

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ
СІКОРСЬКОГО»**

**НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу**

До захисту допущено
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___» _____ 2023 р.

Дипломна робота

**на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Інформаційна система обліку медіаконтенту на основі баз
даних»**

Виконав:

студент IV курсу, групи КА-91
Чеманов Олександр Борисович _____

Керівник:

зав. каф. СП НН ІПСА, д.т.н., професор
Мухін Вадим Євгенійович _____

Консультант з економічного розділу:

доцент каф. ЕК ФММ, к.е.н.
Рощина Надія Василівна _____

Консультант з нормоконтролю:

доцент каф. ММСА НН ІПСА, к.ф.-м.н.
Статкевич Віталій Михайлович _____

Рецензент:

зав. каф. ІТ ДУІТ, д.т.н., професор
Завгородній Валерій Вікторович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студент _____

Київ – 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
ІН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___» _____ 2023 р.

ЗАВДАННЯ

на дипломну роботу студенту

Чеманову Олександрю Борисовичу

1. Тема роботи: «Інформаційна система обліку медіаконтенту на основі баз даних», керівник роботи професор, д.т.н. Мухін В.Є., затверджені наказом по університету від «30» травня 2023 р. № 2065-с.
2. Термін подання студентом роботи 12.06.2023 р.
3. Вихідні дані до роботи: зразки існуючого медіаконтенту (відео, зображення) у формі окремих файлів та інформація пов'язана з ними.
4. Зміст роботи: 1. Огляд та аналіз існуючих баз даних в контексті створення інформаційних систем; 2. Теоретичні відомості інформаційної системи веб сервісу відеохостингу; 3. Проектування та розробка застосунку для перегляду відеоконтенту; 4. Функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу: схеми архітектур серверних додатків, схеми та діаграми компонентів розробленого додатку, зображення з прикладами роботи розробленої системи та існуючих систем, схематичне представлення баз даних, презентація.
6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н.В., доцент ЕК ФММ		

7. Дата видачі завдання: 07.02.2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання та затвердження теми БДР	17.04.2023	виконано
2	Збір інформації та пошук літератури	24.04.2023	виконано
3	Аналіз існуючих підходів та рішень	30.04.2023	виконано
4	Проектування інформаційної системи	05.05.2023	виконано
5	Підготовка даних для заповнення БД системи	07.05.2023	виконано
6	Розробка інформаційної системи	15.05.2023	виконано
7	Тестування розробленого продукту	17.05.2023	виконано
8	Оформлення дипломної роботи та презентації	29.05.2023	виконано

Студент

Олександр ЧЕМАНОВ

Керівник

Вадим МУХІН

РЕФЕРАТ

Дипломна робота містить 143 с., 36 рис., 10 табл., 2 додатки, 24 джерела.

ІНФОРМАЦІЙНА СИСТЕМА, ОБЛІК МЕДІАКОНТЕНТУ, БАЗИ ДАНИХ, МЕДІАФАЙЛИ, УПРАВЛІННЯ МЕДІАКОНТЕНТОМ, КАТАЛОГІЗАЦІЯ, ПОШУК МЕДІАКОНТЕНТУ, ДОСТУП ДО МЕДІАФАЙЛІВ, ХРАНІННЯ МЕДІАКОНТЕНТУ, РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ, ОБ'ЄКТНО-ОРІЄНТОВАНА МОДЕЛЬ ДАНИХ, SQLITE, MYSQL, POSTGRESQL.

Об'єктом дослідження виступає спосіб розповсюдження відео в мережі Інтернет.

Предмет дослідження – програмне забезпечення для завантаження та перегляду відеофайлів.

Метою проекту є розробка програмного забезпечення для розміщення відео, яке дозволяє користувачам завантажувати, переглядати та взаємодіяти з відеофайлами.

Було розроблено сервіс для перегляду відеоконтенту. Розроблено алгоритм рекомендацій для кожного користувача заснований на історії їх переглядів. Таким чином, це дозволяє користувачам переглядати фільми які він захоче або фільми з рекомендацій для нього. Також був проведений експериментальні дослідження системи обробки запитів гетерогенної розподіленої бази даних, метою якого було визначення вагових коефіцієнтів. Основною мовою програмування, що використовувалася при написанні сервісу для перегляду відеоконтенту, була Java.

ABSTRACT

The thesis contains 143 pages, 36 figures, 10 tables, 2 appendices, 24 sources.

INFORMATION SYSTEM, MEDIA CONTENT ACCOUNTING, DATA BASES, MEDIA FILES, MEDIA CONTENT MANAGEMENT, CATALOGING, MEDIA CONTENT SEARCH, MEDIA FILES ACCESS, MEDIA CONTENT STORAGE, RELATIONAL DATA MODEL, OBJECT ORIENT NETWORKED DATA MODEL, SQLITE, MYSQL, POSTGRESQL.

The object of the study is the method of video distribution on the Internet.

The subject of research is software for downloading and viewing video files.

The goal of the project is to develop video hosting software that allows users to upload, view, and interact with video files.

A service for viewing video content was developed. A recommendation algorithm has been developed for each user based on their browsing history. Thus, it allows users to watch movies that he wants or movies with recommendations for him. An experimental study of the query processing system of the heterogeneous distributed database provided for determining the weighting coefficients was also conducted.

The main programming language used when writing a service for viewing video content was Java.

ЗМІСТ

ВСТУП	9
1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ БАЗ ДАНИХ В КОНТЕКСТІ СТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ	10
1.1 Архітектури інформаційних систем	10
1.1.1 Архітектурна побудова сучасних інформаційних систем	10
1.1.2 Архітектура файл-сервера	11
1.1.3 Архітектура клієнт-сервера	13
1.1.4 Багаторівнева архітектура	17
1.2 Теоретичне порівняння реляційної та об'єктно-орієнтованої моделі даних	18
1.3 SQLite	21
1.4 MySQL	23
1.5 PostgreSQL	24
Висновки до розділу 1	26
2 ТЕОРЕТИЧНІ ВІДОМОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВЕБ СЕРВІСУ ВІДЕОХОСТИНГУ	27
2.1 Характеристика потенційної аудиторії клієнтів додатку	27
2.2 Ціль і призначення проекту	28
2.3 Опис процесу розробки інформаційної системи відеохостингу	30
2.4 Опис структури програми	31
2.5 Опис технологій для створення веб сервісу	36
2.5.1 SQL, MySQL	36
2.5.2 Java, Spring Framework	37

	7
2.5.3 JavaScript	37
2.5.4 Telegram API	38
Висновки до розділу 2	41
3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ПЕРЕГЛЯДУ ВІДЕОКОНТЕНТУ	42
3.1 Архітектура застосунку	42
3.1.1 Список обраних мов програмування та їх переваги.....	42
3.1.2 Опис розробленого застосунку	43
3.1.3 Опис архітектури сервісів	48
3.2 Огляд бази даних розробленого застосунку	49
3.3 Тестування розробленого застосунку	51
3.3.1 Реєстрація.....	52
3.3.2 Вхід у систему	55
3.3.3 Перегляд відеоконтенту	55
3.4 Експериментальні дослідження для визначення вагових коефіцієнтів в системі обробки запитів гетерогенної розподіленої бази даних.	60
Висновки до розділу 3	67
4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	69
4.1 Постановка задачі проектування	69
4.2 Обґрунтування функцій програмного продукту	70
4.3 Обґрунтування системи параметрів програмного продукту	74
4.4 Аналіз експертного оцінювання параметрів	76
4.5 Аналіз рівня якості варіантів реалізації функцій.....	80
4.6 Економічний аналіз варіантів розробки програмного продукту	81

	8
4.7 Вибір кращого варіанту ПП техніко-економічного рівня	88
Висновки до розділу 4	89
ВИСНОВКИ.....	90
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	92
ДОДАТОК А	95
А.1 Створення баз даних	95
А.2 Основний код мікросервісів.....	98
А.3 Основний код інтерфейсу користувача	125
ДОДАТОК Б.....	134

ВСТУП

У сучасному інформаційному суспільстві обсяг медіаконтенту зростає з кожним днем. Разом з цим виникає необхідність ефективного управління та обліку медіаконтенту для користувачів мережі Інтернет, різних організацій та підприємств. У рамках цієї дипломної роботи пропонується розробка інформаційної системи обліку медіаконтенту на основі баз даних.

Для реалізації цієї системи використовуватиметься підхід, заснований на базах даних. База даних буде основним сховищем медіаконтенту, а також містити інформацію про файли, їх властивості, метадані та зв'язки між ними. Крім того, система забезпечуватиме зручний інтерфейс користувача для взаємодії з медіаконтентом, дозволяючи користувачам виконувати пошук, перегляд, редагування та керування файлами.

Очікується, що розроблена інформаційна система обліку медіаконтенту значно спростить процес управління медіаконтентом для організацій, підвищить ефективність роботи з файлами та забезпечить надійне зберігання та доступ до медіафайлів.

Незважаючи на різноманітність існуючих рішень, проблема відеохостингу залишається актуальною на даний момент, оскільки недостатньо простих у використанні відеохостингів з низькими апаратними вимогами, які не перевантажені маловажливими функціями, які ускладнюють взаємодію з основним функціоналом хостингу.

1 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ БАЗ ДАНИХ В КОНТЕКСТІ СТВОРЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Архітектури інформаційних систем

Архітектура інформаційної системи (ІС) – це структурна організація та концептуальний дизайн системи, яка включає в себе компоненти, модулі, зв'язки та інтерфейси, необхідні для забезпечення її функціонування та досягнення поставлених цілей. Архітектура визначає загальну форму та поведінку ІС, включаючи розподіл ролей, взаємодію компонентів, структуру даних, алгоритми обробки та інші аспекти системи.

1.1.1 Архітектурна побудова сучасних інформаційних систем

Внаслідок прискореного темпу просування комп'ютерних та мережевих технологій стала можливою поява величезної кількості інформаційних систем різного призначення як загальнодоступних у рамках глобальної мережі Інтернет, так і вузькоспеціалізованих. Дані інформаційні системи створюються для потреби однієї чи кількох корпорацій чи підприємств. У існуючих реалізаціях систем, що розглядаються, інформація, як правило, зберігається в базі даних. Паралельний доступ до інформаційної системи кількох користувачів створюється через спеціальне програмне забезпечення, спрямоване на роботу в мережі. Програмне забезпечення, яке реалізує взаємозв'язок має перебувати на обох сторонах мережі – на сервері (програма-сервер, що реалізує паралельну обробку запитів до інформаційної системи кількох користувачів) та на комп'ютері користувача (клієнтська частина, що

утворює запити до сервера та показує результати). Ефективна робота інформаційної системи залежить від її архітектури.

Архітектура – це перелік суттєвих рішень, спрямованих на реалізацію програмного забезпечення, структурних елементів та структурних інтерфейсів, за допомогою яких компонується система, разом з їх поведінкою, що встановлюється взаємозв'язок між даними елементами, складання елементів з часом підсистеми, що збільшуються, так само стиль архітектури, який наставляє цю організацію – інтерфейс елементів та самих елементів, взаємозв'язок і компонування.

Архітектура сучасних інформаційних систем та їх конструкція виділяється величезним спектром технічних засобів та програмних засобів. Значний вплив на технічні рішення, що затверджуються, проявляють такі фактори, як тип ЕОМ, структура бази даних, обсяг бази даних і т.д. Зараз сучасні централізовані архітектури інформаційних систем поділяють такі типи:

- система, що базується на архітектурі файл-сервера;
- система, що базується на архітектурі клієнт-сервера;
- система, що базується на багаторівневій архітектурі.

1.1.2 Архітектура файл-сервера

В архітектурі файл-сервера база даних розташована на сервері, клієнт може звертатися до сервера з файловими командами, а структура регулювання всіма інформаційними забезпеченнями розташована на комп'ютері клієнта.

На малюнку 1.1 представлено структуру архітектури файл-сервера.

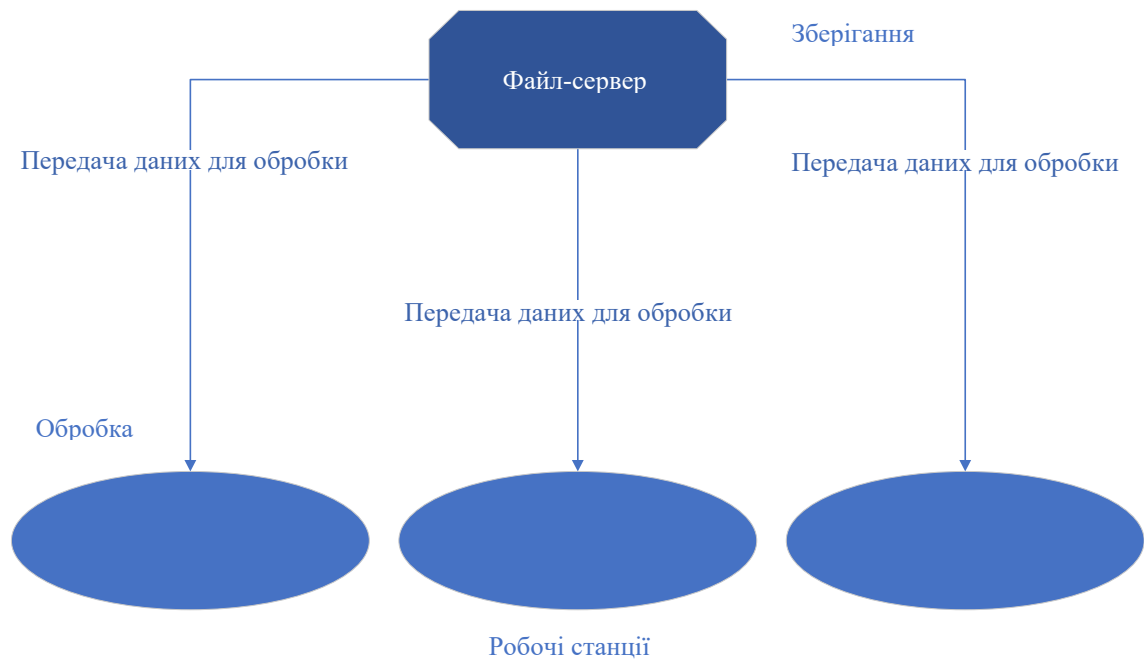


Рисунок 1.1 – Структура інформаційної системи із файл-сервером

Дана архітектура має досить велику кількість недоліків, що стримують діапазон виконуваних нею завдань. Простим прикладом є подія, коли дані фізично розташовані на комп'ютері, в якому знаходиться сама програма.

Перевагами аналізованої архітектури є:

- розрахований на багато користувачів режим роботи з даними;
- зручність централізованого керування доступом;
- невисока вартість реалізації;
- підвищена швидкість реалізації;
- низька вартість оновлення та зміни програмного забезпечення.

Недоліками архітектури, що розглядається, є:

- відсутність гарантії цілісності;
- невисока продуктивність (залежить від продуктивності мережі, сервера, клієнта);
- неякісна можливість підключення нових клієнтів;
- нестійкість системи.

Суттєве завантаження локальної мережі переданими даними. Завдання сервера полягає у зберіганні даних та кодів програми.

Завдання клієнта у тому, щоб виправляти дані, які здійснюються лише за клієнта.

Вся складність обчислювальної роботи під час доступу до бази даних лягає на додаток клієнта. Це відбувається в результаті принципу обробки інформації у системах «файл-сервер». При наданні запиту на отримання даних з таблиці баз даних, вся таблиця копіюється на клієнтське місце, і вибірка реалізується на клієнтському місці. Локальні СУБД скористаються так званим «навігаційним підходом», орієнтованим на виконання окремих даних.

Архітектура файл-сервера має очевидні недоліки. Основний недолік у тому, що дані обробляються та зберігаються в окремих місцях. Отже, їх треба посилати по мережі, що створює дуже високе навантаження на мережу і, в результаті різко знижується продуктивність програми у разі збільшення числа паралельних клієнтів. Іншим значним недоліком даної архітектури є децентралізоване вирішення проблем цілісності та злагоженості даних та одночасного доступу до даних. Такий підхід не є надійним, і тому він знижує безпеку програми. У великих і заплутаних випадках архітектура файл-сервера стає недостатньою, тому в складних випадках ними не користуються.

1.1.3 Архітектура клієнт-сервера

Ця архітектура є наступним кроком у прогресії СУБД. Особливість цієї архітектури у цьому, що дозволяє перенести обчислювальну навантаження на сервер бази даних (SQL-сервер) і цим максимально розвантажує додаток клієнта від обчислювальної роботи. Також слід зазначити значне зміцнення надійності даних – як від внесення шкоди, і від неправильної зміни даних.

База даних, поштові служби, служби друку та файлові системи є типом ресурсу для комп'ютерної мережі. Тому тип сервера залежить від типу ресурсу

сервера. Наприклад, сервер називається сервером бази даних за умови, що сама база даних є керованим ресурсом.

Клієнт-серверна архітектура розрахована для знаходження вирішення проблем файл-серверних програм за допомогою розподілу та розташування частин програми там, де вони найбільш якісно функціонуватимуть. Відмінною рисою клієнт-серверної архітектури є існування окремих серверів баз даних, які приймають запити на мові SQL, і які проводять сортування даних, пошук даних та агрегування даних. Характерна властивість для серверів баз даних – це наявність сховища метаданих. У цьому сховищі записуються обмеження цілісності інформації, конструкція баз даних, серверні та формати процедур, які обробляють дані за подіями або викликом у програмі. Об'єктами реалізації в таких додатках, за винятком логіки та діалогу обробки, є спочатку модель реляційної бази даних і набір SQL-операторів, що відноситься з ними, для типових запитів до бази даних.

У клієнт-серверній платформі розробляються бази даних із величезною кількістю користувачів. Використовуючи таку платформу, отримання доступу до баз даних для групи користувачів відбувається спеціальним комп'ютером – сервером. Клієнт дає доручення серверу зробити різні процеси оновлення чи пошуку баз даних. І потужний сервер, націлений на роботи із запитами, реалізовує їх та повідомляє клієнта про отримані результати. Подібна реалізація роботи збільшує якість реалізація програм за допомогою використання потужності сервера, звільняє мережу, гарантує хороший контроль цілісності даних.

У клієнт-серверних баз даних виявляється додаткова скрута – потрібно додаток реалізовувати в такому вигляді, щоб ця програма використовувала можливості сервера максимально, а навантаження на мережу було б мінімальним, відправляючи через неї мінімум даних. Для зменшення продуктивності на мережу, спрощення адміністрування додатків та збільшення швидкості виконання клієнтських додатків з віддаленої бази даних вся логіка прийняття рішень оформляється у вигляді збережених процедур і

реалізується на сервері баз даних. Збережені процедури записуються спеціальною алгоритмічною мовою. У цих процедурах пишуться послідовності запитів до бази даних, що часто відтворюються. Текст процедури зберігається на сервері у скомпільованому вигляді. Переваги в застосуванні процедур, що зберігаються, очевидні:

- виключає потребу в синтаксичній перевірці запитів та їх компілювання перед реалізацією;
- виключає потребу здійснення у клієнтській програмі запитів, які визначені в тілі процедур, що зберігаються.

Збережені процедури збільшують цілісність додатків та базу даних, гарантують актуальність спільних операцій та обчислень. Підвищується ефективність проведення таких процедур, а також підвищує надійність (тобто немає можливості прямого доступу до даних).

Слід зазначити, що перевантаженість процедур, що зберігаються прикладною логікою може навантажувати сервер, внаслідок цього погіршується продуктивність. При реалізації великих інформаційних систем, сервер яких паралельно може звертатися одразу до кількох клієнтів, така проблема є особливо актуальною. Через це в багатьох випадках необхідно ухвалити компромісні рішення: частину логіки додатку поміщати на стороні клієнта та частину на стороні сервера. Клієнт-серверні системи з таким компромісним рішенням називаються системами з розділеною логікою. Описана архітектура є дворівневою і називається «товстим клієнтом», яка зображена на рис 1.2.

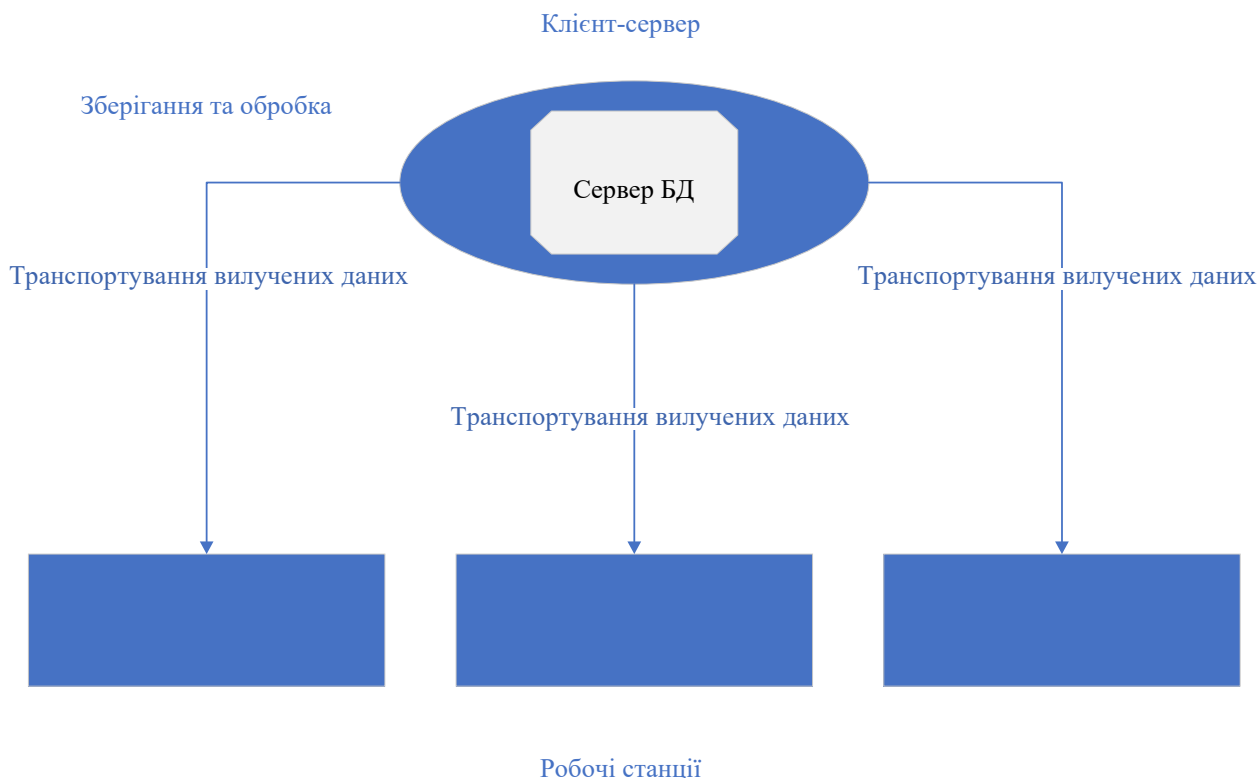


Рисунок 1.2 – Структура інформаційної системи із клієнт-сервером

В даний час клієнт-серверна архітектура як метод розробки додатків для робочих груп та інформаційних систем корпоративного рівня, набула визнання, і стало широко поширюватися. Збільшення безпеки даних у розглянутій архітектурі пов'язане з тим, що виконання запитів всіх клієнтів реалізується єдиною програмою, яка розміщена на сервері. Сервер забезпечує загальні всім користувачів правила використання баз даних, може керувати режимами доступу клієнтів до інформації, не дозволяючи, зокрема, паралельне виправлення однієї записи кількома користувачами. Скорочується складність клієнтських додатків, оскільки в них відсутній код, який пов'язаний з контролем баз даних та розмежованим доступом до неї. Для роботи зі складними інформаційними програмами, в яких безліч користувачів та заплутана логіка, в більшості випадків не використовуються дворівневі схеми архітектури клієнт-сервера, оскільки вони можуть призвести до проблем. У разі застосовується багаторівнева архітектура.

1.1.4 Багаторівнева архітектура

Багаторівнева архітектура є вдосконаленням вище розглянутих архітектур. На останньому рівні на комп'ютерах користувача розміщені програми клієнтів, призначені для проведення функцій і логіки відображень, які гарантують виклик додатків у програмному інтерфейсі на середньому рівні. Сервер додатків, що знаходиться на середньому рівні, здійснює прикладну логіку, та який здійснює операції логіки з обробки даних з бази даних. Тобто на середньому рівні забезпечується обмін даними між розподіленою базою даних та користувачем. Таким чином, сервер додатків міститься у вузлі мережі, до якого мають доступ усі клієнти.

Класична форма багаторівневої архітектури складається з трьох рівнів, як показано на рис 1.3.

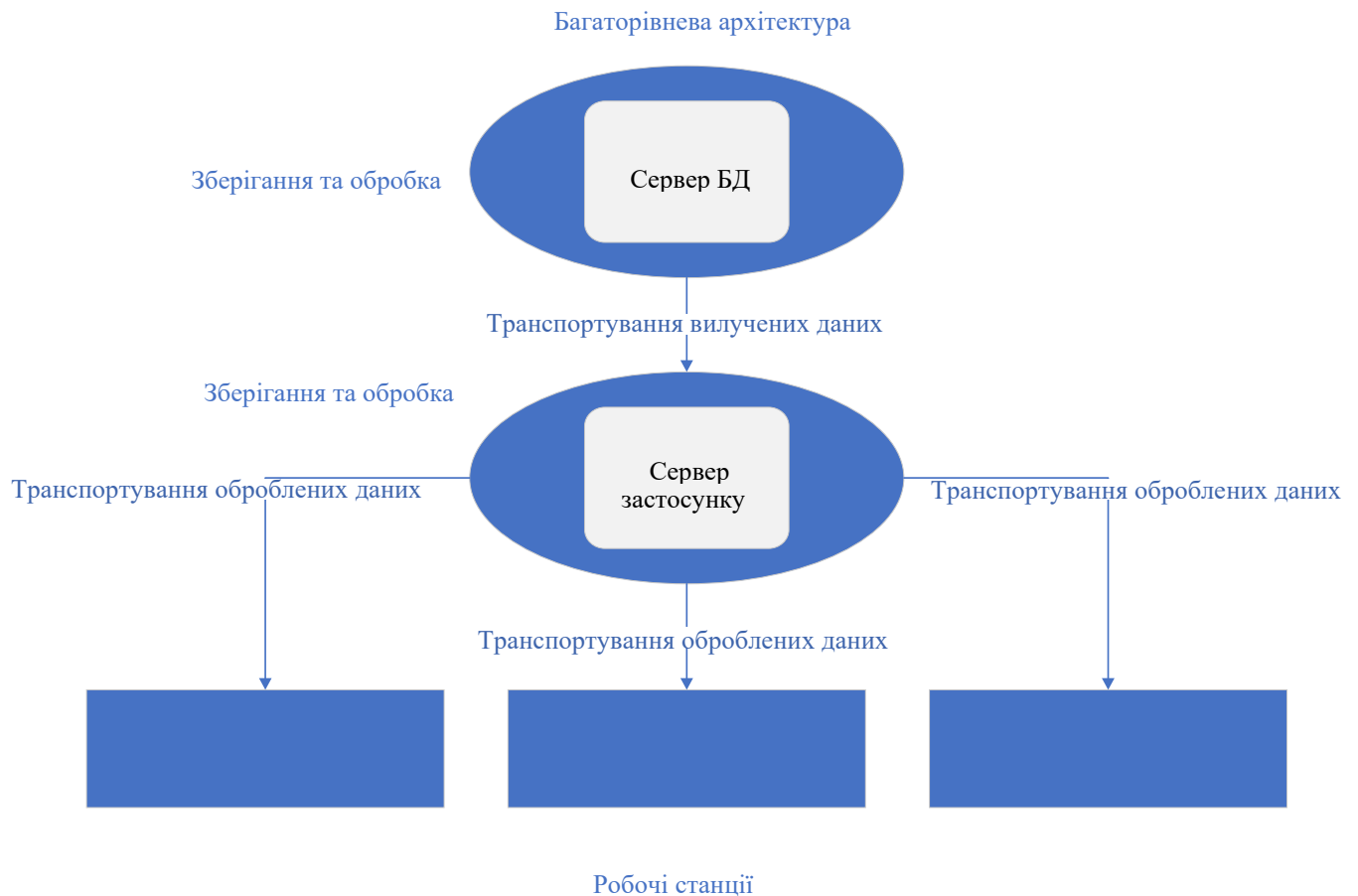


Рисунок 1.3 – Структура багаторівневої архітектури

На третьому верхньому рівні розташований віддалений спеціалізований сервер баз даних, що приймає інформацію від сервера додатків. Сервер програм надсилає інформацію на даний сервер. Послуги сервера баз даних є обробка отриманих даних та файлова операція.

