунок 3.2.13 – Алгоритм відбору сутності до якої зберігається коментар

Також, користувач, що написав відповідь, може його видалити, відповідно такий запис буде прибрано з бази даних та з ElasticSearch, а кількість відповідей в запитання, до якого воно належить, буде зменшено, все це можна переглянути в лістингу на рисунку 3.2.14.

```
public function destroy($entityType, $commentId)
{
    try {
      $entity = null;
      // Switch case logic here
      $comment = Comment::find($commentId);
      $entity->comments_qty = --$entity->comments_qty;
      $entity->save();
      $entity->_id = $entity->id;
      $esEntity->updateOrCreate($entity->toArray());

      if ($comment) {
        $comment->delete();
        EsComment::delete($comment->id);
        return response()->json(['success' => true, 'message' => 'The comment was deleted successfully.'], 204);
      }
      return response()->json(['error' => 'Requested comment does not exists'], 404);
    } catch (\Exception $e) { report($e);
      return response()->json(['error' => 'Something went wrong: ' . $e->getMessage()]);
    }
}
```

Рисунок 3.2.14 – Функція видалення коментаря

Останнім, що хочеться розглянути це загрузка зображення, перед збереженням сутності. Всі файли зберігаються в окрему папку і для зображень генерується унікальне ім’я, на основі сутності, що зберігається. Даний функціонал можна переглянути в лістингу на рисунку 3.2.15.

```php
public function uploadNewsLogo(Request $request): \Illuminate\Http\JsonResponse
{
    if (!$request->hasFile('logo')) {
      return response()->json(['Uploaded file is not found'], 400);
    }

    $file = $request->file('logo');
    if(!$file->isValid()) {
      return response()->json(['invalid_file_upload'], 400);
    }

    $path = "/images/$request['entity_type']/logo/" . auth()->id();
    $file->move(public_path() . $path, $file->getClientOriginalName());
    $path = env('APP_URL') . '/' . "images/$request['entity_type']/logo/" . auth()->id() . '/' . $file->getClientOriginalName();
    return response()->json(compact('path'));
}
```

Рисунок 3.2.14 – Функція загрузки зображення на сервер

3.3 Реалізація бази даних

Дані, збережені у базі даних, були оброблені через технологію MySQL для забезпечення коректності введення та отримання інформації, а також для забезпечення надійності даних. Слід зауважити, що MySQL та SQL – різні речі. MySQL є однією з найбільш популярних систем управління базами даних (СУБД), яка втілює модель клієнт-сервер та використовує мову структурованих запитів SQL [20].

Оскільки вся серверна частина сайту та взаємодія з базою даних були організовані через Laravel, то ORM (Об'єктно-реляційне відображення) було реалізовано з використанням бібліотеки Illuminate. ORM дозволяє зручніше працювати з базою даних, перетворюючи дані у вигляд об'єктів, що спрощує їхню обробку та використання.

IDE DataGrip - це потужне середовище для управління базами даних, спроектоване для розробників. Воно дозволяє працювати з базами даних через користувацький інтерфейс і включає підтримку різних типів баз даних, включаючи MySQL, PostgreSQL, Microsoft SQL Server, Oracle та інші [21].

DataGrip пропонує кілька значних переваг, які роблять його привабливим для використання:

1. Підтримка декількох драйверів: Він забезпечує можливість працювати з різними типами баз даних, надаючи гнучкість у виборі технологій.

2. Інтелектуальне доповнення коду: Має функціонал автодоповнення, який враховує контекст і структуру бази даних, що спрощує написання запитів.

3. Візуальний редактор таблиць: Дозволяє зручно додавати, видаляти, редагувати та клонувати рядки даних у таблицях.

4. Підтримка впровадження змін: Надає засоби для зміни структури бази даних без втрати даних або функціональності.

5. Вибір версії SQL драйвера: Під час підключення до бази даних можна обирати певну версію SQL драйвера, що є важливим для забезпечення сумісності з конкретною базою даних.

Нижче буде описано, основні таблиці, що активно задіяні в роботі сайту. Деякі з таблиць, створені за допомогою сторонніх бібліотек, тому їх опис, немає фактичного змісту.

Таблиця «users» – відповідає за збереження інформації про усіх зареєстрованих користувачів.

Таблиця 3.1 – Таблиця «users»

Ім’я поля	Тип поля	Ключ	Додаткові параметри
id	int(11)	PRIMARY_KEY	AUTO_INCREMENT
name	varchar(255)	-	-
email	varchar(255)	-	-
password	varchar(255)	-	-
created_at	timestamp	-	-
updated_at	timestamp	-	-

Таблиця «roles» – відповідає за збереження інформації про усі наявні ролі та їх опис.

Таблиця 3.2 – Таблиця «roles»

Ім’я поля	Тип поля	Ключ	Додаткові параметри
id	int(11)	PRIMARY_KEY	AUTO_INCREMENT
name	varchar(255)	-	-
slug	varchar(255)	-	-
description	text	-	-
created_at	timestamp	-	-
updated_at	timestamp	-	-

Таблиця «permissions» – відповідає за збереження інформації про усі доступні дозволи.

Таблиця 3.3 – Таблиця «permissions»

Ім’я поля	Тип поля	Ключ	Додаткові параметри
id	int(11)	PRIMARY_KEY	AUTO_INCREMENT
name	varchar(255)	-	-
slug	varchar(255)	-	-
description	text	-	-
created_at	timestamp	-	-
updated_at	timestamp	-	-

Таблиця «permission_role» – відповідає за збереження даних, щодо зв’язків між таблицями role та permissions.

Таблиця 3.4 – Таблиця «permission_role»

Ім’я поля	Тип поля	Ключ	Додаткові параметри
id	int(11)	PRIMARY_KEY	AUTO_INCREMENT
permission_id	int(11)	FOREIGN KEY	-
role_id	int(11)	FOREIGN KEY	-
created_at	timestamp	-	-
updated_at	timestamp	-	-

Таблиця «permission_user» – відповідає за збереження даних, щодо зв’язків між таблицями permissions та user.

Таблиця 3.5 – Таблиця «permission_user»

Ім’я поля	Тип поля	Ключ	Додаткові параметри
id	int(11)	PRIMARY_KEY	AUTO_INCREMENT
permission_id	int(11)	FOREIGN KEY	-
user_id	int(11)	FOREIGN KEY	-
created_at	timestamp	-	-
updated_at	timestamp	-	-

Таблиця «role_user» – відповідає за збереження даних, щодо зв’язків між таблицями role та user.

