ДИПЛОМНА РОБОТА
БАКАЛАВРА

ДРБ. ІП - 15.00.00.000 ПЗ

Група ІП-20-1

Михайлович Василь

2024

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: “ Реалізація програмного забезпечення системи зберігання та поширення мультимедійного контенту ”

(назва згідно з наказом ректора)

спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІП - 15.00.00.000 ПЗ

(позначення)

Консультанти:

з питань нормоконтролю

асист. /М.М. Піх/
(посада) (підпис) (дата)(розшифровка підпису)

Студент

ІП-20-1 /В.В.Михайлович/
(шифр групи) (підпис) (дата) (розшифровка підпису)

Керівник роботи

к.т.н., доц. /Р.Б.Вовк/
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту
Завідувач кафедри

к.т.н., доц. /В.В.Бандура/
(посада) (підпис) (дата) (розшифровка підпису)

2024
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2024 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Михайловичу Василю Вікторовичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Реалізація програмного забезпечення системи зберігання та поширення мультимедійного контенту”

керівник проекту (роботи) Вовк Роман Богданович, доцент

затверджені наказом закладу вищої освіти від **“** ____ ” травня 2024 р. № _____

2. Строк подання студентом проекту (роботи) 17 червня 2024 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз сучасних інформаційних рішень побудови систем поширення контенту
2. Проектування структури системи та алгоритмічного забезпечення
3. Розробка математичного забезпечення системи поширення мультимедійного контенту
4. Представлення інтерфейсу проектованої системи
5. Виконання процесу тестування системи зберігання та поширення мультимедійного контенту

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Вкладка для завантаження фотографій на сервісі SugarSync (рис. 1.1)
2. Вигляд MediaFire додатку для Android (рис. 1.2)
3. Вигляд сервісу ZippyShare (рис. 1.3)
4. Головна сторінка ресурсу зберігання контенту Mega (рис. 1.4)
5. Головна сторінка сайту Imgur (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **22 травня 2024 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз сучасних інформаційних рішень побудови систем поширення контенту	26.05.2024	виконано
2	Проектування структури системи та алгоритмічного забезпечення	29.05.2024	виконано
3	Розробка математичного забезпечення системи поширення мультимедійного контенту	31.05.2024	виконано
4	Представлення інтерфейсу проектованої системи	05.06.2024	виконано
5	Виконання процесу тестування системи зберігання та поширення мультимедійного контенту	11.06.2024	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	17.06.2024	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Темою дипломної роботи є реалізація програмного забезпечення системи обробки мультимедійного контенту. В результаті розроблено веб-сервіс який можуть використовувати користувачів, які цікавляться музикою, а також для перегляду відеозаписів та фотографій, і які бажають зберігати свої медіафайли в одному сервісі, ділитися з ними з іншими користувачами, підписуватись на оновлення інших користувачів сайу та завантажувати їхній мультимедійний контент.

КЛЮЧОВІ СЛОВА: ПОШИРЕННЯ МУЛЬТИМЕДІА, ЗБЕРІГАННЯ ФАЙЛІВ, ER-ДІАГРАМА, БАЗА ДАНИХ, РУЧНЕ ТЕСТУВАННЯ, ВЕБ-СЕРВІС, МУЛЬТИМЕДІЙНИЙ КОНТЕНТ

ANNOTATION

The topic of the thesis is the implementation of the multimedia content processing system software. As a result, a web service has been developed that can be used by users who are interested in music, as well as watching videos and photos, and who want to store their media files in one service, share them with other users, subscribe to the updates of other sayu users and download their multimedia. content

KEYWORDS: MULTIMEDIA DISTRIBUTION, FILE STORAGE, ER DIAGRAM, DATABASE, MANUAL TESTING, WEB SERVICE, MULTIMEDIA CONTENT

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 8
ВСТУП 9

1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПОБУДОВИ СИСТЕМ ЗБЕРЕЖЕННЯ ТА ОБМІНУ ФАЙЛАМИ МУЛЬТИМЕДІА ... 12
1.1. Актуальність програмного продукту 12
1.2. Огляд аналогічних додатків зберігання контенту 14
1.3. Формування вимог до програмного сервісу поширення мультимедійного контенту 20
1.4. Аналіз та вибір засобів для реалізації проекту 22
1.5. Висновки по розділу 31

2. РОЗРОБКА АЛГОРИТМІЧНОГО ТА МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ПОШИРЕННЯ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ 32
2.1. Реалізація алгоритмічних схем функціонування системи 32
2.2. Представлення математичного забезпечення системи 37
2.3. Проектування алгоритмів хешування, пошуку та сортування даних 40
2.4. Висновки по розділу 47

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗБЕРІГАННЯ ТА ПОШИРЕННЯ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ 48
3.1. Побудова бази даних 48
3.2. Проектування класів та архітектури програмної системи 53

3.3. Розробка функціональності та інтерфейсу системи 57
3.4. Виконання тестування програмного продукту 69
3.5. Висновки до розділу 71

ВИСНОВКИ 72
СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА 74
ДОДАТКИ
БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

URL – Uniform Resource Locator

CGI – Common Gateway Interface

IT – Information Technology

VDI – Virtual Desktop Infrastructure

QR – Quick Response

GIF – Graphics Interchange Format

DBA – Database Administrator

DDL – Dynamic-Link Library

DML – Data Manipulation Language

API – Application Programming Interface

JMX – Java Management Extensions

VM – Virtual Machine

LVRA – Las Vegas Randomized Algorithm

FTL – FreeMarker Template Language

ВСТУП

На сьогоднішній день ми живемо в часи інформативності, а перегляд інформаційних сервісів є щоденним явищем для більшості людей. Якщо декілька десятиліть тому була проблема з пошуком інформації, то в даний момент існує безліч сервісів для її надання, і виникають труднощі з їх пересиченням. Серед великої конкуренції в даній сфері важливою є цікава і зручна подача інформації, яка в результаті буде цікавою для користувачів і що в результаті призведе до постійного використання сервісу. Говорячи про один з способів відтворення інформації важливо відзначити про застосування медіафайлів.

Мультимедійний контент представляє собою різноманіття фотографій, музики, відеозаписів та інших документів з різними форматами. Жоден інформаційний сервіс не обходиться без використання медіафайлів, оскільки вони в тій чи іншій мірі використовуються усіх сферах, які пов’язані з ІТ. Кожна соціальна мережа представляє собою можливість завантаження та роботу з фотографіями (використовуються і інші медіафайли, однак фотографії переважають). Крім того, існують безліч окремих сервісів для окремого перегляду відеофайлів та прослухування аудіозаписів.

Є декілька основних причин поширеності використання медіафайлів, а саме: можливість подачі інформації в зручному і якісному форматі, а також можливість завантаження і обміну файлів. Саме завантаження і збереження медіафалів стало основною метою реалізації дипломного проекту.

Дана тематика була обрана з наступних причин:

– актуальність з точки зору користувача. Для користувачів важлива чіткість і лаконічність в подачі інформації і приємний вигляд для людського ока. Варто відзначити що існує багато відомих сервісів для зручного представлення та зберігання медіафайлів, однак виникає проблема з комфортним спілкуванням між окремими користувачами сервісів. Саме даний

аспект являється одним з ключових факторів для реалізації подібного сервісу;

– актуальність з точки зору розробника. Робота з надсиланням та збереженням медіафайлів є поширеною, це важливі навички для освоєння і для можливості отримання роботи в IT-сфері.

Варто відзначити і про саме поняття веб-сервісу. Під веб-сервісом розуміється прикладна програма, розроблена з архітектури «клієнт-сервер», що використовує в якості клієнта веб-браузер і працює на стороні веб-сервера. При цьому взаємодія між клієнтом і сервером виконується з використанням протоколу HTTP.

Користувач за допомогою браузера відправляє HTTP-запит за певним URL-адресою, який зіставляється з ресурсами на веб-сервері. Якщо даний URL-адреса вказує на динамічний ресурс, то сервер запускає деяку зовнішню програму (веб-сервіс), яка формує HTML-сторінку, яка може бути показана браузером, і повертає її клієнту. Основною частиною веб-сервісу є його логіка на стороні веб-сервера.

Найчастіше веб-сервіс не є самостійною програмою (крім технології CGI), а виконується під управлінням деякої зовнішньої програми - середовища виконання. В якості такої програми можуть виступати сервер додатків або контейнер додатків. Середовище виконання робить такі дії:

– приймає від веб-сервера всю інформацію, пов'язану з HTTP-запитом;

– визначає, яка програма, створене для даного середовища, має бути виконано;

– створює всі допоміжні об'єкти, які можуть знадобитися для роботи цього додатку;

– завантажує запитувана додаток і передає йому управління;

– веб-сервіс виконується і створює HTML-сторінку, яку записує в спеціально створений для цього об'єкт (зазвичай об'єкт Response) і після цього повертає управління веб-серверу;

– веб-сервер формує HTTP-відповідь, включає в нього сформовану web-додатком HTML-сторінку і відправляє його клієнту, який надіслав запит до даного web-додатком.

Зазвичай HTML-сторінки, створені веб-сервісом, включають HTML-форми, з якими працює користувач. При натисканні користувачами на кнопки типу submit браузер відправляє новий HTTP-запит до того ж самому веб-сервісоу. Послідовність викликів веб-сервіса користувачем становить сеанс його роботи. Даний сеанс завершується або в результаті вибору користувачем команди про закінчення його роботи з веб-сервісом, або в результаті тривалого проміжку часу між відправленнями наступних HTTP-запитів (зазвичай більше 20 хвилин). Варто відзначити, що веб-сервісом призначений для вирішення деяких завдань користувачів і обов'язково має користувальницький графічний інтерфейс.

При виконанні дипломної роботи було розроблено веб-сервіс, основною метою якого є надання зручного сервісу, пов’язаного з поширенням та зберігання медіафайлів.

1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПОБУДОВИ СИСТЕМ ЗБЕРЕЖЕННЯ ТА ОБМІНУ ФАЙЛАМИ МУЛЬТИМЕДІА

1.1. Актуальність програмного продукту

Обмін файлами – це одна з найважливіших розробок у ІТ-галузі сьогодні. Усі компанії, що працюють в Інтернеті, повинні зберігати файли, які використовуються для резервного копіювання. Як правило, для цього резервного копіювання потрібен простір на диску, який людина повинна купувати у своїх господарів. Служби обміну файлами – рішення цієї проблеми непотрібних витрат. Є безліч безкоштовних сайтів для обміну файлами, які також ефективні. Через сайт для обміну файлами є можливість ділитися файлами з друзями, родиною, клієнтами та тисячами людей по всьому світу.

Сайти для обміну файлами здебільшого надійні, прості та швидкі у використанні. Вони ідеально використовуються для розміщення файлів, оскільки будь-який тип файлів можна завантажувати та використовувати для власних потреб.

Безкоштовні веб-сайти для обміну файлами важливі для веб-хостингу, оскільки це безпечний, простий і економічно вигідний спосіб обміну файлами. Одним важливим недоліком є обмеження на використаний простір на диску. Також варто відзначити що подібні програмні продукти є хорошою ідеєю для починаючих підприємств, щоб спробувати свої сили.

Однією з головних переваг онлайн-зберігання та обміну файлів є те, що співпраця відбувається безперебійно. Є можливість миттєво ділитися файлами, швидко отримувати доступ до інформації та уникати зайвих затримок. Завдяки даним системам є можливість працювати з будь-якого місця та співпрацювати в режимі реального часу з людьми в будь-якій частині світу. Сьогодні фізичне розташування не повинно бути перешкодою у прагненні створити найкращу команду на своєму робочому місці. Робітники можуть виконувати свої

обов'язки з будь-якого місця, ніби вони фізично присутні в офісі. Обмін файлами є одним з важливіших факторів для реалізації цього.

Кожна організація будується на командній роботі. Можливість спільного використання файлів, співпраці та безперебійної комунікації є вирішальним для всіх процесів на робочому місці. У минулому обмін файлами та співпраця часто були трудомісткими та дорогими. Однак, просуваючись в сучасних технологіях, ніколи не було простіше обмінюватися файлами та співпрацювати. Сьогодні все, що вам потрібно це кілька клацань, щоб виконати справи. Не потрібно витрачати папір на друк файлів або чекати кілька годин або навіть днів, щоб кур'єр доставив документи. Справа в тому, що такі технології, як хмарне сховище та віртуальна інфраструктура настільних ПК (VDI), повністю все змінили.

Основні переваги обміну файлами:

– зручність. За допомогою «хмарних проектів» є можливість легко зберігати та ділитися своїми файлами з ким завгодно та коли завгодно. Усі користувачі можуть віддалено отримувати доступ до всіх даних та програмного забезпечення з будь-якого місця, використовуючи будь-який пристрій, не зберігаючи нічого на своєму комп’ютері.

– покращення співпраці. Співпраця стає набагато простішою, коли є виділене місце у хмарі, до якого можна отримати доступ з будь-якого місця, і воно просте у використанні.

– дешево. Дослідження показують, що скорочення витрат насправді є однією з головних причин, через яку організації вирішують використовувати хмарні сервіси, а онлайн-обмін файлами - лише одна з можливостей.

– безпечність. Хмарні рішення, такі як програмне забезпечення для управління цифровими ресурсами, дозволяють легко надати доступ до своїх файлів, налаштувати різні рівні дозволів на конкретні файли та відстежувати їх використання, включаючи будь-які внесені зміни.

1.2. Огляд аналогічних додатків зберігання контенту

Перед плануванням вимог та реалізацією дипломної роботи було проаналізовано основні переваги можливості наступних сервісів зберігання файлів:

- SugarSync;
- MediaFire;
- Zippy Share;
- Mega;
- Imgur.

По суті, SugarSync представляє собою резервне копіювання файлів, яке зберігає файли у хмарі. В останні роки люди мають так багато пристроїв, що робить незручним зберігання своїх файлів лише на одному пристрої. Синхронізація дозволяє оновлювати важливі файли на багатьох комп’ютерах та пристроях, таких як iPhone або планшет.

Перш за все варто відзначити в даному додатку синхронізацію папок. Натиснувши кнопку «Керування папками синхронізації», можна розпочати переміщення файлів. На відміну від Google Drive, у даному додатку не потрібно переміщувати чи копіювати файли у вказану папку, під назвою «Синхронізація». За допомогою SugarSync є можливість зробити будь-яку папку «Синхронізованою».

Ще одним яскравим випадком роботи SugarSync є ситуація коли користувач бажає зберегти свою поточну структуру файлів. Скажімо, у користувача робочий комп'ютер і домашній комп'ютер. На його робочому комп’ютері є половина документів для особистих банківських виписок, і виникає необхідність поділитися ними з «домашнім комп'ютером». У менеджері SugarSync можна натиснути «Додати папки до SugarSync». Після чого вже й обирати на яких комп’ютерах спільно використовувати ці папки.

При завантаження фотографій SugarSync створює папку, спеціально для мобільних завантажень під час завантаження програми. Для цього просто

потрібно натиснути на «Фотографії» та торкнутися значка камери у верхньому лівому куті. Останнім кроком є вибір зображення, яке користувач бажає завантажити, і вони автоматично завантажуватимуться у мобільну папку.

Файли автоматично додаються на настільний комп'ютер користувача у папці «Фотографії» в папці «Мобільні завантаження». Єдиною стороною недоліку цього є те, що не існує способу несинхронізувати мобільну папку лише на одному пристрої, оскільки він або підключений до всіх пристроїв, або його немає. Приклад даної функції можна переглянути на рисунку 1.1.

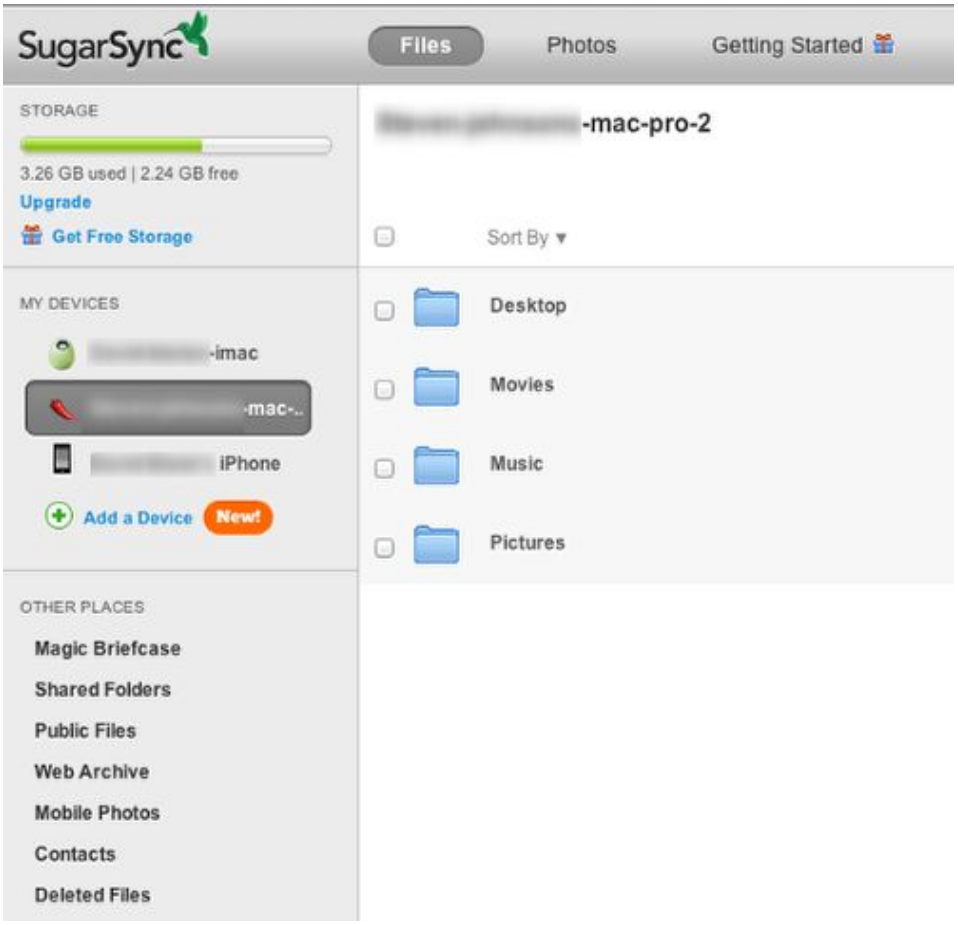


Рисунок 1.1 – Вкладка для завантаження фотографій на сервісі SugarSync

Ще одною перевагою, яка була знайдена при аналізі системи є веб-архів. У веб-архіві зберігаються файли, які користувач бажає створити для резервного копіювання. Файли завантажуються в цю папку і зберігаються лише на веб-сайті.

MediaFire – це мережева компанія для зберігання та співпраці, орієнтована на надання світові повного і простого у користуванні рішення для управління своїми цифровими речами в Інтернеті.

MediaFire було обрано для зразку через те що щомісяця понад 150 мільйонів людей використовують даний додаток для швидкого та безпечного зберігання, організації та обміну всіма своїми особистими та професійними даними у хмарі.

MediaFire є компанією, яка доставляє безкоштовні нульові терміни очікування та необмежену кількість завантажень і доступна на різних платформах.

Даний додаток доступний для:

– iPhone;
– Android;
– Windows.

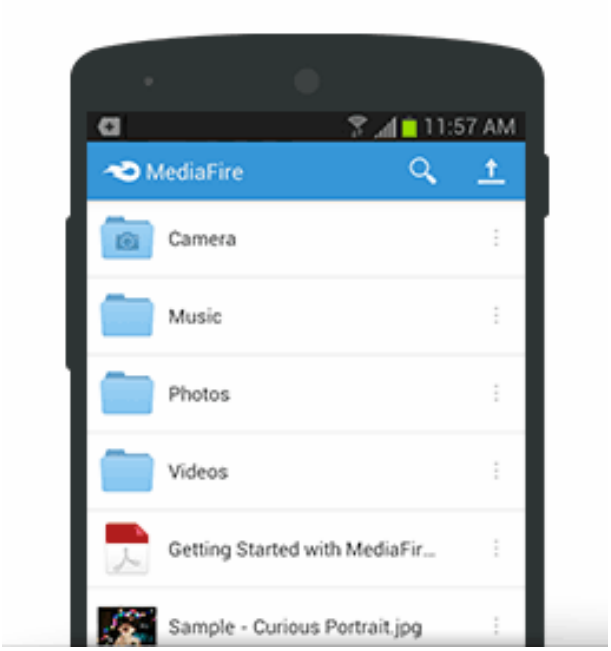


Рисунок 1.2 – Вигляд MediaFire додатку для Android

ZippyShare – служба обміну файлами, створена вже більше десяти років. ZippyShare користується особливою популярністю для людей, які використовують веб-сайт для обміну файлами як джерело матеріалів та обмін

ідеями.

Маючи максимальний ліміт завантаження розміром 500 МБ підтверджує факт, що дана служба обміну файлами спрямована на більш випадкових та орієнтованих на дозвілля користувачів.

Переваги ZippyShare:

– Безкоштовна служба;

– Легкий доступ;

– Швидке завантаження медіафайлів.

Недоліки ZippyShare:

– Файли видаляються через 30 днів;

– Мало додаткових функцій.

Говорячи детальніше про недоліки, багато користувачів ZippyShare скаржилися на автоматичне видалення файлів через 30 днів бездіяльності. Однак це насправді є досить поширеним у даній галузі, особливо це стосується менших службами для обміну файлами.



Рисунок 1.3 – Вигляд сервісу ZippyShare

ZippyShare робить зберігання та обмін файлами швидким та простим. Однак це, безумовно, не платформа, яка підходить для ділового співтовариства, а також не рекомендується для довготривалого зберігання.

Якщо користувач бажає поділитися кількома відносно невеликими файлами за короткий проміжок часу ZippyShare є чудовим додатком для цього, але для більш досконалих функцій необхідно шукати інший сервіс.