Перевагою трирівневої архітектури є:

- розвантаження сервера бази даних від реалізації частини операцій, які перенесені на сервер додатків;
- розвантаження клієнтської програми за рахунок видалення зайвого коду, що призводить до мінімізації розміру;
- усі клієнти мають єдину поведінку;
- спрощення налаштування клієнтів, тобто після зміни коду на сервері, поведінка клієнтських програм автоматично змінюється.

Трирівнева модель архітектури ліквідує недоліки, що зустрічають у дворівневій моделі. Трирівнева архітектура дає можливість ще більше встановити рівновагу навантаження на мережу. Це дає їй перевагу над рештою архітектур.

Кількість систем з клієнт-сервером збільшується з кожним роком. Тому необхідність трирівневої архітектури стає дедалі очевидною. Розглянута архітектура підвищує якість виконання роботи інформаційних систем та покращує розташування її ресурсів.

1.2 Теоретичне порівняння реляційної та об'єктно-орієнтованої моделі даних

Для порівняння об'єктно-орієнтованих і реляційних баз даних необхідно аналізувати їх особливості.

Незважаючи на поширеність об'єктно-орієнтованого програмування, у технології створення баз даних цей метод не дуже поширений. І тому є як суб'єктивні, і об'єктивні причини.

- Популярність. Багато відомих і широко використовуваних продукти реалізовані реляційним методом. Ці продукти і надалі розвиватимуться і підтримуватимуться, тому що замовники в них вклали великі гроші, і вони готові далі фінансувати та розвивати їх. Навпаки, під об'єктно-орієнтовані СУБД реалізовано значно менше хороших комерційних товарів порівняно з реляційними, тобто на сьогоднішній день реалізовано мало потужних об'єктно-орієнтованих СУБД.

- Мова запитів та її стандартизація. У 1986 році вперше було прийнято мову запитів SQL-86. Після прийняття цього стандарту всі розробники реляційних баз даних почали слідувати йому. Навпаки, для об'єктно-орієнтованих баз даних нині немає мови запиту. Що значно ускладнює процес розробки об'єктно-орієнтованих баз даних. У розробників навіть немає єдиної думки про те, що має робити мова запитів, не кажучи вже про те, яким методом вона працюватиме.

- Математичний апарат. Едгар Кодд для реляційних баз даних створив і описав математичний апарат реляційної алгебри. Цей апарат пояснює, як будуть реалізуватися операції над зв'язками в базі даних, а також доводить їх оптимальність (або з нього видно, де треба оптимізувати). Незважаючи на те, що цей метод був розроблений у 80-х роках, але досі для неї не створено такого апарату. Тому в об'єктно-орієнтованих базах даних не з'явилися такі суворі терміни, як відношення, декартовий добуток тощо.

- Проблема зберігання даних та методів. У реляційних базах даних вся інформація написана та зберігаються в таблицях. Але в об'єктно-орієнтованих базах даних вся інформація міститься в об'єктах. У даний момент також немає єдиної думки, як у даній моделі створення бази даних повинні реалізовуватися зберігання об'єктів і взагалі, як розробники повинні розробити

та спроектувати ці об'єкти. Згодом виникають такі проблеми, як збереження ієрархії об'єктів, збереження абстрактних класів тощо.

Ці дві архітектури інформаційних систем багато в чому різняться. Основні ознаки, якими розрізняються об'єктно-орієнтована і реляційна база даних, наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння основних ознак база даних реляційними та об'єктно-орієнтованими СУБД

Ознака	Об'єктно-орієнтована СУБД	Реляційна СУБД
Модель даних	Об'єктна	Таблична
Спосіб читання об'єктів	Дані об'єкта поміщаються на зберігання як єдине ціле	Об'єкт збирається з окремих елементів і тільки потім використовується
Характеристика зв'язки між об'єктами	Кращі інструменти для відображення реального світу. Природне представлення даних на різних рівнях	Всі зв'язки належать одному рівню
Нові типи даних	Модифікація ядра не потрібна при додаванні нового типу даних	Потрібна модифікація ядра
Оптимізації ядра СУБД	Ядро оптимізовано для операцій з об'єктами	Ядро залишається реляційним
Мова СУБД й запити	Варіація реалізації OQL	Різні варіанти SQL

Відомий ідеолог СУБД Мері Луїнс сказав, що об'єктно-орієнтована СУБД актуальніша і ближча до сутностей реального світу. Усі дані можна використовувати, не розміщуючи їх у різних таблицях. Також можна використовувати великий набір певних типів.

Незважаючи на це можна зробити висновок, що реляційна модель на сьогоднішній день є найбільш укоріненою і підтримуваною.

1.3 SQLite

SQLite – це автономна система керування реляційною базою даних із відкритим вихідним кодом, яка базується на файлах. Універсальність і стабільність у програмах з малим об'ємом пам'яті, а також мобільність роблять його популярним. Навіть коли система виходить з ладу або відбувається відключення електроенергії, вона використовує транзакції, за стандартами з ACID.

SQLite вважається «безсерверною» базою даних. Серверні операції в традиційних реляційних базах даних включають програми, які підключаються до хост-сервера через запити. Проте в SQLite кожен процес, який взаємодіє з базою даних, читає або записує файл на диску. Метод налаштування SQLite значно легший, ніж стандартна архітектура реляційної бази даних, оскільки він не потребує налаштування сервера. Програми SQLite також не вимагатимуть додаткових налаштувань, тому що їм потрібно лише контролювати доступ до диска.

SQLite є відкритим і безкоштовним, тому додаткова ліцензія не потрібна. Ініціатива надає різні платні розширення за одноразову плату. Вони допомагають у шифруванні та стисненні даних.

SQLite підтримує широкий спектр форматів даних.

До переваг SQLite можна віднести наступне.

- Невеликий розмір: оскільки SQLite автономний, він не потребує встановлення додаткових компонентів у вашій системі для роботи. SQLite займає дуже мало місця, як впливає з назви. Хоча це залежить від системи, зазвичай вона займає менше 600 КБ.

- SQLite, також відомий як база даних без конфігурації, практично готовий до використання після встановлення. SQLite ніколи не потрібно запускати, перезапускати чи зупиняти, оскільки він не працює на сервері. Також немає файлів конфігурації, про які варто турбуватися. Тому SQLite легко встановити та налаштувати для інтеграції з програмою.

- Портативність: база даних SQLite міститься в одному файлі. З іншого боку, інші системи керування базами даних часто зберігають дані у великих пакетах окремих файлів. Один SQLite може бути розташований будь-де в ієрархії каталогів і може бути переміщений за допомогою протоколу передачі та портативного носія.

До недоліків SQLite відносять зазвичай.

- Обмеження паралелізму: база даних SQLite підтримує кілька з'єднань. Однак одночасно може відбуватися лише одна зміна даних. Хоча SQLite підтримує рудиментарний паралелізм, йому не вистачає адаптивності MySQL, PostgreSQL та інших клієнт/серверних СУБД.

- Надсильний контроль користувача: заборони доступу SQLite є звичайним обмеженням операційних систем. Таким чином, SQLite не є хорошим рішенням для програм, які вимагають від користувачів персональних прав доступу.

- Низький рівень безпеки: оскільки складніше завдати шкоди стороні сервера з точки зору цілісності даних, помилки програми завдають набагато меншої шкоди механізму бази даних, який використовує сервер. Система клієнт-сервер перевершує безсерверну СУБД з точки зору контролю доступу, паралелізму та детального блокування.

1.4 MySQL

MySQL є найпопулярнішою реляційною СУБД з відкритим вихідним кодом. MySQL запускає деякі з найпопулярніших веб-сайтів у світі, включаючи Netflix, Spotify, Twitter і Facebook. Початок роботи з MySQL є легкою процедурою через величезну кількість документації та велику спільноту розробників, а також багато Інтернет-ресурсів, що стосуються цього.

MySQL не повністю відповідає «стандартам SQL», оскільки він був розроблений з урахуванням надійності та зручності. У той же час розробники MySQL наполегливо працюють над їх відповідністю. Однак цей процес часто відбувається повільно. На відміну від SQLite, який забезпечує прямий доступ до програм, MySQL дозволяє отримати доступ до бази даних через окремий запит до сервера. Оскільки між програмами та базою даних є сервер, це забезпечує більший контроль над доступом до бази даних.

MySQL випустила безліч програм сторонніх розробників, а також інтегрованих бібліотек, які не тільки полегшують роботу, але й підвищують її можливості. HeidiSQL, phpMyAdmin і DBeaver є одними з найпопулярніших утиліт сторонніх розробників.

До переваг MySQL належить.

- Простота використання та популярність: популярність MySQL призвела до появи великої кількості веб-документів і друкованої документації. Крім того, MySQL містить низку інших інструментів, таких як phpMyAdmin, які допомагають у процесі початкового налаштування.

- Безпека: під час інсталяції MySQL містить сценарій, який допомагає підвищити безпеку бази даних, знаходячи пароль користувача root, видаляючи бази даних, доступні всім користувачам за замовчуванням, і налаштовуючи параметри захисту паролем. Крім того, на відміну від SQLite, MySQL дозволяє регулювати вибір доступу для привілеїв користувачів.

– Розробники MySQL змогли підкреслити продуктивність, не реалізувавши всю функціональність SQL. Незважаючи на недавній успіх конкуруючих систем керування базами даних, MySQL зберігає репутацію однієї з найшвидших СУБД.

– MySQL дозволяє багаторазово повторювати базу даних. Переносячи інформацію між багатьма хостами, реплікація підвищує доступність бази даних, надійність і відмовостійкість.

До недоліків MySQL відносять наступне.

– Оскільки MySQL створено для простоти використання, швидкості та зручності, існують певні функціональні обмеження через невідповідність SQL. Одним із прикладів є підтримка речень FULL JOIN.

– Повільний розвиток: користувачі помітили повільніший процес розробки після того, як у 2008 році MySQL було куплено компанією Sun Microsystems, а наступного року – корпорацією Oracle, оскільки спільнота з відкритим кодом більше не реагує на виправлення помилок і розробку.

1.5 PostgreSQL

PostgreSQL, іноді відомий як Postgres, вважає себе найскладнішою реляційною базою даних з існуючих. PostgreSQL, об'єктно-реляційна система керування базами даних, включає такі можливості, як успадкування таблиць і перевантаження функцій для відповідності стандартам і розширенням. Ці функції часто пов'язані з об'єктними базами даних.

Паралельність (здатність обробляти багато процесів одночасно) є дуже популярною функцією PostgreSQL. Це досягається за допомогою впровадження багатоверсійного керування паралелізмом (MVCC), яке забезпечує відповідність транзакції ACID.

Хоча PostgreSQL не так широко використовується, як MySQL, існує багато рішень, які полегшують його використання. Postbird і pgAdmin є двома прикладами таких програм.

PostgreSQL, як і MySQL, приймає числові типи даних, дату/час і текстові типи даних. Однак він обробляє додаткові типи даних, такі як мережеві адреси, геометричні форми, бітові рядки, записи JSON, текстовий пошук та інші спеціалізовані типи даних.

Перевагами PostgreSQL є таке.

- Відповідність SQL: PostgreSQL працює значно складніше, ніж MySQL або SQLite, щоб забезпечити відповідність SQL. Він підтримує 160 із 179 необхідних функцій SQL Baseline: 2011. Відповідно до офіційної літератури, він також має кілька додаткових функцій.

- PostgreSQL має велику та активну спільноту розробників, яка підтримує та надає численні онлайн-ресурси, які описують, як ефективно її використовувати. Є кілька онлайн-форумів, авторитетна документація і навіть PostgreSQL Wiki.

- Розширюваність: PostgreSQL можна розширити на льоту за допомогою динамічного завантаження та керування каталогами. Якщо PostgreSQL цього вимагає, можна вказати для завантаження файл об'єктного коду (наприклад, спільну бібліотеку).

До недоліків PostgreSQL відносять.

- Проблеми з продуктивністю пам'яті: кожне підключення клієнта створює новий процес PostgreSQL. Для кожного процесу потрібно приблизно 10 МБ оперативної пам'яті. Це може призвести до величезної кількості підключень користувачів, які споживають багато оперативної пам'яті. Це означає, що PostgreSQL не працює так само добре, як конкуруючі СУБД у налаштуваннях, які потребують інтенсивного читання.

- Популярність: незважаючи на значний рівень використання в останні роки, PostgreSQL значно відстає від MySQL з точки зору популярності.

Однією з причин є те, що PostgreSQL має менше сторонніх утиліт. Крім того, менше адміністраторів баз даних з досвідом PostgreSQL, ніж з MySQL.

Висновки до розділу 1

Отже, було визначено що таке архітектура в контексті сучасних інформаційних систем та надано основні її характеристики й варіації; було проведено багатогранний аналіз й порівняння двох моделей даних, а саме реляційної та об'єктно-орієнтовної; приведено перелік основних переваг та недоліків PostgreSQL, MySQL та SQLite.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВЕБ СЕРВІСУ ВІДЕОХОСТИНГУ

2.1 Характеристика потенційної аудиторії клієнтів додатку

Характеристика потенційної аудиторії клієнтів сервісу відеохостингу може включати наступні аспекти.

- Вікова група: аудиторія може бути різноманітною віковою групою, від підлітків до дорослих. Залежно від специфіки контенту, сервіс може привертати підлітків, молодь або дорослих користувачів.
- Інтереси та хобі: аудиторія може мати різні інтереси та хобі, такі як музика, фільми, спорт, мода, геймінг, кулінарія, подорожі, наука, освіта тощо. Сервіс може надавати різноманітний контент, щоб задовольнити різні інтереси аудиторії.
- Географічна приналежність: аудиторія може бути розподілена по різних регіонах світу або бути орієнтована на певну географічну область. Мова та культурні впливи можуть грати роль у виборі контенту та взаємодії з сервісом.
- Професійні інтереси: деякі відеохостинги можуть спеціалізуватися на конкретних професійних групах, таких як музиканти, відеографи, блогери, викладачі тощо. Аудиторія може складатися з фахівців та аматорів відповідної галузі, які шукають платформу для спільного обміну своїм вмінням та взаємодії з однодумцями.
- Технічна готовність: аудиторія може мати різний рівень технічних навичок та здатність використовувати різноманітні функції сервісу. Деякі користувачі можуть бути більш технічно обізнані і зручно володіти функціями завантаження, відтворення та взаємодії з відео, тоді як інші можуть виявляти певні труднощі і бажати більш простого та інтуїтивного інтерфейсу.

- Мобільність: зростання використання смартфонів і планшетів призводить до збільшення аудиторії, яка споживає відеоконтент на мобільних пристроях. Така аудиторія може цінувати оптимізовані додатки та адаптивний дизайн для зручного перегляду відео на різних пристроях.

- Соціальні зв'язки: деякі користувачі можуть шукати можливості взаємодії та спілкування з іншими користувачами. Сервіс відеохостингу може надавати функції коментування, обміну повідомленнями, створення спільнот та можливість підписатися на улюблені канали.

- Зацікавленість у контенті: аудиторія може бути зацікавлена в різних типах контенту, таких як розважальний, освітній, спортивний, музичний, науковий, політичний тощо. Сервіс може надавати різноманітні категорії та рекомендації для того, щоб відповідати потребам різних користувачів.

Ці характеристики можуть варіюватися в залежності від конкретного відеохостингу та його цільової аудиторії. Комбінація цих факторів допомагає розуміти, яка аудиторія має бути залучена та яким контентом вона може бути зацікавлена.

Важливо розуміти свою потенційну аудиторію, аналізувати її потреби, інтереси та звички споживання відеоконтенту. Це допоможе розробити та просувати відповідний контент, забезпечуючи задоволення потреб клієнтів та залучення нових користувачів до сервісу відеохостингу.

2.2 Ціль і призначення проекту

Ціль і призначення проекту розробки сервісу відеохостингу полягають у створенні і наданні платформи, де користувачі можуть завантажувати, зберігати, відтворювати та обмінюватися відеоконтентом. Основні аспекти цілей та призначення проекту включають наступне.

– Забезпечення простоти та зручності: однією з головних метою розробки сервісу відеохостингу є створення зручного та простого у використанні інтерфейсу для користувачів. Це включає у себе легке завантаження відео, швидке відтворення, просту навігацію та зручну систему керування вмістом.

– Забезпечення надійності та безпеки: проект має на меті забезпечити безпеку відеоконтенту, запобігаючи несанкціонованому доступу до відеофайлів або порушенню авторських прав. Надійна система зберігання та резервне копіювання даних також є важливими аспектами для забезпечення безпеки відеоконтенту.

– Підтримка широкого спектру форматів та якостей відео: сервіс повинен підтримувати різні формати відео (наприклад, MP4, AVI, MKV) та якості відтворення (SD, HD, 4K), щоб задовольнити потреби різних користувачів і їхні можливості з пристроїв та інтернет-з'єднання.

– Забезпечення функціональних можливостей: проект повинен надавати різноманітні функції, які полегшують споживання та взаємодію з відео. Це може включати можливість коментувати, оцінювати, підписуватися на канали, рекомендації на основі особистих інтересів та інші соціальні функції.

– Розширення аудиторії та залучення: проект повинен привертати як можна більше користувачів до своєї платформи. Це може включати просування сервісу, створення партнерських програм, співпрацю з відомими відеоблогерами та інші маркетингові заходи, що сприяють залученню нових користувачів.

– Аналітика та звітність: проект повинен забезпечувати інструменти аналізу та звітності для контент-творців, щоб вони могли вимірювати успішність свого відеоконтенту, розуміти поведінку аудиторії та приймати на цій основі відповідні рішення.

– Створення спільноти: сервіс відеохостингу може створити простір для спілкування, обміну думками та ідеями серед користувачів. Це допомагає

створити спільноту відеолюбителів, яка взаємодіє, співпрацює та обговорює відео контент.

– Забезпечення доступу до різноманітного вмісту: сервіс відеохостингу дозволяє користувачам отримати доступ до широкого спектру відео – від розважального контенту до навчальних матеріалів. Це допомагає розширити знання, збагатити розваги та забезпечити різноманіття вмісту для користувачів.

2.3 Опис процесу розробки інформаційної системи відеохостингу

Процес розробки системи може бути поділений на кілька етапів. Першим етапом є аналіз специфікацій і потреб, де визначаються можливості системи, обмеження, вимоги до якості та безпеки, а також функціональні та нефункціональні вимоги.

Після аналізу переходимо до етапу проектування архітектури системи, де встановлюється структура системи, розподіл на частини та опис того, як вони взаємодіють між собою. На цьому етапі розробники також обирають необхідні технології, інструменти та фреймворки для впровадження системи.

Наступним етапом є створення бази даних, де збирається інформація про користувачів, коментарі, відео та інші дані, необхідні для системи. Розробка структури бази даних та зв'язків має гарантувати ефективний доступ до інформації і захист від небажаного доступу.

На етапі розробки інтерфейсу користувача дизайнери та розробники працюють над створенням зручного та привабливого інтерфейсу, що відповідає потребам користувачів.

При розробці функціональності системи розробники створюють програмну логіку, яка забезпечує працездатність системи, включаючи

управління користувачами, завантаження, зберігання, перегляд та коментування відео.

Після завершення розробки система піддається налагодженню та тестуванню. Цей етап включає перевірку системи на відповідність вимогам, виявлення помилок, проведення функціональних, навантажувальних, безпекових тестів та інших видів тестувань. Це дозволяє виявити помилки та недоліки системи та виправити їх перед запуском.

Коли тестування проведено успішно, система готова до запуску. Розробники випускають її на ринок і починають надавати підтримку, оновлювати та покращувати систему на основі відгуків користувачів та змін у вимогах та технологіях.

Весь процес розробки системи включає аналіз потреб та специфікацій, проектування архітектури, створення бази даних, розробку інтерфейсу користувача, зростання функціональності, налагодження та тестування, а також запуск та підтримку. Це ітеративний процес, де кожен етап взаємозв'язаний з попереднім і наступним, що дозволяє створити функціональну та якісну систему.

2.4 Опис структури програми

В описі структури програми для інформаційної системи відеохостингу можуть бути вказані такі компоненти.

1. Інформація про користувачів (рис. 2.1), відео (рис. 2.2), коментарі (рис. 2.3), налаштування системи та інші сутності, необхідні для функціонування системи, зберігаються в таблицях бази даних. Щоб гарантувати зв'язок даних і цілісність бази даних, у базі даних повинні бути сформовані зв'язки між таблицями.

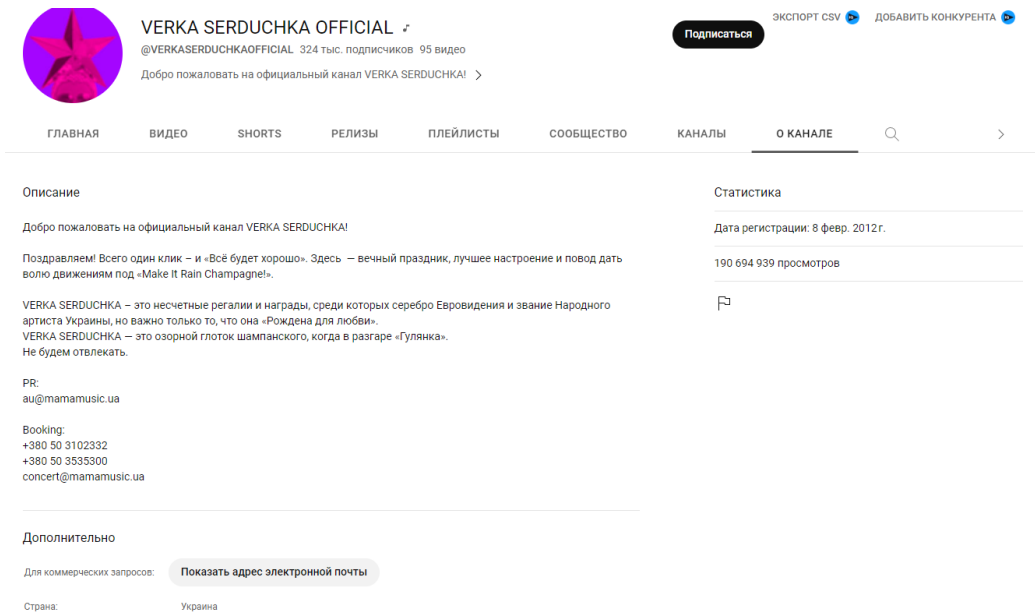


Рисунок 2.1 – Приклад сторінки інформації про користувача

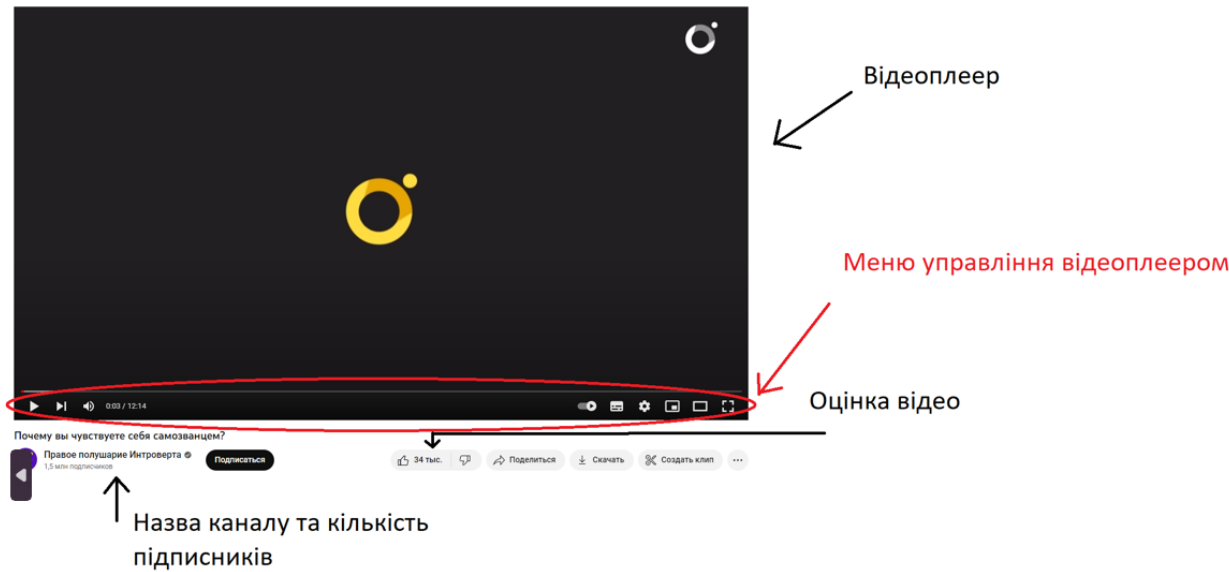


Рисунок 2.2 – Огляд на блок відео

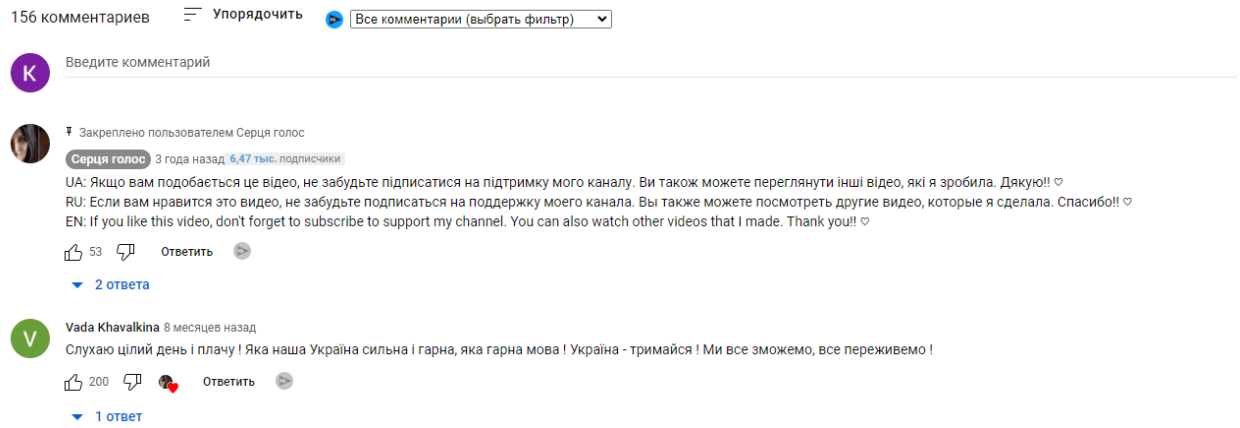


Рисунок 2.3 – Приклад блоку коментарів

2. Основний компонент системи - сервер, відповідає за обробку запитів користувачів і забезпечує безперебійну роботу всього. Прикладами серверного компонента є веб-сервер (рис. 2.4), внутрішній сервер і системи обробки та зберігання відеоінформації.

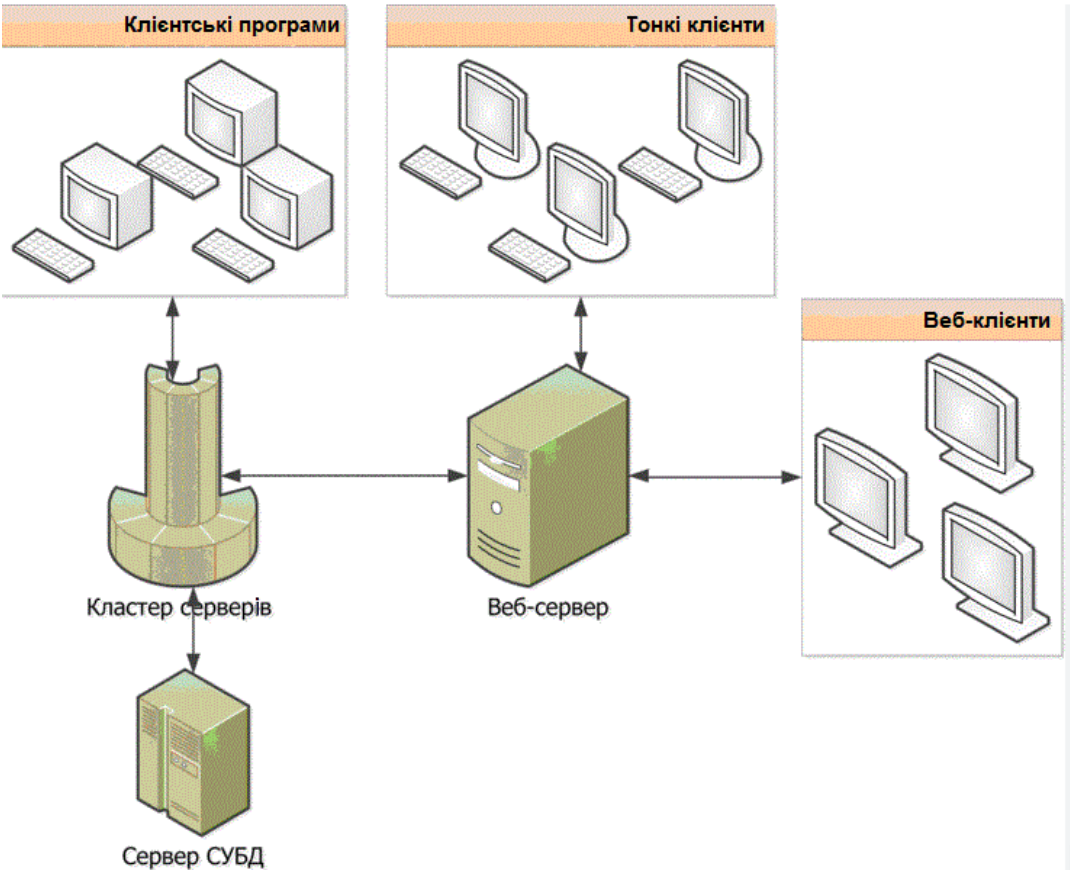


Рисунок 2.4 – Приклад роботи веб-серверу

3. Клієнтська сторона відноситься до досвіду користувача з системою та інтерфейсом, який він використовує для спілкування з нею. Користувачі можуть переглядати, завантажувати та коментувати фільми на стороні клієнта, яка може відображатися як веб-інтерфейс, мобільний додаток або інший інструмент.