Таблиця 3.6 – Таблиця «role_user»

Ім’я поля	Тип поля	Ключ	Додаткові параметри
id	int(11)	PRIMARY_KEY	AUTO_INCREMENT
role_id	int(11)	FOREIGN KEY	-
user_id	int(11)	FOREIGN KEY	-
created_at	timestamp	-	-
updated_at	timestamp	-	-

Таблиця «game» – відповідає за збереження інформації про усі наявні новини.

Таблиця 3.7 – Таблиця «game»

Ім’я поля	Тип поля	Ключ	Додаткові параметри
id	int(11)	PRIMARY_KEY	AUTO_INCREMENT
user_id	int(11)	FOREIGN KEY	-
name	varchar(255)	-	-
logo	varchar(255)	-	-
developer	text	-	-
publisher	text	-	-
description	text	-	-
tags	text	-	-
release_at	timestamp	-	-
created_at	timestamp	-	-
updated_at	timestamp	-	-

Таблиця «blog» – відповідає за збереження інформації про усі наявні новини.

Таблиця 3.8 – Таблиця «blog»

Ім’я поля	Тип поля	Ключ	Додаткові параметри
id	int(11)	PRIMARY_KEY	AUTO_INCREMENT
user_id	int(11)	FOREIGN KEY	-
game_id	int(11)	FOREIGN KEY	
title	varchar(255)	-	-
logo	varchar(255)	-	-
content	text	-	-
created_at	timestamp	-	-
updated_at	timestamp	-	-

Таблиця «question» – відповідає за збереження інформації про усі наявні запитання.

Таблиця 3.9 – Таблиця «question»

Ім’я поля	Тип поля	Ключ	Додаткові параметри
id	int(11)	PRIMARY_KEY	AUTO_INCREMENT
user_id	int(11)	FOREIGN KEY	-
game_id	int(11)	FOREIGN KEY	
title	varchar(255)	-	-
content	text	-	-
comment_qty	int(11)	-	-
created_at	timestamp	-	-
updated_at	timestamp	-	-

Таблиця «answer» – відповідає за збереження інформації про усі наявні відповіді та їх зв’язок із запитаннями.

Таблиця 3.10 – Таблиця «answer»

Ім’я поля	Тип поля	Ключ	Додаткові параметри
id	int(11)	PRIMARY_KEY	AUTO_INCREMENT
user_id	int(11)	FOREIGN KEY	-
question_id	int(11)	FOREIGN KEY	-
content	text	-	-
created_at	timestamp	-	-
updated_at	timestamp	-	-

3.4 Тестування розробленого ігрового порталу

Етап тестування веб-сайту визначається як критичний компонент у циклі розробки проекту, оскільки він спрямований на виявлення і усунення потенційних помилок та неполадок, які можуть виникнути в процесі взаємодії користувача з веб-інтерфейсом. В ході тестування перевіряється функціональність та коректність роботи різноманітних елементів інтерфейсу, зокрема тих, що прямо взаємодіють з користувачем.

Слід зауважити, що веб-сайти мають велику кількість інтерактивних елементів, які стосуються дій та процесів, що відбуваються через участь користувача. Ці процеси, в основному, управляються за допомогою елементів інтерфейсу та можуть бути важливими для сприйняття та задоволення користувача під час використання веб-сайту. Прикладом слугують: форма реєстрації та авторизації, форма створення сутностей, форми на форумі і блогах.

Спеціально для точної перевірки всіх введених даних, був створений відокремлений файл, призначений для виконання перевірки усіх полів на відповідність початково встановленим вимогам. Цей файл також містить масив об’єктів, який включає набір унікальних полів, що повторно використовуються в інтерфейсі. В цьому наборі даних містяться наступні параметри: тип поля, шаблон регулярного виразу для перевірки та повідомлення про помилку. Основна перевірка виконується при завантаженні секції з формою, що містить відповідне поле.

У модальних вікнах для реєстрації, авторизації та створення сутностей і форумі застосовуються перевірки для виявлення наявності порожніх полів. Якщо хоча б одне поле залишиться незаповненим, процес валідації спрацює, викликавши помилку (рисунок 3.4.1).

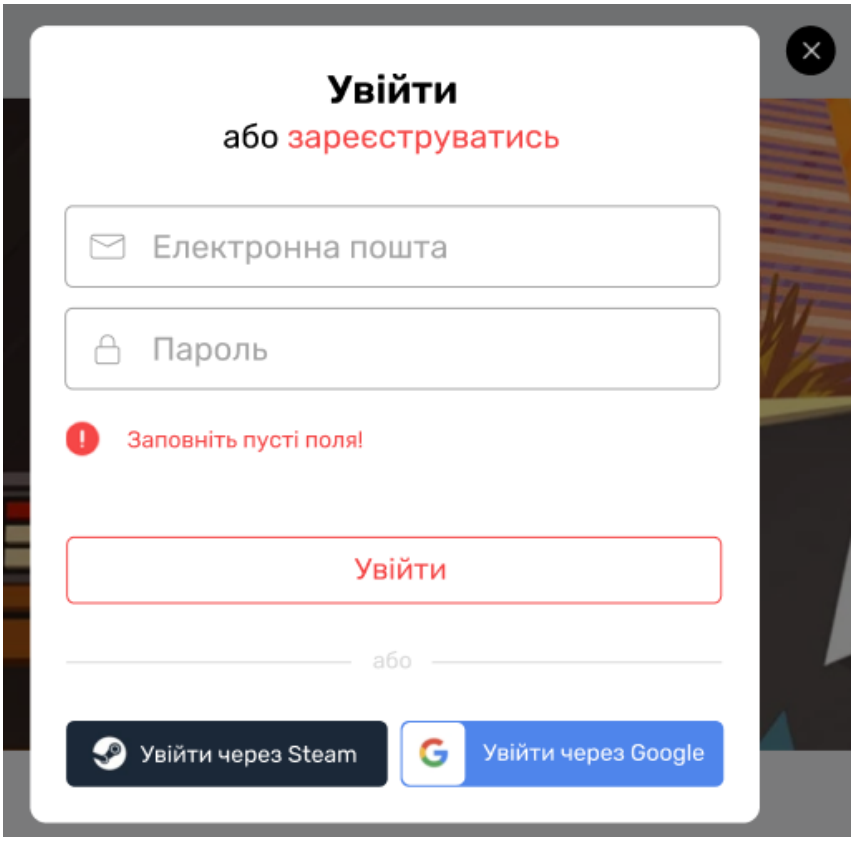

Рисунок 3.4.1 – Помилка, яка показується при пустих полях

Під час реєстрації обов'язково потрібно вказати унікальну електронну адресу, яка не фігурує в базі даних. У разі введення адреси, яка вже присутня у базі, сервер поверне помилку щодо наявності користувача з такою самою електронною адресою. (рисунок 3.4.2)