Mega являється оптимальним варіантом для користувачів, для яких конфіденційність та безпека є надзвичайно важливими. Ще однією перевагою даного сервісу є його набір додатків, які не тільки обслуговують постійних користувачів, але й мають можливості для незареєстрованих користувачів, які зайшли на сайт.

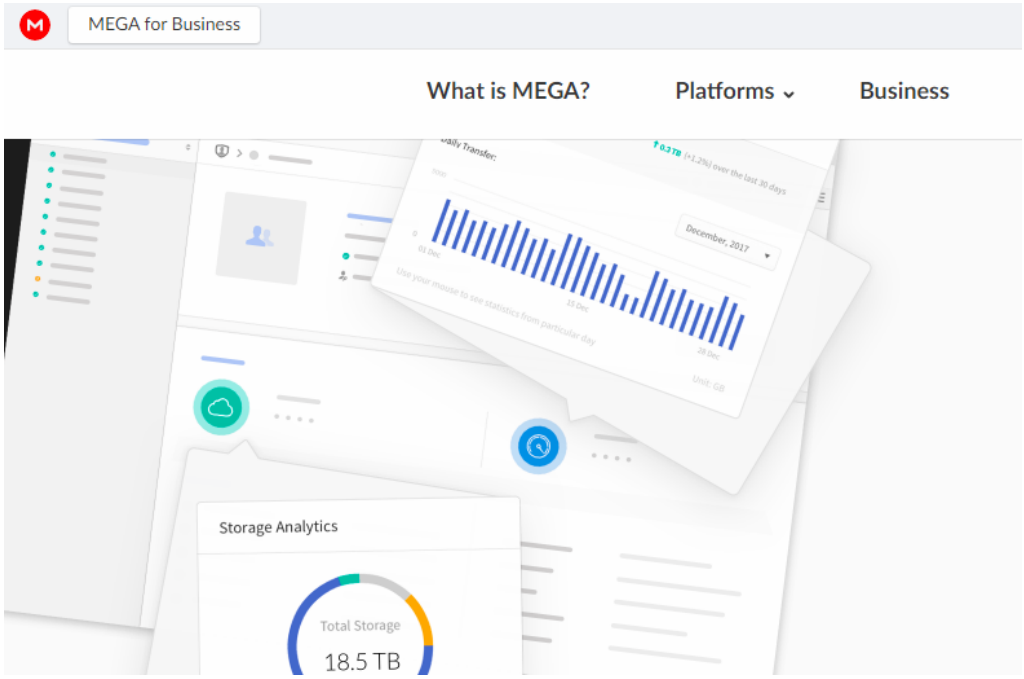


Рисунок 1.4 – Головна сторінка ресурсу зберігання контенту Mega

На відміну від аналогічних додатків, Mega розміщує на GitHub публікацію вихідного коду для свого сервісу та різних інструментів. Код наділений обмеженою ліцензією на перегляд.

Окрім того, даний сервіс є досить інтуїтивно зрозумілим та являється крос-платформенним засобом, який доступний для Windows, Mac та Linux.

Також є безкоштовний варіант використання, при якому пропонується 10 Гб пам’яті [7]. На Mega використовується система досягнень, для того щоб винагородити користувача додатковою вільною пам’ятю при виконанні певного

завдання. Однак варто зауважити що додаткову пам’ять для зберігання, який користувач отримує завдяки даній можливості, дається на тимчасовий період.

Варто відзначити і про переваги користувачів, які використовують платні послуги. Безкоштовні аккаунти мають динамічну пропускну здатність, що означає, що їхня квота передачі змінюється залежно від використання системи Mega. Користувачам, які платять, розподіляється квота переказу, яка діє протягом усього періоду платежів, а завантаження припиняються, коли вони вичерпають свою квоту.

Оскільки дана служба використовує шифрування на стороні клієнта, то у випадку якщо ви забудете свій пароль, він не зможете розшифрувати свої файли. Одним з методів для вирішення даної проблеми є ключ відновлення, який служба створює під час реєстрації.

Окрім пароля, є можливість забезпечити доступ до свого облікового запису, додатково включивши двофакторну автентифікацію. Увімкнувши налаштування, Mega попросить сканувати QR-код у додатку Authenticator. Сервіс Mega не використовує власну програму для автентифікації, а замість цього пропонує використовувати інші засоби, такі як Google Authenticator, Microsoft Authenticator, Authy тощо.

Черговим проектом для аналізу було вибрано Imgur, веб-сайт для обміну зображеннями та іншими медіафайлами, який надає перевагу користувачам соціальних медіа та соціальних новин. На головній сторінці Imgur розміщена колекція зображень та GIF-файлів, які можна сортувати за найбільш найпопулярнішою, новітньою чи найвищою оцінкою. Випадковий режим вибирає розважальні медіафайли для повного відображення та дозволяє лінійно переглядати. Зареєстровані користувачі завантажують зображення або починають з існуючих інформаційних а розважальних фото, а потім після підтвердження електронною поштою можуть обрати збереження приватного або спільного використання, використовуючи шість методів посилань, включаючи дошки для оголошень, форуми та коментарі Reddit.

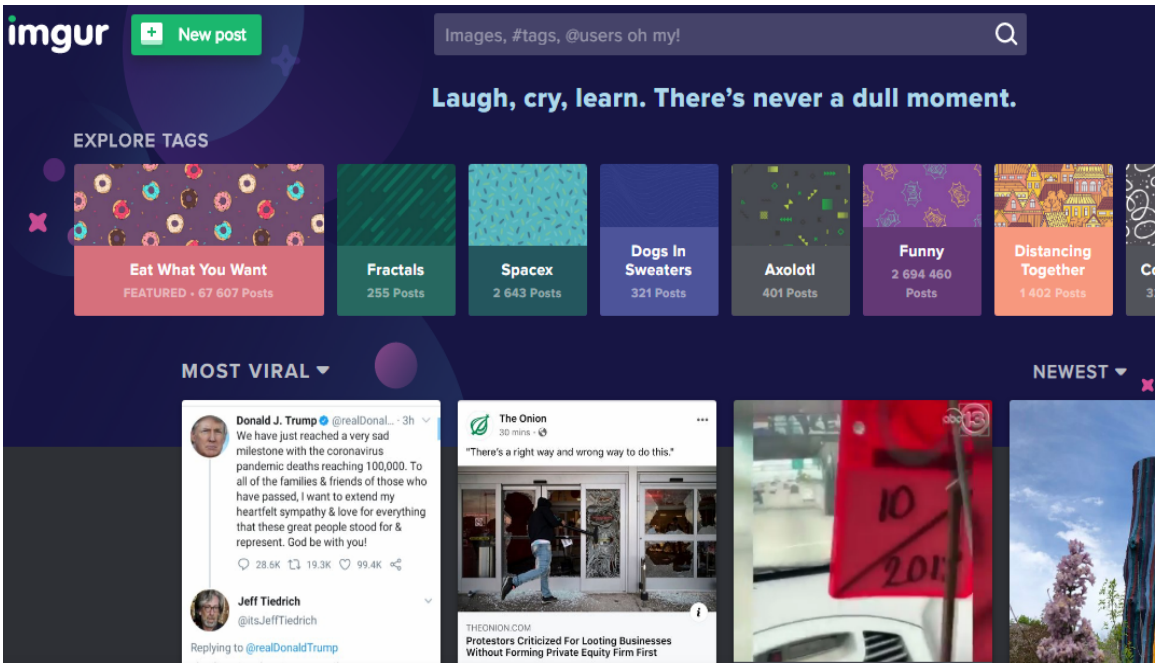


Рисунок 1.5 – Головна сторінка сайту Imgur

Згідно з статистикою, на сьогоднішній день на сайт завантажено більше ніж 300 мільйонів зображень, окрім цього перераховано 364 мільярди переглядів зображень та передано 42 петабайта даних, що являється дивовижним показником. Сайт більше орієнтований на молоде покоління, однак в цей же час він «зрілий», а також «гострий» на тематику медіа файлів.

1.3. Формування вимог до програмного сервісу поширення мультимедійного контенту

Розроблено веб-сервіс для надання зручного сервісу для зберігання та поширення медіаконтенту.

– Ролі користувачів:

– Головний адміністратор – одержує можливість видалення користувача з сайту, видалення повідомлень користувачів на сайті, а також має можливість додати нового адміністратора з списку зареєстрованих користувачів;

– Адміністратор – одержує можливість видалення повідомлень користувачів на сайті, а також має можливість додати нового адміністратора з списку зареєстрованих користувачів;

– Зареєстрований користувач – отримує власну профільну сторінку, а також має можливість додавати власні повідомлення на «форумі»;

– Незареєстрований користувач – має можливість перегляду головної сторінки сайту та можливість зареєструватись на сайті.

Можливості користувачів:

1. Головний адміністратор може видаляти користувачів з сайту;
2. Головний адміністратор може зробити зареєстрованого користувача сайту адміністратором;
3. Головний адміністратор може видаляти повідомлення користувачів на «аудіо-форумі»;
4. Адміністратор може зробити зареєстрованого користувача сайту адміністратором;
5. Адміністратор може видаляти повідомлення користувачів на «форумі»;
6. Зареєстрований користувач може пройти процес авторизації;
7. Зареєстрований користувач може змінити свій пароль на сайті;
8. Зареєстрований користувач може додавати власні медіафайли на свою сторінку, отриману після реєстрації на сайті;
9. Зареєстрований користувач може редагувати дані про доданий медіа- файл;
10. Зареєстрований користувач може видалити власний медіафайл;
11. Зареєстрований користувач може занести свій медіафайл до категорії «Улюблені»;
12. Зареєстрований користувач може занести свій медіафайл до категорії «Приховані»(інші користувачі не зможуть побачити медіафайли даної категорії);

13. Зареєстрований користувач може додати повідомлення на «аудіо-форумі»;

14. Зареєстрований користувач може прослухувати переглянути повідомлення на «форумі» від інших користувачів;

15. Зареєстрований користувач може здійснити пошук повідомлень на «аудіо-форумі» за тегом;

16. Зареєстрований користувач може редагувати власні повідомлення на «форумі»;

17. Зареєстрований користувач може переглядати всі свої надіслані повідомлення на «форумі»;

18. Зареєстрований користувач може переглядати надіслані повідомлення на «форумі» інших користувачів;

19. Зареєстрований користувач може переглянути медіафайли інших користувачів, які зберегли їх на власній профільній сторінці;

20. Зареєстрований користувач може підписатись на іншого користувача

21. Зареєстрований користувач може переглянути оформлені підписки інших користувачів сайту;

22. Незареєстрований користувач може пройти процес реєстрації на сайті;

23. Незареєстрований користувач може переглянути головну сторінку сайту.

1.4. Аналіз та вибір засобів для реалізації проекту

Для розробки дипломної роботи використовувались наступні технології:

– база даних MySQL;

– Spring Framework для розробки «бекенду» на платформі платформ IntelliJ IDEA;

– Bootstrap для розробки «фронтенду».

MySQL Workbench – це візуальний інструмент для архітекторів баз даних. MySQL Workbench забезпечує моделювання даних, розробку SQL та комплексні інструменти адміністрування для налаштування сервера, адміністрування користувачів, резервного копіювання та багато іншого. MySQL Workbench.

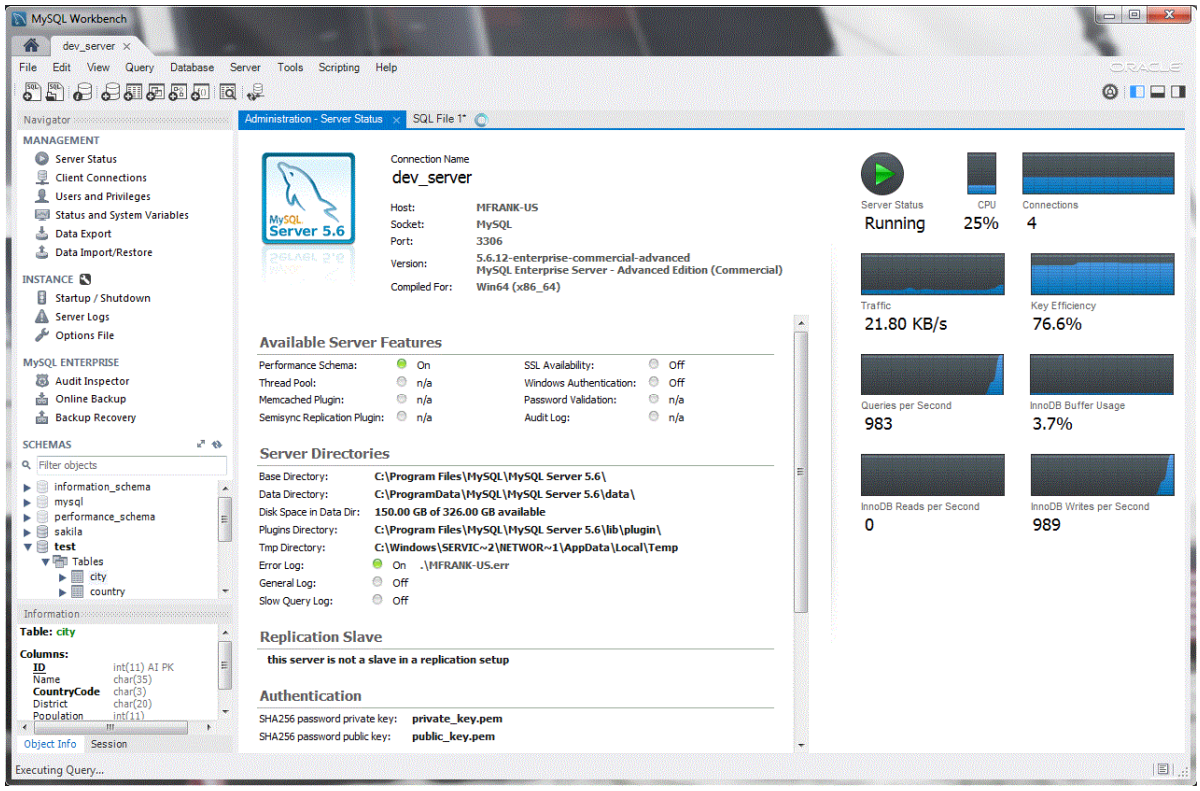


Рисунок 1.6 – Вигляд середовища MySQL

MySQL Workbench спрощує розробку та обслуговування баз даних, автоматизує трудомісткі та схильні до помилок завдання, а також покращує спілкування між DBA та командами розробників. Це дозволяє архітекторам даних візуалізувати вимоги, спілкуватися із зацікавленими сторонами та вирішувати питання дизайну до того, як будуть вкладені значні вкладення часу та ресурсів. Це дає змогу орієнтуватися на моделювання дизайну баз даних, що є найбільш ефективною методологією для створення дійсних та якісних баз даних, забезпечуючи при цьому гнучкість реагувати на розвинуті бізнес-вимоги. Утиліти для перевірки моделей та схем застосовують стандарти

найкращої практики моделювання даних, а також застосовують специфічні для MySQL стандарти фізичного проектування, щоб не було помилок під час створення нових діаграм ER або створення фізичних баз даних MySQL.

MySQL Workbench надає можливості для прямої інженерії фізичних конструкцій баз даних. Візуальну модель даних можна легко перетворити у фізичну базу даних на цільовому сервері MySQL за допомогою лише декількох клацань миші. Весь код SQL автоматично генерується та працює прямо з першого разу, що виключає звичайний схильний до помилок процес ручного запису складного SQL-коду. MySQL також дозволяє повернути інженеру існуючу базу даних або пакетну програму, щоб краще зрозуміти її дизайн бази даних. Дана СУБД не тільки може переадресовувати та реверсувати існуючі бази даних, але також може імпортувати SQL-скрипти для побудови моделей та експорту моделей до сценаріїв DDL, які можна запустити пізніше.

Візуальний редактор SQL дозволяє розробникам створювати, редагувати та виконувати запити, створювати та редагувати дані, а також переглядати та експортувати результати. Підсвічування синтаксису кольорів, довідкова контекстна допомога та автоматичне доповнення під час написання та налагодження SQL-операторів. Редактор SQL представляє з себе:

– редактор таблиць – легко редагувати дані та вносити зміни за допомогою простого формату сітки;

– вікно результатів – виконувати одночасно кілька запитів і переглядати результати тексту, зображення та просторові подання та статистику ефективності запитів на окремих вкладках;

– панель Snippet SQL – зберігайте та легко використовуйте загальний код Selects, DML та DDL;

– панель історії – перегляньте повну історію сеансів запитів та висловлювань, що показують, які запити виконувались і коли;

– панель пошуку даних таблиці – можливість пошуку даних по всій базі даних, пошук тексту в будь-якій кількості таблиць і схем, показуючи

рядки, що відповідають заданому шаблону;

– результати експорту – експорт даних результатів у загальні формати, включаючи CSV, HTML та XML.

Веб-переглядач дозволяє розробникам легко переміщуватися по схемах та об'єктах баз даних. DBA можуть візуально вибирати таблиці та поля для запиту, редагувати таблиці, створювати нові таблиці та бази даних.

Schema Inspector – використання набору детальних сіток для всіх об'єктів у їхніх схемах баз даних для перегляду, сортування та аналізу різних деталей та статистики. Зсередини схеми розробники можуть легко викликати операції з обслуговування таблиць, такі як ANALYZE та OPTIMIZE TABLE.

Оглядач об’єктів – легкий перехід, досліджування та виконання дії над об’єктами бази даних (таблиці, представлення, тригери тощо)

Object Editor – легке редагування, створення або видалення об'єктів бази даних (таблиці, представлення, тригери тощо)

Майстер з міграції MySQL Workbench призначений для економії часу DBA та розробника, забезпечуючи простоту використання візуального, точкового та натискання на всіх етапах налаштування та управління складним процесом міграції:

– управління міграційними проектами – дозволяє налаштовувати, копіювати, редагувати, виконувати та планувати міграції;

– вибір джерела та цілі – дозволяє користувачам визначати конкретні джерела даних та аналізувати вихідні дані до початку міграції.

– міграція об'єктів – дозволяє користувачам вибирати об'єкти для міграції, призначати джерело цільовим відображенням, де це необхідно, редагувати сценарії міграції та створювати цільову схему;

– оновлення версій – користувачі міграції можуть легко переміщувати бази даних зі старих версій MySQL до останніх.

Міграція баз даних дозволяє переходити з Microsoft SQL Server, Microsoft Access, PostgreSQL, Sybase ASE, Sybase SQL Anywhere, SQLite тощо.

Для реалізації програмного продукту використовувалось програмне забезпечення платформ IntelliJ IDEA останньої версі. Дане програмне середовище є однією з найкращих інтегрованих праграмних середовищ.

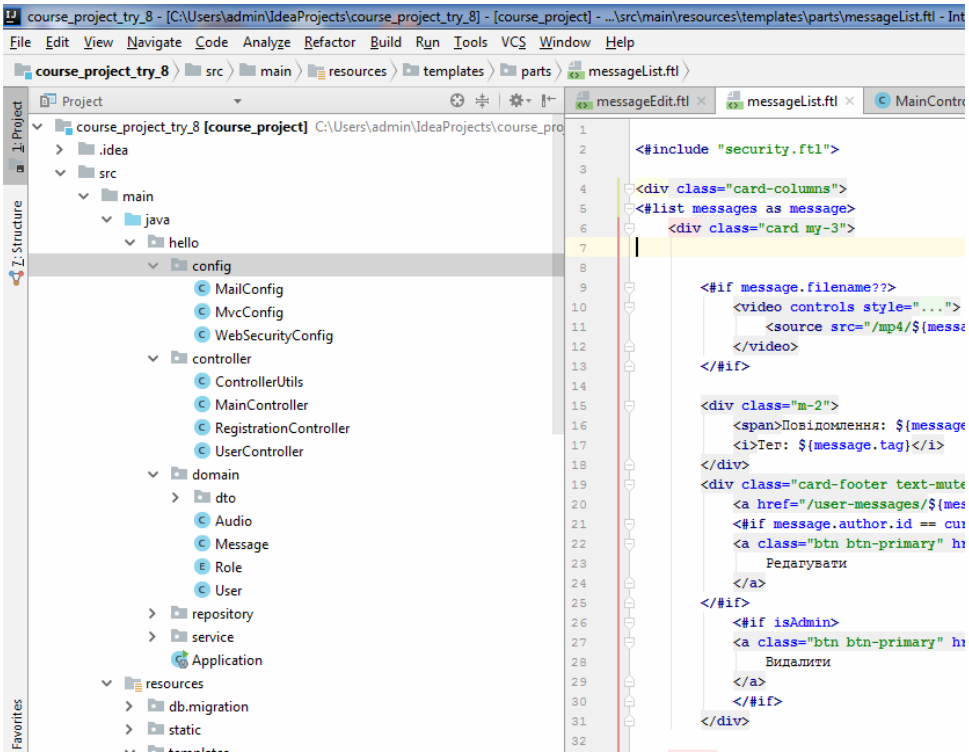


Рисунок 1.7 – Вигляд середовища IntelliJ IDEA

Перед тим як охарактеризувати Spring Framework необхідно зазначити про шаблон MVC. Шаблон Model-View-Controller (MVC) - це архітектурний шаблон, який використовується для побудови програмних систем з розділенням логіки додатку на три окремі компоненти: Model (Модель), View (Вид) та Controller (Контролер). Ось деякі характеристики кожного з цих компонентів:

- Модель (Model):

- Представляє бізнес-логіку, дані та правила обробки даних.

- Не залежить від інтерфейсу користувача.

- Може містити класи, які відповідають за доступ до бази даних або збереження/зчитування даних.

- Вид (View):

- Представляє інтерфейс користувача, через який користувач бачить і взаємодіє з додатком.

- Отримує дані з моделі та відображає їх користувачу.

- Не містить логіки додатку, вона лише відображає інформацію.

- Контролер (Controller):

- Обробляє вхідні дані від користувача, викликає відповідні методи моделі та оновлює відповідний вид.

- Відповідає за маршрутизацію та обробку запитів користувача.