На рисунку 2.5 зображено приклад інтерфейсу веб-сервісу для перегляду фільмів MEGOGO.

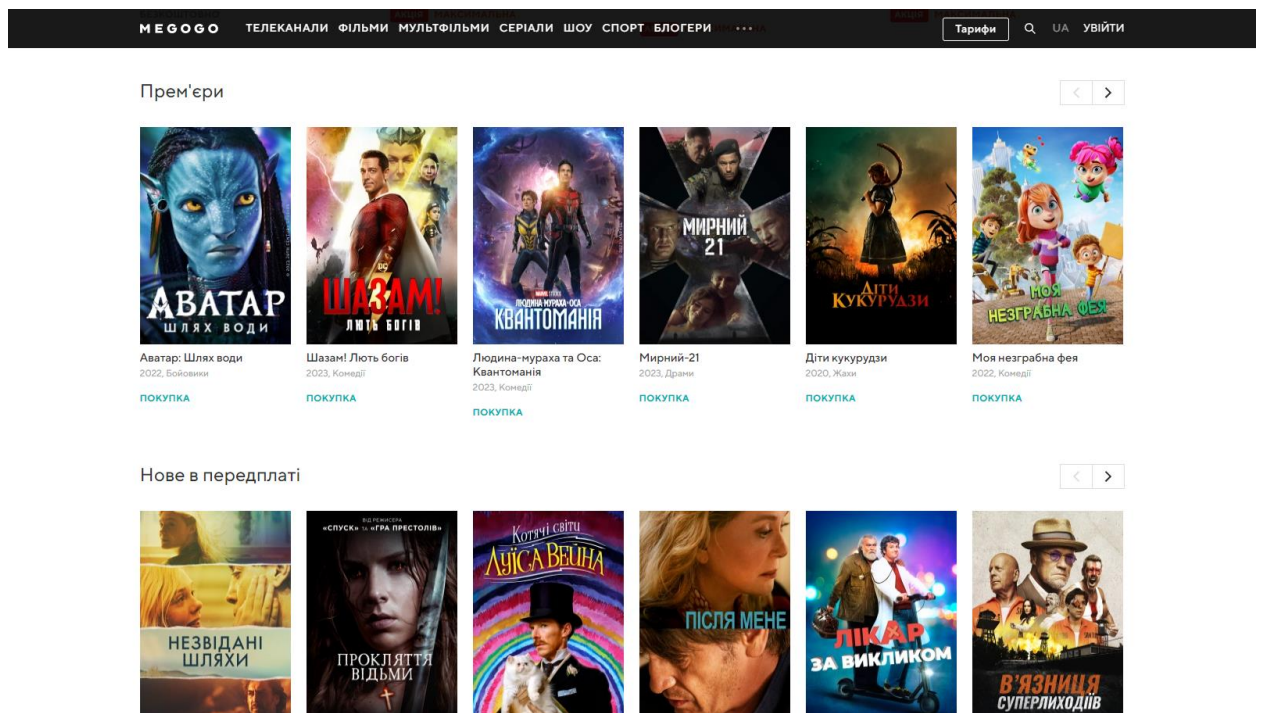


Рисунок 2.5 – Інтерфейс веб-сервісу MEGOGO

4. API (інтерфейс прикладного програмування): API дозволяє стороннім розробникам інтегрувати свої програми з системою для розміщення відео. Розробники мають доступ до численних функцій і даних системи через API, наприклад можливість надсилати фільми та отримувати інформацію про користувачів.

На рисунку 2.6 зображено приклад інтегрування API YouTube.



Рисунок 2.6 – Відео інтегроване завдяки YouTube API

5. Модуль аналітики відповідає за збір і оцінку інформації про поведінку користувачів на веб-сайті, включаючи кількість залишених коментарів, кількість переглядів відео, кількість часу, проведеного на сайті, тощо. Взаємодія з користувачем, маркетингові ініціативи, і інші способи використання цих даних можливі.

6. Модуль безпеки: цей компонент системи відповідає за захист від спроб злому та інших ризиків безпеки. Функції для ідентифікації та авторизації користувачів, захисту від DDoS-атак та інші функції можуть бути включені в модуль безпеки.

Крім того, розробники повинні дотримуватися сучасних стандартів розробки, таких як гнучкі методології та методи DevOps, а також враховувати унікальні характеристики архітектури та обраних технологій, щоб забезпечити надійне та безпечне функціонування системи відеохостингу.

2.5 Опис технологій для створення веб сервісу

Для створення відеохостингу можна використовувати кілька технологій, включаючи MySQL, Java, Spring Framework, JavaScript і службу API Telegram.

2.5.1 SQL, MySQL

SQL (Structured Query Language) є мовою програмування, що використовується для управління базами даних. Система керування базами даних (СКБД) SQL дозволяє зберігати та керувати великою кількістю даних у структурованому форматі.

MySQL – це одна з найпопулярніших СУБД у світі. Вона використовується для зберігання даних на багатьох веб-сайтах, включаючи відеохостинги, електронні комерційні платформи та соціальні мережі.

У відеохостингу MySQL використовується для зберігання відео та даних користувачів, які завантажують відео на сервіс. За допомогою MySQL можна легко керувати великою кількістю даних та здійснювати пошук інформації за різними критеріями, такими як тип відео, дата завантаження та кількість переглядів.

MySQL має багато корисних функцій, таких як масштабованість, надійність та швидкість, що роблять її ідеальним вибором для зберігання великої кількості даних на веб-сайтах та додатках. Веб-програми можна створювати за допомогою комп'ютерної мови Java. Java використовується для побудови різноманітних служб відеохостингу, таких як обробка відео та забезпечення безпеки платформи. Крім того, Java дозволяє розділити функціональні можливості на модулі, що робить код легшим для розуміння та розширення.

2.5.2 Java, Spring Framework

Java є однією з найпопулярніших мов програмування, і вона використовується для розробки різних програм, включаючи веб-додатки. Spring Framework є одним з найпопулярніших Java-фреймворків, який забезпечує зручність розробки веб-додатків.

Фреймворк Spring використовує конструкцію MVC (Model-View-Controller), яка дозволяє розділити логіку програми на три компоненти: модель (Model), представлення (View) та контролер (Controller). Модель відповідає за обробку даних та зберігання їх у базі даних, представлення забезпечує відображення даних для користувача, а контролер відповідає за забезпечення взаємодії між моделлю та представленням.

У відеохостингу Spring Framework використовується для розробки веб-додатків та реалізації конструкції MVC. Це дозволяє забезпечити ефективне та організоване управління великою кількістю даних та забезпечити користувачів зручним та швидким доступом до відео та інших даних.

Крім того, Spring Framework дозволяє розробникам додавати нові функції та покращувати існуючі шляхом швидкого та простого інтегрування з іншими інструментами та бібліотеками. Це дозволяє розробникам швидко та ефективно реагувати на зміни вимог користувачів та покращувати додаток відповідно до їх потреб.

2.5.3 JavaScript

JavaScript є однією з найпоширеніших мов програмування у світі, і вона використовується для розробки різноманітних веб-додатків та динамічних веб-сторінок. У відеохостингу JavaScript використовується для досягнення

різних функцій, які дозволяють користувачам зручно та ефективно взаємодіяти з сайтом та відео контентом.

Одна з головних функцій JavaScript в відеохостингу - це створення віджетів, які показують кількість переглядів відео та кількість коментарів на сторінці. Це дає користувачам можливість бачити, як популярний є конкретний відео контент та яка його актуальність для інших користувачів. Крім того, JavaScript дозволяє розробникам відображати на сторінці рекламні блоки, підказки та інші елементи, які забезпечують користувачів інформацією та поліпшують їх досвід взаємодії з сайтом.

Ще одна функція JavaScript в відеохостингу — це можливість динамічного оновлення веб-сторінок без необхідності перезавантаження всієї сторінки. Наприклад, це може включати можливість завантаження додаткового відео контенту на сторінку при прокручуванні, або можливість оновлення віджетів та елементів сторінки без перезавантаження сторінки. Це підвищує швидкість та ефективність взаємодії користувачів з сайтом та дозволяє розробникам реалізувати різноманітні та складні функції без затримок та перебоїв.

2.5.4 Telegram API

Telegram API дозволяє взаємодіяти з платформою Telegram та здійснювати різноманітні дії, такі як відправка повідомлень, зміна статусу користувача, отримання інформації про користувачів та ботів і багато іншого. У відеохостингу Telegram можна використовувати API для взаємодії з користувачами та підписниками каналу, що дає можливість підтримувати комунікацію з ними.

За допомогою Telegram API можна створювати ботів, які можуть відповідати на запитання користувачів та забезпечувати підтримку для

відеохостингу Telegram. Боти можуть взаємодіяти з іншими службами, такими як поштові сервіси, інформаційні портали та соціальні мережі, що дозволяє підтримувати різноманітні функції відеохостингу.

Використання Telegram API дозволяє відеохостингу Telegram швидко та ефективно реалізовувати нові функції та поліпшувати існуючі, що робить його більш привабливим для користувачів. Також Telegram API дозволяє забезпечити безпеку та конфіденційність даних, що є важливим фактором для відеохостингу. При розробці відеохостингу необхідно також враховувати інші технології, такі як HTML і CSS. Хоча CSS – це мова таблиць стилів, яка дозволяє визначати зовнішній вигляд елементів на сторінці, HTML – це мова розмітки, яка використовується для створення структури веб-сторінок.

YouTube є одним із прикладів того, як ці технології використовувалися для створення відеохостингу. Мільйони людей щодня користуються YouTube, що робить його однією з найвідоміших платформ відеохостингу в усьому світі. Java і Spring Framework використовуються платформою YouTube для створення функціональних можливостей програми, JavaScript використовується для взаємодії з користувачем.

Сервіс Telegram API використовується для отримання повідомлень про нові відео на каналі та для реєстрації. MySQL використовується для зберігання відео та даних користувачів.

Веб-дизайнери можуть створити високоякісний відеохостинг, який задовольняє користувачів, враховуючи різноманітні технології. Використовуючи ці технології, ви можете розробити практичний і простий у використанні інтерфейс користувача, який допоможе залучити нових користувачів і збільшити кількість переглядів відео.

Крім того, платформа є масштабованою та готовою до майбутнього розширення завдяки використанню таких технологій, як MySQL, що дозволяє зберігати значні обсяги відео та даних користувача.

Крім того, для створення відеохостингу можна використовувати додаткові технології, зокрема Ruby on Rails, React і Angular. Структура мови

програмування Ruby під назвою Ruby on Rails дає змогу швидко й ефективно створювати програми. React і Angular — це фреймворки для створення користувацьких інтерфейсів, які дозволяють розробити практичний і зручний інтерфейс.

Створення відеохостингу є складним процесом, який вимагає значних зусиль та ресурсів. Необхідно розробляти та впроваджувати різноманітні технічні рішення, які забезпечать якість передачі відео та надійність роботи сервісу в цілому.

Для створення відеохостингу необхідно враховувати такі фактори.

1. Масштаб проекту. Він може варіюватись від невеликого відеоблогу до великого міжнародного сервісу. Масштаб проекту визначається цільовою аудиторією, обсягом контенту та кількістю користувачів.

2. Технічні знання та досвід команди. Розроблення та підтримка відеохостингу вимагає від команди технічних спеціалістів високого рівня знань та досвіду у сферах програмування, мережевих технологій, баз даних та інших технічних областей.

3. Вибір технологій. Для створення відеохостингу необхідно вибрати технології, які найкраще відповідають потребам проекту. Наприклад, для забезпечення швидкої передачі відео можна використовувати Content Delivery Network (CDN), а для зберігання відео – хмарні сервіси.

4. Надійність та безпека. Відеохостинг повинен працювати безперебійно та забезпечувати захист від зловмисників та хакерів. Для цього потрібно використовувати захист від DDoS атак, SSL шифрування та інші заходи безпеки.

Крім того, для створення відеохостингу необхідно враховувати питання безпеки і захисту даних, так як на платформі будуть зберігатися великі обсяги конфіденційної інформації. Також важливо розробляти зручний і привабливий інтерфейс користувача, щоб забезпечити зручність і комфорт при користуванні.

При створенні відеохостингу також важливо розробити і впровадити ефективні алгоритми сортування та пошуку відеоконтенту, що значно полегшить навігацію користувачів по платформі. Також потрібно враховувати технічні особливості і можливості різних пристроїв, на яких будуть переглядатися відео, щоб забезпечити максимальну сумісність і високу якість відтворення.

Висновки до розділу 2

Отже, створення відеохостингу – це складний і багатогранний процес, який вимагає багато ресурсів і технічних знань. Проте, якщо правильно підійти до розробки і врахувати всі необхідні фактори, можна створити високоякісну і популярну платформу для перегляду відеоконтенту.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ПЕРЕГЛЯДУ ВІДЕОКОНТЕНТУ

3.1 Архітектура застосунку

Застосунок розроблений за допомогою MySQL, Java, Spring Framework, JavaScript, Node.js, React.js, MongoDB та Telegram API, що дає купу переваг.

3.1.1 Список обраних мов програмування та їх переваги

Використання MySQL, Java, Spring Framework, JavaScript, Node.js, React.js, MongoDB та Telegram API для створення веб-додатка для перегляду відеоконтенту принесе такі переваги.

- MySQL забезпечує надійне зберігання і взаємодію зі структурованими даними, що дозволяє ефективно організовувати великий обсяг даних про відеоконтент.
- Java та Spring Framework дозволяють розробникам створювати гнучкі та ефективні веб-додатки. Spring Framework надає вбудовану підтримку для розробки RESTful API, спрощуючи взаємодію з відеоконтентом через HTTP-запити.
- Використання JavaScript та Node.js дозволяє розробникам створювати динамічні та інтерактивні веб-сторінки, використовувати єдину мову (JavaScript) на фронтенді, а також скористатися широким набором пакетів та бібліотек для прискорення розробки.
- React.js є потужною бібліотекою для розробки користувацького інтерфейсу, що дозволяє створювати динамічні та ефективні компоненти для перегляду відеоконтенту.

- MongoDB спрощують створення серверної частини додатка та зберігання даних про відеоконтент. MongoDB є гнучкою та масштабованою системою управління базами даних NoSQL.

- Використання Telegram API дозволяє забезпечити зручну взаємодію з користувачами через месенджер Telegram, зокрема для реєстрації користувачів.

Ці технології в поєднанні забезпечують потужність, гнучкість, ефективність та зручність у розробці веб-додатка для перегляду відеоконтенту.

3.1.2 Опис розробленого застосунку

Розроблений застосунок спрямований на перегляд відеоконтенту і у створенні рекомендацій для кожного користувача шляхом ретельного моніторингу всіх відео або фільмів, що переглядає користувач.

Перегляд відеоконтенту відбувається шляхом придбанням відео, після чого користувач може його переглянути через особистий кабінет.

Додаток розроблено з використанням архітектури мікросервісів. На рисунку 3.1 представлено зв'язок мікросервісів між собою.

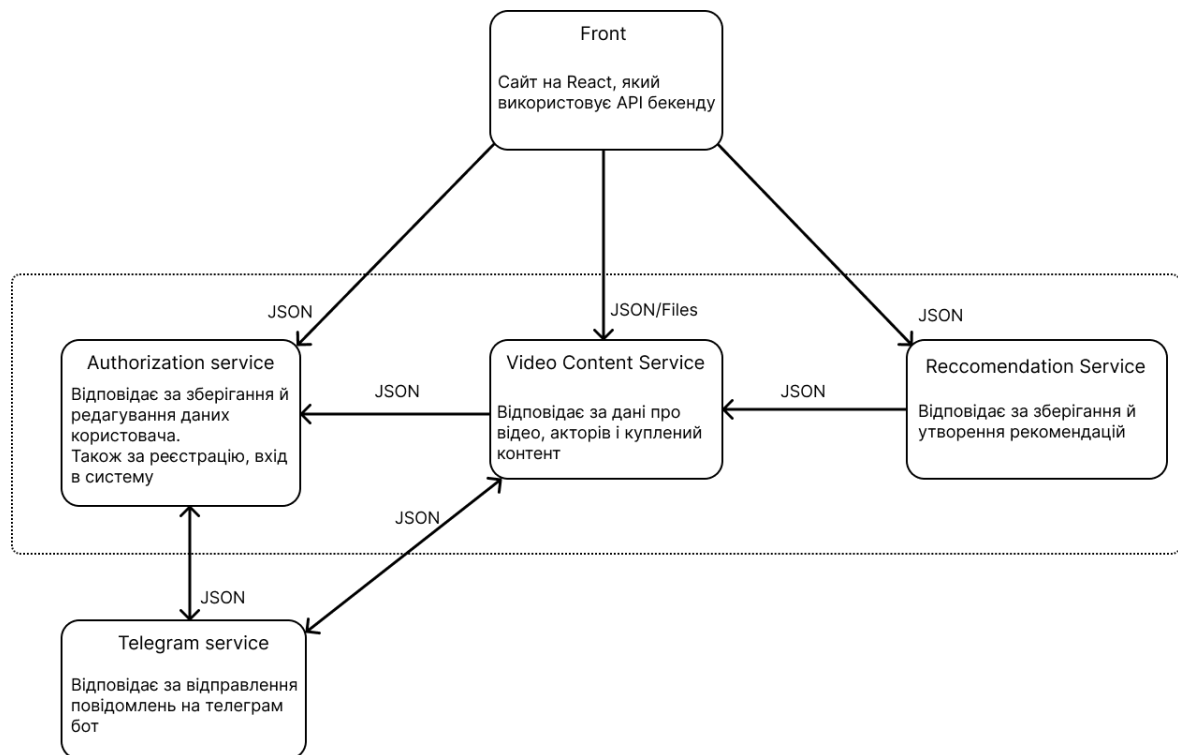


Рисунок 3.1 – діаграма взаємодії сервісів

На діаграмі продемонстровано взаємодію різних сервісів між собою. Також можна виділити, що мікросервіси використовують REST API для обміну даними.

3.1.2.1 Опис сервісу рекомендацій

Рекомендації засновуються на списку тих відео або фільмів, які переглядає користувач. Цей список формується на історії перегляду, та раз у 4 дні оновлюється. Якщо історія перегляду у користувача пуста, то рекомендації спочатку засновуються на популярних фільмах, у яких високий рейтинг. В іншому випадку, коли історія перегляду не порожня, за основу беруться жанри, які переглядав користувач, і виводить їх.

На рисунку 3.2 представлено блок-схему для створення рекомендацій для користувачів.

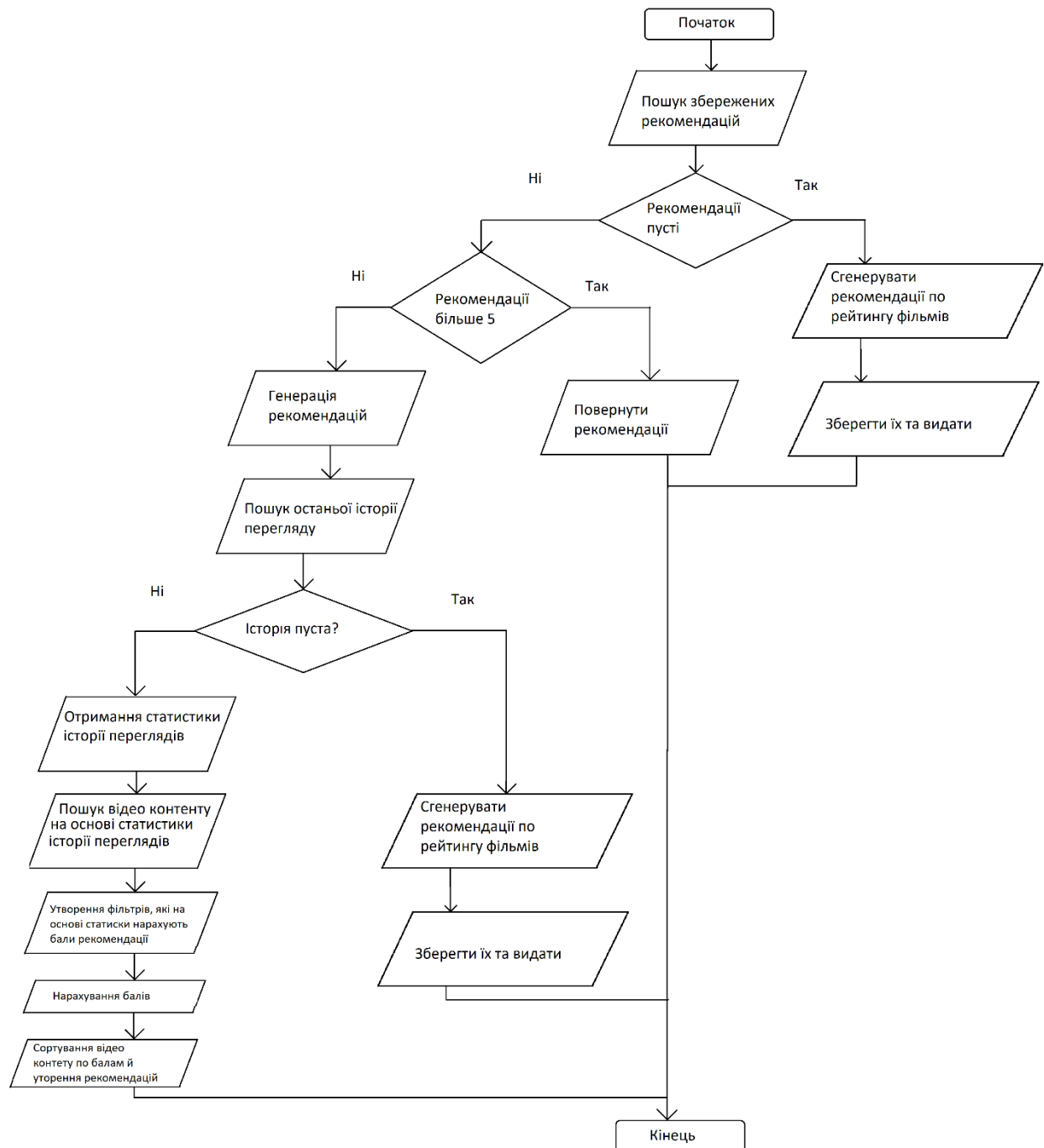


Рисунок 3.2 – Блок-схема алгоритму рекомендацій

На основі даної блок-схеми можна виділити наступний алгоритм рекомендації.

- За основу береться історія перегляду користувача.
- Розділення отриманого результату на кроці 1 для того, щоб перевірити, порожня історія чи ні.

- Якщо історія порожня, то рекомендації основуються на рейтингу відеоконтенту та по віковому рейтингу користувача.
- Якщо історія не порожня, то збирається статистика перегляду відеоконтенту за жанром, продюсер, тип фільму, скільки часу користувач переглядав відео та інше.
- Створюється контейнер, де зберігається ідентифікатор відеоконтенту та його бал.
- Для вибору відеоконтенту використовується статистика та вікові обмеження користувача. Застосовуються фабрики рекомендацій, які на основі цих даних генерують рекомендації. Через ці рекомендації користувач може переглядати весь відеоконтент, який відповідає його вимогам.
- Розраховувач розраховує бали щодо відеоконтенту, та завантажує до контейнеру з відеоконтентом.
- На основі балів сортується відеоконтент у контейнері за спаданням.

3.1.2.2 Опис сервісу Telegram

Сервіс Telegram спрямований на роботу з соціальною мережею Telegram для реєстрації користувача, отримання даних про рахунок користувача та його поповнення.

Алгоритм роботи Telegram бота:

- для реєстрації використовуються дані користувача (номер телефону), який зберігається у базі даних;
- для отримання даних про рахунок використовується номер телефону користувача, а саме утворюється HTTP запит до сервісу відеоконтенту;
- для поповнення користувач вводить номер картки, кінцеву дату картки, CVV код, назву картки та суму поповнення, після чого утворюється HTTP запит до сервісу відеоконтенту.

На рисунку 3.3 представлена таблиця з зареєстрованими користувачами.

id	phone_number	password	nickname	first_name	last_name	birth_date	active	selected_currency	user_state	email	role_table_id	date_registration
1	1111111111	\$2a\$10\$ZPPyFribbEYre9H0ZUObcC.Wn5xfl/.d...	user1	John	Doe	1990-01-01	1	USD	REGISTERED	user1@example.com	10	2023-01-20 17:05:55
2	2222222222	\$2a\$10\$DfXlvvghDwzWwvF38HmJPeQLQsbGF#b...	user2	Jane	Smith	1995-05-05	0	EUR	REGISTERED	user2@example.com	10	2023-02-20 17:05:55
3	3333333333	\$2a\$10\$eJZOIDSEGtIWzKZyUAe1H.GMbZByOn...	user3	Bob	Johnson	1985-10-10	1	GBP	REGISTERED	email@nure.ua	1	2023-03-15 16:51:55
4	4444444444	\$2a\$10\$MISgmv0.ukOSu.aJ.vHW0EHj3p1R...	TelegramBot921	Message	Bot	2023-05-01	1	USD	REGISTERED	telegram@bot.tg	20	2023-01-19 17:05:55
28	380660660066	\$2a\$10\$FfN54Lt4gn8V/tar4nZHL_QzMp2mdTr...	victoria921	Вікторія	Шеєченко	2000-01-01	1	UAH	REGISTERED		10	2023-05-08 19:54:19
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 3.3 – Таблиця зареєстрованих користувачів

3.1.2.3 Опис сервісу аутентифікації

Цей сервіс створений для роботи з персональними даними клієнта, а саме для реєстрації та авторизації на сайті. Без авторизації користувач не зможе користуватися усіма функціями розробленого додатку.

Алгоритм роботи сервісу аутентифікації наступний:

- користувач вводить свій номер телефону та пароль, який він використовував при реєстрації, та у базу даних надходить запит на пошук користувача за заданим номером;
- введений пароль порівнюється з зашифрованим у базі даних;
- для реєстрації користувач вводить свої дані у форму, після чого вони перевіряються на валідність і на те, щоб не було схожого вже зареєстрованого користувача;
- коли дані пройшли валідність, пароль шифрується і зберігається у базі даних зі статусом «реєструється»;
- після того, як користувач підтвердив у Telegram реєстрацію, статус змінюється на «зареєстрований».

3.1.3 Опис архітектури сервісів

Архітектура сервісів складається з трьох шарів, а саме робота з базою даних, бізнес логіка та робота с веб.

Робота з базою даних складається з таких класів:

- 1) entity – класи, які описують сутності бази даних, призначені для зберігання даних;
- 2) search criteria – класи, які описують параметри для пошуку entity у базі даних;
- 3) repository – призначені для отримання даних, і перетворення їх в entity, також перетворення entity в запит для зберігання або оновлення бази даних, а також на основі search criteria утворювати умови для пошуку у базі даних.

Бізнес логіка складається з таких класів:

- 1) Model – класи для збереження даних;
- 2) Service – клас який використовує repository та описує бізнес логіку.

Веб складається з таких класів:

- 1) DTO – data transfer object, класи для збереження даних та валідації;
- 2) Controller – призначений для отримання HTTP запитів, конвертації їх в DTO та передачі в класи services.

Між шарами знаходяться класи mapper, які конвертують класи одного шару в класи іншого.

3.2 Огляд бази даних розробленого застосунку

На рисунку 3.4 зображена ER діаграма сутностей відеоконтенту.

