МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: Соціальна мережа створення і поширення гумористичного контенту

Здобувача групи ІТ-01 Алексєєв Дмитро Сергійович

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Дмитро АЛЕКСЄЄВ
(підпис)

Керівник доц. к. т. н., НЕНЯ Анна _____
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В. о. зав. кафедри ІТ

Світлана ВАЩЕНКО

«__» _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Алексєєву Дмитру Сергійовичу

1 Тема роботи Соціальна мережа створення і поширення гумористичного контенту

керівник роботи *НЕНЯ Анна к. т. н., доц.* _____,

затверджені наказом по університету від « *07* » травня 2024 р. №0482-VI

2 Строк подання студентом роботи « 26 » травня 2024 р.

3 Вхідні дані до роботи

Завдання на кваліфікаційну роботу бакалавра

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Огляд останніх досліджень і публікацій, Аналіз програмних продуктів аналогів, Мета та задачі дослідження, Моделювання, Архітектура додатку, Проектування бази даних, програмна реалізація, використання розробленого продукту

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Проблематика предметної області, Мета, Обрані аналоги для аналіз Функціональні вимоги, Засоби розробки, Функціональне моделювання, Діаграма варіантів використання, Структура проекту, База даних, Структура back-end Структура front-end, Візуальний стиль сторінок, Особливості створеного продукту, Висновки

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання *20.04.2024*

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	20.04.2024-24.04.2024	
2	Огляд продуктів аналогів	25.04.2024-28.04.2024	
3	Постановка задачі	29.04.2024-30.04.2024	
4	Визначення засобів реалізації	1.05.2024-2.05.2024	
5	Моделювання та проектування	3.05.2024-4.05.2024	
6	Розробка додатку	5.05.2024-17.05.2024	
7	Тестування	18.05.2024-19.05.2024	
8	Складання звіту	20.05.2024-26.05.2024	

Студент _____ АЛЕКСЄЄВ Дмитро
(підпис)

Керівник роботи _____ к. т. н., доц. НЕНЯ Анна
(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Соціальна мережа створення і поширення гумористичного контенту».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 14 найменувань, додатків. Загальний обсяг роботи − 58 сторінок, у тому числі 43 сторінки основного тексту, 2 сторінки списку використаних джерел, 15 сторінок додатків.

Актуальність даної роботи обумовлена потребою в створенні сучасного, продуктивного та масштабованого веб-додатку для обміну жартами, який відповідає зростаючим вимогам користувачів та тенденціям розвитку цифрових технологій. Мета роботи: створення інноваційної соціальної мережі.

В роботі представлені результати аналізу стану розвитку технологій розроблення та впровадження соціальних мереж, огляд аналогічних додатків. Результати моделювання роботи додатку представлені у вигляді діаграм структурно-функціонального моделювання в нотації IDEF0 та діаграм варіантів використання в нотації UML, діаграмами класів, діяльності.

Для ефективного використання розробленого веб-додатку рекомендується регулярно оновлювати його для забезпечення безпеки, продуктивності та відповідності сучасним технологічним стандартам. Важливим аспектом є моніторинг активності користувачів, що дозволяє виявляти їх потреби та впроваджувати нові функції для покращення користувацького досвіду. Ефективна модерація контенту є ключовим інструментом для підтримки безпечного та приємного середовища, запобігання розповсюдженню небажаного або образливого матеріалу.

Ключові слова: База даних, жарт, соціальна мережа, React, .NET.

ЗМІСТ

ВСТУП 5
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 6
1.1 Огляд останніх досліджень і публікацій 6
1.2 Аналіз програмних продуктів аналогів 8
1.3 Мета та задачі дослідження 10
2. МОДЕЛЮВАННЯ ДОДАТКУ 12
2.1 Структурно-функціональне моделювання 12
2.2 Архітектура додатку 20
2.3 Проектування бази даних 22
3. ПРАКТИЧНА РЕАЛІЗАЦІЯ 23
3.1 Програмна реалізація 23
3.1.1 Back-end розробка. 23
3.1.2 Front-end розробка 31
3.2 Настанови з використання розробленого продукту 36
ВИСНОВКИ 39
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 41
ДОДАТОК А 43
ДОДАТОК Б 53

ВСТУП

В сучасному цифровому суспільстві соціальні мережі відіграють ключову роль у спілкуванні, обміні інформацією та розвагах. Однак наявний ринок соціальних мереж часто виявляється насиченим та здебільшого одноманітним. У рамках цієї роботи досліджується ідея створення новаторської соціальної мережі, яка спрямована на підвищення емоційного благополуччя користувачів через гумор та розважальний контент.

Проєкт, не обмежується лише обміном жартами, але створює унікальний екосистемний підхід, де гумор і соціальні взаємодії взаємодоповнюють один одного. Соціальна мережа пропонує інноваційні можливості для вираження творчості, обміну веселими ідеями та побудови активної спільноти, яка сприяє позитивному емоційному впливу на кожного користувача.

У даній роботі будуть досліджені технічні та функціональні аспекти створення соціальної мережі, включаючи архітектуру системи та механізми захисту приватності.

Мета даної роботи – створення соціальної мережі створення і поширення гумористичного контенту

Основні задачі: аналіз останніх досліджень і публікацій, огляд аналогів, постановка задачі, моделювання та проектування, розробка додатку, тестування.

Об’єкт дослідження – соціальні мережі поширення контенту.

Предмет дослідження – розробка соціальної мережі створення і поширення гумористичного контенту

Практична цінність дослідження полягає у підвищенні емоційного благополуччя користувачів через гумор через використання соціальної мережі для обміну жартами та іншим гумористичним контентом.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

На основі останніх досліджень і публікацій можна виділити кілька ключових підходів та технологій, які можуть бути найкращими для створення соціальних мереж. Розглянемо ці підходи детальніше.

Автоматизовані алгоритми: Одним із основних підходів є використання автоматизованих алгоритмів для модерації контенту. Такі алгоритми використовують машинне навчання та штучний інтелект для виявлення та фільтрування небажаного контенту, спаму або дезінформації. Наприклад, TikTok та Instagram активно застосовують алгоритми для аналізу текстів, зображень і відео з метою швидкого виявлення порушень правил [1].

Людський фактор: Проте, як зазначають дослідники [2-4], людське втручання залишається критичним. Алгоритми можуть не завжди правильно інтерпретувати контекст або нюанси мови, що вимагає участі модераторів для прийняття остаточних рішень у складних або суперечливих випадках. Людська перевірка важлива для забезпечення справедливості та точності модерації.

Прозорі алгоритми: Автор праці [4] акцентує увагу на важливості прозорості алгоритмів. Соціальні мережі повинні забезпечувати користувачів зрозумілими правилами модерації та поясненнями щодо роботи алгоритмів. Це допомагає підвищити довіру користувачів до платформи та запобігти обвинуваченням у цензурі.

Механізми оскарження: Важливим аспектом є наявність механізмів для оскарження рішень модераторів. Це дозволяє користувачам звертатися з проханням про перегляд випадків видалення контенту або блокування акаунтів, що сприяє справедливості та підвищує задоволеність користувачів [5].

Шифрування та аутентифікація: Одним із ключових аспектів є забезпечення безпеки та приватності користувачів. Використання шифрування для передачі даних та двофакторної аутентифікації допомагає захистити особисту інформацію користувачів від несанкціонованого доступу [6].

Захист від кіберзагроз: Соціальні мережі повинні впроваджувати заходи для захисту від кіберзагроз, таких як фішинг, злом акаунтів та витоки даних. Це включає регулярне оновлення безпекових протоколів та моніторинг активності на платформі.

Аналіз поведінки користувачів: Для залучення та утримання аудиторії важливо аналізувати поведінку користувачів на платформі. Це дозволяє створювати персоналізований контент, який відповідає інтересам та потребам користувачів [8]. Алгоритми рекомендацій можуть допомогти користувачам знаходити релевантний контент, підвищуючи їхню взаємодію з платформою.

Інтерактивність та соціальна взаємодія: Функції групових чатів, форумів та спільнот сприяють соціальній взаємодії та підтримці активності користувачів. Це дозволяє створювати більш згуртовані та активні спільноти, які підтримують довготривалу взаємодію з платформою.

Адаптивні механізми: Соціальні мережі повинні враховувати культурні особливості користувачів. Використання адаптивних механізмів, таких як локалізація контенту та співпраця з культурними консультантами, допомагає забезпечити культурну чутливість та уникнути непорозумінь [9-10].

Глобальна адаптація: Адаптація платформ до глобальної аудиторії включає не тільки переклад інтерфейсу, але й врахування різних культурних норм та очікувань. Це допомагає створювати більш інклюзивне середовище для користувачів з різних куточків світу.

На основі розглянутих аспектів розроблення ефективної соціальної мережі можна зробити висновок про важливість інтеграції передових технологій автоматизації модерації, забезпечення прозорості та справедливості у прийнятті рішень, впровадження надійних заходів безпеки та приватності, аналіз поведінки користувачів для персоналізації контенту та враховування культурних

особливостей. Ці підходи допоможуть створити безпечне, інклюзивне та привабливе середовище для користувачів, що підвищить їхню взаємодію та задоволеність платформою.

1.2 Аналіз програмних продуктів аналогів

Розглянувши програмні продукти Instagram, Facebook, Reddit та Twitter, можна визначити їхні сильні та слабкі сторони. Instagram [11] (рис. 1.1) славиться своїм візуальним спрямуванням, зручним інтерфейсом та великою активною аудиторією, хоча має обмежений функціонал щодо обробки та взаємодії з контентом.

Рисунок 1.1 – Головна сторінка Instagram

Джерело: [11]

Facebook [12] (рис. 1.2), натомість, має широкий функціонал, що включає створення різноманітного контенту, груп, подій та прямих ефірів, але часто стикається з проблемами з приватністю та має складний інтерфейс.

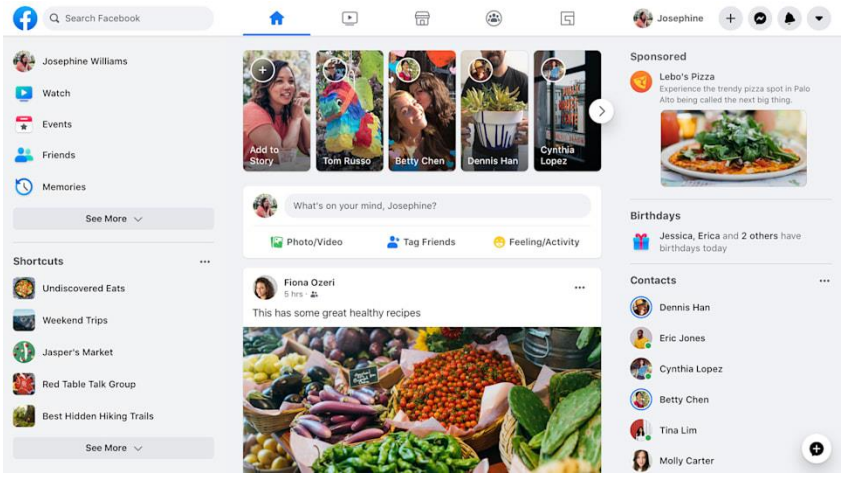


Рисунок 1.2 – Головна сторінка Facebook

Джерело: [12]

Reddit [13] (рис. 1.3) відомий своєю можливістю обговорення різноманітних тем та глибоким підключенням до спільнот, хоча й не має інтегрованого відео- та фото контенту.

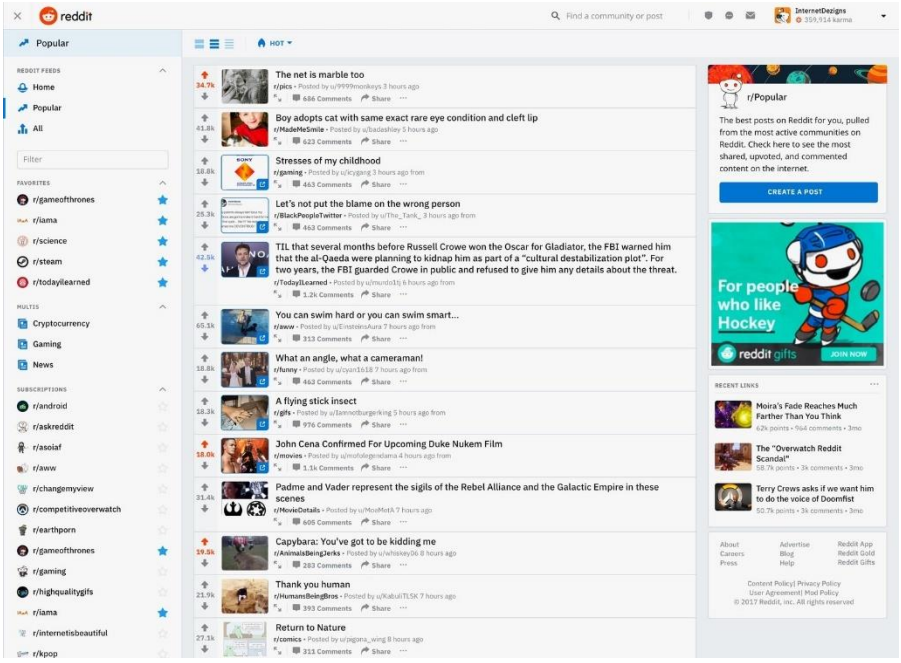


Рисунок 1.3 – Головна сторінка Reddit

Джерело: [13]

У свою чергу, Twitter (рис. 1.4) [14] відрізняється миттєвим оновленням інформації та широким поширенням контенту, але має обмежений об'єм текстових повідомлень та високий рівень спаму. Оцінка сильних та слабких сторін кожної платформи допоможе нам використати візуальну складову Instagram, функціонал Facebook, спільноти Reddit та роботу з інформацією Twitter в нашому проекті.