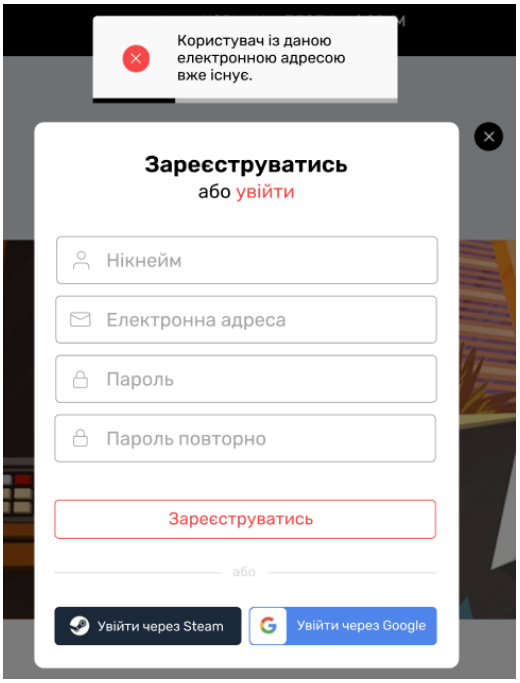

Рисунок 3.4.2 – Помилка про наявність даної електронної адреси в базі

За несумісністю з вимогами форматування у полі відображається відповідне повідомлення про помилку, що підтверджує успішну валідацію за певними шаблонами RegExp (рисунок 3.4.3).

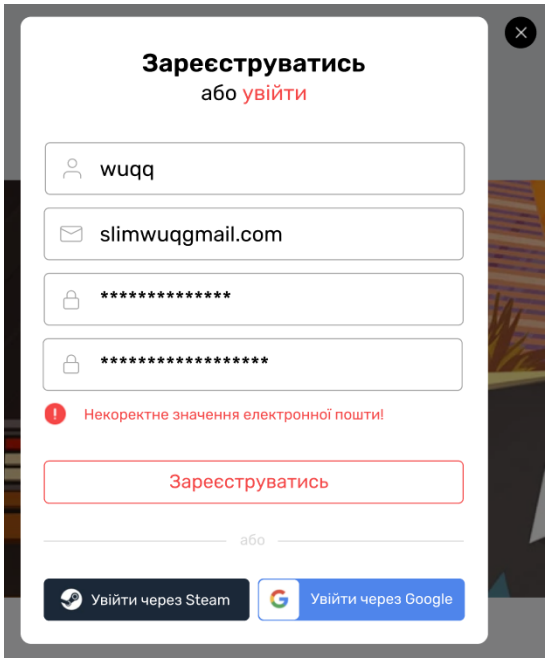

Рисунок 3.4.3 – помилка, що сигналізує про некоректні дані

Під час реєстрації вимагається введення однакових паролів для запобігання шахрайства з боку користувачів. У разі некоректного введення одного з паролів система відображає відповідне повідомлення про помилку (рисунок 3.4.4).

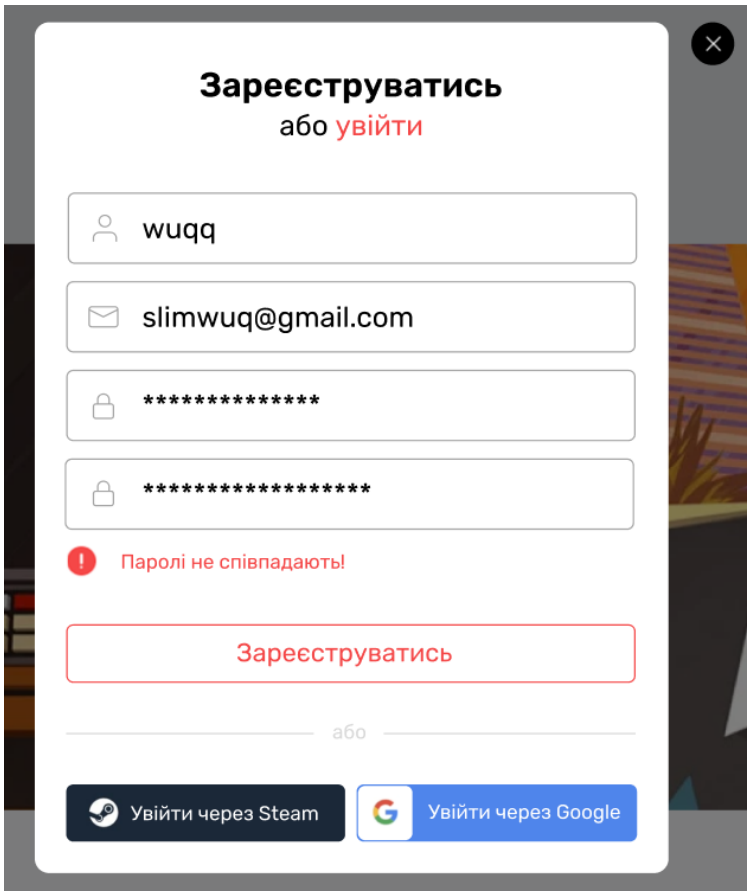

Рисунок 3.4.4 – Помилка яка свідчить, що паролі не співпадають

Під час авторизації можливе неправильне введення обов'язкових даних для авторизації, що призведе до відображення повідомлення про некоректність відправленої інформації (рисунок 3.4.5).

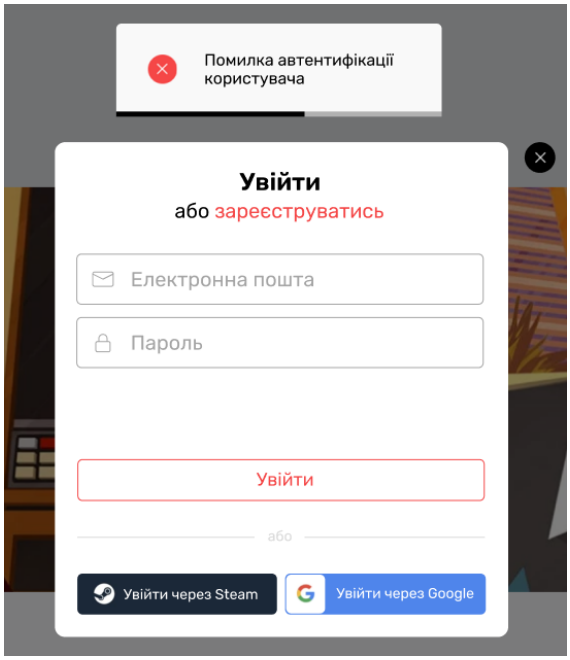

Рисунок 3.4.5 – Помилка, яка вказує, що якісь із даних користувача є неправильні

При коректному введенні даних під час реєстрації нового користувача з’явиться сповіщення про успішне створення нового облікового запису (рисунок 3.4.6).