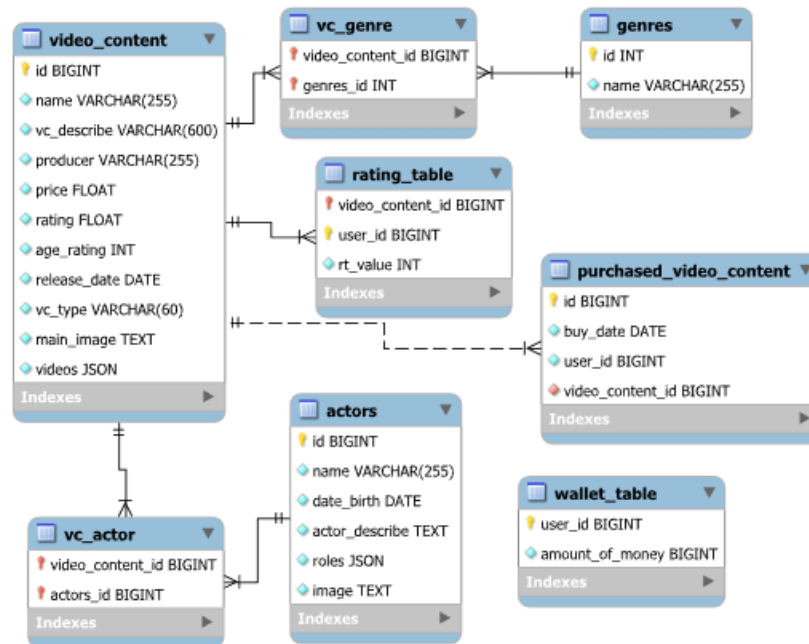


Рисунок 3.4 – ER діаграма сутностей відеоконтенту.

База даних відеоконтенту складається з таких таблиць:

- video_content – дані про контент;
- genres – дані про жанри до фільмів;
- vc-genre – таблиця зв'язку багато-до-багатьох;
- rating-table – рейтинг фільму від користувача;
- purchased_video_content – дані про куплений відеоконтент;
- actors – дані про акторів фільмів;
- vc_actors – таблиця зв'язку багато-до-багатьох;
- wallet – дані про рахунок користувача.

В сукупності всі ці таблиці створюють базу даних для сервісу відеоконтенту.

На рисунку 3.5 зображена ER діаграма сутностей бази даних авторизації.

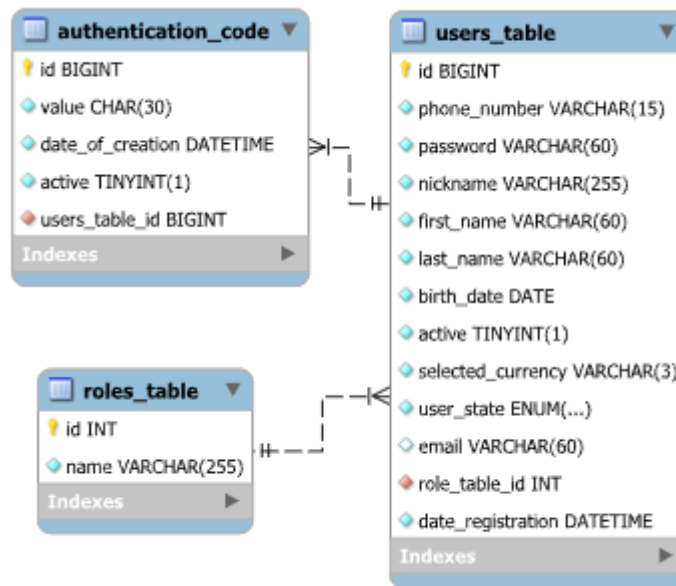


Рисунок 3.5 – ER діаграма авторизації

База даних авторизації складається з таких таблиць:

- users_table – дані про користувачів;
- roles_table – дані про ролі користувачів;
- authentication_code – дані про код який використовується при авторизації.

В сукупності ці таблиці складають базу даних сервісу авторизації.

База даних до сервісу Telegram складається з одної таблиці user, що зберігає у собі данні користувача для телеграм бота. На рисунку 3.6 зображена база даних сервісу Telegram.

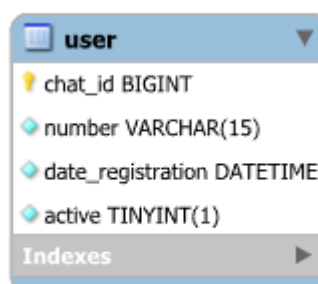


Рисунок 3.6 – ER діаграма бази даних Telegram

База даних до сервісу рекомендацій складається з двох колекцій:

- `vc_history` – зберігає дані про історію користувача;
- `recommendation` – зберігає рекомендації користувача за історією перегляду.

База даних для рекомендації створена за допомоги MongoDB. На рисунку 3.7 зображена ER діаграма бази даних сервісу рекомендації.

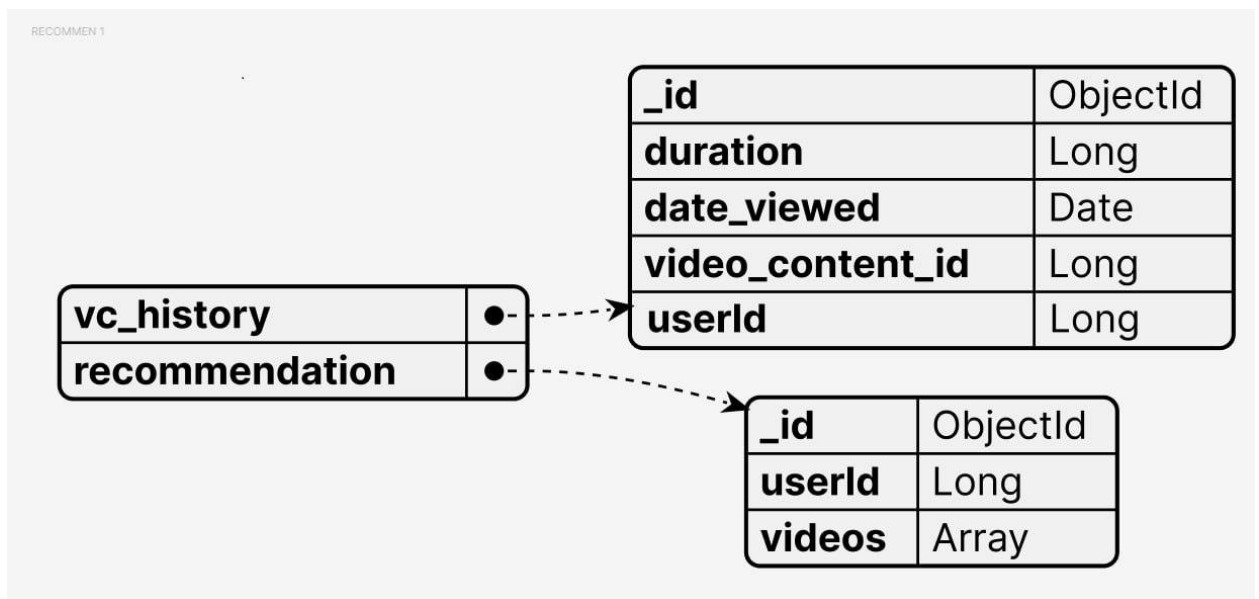


Рисунок 3.7 – ER діаграма бази даних сервісу рекомендації

3.3 Тестування розробленого застосунку

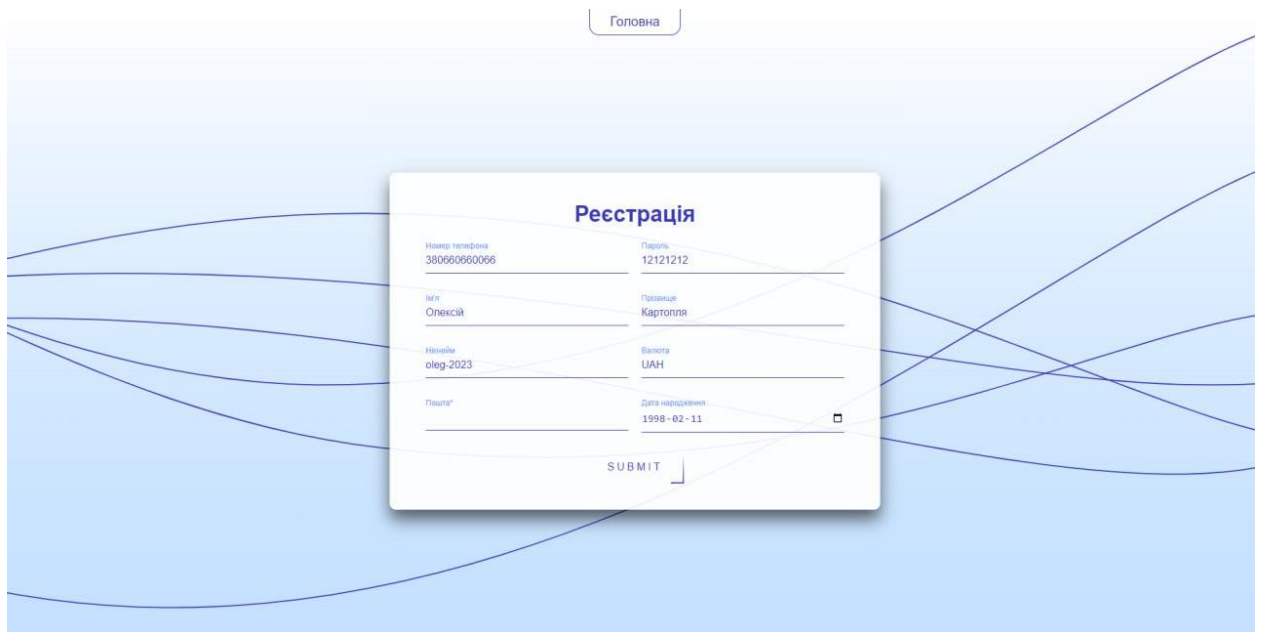
Буде протестовано такі речі:

- реєстрація – повний огляд процесу реєстрації у застосунку;
- вхід у систему – повний огляд процесу входу у систему застосунку;
- перегляд відеоконтенту – огляд основної роботи продукту.

3.3.1 Реєстрація

Перед тим як почати перегляд фільму, необхідно зареєструватись. Для цього користувач повинен заповнити реєстраційну форму і підтвердити свою особу через бота у Telegram.

На рисунку 3.8 зображено форму реєстрації



The image shows a registration form titled "Реєстрація" (Registration) centered on a blue background with abstract white lines. At the top center, there is a button labeled "Головна" (Home). The form itself is a white rectangle with the following fields and values:

Реєстрація	
Номер телефону 380660960066	Пароль 12121212
Ім'я Олексій	Політика Картоння
Емейл oleg-2023	Валюта UAH
Повість	Дата народження 1998-02-11
SUBMIT	

Рисунок 3.8 – Форма реєстрації

Після заповнення форми з'явиться вікно з інструкціями, яке міститиме докладний опис кроків для успішної реєстрації.

На рисунку 3.9 представлено інструкцію з подальшими кроками для успішної реєстрації.

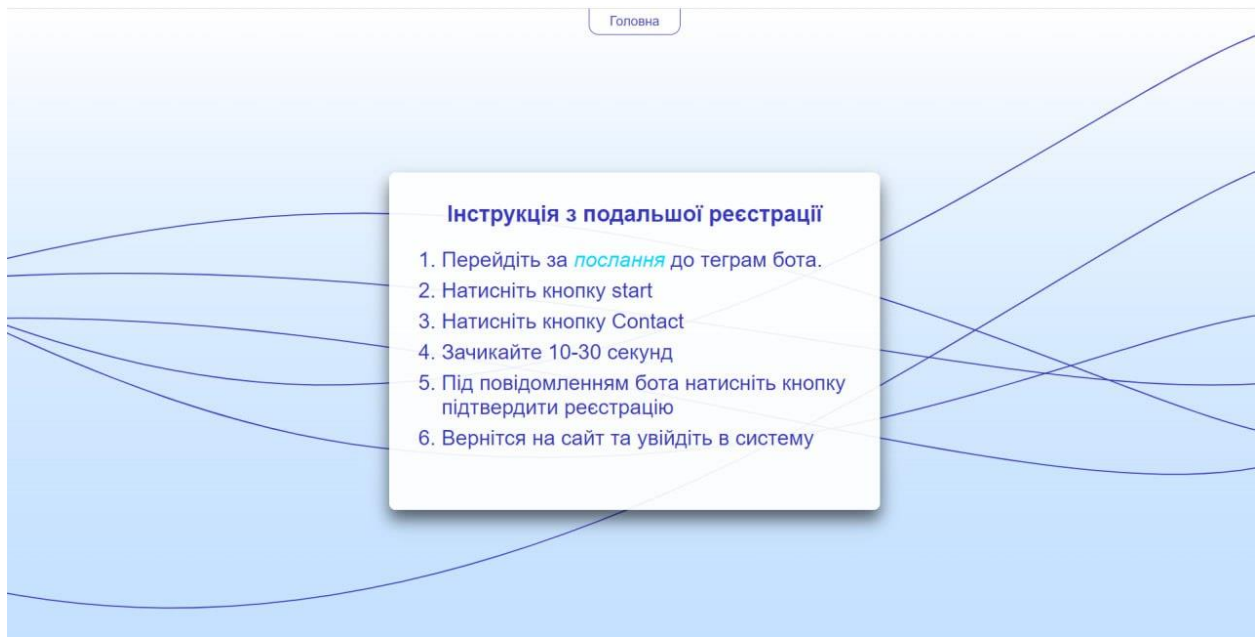


Рисунок 3.9 – Інструкція з подальшої реєстрації

Після переходу за посиланням користувачеві буде запропоновано увійти до Telegram. (рис.3.10).

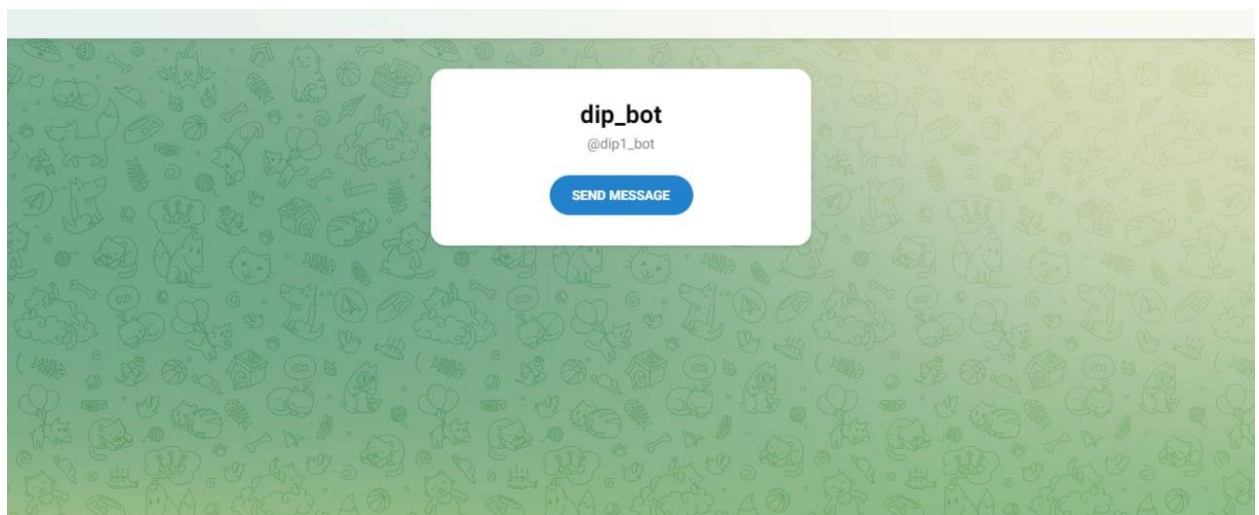


Рисунок 3.10 – Перехід у Telegram до сторінки бота

Після переходу до бота, вам потрібно натиснути кнопку «Start». Після цього бот запропонує вам відправити свій контакт (рис. 3.11). Після відправлення контакту, бот повідомить вас про успішну реєстрацію (рис. 3.12).

Після отримання повідомлення про успішну реєстрацію, бот попросить вас натиснути кнопку для підтвердження реєстрації (рис. 3.13).

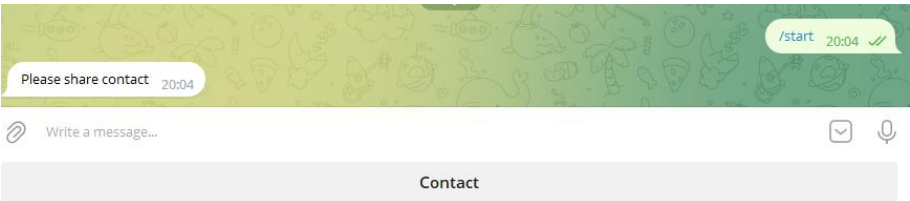


Рисунок 3.11 – відправлення контакту



Рисунок 3.12 – успішна реєстрація

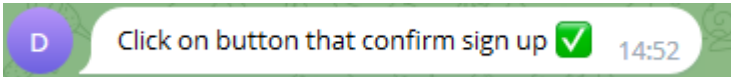


Рисунок 3.13 – підтвердження реєстрації користувачем

На рисунку 3.14 зображено таблицю у MySQL, де знаходяться зареєстровані користувачі.

id	phone_number	password	nickname	first_name	last_name	birth_date	active	selected_currency	user_state	email	role_table_id	date_registration
1	1111111111	\$2a\$10\$2PPyFnbEYre9H0ZUObC.Wn5xft/.d...	user1	John	Doe	1990-01-01	1	USD	REGISTERED	user1@example.com	10	2023-01-20 17:05:55
2	2222222222	\$2a\$10\$DfVvghDwzWwf38HtMjPeQLQabGF4b...	user2	Jane	Smith	1995-05-05	0	EUR	REGISTERED	user2@example.com	10	2023-02-20 17:05:55
3	3333333333	\$2a\$10\$eJZOIDSEGIW2cK2YUae1H.GMbZByOn...	user3	Bob	Johnson	1985-10-10	1	GBP	REGISTERED	email@nure.ua	1	2023-03-15 16:51:55
4	4444444444	\$2a\$10\$MISgnvm0.ukOSu.aJ.vHWOEHj3Yp1R...	TelegramBot921	Message	Bot	2023-05-01	1	USD	REGISTERED	telegram@bot.tg	20	2022-01-19 17:05:55
28	380660660066	\$2a\$10\$MFn54Lt4gn8V/tar4nZHL.QzMp2mdTr...	victoria921	Вікторія	Шевченко	2000-01-01	1	UAH	REGISTERED		10	2023-05-08 19:54:19

Рисунок 3.14 – Зареєстровані користувачі

3.3.2 Вхід у систему

Щоб увійти до системи, потрібно ввести номер телефону та пароль, після чого натиснути на кнопку submit.

На рисунку 3.15 зображено сторінку входу до системи

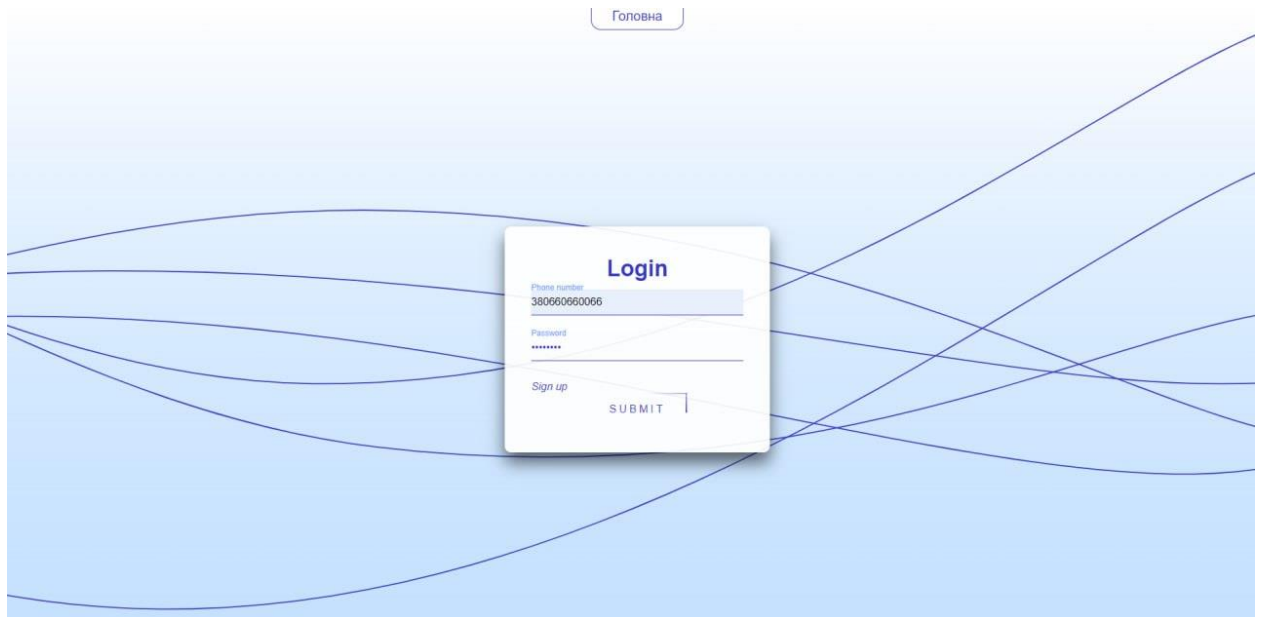


Рисунок 3.15 – Сторінка входу до системи

3.3.3 Перегляд відеоконтенту

Для того, щоб почати перегляд відеоконтенту потрібно обрати фільм із запропонованих на сайті (рис. 3.16)

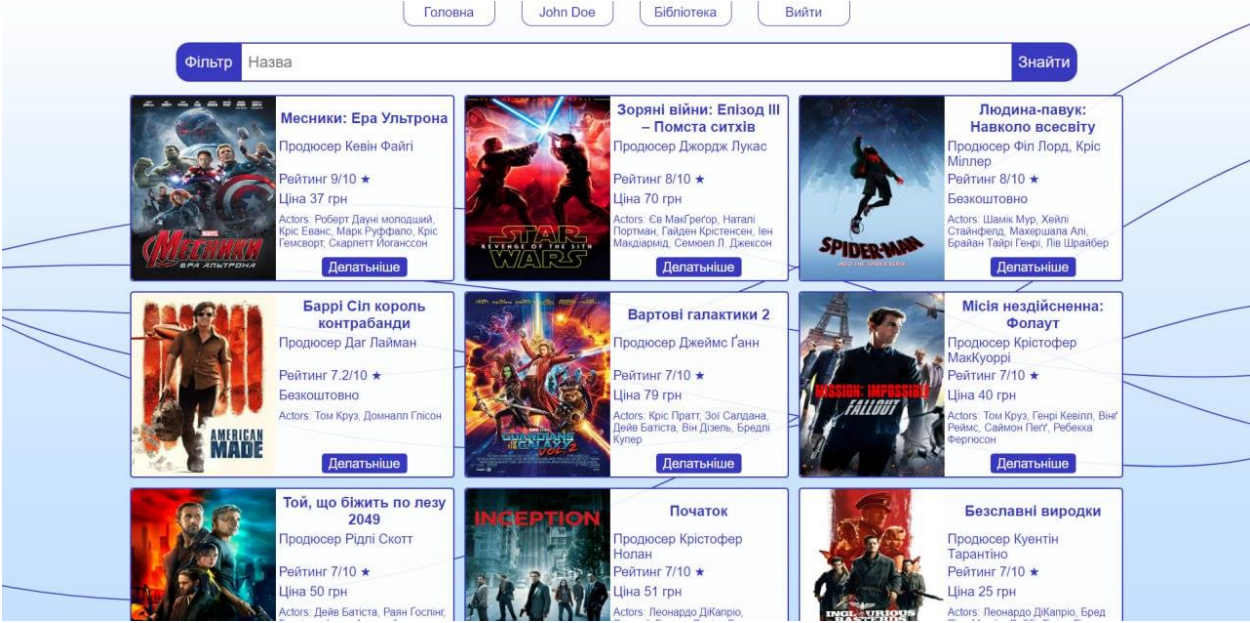


Рисунок 3.16 – Список фільмів

На рисунку 3.17 зображена сторінка повної інформації про фільм.

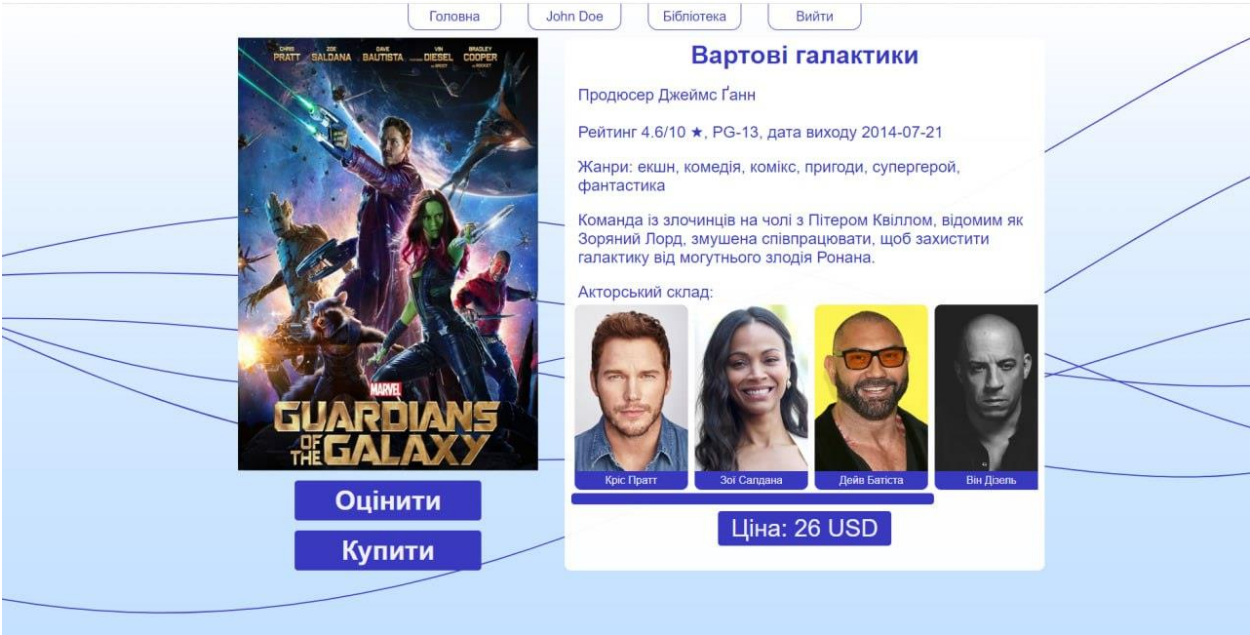


Рисунок 3.17 – Сторінка повної інформації про фільм

Щоб почати перегляд фільму спочатку його треба придбати (рис. 3.18). Для того, щоб фільм придбати, баланс у користувача повинен бути такий самий або більше ніж коштує сам фільм. Баланс можна переглянути у

власному кабінеті користувача (рис 3.19). Баланс можна поповнити через телеграм бота командою /put_money та заповнивши форму (рис. 3.20). Командою /wallet можна дізнатися поточний баланс (рис. 3.21).