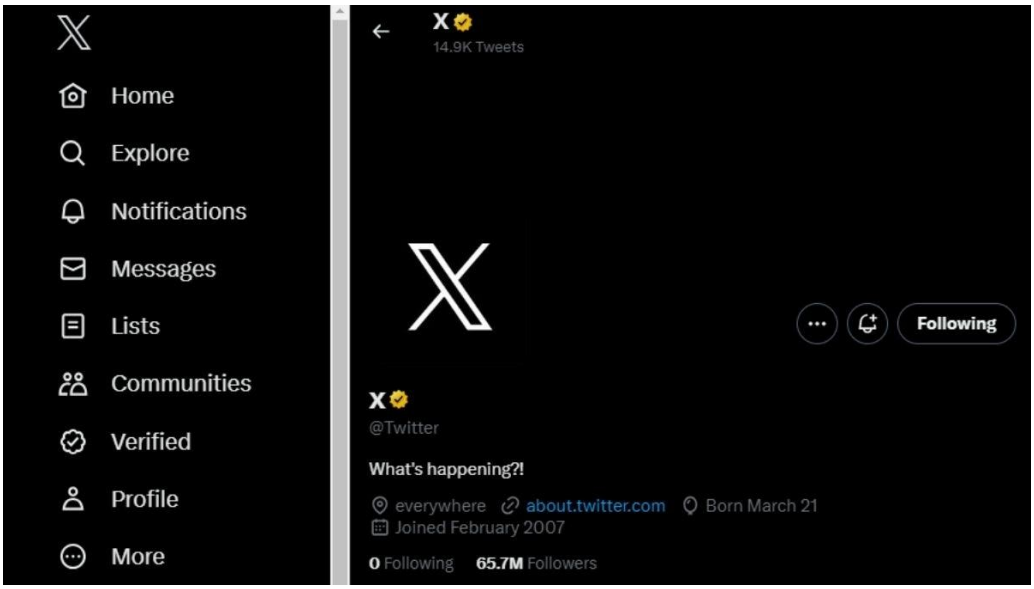


Рисунок 1.4 – Головна сторінка Twitter

Джерело: [14]

1.3 Мета та задачі дослідження

Основна мета проекту полягає у створенні інноваційної соціальної мережі, яка надасть користувачам унікальний та позитивний досвід спілкування через гумористичний контент. Для досягнення цієї мети треба вирішити наступні завдання:

— Проаналізувати сучасні технології розроблення ефективної соціальної мережі.

- Провести огляд програмних аналогів з метою визначення їх сильних сторін і використання їх.
- Розробити структурно-функціональні моделі вебдодатку соціальної мережі.
- Розробити механізми соціальної взаємодії користувачів.
- Створити механізм, який дозволяє користувачам легко повідомляти про неприйнятний контент для подальшого аналізу модераторами.
- Забезпечити можливість налаштовування рівня конфіденційності та забезпечити високий рівень безпеки особистої інформації користувачів.

Основні вимоги до проекту включають в себе ефективну модерацію контенту, забезпечення безпеки та конфіденційності, а також високий рівень персоналізації гумористичного контенту для задоволення потреб різних користувачів.

2. МОДЕЛЮВАННЯ ДОДАТКУ

2.1 Структурно-функціональне моделювання

Моделювання складається з етапів, які пов’язані між собою. Процес моделювання починається з абстрактної концептуальної схеми, після неї створюються логічна та фізична моделі.

Комплексність проекту вимагає системного підходу до аналізу, і IDEF0 надає ефективний засіб для покрокового розгляду кожної функції системи. Це дозволяє виявити ключові аспекти, визначити послідовність дій та виявити можливості оптимізації.

Функціональне моделювання в нотації IDEF0 представлене на рисунку 2.1.

Рисунок 2.1 – IDEF0

Джерело: [Розроблено автором]

В якості вхідних даних виступає запит на реєстрацію і додавання контенту від користувача. Далі в процесі використання застосовується ком'ютер, програмне забезпечення та інтернет, а також беруться до уваги правила платформи,

налаштування конфіденційності і стандарти безпеки і в результаті отримуємо готовий продукт

Декомпозиція дозволяє розбити велику та складну систему на менші, керовані елементи. Кожна функція може бути самостійною одиницею роботи, що полегшує розуміння та управління. Декомпозиція функціональної моделі представлена на рисунку 2.2

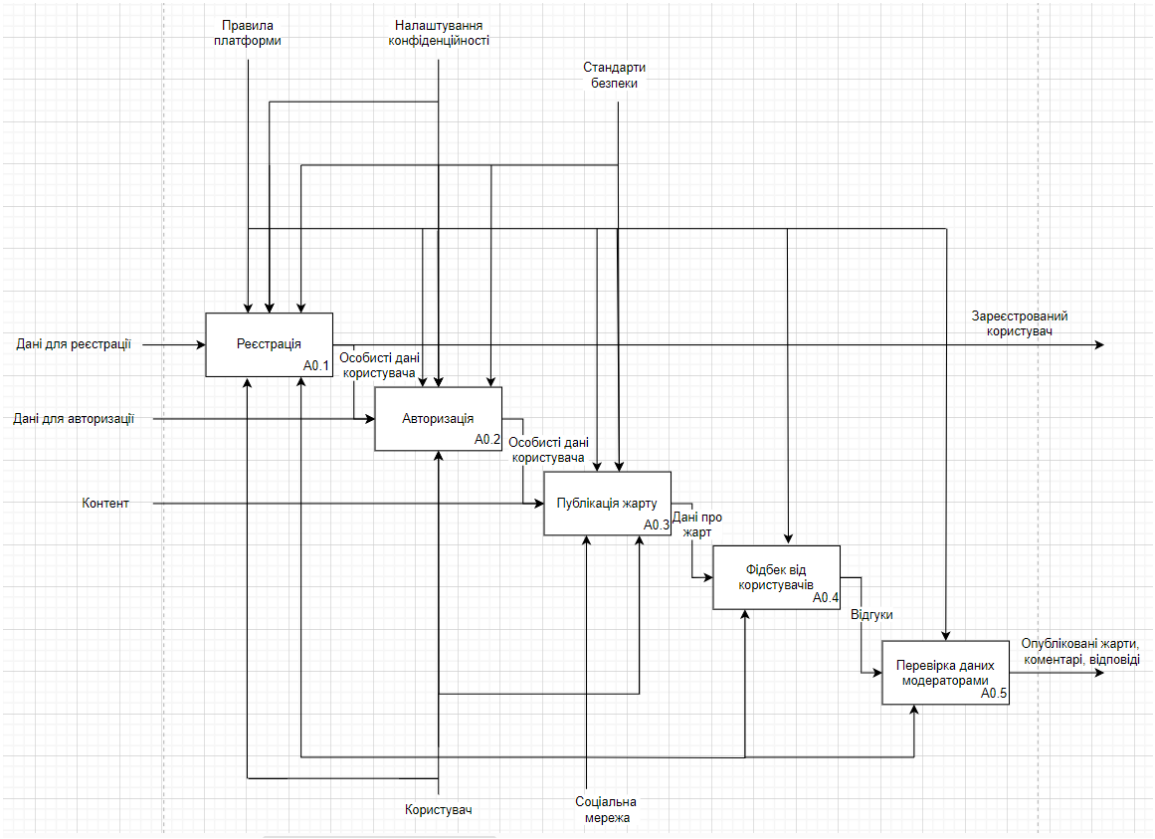


Рисунок 2.2 – Декомпозиція функціональної моделі

Джерело: [Розроблено автором]

На прикладі запиту користувача можна пройти всіма етапами роботи вебдодатку і врахувати ресурси, що використовуються на кожному етапі

Діаграма варіантів використання допомагає чітко визначити всі можливі способи взаємодії користувачів з системою. Кожен варіант представляє конкретний сценарій використання, що допомагає визначити потреби користувачів. Діаграма варіантів використання в нотації UML представлена на рисунку 2.3

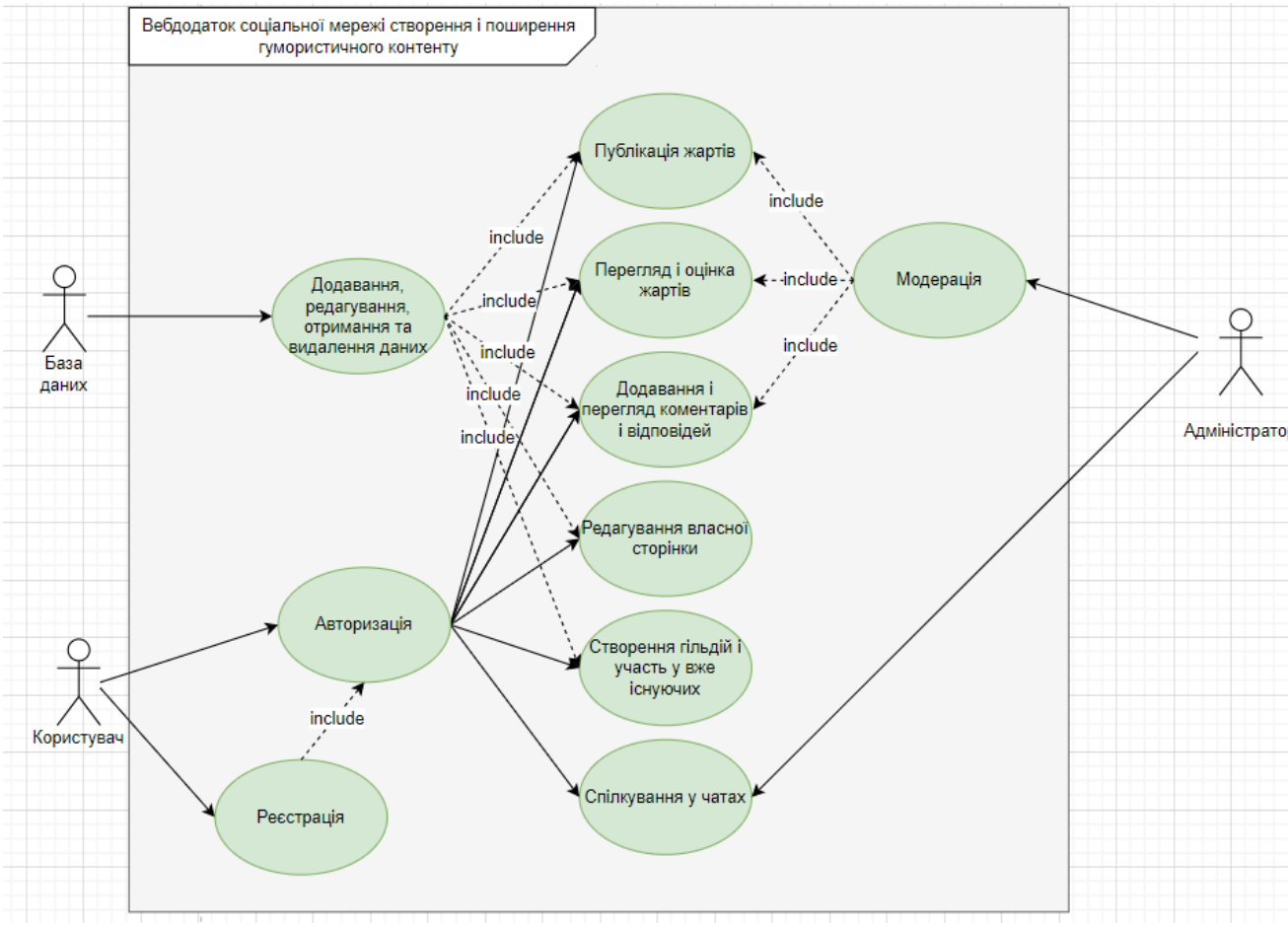


Рисунок 2.3 – Діаграма варіантів використання

Джерело: [Розроблено автором]

Використання діаграми класів у фазі аналізу дозволяє здійснити розбір логіки та структури системи, забезпечуючи ефективну підготовку до подальшого проектування та реалізації. Діаграма класів представлена на рисунку 2.4.