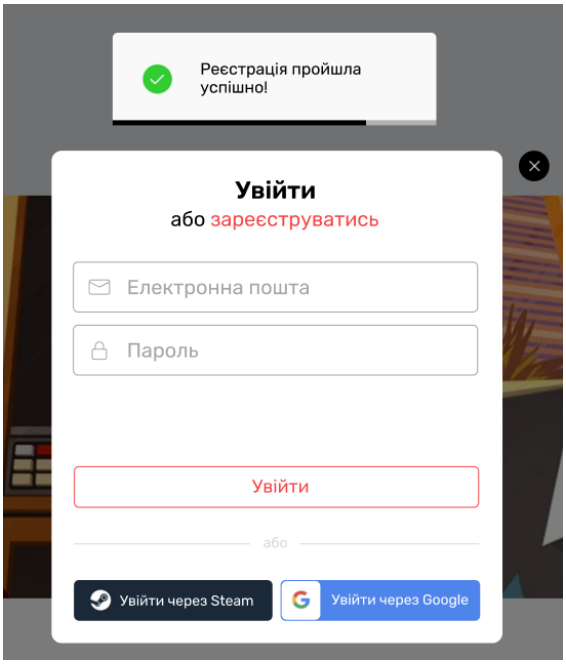

Рисунок 3.4.6 – Сповіщення про успішне створення аккаунту

В разі успішного входу користувача до свого облікового запису буде показане сповіщення про вдалу авторизацію (рисунок 3.4.7). При виході користувача з облікового запису також з’явиться сповіщення про успішний вихід (рисунок 3.4.8).

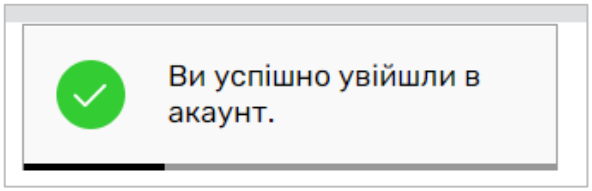

Рисунок 3.4.7 – Успішна авторизація до акаунту

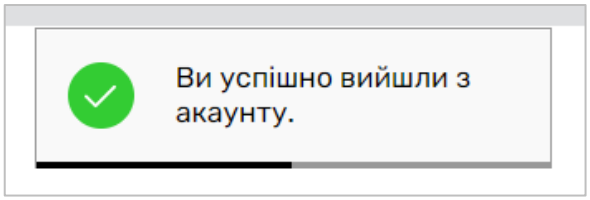

Рисунок 3.4.8 – Успішний вихід з акаунту

При створенні гри у модальному вікні, коли користувач вводить необхідну інформацію та успішно додає дані, відображається сповіщення про успішне створення такої гри (рисунок 3.4.9). Форма для створення ігор, що містить всі поля для введення інформації, зображена на рисунках 3.4.10.

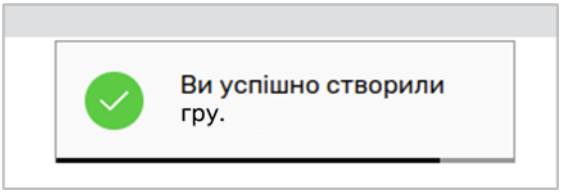

Рисунок 3.4.9 – Успішне створення гри

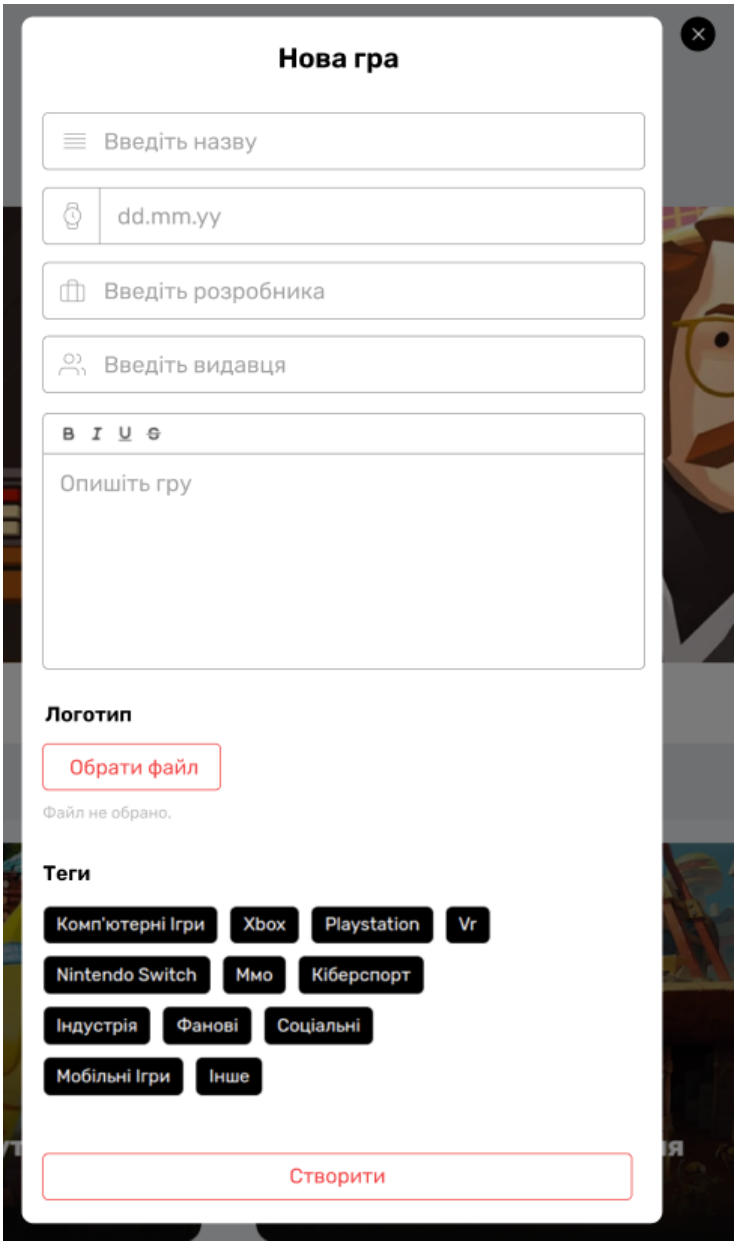

Рисунок 3.4.10 – Форма створення гри

Після видалення гри, аналогічне сповіщення буде висвітлено (рисунок 3.4.11).