- Не має власного стану, але він може змінювати стан моделі.

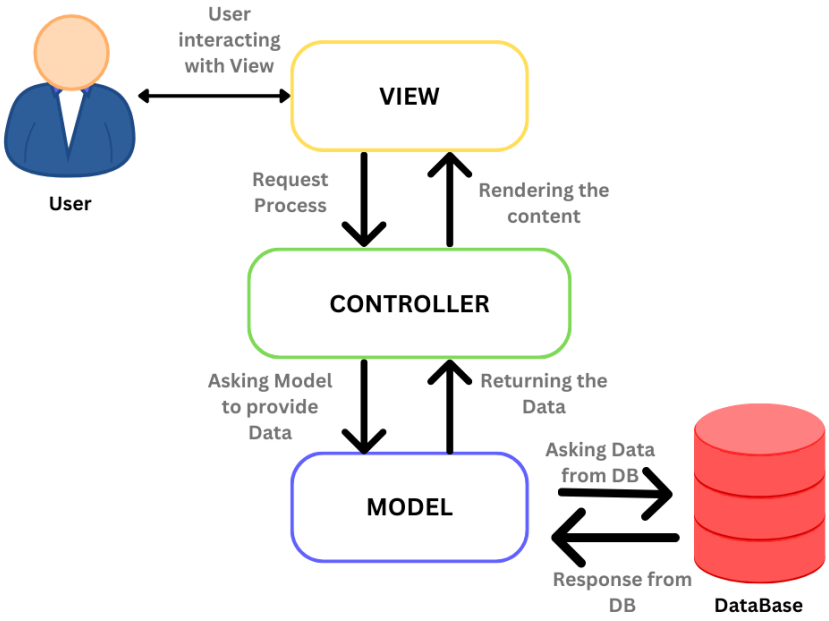


Рисунок 1.8 – Структура MVC

Spring Framework - це один з найпопулярніших інструментів у світі Java для розробки програмного забезпечення. Він надає широкий набір функцій та інструментів для побудови різноманітних застосунків, включаючи веб-додатки, мікросервіси, батч-процеси та багато іншого. Ось деякі характеристики Spring Framework:

- Інверсія управління (Inversion of Control, IoC). Spring використовує принцип Інверсії управління для керування об'єктами та їх залежностями. Це

дозволяє знизити залежності між класами і сприяє створенню більш модульного та підтримуваного коду.

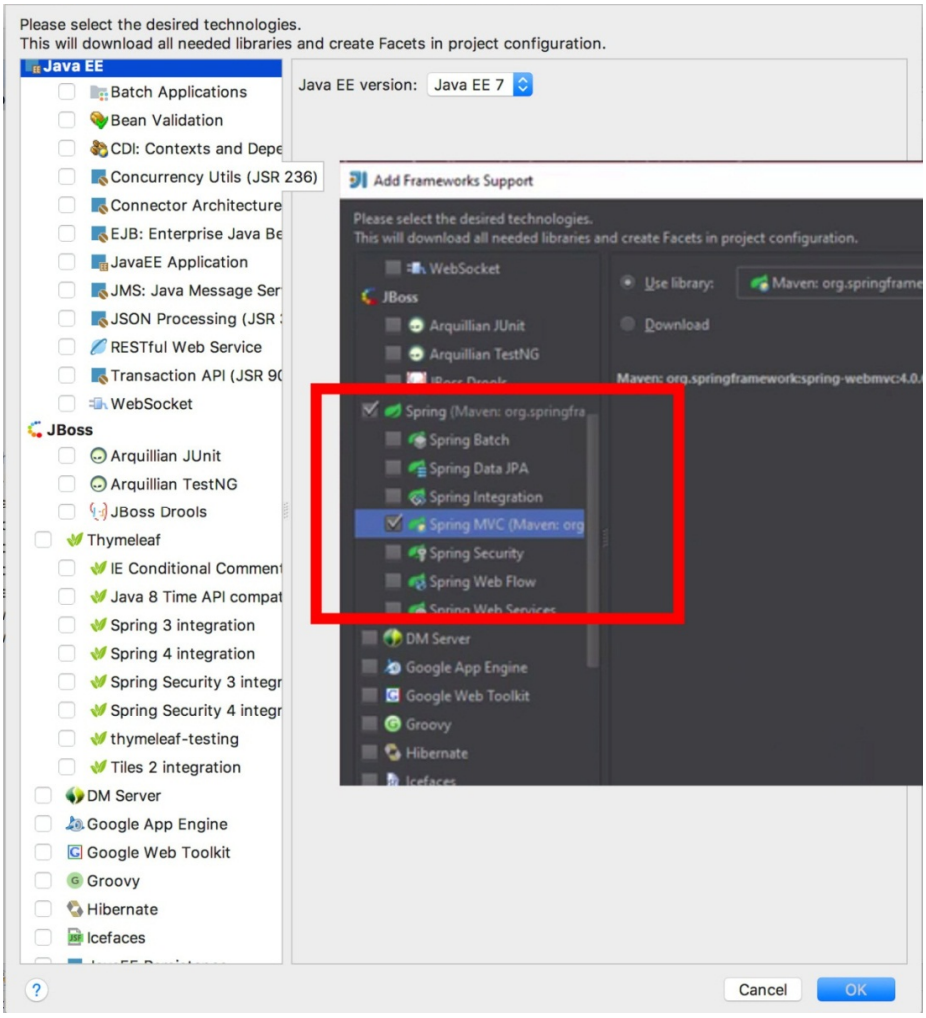


Рисунок 1.9 – Spring framework в середовищі IntelliJ IDEA

- Введення (Dependency Injection, DI). Spring використовує механізм Введення для надання залежностей об'єктам в часі виконання. Це полегшує створення та управління об'єктами та знижує зв'язаність коду.

- Модульність та розширюваність. Spring Framework складається з ряду модулів, які можна використовувати окремо або разом залежно від потреб проекту. Це забезпечує гнучкість та розширюваність для розробки.

- Підтримка аспектно-орієнтованого програмування (Aspect-Oriented Programming, AOP). Spring надає підтримку для AOP, що дозволяє виділяти

повторюваний код (наприклад, логування, транзакції) в окремі аспекти, що полегшує управління бізнес-логікою.

- Підтримка веб-розробки. Spring Framework надає різноманітні інструменти для розробки веб-додатків, включаючи підтримку моделі MVC (Model-View-Controller), інтеграцію з різними фронтенд-технологіями та підтримку безлічі протоколів, таких як HTTP та WebSocket.

- Підтримка транзакцій. Spring Framework надає підтримку декларативного управління транзакціями, що дозволяє просто налаштувати та управляти транзакціями в додатку.

- Безпека. Spring Security - це модуль Spring Framework, який забезпечує потужні засоби для аутентифікації, авторизації та управління безпекою веб-додатків.

Bootstrap - це популярний фреймворк для розробки веб-додатків та веб-сайтів з використанням HTML, CSS і JavaScript. Ось деякі характеристики Bootstrap:

- Відзначена адаптивність (Responsive Design). Bootstrap надає готові компоненти та класи для створення адаптивних веб-сайтів, які коректно відображаються на різних пристроях та розмірах екранів.

- Гнучка сітка (Grid System). Фреймворк має вбудовану гнучку сітку з колонками, яка дозволяє легко створювати різноманітні макети веб-сторінок, забезпечуючи їх адаптивність.

- Готові компоненти (Pre-styled Components). Bootstrap містить велику кількість готових компонентів, таких як кнопки, форми, навігаційні панелі, каруселі, модальні вікна та багато інших, що дозволяє швидко розробляти веб-сайти.

- Типографіка та стилізація (Typography and Styling). Bootstrap має вбудовані стилі для різних елементів HTML та розміщення тексту, що спрощує стилізацію веб-сайтів та забезпечує єдність в оформленні.

- Підтримка браузерів (Cross-browser Compatibility). Bootstrap забезпечує сумісність з різними веб-переглядачами, включаючи Chrome, Firefox, Safari, Edge та Internet Explorer, що полегшує розробку веб-сайтів.

- Розширення та налаштування (Extensions and Customization). Фреймворк дозволяє розширювати та налаштовувати його за допомогою власних CSS-стилів та JavaScript-функцій, щоб відповідати потребам конкретного проекту.

- Спільнота та документація (Community and Documentation). Bootstrap має велику спільноту користувачів та активну документацію, яка надає вичерпні інструкції та приклади використання, що полегшує вивчення та використання фреймворку.

Офіційний сайт Bootstrap представлено на рисунку 1.10.

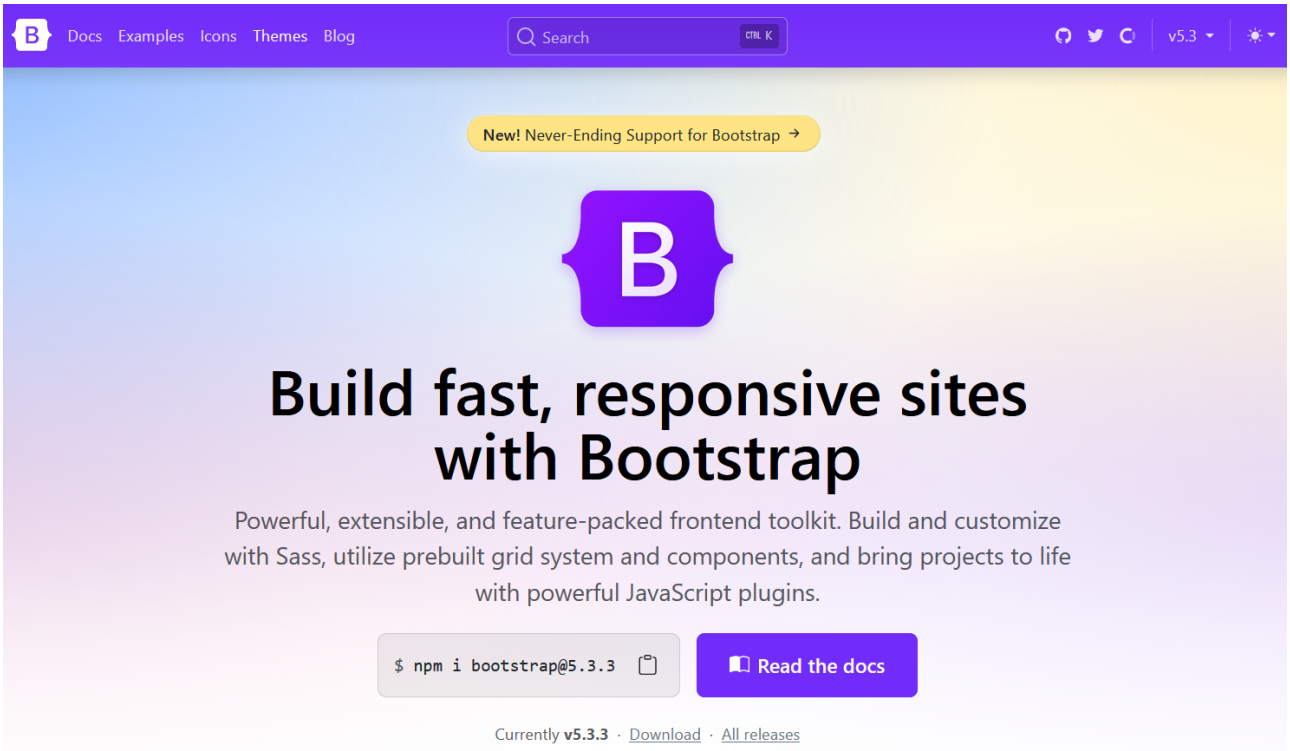


Рисунок 1.10 – Офіційний сайт Bootstrap

Основні переваги Bootstrap:

– мінімальна кількість помилок у браузері;

– гнучка система, яка підтримує сумісність з різними браузерами браузерів та CSS;

– легко налаштовується;

– чіткі структури та стилі;

– наявність плагінів JavaScript, реалізованих за допомогою jQuery;

– хороша документація та підтримка ;

– безкоштовні та професійні шаблони;

– відмінна система сітки.

Недоліками Bootstrap є:

– необхідна велика кількість скасування стилів або перезаписування файлів у випадку якщо дизайн відхиляється від звичних стилів, що використовується в Bootstrap. Таким чином, це може призвести до великої кількості втраченого часу на розробку та кодування веб-сайту;

– необхідно розбиратись в налаштуваннях для створення дизайну, в іншому випадку програмні продукти з Bootstrap виглядатимуть однаково;

– стилі є багатомісткими і можуть призвести до значного виводу в HTML;

– JavaScript прив’язаний до jQuery і є однією із найпоширеніших бібліотек, завдяки чому більшість плагінів не використовуються;

1.5. Висновки по розділу

В даному розділі було розглянуто програмні продукти з подібними функціональними вимогами. Сформовано аналіз вимог до власної дипломної роботи, а також проаналізовано засоби для його подальшої реалізації. Окрім цього, було встановлено такий недолік як зручна можливість спілкування та комунікації між користувачами даних сервісів, що являється однією з вимог до дипломної роботи.

2. РОЗРОБКА АЛГОРИТМІЧНОГО ТА МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ПОШИРЕННЯ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ

2.1. Реалізація алгоритмічних схем функціонування системи

Планування функцій дипломної роботи є одним з ключових етапів для його успішної реалізації. Згідно з попередньо сформованими вимогами розроблено наступні алгоритми по функціоналу:

- Реєстрація користувачів;
- Авторизація користувачів;
- Зміна паролю;
- Ролі користувачів;
- Процес додавання медіафайлу;
- Медіафайли в профілі користувача;
- Медіафайли в загальному чаті;
- Процес редагування записів або;
- Процес видалення записів.

Алгоритми було відображено за допомогою блок схем. Блок схеми представляють собою графічну модель, за допомогою якої часто здійснюються представлення планування програмних функцій, оскільки за допомогою даних схем можна зручно відобразити такі операції як розгалуження, цикли, ввід даних тощо. Дані схеми являються зручними за допомогою відповідних елементів, які відповідно представляють певну окрему можливість. Саме вищеперечислені алгоритми по функціоналу і будуть зображені у вигляді даних схем.

Алгоритми реєстрації і авторизації є першочерговими при плануванні, оскільки тільки зареєстровані користувачі на сайті матимуть можливість

виконувати операції з медіафайлами. Алгоритм реєстрації представлена на рисунку 2.1.



Рисунок 2.1 – Алгоритм реєстрації користувачів

Процес авторизації відрізняється кількістю полів для перевірки, а також відсутністю ключа активації. Даний алгоритм відображений на рисунку 2.2.

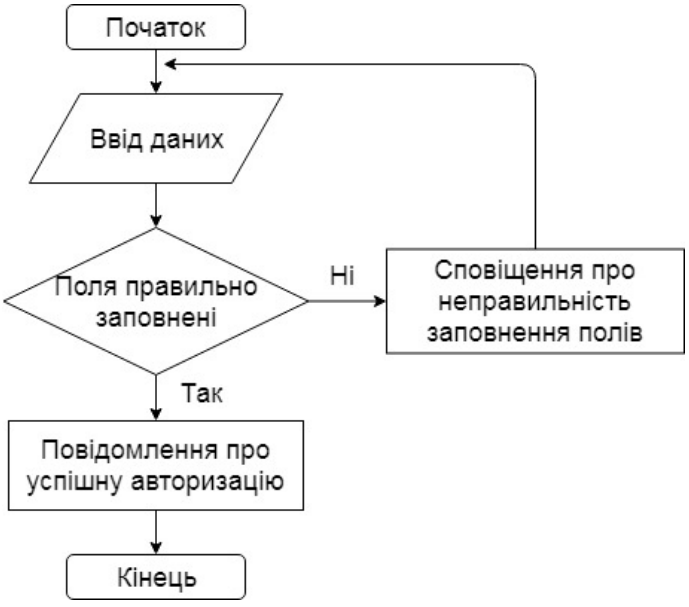


Рисунок 2.2 – Алгоритм авторизації користувачів

Варто врахувати необхідність в зміні паролю у випадку якщо користувач його забуде. Даний алгоритм буде відбуватись безпосередньо через електронну пошту та представлений на рисунку 2.3.

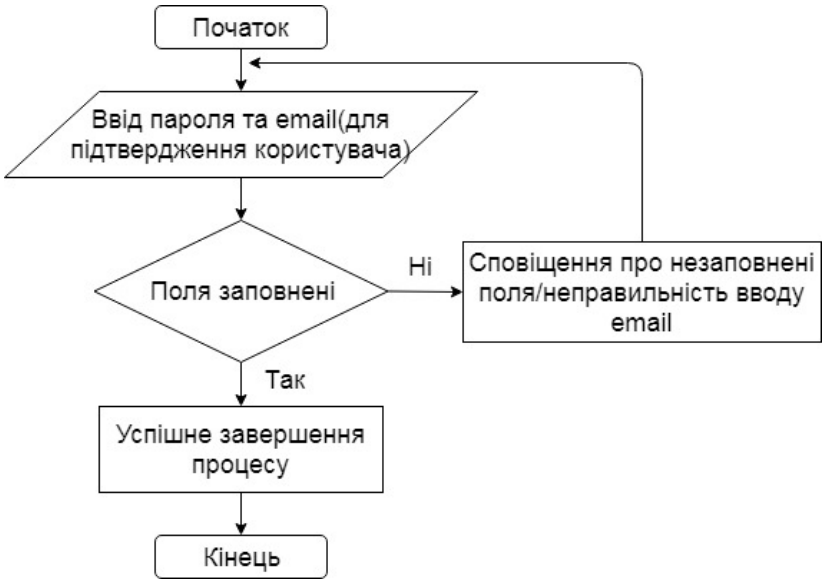


Рисунок 2.3 – Алгоритм зміни паролю

Ще одним кроком являється розподіл користувачів на ролі, оскільки окрім зареєстрованих та незареєстрованих користувачів є інші ролі, представлені на рисунку 2.4.

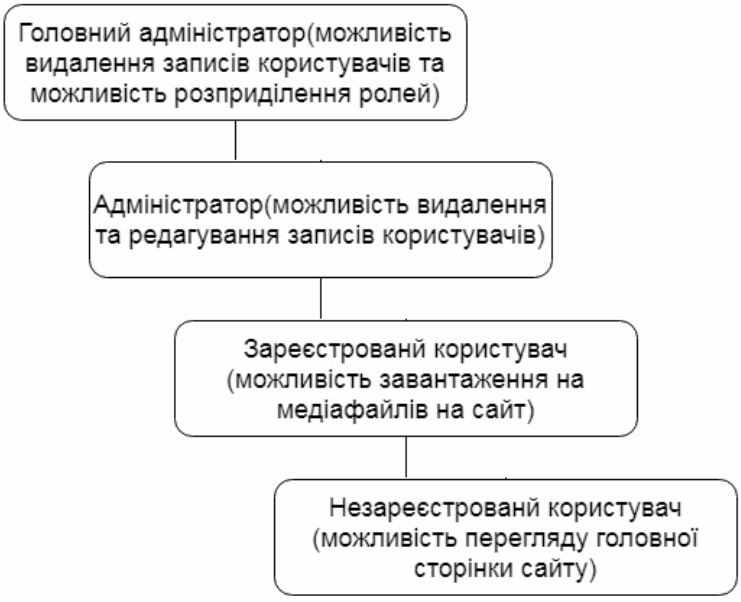


Рисунок 2.4 – Ролі користувачів

Після розприділення ролей та можливістю реєстрації наступним плануванням є безпосередня робота з медіа файлами. Алгоритм завантаження медіафайлів відображена на рисунку 2.5

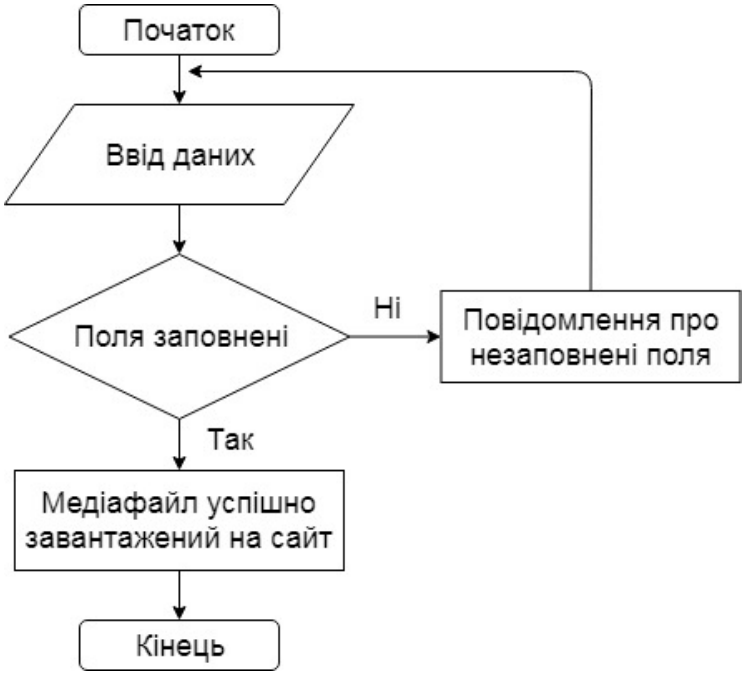


Рисунок 2.5 – Процес додавання медіафайлів

Передбачена можливість додавання медіа файлів на профільну сторінку користувача, а також на загальний час для всіх користувачів. Схема основних властивостей медіа файлів в профілі представлено на рисунку 2.6.



Рисунок 2.6 – Властивості медіафайлів в профілі користувача

Схема основних властивостей медіафайлів в загальному чаті представлено на рисунку 2.7.