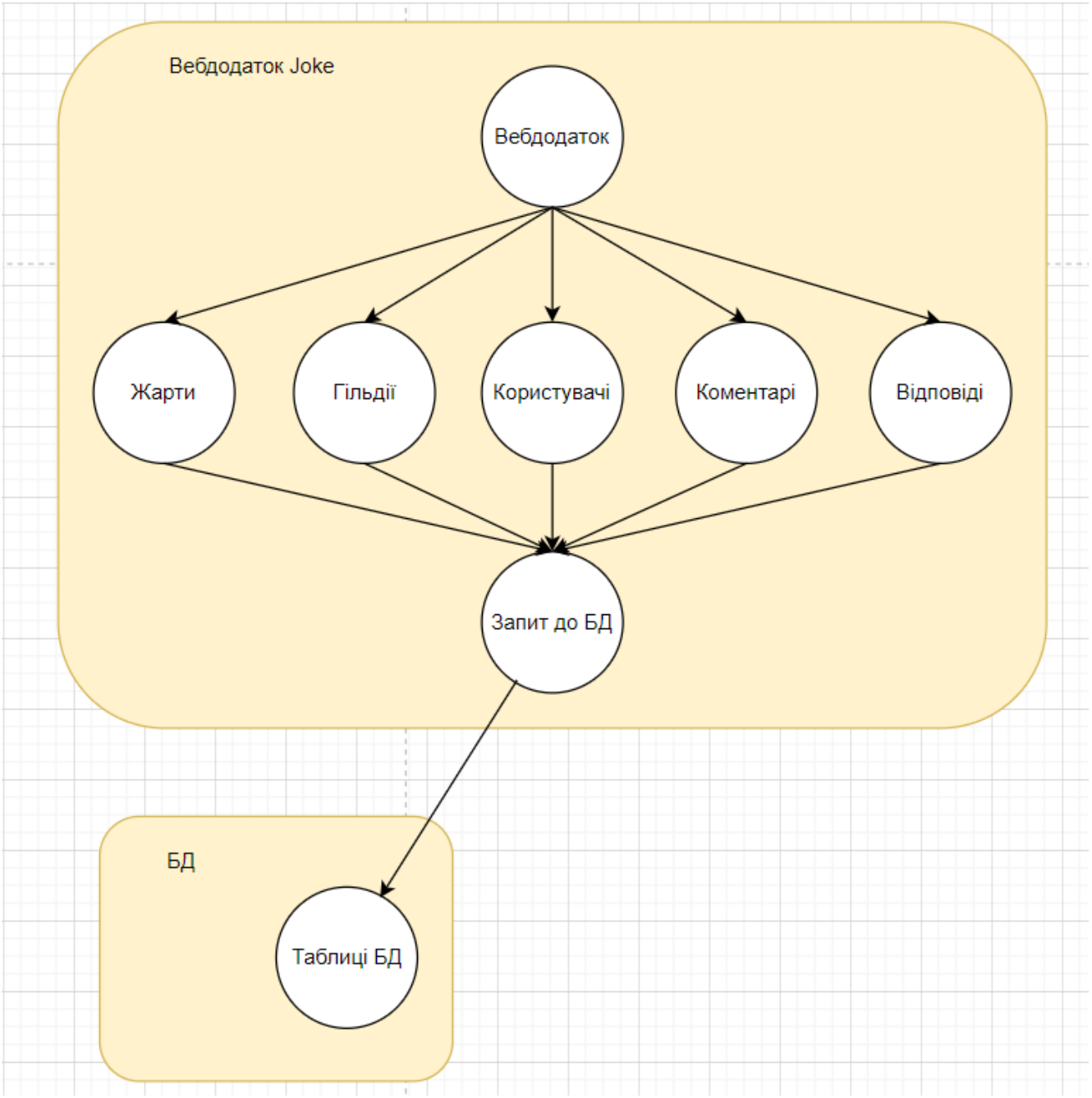


Рисунок 2.4 – Діаграма класів

Джерело: [Розроблено автором]

Діаграма діяльності дозволяє моделювати послідовність дій та процесів в системі. Це особливо корисно для візуалізації взаємодій між різними елементами соціальної мережі. На рисунку 2.5 представлена діаграма діяльності.

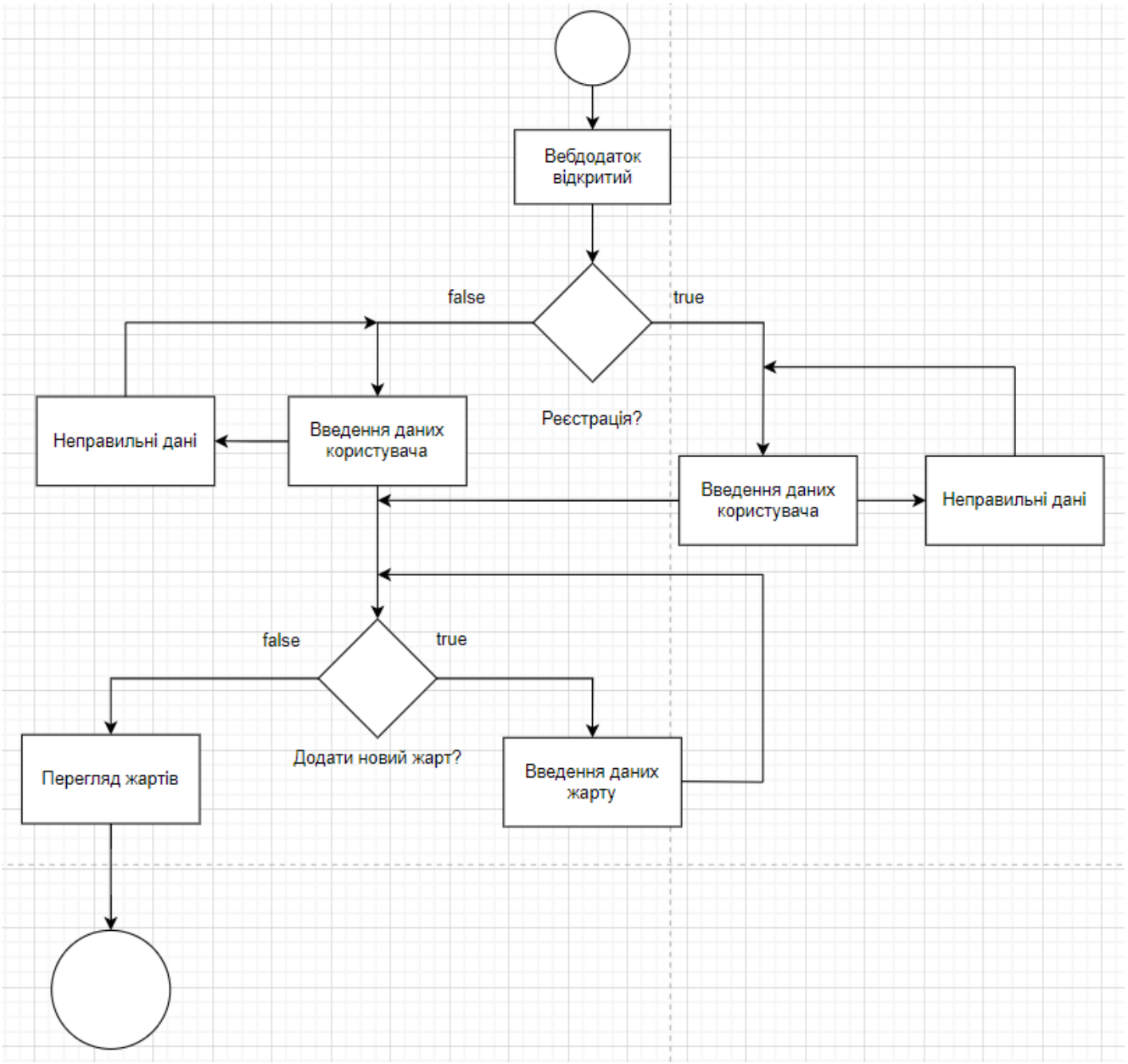


Рисунок 2.5 – Діаграма діяльності

Джерело: [Розроблено автором]

На діаграмі представлений типовий сценарій роботи з web-додатком

На рисунку 2.6 представлена діаграма діяльності створення даних

Рисунок 2.6 – Діаграма діяльності створення даних

Джерело: [Розроблено автором]

На діаграмі представлений спрощений сценарій створення даних

На рисунку 2.7 представлена діаграма діяльності редагування даних.

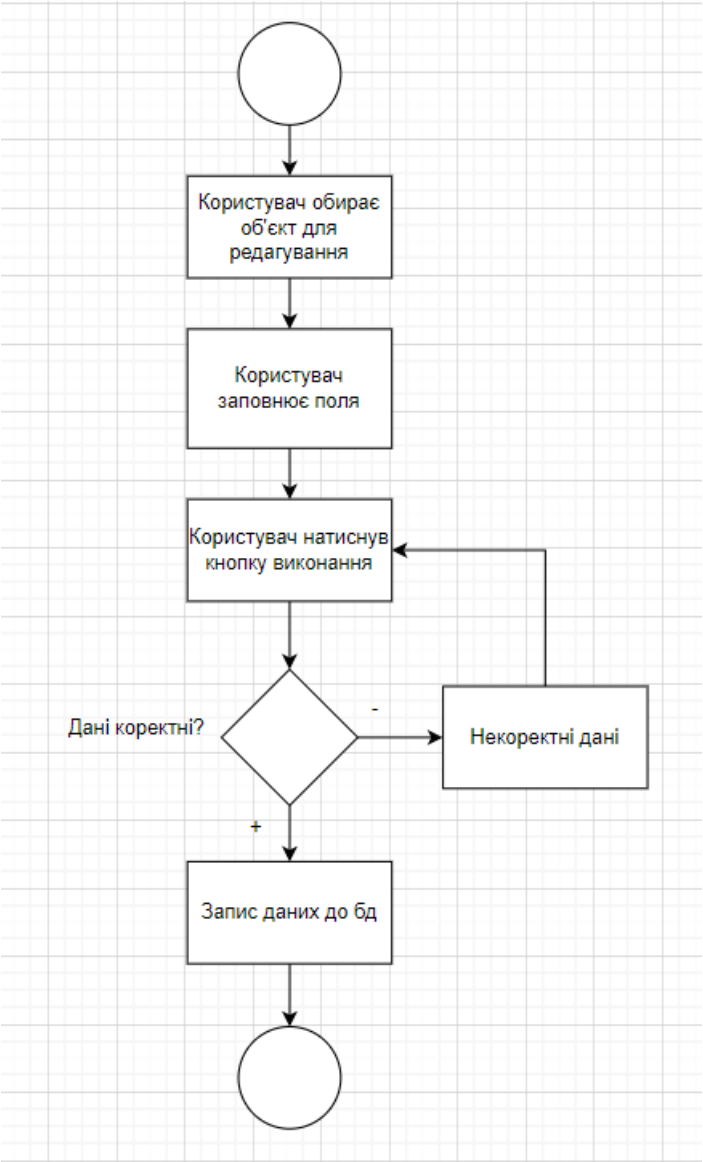


Рисунок 2.7 – Діаграма діяльності «Редагування інформації»

Джерело: [Розроблено автором]

На діаграмі представлений спрощений сценарій редагування даних

Діаграма розгортання дозволяє візуалізувати та моделювати інфраструктуру системи, вказуючи на сервери, бази даних, мережеві з'єднання та інші компоненти. На рисунку 2.8 представлена діаграма розгортання

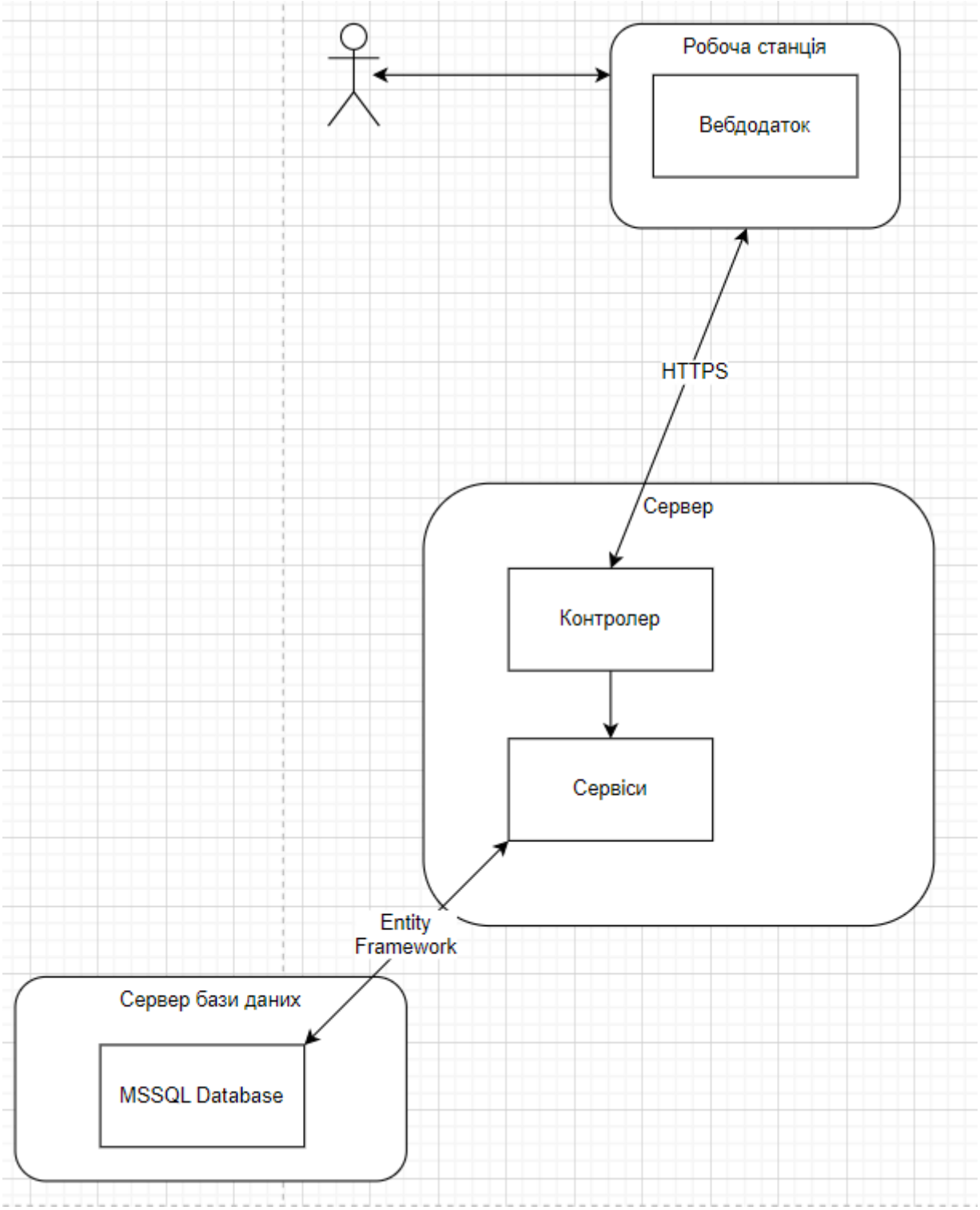


Рисунок 2.8 – Діаграма розгортання

Джерело: [Розроблено автором]

На діаграмі висвітлено принцип взаємодії клієнта і сервера.