Після придбання відео треба перейти до вкладки бібліотека (рис. 3.22), знайти придбаний фільм і натиснути кнопку перегляду. Коли користувач натиснув на кнопку перегляду, відкривається сторінка перегляду фільму (рис. 3.23)

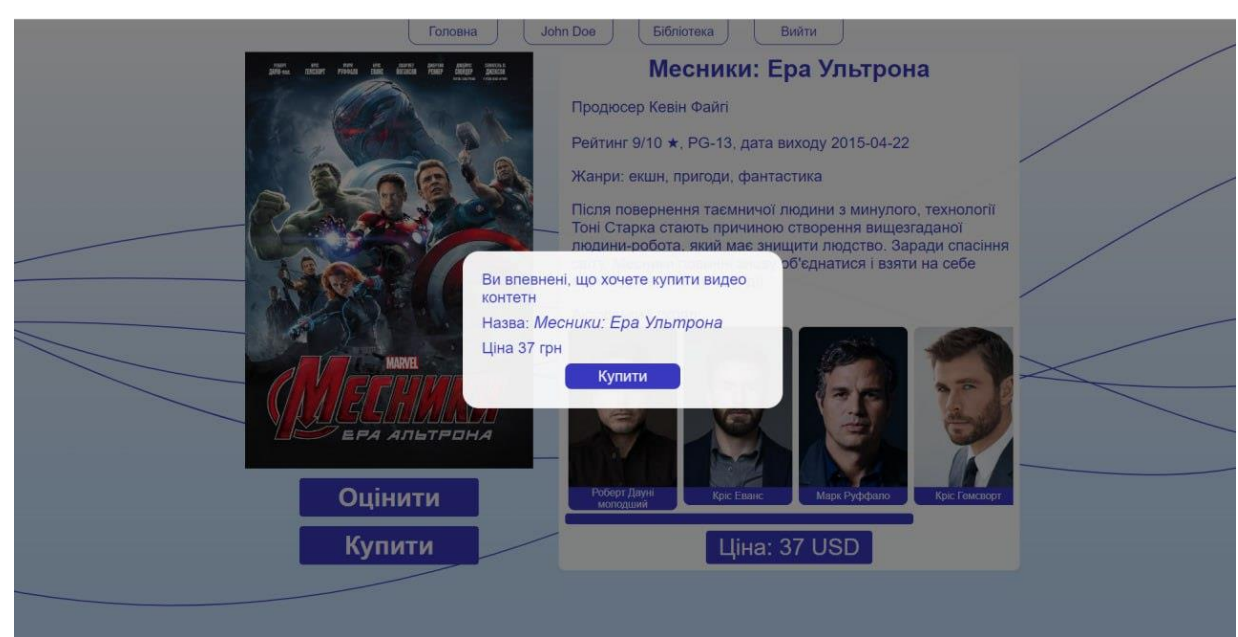


Рисунок 3.18 – Сторінка придбання фільму

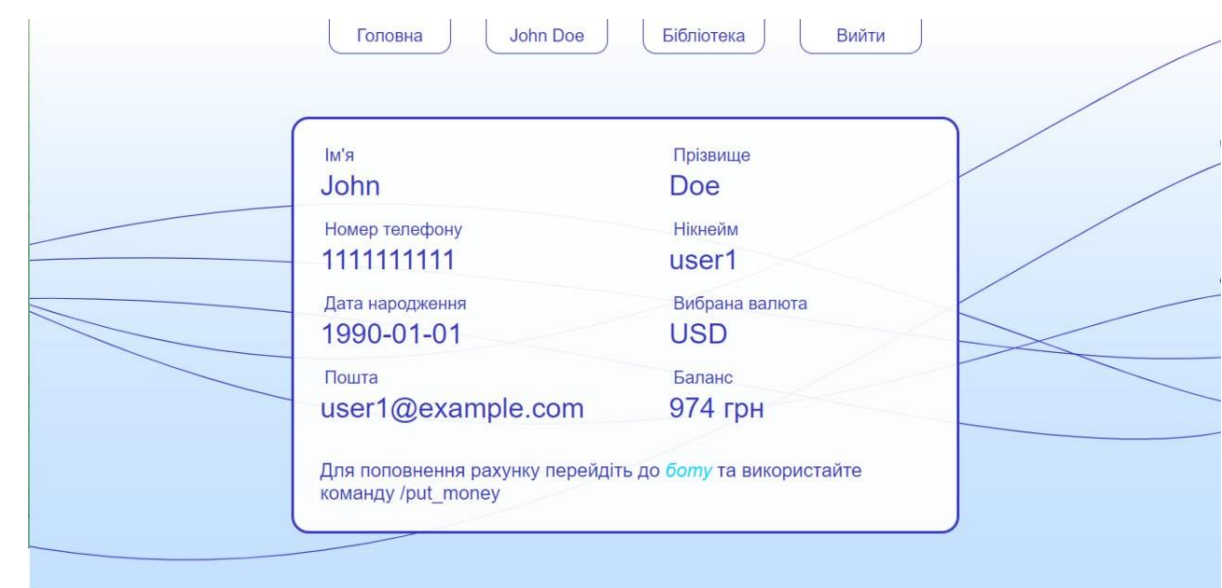


Рисунок 3.19 – Особистий кабінет



Рисунок 3.20 – Поповнення балансу

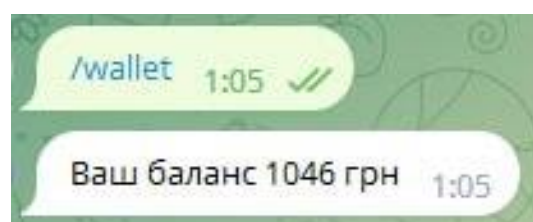


Рисунок 3.21 – Поточний баланс

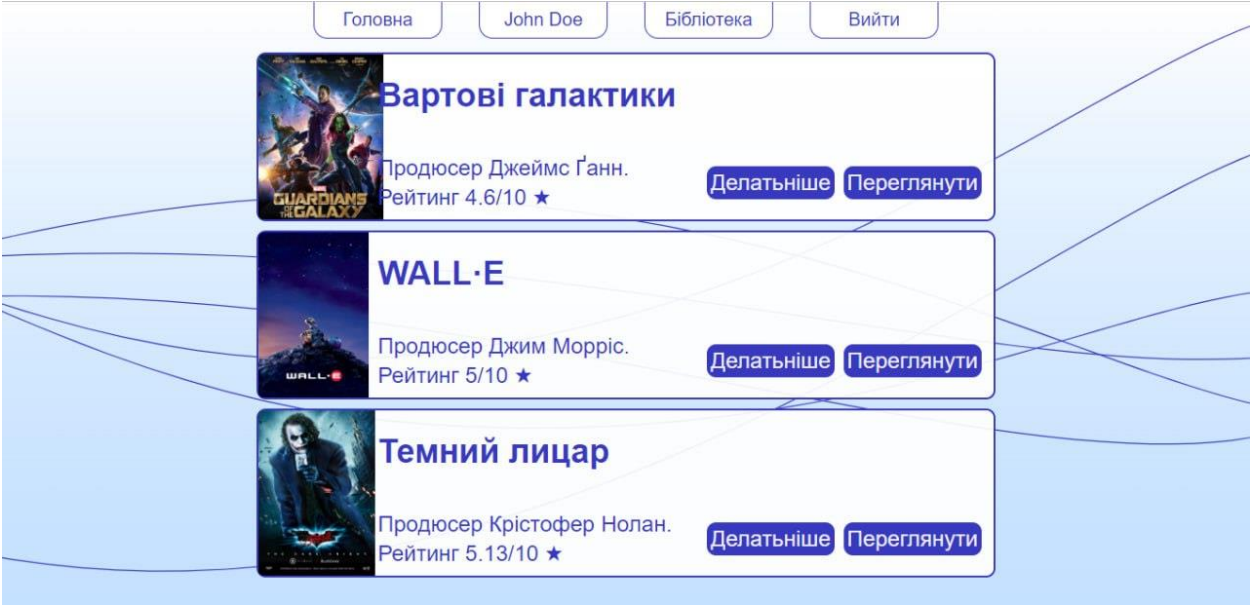


Рисунок 3.22 – Сторінка бібліотеки

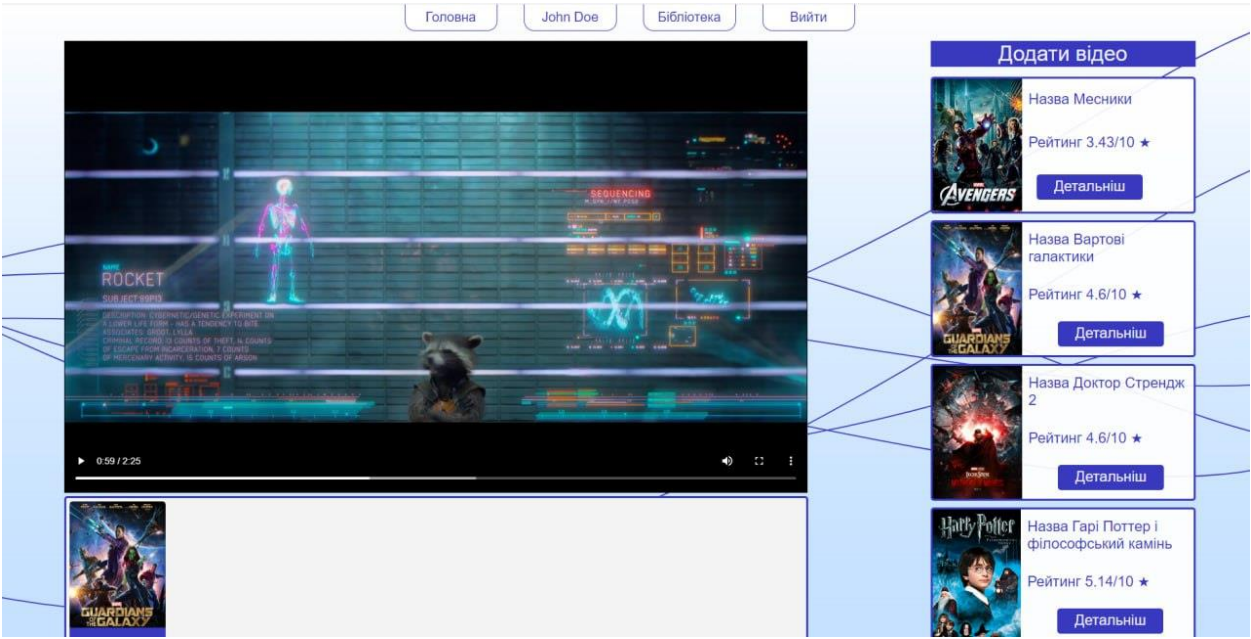


Рисунок 3.23 – Сторінка перегляду фільму

Щоб обрати відео яке користувач хоче переглянути, треба натиснути на кнопку додати відео або вибрати із списку рекомендацій під відео.

3.4 Експериментальні дослідження для визначення вагових коефіцієнтів в системі обробки запитів гетерогенної розподіленої бази даних.

У цьому експериментальному дослідженні ми зосередимося на системі обробки запитів у розподіленій базі даних з різноманітними компонентами. Наша основна мета полягає в тому, щоб встановити вагові коефіцієнти запитів, які будуть використовуватися в цій системі. Ми плануємо провести серію експериментів, які допоможуть нам оцінити продуктивність системи та знайти оптимальні налаштування, спрямовані на підвищення швидкодії та ефективності обробки запитів. Загальна структура експерименту описано в роботі [24].

Для проведення експериментів ми використаємо три таблиці `video\_content`, `purchased\_video\_content`, `actors`. Перша таблиця `video\_content` містить дані про відео контент, друга таблиця `purchased\_video\_content` про куплений відео контент клієнтом, третя таблиця `actors` містить дані про акторів.

Далі, ми проведемо чотири типи запитів до бази даних для роботи з таблицею `video\_content` під різним навантаженням.

1. SELECT-запит: ми виконаємо запит, щоб вибрати весь відео контент, де режисер «Джеймс Ганн».

```
SELECT * FROM video_content WHERE producer = 'Джеймс Ганн';
```

2. INSERT-запит: ми виконаємо запит, щоб зберегти фільм «Вартові галактики».

```
INSERT INTO `video_content` VALUES (41,'Вартові галактики', 'Команда із злочинців на чолі з Пітером Квіллом, відомим як Зоряний Лорд, змушена співпрацювати, щоб захистити галактику від могутнього злодія Ронана.','Джеймс Ганн',26,4.6,13,'2014-07-21','FILM','71K+P8n6sGL._AC_SL1024_.jpg','{"quality": "480", "duration":
```

420000, \"fileName\": \"Guardians_of_the_galaxy_UA.mp4\", \"language\": \"EN\"}]);

3. UPDATE-запит: ми виконаємо запит, щоб збільшити вартість фільму на 5 гривень.

UPDATE `video\_content` SET price = price + 5 WHERE id=41;

4. DELETE-запит: ми виконаємо запит, щоб видалити фільм «Вартові галактики».

DELETE FROM video_content WHERE id = 41;

Результати досліджень продемонстровано на таблиці 3.1.

Таблиця 3.1 – Дослідження запитів під навантаження для
`video\_content`

Номер експерименту	Select	Insert	Update	Delete
1	0,00562525	0,22426975	0,10293125	0,10592875
2	0,00056900	0,039068	0,03044200	0,03384975
3	0,00065075	0,03198025	0,03219325	0,03999000
4	0,00078125	0,0637865	0,04487300	0,03054350
5	0,00063300	0,05383225	0,04014350	0,03637125
6	0,00074250	0,05335225	0,03138275	0,03767325
7	0,00084450	0,05387925	0,04742950	0,05326075
8	0,00067700	0,0534255	0,03734175	0,02940675
9	0,00064975	0,05266475	0,03182900	0,03551350
10	0,00055925	0,02888275	0,06927725	0,07572025
11	0,00082725	0,02045525	0,25368925	0,22046775
12	0,00062725	0,075078	0,21115250	0,14735475
13	0,00075200	0,03964225	0,16270175	0,11068450
14	0,00067400	0,05346925	0,14554275	0,18287400
15	0,00120675	0,0196895	0,09847025	0,15119550

Продовження таблиці 3.1

16	0,00562525	0,019392	0,07112200	0,11921775
17	0,00078125	0,0206495	0,03201125	0,24304175
18	0,00067700	0,06451675	0,03142325	0,07936750
19	0,00082725	0,029999	0,05565100	0,03096450
20	0,00067400	0,019835	0,02893400	0,03195450
Середнє	0,001220213	0,050893388	0,077927063	0,089769013

На основі наданих даних можна зробити висновок про середнє значення часу виконання запитів для таблиці 'video_content':

1. для Select середній час виконання складає 0,001220213;
2. для Insert середній час виконання складає 0,050893388;
3. для Update середній час виконання складає 0,077927063;
4. для Delete середній час виконання складає 0,089769013.

Проведемо чотири типи запитів до бази даних для роботи з таблицею 'purchased_video_content' під різним навантаженням.

1. SELECT-запит: ми виконаємо запит, щоб вибрати весь куплений відео контент, де режисер дата покупки йде після 2023-05-04.

```
SELECT * FROM purchased_video_content WHERE buy_date > '2023-05-04';
```

2. INSERT-запит: ми виконаємо запит, щоб зберегти покупку гри з ідентифікатором 10, користувачем з ідентифікатором 19.

```
INSERT INTO `purchased_video_content` VALUES (10,'2023-04-24',19,10);
```

3. UPDATE-запит: ми виконаємо запит, щоб змінити у купленого контенту фільм на наступний.

```
UPDATE `purchased_video_content` set video_content_id = video_content_id + 1 where id = 41;
```

4. DELETE-запит: ми виконаємо запит, щоб видалити куплений відеоконтент з індикатором 19.

delete FROM purchased_video_content WHERE id = 19;

Результати досліджень продемонстровано на таблиці 3.2.

Таблиця 3.2 – Дослідження запитів під навантаження для сутності
купленого відео контенту

Номер експеременту	Select	Insert	Update	Delete
1	0,00095275	0,074484	0,06203550	0,078
2	0,00459975	0,09349	0,03191400	0,07939600
3	0,00052200	0,07375475	0,06407875	0,03183050
4	0,00044800	0,05394825	0,08029400	0,03055825
5	0,00314375	0,05319575	0,14937650	0,03233525
6	0,00054650	0,0588435	0,08076375	0,03118600
7	0,00046800	0,059289	0,16421425	0,05393400
8	0,00127850	0,14808725	0,14332175	0,05795400
9	0,00061725	0,11471475	0,09293150	0,02695700
10	0,00053550	0,1514185	0,05416025	0,03060525
11	0,00053800	0,153357	0,03059725	0,04273075
12	0,00049500	0,0582805	0,04399475	0,03093975
13	0,00056200	0,064715	0,04049650	0,03174300
14	0,00055350	0,0579795	0,04752750	0,03145700
15	0,00053150	0,06466125	0,08992475	0,03223575
16	0,00099050	0,09421075	0,03150550	0,03096200
17	0,00052975	0,0796355	0,02824000	0,03203725
18	0,00051700	0,087965	0,03169575	0,03124700
19	0,00050900	0,07412525	0,03094900	0,03228325
20	0,00127850	0,07647725	0,03213525	0,04156500
Середнє	0,000980838	0,084631638	0,066507825	0,03949785

На основі наданих даних можна зробити висновок про середнє значення часу виконання запитів для таблиці 'purchased_video_content':

1. для Select середній час виконання складає 0,000980838;
2. для Insert середній час виконання складає 0,084631638;
3. для Update середній час виконання складає 0,066507825;
4. для Delete середній час виконання складає 0,03949785.

Проведемо чотири типи запитів до бази даних для роботи з таблицею 'actors' під різним навантаженням.

1. SELECT-запит: ми виконаємо запит, щоб знайти актора з ім'ям «Кріс Пратт».

```
SELECT * FROM actors WHERE name = 'Кріс Пратт';
```

2. INSERT-запит: ми виконаємо запит, щоб зберегти актора.

```
INSERT INTO `actors` VALUES (415,'Бенедикт Камбербетч','1976-07-19',
```

```
'Бенедикт Тімоті Карлтон Камбербетч народився 19 липня 1976 року в Лондоні, Англія. Він є англійським актором, який отримав різноманітні нагороди, включаючи премію British Academy Television Award, премію Primetime Emmy та премію Laurence Olivier.','["Актор\","Режисер"]','benedict_timothy_carlton_cumberbatch.jpg');
```

3. UPDATE-запит: ми виконаємо запит, щоб дату народження актора.

```
UPDATE actors set date_birth= "1976-07-29" where id = 415;
```

4. DELETE-запит: ми виконаємо запит, щоб видалити актора за ідентифікатором 415.

```
delete FROM purchased_video_content WHERE id = 19;
```

Результати досліджень продемонстровано на таблиці 3.3.

Таблиця 3.3 – Дослідження запитів під навантаження для сутності акторів

Номер експеременту	Select	Insert	Update	Delete
1	0,00167200	0,06915750	0,00074325	0,06716225
2	0,00077425	0,03240300	0,00045500	0,05184825
3	0,00074800	0,03440200	0,00107200	0,03188900
4	0,00071400	0,03160300	0,00038675	0,03064000
5	0,00079525	0,03168775	0,00080800	0,03195850
6	0,00080375	0,03875400	0,00041575	0,03125750
7	0,00121325	0,09158450	0,00116225	0,03180525
8	0,00078800	0,19194825	0,00255125	0,03087700
9	0,00546750	0,07290550	0,00054650	0,03206175
10	0,00080450	0,05806400	0,00067625	0,04396750
11	0,00912825	0,07601675	0,00047650	0,03003175
12	0,00084125	0,04883400	0,00044600	0,03084125
13	0,00515600	0,05410425	0,00078300	0,03214900
14	0,00133225	0,04714875	0,00118300	0,06402250
15	0,00073200	0,05352450	0,00137125	0,10913400
16	0,00257925	0,05839900	0,00050675	0,00050825
17	0,00823950	0,05952725	0,00356525	0,00041475
18	0,00081175	0,03101425	0,00093400	0,00038350
19	0,00079425	0,10838500	0,00040400	0,00046750
20	0,00073025	0,05409725	0,00047425	0,00073800
Середнє	0,002206263	0,062178025	0,00094805	0,032607875

На основі наданих даних можна зробити висновок про середнє значення часу виконання запитів для таблиці ‘actors’:

1. для Select середній час виконання складає 0,002206263;

2. для Insert середній час виконання складає 0,062178025;
3. для Update середній час виконання складає 0,00094805;
4. для Delete середній час виконання складає 0,032607875.

Наступним кроком обчислимо середнє значення часу використання запитів за їх видами.

Для операції Select середній час використання для серед трьох таблиць:

$$t_s = \frac{0,001220213 + 0,000980838 + 0,002206263}{3} = 0,00146912$$

Для Insert середній час використання серед трьох таблиць:

$$t_i = \frac{0,050893388 + 0,084631638 + 0,062178025}{3} = 0,065901017$$

Для Update середній час використання серед трьох таблиць:

$$t_u = \frac{0,077927063 + 0,066507825 + 0,00094805}{3} = 0,048461$$

Для Delete середній час використання серед трьох таблиць:

$$t_d = \frac{0,089769013 + 0,03949785 + 0,032607875}{3} = 0,053958246$$

Звідси можна визначити коефіцієнт використання розподілених систем обробки даних для запитів всіх видів запитів.

Для типу Select обчислюємо за наступною формулою:

$$k_{select} = \frac{t_s}{t_{all}} = 0,0086526$$

Коефіцієнт використання розподілених систем даних для запитів типу Insert обчислюємо за формулою:

$$k_{insert} = \frac{t_i}{t_{all}} = 0,388134$$

Коефіцієнт використання розподілених систем даних для запитів типу Update обчислюємо за формулою:

$$k_{update} = \frac{t_u}{t_{all}} = 0,267749$$

Коефіцієнт використання розподільних систем даних для запитів типу Delete обчислюємо за формулою:

$$k_{delete} = \frac{t_d}{t_{all}} = 0,317795$$

Дані коефіцієнти можна використати для визначення середнього часу обробки запитів розподіленої системи з використанням гетерогенних баз даних. [24] Вони показують, яка доля часу, що відводиться на операції з базою даних, припадає на ті чи інші види запитів.

Висновки до розділу 3

У цьому розділі представлено та розроблено сервіс для перегляду відеоконтенту. Розроблено алгоритм рекомендацій для кожного користувача заснований на історії їх переглядів. Таким чином, це дозволяє користувачам переглядати фільми які він захоче або фільми з рекомендацій для нього. Також

були проведені експериментальні дослідження системи обробки запитів гетерогенної розподіленої бази даних, метою якого було визначення вагових коефіцієнтів.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

Даний розділ присвячено оцінюванню основних характеристик розробленого програмного продукту, який використовується для обліку медіаконтенту, такого як фільми та серіали.

У даному розділі описано різні варіанти реалізації програмного забезпечення та проведено їх дослідження для знаходження найбільш оптимальної з стратегії з економічної та функціональної точки зору. Дослідження проводилося із застосуванням апарату функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки інформаційної системи обліку медіаконтенту.

Оскільки рішення стосовно проектування та реалізації окремих компонентів системи, що розробляється, впливають на неї в цілому, то кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для пошуку, збереження і поширення медіаконтенту.

Визначимо технічні вимоги до програмного продукту:

- нормальне функціонування серверної частини програмного продукту на серверному апаратному забезпеченні;
- нормальне функціонування інтерфейсу користувача програмного продукту на персональних комп'ютерах зі встановленим браузером;
- зручність та зрозумілість інтерфейсу користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування системи;
- мінімальні витрати на розгортання та підтримку програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція, дослідження якої проводимо F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Змінюючи значення цих характеристик, можемо отримати різні варіанти реалізації продукту.

Беручи за основу дану функцію, виділимо наступні характеристики:

- F_1 – вибір стеку використовуваних технологій розробки;
- F_2 – вибір основної СУБД;
- F_3 – вибір архітектури веб-додатку.

Кожна з функцій можна реалізувати кількома способами:

Функція F_1 :

A) PHP/HTML/CSS+JavaScript;

Б) Java + Spring Framework + JavaScript + Node.js + React.js.

Функція F_2 :

A) SQLite;

Б) MySQL;

В) PostgreSQL.

Функція F_3 :

A) моноліт;

Б) багаторівнева архітектура;

В) мікросервіси.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

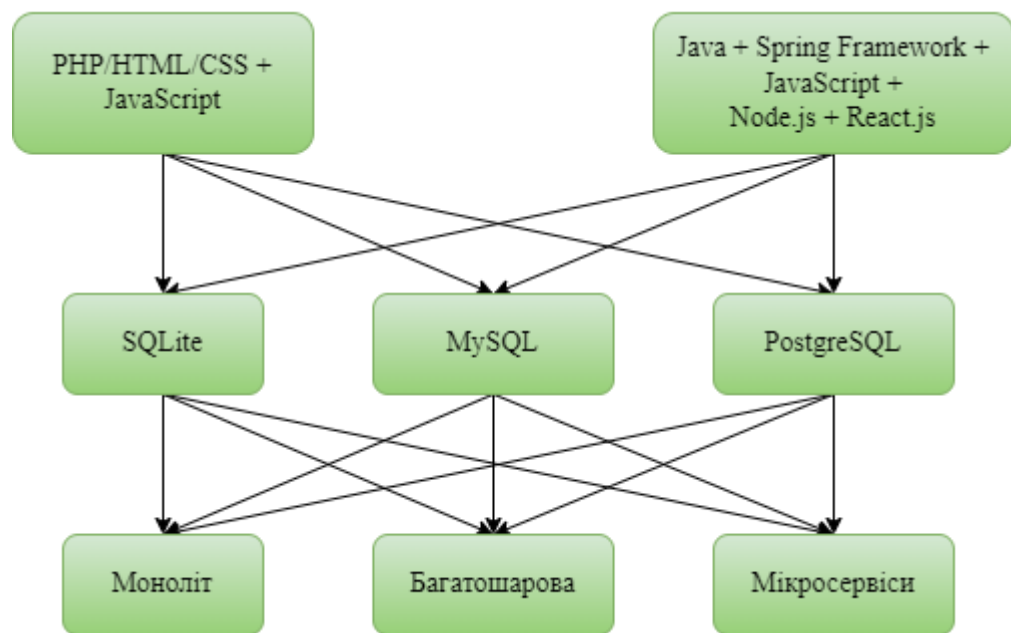


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Побудуємо для заданих функцій позитивно-негативну матрицю. Вона представлена в таблиці 4.1.

Таблиця 4.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	A	Простота у написанні взаємодії з БД та генерації веб-сторінок, швидкість роботи	Складна структурованість коду, менші можливості при реалізації інтерфейсу користувача, труднощі при реалізації складних архітектур, зменшення популярності, досить старий підхід
	B	Більші можливості при реалізації функціоналу та різних архітектур, проста структура коду, поширеність підходу, більш сучасний підхід	Потребує знань двох мов програмування та необхідних бібліотек
F_2	A	Швидке та просте налаштування, займає мало пам'яті, просте перенесення БД, безкоштовне використання	Погано підходить для великих об'ємів даних, можливі проблеми з безпекою даних, обмежений функціонал, складна масштабованість
	B	Популярність та простота у використанні, гарно працює з великими об'ємами даних, безпека, швидкість, масштабованість	Відсутність деякого функціоналу згідно зі стандартами мови SQL, деякі функції є платними
	B	Можливість розширення функціоналу, відкритість вихідного коду, повна відповідність стандарту мови SQL	Мала популярність та поширеність, збільшене використання пам'яті

Продовження таблиці 4.1

F_3	A	Швидкість розробки та розгортання, проста для розуміння, висока продуктивність	Складність або неможливість масштабування, складність у пошуку помилок, ненадійність при помилках, необхідність розгортати знову всю систему при виправленні помилок у роботі програми
	B	Простота та швидкість розробки та тестування	Складне масштабування, порівняно невелика продуктивність, складність розгортання, низька гнучкість архітектури
	B	Порівняно простий процес розробки, швидке та просте тестування та пошук і виправлення помилок, гарна масштабованість, просте розгортання, висока гнучкість	Порівняно невелика продуктивність роботи програми

У ході аналізу матриці (табл. 4.1) можна зробити висновок, що деякі варіанти реалізації функцій має сенс відкинути, бо вони не задовольняють поставленим перед програмним продуктом задачам.

Для функції F_1 .

Обираємо тільки варіант А, бо такий підхід є новішим та дозволяє реалізувати більш складний функціонал.

Для функції F_2 .

Варіант А не підходить для реалізації поставленої задачі через простоту та обмежений функціонал. Також у даного варіанту можуть бути проблеми з

безпекою, що неприпустимо для поставленої задачі. Варіанти Б та В є прийнятними для розгляду.

Для функції F_3 .

Архітектури А та Б мають досить багато недоліків і є застарілими підходами при розробці подібного програмного забезпечення. Варіант В є сучасним та популярним підходом з великою кількістю переваг, тому обираємо тільки його.

Отже, до розгляду маємо два варіанти реалізації програмного продукту:

1. $F_1A - F_2B - F_3B$;
2. $F_1A - F_2B - F_3B$.

Для оцінювання якості розглянутих функцій обрана система параметрів, яка описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

Визначимо основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня, керуючись даними, розглянутими вище.

Для характеристики ПП використаємо такий набір параметрів:

- $X1$ – швидкодія мови програмування та бібліотек;
- $X2$ – об'єм оперативної пам'яті, що використовується серверною частиною;
- $X3$ – потенційний об'єм програмного коду.

На основі вимог замовника та експлуатації обираються гірші, середні та кращі значення параметрів. Ці значення показано у таблиці 4.2.

Таблиця 4.2 - Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування та бібліотек	X1	оп/с	200	370	510
Об'єм оперативної пам'яті, що використовується серверною частиною	X2	Мб	4000	2900	1500
Потенційний об'єм програмного коду	X3	кількість рядків коду	20000	17000	10000

За даними таблиці 4.2 будуюмо графічні залежності бальної оцінки визначених параметрів від значень. Вони зображені на рис. 4.2 – рис. 4.4.