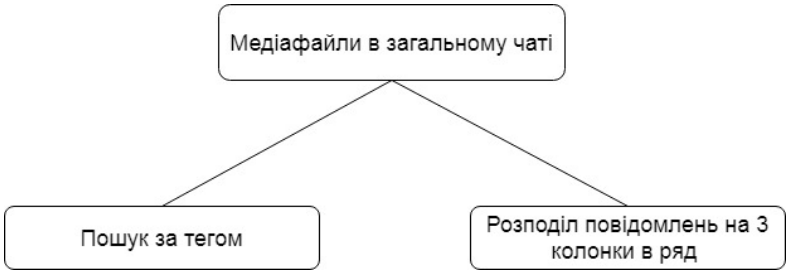


Рисунок 2.7 – Властивості медіафайлів в загальному чаті

Схема процесу видалення представлена на рисунку 2.8

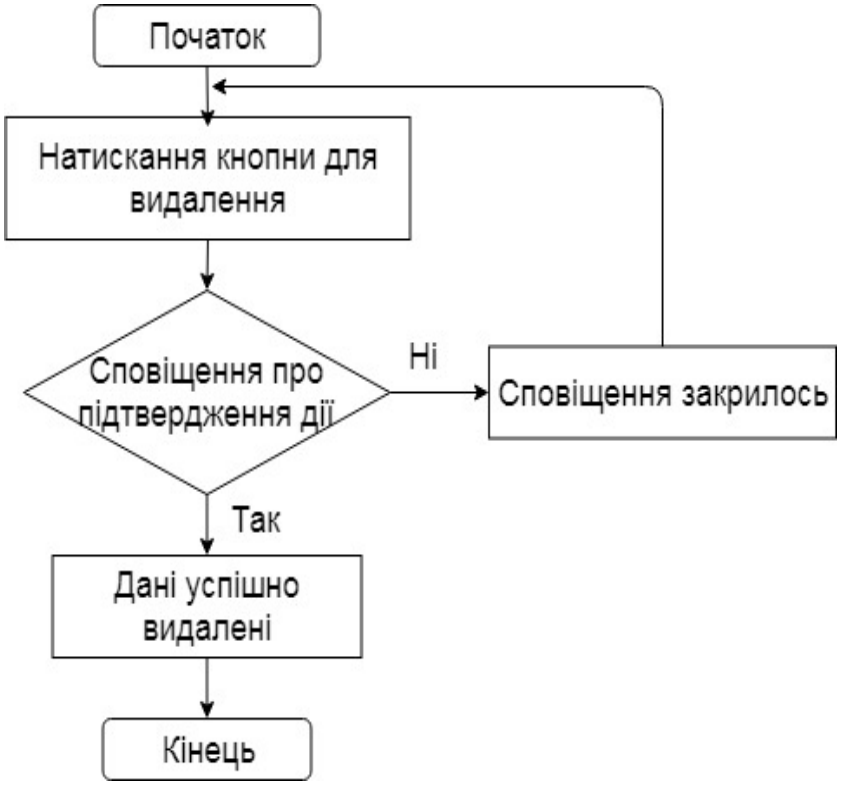


Рисунок 2.8 – Алгоритм процесу видалення

Важливим етапом планування дипломної роботи являється редагування і видалення записів. Кожен користувач матиме можливість змінити власні записи, в той же час адміністратори матимуть можливість змінити записи кожного користувача. Окрім цього варто відзначити що головний

адміністратор, який матиме можливість редагувати профілі користувачів в адміністративній панелі, за допомогою яких і надати певному користувачу статус адміністратора. Схема процесу редагування представлена на рисунку 2.9.

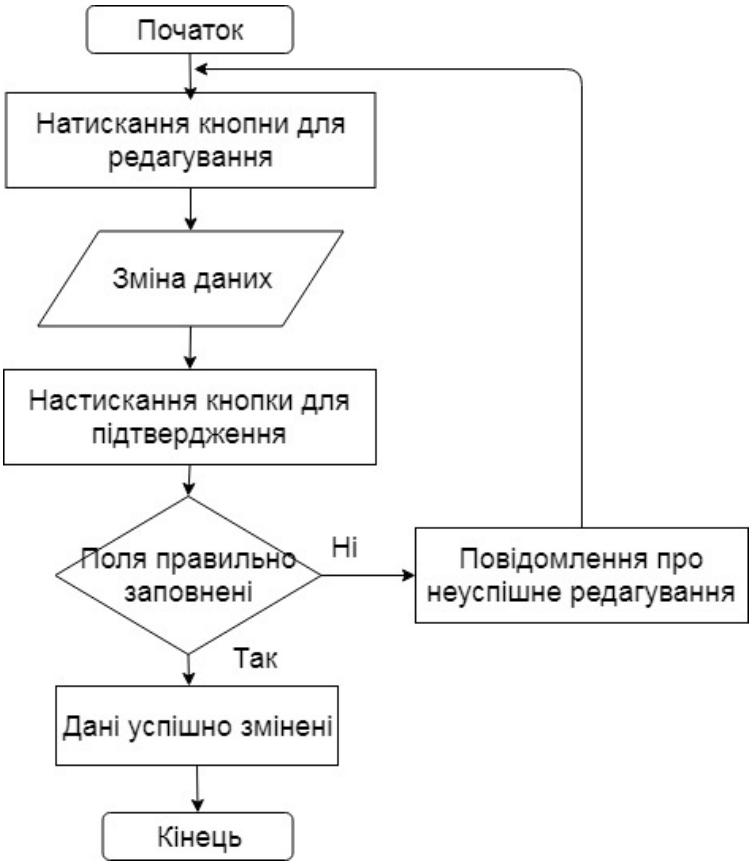


Рисунок 2.9 – Алгоритм процесу редагування

В результаті реалізації даних схем було представлено основні алгоритми роботи системи зберігання та поширення мультимедійного контенту.

2.2. Представлення математичного забезпечення системи

Захист та безпека програмного продукту є важливо складовою для його успішної працездатності. Існують загрози безпеки, які базуються на нападі окремих систем, але є й такі, які є поширені у хмарних системах. До них належать зловживання та недобросовісне використання хмарних ресурсів,

незахищених інтерфейсів та API, невідомого профілю ризику, а також втрата чи витік даних.

Кожна категорія загрози, описана в STRIDE, має відповідний набір контрзаходів, які слід використовувати для зменшення ризику. STRIDE – це абревіатура, що включає в себе

- підробка ідентичності користувача,
- підробка даних,
- відмова,
- розкриття інформації,
- відмова в обслуговуванні,
- підвищення привілегій.

Список загроз та контрзаходів від STRIDE наведений в таблиці 2.1.

Таблиця 2.1 – Список загроз та контрзаходів від STRIDE

Загрози	**Заходи**
Підробка ідентичності користувача	– Використання сильної аутентифікації; – Не зберігати паролі в простому тексті; – Не передавати повноваження в простому тексті; – Захистити файли cookie аутентифікації за допомогою SSL; – Використовувати хешування; – Використовувати цифрові підписи.
Підробка даних	– Використання «сильної» авторизації; – Використання протоколів, захищених від несанкціонованого втручання.
Відмова	– Використання цифрових підписів; – Використання «сильної» авторизації.
Розкриття інформації	– Використання «сильного» шифрування; – Захищені комунікаційні зв’язки з протоколами.
Відмова в обслуговуванні	– Валідація та фільтри для введення.

Те, як віруси атакують організм людської клітини, нагадує потенційні атаки на хмарні системи. Хмарні провайдери намагаються ізолювати віртуальні машини від фізичної машини для підтримки безпеки даних, які можуть атакуватися та знищувати всю систему. Динамічний обмін ресурсами з фізичного сервера або серед віртуальних машин самі допоможуть вірусам розгортатися їхню атаку від одного VM до іншого, поки вся система не постраждає.

Можна припустити, що VM 1 є джерелом атаки та переносить віруси. Завдяки спільному доступу до ресурсів, це впливає на будь-яку машину в хмарі. У випадку якщо фізична машина уражена, є ймовірність що VM 2, VM 3, ... , VM n постраждають і процес триває, поки вся хмара не буде уражена. Цей сценарій показаний на рисунку 2.10 урахуванням багатьох віртуальних машин:

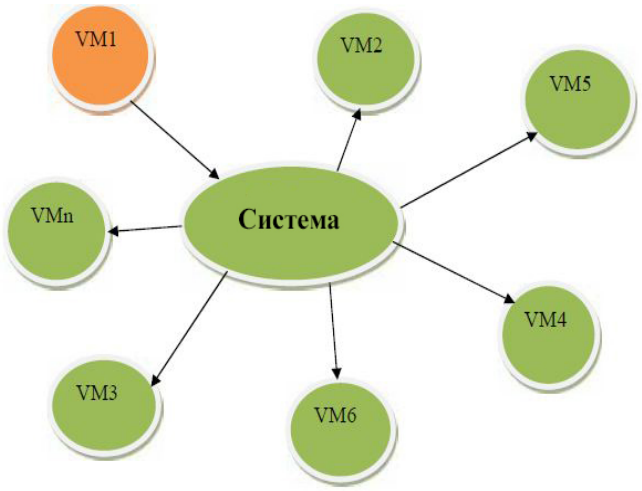


Рисунок 2.10 – Сценарій атаки на VM

Припустимо, що принаймні одна VM уражена, віруси поширюється на інші VM. Динамічний спільний доступ до ресурсів серед віртуальних машин дозволяє зазнати впливу від вірусу на всю хмару. У цій моделі варто використовувати рандомізований алгоритм Лас-Вегаса (LVRA) який стверджує, що «ви завжди знайдете рішення, якщо рішення взагалі знайдеться». Завдання полягає в тому, щоб з’ясувати, як успішно моделювати першу атаку.

У цій моделі використовуються випадкові послідовності Фібоначчі. Це можна продемонструвати нижче:

Враховуючи, що:

$$L_n = \frac{1}{y} \cdot (a_{n-1} + a_{n+1}) \quad (2.1)$$

де $L_0 = 2$, $L_1 = 1$,

За початкову умову береться той факт що жодних вузлів не постраждало, однак пізніше «вузол 1» може бути уражений. L_n – кількість вузлів, які можуть бути ураженими. a_n – детермінована різниця, представлена за формулою 2.2:

$$a_{n+1} = a_{n-1} + a_n \quad (2.2)$$

де $a_n = 0$, $a_1 = 1$, $n \geq 2$

Згідно з формули 2.1, можна вважати що всі VM взаємодіють з кожним, а y – випадкова послідовності Фібоначчі.

2.3. Проектування алгоритмів хешування, пошуку та сортування даних

Важливими для успішної реалізації проекту системи поширення мультимедійного контенту є проектування наступних алгоритмів:

– хешування даних;

– алгоритм пошуку;

– алгоритм сортування.

Для прискорення доступу до даних можна використовувати попереднє їх впорядкування у відповідності зі значеннями ключів. При цьому можуть використовуватися методи пошуку в упорядкованих структурах даних,

наприклад, метод дихотомічного пошуку, що суттєво скорочує час пошуку даних за значенням ключа. Проте при добавленні нового запису потрібно дані знову впорядкувати. Втрати часу на повторне впорядкування можуть значно перевищувати виграш від скорочення часу пошуку. Тому для скорочення часу доступу до даних використовується так зване випадкове впорядкування або хешування. При цьому дані організуються у вигляді таблиці за допомогою хеш-функції h, яка використовується для «обчислення» адреси за значенням ключа.



Рисунок 2.11 – Хешування даних

Хешування використовує аналогічний підхід, відображаючи елементи в хеш-таблиці. Алгоритм хешування використовує деяку функцію, яка визначає імовірне розміщення елемента в таблиці на основі значення шуканого елементу.

Ідеальною хеш-функцією є така хеш-функція, яка для будь-яких двох неоднакових ключів дає неоднакові адреси. Підібрати таку функцію можна у випадку, якщо всі можливі значення ключів відомі наперед. Така організація даних носить назву «досконале хешування».

Наприклад, потрібно запам’ятати декілька записів, кожен з яких має унікальний ключ зі значенням від 1 до 100. Для цього можна створити масив з 100 комірками і присвоїти кожній комірці нульовий ключ. Щоб добавити в масив новий запис, дані з нього просто копіюються у відповідну комірку масиву. Щоб добавити запис з ключем 37, дані з нього копіюються в 37 позицію в масиві. Щоб знайти запис з певним ключем – вибирається відповідна комірка масиву. Для вилучення запису ключу відповідної комірки масиву

просто присвоюється нульове значення. Використовуючи цю схему, можна добавити, знайти і вилучити елемент із масиву за один крок.

У випадку наперед невизначеної множини значень ключів і обмеженні розміру таблиці підбір досконалої функції складний. Тому часто використовують хеш-функції, які не гарантують виконанні умови.

Наприклад, база даних співробітників може використовувати в якості ключа ідентифікаційний номер. Теоретично можна було б створити масив, кожна комірка якого відповідала б одному з можливих чисел; але на практиці для цього не вистарчить пам’яті або дискового простору. Якщо для зберігання одного запису потрібно 1 КБ пам’яті, то такий масив зайняв би 1 ТБ (мільйон МБ) пам’яті. Навіть якщо можна було б виділити такий об’єм пам’яті, така схема була б дуже неекономною. Якщо штат фірми менше 10 мільйонів співробітників, то більше 99 процентів масиву буде пустою.

Щоб вирішити цю проблему, схеми хешування відображають потенційно велику кількість можливих ключів на достатньо компактну хеш-таблицю. Якщо на фірмі працює 700 співробітників, можна створити хеш-таблицю з 1000 комірок. Схема хешування встановлює відповідність між 700 записами про співробітників і 1000 позиціями в таблиці. Наприклад, можна розміщати записи в таблиці у відповідності з трьома першими цифрами ідентифікаційного номеру. При цьому запис про співробітника з номером 123456789 буде знаходитися в 123 комірці таблиці.

Очевидно, що оскільки існує більше можливих значень ключа, ніж комірок в таблиці, то деякі значення ключів можуть відповідати одним і тим коміркам таблиці. Даний випадок носить назву «колізія», а такі ключі називаються «ключі-синоніми».

Щоб уникнути цієї потенційної проблеми, схема хешування повинна включати в себе алгоритм вирішення конфліктів, який визначає послідовність дій у випадку, якщо ключ відповідає позиції в таблиці, яка вже зайнята іншим записом.

Усі методи використовують для вирішення конфліктів приблизно однаковий підхід. Вони спочатку встановлюють відповідність між ключем запису і розміщенням в хеш-таблиці. Якщо ця комірка вже зайнята, вони відображають ключ на іншу комірку таблиці. Якщо вона також вже зайнята, то процес повторюється знову до тих пір, поки нарешті алгоритм не знайде пусту комірку в таблиці. Послідовність позицій, які перевіряються при пошуку або вставці елемента в хеш-таблицю, називається тестовою послідовністю.

В результаті, для реалізації хешування необхідні три речі:

– Структура даних (хеш-таблиця) для зберігання даних;

– Функція хешування, яка встановлює відповідність між значеннями ключа і розміщенням в таблиці;

– Алгоритм вирішення конфліктів, який визначає послідовність дій, якщо декілька ключів відповідають одній комірці таблиці.

Одна з тих дій, які найбільш часто зустрічаються в програмуванні – пошук. Існує декілька основних варіантів пошуку, і для них створено багато різноманітних алгоритмів.

Задача пошуку – відшукати елемент, ключ якого рівний заданому «аргументу пошуку». Отриманий в результаті цього індекс забезпечує доступ до усіх полів виявленого елемента.

Одним з найпростіших методів пошуку є послідовне порівняння першого символу *s* з символами масиву *a*. Якщо наявний збіг, тоді порівнюються другі, треті,... символи аж до повного збігу рядка *s* з частиною вектору такої ж довжини, або до незбігу у деякому символі. Тоді пошук продовжується з наступного символу масиву *a* та першого символу рядку *s*. Це визначається елементарною програмою:

```
i = 0;     // номер символу масиву a
while (i < N - lenghts)
{
 j = 0;    // номер символу рядка s
 while ((s[j] == a[i+j]) && (j<lenghts)
```

```
    j++;
  if (j==lenghts)
    return i;     // успіх
}
```

Якщо збіги відбуватимуться досить часто, то час роботи програми може бути досить значним.

Існує варіант удосконалення цього алгоритму – це починати пошук після часткового збігу не з наступного елементу масиву, а з символу, наступного після тих, що переглядалися, якщо у рядку *s* немає фрагментів, що повторюються.

```
j = 0; // j - номер символа у a
found = false;
while (!found)
{
 i = 0;    // i - номер символа у s
 while ((s[i] == a[j]) && (s[i] != '/0')
 {
   i++;
   j++;
 };
 if (s[i] == '/0'
   found = true;
 else
   j -= i-1;
};
```

Зі всіх задач програмування сортування має найбагатший вибір алгоритмів розв'язку. Є багато чинників, які впливають на вибір алгоритму, а саме:

1. Наявний ресурс пам'яті: повинні вхідна й вихід множини розташовуватися в різних ділянках пам'яті, чи вихідна множина може бути сформована на місці вхідної. В останньому випадку наявна ділянка пам'яті

повинна в ході сортування динамічно перерозподілятися між вхідною і вихідною множинами; для одних алгоритмів це пов’язане з великими витратами, для інших – з меншими.

2. Початкова впорядкованість вхідної множини: у вхідній множині можуть попадатися впорядковані ділянки. В граничному випадку вхідна множина може виявитися вже впорядкованою. Одні алгоритми не враховують початкової впорядкованості і вимагають одного і того ж часу для сортування будь-якої множини даного обсягу, інші виконуються тим швидше, чим краще впорядкованість на вході.

3. Часові характеристики операцій: при визначенні порядку алгоритму час виконання вважається звичайно пропорційним кількості порівнянь ключів. Ясно, проте, що порівняння числових ключів виконується швидше, ніж стрічкових, операції пересилки, характерні для деяких алгоритмів, виконуються тим швидше, ніж менший об’єм записів. Залежно від характеристик запису таблиці може бути вибраний алгоритм, що забезпечує мінімізацію числа тих чи інших операцій.

4. Складність алгоритму є не останнім міркуванням при його виборі. Простий алгоритм вимагає меншого часу для його реалізації і вірогідність помилки в реалізації його менше. При промисловому виготовленні програмного продукту вимоги дотримання термінів розробки і надійності продукту можуть навіть превалювати над вимогами ефективності функціонування.

Алгоритм сортування називається усталеним, якщо у відсортованому масиві він не змінює порядку розташування елементів.

Ефективність методів сортування визначається двома параметрами:

– кількістю порівнянь;

– кількістю пересилань елементів.

Різноманітність алгоритмів сортування вимагає деякої їхньої класифікації. Вибраний один з вживаних для класифікації підходів, орієнтований перш за все на логічні характеристики використовуваних

алгоритмів. Згідно цьому підходу будь-який алгоритм сортування використовує одну з наступних чотирьох стратегій (або їхню комбінацію).

1. Стратегія вибірки. З вхідної множини вибирається наступний за критерієм впорядкованості елемент і включається в вихідну множину на місце, наступне за номером.

2. Стратегія включення. З вхідної множини вибирається наступний за номером елемент і включається в вихідну множину на те місце, яке він повинен займати відповідно до критерію впорядкованості.

3. Стратегія розподілу. Вхідна множина розбивається на ряд підмножин і сортування ведеться у середині кожної такої підмножини.

4. Стратегія злиття. Вихідна множина отримується шляхом злиття маленьких впорядкованих підмножин.

При використанні операції обміну використовується функція обміну значень двох комірок з цілими числами (функції пересилки даних можна уникнути, якщо сортування проводиться на зв'язних структурах даних – тоді відбувається зміна посилань на відповідні комірки):

```
void Swap(int & a, int & b)
{
     int tmp = a; a = b; b = tmp;
}
```

Сортування простим включенням – „дослівна" реалізації стратегії включення. Порядок алгоритму сортування простим включенням – , якщо враховувати тільки операції порівняння. Але сортування вимагає ще й в середньому переміщень, що робить її в такому варіанті значне менш ефективною, ніж сортування вибіркою.

Алгоритм сортування простим включенням ілюструється прикладом.

```
void Sort(int *a, int *b)
{
     int i, j, k;
```

```
        for ( i=0; i<N; i++ )  // перебір вхідного масиву
        {
             // пошук місця для в масиві виходу
             j = 0;
             while ((j<i) && (b[j]<=a[i]))
                  j++;
             // звільнення місця для нового ел-та
             for ( k=i; k>j; k-- )
                  b[k] = b[k-1];
             b[j] = a[i];// запис в масив виходу
        };
   }
```

В результаті даних алгоритмів було реалізовано хешування, пошуку та сортування, які було реалізовано в дипломній роботі.

2.4. Висновки по розділу

У цьому розділі було представлено алгоритмічне і математичне забезпечення системи зберігання та поширення мультимедійного контенту, проведено аналіз існуючих методів, моделей і алгоритмів для реалізації програмного продукту.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗБЕРІГАННЯ ТА ПОШИРЕННЯ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ

3.1. Побудова бази даних

В результаті опрацювання предметної області, було виділено наступні об’єкти предметної області:

- Користувачі;
- Ролі;
- Медіафайли;
- Повідомлення;
- Підписки.

ER-діаграма системи зберігання та поширення мультимедійного контенту представлена на рисунку 3.1.