2.2 Архітектура додатку

Архітектура вебдодатку складається з клієнтської частини, реалізованої на основі бібліотеки React, та серверної частини, розробленої за допомогою ASP.NET API. Вебдодаток, який ми створюємо, поєднує в собі сучасні технології фронтенд-розробки та надійні інструменти для створення бекенду, що дозволяє забезпечити високу продуктивність, масштабованість та зручність у використанні.

Клієнтська частина на React забезпечує інтерактивний та динамічний користувацький інтерфейс. React дозволяє створювати компоненти, які легко переробляти та повторно використовувати, що сприяє швидкому розвитку та підтримці додатку. Використання сучасних підходів, таких як односпрямований потік даних та віртуальний DOM, дозволяє React ефективно обробляти зміни в інтерфейсі, забезпечуючи високу швидкість роботи додатку.

Серверна частина на основі ASP.NET API відповідає за обробку запитів від клієнтської частини та взаємодію з базою даних. ASP.NET надає потужні інструменти для створення масштабованих веб-служб, які можуть обробляти великий обсяг даних та підтримувати високу навантаженість. Використання RESTful архітектури дозволяє створити зрозумілий та стандартизований інтерфейс для взаємодії між клієнтом та сервером. На рисунку 2.9 представлено схематичне зображення роботи додатку.

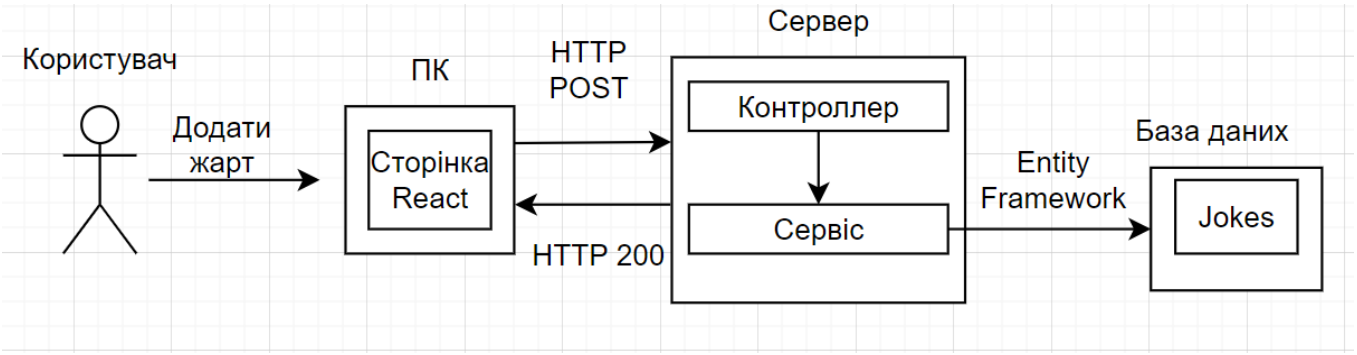


Рисунок 2.9 – Схема роботи додатку

Джерело: [Розроблено автором]

Можна стверджувати, що додаток відповідає архітектурному стилю REST, оскільки він дотримується наступних обмежень:

— Клієнт-серверна архітектура.

  В додатку обов’язково повинна розділятися відповідальність між компонентами, які виконують обчислення і збереження даних та компонентами які займаються відображенням даних на сторінці.

— Відсутність стану.

  В запиті до сервера повинна надаватись вся необхідна інформація для обробки, тобто сервер не повинен знати нічого з попередніх запитів.

— Кешування

  За необхідності, дані які передаються сервером повинні зберігатися на стороні клієнта для того, щоб уникнути зайвих запитів

— Однорідний інтерфейс

  Всі компоненти сервера повинні підтримувати однорідний інтерфейс. Це надасть можливість швидко замінювати їх за необхідності та спростить роботу.

— Шари абстракції

  Будь-яку складну систему необхідно розділяти на простіші частини, які приховують особливості реалізації та працюють незалежно один від одного

Для зв’язку з базою даних використовується Entity Framework Core. Він дає можливість працювати з базами даних без необхідності написання SQL запитів, а також виконує роботу по створенню, оновленню, видаленню та редагуванню структурних елементів автоматично, за допомогою інструменту міграцій.

2.3 Проектування бази даних

База даних повинна мати 5 таблиць, по одній на кожну сутність додатку. Таблиця Guild зберігає інформацію про гільдії. Особисті дані користувачів та посилання на гільдію зберігаються в User. Таблиця Joke призначена для зберігання жартів та посилання на користувача, який опублікував його. Коментарі і посилання на жарти та користувачів зберігаються в таблиці Comment. В таблиці Reply зберігається посилання на жарт, коментар та користувача, що залишив відповідь, а також на текст відповіді.

Логічна модель даних представлена на рисунку 2.10.

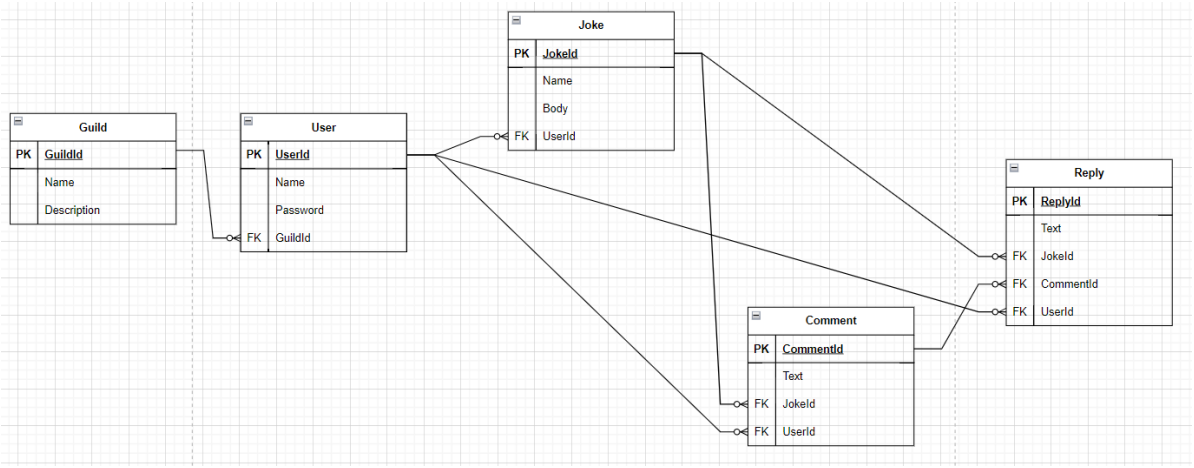


Рисунок 2.10 – Логічна модель даних

Джерело: [Розроблено автором]

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Програмна реалізація

3.1.1 Back-end розробка.

Основою додатку стане його серверна частина, яка буде виконувати основну роботу по обчисленню та зберіганню даних, тому розробку необхідно розпочинати з неї. Першим етапом розробки є створення необхідних сутностей, які стануть таблицями майбутньою бази даних. Для цього створюємо папку Model і додаємо до неї 5 класів сутностей моделі та один клас, який допоможе з шифруванням даних користувачів (рисунок 3.1)

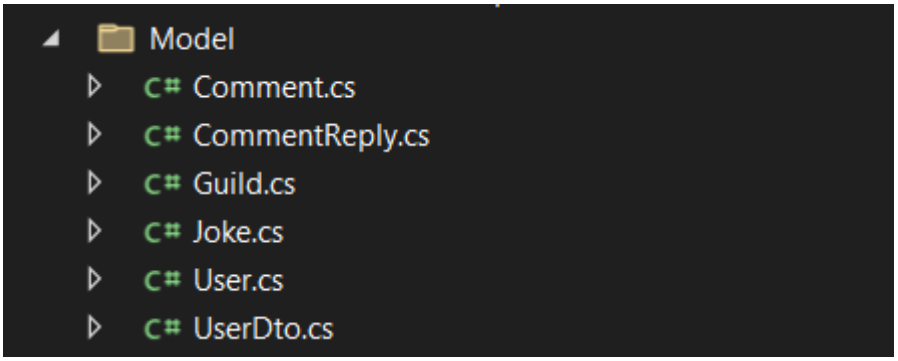


Рисунок 3.1 – Класи папки Model

Джерело: [Розроблено автором]

Розглянемо різницю між класами UserDto та User. Абревіатура DTO (Data transfer object) означає, що задача цього класу - отримувати дані від користувача для їх подальшої обробки. Відповідно до цього в класі UserDto є тільки 2 поля: імені користувача та пароля. Але оскільки в базу даних необхідно зберігати зашифровані дані, необхідний також клас, поля в якому будуть відповідати рядкам таблиці бази даних. Для цього створюється клас User (рисунок 3.2)

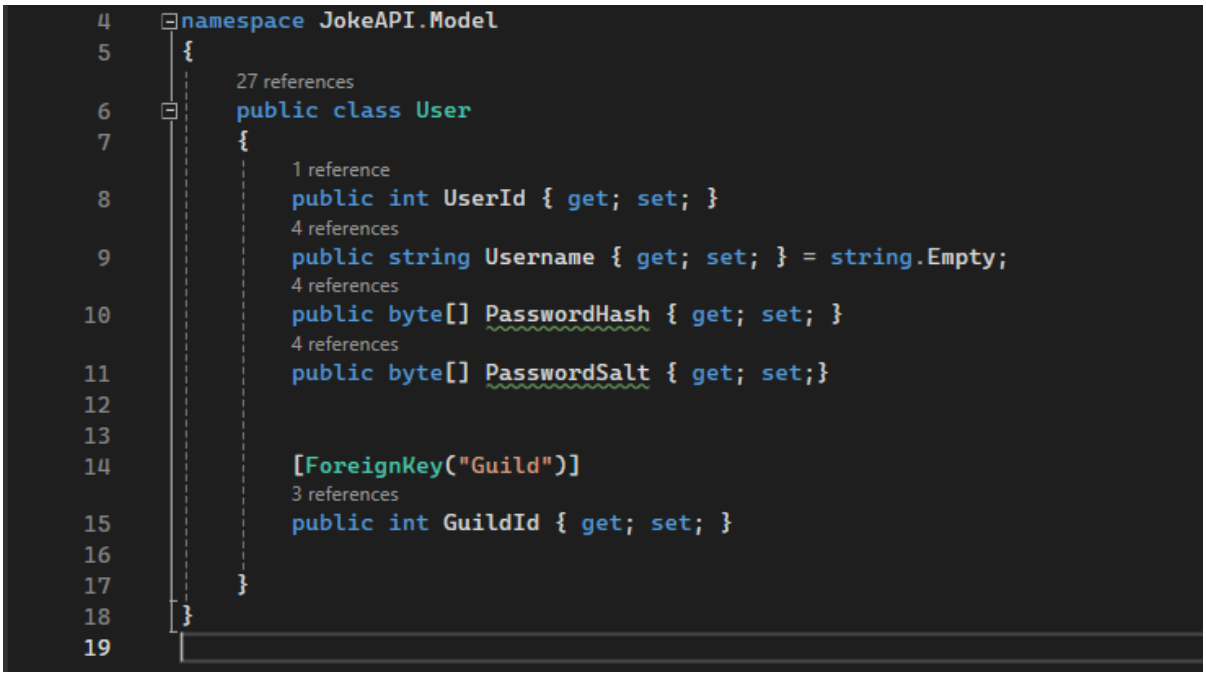


Рисунок 3.2 – Клас User

Джерело: [Розроблено автором]

В цьому класі є поле для номеру користувача в базі даних, поле імені користувача, зовнішній ключ на таблицю гільдії, а також поля PasswordHash та PasswordSalt, які будуть зберігатися в базі даних. У випадку, якщо зловмисники отримають доступ до якогось з цих полів, справжні паролі користувачів залишуться у безпеці. Інші класи сутностей створюються аналогічно і в них прописуються необхідні поля для збереження даних.

Наступним етапом роботи стане створення контексту бази даних, який наслідує стандартний клас DbContext. Важливо в ньому додати всі створені моделі, які стануть таблицями майбутньої бази даних. Фрагмент коду представлено на рисунку 3.3

```
8 references
public DbSet<Joke> Jokes{ get; set; }
11 references
public DbSet<User> Users { get; set; }
7 references
public DbSet<Guild> Guilds { get; set; }
6 references
public DbSet<Comment> Comments { get; set; }
6 references
public DbSet<CommentReply> CommentsReplies { get; set; }
```

Рисунок 3.3 – Фрагмент коду класу DataContext

Джерело: [Розроблено автором]

Наступним кроком розробки є створення інтерфейсів та сервісів, що їх реалізують. Розглянемо розробку сервісів на прикладі UserService. Спочатку створюється інтерфейс IUserService, в якому прописуються всі методи, що будуть реалізовані(рисунок 3.4).

```
namespace JokeAPI.Services.UserService
{
    4 references
    public interface IUserService
    {
        2 references
        Task<User?> Login(UserDto request);
        2 references
        Task<bool> Register(UserDto request);
        2 references
        Task<List<User>?> GetUser(int id);
        2 references
        Task<bool> DeleteUser(int id);
        2 references
        Task<bool> UpdateUser(int id, UserDto request);
        2 references
        Task<List<User>> GetAllUsers();
        2 references
        Task<Guild> UpdateUserGuild(int userid, int guildid);
        2 references
        Task<List<Joke>?> GetAllJokes(int userid);


    }
}
```

Рисунок 3.4 – Інтерфейс IUserService

Джерело: [Розроблено автором]

Далі необхідно створити сервіс і реалізувати всі необхідні методи в ньому. Для прикладу розглянемо роботу методу DeleteUser. (рисунок 3.5)

```
private readonly DataContext _context;
0 references
public UserService(DataContext context)
{
    _context = context;
}
2 references
public async Task<bool> DeleteUser(int id)
{
    var response = await _context.Users.FindAsync(id);
    if(response == null) return false;
    _context.Users.Remove(response);
    await _context.SaveChangesAsync();
    return true;
}
```

Рисунок 3.5 – Фрагмент коду сервісу UserService

Джерело: [Розроблено автором]

Після ініціалізації контексту бази даних в конструкторі, можна використовувати його для надсилання запитів до бази даних, що і відбувається в першому рядку метода DeleteUser, де надсилається запит на асинхронний пошук користувача за його номером. Далі необхідний користувач видаляється з бази даних, всі зміни зберігаються і повертається результат. Схожим чином працюють всі методи всіх сервісів додатку.