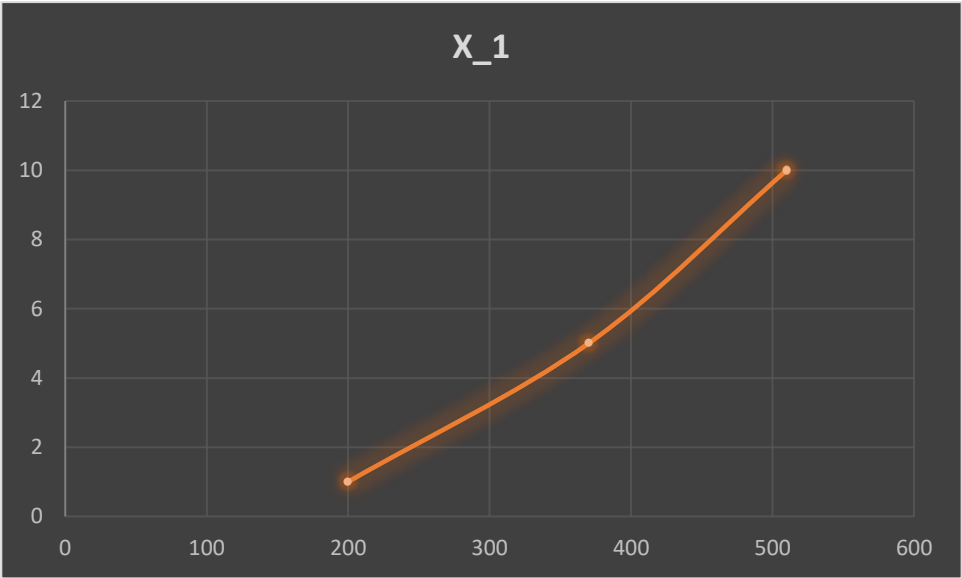


Рисунок 4.2 – X_1 , швидкодія мови програмування та бібліотек

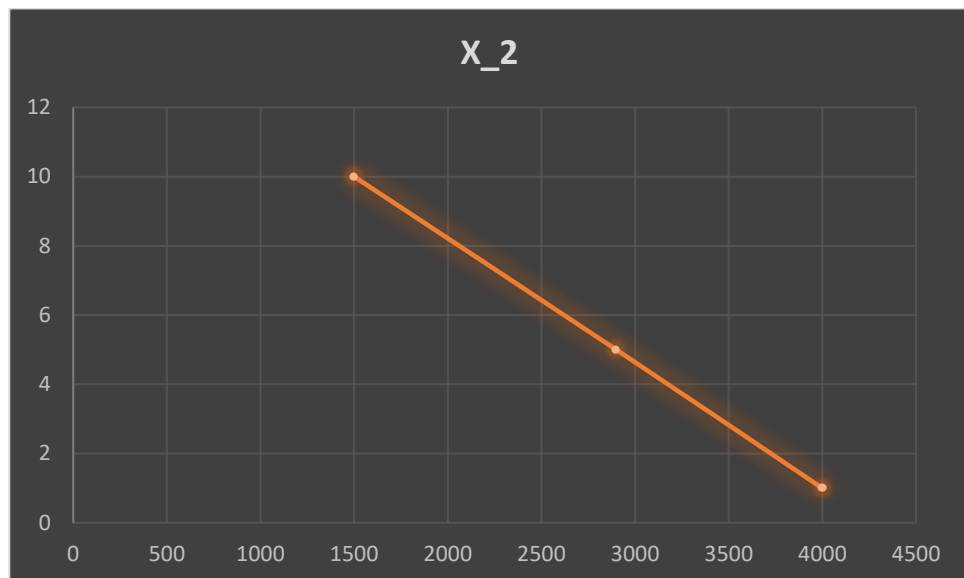


Рисунок 4.3 – X_2 , об'єм оперативної пам'яті, що використовується серверною частиною

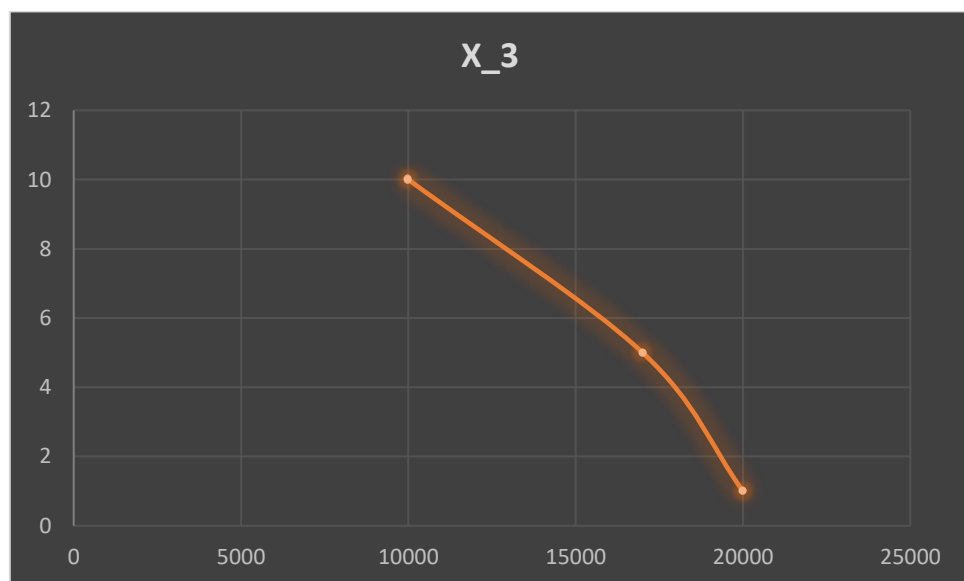


Рисунок 4.4 – X_3 , потенційний об'єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Проводиться обговорення та аналіз визначених параметрів, після якого експерти оцінюють ступінь важливості параметрів.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування та бібліотек	оп/с	1	1	1	1	1	1	1	7	-7	49
X2	Об'єм оперативної пам'яті, що використовується серверною частиною	Мб	2	3	2	3	2	2	3	17	3	9
X3	Потенційний об'єм програмного коду	Кількість рядків коду	3	2	3	2	3	3	2	18	4	16
	Разом		6	6	6	6	6	6	6	42	0	74

Параметри в таблиці визначаються за такими формулами.

а) Сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^N R_{ij} = \frac{Nn(n+1)}{2} = 42, \quad (4.1)$$

де N – число експертів, n – кількість параметрів.

б) Середня сума рангів:

$$T = \frac{1}{n} \sum_{i=1}^n R_i = 14 \quad (4.2)$$

в) Відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \quad (4.3)$$

Сума відхилень за всіма параметрам повинна дорівнювати 0.

г) Загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 74 \quad (4.4)$$

Обчислимо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 74}{7^2(3^3 - 3)} = 0,755 > W_k = 0,67. \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює за довідковою інформацією становить для даного виду проектів 0,67.

Скориставшись результатами ранжування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	<	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X3	<	>	<	>	<	<	>	<	0,5

Числове значення, що визначає ступінь переваги i -го параметра над j -тим, a_{ij} визначається за формулою:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \|a_{ij}\|$.

Для кожного параметра зробимо розрахунок вагомості K_{gi} за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^n b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^N a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{bi} = \frac{b'_i}{\sum_{i=1}^n b'_i}, \quad (4.9)$$

$$b'_i = \sum_{i=1}^N a_{ij} b_i \quad (4.10)$$

З таблиці 4.5 бачимо, що різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j			Ітер. 1		Ітер. 2		Ітер. 3		Ітер. 4	
	X1	X2	X3	b_i	K_{Bi}	b_i^1	K_{Bi}^1	b_i^2	K_{Bi}^2	b_i^3	K_{Bi}^3
X1	1	0,5	0,5	2	0,222	18	0,152	236	0,222	2124	0,222
X2	1,5	1	0,5	3	0,333	48	0,407	354	0,333	3186	0,333
X3	1,5	1,5	1	4	0,444	52	0,441	472	0,444	4248	0,444
Всього:				9	1	118	1	1062	1	9558	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^n K_{ei,j} B_{i,j}, \quad (4.11)$$

де n – кількість параметрів;

K_{ei} – коефіцієнт вагомості i -го параметра;

B_i – оцінка i -го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	A	X1	400	6	0,222	1,332
F2	Б	X2	2700	5,8	0,333	1,9314
	В	X2	3100	4,5	0,333	1,4985
F3	В	X3	14500	7	0,444	3,108

За даними з таблиці 4.6 за формулою:

$$K_K = K_{Ty}[F_{1k}] + K_{Ty}[F_{2k}] + \dots + K_{Ty}[F_{zk}], \quad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1,332 + 1,9314 + 3,108 = 6,3714;$$

$$K_{K2} = 1,332 + 1,4985 + 3,108 = 5,9385.$$

Як видно з розрахунків, кращим є варіант 1, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки програмного продукту

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Обидва варіанти розробки складаються з таких завдань.

1. Проектування бази даних з використанням MySQL Workbench.
2. Розробка серверної частини.
3. Розробка інтерфейсу користувача.

У другому варіанті розробки 1-ше завдання замінюється на наступне.

4. Проектування бази даних без візуального інтерфейсу на PostgreSQL

За ступенем новизни завдання 1 відноситься до групи Б, завдання 2 відноситься також до групи Б, завдання 3 – до групи В, завдання 4 – до групи Б. За рівнем складності завдання 1 один відноситься до групи 1, завдання 2 – до групи 2, завдання 3 – до групи 2, завдання 4 – до групи 1.

В завданні 1 та 4 використовується довідкова інформація, в завданні 2 і 3 використовується інформація у вигляді даних.

Беручи до уваги вищесказане проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється за формулою (4.13):

$$T_0 = T_P \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.13)$$

де T_P – трудомісткість розробки ПП;

K_{Π} – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни Б та групи складності алгоритму 1, трудомісткість дорівнює: $T_P = 64$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_{\Pi} = 1,021$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $K_{СК} = 1$. Оскільки при

розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{CT} = 0,6$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 64 \cdot 1,021 \cdot 0,6 = 39,2064 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм другої групи складності, степінь новизни Б), тобто $T_p = 27$ людино-днів, $K_{\Pi} = 0,9$, $K_{СК} = 1$, $K_{CT} = 0,8$:

$$T_2 = 27 \cdot 0,9 \cdot 0,8 = 19,44 \text{ людино-днів.}$$

Для третього завдання (використовується алгоритм третьої групи складності, степінь новизни В), тобто $T_p = 19$ людино-днів, $K_{\Pi} = 0,6$, $K_{СК} = 1$, $K_{CT} = 0,8$:

$$T_3 = 19 \cdot 0,6 \cdot 0,8 = 9,12 \text{ людино-днів.}$$

Для четвертого завдання (використовується алгоритм першої групи складності, степінь новизни Б), тобто $T_p = 64$ людино-днів, $K_{\Pi} = 1,021$, $K_{СК} = 1$, $K_{CT} = 0,9$:

$$T_4 = 64 \cdot 0,6 \cdot 0,9 = 58,8096 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (39,2064 + 19,44 + 9,12) \cdot 8 = 542,13 \text{ людино-годин.}$$

$$T_{II} = (58,8096 + 19,44 + 9,12) \cdot 8 = 698,96 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

У розробці продукту беруть участь спеціаліст з баз даних з окладом 13500 грн., розробник серверної частини з окладом 27000 грн. та розробник програмного інтерфейсу з окладом 18500 грн. Визначимо середню зарплату за годину за формулою:

$$C_{\text{ч}} = \frac{M}{T_m \cdot t} \text{ грн./год,} \quad (4.14)$$

де M – місячний оклад працівників;

T_m – кількість робочих днів на місяць;

t – кількість робочих годин в день.

$$C_{\text{ч}} = \frac{13500 + 27000 + 18500}{3 \cdot 21 \cdot 8} = 117,06 \text{ грн./год} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за наступною формулою:

$$C_{\text{зп}} = C_{\text{ч}} \cdot T_i \cdot K_d, \quad (4.16)$$

де $C_{\text{ч}}$ – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

K_d – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$\text{I. } C_{\text{зп}} = 117,06 \cdot 542,13 \cdot 1,2 = 76154,08 \text{ грн.}$$

$$\text{II. } C_{\text{зп}} = 117,06 \cdot 698,96 \cdot 1,2 = 98184,31 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становить 22%, тому маємо:

$$\text{I. } C_{\text{ВІД}} = C_{\text{ЗП}} \cdot 0,22 = 76154,08 \cdot 0,22 = 16753,9 \text{ грн.}$$

$$\text{II. } C_{\text{ВІД}} = C_{\text{ЗП}} \cdot 0,22 = 98184,31 \cdot 0,22 = 21600,55 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години (C_M).

Всього використовується три ПК, кожен з яких обслуговується відповідним програмістом з окладом 7000 грн., з коефіцієнтом зайнятості 0,2. Тоді для однієї машини отримаємо:

$$C_G = 12 \cdot M \cdot K_3 = 12 \cdot 7000 \cdot 0,2 = 16800 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{\text{ЗП}} = C_G \cdot (1 + K_3) = 16800 \cdot (1 + 0,2) = 20160 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{\text{ВІД}} = C_{\text{ЗП}} \cdot 0,22 = 20160 \cdot 0,22 = 4435,2 \text{ грн.}$$

Амортизаційні відрахування для одного ПК розраховуємо при амортизації 25% та вартості ЕОМ – 15000 грн.

$$C_A = K_{\text{ТМ}} \cdot K_A \cdot Ц_{\text{ПР}} = 1,4 \cdot 0,25 \cdot 15000 = 5250 \text{ грн.,}$$

де $K_{\text{ТМ}}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

K_A – річна норма амортизації;

$\text{Ц}_{\text{ПР}}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$\text{С}_{\text{Р}} = \text{К}_{\text{ТМ}} \cdot \text{Ц}_{\text{ПР}} \cdot \text{К}_{\text{Р}} = 1,4 \cdot 15000 \cdot 0,08 = 1680 \text{ грн.},$$

де $\text{К}_{\text{Р}}$ – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$\begin{aligned} \text{Т}_{\text{ЕФ}} &= (\text{Д}_{\text{К}} - \text{Д}_{\text{В}} - \text{Д}_{\text{С}} - \text{Д}_{\text{Р}}) \cdot \text{t}_{\text{З}} \cdot \text{К}_{\text{В}} = (365 - 104 - 12 - 16) \cdot 8 \cdot 0,35 = \\ &= 652,4 \text{ години}, \end{aligned}$$

де $\text{Д}_{\text{К}}$ – календарна кількість днів у році;

$\text{Д}_{\text{В}}$, $\text{Д}_{\text{С}}$ – відповідно кількість вихідних та святкових днів;

$\text{Д}_{\text{Р}}$ – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

$\text{К}_{\text{В}}$ – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$\text{С}_{\text{ЕЛ}} = \text{Т}_{\text{ЕФ}} \cdot \text{N}_{\text{С}} \cdot \text{К}_{\text{З}} \cdot \text{Ц}_{\text{ЕН}} = 652,4 \cdot 0,2 \cdot 0,3 \cdot 4,87 = 190,63 \text{ грн.},$$

де $\text{N}_{\text{С}}$ – середньо-споживча потужність приладу;

$\text{К}_{\text{З}}$ – коефіцієнтом зайнятості приладу;

$\text{Ц}_{\text{ЕН}}$ – тариф за 1 кВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$\text{С}_{\text{Н}} = \text{Ц}_{\text{ПР}} \cdot 0,67 = 15000 \cdot 0,67 = 10050 \text{ грн.}$$

Тоді, річні експлуатаційні витрати на один ПК будуть:

$$C_{\text{ЕКС}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{А}} + C_{\text{Р}} + C_{\text{ЕЛ}} + C_{\text{Н}}, \quad (4.17)$$

$$C_{\text{ЕКС}} = 20160 + 4435,2 + 5250 + 1680 + 190,63 + 10050 = 41765,83 \text{ грн.}$$

Тоді для 3-х ПК маємо:

$$C_{\text{ЕКСЗ}} = 3 \cdot C_{\text{ЕКС}} = 3 \cdot 41765,83 = 125297,49 \text{ грн.}$$

Собівартість однієї машино-години 3-х ЕОМ дорівнюватиме:

$$C_{\text{М-Г}} = C_{\text{ЕКСЗ}} / T_{\text{ЕФ}} = 125297,49 / 652,4 = 192,06 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_{\text{М}} = C_{\text{М-Г}} \cdot T, \quad (4.18)$$

$$\text{I. } C_{\text{М}} = 192,06 \cdot 542,13 = 104121,49 \text{ грн.}$$

$$\text{II. } C_{\text{М}} = 192,06 \cdot 698,96 = 134242,26 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_{\text{Н}} = C_{\text{ЗП}} \cdot 0,67, \quad (4.19)$$

$$\text{I. } C_{\text{Н}} = 76154,08 \cdot 0,67 = 51023,23 \text{ грн.}$$

$$\text{II. } C_{\text{Н}} = 98184,31 \cdot 0,67 = 65783,49 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{\text{ПП}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{М}} + C_{\text{Н}}, \quad (4.20)$$

$$\text{I. } C_{\text{ПП}} = 76154,08 + 16753,9 + 104121,49 + 51023,23 = 248052,7 \text{ грн.}$$

$$\text{II. } C_{\text{ПП}} = 98184,31 + 21600,55 + 134242,26 + 65783,49 = 319810,61 \text{ грн.}$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{\text{К}j} / C_{\text{Ф}j}, \quad (4.21)$$

$$K_{\text{ТЕР}1} = 6,3714 / 248052,7 = 2,569 \cdot 10^{-5},$$

$$K_{\text{ТЕР}2} = 5,9385 / 319810,61 = 1,857 \cdot 10^{-5}.$$

Отже, найбільш ефективним є перший варіант реалізації ПП з коефіцієнтом техніко-економічного рівня $K_{\text{ТЕР}1} = 2,569 \cdot 10^{-5}$. Таким чином в результаті виконання функціонально-вартісного аналізу продукту, що розробляється бачимо, що з двох альтернатив, що були відібрані для аналізу найбільш оптимальною є перша.

Цей варіант реалізації ПП передбачає такі характеристики:

- вибір стеку використовуваних технологій розробки – Java + Spring Framework + JavaScript + Node.js + React.js;
- вибір основної СУБД – MySQL;

- вибір архітектури веб-додатку – мікросервіси;

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми, швидкодію готового продукту та швидкий, широкий та доступний функціонал для роботи. А також він є оптимальним з економічної точки зору.

Висновки до розділу 4

У даному розділі було проведено функціонально-вартісний аналіз програмного продукту, що розробляється. Було виділено кілька альтернативних варіантів реалізації програмного продукту.

В ході виконання аналізу було визначено та проведено оцінку основних функцій продукту, знайдено параметри які його характеризують. Також для кожної альтернативи реалізації продукту проведено розрахунок загальної вартості програмного продукту, показників рівня якості варіантів реалізації основних функцій ПП та коефіцієнтів техніко-економічного рівня.

У результаті було обрано оптимальний варіант розробки програмного продукту на основі значення коефіцієнтів техніко-економічного рівня.

ВИСНОВКИ

У даній дипломній роботі була розглянута тема «Інформаційна система обліку медіаконтенту на основі баз даних». В ході дослідження було проведено огляд та аналіз існуючих баз даних в контексті створення інформаційних систем. Була досліджена архітектурна побудова сучасних інформаційних систем, включаючи архітектуру файл-сервера та архітектуру клієнт-сервер. Також було проведене теоретичне порівняння реляційної та об'єктно-орієнтованої моделі даних.

Були розглянуті теоретичні відомості інформаційної системи веб-сервісу відеохостингу. Була надана характеристика потенційної аудиторії клієнтів додатку, визначено ціль і призначення проекту. Описано процес розробки інформаційної системи відеохостингу, включаючи опис структури програми та технологій для створення веб сервісу, таких як SQL, MySQL, Java, Spring Framework, JavaScript та Telegram API.

Було проведено проектування та розробку застосунку для перегляду відеоконтенту, розроблена архітектура застосунку та надано опис розробленого застосунку, включаючи опис сервісу рекомендацій, сервісу Telegram та сервісу аутентифікації. Також було надано опис архітектури сервісів та проведено тестування розробленого застосунку. В рамках дипломної роботи були проведені експериментальні дослідження системи обробки запитів гетерогенної розподіленої бази даних для визначення вагових коефіцієнтів.

Загальними висновками роботи є те, що розроблена інформаційна система обліку медіаконтенту на основі баз даних є ефективним інструментом для управління медіаконтентом. Проведені експериментальні дослідження підтвердили ефективність системи обробки запитів гетерогенної розподіленої бази даних.

Отже, інформаційна система обліку медіаконтенту на основі баз даних є значним кроком у поліпшенні управління медіаконтентом і забезпеченні зручного доступу до медіафайлів. Результати дослідження можуть бути використані для подальшого розвитку і вдосконалення подібних систем у медіаіндустрії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. В. В. Тарасюк, Р. П. Судик. Інформаційно-пошукові системи. Енциклопедія Сучасної України: онлайн-версія / редкол.: І. М. Дзюба та ін.; НАН України, НТШ. Київ: Інститут енциклопедичних досліджень НАН України, 2011. URL: <https://esu.com.ua/article-12483> (дата звернення 20.05.23р).
2. Завадський І. О. Основи баз даних. К.: Видавець ІО Завадський, 2011.
3. Коломейчук В. В. Розробка та дослідження бази даних для систем обробки статистичної інформації. Математичні машини и системи, 2009, 1(4), 89-95.
4. Корченко О.Г., Дрейс Ю.О. Охорона конфіденційної інформації підприємства: навчальний посібник – Житомир: ЖВІ НАУ, 2011, 172 с.
5. Зінов'єва І. С., Артемчук В. О. Сучасні підходи до подальшої еволюції концепції баз даних. In III International scientific and practical conference «Dynamics of the development of world science», Canada, 2019 (pp. 34-44).
6. СУБД Postgres Pro. URL: <https://postgrespro.com/> (дата звернення 20.05.23р).
7. MySQL. URL: <https://www.mysql.com/> (дата звернення 20.05.23р).
8. Пасічник В. В., Резніченко В.А. Організація баз даних та знань.: Видавнича група BHV, 2006. – 384 с.
9. Пономаренко В. С., Журавльова І. В., Латишева І. Л. Інформаційні системи в управлінні персоналом. Навчальний посібник – Харків: Вид. ХНЕУ, 2008. – 336 с. (Укр. мов.) URL: <http://www.repository.hneu.edu.ua/jspui/handle/123456789/3059> (дата звернення 20.05.23р).
10. Сахно Ю. М. Створення бази даних для розробки проєктів, 2005.

11. Шаров С. В., Петровський В. В. Огляд нереляційних баз даних. Всеукраїнська Internet-конференція «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення», 2015, (14), 16-18.
12. About SQLite. URL: <https://www.sqlite.org/index.html> (дата звернення 20.05.23р).
13. Migler A., Dekhtyar A. Mapping the SQL learning process in introductory database courses. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education, 2020, February (pp. 619-625).
14. Mistry R., Misner S. Introducing Microsoft SQL Server 2008 R2. Microsoft press, 2010.
15. Mistry R., Misner S. Introducing Microsoft SQL Server 2014. Microsoft Press, 2014.
16. Mitrovic A. Learning SQL with a computerized tutor. In Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education, 1998, March (pp. 307-311).
17. Official MySQL Database Site. URL: <https://www.mysql.com/> (дата звернення 20.05.23р).
18. Beaulieu A. Learning SQL: master SQL fundamentals. " O'Reilly Media, Inc.", 2009.
19. Berg K. L., Seymour T., Goel R. History of databases. International Journal of Management & Information Systems (IJMIS), 2013, 17(1), 29-36.
20. Date C. J. A Guide to the SQL Standard. AddisonWesley Longman Publishing Co., Inc., 1989 .
21. Donahoo M. J., Speegle G. D. SQL: Practical guide for developers. Elsevier., 2010.
22. Emerson S. L., Darnovsky M., Bowman J. The practical SQL handbook: using structured query language. Addison-Wesley Longman Publishing Co., Inc., 1989.
23. Pratt P. J., Last M. Z. A guide to SQL. Cengage Learning, 2017.

24. Корнага Я. І. Моделі та методи організації та управління гетерогенними розподіленими базами даних з динамічною структурою на основі мережецентричного підходу, 2020. URL: https://dut.edu.ua/uploads/p_86_14893265.pdf (дата звернення 20.05.23р).

ДОДАТОК А

А.1 Створення баз даних

База даних video_content_service_db

```

DROP DATABASE IF EXISTS video_content_service_db;
CREATE DATABASE video_content_service_db;
USE video_content_service_db;

CREATE TABLE actors
(
    id            bigint            NOT NULL AUTO_INCREMENT,
    name          varchar(255) NOT NULL,
    date_birth    date             NOT NULL,
    actor_describe text            NOT NULL,
    roles         json             NOT NULL,
    image         text            NOT NULL,
    CONSTRAINT actors_pk PRIMARY KEY (id)
);

CREATE TABLE genres
(
    id    int            NOT NULL AUTO_INCREMENT,
    name  varchar(255) NOT NULL,
    CONSTRAINT genres_pk PRIMARY KEY (id)
);

CREATE TABLE purchased_video_content
(
    id                bigint AUTO_INCREMENT NOT NULL,
    buy_date          date             NOT NULL,
    user_id           bigint            NOT NULL,
    video_content_id  bigint            NOT NULL,
    CONSTRAINT purchased_video_content_pk PRIMARY KEY (id)
);

CREATE TABLE rating_table
(
    video_content_id bigint NOT NULL,
    user_id          bigint NOT NULL,
    rt_value         int    NOT NULL,
    CONSTRAINT rating_table_pk PRIMARY KEY (video_content_id, user_id)
);

CREATE TABLE vc_actor
(
    video_content_id bigint NOT NULL,
    actors_id        bigint NOT NULL,
    CONSTRAINT vc_actor_pk PRIMARY KEY (video_content_id, actors_id)
);

```

```

CREATE TABLE vc_genre
(
    video_content_id bigint NOT NULL,
    genres_id        int    NOT NULL,
    CONSTRAINT vc_genre_pk PRIMARY KEY (video_content_id, genres_id)
);

CREATE TABLE video_content
(
    id            bigint      NOT NULL AUTO_INCREMENT,
    name          varchar(255) NOT NULL,
    vc_describe   varchar(600) NOT NULL,
    producer      varchar(255) NOT NULL,
    price         float       NOT NULL,
    rating        float       NOT NULL,
    age_rating    int         NOT NULL,
    release_date  date        NOT NULL,
    vc_type       varchar(60)  NOT NULL,
    main_image    text        NOT NULL,
    videos        json        NOT NULL,
    UNIQUE INDEX video_content_ak_1 (name),
    CONSTRAINT video_content_pk PRIMARY KEY (id)
);

ALTER TABLE purchased_video_content
    ADD CONSTRAINT purchased_video_content_video_content FOREIGN KEY
purchased_video_content_video_content (video_content_id)
    REFERENCES video_content (id);

ALTER TABLE rating_table
    ADD CONSTRAINT rating_table_video_content FOREIGN KEY rating_table_video_content
(video_content_id)
    REFERENCES video_content (id);

ALTER TABLE vc_actor
    ADD CONSTRAINT vc_actor_actors FOREIGN KEY vc_actor_actors (actors_id)
    REFERENCES actors (id);

ALTER TABLE vc_actor
    ADD CONSTRAINT vc_actor_video_content FOREIGN KEY vc_actor_video_content (video_content_id)
    REFERENCES video_content (id);

ALTER TABLE vc_genre
    ADD CONSTRAINT vc_genre_genres FOREIGN KEY vc_genre_genres (genres_id)
    REFERENCES genres (id);

ALTER TABLE vc_genre
    ADD CONSTRAINT vc_genre_video_content FOREIGN KEY vc_genre_video_content (video_content_id)
    REFERENCES video_content (id);

CREATE TABLE `wallet_table`
(

```

```

        `user_id`          BIGINT NOT NULL,
        `amount_of_money` BIGINT NOT NULL DEFAULT 0,
        PRIMARY KEY (`user_id`)
    );

```

База даних auth_bd

```

DROP DATABASE IF EXISTS auth_bd;
CREATE DATABASE auth_bd;
USE auth_bd;

```

```

CREATE TABLE authentication_code
(
    id          bigint    NOT NULL AUTO_INCREMENT,
    value       char(30)  NOT NULL,
    date_of_creation datetime NOT NULL,
    active      bool      NOT NULL,
    users_table_id bigint  NOT NULL,
    CONSTRAINT authentication_code_pk PRIMARY KEY (id)
);

```

```

CREATE TABLE roles_table
(
    id    int          NOT NULL AUTO_INCREMENT,
    name  varchar(255) NOT NULL,
    UNIQUE INDEX role_table_ak_1 (name),
    CONSTRAINT roles_table_pk PRIMARY KEY (id)
);