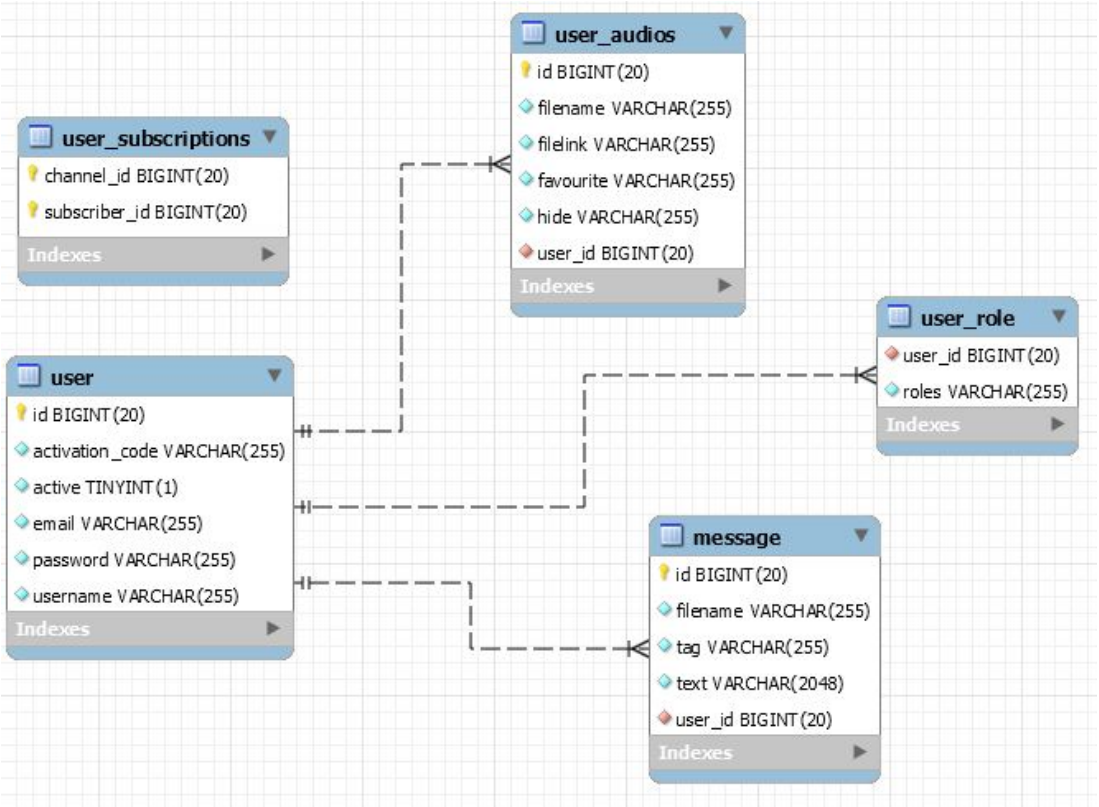


Рисунок 3.1 – ER-діаграма БД розроблюваної системи

Об’єкти типу «Користувачі» зберігаються у таблиці user, яка містить 6 полів, кожне з яких містить наступну інформацію про користуачів:

– id – ключове поле таблиці;
– activation_code – поле, яке містить ключ активації сторінки користувача;
– active – поле, в якому відображається наявність користувача в системі;
– email – поле, яке зберігає email користувача;
– password – поле, яке зберігає пароль користувача;
– username – ім’я зареєстрованого користувача.

Об’єкти типу «Ролі» зберігаються у таблиці user_roles, яка містить 6 полів, кожне з яких містить наступну інформацію про ролі користувачів:

– user_id – ключове поле таблиці;
– roles – роль користувача на сайті;

Об’єкти типу «Медіафайли» зберігаються у таблиці user_audios, яка містить 6 полів, кожне з яких містить наступну інформацію про медіафайли:

– id – ключове поле таблиці;
– file_name – назва медіафайлу, який завантажується користувачем;
– file_link – поле, у якому зберігається формат файлу;
– favourite – поле, яке відображає наявність медіафайлу до категорії «Улюблені»;
– hide – поле, яке відображає наявність медіафайлу до категорії «Приховані»;
– user_id – поле для ідентифікації профільної сторінки користувача.

Об’єкти типу «Повідомлення» зберігаються у таблиці message, яка містить 5 полів, кожне з яких містить наступну інформацію про повідомлення на «форумі»:

– id – ключове поле таблиці;
– filename – поле, де зберігається назва і формат файлу;

– tag – поле, яке відображає тег, прикріплений до надісланого медіафайлу, за допомогою якого користувачі сайту можуть здійснювати пошук повідомлень;

– text – поле, яке відображає текстову інформацію, прикріплену до надісланого медіафайлу;

– user_id – поле для ідентифікації користувача.

Об’єкти типу «Підписки» зберігаються у таблиці user_subscriptions, яка містить 2 поля, кожне з яких містить наступну інформацію про користувачів:

– channel_id – поле, у якому містяться дані про підписаних користувачів на певного користувача;

– subscriber_id – поле, у якому містяться дані про підписки певного користувача;

Ватро відзначити що ключовим об’эктом, через який здійснюється взаємодія в ER-діаграма, являється об’єкт типу «Користувачі», оскільки в даній дипломній роботі першочерговою задачею являється успішна комунікація між користувачами.

При створенні бази даних було використано СКБД MySQL та середовище розробки MySQL Workbench. Нижче наведений SQL-код створення бази даних:

```
create database audio;
create table audio.usraudios (
    id int8 not null,
    filename varchar(255),
    filelink varchar(255),
    favourite varchar(255),
    hide varchar(255),
    user_id int8,
    primary key (id)
);
drop table audio.usraudios;
select * from audio.usraudios;
create table audio.message (
```

```
        id int8 not null,
        filename varchar(255),
        tag varchar(255),
        text varchar(2048) not null,
        user_id int8,
        primary key (id)
    );
    select * from audio.message;

    create table audio.user_role (
        user_id int8 not null,
        roles varchar(255)
    );
    select * from audio.user_role;
    create table usr (
        id int8 not null,
        activation_code varchar(255),
        active boolean not null,
        email varchar(255),
        password varchar(255) not null,
        username varchar(255) not null,
        primary key (id)
    );
    select * from audio.usr;
    alter table message add constraint message_user_fk foreign
key (user_id) references usr (id);
    alter table user_role add constraint user_role_user_fk
foreign key (user_id) references usr (id);
    select * from audio.user_subscriptions;
    create table audio.user_subscriptions (
        channel_id int8 not null references usr,
        subscriber_id int8 not null references usr,
        primary key (channel_id, subscriber_id)
    )
```

При встановленні взаємодію програми і бази даних перш за все було передано інформацію бази даних в файл application.properties для її успішної взаємодії:

```
spring.datasource.url=jdbc:mysql://localhost:3306/audio?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=1111
spring.jpa.generate-ddl=false
```

Для прикладу представлення взаємодії візьмемо об’єкт предметної області «Користувачі» представлений за допомогою класу User, який зберігається в пакеті domain. В цьому класі описані основні поля та методи доступу до них:

```
// Оголошення закритих полів
private Long id;
private String username;
private String password;
private String password2;
private boolean active;
private String email;
private String activationCode;
// Оголошення методів доступу
// Метод доступу get використовується для повернення значення
public Long getId() {
      return id;
}
// Метод доступу set - для присвоєння нового значення
public void setId(Long id) {
      this.id = id;
}
```

Програмний Java-код даного класу наведений у додатку.

3.2. Проектування класів та архітектури програмної системи

Оскільки проект виконується за допомогою Spring framework, то сама структура роботи розроблювалась по всім принципам паттерну MVC.

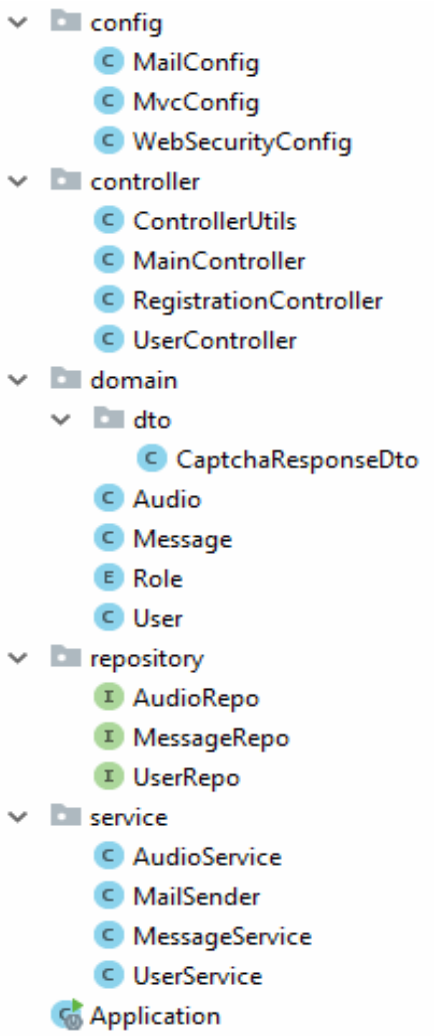


Рисунок 3.2 – Структура класів

В результаті аналізу розробки необхідного функціоналу було розроблено наступну ієрархію класів:

– Пакет config, в якій знаходяться наступні класи:

• Клас MailConfig – конфігурація зв’язку з електронною поштою, через яку будуть надсилатись активаційні коди щойно зареєстрованим користувачем;

• Клас MvcConfig – клас конфігурації для налаштування Spring MVC у додатку;

• Клас WebSecurityConfig – конфігурація безпеки.

– Пакет controller, в якій знаходяться наступні класи:

• Клас ControllerUtils – клас, який відповідає за виведення помилок у разі не пройденої валідації відповідних полів;

• Клас MainController – встановлення зв’язку між основним функціоналом дипломної роботи та web-інтерфейсом;

• Клас RegistrationController – встановлення зв’язку між розробленим функціоналом реєстрації та web-інтерфейсом;

• Клас UserController – встановлення зв’язку між розробленим функціоналом для взаємодії користувачів та web-інтерфейсом.

– Пакет domain, в якій знаходяться наступні класи:

• Клас CaptchaResponseDto – клас для реалізації «капчі», яка використовується для реєстрації на сайт;

• Клас User – сутність, яка реалізовує дані про користувачів;

• Клас Role – сутність, яка реалізовує дані про ролі користувачів;

• Клас Audio – сутність, яка реалізовує дані про аудіозаписи користувачів;

• Клас Message – сутність, яка реалізовує дані про повідомлення користувачів.

– Пакет repository, в якій знаходяться наступні класи:

• Клас UserRepo – інтерфейс для вказівки подальшого розроблюваного функціоналу, пов’язаного з користувачами;

• Клас AudioRepo – інтерфейс для вказівки подальшого розроблюваного функціоналу, пов’язаного з аудіозаписами;

• Клас MessageRepo – інтерфейс для вказівки подальшого розроблюваного функціоналу, пов’язаного з повідомленнями.

– Пакет service, в якій знаходяться наступні класи:

- Клас AudioService – клас, у якому реалізовуються функціонал web-додатку, який пов’язаний з аудіозаписами;
- Клас MailSender – клас, у якому реалізовуються функціонал web-додатку, який виконує надсилання повідомлень користувачу при його реєстрації на сайті(код активації);
- Клас MessageService – клас, у якому реалізовуються функціонал web-додатку, який пов’язаний з повідомленнями;
- Клас UserService – клас, у якому реалізовуються функціонал web-додатку, який пов’язаний з користувачами.

Для відображення та успішної реалізації та оброки виконаного функціоналу для користувачів було використано такий пагін як Freemarker.

Freemarker – це інструмент для генерації тексту на підставі шаблонів і вбудованих даних. Він простий у використанні, а також працює з будь-якими текстовими файлами.

Шаблон складається з використанням мови шаблонів (FreeMarker Template Language - FTL). За великим рахунком, це не є повноцінною мовою, швидше це розмітка, яка дозволяє модулю Freemarker додавати в шаблон необхідну інформацію.

Ця архітектура і є головною перевагою Freemarker. Використання такого підходу дозволяє розділити програмістів і дизайнерів, принаймні це повною мірою відноситься до побудови динамічних web-додатків.

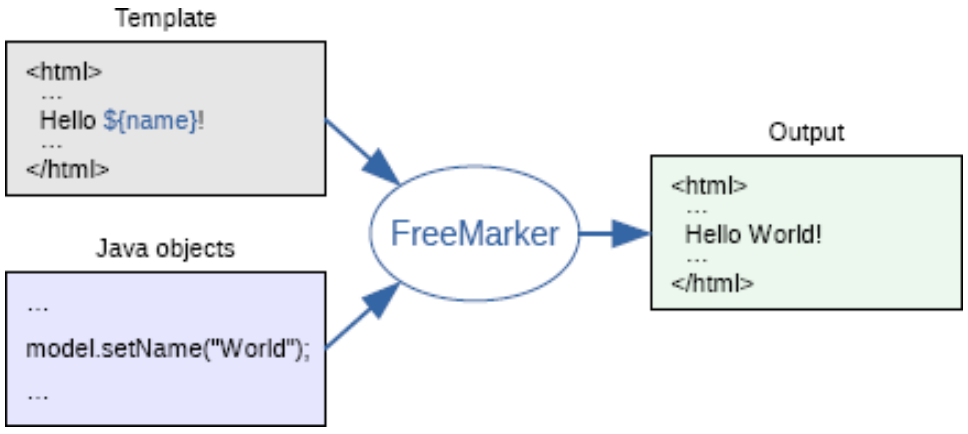


Рисунок 3.3 – Принцип роботи Freemarker

В результаті аналізу розробки «бекенду» було реалізовано наступну ієрархію FTL-файлів проекту (рис. 3.4):

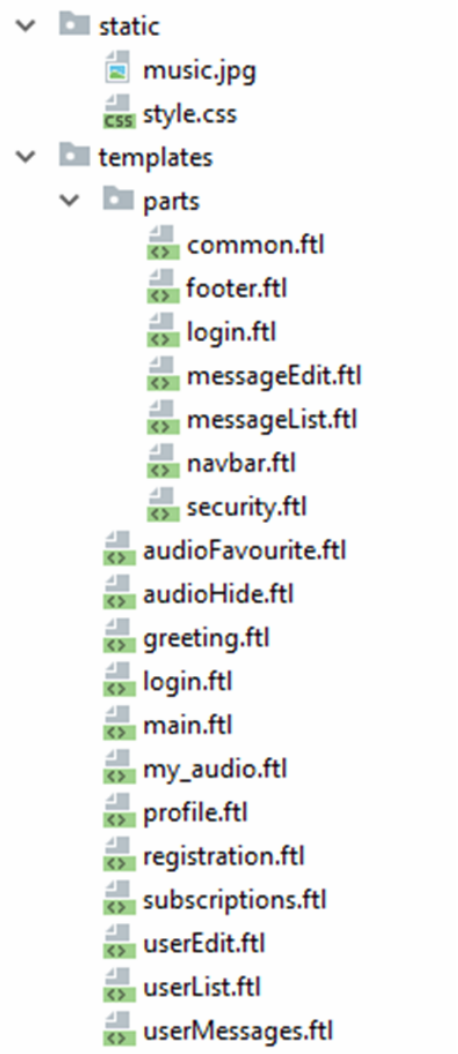


Рисунок 3.4 – Структура FTL-файлів

Папка template, в якій знаходяться наступні FTL-файли:

- audioFavourite – сторінка з аудіозаписами користувача, які відносяться до категорії «Улюблені»;
- audioHide – сторінка з аудіозаписами користувача, які відносяться до категорії «ховані»;
- greeting – головна сторінка сайту;
- login – сторінка з авторизацією користувача;
- main – сторінка з «чатом»;

- my_audio – сторінка з медіафайлами користувача;
- profile – сторінка для зміни паролю;
- registration – сторінка з реєстрацією користувача;
- subscriptions – сторінка з відображенням підписників певного користувача;
- userEdit – сторінка для редагування даних користувача адміністратором;
- userList – список усіх наявних користувачів в системі(доступна тільки для адміністраторів сайту);
- userMessages – сторінка з відображенням всіх повідомлень певного користувача з «чату».

– Папка parts(знаходиться в папці template), в якій FTL-файли, складова частина яких неодноразово повторюється в інших FTL-файли. В даній папці знаходяться:

- common – файл, в якому підключається bootstrap ;
- footer – «футер» сайту;
- login – поля, які використовуються для реєстрації і авторизації;
- messageEdit – фрагмент для редагування повідомлення;
- messageList – фрагмент для виведення списку всіх наявних повідомлень;
- navbar – головне меню сайту;
- security – відповідає за надання доступу до певних сторінок в залежності від ролі(головний адміністратор, адміністратор, зареєстрований користувач, незареєстрований користувач.

3.3. Розробка функціональності та інтерфейсу системи

В результаті виконання програмного продукту продемонстровано функціонал за наступними категоріями

– головна сторінка ;
– реєстрація та авторизація;
– зміна паролю зареєстрованого користувача;
– можливості адміністратора сайту;
– профільна сторінка користувача;
– сторінка з «чатом»;
– підписки на інших користувачів сайту.

Зовнішній вигляд головної сторінки сайту представлено на рисунку 3.5.

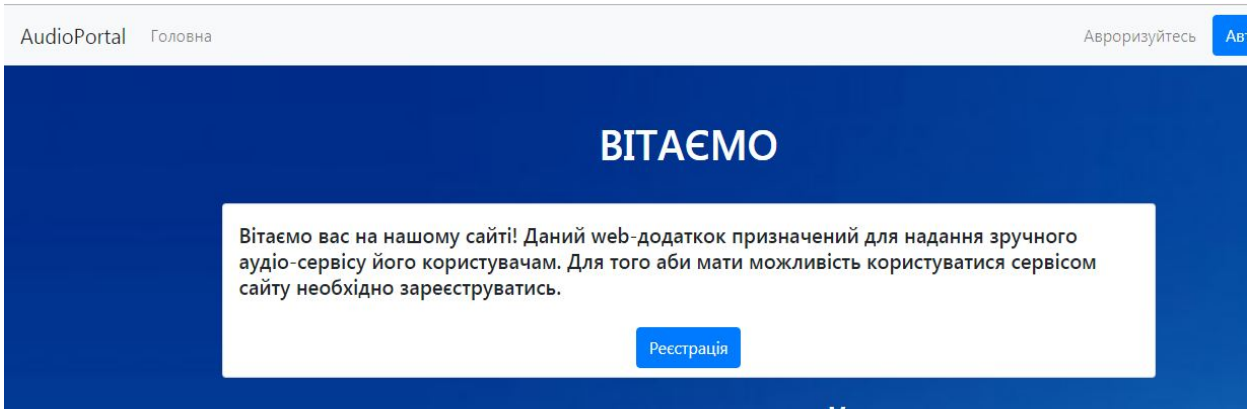


Рисунок 3.5 – Перша частина головної сторінки

Нижня частина головної сторінки представлено на рисунку 3.6.

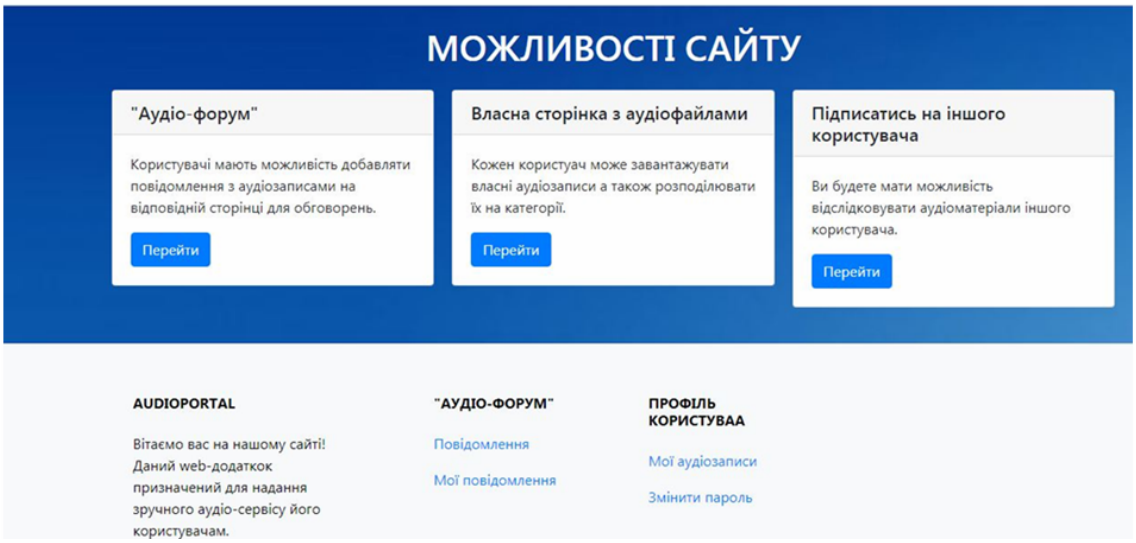


Рисунок 3.6 – Друга частина головної сторінки

Поля для реєстрації і авторизації успішно проходять валідацію. У випадку, коли користувач проходить реєстрацію на сайті, на задану електронну пошту приходить код активації, після відкриття користувач повідомлений про успішну реєстрацію. Сторінка авторизації представлена на рисунку 3.7.

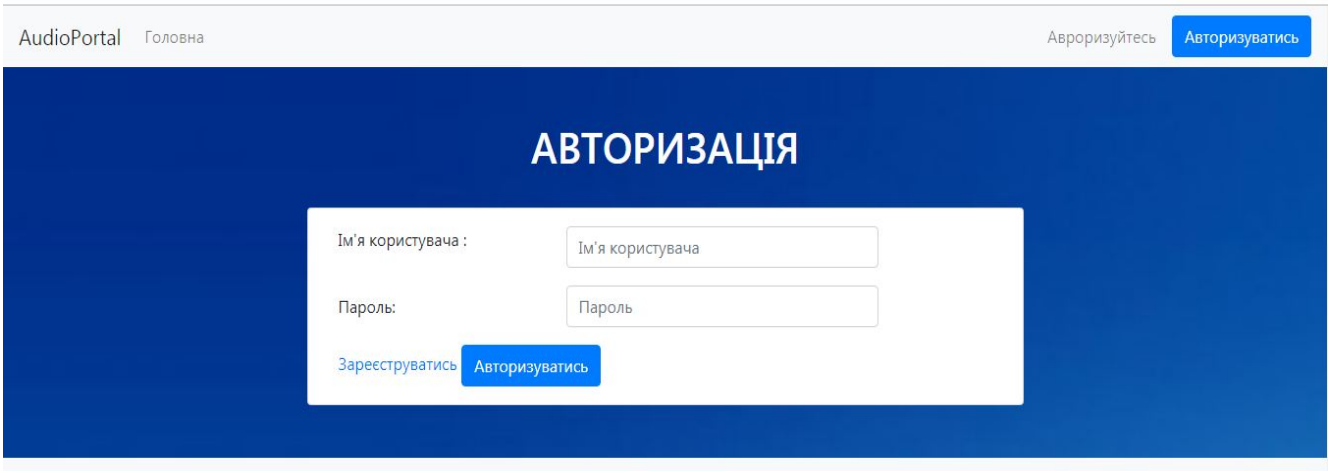


Рисунок 3.7 – Сторінка авторизації

Сторінка реєстрації представлена на рисунку 3.8.