Наступний етап розробки це створення контролерів. Для прикладу розглянемо роботу контролера UserController, зокрема його метода Register. (рисунок 3.6)

```
[Route("api/[controller]")]
[ApiController]
1 reference
public class UserController : ControllerBase
{
    private readonly IUserService _service;
    0 references
    public UserController(IUserService service)
    {
        _service = service;
    }

    [HttpPost("register")]
    0 references
    public async Task<ActionResult<User>> Register(UserDto request)
    {
        if ( await _service.Register(request)) return Ok("Success");
        return BadRequest("Wrong data");
    }
```

Рисунок 3.6 – Метод Register контролеру UserController

Джерело: [Розроблено автором]

Перш за все, ми прописуємо шлях до контролера в атрибутих. Далі вказується назва контролера, який обов'язково повинен наслідувати базовий. Потім ініціалізується сервіс, за допомогою якого відбуватимуться запити до бази даних. В самому методі ми прописуємо запит до сервісу, з метою зареєструвати користувача і, якщо сервіс повертає значення true – повертаємо код 200 клієнту. Якщо виникла помилка даних, то сервіс поверне false, а до користувача буде надіслано код 400. Схожим чином працюють всі методи всіх контролерів.

Останнім етапом розробки є підключення всіх сервісів і налаштування правил CORS. Для цього нам необхідно відповідним чином змінити файл Program.cs . (рисунок 3.7)

```
builder.Services.AddScoped<IJokeService, JokeService>();
builder.Services.AddScoped<IGuildService, GuildService>();
builder.Services.AddScoped<IUserService, UserService>();
builder.Services.AddScoped<ICommentService, CommentService>();
builder.Services.AddScoped<ICommentReplyService, CommentReplyService>();


builder.Services.AddDbContext<DataContext>();

builder.Services.AddCors(o => o.AddPolicy("AllowAll", builder =>
{
    builder.AllowAnyOrigin()
           .AllowAnyMethod()
           .AllowAnyHeader();
}));
```

Рисунок 3.7 – Фрагмент файлу Program.cs

Джерело: [Розроблено автором]

Було додано 5 сервісів, які були створені раніше, а також контекст бази даних. В правилах CORS вказано, що необхідно дозволити все. Тепер залишилось лише перевірити роботу серверу. Стандартним способом зробити це є використання Swagger. Для прикладу спробуємо створити користувача (рисунок 3.8), а потім авторизуватись у систему(рисунок 3.9), за допомогою Swagger.

Рисунок 3.8 – Реєстрація користувача Albert

Джерело: [Розроблено автором]

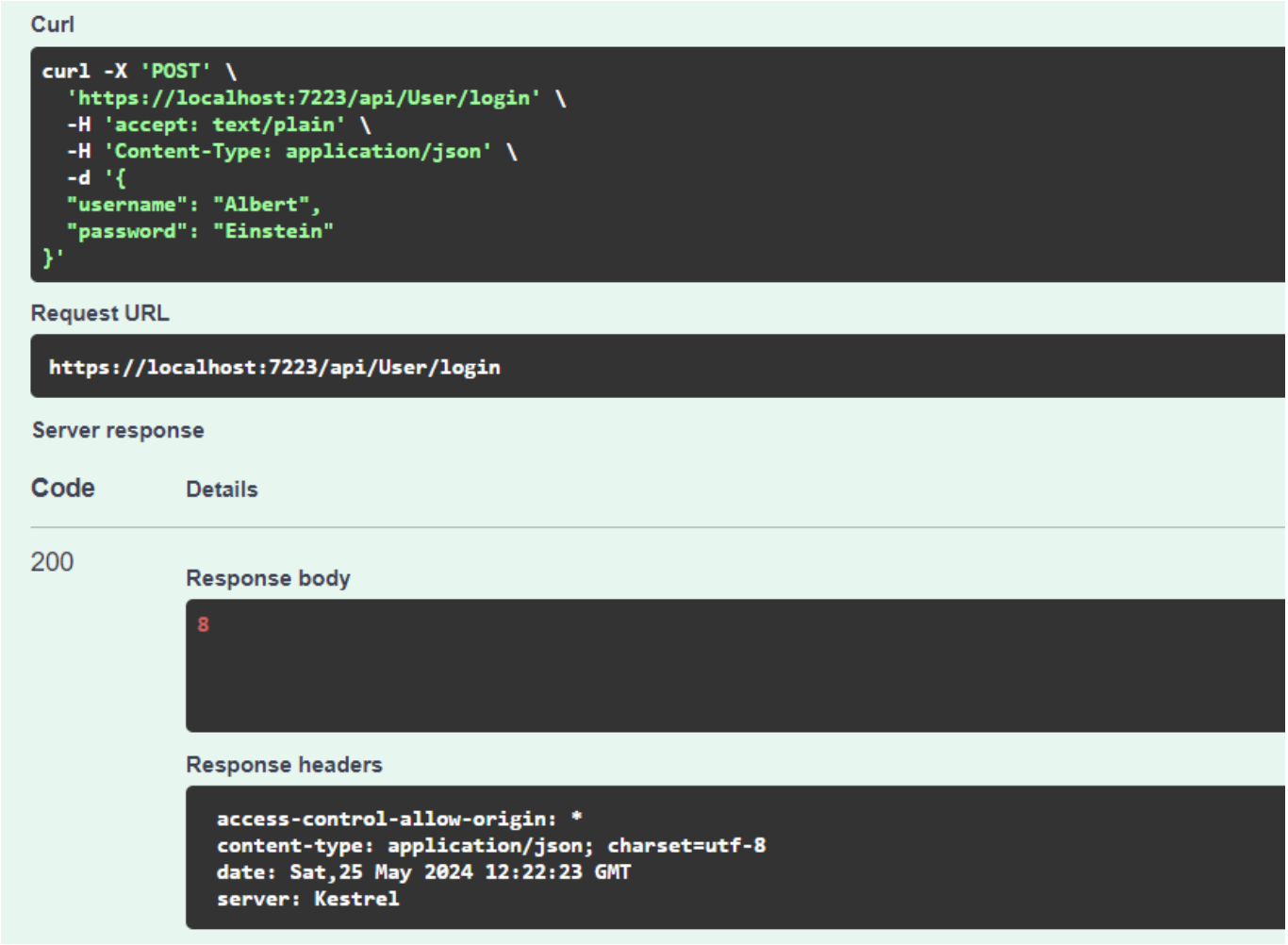


Рисунок 3.9 – Успішна авторизація користувача Albert

Джерело: [Розроблено автором]

В результаті ми успішно створили користувача з ім’ям Albert та паролем Einstein, а потім успішно авторизувались в систему і у відповідь отримали номер користувача. Схожим чином тестуються інші методи контролерів.

Важливо відмітити, що при роботі часто виникає необхідність звертатися до бази даних напряму, для перевірки правильності роботи різних компонентів системи. В нашому випадку для цього доцільно використати програму SQL Server Manager. Спробуємо перевірити наявність користувача, який був створений раніше, у базі даних. (рисунок 3.10)

	UserId	Username	PasswordH...	PasswordS...	GuildId
	8	Albert	<Binary dat...	<Binary dat...	1

Рисунок 3.10 – Ілюстрація доданого користувача Albert у базі даних

Джерело: [Розроблено автором]

3.1.2 Front-end розробка

Для розробки клієнтської частини додатку буде використано бібліотеку React. Структурно проект розбивається на папки для спрощення написання і читання коду. Для прикладу розглянемо папку Pages на рисунку 3.11

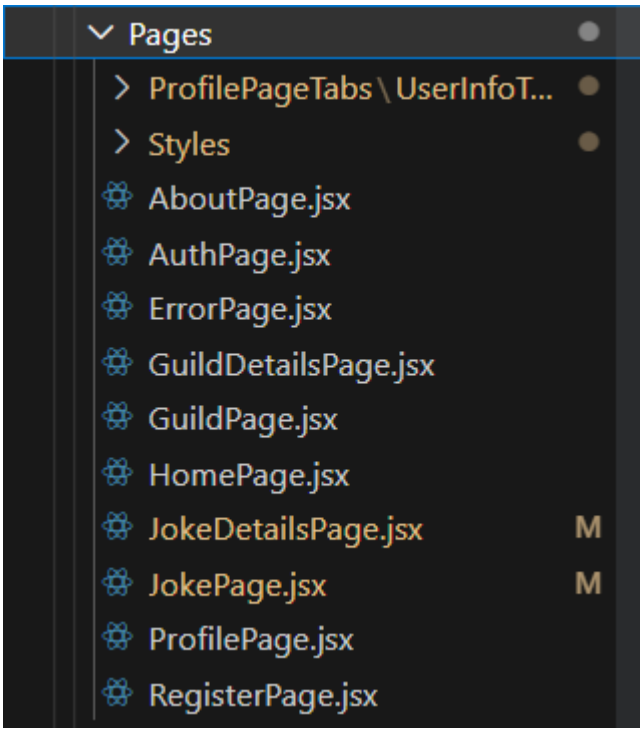


Рисунок 3.11 – Папка Pages

Джерело: [Розроблено автором]

Як бачимо з рисунку вище кожен файл з розширенням jsx відповідає одній сторінці. Також всередині є ще одна папка з назвою Styles(рисунок 3.12). В ній розміщені файли з розширенням module.css. Такі файли містять всередині всі стилі, що стосуються сторінки з відповідною назвою. Такий підхід дозволить не доспутити колізій.

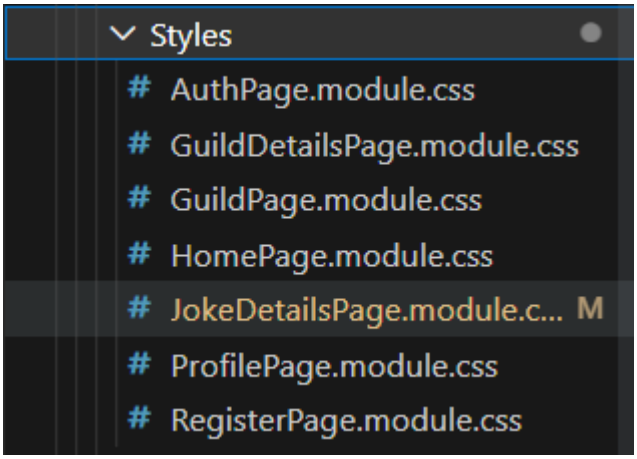


Рисунок 3.12 – Папка Styles

Джерело: [Розроблено автором]

Кожний файл сторінок повертає код, який далі відображається на сторінці. Можна побачити це на прикладі фрагменту коду з файлу JokePage.jsx(рисунок 3.13)

```
return (
  <div className="App">
    <ModalCreation isActive={isVisible} setIsActive={setVisible}>
      <PostForm Create={AddNewPost}></PostForm>
    </ModalCreation>

    <div className="main">
      <div className="navbar">
        <DefaultButton onClick={() => setVisible(true)}>
          Create joke
        </DefaultButton>
      </div>
      <div className="text">
        <SearchBar query={query} setQuery={setQuery} />

        <JokeList posts={sortedPosts} deletejoke={DeletePost} />
      </div>
    </div>
  </div>
);
```

Рисунок 3.13 – Фрагмент коду JokePage.jsx

Джерело: [Розроблено автором]

Як видно цей код схожий на звичайний html, але в ньому також присутні і не стандартні блочні елементи, наприклад ModalCreation. Це елемент, що відповідає за роботу модального вікна. Він універсальний для всього проекту і використовується одразу на багатьох сторінках. Розглянемо код цього компонента і на його прикладі визначимось з тим як працюють всі компоненти проекту (рисунок 3.14)

```
import React from "react";
import classes from "./ModalCreation.module.css";

const ModalCreation = ({ children, isActive, setIsActive }) => {
  const rootClasses = [classes.modalCreation];
  if (isActive) {
    rootClasses.push(classes.modalCreationActive);
  }

  return (
    <div className={rootClasses.join(" ")} onClick={() => setIsActive(false)}>
      <div
        className={classes.modalContent}
        onClick={(e) => e.stopPropagation()}
      >
        {children}
      </div>
    </div>
  );
};

export default ModalCreation;
```

Рисунок 3.14 – Компонент ModalCreation

Джерело: [Розроблено автором]

Спочатку ми імпортуємо бібліотеку і стилі. Далі ми прописуємо параметри для компоненти. Chilren – будь-який вміст, який повинен зберігатись всередині модального вікна, наприклад повідомлення або поле для вводу інформації. isActive та setIsActive відповідають за перевірку активності модального вікна та зміну цього значення відповідно. Наприклад, якщо на нашій сторінці є кнопка, при натисканні на яку повинно відкриватись модальне вікно, то ця кнопка повиння задавати isActive = true. Приклад роботи модального вікна на рис. 3.15.