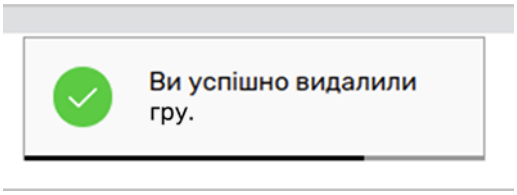

Рисунок 3.4.11 – Успішне видалення гри

До всіх сутностей на сайт, будь то гра, новина, блог чи звичайне запитання від іншого користувача, можна залишити коментар, аби поділитись свою думкою. Для цього було розроблено окрему форму, де свою думку можна прописати. При успішній публікації коментаря, буде висвітлено сповіщення (рисунок 3.4.11). А форма для написання коментаря представлена на рисунку 3.4.12.

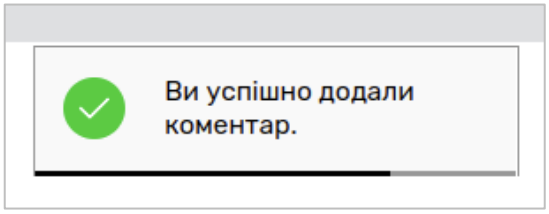

Рисунок 3.4.11 – Успішне створення коментаря

Рисунок 3.4.12 – Форма написання коментаря

У випадку, якщо користувач має бажання видалити коментар, він отримає сповіщення про успішне здійснення даного процесу (рисунок 3.4.13).

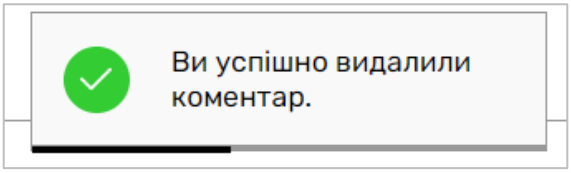

Рисунок 3.4.13 – Успішне видалення коментаря

Кожен зареєстрований користувач може відвідати форум, де може поставити своє запитання, якщо ж на інших сторінках порталу відповіді не знайшлось. При успішному створені такого запитання, це буде просигналізовано відповідним сповіщенням (рисунок 3.4.14).

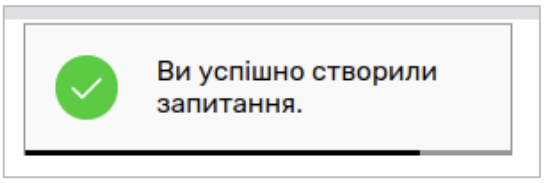

Рисунок 3.4.14 – Успішне створення запитання

Після видалення запитання, аналогічне сповіщення буде висвітлено (рисунок 3.4.15). Усі кнопки видалення, мають одинаковий графічний інтерфейс, тому та одинаково реагують на натискання. Кнопку видалення запитання зображено на рисунку 3.4.16.

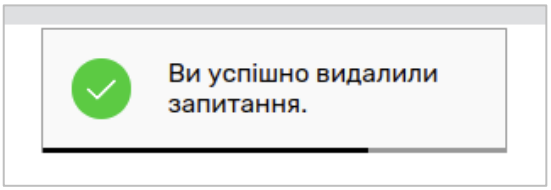

Рисунок 3.4.15 – Успішне видалення запитання

Рисунок 3.4.16 – Запитання із кнопкою видалення

Значна частина тестування веб-сайту зосереджена на перевірці адаптивності його структури на різних платформах та пристроях. У даній роботі впроваджено адаптивну верстку та локальну стилізацію різних елементів для оптимального відображення на різних розмірах екранів.

У мобільній адаптації веб-сайту стандартна панель меню замінюється на бургер-меню, розташоване у лівій частині шапки (рисунок 3.4.17). Це рішення спроектовано з метою забезпечення комфортного користувацького досвіду для мобільних (рисунок 3.4.18) та планшетних пристроїв (рисунок 3.4.19).

Рисунок 3.4.17 – Бургер-меню веб-сайту при мобільній адаптації

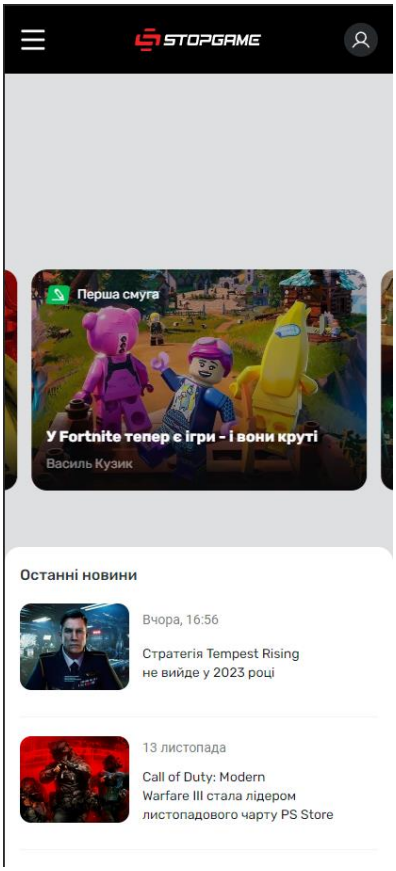

Рисунок 3.4.18 – Веб-сайт на мобільному девайсі

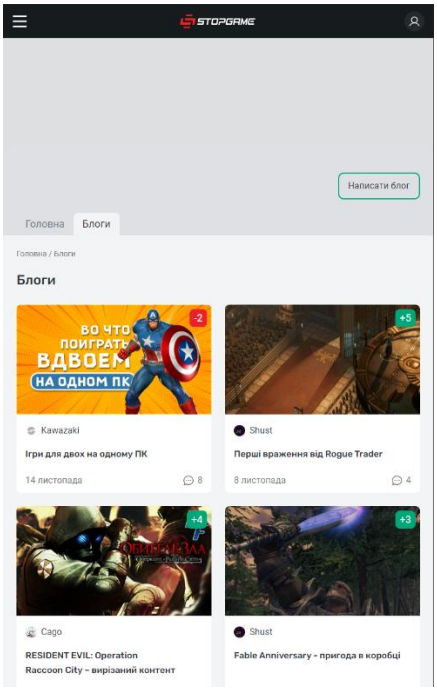

Рисунок 3.4.19 – Веб-сайт на планшетному девайсі

Серед усіх елементів, які зазнали змін після адаптації, сторінка ігор отримати значні модифікації у структурі. Бічна панель, розташована спочатку ліворуч, була переміщена до верхньої частини сторінки та перероблена для зручного вибору всіх категорій (рисунок 3.4.20).