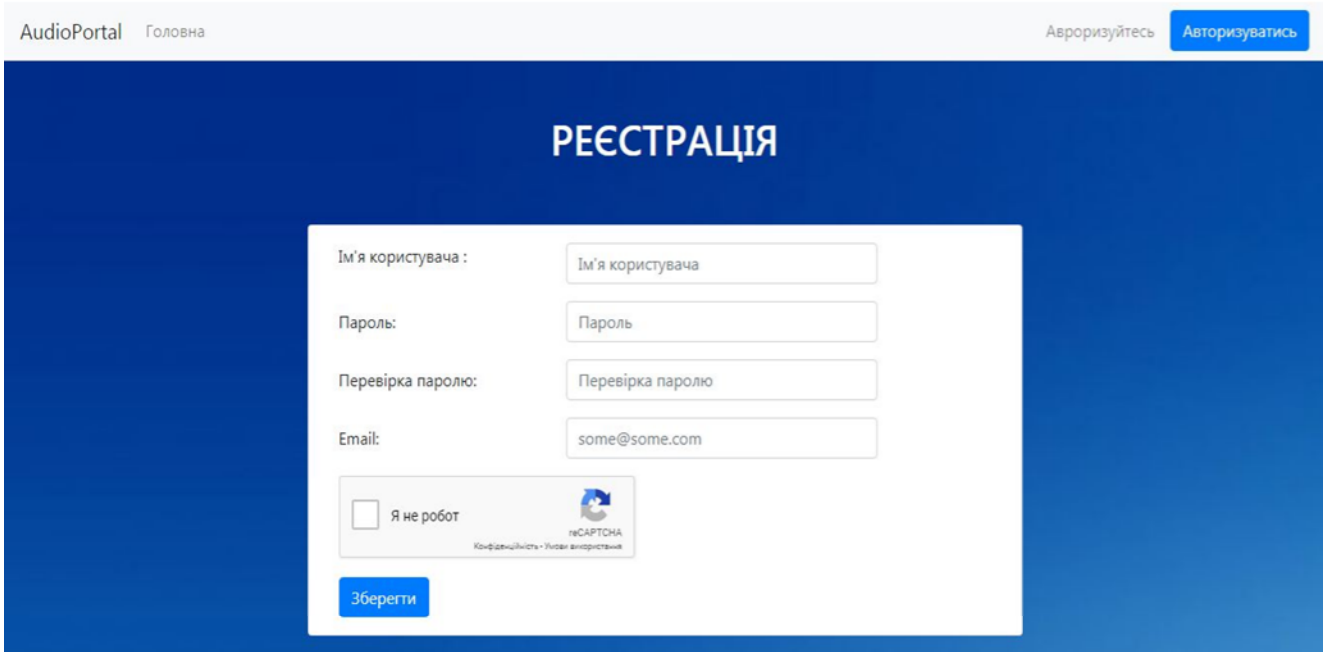


Рисунок 3.8 – Сторінка реєстрації

Перевірка полів на наявність введення даних представлена на рис. 3.9.

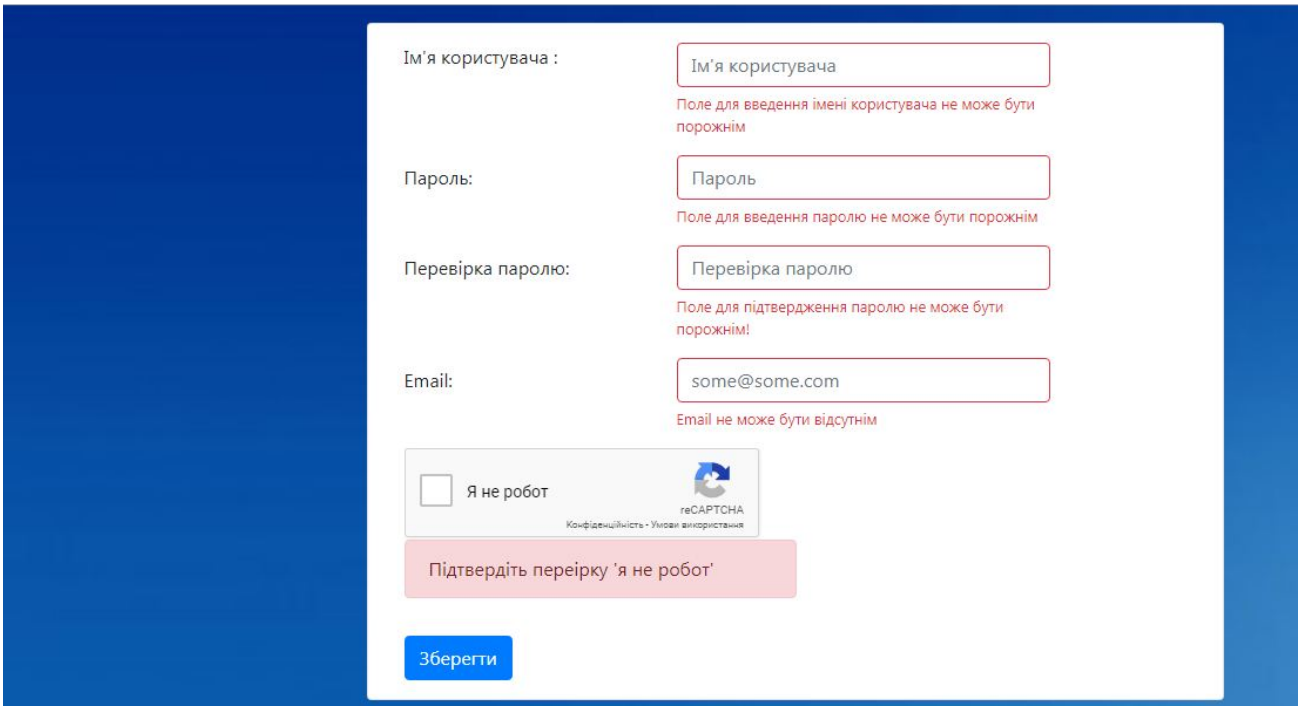


Рисунок 3.9 – Перевірка полів на наявність введення даних

Повідомлення на електронну пошту з активаційним кодом представлене на рисунку 3.10.

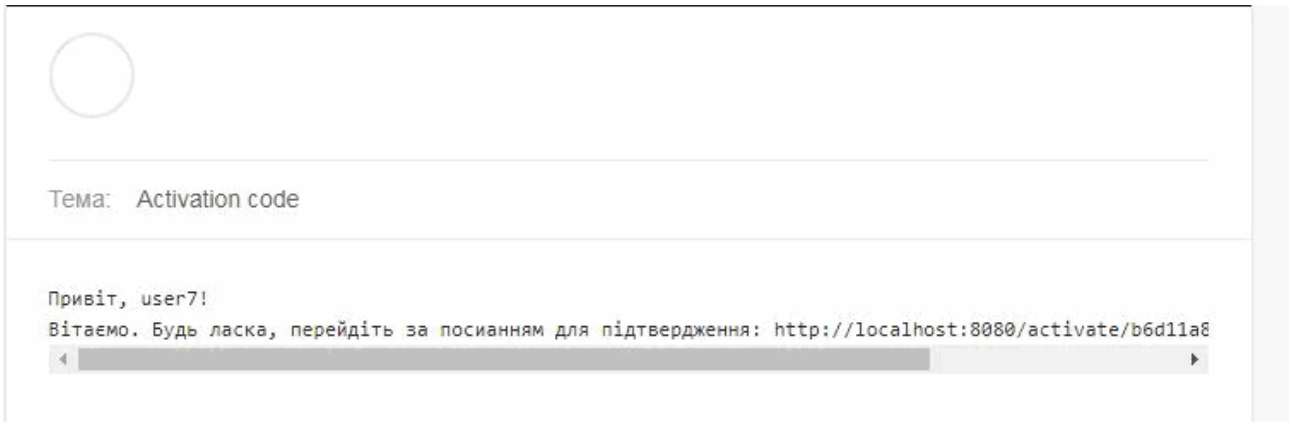


Рисунок 3.10 – Повідомлення на електронну пошту з активаційним кодом

Успішність реєстрації після переходу за посиланням з активаційним кодом представлена на рисунку 3.11.

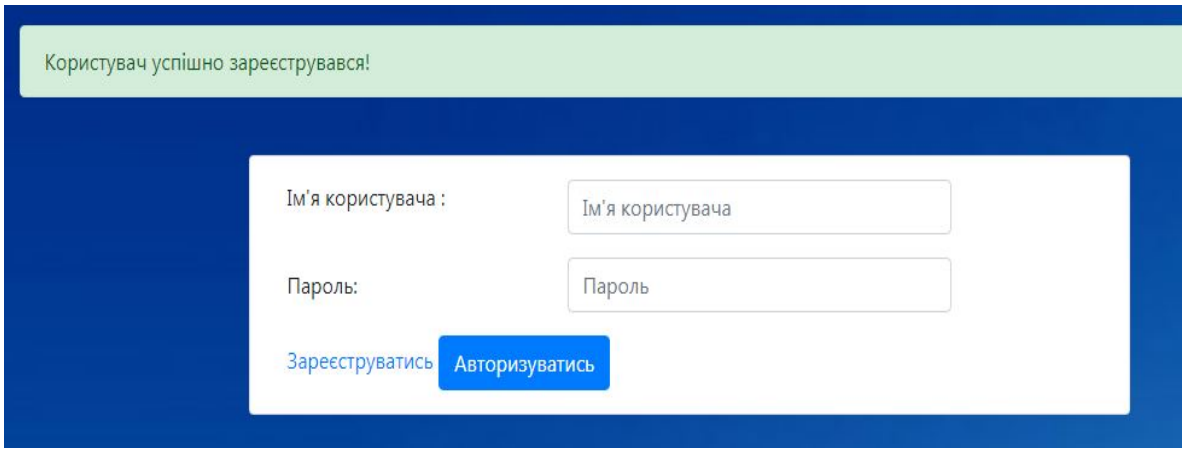


Рисунок 3.11 – Успішність реєстрації після переходу за посиланням з активаційним кодом

Користувач має можливість для зміни свого паролю у разі такої необхідності. Процес зміни паролю представлено на рисунку 3.12.

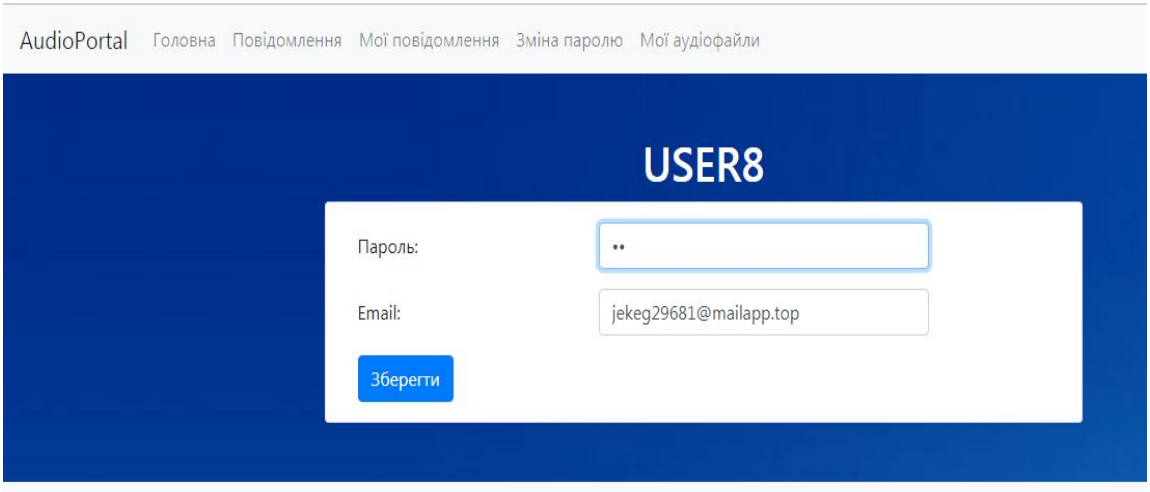


Рисунок 3.12 – Сторінка для зміни паролю

Наявність адміністратора є необхідною для кожного веб-додатку. Адміністратори мають можливість переглянути список всіх користувачів, а також можуть видаляти їхні повідомлення та надавати їм адміністраторські права. Можливість видалення повідомлень користувачів представлено на рисунку 3.13.

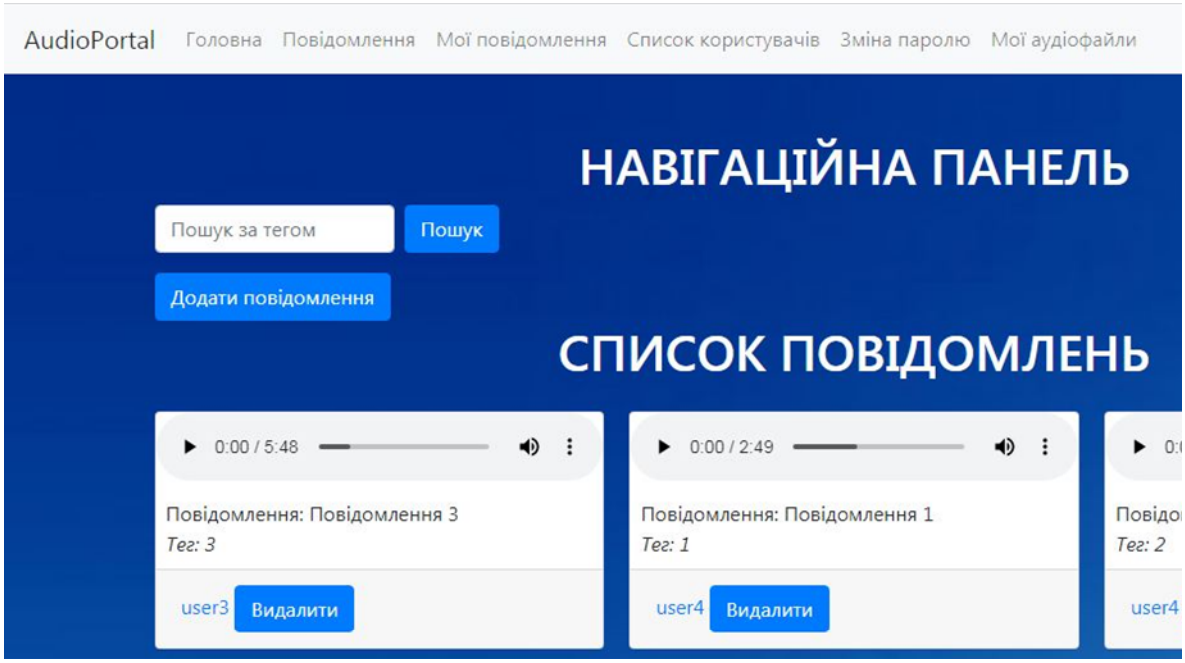


Рисунок 3.13 – Можливість видалення повідомлень користувачів

Список зареєстрованих користувачів представлено на рисунку 3.14.

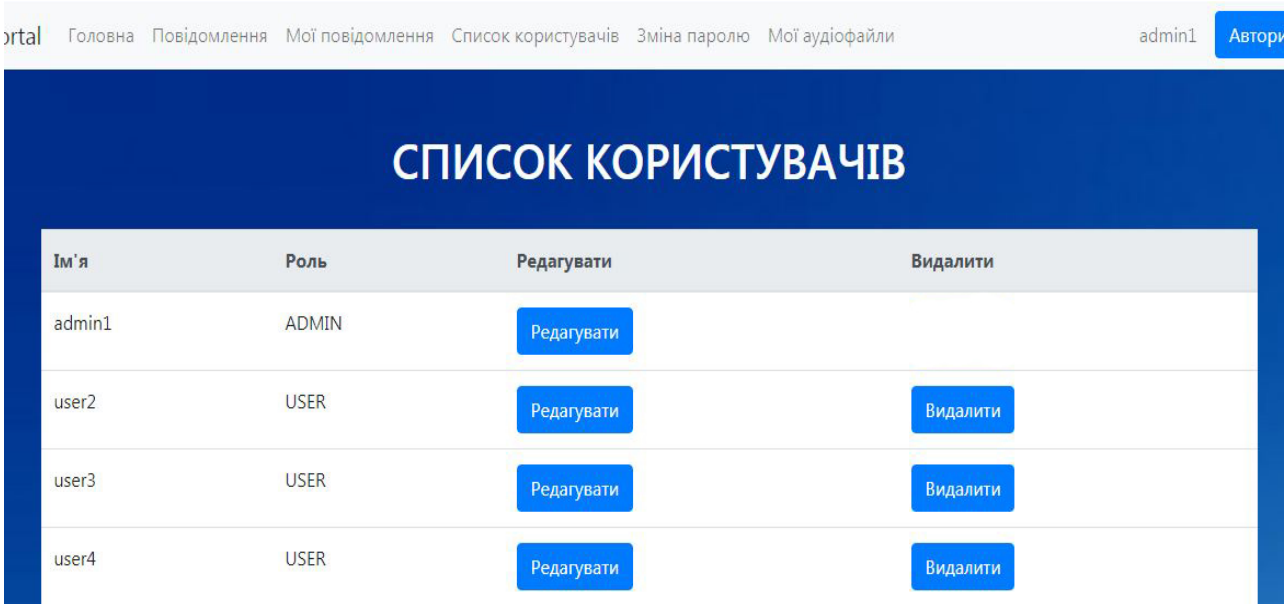


Рисунок 3.14 – Список зареєстрованих користувачів

Підтвердження дії про видалення користувача представлено на рисунку 3.15.

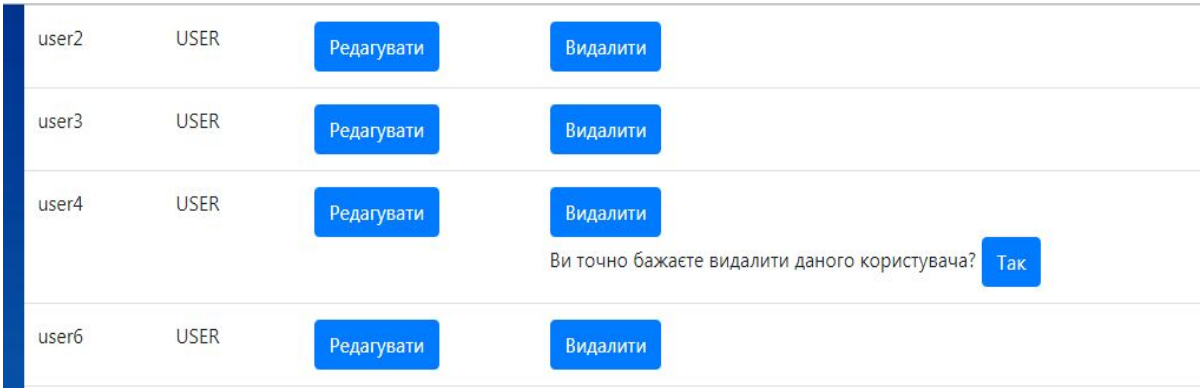


Рисунок 3.15 – Підтвердження дії про видалення користувача

Сторінка редагування даних користувача представлена на рисунку 3.16.

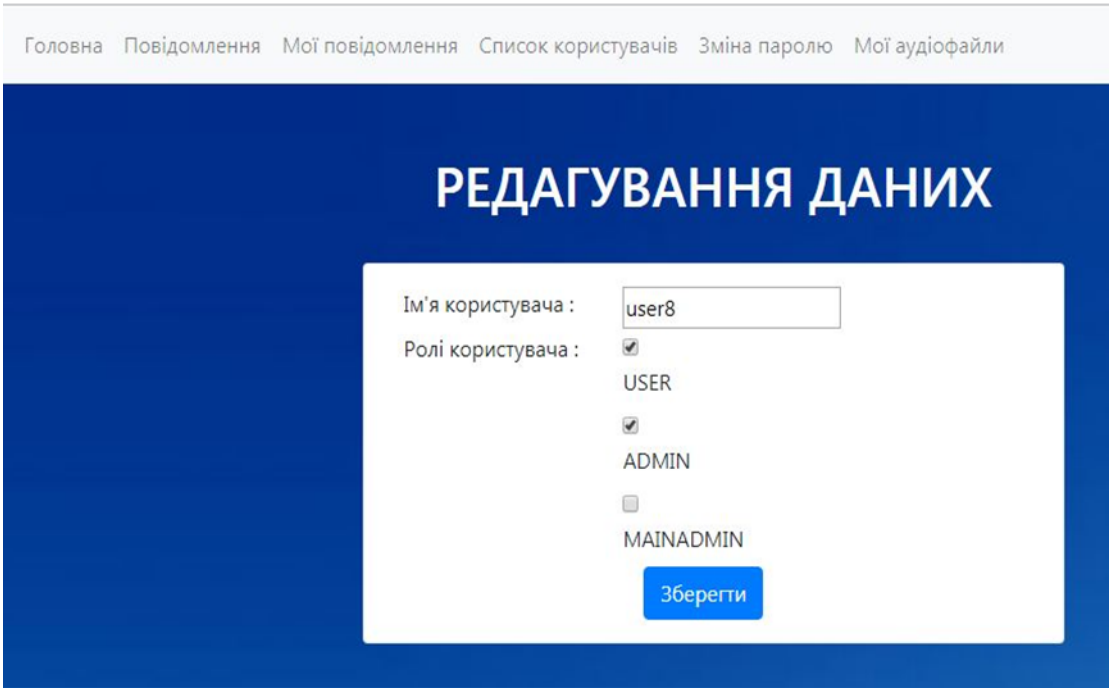


Рисунок 3.16 – Сторінка редагування даних користувача

Кожен користувач має можливість завантажити власні медіа контент на свою сторінку для прослуховування. Окрім цього користувачі мають можливість видалити дані медіафайли, редагувати їх, а також занести до відповідної категорії. Сторінка користувача системи представлена на рисунку 3.17.