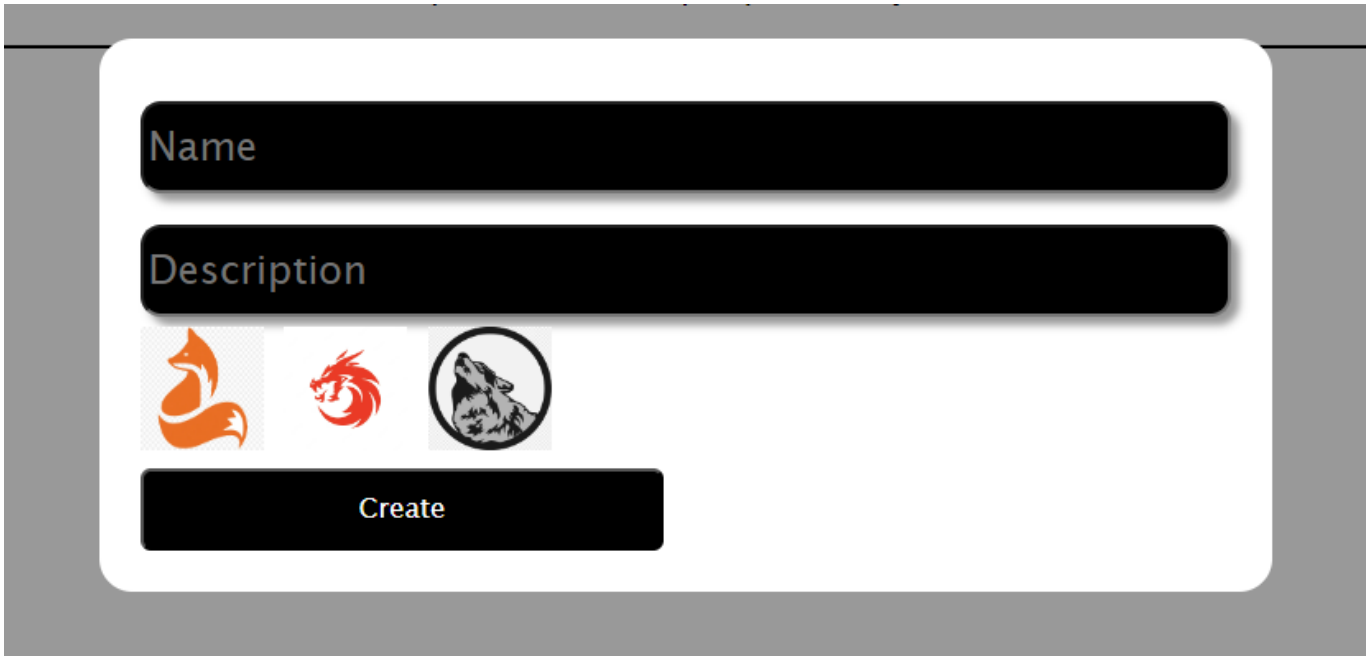


Рисунок 3.15 – Створення нової гільдії

Джерело: [Розроблено автором]

Для зв’язку клієнтської частини з сервером буде використовуватись бібліотека axios. Структурно всі файли які будуть стосуватись роботи з сервером будуть розміщені в окремій папці API. Розглянемо це на прикладі фрагменту коду з файлу UserService.jsx (рисунок 3.16)

```
10  static async Login(user) {
11    try {
12      const userId = await axios.post(
13        "https://localhost:7223/api/User/login",
14        user
15      );
16      return userId;
17    } catch {
18      return null;
19    }
20  }
```

Рисунок 3.16 – Фрагмент коду UserService.jsx

Джерело: [Розроблено автором]

За допомогою команди axios.post надсилається запит до сервера з метою перевірити чи є в базі даних необхідний нам користувач і повертається відповідний результат, який потім можна обробити в файлі сторінки або компонента.

Для навігації використовується бібліотека react-router-dom. Для прикладу порівняємо адреси які доступні авторизованим та новим користувачам (рис. 3.17)

```
12  export const publicroutes = [
13    { path: "/register", element: <RegisterPage />, exact: true },
14    { path: "/home", element: <HomePage />, exact: true },
15    { path: "*", element: <AuthPage />, exact: true },
16  ];
17
18  export const privateroutes = [
19    { path: "/about", element: <AboutPage />, exact: true },
20    { path: "/register", element: <RegisterPage />, exact: true },
21    { path: "/auth", element: <AuthPage />, exact: true },
22    { path: "/jokes", element: <JokePage />, exact: true },
23    { path: "/joke/:id", element: <JokeDetailsPage />, exact: true },
24    { path: "/guilds", element: <GuildPage />, exact: true },
25    { path: "/guild/:id", element: <GuildDetailsPage />, exact: true },
26    { path: "/home", element: <HomePage />, exact: true },
27    { path: "/profile", element: <ProfilePage />, exact: true },
28    { path: "*", element: <ErrorPage />, exact: true },
29
```

Рисунок 3.17 – Фрагмент коду routes.jsx

Джерело: [Розроблено автором]

Як бачимо, новим користувачам доступні лише сторінки реєстрації, авторизації та початкова сторінка, в той час як авторизованим доступні всі сторінки додатку.

3.2 Настанови з використання розробленого продукту

Перед розгортанням додатку спробуємо пройти всі етапи, через які будуть проходити майбутні користувачі. Спочатку створимо власну сторінку (рисунок 3.18).

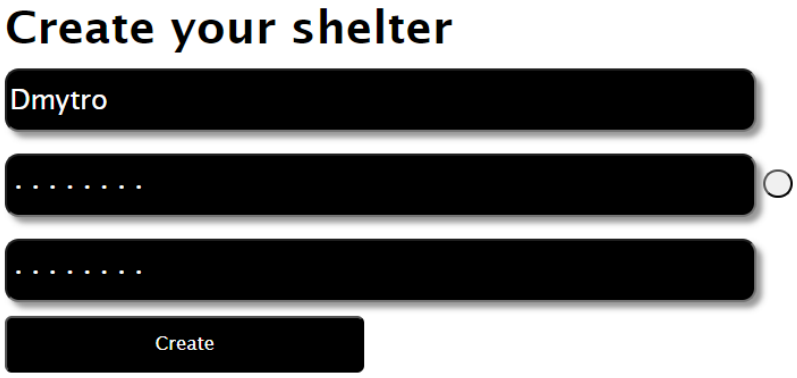


Рисунок 3.18 – Ілюстрація форми реєстрації користувача

Джерело: [Розроблено автором]

Після авторизації додаток надає функціонал створення жарту (рисунок 3.19).

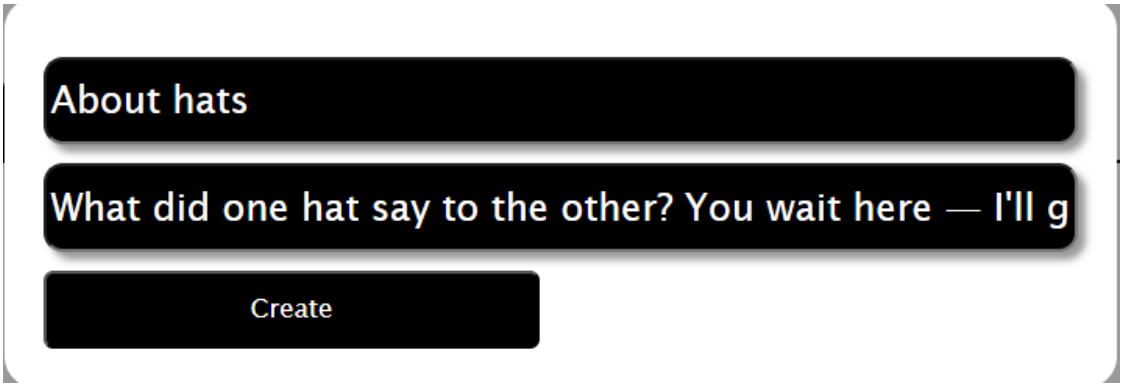


Рисунок 3.19 – Ілюстрація створення жарту

Джерело: [Розроблено автором]

Для додавання коментаря до жарту можна скористатись відповідною кнопкою та заповнити форму введення коментаря (рисунок 3.20).

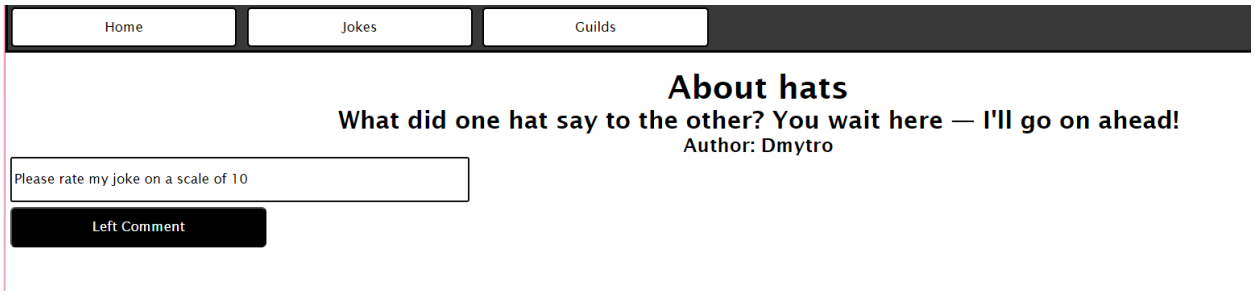


Рисунок 3.20 – Додавання коментаря

Джерело: [Розроблено автором]

Для відповіді на коментар треба заповнити форму коментування-відповіді, як зображено на рисунку 3.21.

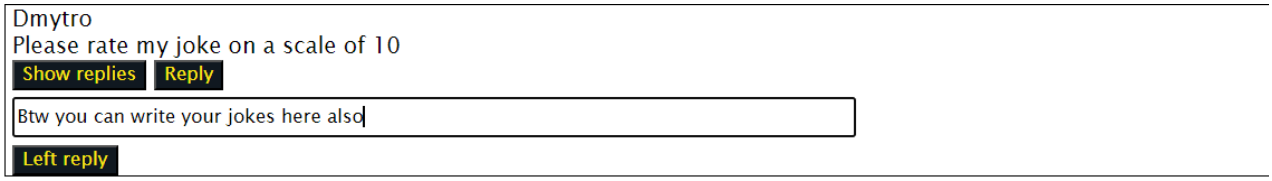


Рисунок 3.21 – Відповідь на коментар

Джерело: [Розроблено автором]

Створення гільдії відбувається за допомогою відповідної форми, ілюстрація заповнення якої приведення на рис. 3.22.

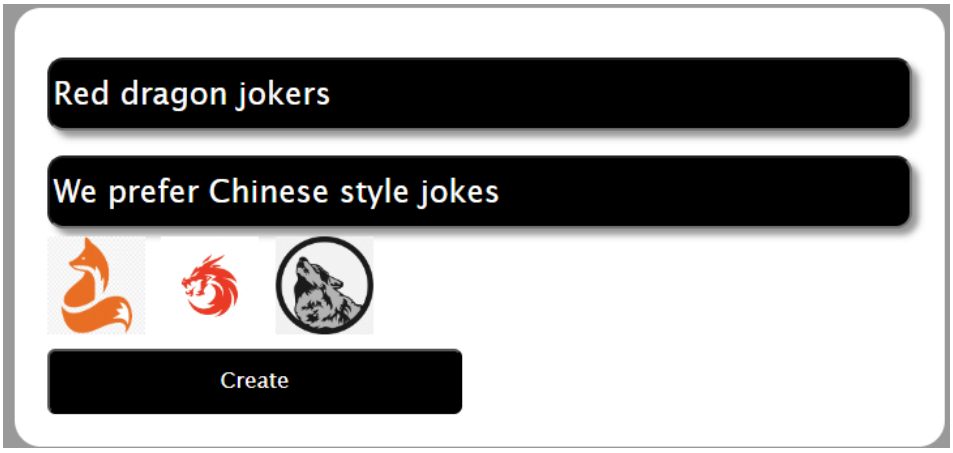


Рисунок 3.22 – Створення гільдії

Джерело: [Розроблено автором]

Додаток дозволяє виконати пошук гільдії з використанням поля загального пошуку (рисунок 3.23).

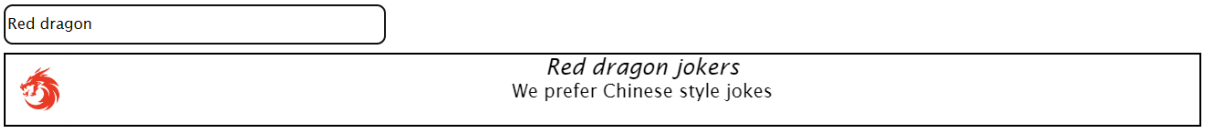


Рисунок 3.23 – Пошук гільдії

Джерело: [Розроблено автором]

Основний функціонал додатку перевірено, помилок не знайдено.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи бакалавра розроблено вебдодаток соціальної мережі поширення гумористичного контенту.

В ході виконання даної роботи було проведено всебічний аналіз предметної області, що дозволило визначити ключові вимоги та особливості вебдодатку для обміну жартами. Аналіз останніх досліджень і публікацій дозволив виявити актуальні тенденції та методи, що застосовуються у розробці подібних продуктів. Огляд програмних продуктів аналогів дав можливість оцінити існуючі рішення на ринку та виокремити їх сильні та слабкі сторони, що стало основою для формулювання мети та задач дослідження.

На етапі моделювання додатку було розроблено концептуальні моделі, які описують основні функціональні та нефункціональні вимоги до системи. Архітектура додатку включає в себе детальний опис клієнтської частини, реалізованої на основі React, та серверної частини на ASP.NET API. Проектування бази даних забезпечило створення ефективної та надійної структури для зберігання та обробки даних.

Практична реалізація проекту включає програмну розробку всіх компонентів системи. Клієнтська частина була створена з використанням React, що забезпечило високу продуктивність та інтерактивність користувацького інтерфейсу. Серверна частина на основі ASP.NET API реалізувала всю необхідну логіку для обробки запитів та взаємодії з базою даних.

У заключній частині роботи було розглянуто тестове використання розробленого продукту, що дозволило продемонструвати його функціональні можливості та практичну цінність. Вебдодаток відповідає сучасним вимогам до продуктивності, безпеки та зручності використання, що підтверджує досягнення поставлених цілей та виконання задач дослідження.