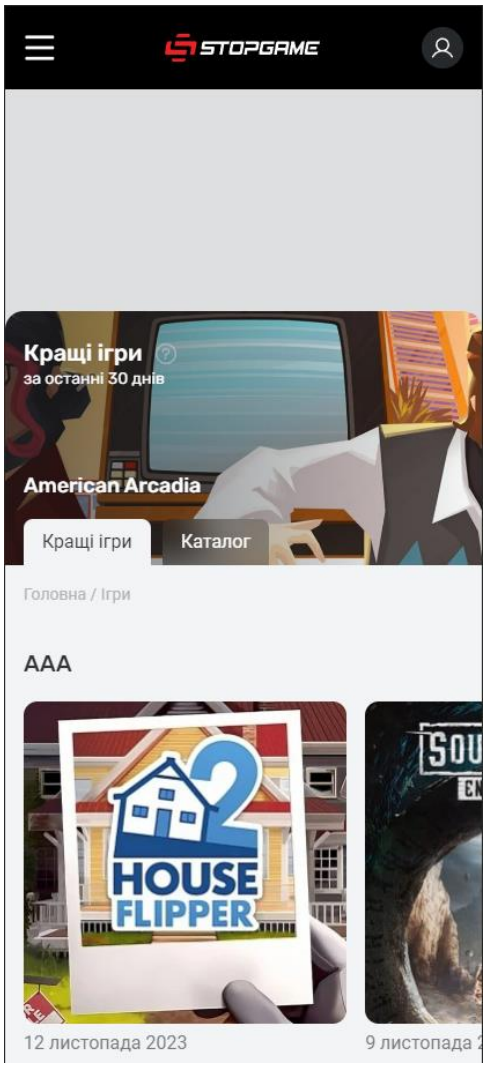

Рисунок 3.4.20 – Адаптивна верстка сторінки ігор

Результати тестування свідчать про те, що розроблений ігровий портал відповідає всім вимогам, що були визначені на етапі проектування. Усі плановані функції були успішно втілені, і веб-сайт підтримує адаптивність, що забезпечує зручний перегляд на різних пристроях.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Темою кваліфікаційної роботи є розробка ігрового порталу з використанням технології Nuxt JS, Vue JS та Elastic Search. Оскільки, розроблений за допомогою технологій пов’язаних із JavaScript портал, реалізовувався з використанням комп’ютерної техніки і розгорнутий на веб-сервері в мережі Інтернет, то важливим було дотримання вимог і правил з охорони праці і техніки безпеки.

Функціями охорони праці є дослідження санітарії та гігієни праці, проведення заходів щодо зниження впливу шкідливих факторів на організм працівників у процесі праці. Основним методом охорони праці є використання техніки безпеки. При цьому вирішуються два основні завдання: створення машин і інструментів, при роботі з якими виключена небезпека для людини, і розробка спеціальних засобів захисту, що забезпечують безпеку людини в процесі праці, а також проводиться навчання працюючих безпечним прийомам праці та використання засобів захисту, створюються умови для безпечної роботи.

Цілі і завдання охорони праці – звести до мінімуму ймовірність нещасних випадків та професійних захворювань працюючих з одночасним забезпеченням нормальних умов праці при її максимальній продуктивності.

Охорона життя і здоров’я людини є пріоритетним напрямком соціальної політики держави. В Україні прийнято закон прямої дії «Про охорону праці», який регламентує захист конституційного права працівників на безпечні умови праці. Законодавство України про охорону праці складається із загальних законів України та спеціальних законодавчих актів. Загальними законами України, що визначають основні положення з охорони праці є Конституція України, Закон України «Про охорону праці», Кодекс законів про працю (КЗпП), Закон України «Про

загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності».

Відповідно до ДСанПіН 3.3.2.007-98 робоче місце працівника не можна розміщувати у підвалах та цокольних поверхах. Площа робочого місця для виконання розробки повинна становити не менше 6м^2, а об’єм 20м^3. Приміщення, в якому працюють на ПК повинно бути забезпечене природнім та штучним освітлення згідно з нормами. Природне освітлення здійснюється крізь світлові отвори, направлені на північний схід і вони забезпечують коефіцієнт природньої освітленості (КПО) не менший ніж 1,5%. Розраховується КПО за методикою, що міститься у ДБН В.2.5-28:2018.

В приміщеннях де проводиться розробка та міститься ПК, штучне освітлення здійснюється системою загального рівномірного освітлення. Джерелом штучного освітлення слугують люмінесцентні лампи. Віконні отвори

приміщення обладнані вертикальними жалюзями та зовнішніми козирками. У даному приміщенні функціонує система опалення. Відповідно до санітарних норм та правил у даному приміщенні знаходиться аптечка першої медичної допомоги. Також, у даному приміщенні щоденно проводиться вологе прибирання.

Згідно вимог НПАОП 0.00-7.15-18 стіни, стеля та підлога приміщень, в яких розміщені комп’ютери, повинні бути виготовлені з матеріалів, дозволених для оформлення приміщень органами державного санітарно-епідеміологічного нагляду.

Відповідно до стандартів, під умовами праці розуміється – це сукупність чинників виробничого середовища трудового процесу, що впливають на здоров'я та працездатність людини в процесі предметної діяльності.

Згідно правил НПАОП 0.00-7.15-18 обладнання та організація робочого місця працюючого з ПК має відповідати ергономічним вимогам ДСТУ 8604:2015, ДСТУ 7299:2013, ДСТУ 7951:2015, а саме:

- оптимальна робоча поза користувача ПК забезпечується конструкцією робочого місця;

- робоче місце розташоване так відносно світлових отворів, що природне світло падає збоку зліва;
- відповідно вимогам, робочий стілець є підйомно-поворотним з регульованою висотою;
- будова робочого стола відповідає вимогам ергономіки та дозволяє оптимально розміщувати на робочій поверхні ПК та допоміжне обладнання.

Для робочої зони приміщення встановлюються оптимальні та допустимі мікрокліматичні умови з урахуванням складності виконуваної роботи та періоду року. При виконанні роботи на ПК, що пов’язано з нервово-емоційним напруженням, у приміщенні потрібно підтримувати температура у межах 22-25 °С, відносну вологість 40-60%. Дані вимоги до параметрів мікроклімату містяться у санітарних нормах ДСН 3.3.6.042-99.

Загальний час роботи з комп’ютером не повинен перевищувати 50% тривалості робочого дня. Якщо виконання роботи пов'язане тільки з використанням комп'ютера, то при неможливості зміни діяльності необхідно робити перерви та паузи. Для робіт, які виконуються з великим навантаженням, слід робити 10 – 15 хвилинну перерву через кожну годину, для мало інтенсивної роботи такі перерви слід робити через 2 години.