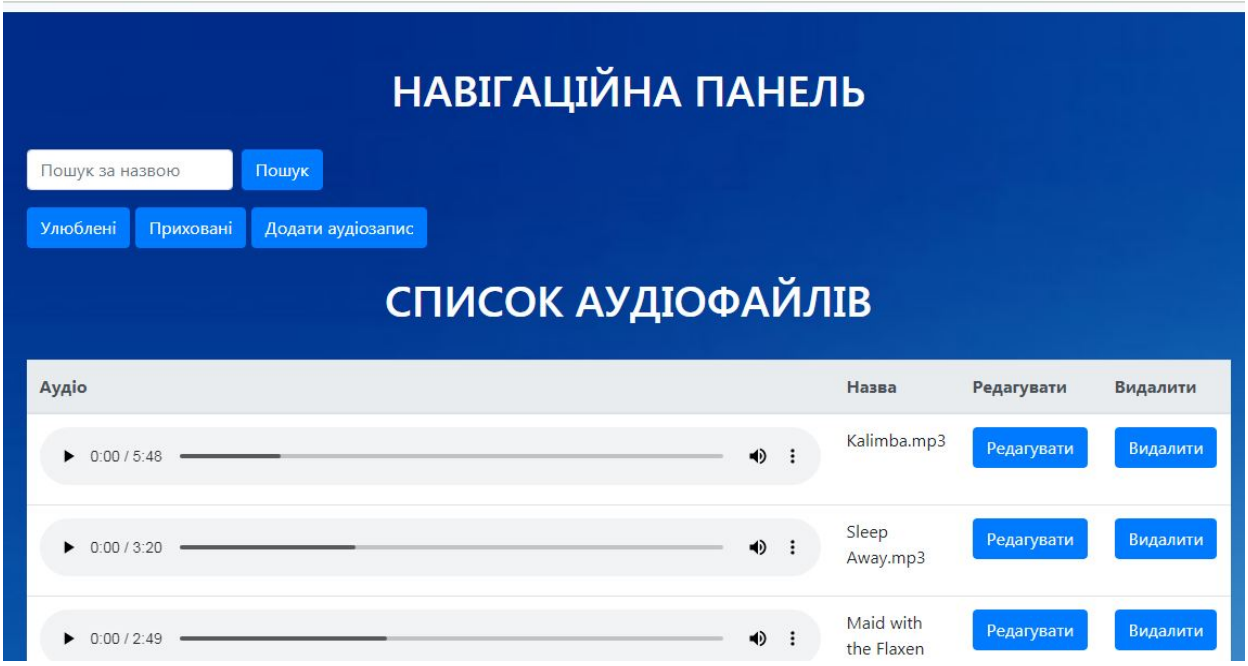


Рисунок 3.17 – Сторінка користувача сайту

Панель для додавання аудіозапису на сторінку представлена на рисунку 3.18.

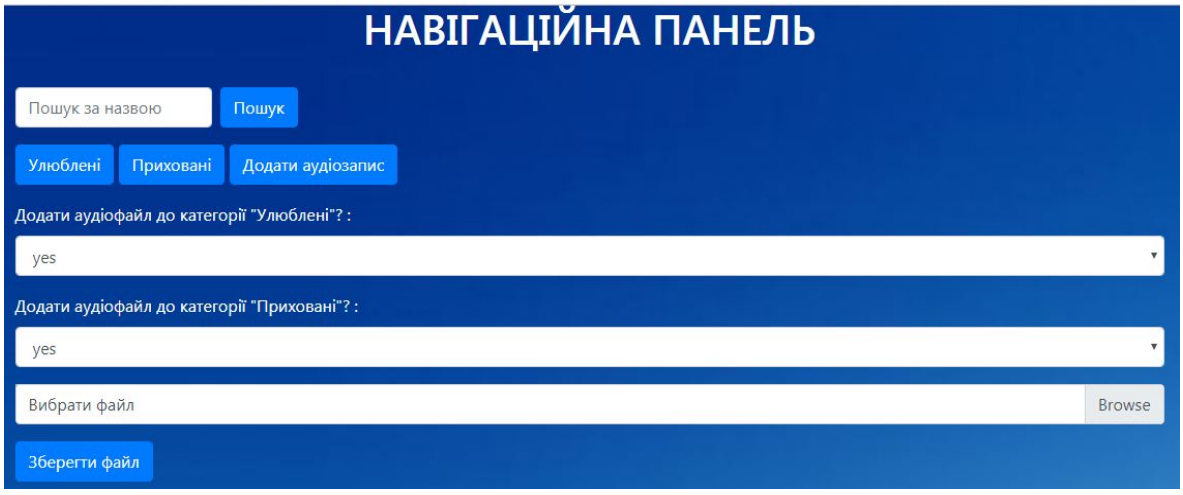


Рисунок 3.18 – Панель для додавання аудіозапису на сторінку

Приклад реалізованого пошуку медіафайлу за його назвою представлено на рисунку 3.19.

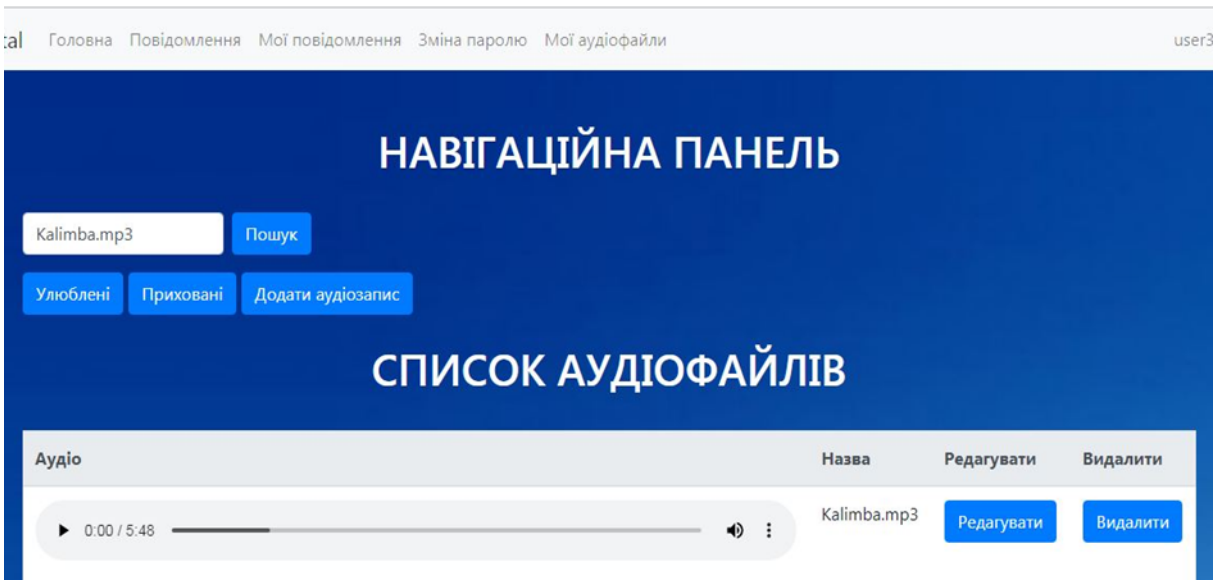


Рисунок 3.19 – Пошук медіафайлу за його назвою

Сторінка з улюбленими аудіо записами представлена на рисунку 3.20.

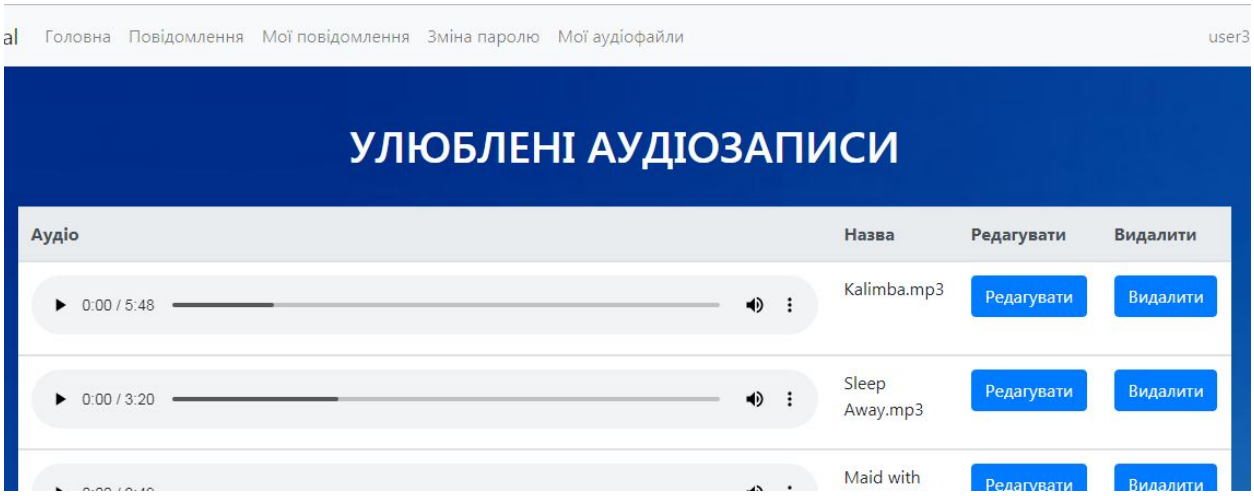


Рисунок 3.20 – Сторінка з улюбленими аудіозаписами

Користувачі також мають можливість ділитись власними повідомленнями на відведеній для цього сторінці. Користувач може здійснити пошук по тегам, які повідомляють про напрям медіафайлу, а також додавати власні повідомлення та редагувати їх. Сторінка з «чатом» представлена на рисунку 3.21.

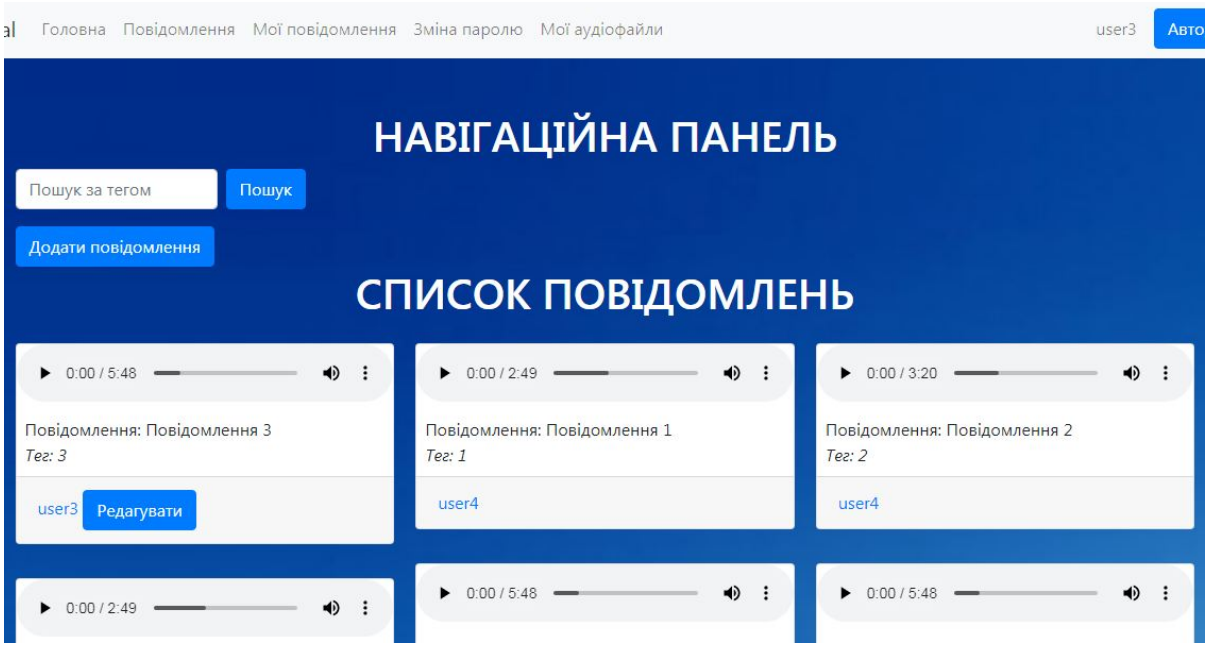


Рисунок 3.21 – Сторінка з «чатом»

Пошук повідомлень за тегом представлено на рисунку 3.22.

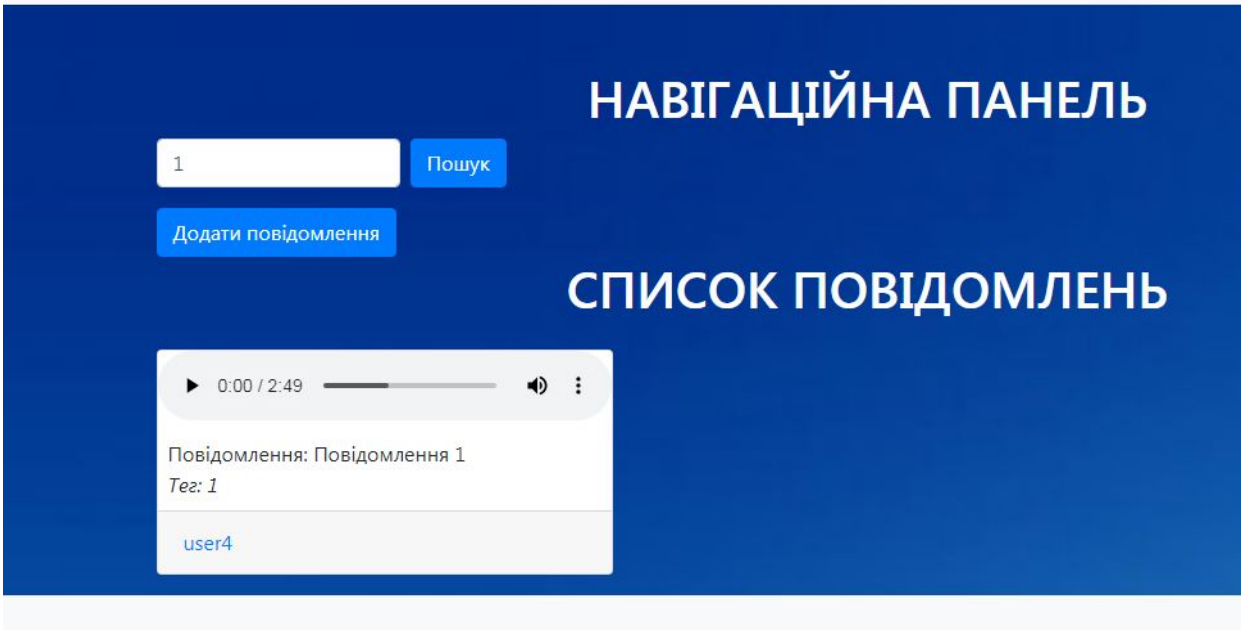


Рисунок 3.22 – Пошук повідомлень за тегом

Панель для додавання та редагування повідомлень представлена на рисунку 3.23.

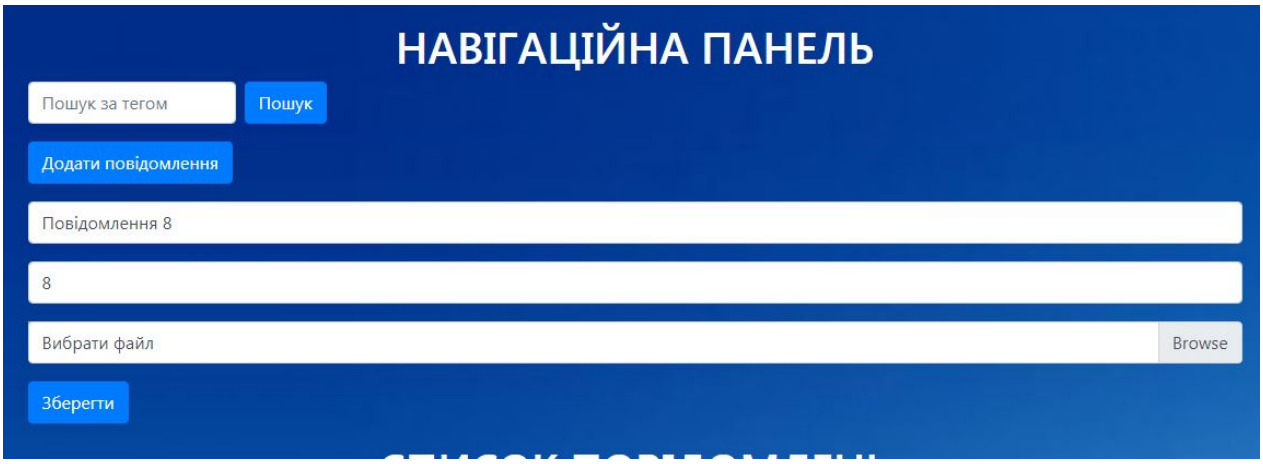


Рисунок 3.23 – Панель для додавання та редагування повідомлень

У випадку, коли користувачу сподобались повідомлення іншого користувача, або його збережені медіафайли на сторінці, можна здійснити підписку на даного користувача, за допомогою якої можна зручніше стежити за його оновленнями. Сторінка іншого користувача представлено на рисунку 3.24.

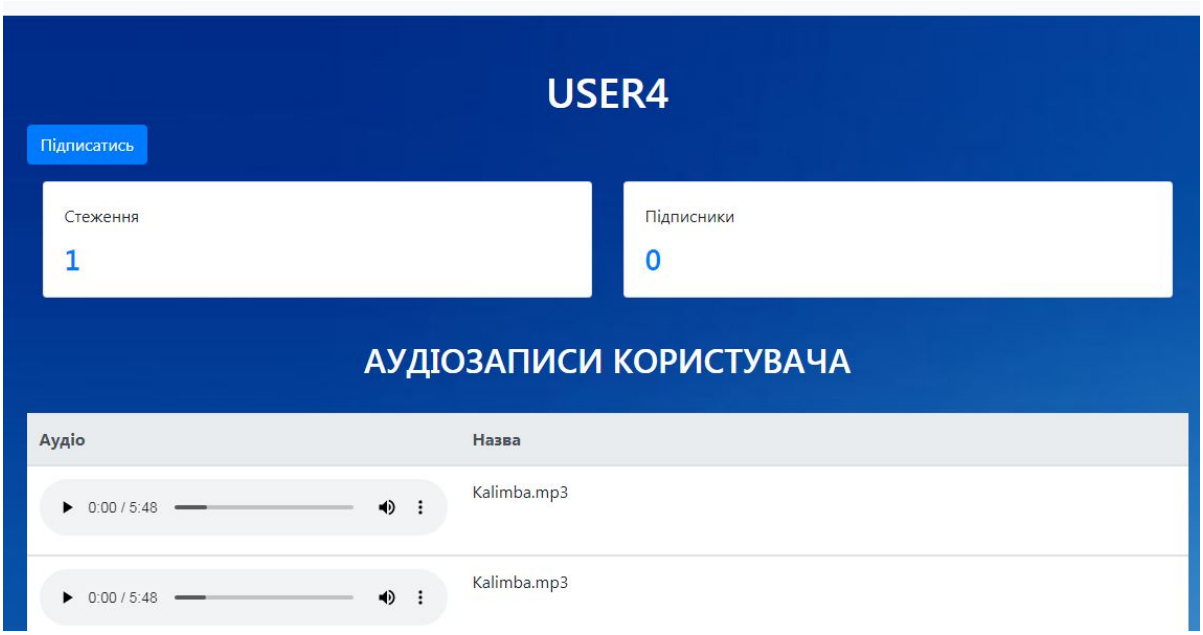


Рисунок 3.24 – Сторінка іншого користувача

Сторінка іншого користувача представлена на рисунку 3.25.

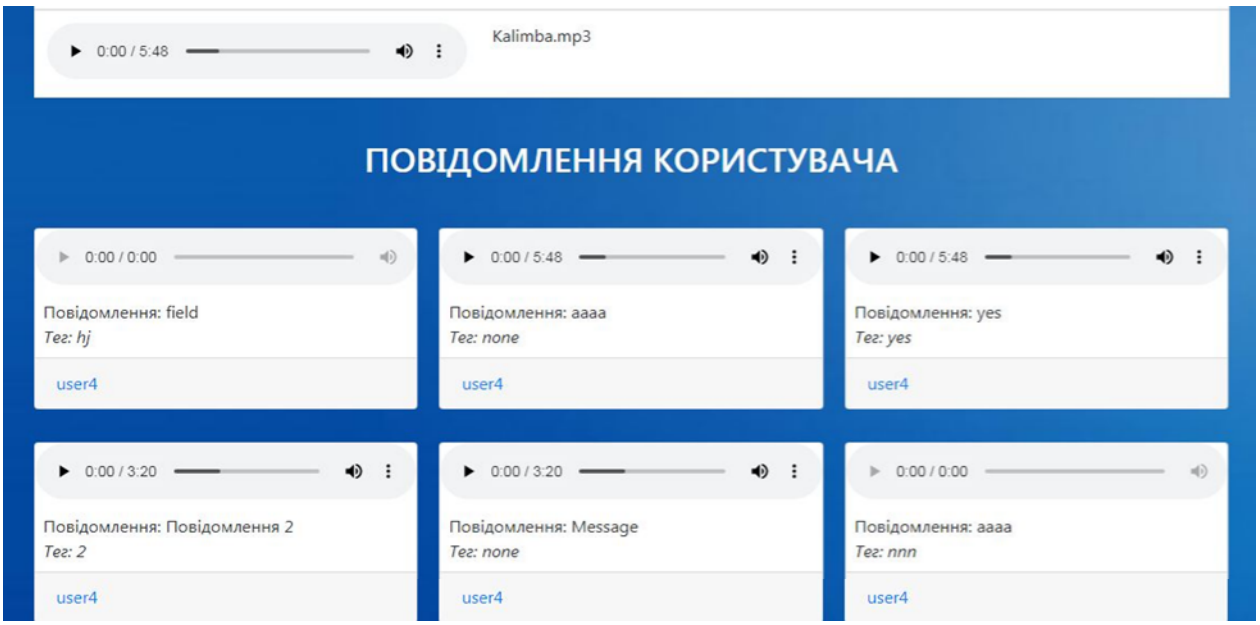


Рисунок 3.25 – Сторінка іншого користувача

Інформативні панелі з стеженнями і підписками представлені на рисунку 3.26.