Загалом, проведена робота дозволила створити повнофункціональний вебдодаток для обміну жартами, який має потенціал для подальшого розвитку та вдосконалення. Результати дослідження можуть бути корисними для інших розробників та дослідників у цій галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bishqemi, Kelsey, & Crowley, Michael. (2022). TikTok Vs. Instagram: Algorithm Comparison. *Journal of Student Research, 11,* 10.47611/jsrhs.v11i1.2428.

2. Halgin, Daniel, Brass, Daniel, & Borgatti, Stephen. (2014). Social Network Research: Confusions, Criticisms, and Controversies. *Advances in Group Processes, 31*, 1-32. 10.1108/S0733-558X(2014)0000040001.

3. Gruzd, Anatoliy, Mai, Philip, & Vahedi, Zahra. (2020). Studying Anti-Social Behaviour on Reddit with Communalytic. *Advances in Group Processes, 37*, 1-24. 10.31124/advance.12453749.v1.

4. Cobbe, Jennifer. (2021). Algorithmic Censorship by Social Platforms: Power and Resistance. *Philosophy & Technology, 34*, 10.1007/s13347-020-00429-0.

5. Cai, Jie, Patel, Aashka, Naderi, Azadeh, & Wohn, Donghee. (2024). Content Moderation Justice and Fairness on Social Media: Comparisons Across Different Contexts and Platforms. *Proceedings of the ACM on Human-Computer Interaction, 6*(CSCW1), Article 39. 10.1145/3613905.3650882.

6. Tang, Kunzhi, Zeng, Chengang, Fu, Yuxi, & Zhu, Gang. (2022). Security Analysis of Social Network Topic Mining Using Big Data and Optimized Deep Convolutional Neural Network. *Computational Intelligence and Neuroscience, 2022*, 1-12. 10.1155/2022/8045968.

7. Pinchot, Jamie, & Paullet, Karen. (2012). What's in your profile? Mapping Facebook profile data to personal security questions. *International Journal of Virtual Communities and Social Networking, 4*(4), 284-293.

8. Arad, Ayala, Barlizaly, Ohad, & Perchick, Maayan. (2023). Facebook, social comparison and happiness: Evidence from a quasi-natural experiment. *Cyberpsychology: Journal of Psychosocial Research on Cyberspace, 17*(4), 10.5817/CP2023-4-4.

9. Arslan, Kader, & Trier, Matthias. (2023). Similar Affordances, Different Use Practices? An Investigation of the Socio-Cultural Contexts in Facebook, Instagram, and Twitter.

10. Piechota, Grażyna. (2023). Cultural Differences in Network Communication. How Polish, German, and Ukrainian Netizens Use Social Media.

11. Instagram. [Електронний ресурс]. Режим доступу: https://www.instagram.com/ (17.05.2024 р)

12. Facebook. [Електронний ресурс]. Режим доступу: https://www.facebook.com/ (17.05.2024 р)

13. Reddit. [Електронний ресурс]. Режим доступу: https://www.reddit.com/ (17.05.2024 р)

14. X. [Електронний ресурс]. Режим доступу: https://www.x.com/ (17.05.2024 р)

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи
«Соціальна мережа Joke»

ПОГОДЖЕНО:

_____ к. т. н., доц.

Неня Анна

Студент групи ІТ-01

_____ Алексєєв Д.С.

Суми 2024

1. Призначення й мета створення вебдодатку

1.1 Призначення вебдодатку

Вебдодаток соціальної мережі має надавати користувачам можливість додавати власні жарти, оцінювати жарти інших користувачів та об’єднуватись в гільдії за інтересами.

1.2 **Мета створення** вебдодатку

Основна мета проекту полягає у створенні вебдодатку підтримки роботи інноваційної соціальної мережі "Joke", яка надасть користувачам унікальний та позитивний досвід спілкування через гумористичний контент

1.3 Цільова аудиторія

До цільової аудиторії вебдодатку можна віднести людей всіх вікових категорій, які зацікавленні в спілкуванні в інтернеті.

2 Вимоги до вебдодатку

2.1 Вимоги до вебдодатку в цілому

2.1.1 Вимоги до структури й функціонування вебдодатку

Вебдодаток має бути доступним в мережі Інтернет. Вебдодаток повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

2.1.2 Вимоги до персоналу

Для підтримки роботи додатку та впровадження нових функцій від персоналу вимагаються знання з ASP.NET, MSSQL, React та базові знання з HTML, CSS.

2.1.3 Вимоги до збереження інформації

Уся інформація надана у вебдодатку буде зберігатися у базі даних реалізованій засобами системи управління базами даних MSSQL. Персональні дані користувачів повинні зберігатися у зашифрованому вигляді

2.1.4 Вимоги до розмежування доступу

Розроблюваний вебдодаток має бути загальнодоступним.

Відповідно до прав доступу до інформації у вебдодатку, усіх користувачів можна поділити на гостей та авторизованих користувачів.

Гості можуть переглядати домашню сторінку та сторінки реєстрації і авторизації.

Авторизовані користувачі можуть переглядати всі сторінки вебдодатку.

2.2 Структура вебдодатку

2.2.1 Загальна інформація про структуру вебдодатку

Структура вебдодатку являє собою набір сторінок, деякі з яких є пунктами головного меню.

Такими розділами є:

Домашня сторінка – на сторінці зображені повідомлення привітання та найкращі жарти за певний проміжок часу, щоб зацікавити користувачів.

Жарти – перелік всіх жартів які створені користувачами сайту.

Детально про жарт – детальна інформація про жарт, автора та коментарі користувачів

Гільдії – інформація про всі створені гільдії.

Детально про гільдію – інформація про учасників гільдії.

Профіль – особиста інформація користувача.

Авторизація – сторінка для входу користувача в систему.

Реєстрація – сторінка для реєстрація користувача в системі.

Помилка – сторінка, на яку користувач потрапляє у виняткових випадках не передбачених розробником.

2.2.2 Навігація

Для навігації у шапці буде створена система контент меню. Меню необхідне для швидкого переміщення користувача по усім доступним сторінкам. Меню буде відображатися на всіх сторінках, щоб відвідувач міг в будь-який момент часу перейти на будь-яку сторінку вебдодатку.

2.2.3 Наповнення вебдодатку (контент)

Наповнення додатку буде здійснюватись користувачами додатку. Зображення для гільдій та інформація для домашньої сторінки будуть додаватись розробниками.

2.2.4 Дизайн та структура додатку

Стиль вебдодатку має бути сучасним, приємним для сприйняття, у якості основних кольорів було запропоновано використати чорний та білі відтінки.

Вебдодаток має бути інтуїтивно зрозумілим для використання.

Розташування елементів на головній сторінці вебдодатку схематично показано на рисунку А.1.

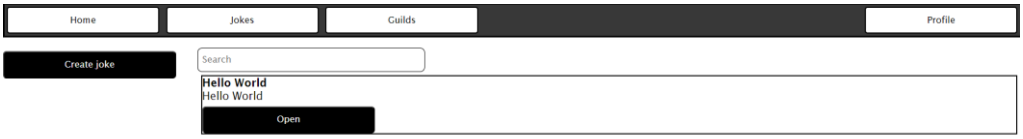


Рисунок А.1 – Схема головної сторінки

Джерело: [Розроблено автором]

2.2.5 Система навігації (карта вебдодатку)

Карта вебдодатку зображена на рисунку А.2.

Рисунок А.2 – Карта вебдодатку

Джерело: [Розроблено автором]

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Створення жартів	Клієнт
UN-02	Коментування жартів	Клієнт
UN-03	Написання відповідей до коментарів інших користувачів	Клієнт
UN-04	Створення гільдій	Клієнт
UN-05	Вступ до гільдій разом з іншими користувачами	Клієнт
UN-06	Реєстрація та авторизація	Клієнт
UN-07	Редагування персональних даних	Клієнт
UN-08	Заповнення домашньої сторінки	Розробник

Джерело: [Розроблено автором]

2.3.2 Функціональні вимоги

1. Створення, редагування та видалення облікових записів користувачів.
2. Можливість створення, публікації та взаємодії з гумористичним контентом.
3. Введення системи гільдій для об’єднання користувачів у групи.
4. Механізми взаємодії з аудиторією, такі як коментарі.

2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на вимоги до системи. Їх перелік наведений в таблиці А.2.

Таблиця А.2 – Вимоги до системи

ID	Вимоги до системи	Пріоритет	Опис
SR-01	Реєстрація та авторизація	M	Надає можливість створювати свою особисту сторінку для доступу в систему
SR-02	Шифрування особистих даних	M	Паролі користувачів зберігаються в базі даних в зашифрованому вигляді
SR-03	Редагування особистих даних	S	Надає можливість змінювати особисті дані після реєстрації
SR-04	Каталог жартів	M	Надає можливість переглядати жарти, які користувачі додали в систему
SR-05	Каталог гільдій	M	Надає можливість переглядати гільдії, які користувачі створили
SR-06	Домашня сторінка	M	Надає можливість зацікавлення нових користувачів
SR-07	Спеціальні можливості кастомізації особистої сторінки	C	Анімовані портрети, власні підписи, обрані жарти, кількість лайків і місце користувача в загальному рейтингу

Джерело: [Розроблено автором]

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

Could have (C) – вимоги, які можуть бути реалізовані, але вони не є центральною ціллю проекту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація вебдодатку відбувається з використанням:

— React

— ASP.NET

— MSSQL

2.4.2 Вимоги до лінгвістичного забезпечення

Вебдодаток має бути виконаний англійською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

— Веб-браузер: Internet Explorer 7.0 і вище, або Firefox 3.5 і вище, або Opera 9.5 і вище, або Safari 3.2.1 і вище, або Chrome 2 і вище.

— Стабільне інтернет з’єднання

— Windows 7 і вище

3 Склад і зміст робіт зі створення вебдодатку

Докладний опис етапів роботи зі створення вебдодатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення вебдодатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	**Попередні дослідження та аналіз:** Зміст робіт: Проведення маркетингового аналізу соціальних мереж. Вивчення конкурентів та аналіз їхніх слабких сторін.	20 днів
2	**Розробка технічного завдання (ТЗ):** Зміст робіт: Формулювання вимог до функціоналу та технічних характеристик. Узгодження замовником.	14 днів
3	**Проектування архітектури та інтерфейсу:** Зміст робіт: Створення архітектурної концепції. Розробка дизайну та інтерфейсу.	20 днів
4	**Розробка програмного забезпечення:** Зміст робіт: Кодування функціональності та компонентів. Тестування проміжних версій.	40 днів
5	**Тестування та оптимізація:** Зміст робіт: Виконання тестів на відповідність вимогам. Виявлення та усунення помилок.	30 днів
6	**Впровадження та підтримка:** Зміст робіт: Запуск соціальної мережі для громадськості. Підтримка та вирішення проблем після випуску.	нескінченно

Джерело: [Розроблено автором]

4 Вимоги до складу й змісту робіт із введення вебдодатку в експлуатацію

Для того, щоб вебдодатком могли користуватися клієнти та відвідувачі необхідно розмістити його у мережі Інтернет, тому необхідно придбати доменне ім’я та місце на хостингу. На хостинг переноситься вебдодаток і наповнення бази даних з подальшою їх доробкою. Для коректного переносу вебдодатку на хостинг необхідно, щоб параметри хостинга відповідали вимогам, зазначеним у ТЗ.

ДОДАТОК Б

Деталізація мети проєкту методом SMART. Для успішності та конкурентоспроможності проєкту треба на концептуальному етапі правильно визначити його мету за допомогою SMART-методу. Вона має ширше формулювання, а саме «Створення платформи для гумористичного контенту з акцентом на модерацію, приватність, персоналізацію та залучення аудиторії для збору користувацької бази і подальшої монетизації через впровадження рекламних оголошень до 1 червня 2024 року». Результати деталізації методом SMART розміщені у таблиці Б.1.

Таблиця Б.1. – Деталізація мети проєкту методом SMART

Specific	Створення платформи для гумористичного контенту з акцентом на модерацію, приватність, персоналізацію та залучення аудиторії.
Measurable	Готовий проєкт, що відповідає всім вимогам.
Achievable	Мета досяжна, є затверджене технічне завдання, розроблений план роботи, а також є запит від аудиторії на новаційну соц-мережу
Relevant	Для збору користувацької бази і подальшої монетизації через впровадження рекламних оголошень
Time-framed	Проєкт має бути завершено до 1 червня 2024 року

Джерело: [Розроблено автором]

Планування змісту робіт. WBS (Work Breakdown Structure – ієрархічна структура робіт) – це графічний вигляд елементів проєкту, які згруповані ієрархією

у єдине ціле з продуктом проєкту. Структура декомпозиції робіт орієнтована на досконале виконання робіт по частинам і сама є ключовою частиною проєкту, яка спрямована на організацію командної роботи. Елементами декомпозиції можуть бути продукти, дані та послуги. Більше того, WBS забезпечує необхідним каркасом для ретельної оцінки термінів та контролю та графіків роботи.

На найвищому (першому) рівні розміщений продукт проєкту. Основні дії та заходи, що забезпечують досягнення мети проєкту, зафіксовані на другому рівні декомпозиції. Декомпозиція робіт виконується до тих пір, поки вони не стануть елементарними (простими).

Елементарні роботи – це дії, які мають однозначний чіткий результат, на які призначена відповідальному одна конкретна особа, для якої можна обчислити витрати праці і тривалість виконання. На рисунку Б.1 представлено WBS з розробки вебдодатку.