Отже, при розробці програмної системи управління зрілістю вимог на стадіях життєвого циклу ПЗ, проаналізовано та враховано необхідні норми щодо охорони праці при використанні електронно-обчислювальної техніки і забезпечено умови для зручної та ефективної роботи працівників.

4.2 Безпека в надзвичайних ситуаціях

При роботі із електронно-обчислювальною машиною (ЕОМ), її оператор, зобов’язаний сумлінно дотримуватись наступних вимог, для уникнення небезпечних ситуацій [22]:

- у всіх випадках виявлення пошкодження проводів електричного живлення, несправності заземлення та інших пошкодженнях електрообладнання, виникненні запаху гарі, диму – негайно вимкнути електричне живлення і повідомити про аварійну ситуацію свого безпосереднього керівника й чергового електрика;
- при попаданні людини під електричну напругу негайно звільнити її від дії струму шляхом вимкнення електричного живлення, до прибуття лікаря надати потерпілому долікарську медичну допомогу;
- при будь-яких випадках порушень роботи технічного обладнання або програмного забезпечення негайно викликати представника технічної служби з питань експлуатації обчислювальної техніки;
- у випадку виникнення різі в очах, різкого погіршення зору, виникнення головного болю, больових відчуттях у пальцях та кистях рук, посилення серцебиття – негайно припинити роботу з використанням ЕОМ, повідомити про те, що сталося, свого безпосереднього керівника й звернутися до медичної установи;
- при загорянні обладнання негайно відключити його від електромережі;
- про загорання повідомити свого безпосереднього керівника, оперативного чергового, пожежну службу;
- ужити заходів щодо ліквідації вогню за допомогою вуглекислотного або порошкового вогнегасника.

Важливим питанням є режим праці і відпочинку при роботі з ЕОМ. Виділяють 7 умов для того, щоб діяльність на робочому місці, оснащеному дисплеєм, здійснювалася без скарг і без втоми [22].

Правильне облаштування робочого столу:

- при фіксованій висоті – оптимальна висота є 720мм;
- повинен забезпечуватися необхідний простір для рук по висоті, ширині і глибині;
- в області сидіння не повинно бути шухляд.

Правильне встановлення робочого стільця:

- висота повинна регулюватися;
- конструкція повинна бути такою, що обертається;
- площа сидіння на 30мм нижче, ніж підколінна западина.

Правильне розташування приладів: необхідно так установити яскравість знаків і яскравість фону дисплея, щоб не було великої відмінності в порівнянні з яскравістю навколишнього оточення, але щоб знаки чітко пізнавалися на відстані читання. Не допускати наступне:

- дуже велику яскравість (викликає мерехтіння);
- дуже слабку яскравість (сильне навантаження на очі);
- дуже чорну фонову яскравість дисплея (сильне навантаження на очі).

Правильне виконання робіт:

- положення тулуба пряме, ненапружене;
- положення голови пряме, вільне, зручне;
- положення рук – зігнуті трохи більше, ніж під прямим кутом;
- положення ніг – зігнуті трохи більше, ніж під прямим кутом;
- правильна відстань для зору, клавіатура і дисплей – приблизно на однаковій відстані для зору; при постійній роботі – близько 500мм, при випадковій роботі – до 700мм.

Правильне освітлення:

- освітлення по можливості із сторони, зліва;
- по можливості – рівномірне освітлення всього робочого простору;

- прилади по можливості встановлювати в місцях, віддалених від вікон;
- вибирати непряме освітлення приміщення або вкривати корпуси світильників;
- світло, що поступає через вікна, пом'якшувати за допомогою штор;
- організувати робоче місце, щоб напрям погляду йшов по можливості паралельно фронту вікон.

Правильне застосування допоміжних засобів: підлокітники використовувати, якщо клавіатура вища 15мм [22].

Правильний метод роботи:

- передбачати по можливості зміну завдань і навантажень;
- дотримувати перерви в роботі: 5 хвилин через 1 годину роботи біля дисплея або 10 хвилин після 2-х годин роботи біля дисплея.

Згідно діючому законодавству, роботодавець повинен створити для працівників безпечні, сприятливі умови праці. Однак, методи та способи організації роботи сучасного роботодавця, особливо на підприємствах недержавної форми власності, не дозволяють розглядати питання охорони праці як першочергові. Це питання залишається на другому місці, тому що, на перший погляд, не сприяють прибутковості організації. При цьому слід відзначити, що міжнародний досвід свідчить – організація праці, при якої інформуються вимоги безпеки і гігієни праці, підриває економічну ефективність установи і не може бути основою для стійкої стратегії розвитку.

ВИСНОВКИ

В даній кваліфікаційній роботі глибоко досліджувалися різноманітні аспекти створення та функціонування ігрового порталу. Ця аналітична та практична робота була спрямована на розгляд технологій, що забезпечують стабільну роботу порталу та сприяють його взаємодії з користувачами.

Один з ключових аспектів цього дослідження – вивчення сучасних ігрових порталів як важливих місць не лише для розваг, але й для соціальної взаємодії ігровій спільноти. Вони стали платформами для обміну досвідом, пошуку нових можливостей та розвитку професійних навичок. Ігровий портал – це віртуальна структура, що об'єднує гравців, створюючи сприятливе середовище для ігрової спільноти.

Цей досліджуваний ігровий портал став результатом глибокого аналізу, відбору оптимальних технологій та розробки відповідного функціоналу. Основними перевагами створеного веб-порталу є зручний користувацький інтерфейс, адаптивність та можливість взаємодії на різних пристроях, а також розділення ролей користувачів.

Розглядаючи різні аспекти функціонування створеного ігрового порталу, варто зазначити, що він включає в сторінку з іграми, розділ новин, блогів та форум. Форум – це місце, де користувачі можуть не лише задавати свої питання, а й отримувати відповіді від інших учасників спільноти. Такий форум створює платформу для обміну думками та досвідом гри.

Вивчення та впровадження різноманітних технологій в досліджуваний ігровий портал дозволили створити не лише платформу для гри, а й простір для спілкування, обміну думками та розвитку в ігровій спільноті. Підсумовуючи, реалізація ігрового порталу була успішною та ефективною, забезпечивши підтримку функцій та алгоритмів для задоволення потреб користувачів у віртуальному ігровому середовищі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Beaird, Jason. "The Principles of Beautiful Web Design." - SitePoint, 2014. - 220 p.