```

```

CREATE TABLE users_table
(
    id          bigint          NOT NULL AUTO_INCREMENT,
    phone_number varchar(15)    NOT NULL,
    password     varchar(60)    NOT NULL,
    nickname     varchar(255)   NOT NULL,
    first_name   varchar(60)    NOT NULL,
    last_name    varchar(60)    NOT NULL,
    birth_date   date           NOT NULL,
    active       bool           NOT NULL,
    selected_currency varchar(3) NOT NULL,
    user_state   enum ('REGISTRATION', 'REGISTERED') NOT NULL,
    email        varchar(60)    NULL,
    role_table_id int           NOT NULL,
    CONSTRAINT users_table_pk PRIMARY KEY (id)
);

```

```

ALTER TABLE authentication_code
    ADD CONSTRAINT authentication_code_users_table FOREIGN KEY authentication_code_users_table
    (users_table_id)
    REFERENCES users_table (id);

```

```

ALTER TABLE users_table
    ADD CONSTRAINT user_table_role_table FOREIGN KEY user_table_role_table (role_table_id)

```

```

REFERENCES roles_table (id);

INSERT INTO roles_table (id, name)
VALUES (1, 'ADMINISTRATION'),
       (10, 'CUSTOMER'),
       (20, 'TELEGRAM_BOT');

ALTER TABLE users_table
ADD date_registration DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP;

```

База даних telegram_bot

```

DROP DATABASE IF EXISTS telegram_bot;
CREATE DATABASE telegram_bot;
USE telegram_bot;

CREATE TABLE user
(
    chat_id          BIGINT          NOT NULL PRIMARY KEY,
    number            VARCHAR(15)    NOT NULL UNIQUE,
    date_registration DATETIME        NOT NULL,
    active            BOOLEAN         NOT NULL
);

```

A.2 Основний код мікросервісів

Файл ua.video.content.service.clients.HttpAuthClient.java

```

@RequiredArgsConstructor
public class HttpAuthClient implements AuthClient {
    private final RestTemplate restTemplate;
    private final String authHeader;
    private final String apiUrl;
    @Override
    public Optional<User> getUser(String token) {
        return sendRequest(token);
    }
    private MultiValueMap<String, String> buildHeader(String token) {
        HttpHeaders headers = new HttpHeaders();
        headers.add(authHeader, token);
        return headers;
    }
    private Optional<User> sendRequest(String token) {
        try {
            var response = restTemplate.postForEntity(apiUrl,
                new HttpEntity<>(buildHeader(token)), User.class);
            if (response.getStatusCode().isError()) {
                return Optional.empty();
            }
            return Optional.ofNullable(response.getBody());
        } catch (RuntimeException re) {
            return Optional.empty();
        }
    }
}

```

Файл ua.video.content.service.clients.HttpTelegramClient.java

```

@RequiredArgsConstructor
public class HttpTelegramClient implements TelegramClient {
    private static final Logger LOG = LoggerFactory.getLogger(HttpTelegramClient.class);
    private final String authHeader;
    private final String token;
    private final String paramPhoneNumber;
}

```

```

private final String urlBot;
private final String urlHasPhoneNumber;
private final String urlSendMessage;
private final String urlSendSurveyMessage;
private final RestTemplate client;
@Override
public boolean hasNumber(@NonNull String phoneNumber) {
    try {
        var result = client.exchange(new RequestEntity<>(buildHeader(), HttpMethod.GET,
            buildUri(urlHasPhoneNumber, Map.of(paramPhoneNumber, phoneNumber))),
Boolean.class);
        if (result.getStatusCode().isError()) {
            return false;
        }
        return Boolean.TRUE.equals(result.getBody());
    } catch (RuntimeException re) {
        LOG.error("Error by uri -> " + urlHasPhoneNumber, re);
        return false;
    }
}
@Override
public boolean sendMessage(@NonNull TextMessage message) {
    return sendMessage(urlSendMessage, message);
}
@Override
public boolean sendSurveyMessage(@NonNull SurveyMessage message) {
    return sendMessage(urlSendSurveyMessage, message);
}
private MultiValueMap<String, String> buildHeader() {
    HttpHeaders headers = new HttpHeaders();
    headers.add(authHeader, token);
    return headers;
}
private URI buildUri(String urlApi, Map<String, String> params) {
    var uriBuilder = UriComponentsBuilder.fromHttpUrl(urlBot + urlApi);
    for (Map.Entry<String, String> param : params.entrySet()) {
        uriBuilder.queryParam(param.getKey(), param.getValue());
    }
    return uriBuilder.build("");
}
private <M> boolean sendMessage(String urlSendMessage, M message) {
    return sendMessage(urlSendMessage, message, Collections.emptyMap());
}
private <M> boolean sendMessage(String urlSendMessage, M message, Map<String, String> query)
{
    try {
        var response = client.postForEntity(buildUri(urlSendMessage, query),
            new HttpEntity<>(message, buildHeader()), String.class);
        if (response.getStatusCode().isError()) {
            return false;
        }
        return !TelegramSendStatus.valueOf(response.getBody()).isError();
    } catch (RuntimeException re) {
        LOG.error("Error by url -> " + urlSendMessage, re);
        return false;
    }
}
}

```

Файл ua.video.content.service.clients.TelegramClient.java

```

public interface TelegramClient {
    boolean hasNumber(@NonNull String phoneNumber);
    boolean sendMessage(@NonNull TextMessage message);
    boolean sendSurveyMessage(@NonNull SurveyMessage message);
}

```

Файл ua.video.content.service.configs.ClientConfig.java

```

@Configuration
@PropertySource("classpath:props/auth.client.yml")
public class ClientConfig {
    @Bean
    AuthClient httpAuthClient(
        @Value("${header}") String authHeader,
        @Value("${api-url}") String apiUrl
    ){
        return new HttpAuthClient(
            new RestTemplate(),

```

```

        authHeader,
        apiUrl
    );
}
@Bean
TelegramClient httpTelegramClient(
    @Value("${auth-header}") String authHeader,
    @Value("${token}") String token,
    @Value("${param-phone-number}") String paramPhoneNumber,
    @Value("${bot}") String urlBot,
    @Value("${has-phone-number}") String urlHasPhoneNumber,
    @Value("${text-message}") String urlSendTextMessage,
    @Value("${survey-message}") String urlSendSurveyMessage
) {
    return new HttpTelegramClient(authHeader, token,
        paramPhoneNumber, urlBot,
        urlHasPhoneNumber, urlSendTextMessage,
        urlSendSurveyMessage,
        new RestTemplate());
}
}

```

Файл ua.video.content.service.configs.MapperConfig.java

```

@Configuration
public class MapperConfig {
    @Bean
    VideoContentMapper videoContentMapperImpl(GenreRepository genreRepository, ActorRepository
actorRepository, ActorMapper actorMapper) {
        return new VideoContentMapperImpl(new ModelMapper(), genreRepository, actorRepository,
actorMapper).init();
    }
    @Bean
    VCESearchCriteriaMapper vceSearchCriteriaMapperImpl() {
        return new VCESearchCriteriaMapperImpl(new ModelMapper());
    }
    @Bean
    PurchasedVideoContentMapper purchasedVideoContentMapperImpl() {
        return new PurchasedVideoContentMapperImpl(new ModelMapper());
    }
    @Bean
    PVCSearchCriteriaMapper pvcSearchCriteriaMapperImpl() {
        return new PVCSearchCriteriaMapperImpl(new ModelMapper());
    }
    @Bean
    VCRatingStatisticalMapper vcRatingStatisticalMapperImpl() {
        return new VCRatingStatisticalMapperImpl();
    }
    @Bean
    VideoDtoMapper videoDtoMapperImpl(
        @Value("${server.port}") String port,
        @Value("${server.servlet.context-path}") String contextPath,
        @Value("${video-content-path}") String videoContentPath
    ) {
        final var url = "http://localhost:" + port + contextPath + videoContentPath;
        return new VideoDtoMapperImpl(url, new ModelMapper());
    }
    @Bean
    VideoContentDtoMapper videoContentDtoMapperImpl(VideoDtoMapper videoDtoMapper,
        @Value("${server.port}") String port,
        @Value("${server.servlet.context-path}")
String contextPath,
        @Value("${main-image-path}") String
mainImagePath) {
        final var url = "http://localhost:" + port + contextPath + mainImagePath + "/";
        return new VideoContentDtoMapperImpl(videoDtoMapper, new ModelMapper(), url);
    }
    @Bean
    VideoContentListMapper videoContentListMapperImpl(VideoContentDtoMapper dtoMapper) {
        return new VideoContentListMapperImpl(dtoMapper);
    }
    @Bean
    VCRatingMapper vcRatingMapperImpl() {
        return new VCRatingMapperImpl();
    }
    @Bean
    RatingSearchCriteriaMapper ratingSearchCriteriaMapperImpl() {
        return new RatingSearchCriteriaMapperImpl(new ModelMapper());
    }
}

```

```

@Bean
RatingEntityMapper ratingEntityMapperImpl() {
    return new RatingEntityMapperImpl();
}
@Bean
RatingModifyMapper ratingModifyMapperImpl() {
    return new RatingModifyMapperImpl(new ModelMapper());
}
@Bean
RatingRespondListMapper ratingRespondListMapperImpl() {
    return new RatingRespondListMapperImpl(new ModelMapper());
}
@Bean
RatingStatisticalListMapper ratingStatisticalListMapperImpl() {
    return new RatingStatisticalListMapperImpl();
}
@Bean
GenreListMapper genreListMapperImpl() {
    return new GenreListMapperImpl();
}
@Bean
PurchasedVideoContentListMapper purchasedVideoContentListMapperImpl() {
    return new PurchasedVideoContentListMapperImpl(new ModelMapper());
}
@Bean
PurchasedVCIdListMapper purchasedVCIdListMapperImpl() {
    return new PurchasedVCIdListMapperImpl();
}
@Bean
PurchasedVCBuyMapper purchasedVCBuyMapperImpl(PurchasedVideoContentService pvcService,
                                                VideoContentService vcService) {
    return new PurchasedVCBuyMapperImpl(pvcService, vcService);
}
@Bean
ActorImageUrlMapper actorImageUrlMapperImpl(
    @Value("${server.host}") String host,
    @Value("${server.port}") String port,
    @Value("${server.servlet.context-path}") String contextPath,
    @Value("${actor-image-path}") String path
){
    return new ActorImageUrlMapperImpl(host, port, contextPath, path);
}
}

```

Файл ua.video.content.service.configs.SecurityConfig.java

```

@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http,
                                           AuthService authService,
                                           @Value("${token-prefix}") String tokenPrefix) throws
Exception {
    http
        .httpBasic().disable()
        .csrf().disable()
        .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
        .and()
        .authorizeHttpRequests()
        .requestMatchers(HttpMethod.OPTIONS, "/*").permitAll()
        .requestMatchers(HttpMethod.GET,
            "/images/**",
            "/videos/**",
            "/genres",
            "/rating",
            "/rating/statistical",
            "/video-content",
            "/actors"
        ).permitAll()
        .requestMatchers(HttpMethod.POST, "/genres")
        .hasAuthority(Role.ADMINISTRATION.name())
        .anyRequest()
        .authenticated()
        .and()
        .addFilterBefore(new JwtAuthFilter(tokenPrefix, authService),
            UsernamePasswordAuthenticationFilter.class);
    return http.build();
}
}

```

```
}
```

Файл ua.video.content.service.configs.ServiceConfig.java

```
@Configuration
@EnableScheduling
public class ServiceConfig {
    @Bean
    GenreService genreServiceImpl(GenreRepository repository) {
        return new GenreServiceImpl(repository);
    }
    @Bean
    VideoContentService videoContentServiceImpl(
        VideoContentRepository vcRepository,
        VCSearchCriteriaMapper searchCriteriaMapper,
        ConverterVCSortField converterVCSortField,
        VideoContentMapper vcMapper
    ) {
        return new VideoContentServiceImpl(
            vcRepository, searchCriteriaMapper, converterVCSortField, vcMapper
        );
    }
    @Bean
    PurchasedVideoContentService purchasedVideoContentServiceImpl(
        PVCSearchCriteriaMapper criteriaMapper,
        PurchasedVideoContentMapper pvcMapper,
        PurchasedVideoContentRepository repository,
        ConverterPVCSortField converterPVCSortField,
        WalletRepository walletRepository,
        VideoContentRepository vcRepository,
        TelegramClient telegramClient
    ) {
        return new PurchasedVideoContentServiceImpl(criteriaMapper,
            pvcMapper, repository, converterPVCSortField,
            walletRepository, vcRepository, telegramClient);
    }
    @Bean
    RatingStatisticalService cacheRatingStatisticalService(
        RatingRepository repository,
        VCRatingStatisticalMapper mapper) {
        return new CacheRatingStatisticalService(repository, mapper);
    }
    @Bean
    VideoContentRatingUpdater videoContentRatingUpdater(
        RatingRepository ratingRepository,
        VideoContentRepository vcRepository) {
        return new VideoContentRatingUpdater(ratingRepository, vcRepository);
    }
    @Bean
    RatingService ratingServiceImpl(RatingRepository repository,
        VCRatingMapper vcRatingMapper,
        RatingSearchCriteriaMapper searchCriteriaMapper,
        RatingEntityMapper entityMapper,
        ConverterRatingSortFiled converterRatingSortFiled) {
        return new RatingServiceImpl(repository, vcRatingMapper,
            searchCriteriaMapper, entityMapper, converterRatingSortFiled);
    }
    @Bean
    AuthService authServiceImpl(AuthClient client) {
        return new AuthServiceImpl(client);
    }
    @Bean
    ActorService actorServiceImpl(
        ActorFilterMapper actorFilterMapper,
        ActorMapper actorMapper,
        ConvectorActorSortField convectorActorSortField,
        ActorRepository actorRepository
    ) {
        return new ActorServiceImpl(actorFilterMapper, actorMapper, convectorActorSortField,
actorRepository);
    }
    @Bean
    WalletService walletServiceImpl(WalletRepository walletRepository){
        return new WalletServiceImpl(walletRepository);
    }
}
```

Файл ua.video.content.service.configs.WebConfig.java

```

@Configuration
public class WebConfig implements WebMvcConfigurer {
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**").allowedMethods("GET", "POST", "PUT");
    }
    @Bean
    HiddenHttpMethodFilter hiddenHttpMethodFilter() {
        return new HiddenHttpMethodFilter();
    }
}

```

Файл ua.video.content.service.controllers.ActorController.java

```

@RestController
@ResponseBody
@RequiredArgsConstructor
@RequestMapping("/actors")
public class ActorController {
    private final ActorListMapper actorListMapper;
    private final ActorService actorService;
    @GetMapping
    ActorList getAll(ActorFilter filter){
        return actorListMapper.toActorList(
            actorService.findBy(filter)
        );
    }
}

```

Файл ua.video.content.service.controllers.GenreController.java

```

@RestController
@ResponseBody
@RequiredArgsConstructor
@RequestMapping("/genres")
public class GenreController {
    private final GenreService genreService;
    private final GenreListMapper listMapper;
    @GetMapping
    GenreList get(@ModelAttribute GenreFilter filter) {
        return listMapper.toGenreList(
            genreService.findAll(filter)
        );
    }
    @PostMapping
    boolean save(@Valid
        @NotBlank(message = "Genre mustn't be blank")
        @NotNull(message = "Genre mustn't be null")
        @Size(min = 1, max = 255, message = "Genre must be between 1 and 255")
        String genre) {
        return genreService.save(genre) != null;
    }
}

```

Файл ua.video.content.service.controllers.PurchasedVideoContentController.java

```

@RestController
@ResponseBody
@RequiredArgsConstructor
@RequestMapping("/bought-video-content")
public class PurchasedVideoContentController {
    private final PurchasedVideoContentService service;
    private final PurchasedVideoContentListMapper purchasedVideoContentListMapper;
    private final PurchasedVCIdListMapper purchasedVCIdListMapper;
    private final PurchasedVCBuyMapper purchasedVCBuyMapper;
    private final WalletService walletService;
    @GetMapping
    PurchasedVideoContentList get(@AuthenticationPrincipal User user, @ModelAttribute
    PurchasedVideoContentFilter filter){
        filter.setUserIds(Set.of(user.getId()));
        return purchasedVideoContentListMapper.toPurchasedVideoContentList(
            service.findBy(filter)
        );
    }
    @GetMapping("only-ids")
}

```

```

        PurchasedVCIdList getIds(@AuthenticationPrincipal User user, @ModelAttribute
PurchasedVideoContentFilter filter){
            filter.setUserIds(Set.of(user.getId()));
            return purchasedVCIdListMapper.toPurchasedVCIdList(
                service.findBy(filter)
            );
        }
        @PostMapping("/buy")
        boolean buy(@AuthenticationPrincipal User user, @Valid @RequestBody
VideoContentIdRequest request){
            return service.save(purchasedVCBuyMapper.toPurchasedVideoContent(
                user.getId(), request.getVideoContentId()
            ),user.getPhoneNumber()) != null;
        }
        @GetMapping("/money")
        Long getMoney(@AuthenticationPrincipal User user){
            return walletService.findByUserId(user.getId()).getAmountOfMoney();
        }
        @PutMapping("/money")
        Long putMoney(@AuthenticationPrincipal User user, @Valid @RequestBody PutMoney money){
            return walletService.putMoney(user.getId(),money.getValue()).getAmountOfMoney();
        }
    }
}

```

Файл ua.video.content.service.controllers.RatingController.java

```

@RestController
@ResponseBody
@RequiredArgsConstructor
@RequestMapping("/rating")
public class RatingController {
    private final RatingRespondListMapper listMapper;
    private final RatingService ratingService;
    private final RatingStatisticalService statisticalService;
    private final RatingModifyMapper ratingModifyMapper;
    private final RatingStatisticalListMapper statisticalListMapper;
    @GetMapping
    public RatingRespondList get(@AuthenticationPrincipal User user, @ModelAttribute RatingFilter
filter){
        filter.setUserIds(List.of(user.getId()));
        return listMapper.toRatingRespondList(
            ratingService.findBy(filter)
        );
    }
    @GetMapping("/statistical")
    public RatingStatisticalList getStatistical(@ModelAttribute RatingStatisticalFilter filter){
        return statisticalListMapper.toRatingStatisticalList(
            statisticalService.findBy(filter)
        );
    }
    @PostMapping
    public boolean saveOrUpdate(@AuthenticationPrincipal User user, @Valid @RequestBody PutRating
rating){
        return ratingService.put(
            ratingModifyMapper.toVCRating(rating, user.getId())
        );
    }
    @DeleteMapping("/:videoContentId")
    public boolean remove(@AuthenticationPrincipal User user, @Valid @NotNull Long
videoContentId){
        return ratingService.remove(videoContentId, user.getId());
    }
}

```

Файл ua.video.content.service.controllers.VideoContentController.java

```

@RequiredArgsConstructor
@RestController
@ResponseBody
@RequestMapping("/video-content")
public class VideoContentController {
    private final VideoContentService service;
    private final VideoContentListMapper mapper;
    @GetMapping
    VideoContentList getAll(@ModelAttribute VideoContentFilter f){
        return mapper.toVideoContentList(service.findBy(f));
    }
}

```

Файл ua.video.content.service.repositories.ActorRepository.java

```
public interface ActorRepository extends
    Repository<ActorEntity, Long>,
    JpaSpecificationExecutor<ActorEntity>,
    ModifyRepository<ActorEntity> {
}
```

Файл ua.video.content.service.repositories.GenreRepository.java

```
public interface GenreRepository extends Repository<GenreEntity, Integer>,
    JpaSpecificationExecutor<GenreEntity>, ModifyRepository<GenreEntity> {
}
```

Файл ua.video.content.service.repositories.ModifyRepository.java

```
public interface ModifyRepository<E> {
    E save(E entity);
    Iterable<E> saveAll(Iterable<E> entities);
    default E update(E entity) {
        return save(entity);
    }
    default Iterable<E> updateAll(Iterable<E> entities) {
        return saveAll(entities);
    }
}
```

Файл ua.video.content.service.repositories.PurchasedVideoContentRepository.java

```
public interface PurchasedVideoContentRepository extends
    Repository<PurchasedVideoContentEntity, Long>,
    JpaSpecificationExecutor<PurchasedVideoContentEntity>,
    ModifyRepository<PurchasedVideoContentEntity>{
}
```

Файл ua.video.content.service.repositories.RatingRepository.java

```
public interface RatingRepository extends
    Repository<RatingEntity, RatingId>,
    JpaSpecificationExecutor<RatingEntity>,
    ModifyRepository<RatingEntity> {
    @Query("SELECT new ua.video.content.service.entities.RatingStatistical(r.id.videoContentId,
count(r.id.userId)) FROM RatingEntity r " +
        "WHERE (:#{#params['vcIds']} IS NULL OR r.id.videoContentId
IN(:#{#params['vcIds']}))" +
        "GROUP BY r.id.videoContentId " +
        "HAVING (:#{#params['hasFromCu']} = FALSE OR count(r.id.userId) >=
(:#{#params['fromCu']})) " +
        "AND (:#{#params['hasToCu']} = FALSE OR count(r.id.userId) <= (:#{#params['toCu']}))"
    )
    Page<RatingStatistical> findStatistical(@Param("params") Map<String, Object> params, Pageable
pageable);
    default Page<RatingStatistical> findStatistical(@NonNull @NotNull
RatingStatisticalSearchCriteria criteria, Pageable pageable) {
        Map<String, Object> param = new HashMap<>();
        param.put("vcIds", Optional.ofNullable(criteria.getVcIds()).filter(vcIds ->
!vcIds.isEmpty()).orElse(null));
        param.put("hasFromCu", Range.hasFrom(criteria.getCountUsers()));
        param.put("fromCu", Range.from(criteria.getCountUsers()));
        param.put("hasToCu", Range.hasTo(criteria.getCountUsers()));
        param.put("toCu", Range.to(criteria.getCountUsers()));
        return findStatistical(
            param, pageable
        );
    }
    default List<RatingStatistical> findStatistical(@NonNull @NotNull
RatingStatisticalSearchCriteria criteria) {
        return findStatistical(criteria, PageRequest.of(0, 100, Sort.by(Sort.Direction.DISC,
"id.videoContentId"))).stream().toList();
    }
}
```

Файл ua.video.content.service.repositories.VideoContentRepository.java

```
public interface VideoContentRepository extends
    Repository<VideoContentEntity, Long>,
    JpaSpecificationExecutor<VideoContentEntity>,
    ModifyRepository<VideoContentEntity> {
}
```

```
}
```

Файл ua.video.content.service.repositories.WalletRepository.java

```
public interface WalletRepository extends Repository<WalletEntity, Long>,
    JpaSpecificationExecutor<WalletEntity>,
    ModifyRepository<WalletEntity>{
    default WalletEntity getByUserIdOrSave(@NonNull Long userId, long defaultAmountOfMoney){
        var walletObj = findOne(WalletSearchCriteria.builder().userId(userId).build());
        return walletObj.orElseGet(() -> save(new WalletEntity(userId, defaultAmountOfMoney)));
    }
}
```

Файл ua.video.content.service.services.ActorServiceImpl.java

```
@RequiredArgsConstructor
public class ActorServiceImpl implements ActorService{
    private final ActorFilterMapper actorFilterMapper;
    private final ActorMapper actorMapper;
    private final ConvectorActorSortField convectorActorSortField;
    private final ActorRepository actorRepository;
    @Override
    public Page<Actor> findBy(ActorFilter filter) {
        return actorRepository.findAll(
            actorFilterMapper.toActorSearchCriteria(filter),
            filter.toPageable(0,100, ActorSortField.NAME, convectorActorSortField::convert)
        ).map(actorMapper::toActor);
    }
    @Override
    public Actor save(Actor actor) {
        return Optional.ofNullable(actor)
            .map(actorMapper::toActorEntity)
            .map(actorRepository::save)
            .map(actorMapper::toActor)
            .orElseThrow();
    }
    @Override
    public Actor update(Actor actor) {
        return Optional.ofNullable(actor)
            .map(actorMapper::toActorEntity)
            .map(actorRepository::update)
            .map(actorMapper::toActor)
            .orElseThrow();
    }
}
```

Файл ua.video.content.service.services.AuthServiceImpl.java

```
@RequiredArgsConstructor
public class AuthServiceImpl implements AuthService{
    private final AuthClient client;
    @Override
    @Cacheable(value = "authUsers",key = "#token", cacheManager = "authUserCacheManager")
    public Optional<User> getUser(@NonNull String token) {
        return client.getUser(token);
    }
}
```

Файл ua.video.content.service.services.CacheRatingStatisticalService.java

```
public class CacheRatingStatisticalService implements RatingStatisticalService {
    private static final Map<Map.Entry<RatingStatisticalFieldColumn, Boolean>,
    Comparator<VCRatingStatistical>> COMPARATORS = buildComparators();
    private static final long FIXED_DELAY_TREE_HOUR = 1000 * 60 * 60 * 4;
    private final RatingRepository repository;
    private final VCRatingStatisticalMapper mapper;
    private Map<Long, VCRatingStatistical> cacheById;
    private NavigableMap<Long, List<VCRatingStatistical>> cacheByCount;
    public CacheRatingStatisticalService(RatingRepository repository, VCRatingStatisticalMapper
    mapper) {
        this.repository = repository;
        this.mapper = mapper;
    }
    @Override
    public List<VCRatingStatistical> findBy(RatingStatisticalFilter f) {
        if (f.isEmpty()) {
            return cacheByCount.values()
                .stream().flatMap(List::stream)
        }
```

```

        .toList();
    }
    if (!CollectionUtils.isEmpty(f.getVcIds()) && f.getCount() == null) {
        return usePage(f.getVcIds()
            .stream()
            .map(cacheById::get), f);
    }
    if (CollectionUtils.isEmpty(f.getVcIds()) && f.getCount() != null) {
        return usePage(filterByCount(f), f);
    }
    if (!PageFilter.isEmpty(f)) {
        return usePage(cacheByCount.values()
            .stream().flatMap(List::stream), f);
    }
    return usePage(filterByCount(f)
        .filter(rs -> f.getVcIds().contains(rs.videoContentId())), f);
}

private List<VCRatingStatistical> usePage(Stream<VCRatingStatistical> stream,
RatingStatisticalFilter f) {
    return stream.sorted(getComparator(f))
        .skip(f.getPageOrDefault(0).longValue() * f.getSizeOrDefault(100))
        .limit(f.getSizeOrDefault(100))
        .toList();
}

private Long to(Range<Long> range) {
    return Range.hasTo(range) ? range.getTo() : cacheByCount.lastKey();
}

private Long from(Range<Long> range) {
    return Range.hasFrom(range) ? range.getFrom() : cacheByCount.firstKey();
}

private Stream<VCRatingStatistical> filterByCount(RatingStatisticalFilter f) {
    return cacheByCount.subMap(
        from(f.getCount()),
        true,
        to(f.getCount()),
        true
    ).values()
        .stream()
        .flatMap(Collection::stream);
}

@Scheduled(initialDelay = 20 * 1000, fixedDelay = FIXED_DELAY_TREE_HOUR)
private void update() {
    Map<Long, VCRatingStatistical> newCacheById = new ConcurrentHashMap<>();
    NavigableMap<Long, List<VCRatingStatistical>> newCacheByCount = Collections
        .synchronizedNavigableMap(new TreeMap<>());
    repository.findStatistical(new RatingStatisticalSearchCriteria())
        .parallelStream()
        .map(mapper::toVCRatingStatistical)
        .forEach(s -> {
            newCacheById.put(s.videoContentId(), s);
            newCacheByCount.compute(s.countUsers(), (key, val) -> {
                if (val == null) {
                    val = Collections.synchronizedList(new ArrayList<>());
                }
                val.add(s);
                return val;
            });
        });
    this.cacheById = newCacheById;
    this.cacheByCount = newCacheByCount;
}

private Comparator<VCRatingStatistical>
getComparator(PageFilter<RatingStatisticalFieldColumn> page) {
    return
    COMPARATORS.get(Map.entry(page.getSortFieldOrDefault(RatingStatisticalFieldColumn.USER_COUNTS),
    !page.isDesc()));
}

private static Map<Map.Entry<RatingStatisticalFieldColumn, Boolean>,
Comparator<VCRatingStatistical>> buildComparators() {
    return Map.of(
        Map.entry(RatingStatisticalFieldColumn.USER_COUNTS, true),
        Comparator.comparingLong(VCRatingStatistical::countUsers),
        Map.entry(RatingStatisticalFieldColumn.VIDEO_CONTENT_ID, true),
        Comparator.comparingLong(VCRatingStatistical::videoContentId),
        Map.entry(RatingStatisticalFieldColumn.USER_COUNTS, false),
        Comparator.comparingLong(VCRatingStatistical::countUsers).reversed(),
        Map.entry(RatingStatisticalFieldColumn.VIDEO_CONTENT_ID, false),
        Comparator.comparingLong(VCRatingStatistical::videoContentId).reversed()
    );
}
}