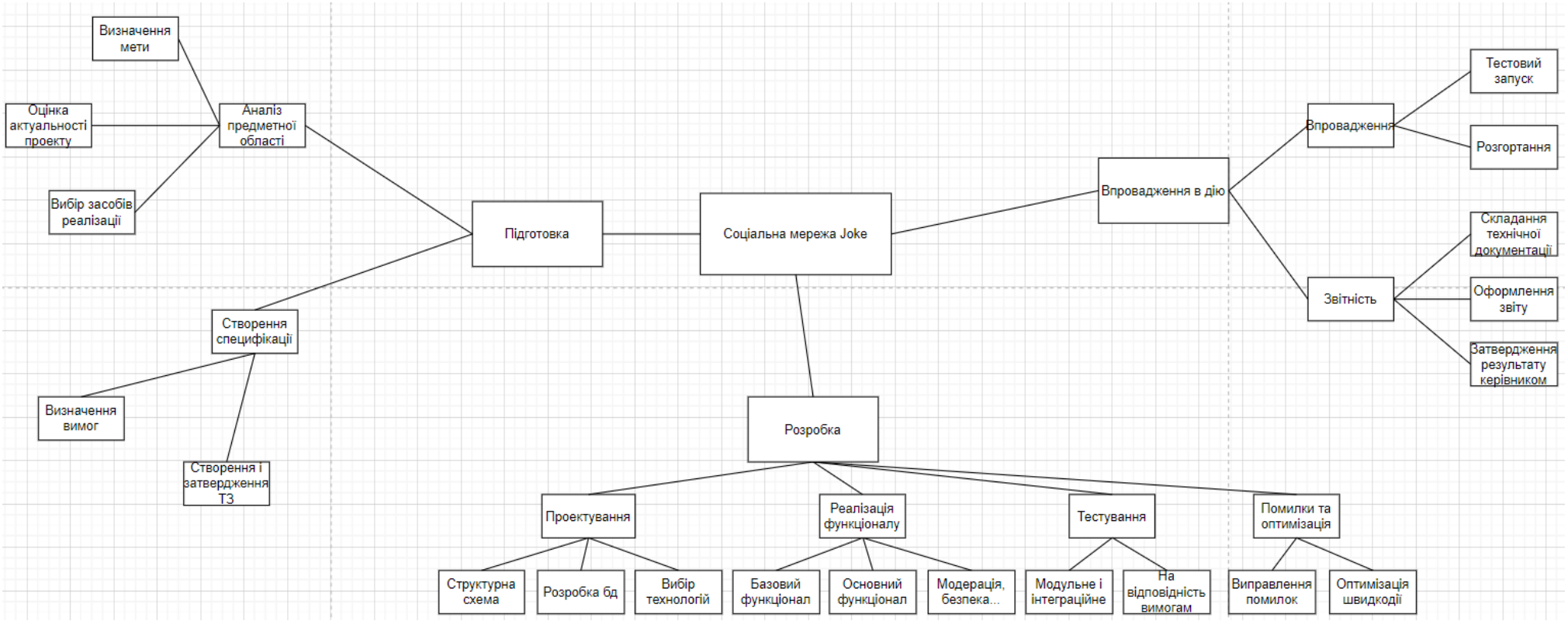


Рисунок Б.1 – WBS-структура робіт проєкту

Джерело: [Розроблено автором]

Планування структури виконавців. Наступним етапом після декомпозиції процесів є розробка організаційної структури виконавців або OBS, яка визначається як графічна структура відображення учасників або відповідальних осіб, які беруть участь у реалізації проєкту.

У ролі відповідальних осіб виступають співробітники, що відповідають за організацію і виконання елементарної роботи, що зазначена у WBS. Кожну елементарну роботи можна розглядати як окремий проєкт. На рисунку Б.2 представлено організаційну структуру планування проєкту.

Список виконавців, що функціонують в проєкті описано в таблиці Б.2.

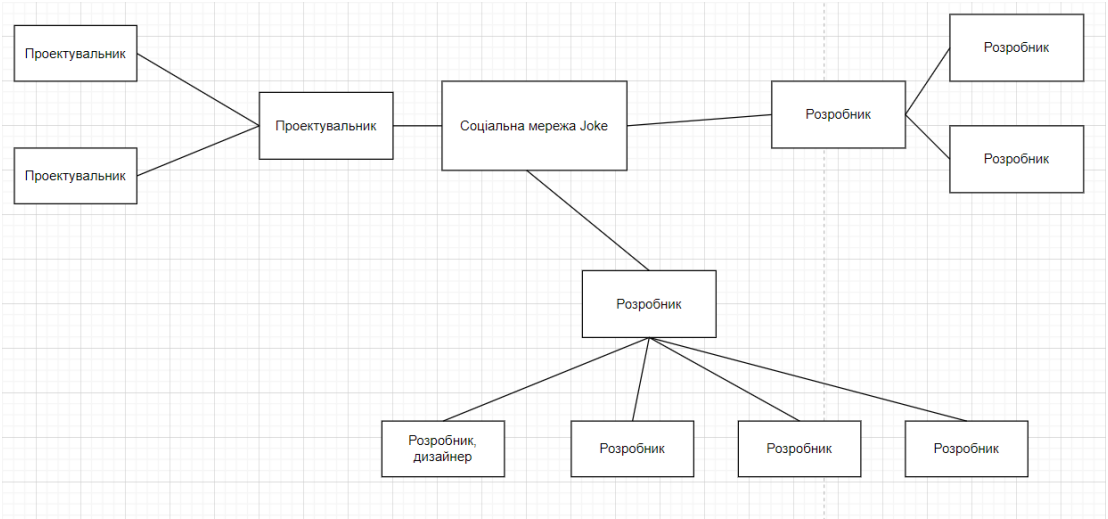


Рисунок Б.2 – OBS-структура робіт проєкту

Джерело: [Розроблено автором]

Таблиця Б2 – Список виконавців

Роль	ПІБ	Проєктна роль
Керівник	Неня А.В.	Формує завдання, вимоги, перевіряє результат
Розробник	Алексєєв Д.С.	Розробляє весь функціонал додатку
Дизайнер	Алексєєв Д.С.	Створює дизайн сайту, проектує бази даних
Тестувальник	Алексєєв Д.С.	Створює тести та перевіряє стабільність роботи додатку

Джерело: [Розроблено автором]

Діаграма Ганта. Побудова календарного графіку (діаграми Ганта) є одним з важливих етапів планування проєкту, що виглядає як розклад виконання робіт з реальним розподілом дат. Завдяки йому можна отримати достовірне уявлення про тривалість процесів з обмеженнями у ресурсах, урахуванням вихідних днів та свят. Календарний графік проєкту представлено на рисунку Б.3, продовження графіку представлено на рисунку Б.4.

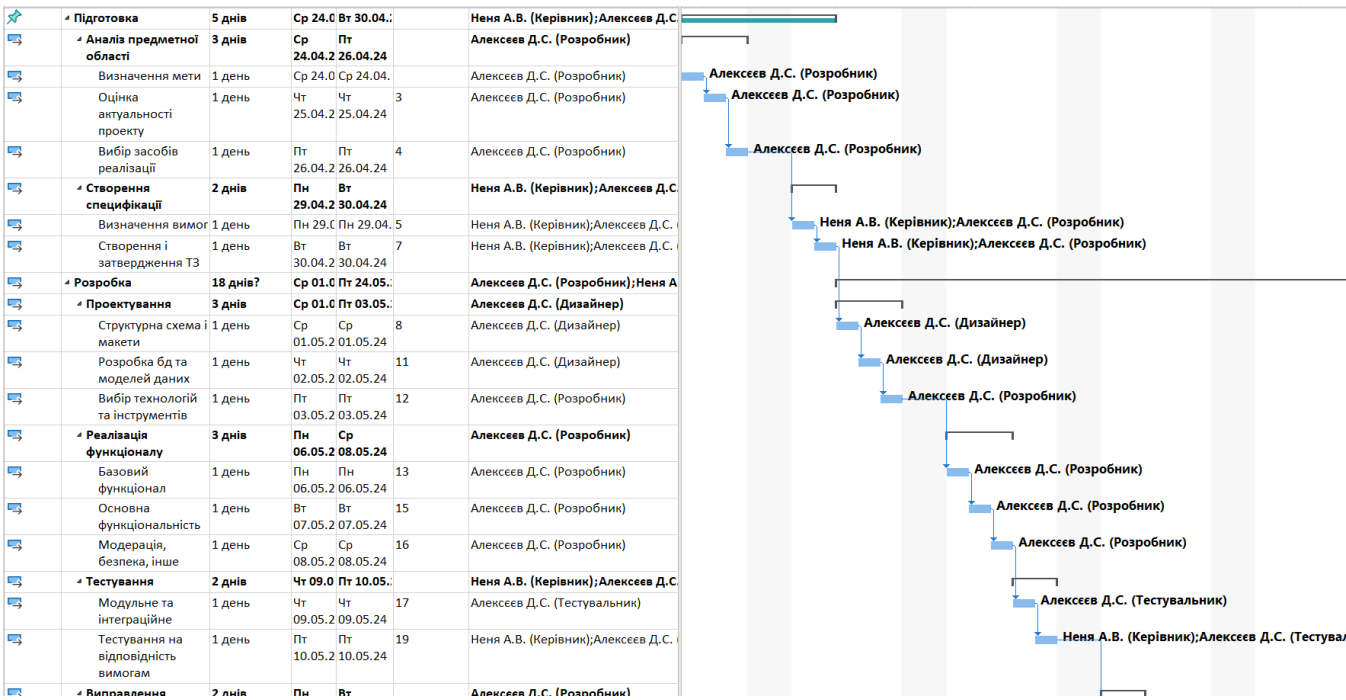
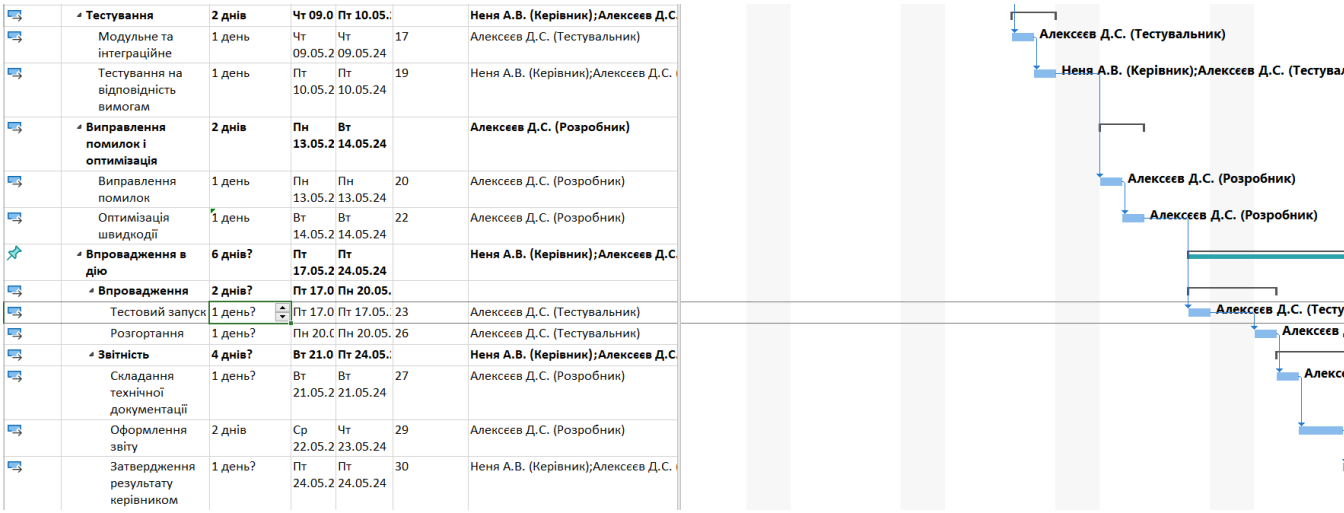


Рисунок Б.3 – Календарний графік проекту

Джерело: [Розроблено автором]



	Тестування	**2 днів**	**Чт 09.0**	**Пт 10.05.**		**Неня А.В. (Керівник);Алексєєв Д.С**
	Модульне та інтеграційне	1 день	Чт 09.05.2	Чт 09.05.24	17	Алексєєв Д.С. (Тестувальник)
	Тестування на відповідність вимогам	1 день	Пт 10.05.2	Пт 10.05.24	19	Неня А.В. (Керівник);Алексєєв Д.С.
	Виправлення помилок і оптимізація	**2 днів**	**Пн 13.05.2**	**Вт 14.05.24**		**Алексєєв Д.С. (Розробник)**
	Виправлення помилок	1 день	Пн 13.05.2	Пн 13.05.24	20	Алексєєв Д.С. (Розробник)
	Оптимізація швидкодії	1 день	Вт 14.05.2	Вт 14.05.24	22	Алексєєв Д.С. (Розробник)
	Впровадження в дію	**6 днів?**	**Пт 17.05.2**	**Пт 24.05.24**		**Неня А.В. (Керівник);Алексєєв Д.С**
	Впровадження	**2 днів?**	**Пт 17.0**	**Пн 20.05.**		
	Тестовий запуск	1 день?	Пт 17.0	Пт 17.05.	23	Алексєєв Д.С. (Тестувальник)
	Розгортання	1 день?	Пн 20.0	Пн 20.05.	26	Алексєєв Д.С. (Тестувальник)
	Звітність	**4 днів?**	**Вт 21.0**	**Пт 24.05.**		**Неня А.В. (Керівник);Алексєєв Д.С**
	Складання технічної документації	1 день?	Вт 21.05.2	Вт 21.05.24	27	Алексєєв Д.С. (Розробник)
	Оформлення звіту	2 днів	Ср 22.05.2	Чт 23.05.24	29	Алексєєв Д.С. (Розробник)
	Затвердження результату керівником	1 день?	Пт 24.05.2	Пт 24.05.24	30	Неня А.В. (Керівник);Алексєєв Д.С.

Рисунок Б.4 – Продовження календарного графіку проекту

Джерело: [Розроблено автором]