2. Problogger: онлайн-ресурс професійного блогінгу.: веб-сайт. URL: https://problogger.com/ (дата звернення 01.10.2023).

3. Legalaid – Електронна комерція та онлайн продажі : веб-сайт. URL: https://www.legalaid.gov.ua/publikatsiyi/elektronna-komertsiya-ta-onlajn-prodazhi/ (дата звернення 01.10.2023).

4. Cases.Media – Портфоліо : веб-сайт. URL: https://cases.media/article/portfolio-z-chogo-pochati-ta-yak-rozvivati (дата звернення 04.10.2023).

5. Economy-Pedia – Брошура – що це таке? : веб-сайт. URL: https://uk.economy-pedia.com/11033278-user-information (дата звернення 04.10.2023).

6. Вікіпедія – Новинний журнал : веб-сайт. URL: https://en.wikipedia.org/wiki/News_magazine (дата звернення 08.10.2023).

7. FutureNow – Що таке соціальні мережі? : веб-сайт. URL: https://futurenow.com.ua/shho-take-sotsialni-merezhi-vydy-klasyfikatsiya-bezpeka/ (дата звернення 08.10.2023).

8. Teachmint – Освітні сайти : веб-сайт. URL: https://www.teachmint.com/glossary/e/educational-sites/ (дата звернення 11.10.2023).

9. Liferay – Що таке веб-портал? : веб-сайт. URL: https://www.liferay.com/resources/l/web-portal (дата звернення 11.10.2023).

10. Monocubed – What is a Web Portal? : веб-сайт. URL: https://www.liferay.com/resources/l/web-portal (дата звернення 14.10.2023).

11. Vue JS : веб-сайт. URL: https://vuejs.org/ (дата звернення 14.10.2023).

12. React : веб-сайт. URL: https://uk.reactjs.org/ (дата звернення 16.10.2023).

13. Angular : веб-сайт. URL: https://angular.io/ (дата звернення 16.10.2023).

14. Kinsta – What Is Nuxt.js? : веб-сайт. URL: https://kinsta.com/knowledgebase/nuxt-js/ (дата звернення 19.10.2023).

15. Intuz – 5 Reasons Why You Should Use Next.JS : веб-сайт. URL: https://www.intuz.com/blog/5-reasons-why-you-should-use-next.js-for-your-front-end-development (дата звернення 19.10.2023).

16. Flexiple – An Intro to ExpressJS : веб-сайт. URL: https://flexiple.com/express-js/deep-dive (дата звернення 23.10.2023).

17. Knowi – Elasticsearch: What It Is, How It Works, And What It's Used For : веб-сайт. URL: https://www.knowi.com/blog/what-is-elastic-search/ (дата звернення 24.10.2023).

18. Intuji – What Is Algolia? : веб-сайт. URL: https://intuji.com/what-is-algolia/ (дата звернення 25.10.2023).

19. Sematext – What Is Apache Solr? : веб-сайт. URL: https://sematext.com/guides/solr/ (дата звернення 26.10.2023).

20. MySQL : веб-сайт. URL: https://www.mysql.com/ (дата звернення 30.10.2023).

21. Jetbrains – About DataGrip : веб-сайт. URL: https://www.jetbrains.com/datagrip/ (дата звернення 02.11.2023).

22. Навчальний посібник «ТЕХНОЕКОЛОГІЯ ТА ЦИВІЛЬНА БЕЗПЕКА. ЧАСТИНА «ЦИВІЛЬНА БЕЗПЕКА»» / автор-укладач В.С. Стручок– Тернопіль: ФОП Паляниця В. А., – 156 с.

ДОДАТКИ

Додаток А.

Тези представлені на конференції зображено на рисунку А.1.

УДК 004.41
В. Шуст, Ю. Стоянов, канд. техн. наук
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РОЗРОБКА ІГРОВОГО ПОРТАЛУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ NUXT JS, VUE JS ТА ELASTICSEARCH

V. Shust, Y. Stoyanov, Ph.D.
GAME PORTAL DEVELOPMENT USING NUXT JS, VUE JS AND ELASTICSEARCH TECHNOLOGIES

Сучасна ділова сфера неможлива без інтенсивного використання комп'ютерних технологій та інформаційних систем. Управління підприємствами вимагає більш широкого застосування інформаційних технологій, а саме розробки ефективних платформ. Створення ігрової платформи використовуючи технології Nuxt JS і ElasticSearch є актуальним завданням, що відповідає вимогам ринку та сучасним трендам в IT-сфері.

Ця робота спрямована на розробку ігрової платформи, яка відповідає сучасним вимогам користувачів у швидкості, ефективності та функціональності. Використання технології Nuxt JS дозволяє створювати веб-додатки з високою продуктивністю та інтерактивністю, що є важливим для ігрових платформ. ElasticSearch забезпечує швидкий та потужний пошук інформації, що є ключовим аспектом для ігрових середовищ.

Однією з ключових переваг розробки ігрової платформи на базі Nuxt JS і ElasticSearch є їхні унікальні можливості в оптимізації продуктивності, забезпеченні гнучкості та стійкості платформи. Це дозволяє створити інтерактивне та динамічне ігрове середовище для користувачів у відповідності з сучасними вимогами ринку.

Паралельно з цим, важливим аспектом використання Nuxt JS є його можливості управління сторінками та стейтами додатків, що сприяє швидкій реакції та оптимізації завантаження контенту для кінцевих користувачів. ElasticSearch, з свого боку, відкриває безліч можливостей у реалізації продуктивного та точного пошуку, що є критичним для ігрових платформ у забезпеченні найкращого користувацького досвіду.

Враховуючи стрімкий розвиток цифрових технологій, важливо підкреслити, що розробка ігрової платформи на основі Nuxt JS і ElasticSearch відповідає вимогам не лише ефективності та продуктивності, але й високим стандартам безпеки даних. Це забезпечить користувачам надійне та захищене середовище для ігор та взаємодії, що стає важливим аспектом у сучасному цифровому світі.

Отже, дана дипломна робота розглядає сутність, можливості та переваги розробки ігрової платформи з використанням технологій Nuxt JS і ElasticSearch в сучасному інформаційному середовищі.

Література

1. Hanchett, Erik, and Listwon, Benjamin. "Vue.js in Action." - Manning Publications, New York, 2018. - 352 p.

2. Gormley, Clinton, and Tong, Zachary. "Elasticsearch: The Definitive Guide." - O'Reilly Media, Sebastopol, 2015. - 724 p.

3. Kyriakidis, Alex, Maniatis, Kostas, and Buivydas, Gediminas. "The Majesty of Vue.js." - Packt Publishing, Birmingham, 2019. - 312 p.

Рисунок А.1 – Тези конференції