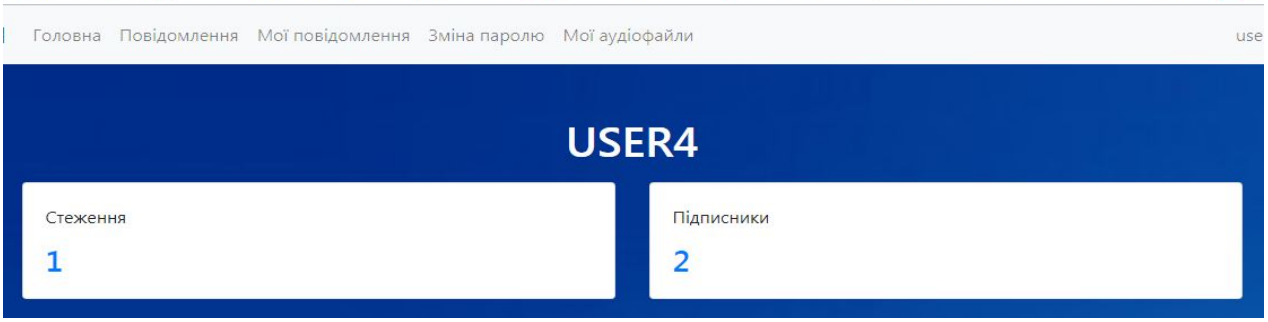


Рисунок 3.26 – Інформативні панелі з стеженнями і підписками

Перегляд підписників певного користувача представлено на рисунку 3.27.

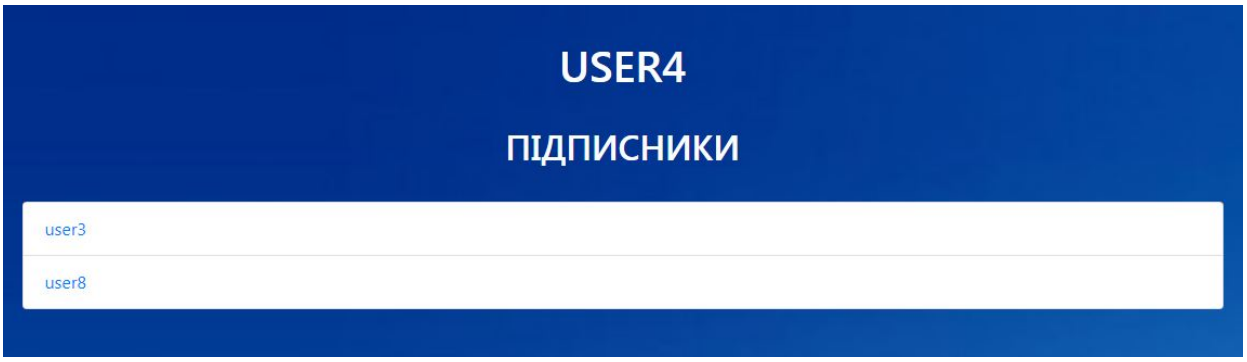


Рисунок 3.27 – Перегляд підписників певного користувача

Оформлення та відображення даних результатів виконання функціоналу було реалізовано за допомогою Bootstrap.

3.4. Виконання тестування програмного продукту

Тестування даної дипломної роботи здійснюється за допомогою ручного тестування. Ручне тестування являється одним з типів тестування, в якому тестовий випадок виконується вручну людиною. Процес ручного тестування проходив за допомогою сервісу Temp Mail (рисунок 3.28).

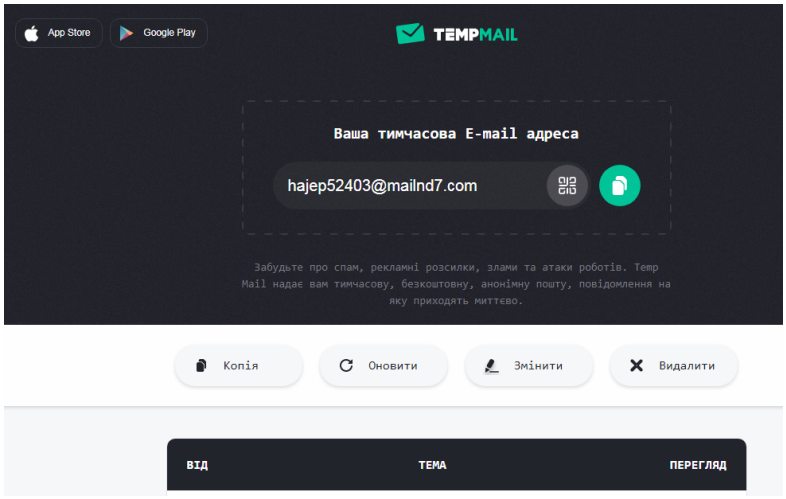


Рисунок 3.28 – Головна сторінка TempMail

Завдяки Temp Mail дуже просто отримати одноразову тимчасову електронну адресу, основною метою якої є боротьба зі спамом в основній адресі електронної пошти. В основному користувачі використовувати Temp Mail, коли вони підписуються на сайти, однак в даному випадку даний сервіс корисний для ручного тестування, оскільки розроблений програмний продукт місить декілька ролей з різним доступним функціоналом.

Все що потрібно зробити, щоб почати використовувати тимчасову адресу електронної пошти це відкрити додаток Temp Mail і вибрати ту, яка підходить. Їх не так просто запам’ятати, оскільки вони генеруються як випадкова асоціація

букв, але у користувача є можливість скопіювати адресу, щоб не вводити її вручну. Тест-кейси, які були виконані при ручному тестуванні, представлені на рис. 3.29.

ID	Опис тестового випадку	Передумови	Тестові дані
1	Перевірка можливості успішної реєстрації	Користувач знаходиться на головній сторінці сайту і натискає кнопку «Зареєструватись»	1. Ім'я користувача: User10 2. Пароль: user10 3. Перевірка паролю: user10 4. Email: hajep52403@gmail.com 5. Перевірка на «Я не робот»:
2	Перевірка можливості успішної авторизації	Користувач знаходиться на головній сторінці сайту і натискає кнопку «Авторизуватись», попередньо зареєструвавшись	1. Ім'я користувача: User10 2. Пароль: user10
3	Перевірка можливості завантаження медіафайлу на сторінку з «чатом»	Користувач знаходиться на сторінці «Повідомлення», попередньо зареєструвавшись	1. Повідомлення: Повідомлення1 2. Тег: Тег1 3. Вибрати файл: завантаження файлу

ID	Очікуваний результат	Фактичний результат	Статус
1	1. Поле «Ім'я користувача» заповнене 2. Поле «Пароль» заповнене і співпадає з полем 3. Поле «Перевірка паролю» заповнене і співбадає з полем 4. Поле «Email» заповнене та відповідає необхідному формату	1. Поле «Ім'я користувача» заповнене 2. Поле «Пароль» заповнене і співпадає з полем 3. Поле «Перевірка паролю» заповнене і співбадає з полем 4. Поле «Email» заповнене	Пройдений
2	1. Поле «Ім'я користувача» попередньо занесено в базу даних 2. Поле «Пароль» співбадає з попередньо збереженими даними	1. Поле «Ім'я користувача» попередньо занесено в базу даних 2. Поле «Пароль» співбадає з попередньо збереженими даними	Пройдений
3	1. Поле «Повідомлення» заповнене 2. Поле «Тег» заповнене 3. Поле «Вибрати файл» заповнене	1. Поле «Повідомлення» заповнене 2. Поле «Тег» заповнене 3. Поле «Вибрати файл» заповнене	Пройдений

Рисунок 3.29 – Пройдені тест-кейси

3.5. Висновки до розділу

У даному розділі було продемонстровано розроблену систему зберігання та поширення мультимедійного контенту програмний продукт, а саме:

– створення бази даних та її впровадження в середовище для розробки;

– представлення спроектованих внутрішньої і зовнішньої структури систему зберігання та поширення мультимедійного контенту;

– представлення виконаного функціоналу;

– представлення сервісу, за допомогою якого виконувалось ручне тестування.

ВИСНОВКИ

В результаті виконаного дипломної роботи було розроблено систему у вигляді веб-сервісу зберігання та поширення мультимедійного контенту. В роботі було розглянуто та проаналізовано реалізацію алгоритмічного та математичного забезпечення системи для перегляду та завантаження мультимедіа файлів. Результатом виконання роботи є система, що за допомогою розробленого алгоритму формує зручний сервіс для обміну медіафайлами з іншими користувачами сайту.

На першому етапі було розглянуто програмні продукти з подібними функціональними вимогами, сформовано аналіз вимог до проекту, а також проаналізовано засоби для його подальшої реалізації. Було встановлено перелік недоліків аналогічних додатків – як зручна можливість спілкування та комунікації між користувачами даних сервісів, що в результаті реалізувалось в розроблюваній системі.

На другому етапі було розроблено алгоритмічне і математичне забезпечення системи зберігання та поширення мультимедійного контенту, проведено аналіз існуючих методів, моделей і алгоритмів для реалізації програмного продукту.

На третьому етапі виконано опрацювання інформації з предметної області і виявлено основні можливості системи зберігання та поширення мультимедійного контенту. Завдяки визначеними прецедентам розроблено логічну структуру додатку, спроектовано ER-модель бази даних з відповідними таблицями та зв’язками між ними. На основі побудованої ER- діаграми створено базу даних мовою SQL у середовищі MySQL Workbench. Окрім цього було проведено написання власне самої програми мовою Java (за допомогою Spring Framework) та її тестування на основі тестових наборів даних, а також проведення рефакторингу програмного коду у середовищі InteliJ IDEA.

В результаті виконання функціоналу було реалізовано наступні сторінки, які знаходяться на головному меню веб-сервісу:

– Головна сторінка сервісу;

– Повідомлення – сторінка з «чатом»;

– Мої повідомлення – сторінка з відображенням повідомлень користувача з «чатом»;

– Список користувачів (відображається тільки для адміністраторів) – список усіх зареєстрованих користувачів в системі;

– Зміна паролю – сторінка для зміни паролю користувача;

– Мої медіафайли – список медіафайлів користувача;

– Авторизуйтесь – сторінка для авторизації користувача на сайті.

За час виконання дипломної роботи було програмно реалізовано поставлене завдання, а саме розроблено Java-додаток для роботи з базою даних, яка вміщує сукупність медіа файлів та мультимедійного контенту.

.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Про використання SugarSync – https://www.cloudstoragereviews.co/ cloud-storage-basics/how-sugarsync-works.html
2. SugarSync – https://www.sugarsync.com/
3. MediaFire – https://www.mediafire.com/
4. ZippyShare – https://www.zippyshare.com/
5. Про використання ZippyShare – https://hosting.review/file-storage/ zippyshare-review/
6. Mega – https://mega.dp.ua/
7. Про використання Mega – https://www.techradar.com/reviews/mega
8. Imgur – https://imgur.com/
9. Alexander Ngenzi Applying mathematical models in cloud computing: A survey, C. 3-11. – 2022,
10. serving-web-content. – URL: https://spring.io/guides/gs/serving-web-content/;
11. https://spring.io/guides/gs/accessing-data-mysql/;
12. https://docs.spring.io/spring-data/jpa/docs/1.5.0.RELEASE/reference/ html/ jpa.repositories.html#jpa.query-methods.query-creation;
13. https://spring.io/guides/gs/securing-web/;
14. Freemarker. – URL: https://freemarker.apache.org/docs/ ref_directive_ list.html;
15. Freemarker. – URL: https://freemarker.apache.org/docs/dgui_misc_ userdefdir.html;
16. Freemarker. – URL: https://freemarker.apache.org/docs/dgui_misc_ namespace.html;
17. https://spring.io/guides/gs/uploading-files/;
18. https://getbootstrap.com/docs/4.1/getting-started/introduction/;
19. https://www.google.com/recaptcha;
20. https://support.google.com/mail/answer/7126229?hl=ru;

21. Bootstrap. – URL: https://getbootstrap.com/docs/4.1/layout/grid/;

22. Bootstrap. – URL: https://getbootstrap.com/docs/4.1/components/navbar/;

23. Bootstrap. – URL: https://getbootstrap.com/docs/4.1/components/buttons/;

24. Bootstrap. – URL: https://getbootstrap.com/docs/4.1/components/card/;

25. Bootstrap. – URL: https://getbootstrap.com/docs/4.1/utilities/spacing/;

26. Bootstrap. – URL: https://getbootstrap.com/docs/4.1/components/list-group/

ДОДАТКИ

ДОДАТОК А

Лістингу коду деяких класів

Клас Role

```
package hello.domain;

import org.springframework.security.core.GrantedAuthority;

public enum Role implements GrantedAuthority{
    USER, ADMIN, MAINADMIN;

        @Override
        public String getAuthority() {
            return name();
        }

}
```

Клас User

```
package hello.domain;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import javax.persistence.*;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import java.util.Collection;
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

@Entity
@Table(name = "usr")
public class User implements UserDetails{
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @NotBlank(message = "Поле для введення імені користувача не
може бути порожнім")
    private String username;
    @NotBlank(message = "Поле для введення паролю не може бути
порожнім")
    private String password;
    @Transient  //@NotBlank(message = "Password confirmation
cannot be empty")
    private String password2;
    private boolean active;
```

```
    @Email(message = "Email введений некоректно")
    @NotBlank(message = "Email не може бути відсутнім")
    private String email;
    private String activationCode;
    @ElementCollection(targetClass = Role.class, fetch =
FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns =
@JoinColumn(name = "user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;

    @OneToMany(mappedBy = "author", cascade = CascadeType.ALL,
fetch = FetchType.LAZY)
    private Set<Message> messages;

    @OneToMany(mappedBy = "audioAuthor", cascade =
CascadeType.ALL, fetch = FetchType.LAZY)
    private Set<Audio> audios;
    @ManyToMany
    @JoinTable(
            name = "user_subscriptions",
            joinColumns = { @JoinColumn(name = "channel_id") },
            inverseJoinColumns = { @JoinColumn(name =
"subscriber_id") }
    )
    private Set<User> subscribers = new HashSet<>();

    @ManyToMany
    @JoinTable(
            name = "user_subscriptions",
            joinColumns = { @JoinColumn(name = "subscriber_id") },
            inverseJoinColumns = { @JoinColumn(name =
"channel_id") }
    )
    private Set<User> subscriptions = new HashSet<>();

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        User user = (User) o;
        return Objects.equals(id, user.id);
    }

    @Override
    public int hashCode() {

        return Objects.hash(id);
    }

    public boolean isAdmin() {
        return roles.contains(Role.ADMIN);
```

```
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getUsername() {
    return username;
}

public String getPassword() {
    return password;
}
public void setPassword(String password) {
    this.password = password;
}

public boolean isActive() {
    return active;
}

public void setActive(boolean active) {
    this.active = active;
}

public Set<Role> getRoles() {
    return roles;
}

public void setRoles(Set<Role> roles) {
    this.roles = roles;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getActivationCode() {
    return activationCode;
}

public void setActivationCode(String activationCode) {
    this.activationCode = activationCode;}
```

```
public String getPassword2() {
    return password2;
}
public void setPassword2(String password2) {
    this.password2 = password2;
}
public Set<Message> getMessages() {
    return messages;
}
public void setMessages(Set<Message> messages) {
    this.messages = messages;
}

public Set<User> getSubscribers() {
    return subscribers;
}
public void setSubscribers(Set<User> subscribers) {
    this.subscribers = subscribers;
}

public Set<User> getSubscriptions() {
    return subscriptions;
}
public void setSubscriptions(Set<User> subscriptions) {
    this.subscriptions = subscriptions;
}
public Set<Audio> getAudios() {
    return audios;
}
public void setAudios(Set<Audio> audios) {
    this.audios = audios;
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return isActive();
}
```

```
    public void setUsername(String username) {
        this.username = username;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities()
{
        return getRoles();
    }
}
```

Клас Audio

```
package hello.domain;
import javax.persistence.*;
@Entity
@Table(name = "usraudios")
public class Audio {
    @Id
    @GeneratedValue(strategy= GenerationType.AUTO)
    private Long id;
    private String filename;
    private String filelink;
    private String favourite;
    private String hide;
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "user_id")
    private User audioAuthor;
    public Audio(){
    }
    public Audio(/*String filename, boolean favourite, boolean
hide,*/String favourite, String hide, User user) {
        this.audioAuthor = user;
        //this.filename = filename;
        this.favourite= favourite;
        this.hide= hide;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getFilename() {
        return filename;
    }
    public void setFilename(String filename) {
        this.filename = filename;
    }
    public String getFilelink() {
        return filelink;
```

```
    }
    public void setFilelink(String filelink) {
        this.filelink = filelink;
    }
    public String getFavourite() {
        return favourite;
    }
    public void setFavourite(String favourite) {
        this.favourite = favourite;}

    public String getHide() {
        return hide;
    }
    public void setHide(String hide) {
        this.hide = hide;
    }

    public User getAudioAuthor() {
        return audioAuthor;
    }

    public void setAudioAuthor(User audioAuthor) {
        this.audioAuthor = audioAuthor;
    }

}
```

Клас CaptchaResponseDto

```
package hello.domain.dto;

import com.fasterxml.jackson.annotation.JsonAlias;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import java.util.Set;

@JsonIgnoreProperties(ignoreUnknown = true)
public class CaptchaResponseDto {
    private boolean success;
    @JsonAlias("error-codes")
    private Set<String> errorCodes;

    public boolean isSuccess() {
        return success;
    }

    public void setSuccess(boolean success) {
        this.success = success;
    }

    public Set<String> getErrorCodes() {
        return errorCodes;
    }
```

```
    public void setErrorCodes(Set<String> errorCodes) {
        this.errorCodes = errorCodes;
    }
}
```

Клас UserService

```
package hello.service;
import hello.domain.Role;
import hello.domain.User;
import hello.repository.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundExce
ption;
import
org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import java.util.*;
import java.util.stream.Collectors;
@Service
public class UserService implements UserDetailsService{
    @Autowired
    private UserRepo userRepo;
    @Autowired
    private MailSender mailSender;
    @Autowired
    private PasswordEncoder passwordEncoder;
    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userRepo.findByUsername(username);
        if (user == null) {
            throw new UsernameNotFoundException("User not found");
        }
        return user;
    }
    public boolean addUser(User user) {
        User userFromDb =
userRepo.findByUsername(user.getUsername());

        if (userFromDb != null) {
            return false;
        }
        user.setActive(true);
        user.setRoles(Collections.singleton(Role.USER));
```

```
        user.setActivationCode(UUID.randomUUID().toString());

user.setPassword(passwordEncoder.encode(user.getPassword()));
        userRepo.save(user);
        sendMessage(user);
        return true;
    }
    private void sendMessage(User user) {
        if (!StringUtils.isEmpty(user.getEmail())) {
            String message = String.format(
                    "Привіт, %s! \n" +
                            "Вітаємо. Будь ласка, перейдіть за
посиланням для підтвердження: http://localhost:8080/activate/%s",
                    user.getUsername(),
                    user.getActivationCode()
            );

            mailSender.send(user.getEmail(), "Activation code",
message);
        }
    }
    public boolean activateUser(String code) {
        User user = userRepo.findByActivationCode(code);

        if (user == null) {
            return false;
        }
        user.setActivationCode(null);
        userRepo.save(user);
        return true;
    }
    public List<User> findAll() {
        return userRepo.findAll();
    }
    public void saveUser(User user, String username, Map<String,
String> form) {
        user.setUsername(username);
        Set<String> roles = Arrays.stream(Role.values())
                .map(Role::name)
                .collect(Collectors.toSet());
        user.getRoles().clear();
        for (String key : form.keySet()) {
            if (roles.contains(key)) {
                user.getRoles().add(Role.valueOf(key));
            }
        }
        userRepo.save(user);
    }
    public void updateProfile(User user, String password, String
email) {
        String userEmail = user.getEmail();
        boolean isEmailChanged = (email != null &&
```

```
!email.equals(userEmail)) ||
                (userEmail != null && !userEmail.equals(email));
        if (isEmailChanged) { user.setEmail(email);

            if (!StringUtils.isEmpty(email)) {

user.setActivationCode(UUID.randomUUID().toString());
            }
        }
        if (!StringUtils.isEmpty(password)) {
            user.setPassword(passwordEncoder.encode(password));
        }
        userRepo.save(user);
        if (isEmailChanged) {
            sendMessage(user);
        }
    }
    public void subscribe(User currentUser, User user) {
        user.getSubscribers().add(currentUser);
        userRepo.save(user);
    }
    public void unsubscribe(User currentUser, User user) {
        user.getSubscribers().remove(currentUser);
        userRepo.save(user);
    }
    public void deleteById(Long id) throws Exception {
        if (!existsById(id)) {
            throw new Exception("Cannot find Note with id: " +
id);
        } else {
            userRepo.deleteById(id);
        }
    }
    private boolean existsById(Long id) {
        return userRepo.existsById(id);
    }
}
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “Реалізація програмного забезпечення системи зберігання та поширення мультимедійного контенту”

Обсяг пояснювальної записки: 75 аркушів

Кількість таблиць 6

Кількість рисунків 50

Дата завершення – 17 червня 2024 р.