```

```
}
```

Файл ua.video.content.service.services.GenreServiceImpl.java

```
@RequiredArgsConstructor
public class GenreServiceImpl implements GenreService {
    private final GenreRepository repository;
    @Override
    public Page<String> findAll(GenreFilter filter) {
        return repository.findAll(
            GenreSearchCriteria.builder()
                .name(filter.getName())
                .build(),
            filter.toPageable(0, 100,
GenreColumn.getProps().getProp(GenreColumn.NAME))
                .map(GenreEntity::getName);
        )
    }
    @Override
    public String update(String oldGenre, String newGenre) {
        var genreOpt = repository.findOne(GenreSearchCriteria.builder()
            .names(List.of(oldGenre))
            .build());
        if (genreOpt.isEmpty()) {
            return null;
        }
        var genre = genreOpt.get();
        genre.setName(oldGenre);
        return repository.update(genre).getName();
    }
    @Override
    public String save(String genre) {
        return repository.save(new GenreEntity(null, genre)).getName();
    }
}
```

Файл ua.video.content.service.services.PurchasedVideoContentServiceImpl.java

```
@RequiredArgsConstructor
public class PurchasedVideoContentServiceImpl implements PurchasedVideoContentService {
    private final PVCSearchCriteriaMapper criteriaMapper;
    private final PurchasedVideoContentMapper pvcMapper;
    private final PurchasedVideoContentRepository repository;
    private final ConverterPVCSortField converterPVCSortField;
    private final WalletRepository walletRepository;
    private final VideoContentRepository vcRepository;
    private final TelegramClient telegramClient;
    @Override
    public Page<PurchasedVideoContent> findBy(PurchasedVideoContentFilter f) {
        return repository.findAll(
            criteriaMapper.toPurchasedVCSearchCriteria(f),
            f.toPageable(0, 100, PVCSortField.BUY_DATE, converterPVCSortField::convert)
        ).map(pvcMapper::toPurchasedVideoContent);
    }
    @Override
    @Transactional
    public PurchasedVideoContent save(PurchasedVideoContent p, String phoneNumber) {
        var purVC = Optional.ofNullable(p).map(pvcMapper::toPurchasedVideoContentEntity)
            .orElseThrow(() -> new RuntimeException("Invalid map"));
        var wallet = this.walletRepository.getByUserIdOrSave(purVC.getUserId(), 1000);
        var vc = this.vcRepository.findOne(VCSearchCriteria
            .builder().vcId(p.getVideoContentId()).build()).orElseThrow();
        long different = (wallet.getAmountOfMoney() - Math.round(vc.getPrice()));
        if (different < 0) {
            throw new BuyVideoContentException("Недостатньо коштів");
        }
        var savedPurVc = repository.save(purVC);
        wallet.setAmountOfMoney(different);
        sendMessage(phoneNumber, vc, this.walletRepository.update(wallet));
        return pvcMapper.toPurchasedVideoContent(savedPurVc);
    }
    @Override
    @Transactional
    public boolean remove(@NonNull Long userId, @NonNull Long vcId) {
        return repository.delete(
            PurchasedVCSearchCriteria
                .builder()
                .vcId(vcId)
                .userId(userId)
                .build()
        );
    }
}
```

```

        ) == 1;
    }
    private void sendMessage(String phoneNumber, VideoContentEntity vc, WalletEntity wallet){
        telegramClient.sendMessage(
            TextMessage
                .builder()
                .phoneNumber(phoneNumber)
                .text("Було придбано " + toText(vc.getType()) + " '" + vc.getName() +
                    "' за " + vc.getPrice() +
                    " грн. \nНа вашому рахунку залишилось " +
                    wallet.getAmountOfMoney() + " грн.")
                .build()
        );
    }
    private String toText(String type){
        return VCType.FILM.equals(VCType.valueOf(type)) ? "фільм" : "серіал";
    }
}

```

Файл ua.video.content.service.services.RatingServiceImpl.java

```

@RequiredArgsConstructor
public class RatingServiceImpl implements RatingService {
    private final RatingRepository repository;
    private final VCRatingMapper vcRatingMapper;
    private final RatingSearchCriteriaMapper searchCriteriaMapper;
    private final RatingEntityMapper entityMapper;
    private final ConverterRatingSortFiled converterRatingSortFiled;
    @Override
    public boolean put(VCRating rating) {
        return Optional.ofNullable(rating)
            .map(entityMapper::toRatingEntity)
            .map(repository::save)
            .isPresent();
    }
    @Override
    @Transactional
    public boolean remove(@NotNull Long videoContentId, @NotNull Long userId) {
        return repository.delete(
            RatingSearchCriteria.builder()
                .userId(userId)
                .vcId(videoContentId)
                .build()
        ) == 1;
    }
    @Override
    public Page<VCRating> findBy(@NotNull RatingFilter filter) {
        return repository.findAll(
            searchCriteriaMapper.toRatingSearchCriteria(filter),
            filter.toPageable(0,100, RatingSortFiled.VALUE,
                converterRatingSortFiled::convert)
        ).map(vcRatingMapper::toVCRating);
    }
}

```

Файл ua.video.content.service.services.VideoContentRatingUpdater.java

```

@RequiredArgsConstructor
public class VideoContentRatingUpdater {
    private static final Logger LOG = LoggerFactory.getLogger(VideoContentRatingUpdater.class);
    private static final long FIXED_DELAY_TREE_HOUR = 1000 * 60 * 60 * 3;
    private final RatingRepository ratingRepository;
    private final VideoContentRepository vcRepository;
    @Scheduled(initialDelay = 20 * 1000 * 1000, fixedDelay = FIXED_DELAY_TREE_HOUR)
    public void updateRating(){
        LOG.info("Start update rating ");
        var result = ratingRepository.findAll(new RatingSearchCriteria())
            .stream()
            .collect(Collectors.groupingBy(re -> re.getId().getVideoContentId(),
                Collectors.averagingInt(RatingEntity::getValue)));
        var vcs = vcRepository.findAll(new VCSearchCriteria()).stream()
            .peek(vc -> vc.setRating(
                toFormat(result.get(vc.getId())))
            ).toList();
        vcRepository.updateAll(vcs);
        LOG.info("End update rating ");
    }
    private Float toFormat(Double rating){
        if(rating == null){

```

```

        return 0f;
    }
    long result = Math.round(rating * 100);
    return result / 100f;
}
}

```

Файл ua.video.content.service.services.VideoContentServiceImpl.java

```

@RequiredArgsConstructor
public class VideoContentServiceImpl implements VideoContentService{
    private final VideoContentRepository vcRepository;
    private final VCESearchCriteriaMapper searchCriteriaMapper;
    private final ConverterVCSortField converterVCSortField;
    private final VideoContentMapper vcMapper;
    @Override
    @Cacheable(value = "videoContents", key = "#f.vcIds", condition = "#f.vcIds != null &&
    #f.vcIds.size() == 1")
    public Page<VideoContent> findBy(VideoContentFilter f) {
        return vcRepository.findAll(
            searchCriteriaMapper.toVCESearchCriteria(f),
            f.toPageable(0,100, VCSortField.RATING,converterVCSortField::convert)
        ).map(vcMapper::toVideoContent);
    }
    @Override
    @CacheEvict(value = "videoContents")
    public VideoContent save(VideoContent videoContent) {
        return Optional.of(videoContent)
            .map(vcMapper::toVideoContentEntity)
            .map(vcRepository::save)
            .map(vcMapper::toVideoContent)
            .orElseThrow();
    }
    @Override
    @CachePut(value = "videoContents", key = "#videoContent.id", condition = "#videoContent.id !=
    null")
    public VideoContent update(VideoContent videoContent) {
        return Optional.of(videoContent)
            .map(vcMapper::toVideoContentEntity)
            .map(vcRepository::update)
            .map(vcMapper::toVideoContent)
            .orElseThrow();
    }
}

```

Файл ua.video.content.service.services.WalletServiceImpl.java

```

@RequiredArgsConstructor
public class WalletServiceImpl implements WalletService{
    private final WalletRepository walletRepository;
    private final ModelMapper mapper= new ModelMapper();
    @Override
    public Wallet findById(Long userId) {
        return mapper.map(
            walletRepository.getByUserIdOrSave(userId, 1000),
            Wallet.class
        );
    }
    @Override
    public Wallet putMoney(Long userId, Long value) {
        var entity = walletRepository.getByUserIdOrSave(userId,1000);
        entity.putMoney(value);
        return mapper.map(
            walletRepository.update(entity),
            Wallet.class
        );
    }
}

```

Файл ua.video.content.service.VideoContentServiceApplication.java

```

@SpringBootApplication(
    exclude = UserDetailsServiceAutoConfiguration.class
)
@EnableCaching
public class VideoContentServiceApplication {
    public static void main(String[] args) {
        SpringApplication.run(VideoContentServiceApplication.class, args);
    }
}

```

```
    }
}
```

Файл

ua.project.player.recommendation.repositories.MongoRecommendationRepository.java

```
public class MongoRecommendationRepository extends
AbstractClassDefaultRepository<RecommendationEntity, ObjectId> implements
RecommendationRepository{
    private final String videoContentIdsName;
    private final String userIdName;
    public MongoRecommendationRepository(MongoTemplate template) {
        super(template, RecommendationEntity.class,

Objects.requireNonNull(RecommendationField.FIELD_PROPS.getProp(RecommendationField.ID)),
        RecommendationEntity::getId,
        new Gson());
        this.videoContentIdsName =
Objects.requireNonNull(RecommendationField.FIELD_PROPS.getProp(RecommendationField.VIDEO_CONTENT_
IDS));
        this.userIdName =
Objects.requireNonNull(RecommendationField.FIELD_PROPS.getProp(RecommendationField.USER_ID));
    }
    @Override
    public UpdateResult removeVideoContentId(Long userId, Long videoContentId) {
        Query query = new Query(Criteria.where(userIdName).is(userId));
        return removeVideoContentIdByQuery(query, videoContentId);
    }
    @Override
    public UpdateResult removeVideoContentId(ObjectId id, Long videoContentId) {
        Query query = new Query(Criteria.where(super.idName).is(id));
        return removeVideoContentIdByQuery(query, videoContentId);
    }
    private UpdateResult removeVideoContentIdByQuery(Query query, Long videoContentId){
        Update update = new Update().pull(videoContentIdsName, videoContentId);
        return template.updateFirst(query, update, RecommendationEntity.class);
    }
}
```

Файл

ua.project.player.recommendation.repositories.MongoVCHistoryEntityRepository.java

```
public class MongoVCHistoryEntityRepository extends
AbstractClassDefaultRepository<VCHistoryEntity, ObjectId> implements VCHistoryRepository {
    private final String durationName;
    public MongoVCHistoryEntityRepository(MongoTemplate template) {
        super(template, VCHistoryEntity.class,
            Objects.requireNonNull(VCHistoryField.FIELD_PROPS.getProp(VCHistoryField.ID)),
            VCHistoryEntity::getId,
            new GsonBuilder().registerTypeAdapter(LocalDateTime.class, new
LocalDateTimeAdapter()).create()
        );
        durationName = VCHistoryField.FIELD_PROPS.getProp(VCHistoryField.DURATION);
    }
    @Override
    public UpdateResult incrementDuration(ObjectId id, Long value) {
        Query query = new Query().addCriteria(Criteria.where(idName).is(id));
        Update update = new Update().inc(durationName, value);
        return template.updateFirst(query, update, VCHistoryEntity.class);
    }
}
```

Файл ua.project.player.recommendation.repositories.MongoVideoContentRepository.java

```
public class MongoVideoContentRepository
    extends AbstractClassDefaultRepository<VideoContentStatisticalEntity, Long>
    implements VideoContentStatisticalRepository {
    private final Query findLastVersionId;
    public MongoVideoContentRepository(MongoTemplate template) {
        super(template,
            VideoContentStatisticalEntity.class,

VideoContentStatisticalField.FIELD_PROPS.getProp(VideoContentStatisticalField.ID),
            VideoContentStatisticalEntity::getId,
            new GsonBuilder()
        );
    }
}
```

```

        .registerTypeAdapter(LocalDate.class, new LocalDateAdapter())
        .create());
    this.findLastVersionId = new Query().with(Sort.by(Sort.Direction.DISC,
VideoContentStatisticalField.FIELD_PROPS.getProp(VideoContentStatisticalField.VERSION_ID)));
    }
    @Override
    public long count(Queryable queryable) {
        return super.template.count(queryable.toQuery(), VideoContentStatisticalEntity.class);
    }
    @Override
    public long lastVersionId() {
        return Objects.requireNonNull(super.template.findOne(findLastVersionId,
VideoContentStatisticalEntity.class)).getVersionId();
    }
}

```

Файл ua.project.player.recommendation.repositories.RecommendationRepository.java

```

public interface RecommendationRepository extends DefaultRepository<RecommendationEntity> {
    UpdateResult removeVideoContentId(Long userId, Long videoContentId);
    UpdateResult removeVideoContentId(ObjectId id, Long videoContentId);
}

```

Файл ua.project.player.recommendation.repositories.VCHistoryRepository.java

```

public interface VCHistoryRepository extends DefaultRepository<VCHistoryEntity> {
    UpdateResult incrementDuration(ObjectId id, Long value);
}

```

Файл

ua.project.player.recommendation.repositories.VideoContentStatisticalRepository.java

```

public interface VideoContentStatisticalRepository extends
DefaultRepository<VideoContentStatisticalEntity>{
    long count(Queryable queryable);
    long lastVersionId();
}

```

Файл ua.project.player.recommendation.services.AuthServiceImpl.java

```

@RequiredArgsConstructor
public class AuthServiceImpl implements AuthService{
    private final AuthClient client;
    @Override
    @Cacheable(value = "authUsers", key = "#token", cacheManager = "authUserCacheManager")
    public Optional<User> getUser(@NonNull String token) {
        return client.getUser(token);
    }
}

```

Файл ua.project.player.recommendation.services.RecommendationGeneratorImpl.java

```

@RequiredArgsConstructor
public class RecommendationGeneratorImpl implements RecommendationGenerator {
    private final VCHistoryRepository vcHistoryRepository;
    private final VideoContentStatisticalRepository vcsRepository;
    private final List<RecommendationCalculateFactory> calculateFactories;
    private final String dateViewedName =
VCHistoryField.FIELD_PROPS.getProp(VCHistoryField.DATE_VIEWED);
    private final String vcRatingName =
VideoContentStatisticalField.getFieldName(VideoContentStatisticalField.RATING);
    @Override
    public List<Long> generate(Long userId, Integer userAge) {
        Optional<LocalDateTime> lastDateViewedOpt = getLastDateViewed(userId);
        if (lastDateViewedOpt.isEmpty()) {
            return getDefaultRecommendation(userAge);
        }
        VCHistoryCountStatistical countStatistical =
getVCHistoryCountStatistical(lastDateViewedOpt.get(), userId, userAge);
        var videoContentList = vcsRepository.findBy(
VCSFindByHistoryCountStatistical.builder().countStatistical(countStatistical).build()
        .and(
VCStatisticalSearchCriteria.builder().ageRating(Range.ofTo(userAge)).build()
        )
    )
}

```

```

    );
    HashMap<Long,Long> assessments = new HashMap<>();
    var calculates = calculateFactories.stream().map(f ->
f.create(countStatistical)).toList();
    videoContentList.forEach(vcs -> calculates.forEach(c -> c.calculate(vcs,assessments)));
    return assessments.entrySet().stream().sorted(new
EntryValueComparator<>()).map(Map.Entry::getKey).toList();
}
private VCHistoryCountStatistical getVCHistoryCountStatistical(LocalDateTime
lastDateViewed,Long userId, Integer userAge){
    var history = vcHistoryRepository.findBy(VCHistorySearchCriteria
        .builder()
        .userId(userId)
        .dateViewed(Range.ofFrom(lastDateViewed.minusWeeks(2)))
        .build(),
        PageRequest.of(0, 10000, Sort.Direction.DESC, dateViewedName)
    );
    var watchedVideoContent =
        vcsRepository
            .findBy(VCStatisticalSearchCriteria
                .builder()
                .vcIds(history.filter(h -> h.getVideoContentId() !=
null).map(VCHistoryEntity::getVideoContentId).toSet())
                .ageRating(Range.ofTo(userAge))
                .build(),
                PageRequest.of(0, 10000, Sort.Direction.DESC, vcRatingName)
            );
    return VCHistoryCountStatistical.of(
        history.getContent(),
        watchedVideoContent.getContent()
    );
}
private Optional<LocalDateTime> getLastDateViewed(Long userId) {
    return vcHistoryRepository.findBy(
        VCHistorySearchCriteria.builder().userId(userId).build(),
        PageRequest.of(0, 10000, Sort.Direction.DESC, dateViewedName)
    ).stream().findFirst().map(VCHistoryEntity::getDateViewed);
}
private List<Long> getDefaultRecommendation(Integer userAge) {
    return vcsRepository.findBy(
        VCStatisticalSearchCriteria
            .builder()
            .ageRating(Range.ofTo(userAge))
            .build()
    ).stream().map(VideoContentStatisticalEntity::getId).toList();
}
}

```

Файл ua.project.player.recommendation.services.RecommendationServiceImpl.java

```

@RequiredArgsConstructor
public class RecommendationServiceImpl implements RecommendationService {
    private final RecommendationRepository repository;
    private final RecommendationSearchCriteriaMapper searchCriteriaMapper;
    private final RecommendationMapper recommendationMapper;
    private final RecommendationSortFieldConverter sortFieldConverter;
    private final RecommendationGenerator recommendationGenerator;
    @Override
    public Page<Recommendation> findBy(RecommendationFilter filter) {
        return repository.findBy(
            searchCriteriaMapper.toRecommendationSearchCriteria(filter),
            filter.toPageable(0, 100, RecommendationSortField.USER_ID,
sortFieldConverter::converter)
        ).map(recommendationMapper::toRecommendation);
    }
    @Override
    public Recommendation getOrGenerate(Long userId, Integer userAge) {
        var recommendationOpt =
repository.findBy(RecommendationSearchCriteria.builder().userId(userId).build())
            .stream().findFirst();
        if (recommendationOpt.isEmpty()) {
            var entity = repository.insert(RecommendationEntity
                .builder()
                .userId(userId)
                .videoContentIds(recommendationGenerator.generate(userId, userAge))
                .build());
            return recommendationMapper.toRecommendation(entity);
        }
        if (recommendationOpt.get().getVideoContentIds().size() > 5) {

```

```

        return recommendationOpt.map(recommendationMapper::toRecommendation).orElseThrow();
    }
    var recommend = recommendationOpt.get();
    recommend.setVideoContentIds(recommendationGenerator.generate(userId, userAge));
    if(repository.updateAll(recommend).getModifiedCount() != 0){
        return recommendationMapper.toRecommendation(recommend);
    }
    throw new RuntimeException("Recommendation didn't update");
}
@Override
public boolean removeVideoContentId(Long userId, Long videoContentId) {
    return repository.removeVideoContentId(userId, videoContentId).getModifiedCount() != 0;
}
@Override
public boolean removeVideoContentId(String id, Long userId, Long videoContentId) {
    var notExists =
!this.exists(RecommendationFilter.builder().userId(userId).recommendationId(id).build());
    if (notExists) {
        return false;
    }
    return repository.removeVideoContentId(new ObjectId(id),
videoContentId).getModifiedCount() != 0;
}
@Override
public boolean exists(RecommendationFilter filter) {
    return !this.findBy(filter).getContent().isEmpty();
}
}
}

```

Файл ua.project.player.recommendation.services.VCHistoryServiceImpl.java

```

@RequiredArgsConstructor
public class VCHistoryServiceImpl implements VCHistoryService {
    private final VCHistoryRepository repository;
    private final VCHistoryMapper vcHistoryMapper;
    private final VCHistorySearchCriteriaMapper searchCriteriaMapper;
    private final VCHistorySortFieldConverter sortFieldConverter;
    @Override
    public Page<VCHistory> findBy(VCHistoryFilter filter) {
        return repository.findBy(
            searchCriteriaMapper.toVCHistorySearchCriteria(filter),
            filter.toPageable(0,100, VCHistorySortField.DATE_VIEWED,
                sortFieldConverter::converter)
        ).map(vcHistoryMapper::toVCHistory);
    }
    @Override
    public boolean save(VCHistory history) {
        return repository.insert(vcHistoryMapper.toVCHistoryEntity(history)).getId() != null;
    }
    @Override
    public boolean incrementDuration(String id, Long value, Long userId) {
        var empty = repository.findBy(VCHistorySearchCriteria.builder().historyId(new
ObjectId(id)).userId(userId).build()).isEmpty();
        if(empty){
            return false;
        }
        return repository.incrementDuration(
            new ObjectId(id), value
        ).getModifiedCount() != 0;
    }
}

```

Файл ua.project.player.recommendation.services.VideoStatsCollectorImpl.java

```

@RequiredArgsConstructor
public class VideoStatsCollectorImpl implements VideoStatsCollector{
    private static final long REFRESH_TIME_ONE_DAY = 1000 * 60 * 60 * 24;
    private final VideoContentClient vcClient;
    private final VCRatingClient ratingClient;
    private final VideoContentStatisticalMapper mapper;
    private final VideoContentStatisticalRepository vcRepository;
    @Override
    @Scheduled(initialDelay = REFRESH_TIME_ONE_DAY, fixedDelay = REFRESH_TIME_ONE_DAY)
    public void updateVideoStats() {
        final var defRatingStatistical = new RatingStatistical(null, 0L);
        final long verId = System.currentTimeMillis();
        var map = ratingClient
            .getRatingStatistical()
            .stream()

```

```

        .collect(Collectors.toMap(RatingStatistical::getVideoContentId, r -> r));
var list = vcClient.getVideoContents()
    .stream()
    .map(vc -> mapper.toVideoContentStatisticalEntity(
        vc,
        map.getOrDefault(vc.getId(), defRatingStatistical),
        verId
    ))
    .toList();
vcRepository.insert(list);
}
}

```

Файл ua.telegram.videoproject.bot.TelegramBot.java

```

@RequiredArgsConstructor
public class TelegramBot extends TelegramLongPollingCommandBot {
    private static final Logger LOGGER = LoggerFactory.getLogger(TelegramBot.class);
    private final String username;
    private final String token;
    private final UserMessageHandle messageHandle;
    @Override
    public String getBotUsername() {
        return username;
    }
    @Override
    public String getBotToken() {
        return token;
    }
    @Override
    public void processNonCommandUpdate(Update update) {
        try{
            var resultMessage = messageHandle.handle(update);
            if(resultMessage.isPresent()){
                execute(resultMessage.get());
            }
        }catch (Exception e){
            LOGGER.error("Error with execute telegram request ",e);
        }
    }
}

```

Файл ua.telegram.videoproject.controller.telegram.CommandController.java

```

@BotCommandController
public class CommandController {
    private final TelegramLongPollingCommandBot bot;
    public CommandController(TelegramLongPollingCommandBot bot) {
        this.bot = bot;
    }
    @BotCommandMapper(commands = {"/download"})
    public void download(Update update) throws TelegramApiException, InterruptedException,
    IOException {
        final Long chat_id = update.getMessage().getChatId();
        String TEXT = "Download file %s";
        final int messageId = bot.execute(SendMessage.builder().text(String.format(TEXT,0) +
        "%").chatId(chat_id.toString()).build()).getMessageId();
        final EditMessageText editMessageText =
        EditMessageText.builder().messageId(messageId).chatId(chat_id).text("").build();
        int process = 5;
        do {
            Thread.sleep(250L);
            editMessageText.setText(String.format(TEXT,process) + "%");
            process+=5;
            bot.execute(editMessageText);
        }while(process < 101);
        Thread.sleep(300L);
        editMessageText.setText("\uD83C\uDF89\uD83C\uDF89\uD83C\uDF89 Successful
        \uD83C\uDF89\uD83C\uDF89\uD83C\uDF89");
        bot.execute(editMessageText);
        ClassPathResource resource = new ClassPathResource("video/good-good-evil-smile.gif");
        InputFile videoFile = new InputFile(resource.getFile());
        SendAnimation animation = new SendAnimation(chat_id.toString(),videoFile);
        bot.executeAsync(animation);
    }
}

```

Файл ua.telegram.videoproject.controller.telegram.MoneyController.java

```

@BotCommandController
@RequiredArgsConstructor
public class MoneyController {
    private final MoneyService moneyService;
    private final TelegramLongPollingCommandBot bot;
    @BotCommandMapper(commands = "/wallet")
    public void getWallet(Update update) throws TelegramApiException {
        var chatId = UpdateUtils.getChatId(update);
        bot.executeAsync(
            SendMessage.builder()
                .chatId(chatId)
                .text("Ваш баланс " + moneyService.getMoney(chatId) + " грн")
                .build()
        );
    }
    @BotCommandMapper(commands = "/put_money")
    public void putMoneyToWallet(Update update){
        moneyService.putMoney(update);
    }
}

```

Файл ua.telegram.videoproject.controller.telegram.SettingController.java

```

@BotCommandController
@RequiredArgsConstructor
public class SettingController {
    private final TelegramSettingService settingService;
    @BotCommandMapper(commands = "/clear")
    public void clear(Update update){
        settingService.clearEvents(UpdateUtils.getChatId(update));
    }
    @BotCommandMapper(commands = {"/help"})
    public void showAllCommands(Update update){
        settingService.showAllCommand(UpdateUtils.getChatId(update));
    }
}

```

Файл ua.telegram.videoproject.controller.telegram.UserController.java

```

@BotCommandController
@RequiredArgsConstructor
public class UserController {
    private final TelegramUserService telegramUserService;
    @BotCommandMapper(commands = {"/show_phone_number"})
    public void show(Update update) {
        telegramUserService.showPhoneNumber(update);
    }
}

```

Файл ua.telegram.videoproject.controller.web.RestInformationUsersController.java

```

@Validated
@RestController
@RequestMapping("/information-user/v1")
@RequiredArgsConstructor
public class RestInformationUsersController {
    private final UserService serviceUser;
    @ResponseBody
    @RequestMapping(value = "/has-number", method = RequestMethod.GET)
    public boolean isConfidentNumber(@RequestParam @Valid @Pattern(regexp = "\\d{10,15}", message = "Number not correct") String phoneNumber) {
        return
            serviceUser.findOne(UserUniqueFieldFilter.builder().phoneNumber(phoneNumber).build()).isPresent();
    }
}

```

Файл ua.telegram.videoproject.controller.web.SendMessageController.java

```

@Validated
@RestController
@RequestMapping("/v1/send-message")
@RequiredArgsConstructor
public class SendMessageController {
    private final MessageService messageService;
    private final TelegramMessageMapper messageMapper;
}

```

```

private final SurveyTelegramMessageMapper surveyTelegramMessageMapper;
@ResponseBody
@Validated(OnValidateForm.class)
@RequestMapping(value = "/text", method = RequestMethod.POST)
public String sendTextMessage(@RequestBody @Valid TelegramMessageRequest request) {
    return messageService.sendMessage(messageMapper.toTelegramMessage(request)).name();
}
@ResponseBody
@Validated(OnValidateForm.class)
@RequestMapping(value = "/survey", method = RequestMethod.POST)
public String sendSurveyMessage(@RequestBody @Valid SurveyTelegramMessageRequest request) {
    return
messageService.sendMessage(surveyTelegramMessageMapper.toSurveyTelegramMessage(request)).name();
}
}

```

Файл ua.telegram.videoproject.events.PutMoneyEvent.java

```

@RequiredArgsConstructor
public class PutMoneyEvent implements Event {
    public static final PutMoneyEvent INSTANCE = new PutMoneyEvent(null, null, null, 0);
    private final TelegramLongPollingCommandBot bot;
    private final TelegramEvents events;
    private final SenderPutMoney sendleMoney;
    private final long chatId;
    private final PutMoney.PutMoneyBuilder putMoneyBuilder = PutMoney.builder();
    private final Queue<PutMoneyEventItem> itemQueue = itemQueueBuild();
    private boolean endTransaction = false;
    @Override
    public void emit(Update update) {
        if (itemQueue.isEmpty()) {
            end();
            return;
        }
        try {
            itemQueue.element().checkAndPut(update.getMessage().getText());
        } catch (PutMoneyException e) {
            sendMessage(e.getMessage());
            return;
        } catch (ValidationException e) {
            sendMessage("[Некоректний формат дати]");
            return;
        }
        itemQueue.poll();
        if (itemQueue.isEmpty()) {
            end();
            return;
        }
        sendMessage(itemQueue.element().message());
    }
    @Override
    public void destroy() {
        if (!endTransaction) {
            sendMessage("Транзакція завершена");
        }
    }
    private void sendMessage(String text) {
        try {
            bot.execute(SendMessage.builder().chatId(chatId).text(text).build());
        } catch (TelegramApiException ex) {
            events.removeIf(chatId, this::equals);
        }
    }
    private void end() {
        events.removeIf(chatId, this::equals);
        try {
            sendleMoney.sendPutMoney(putMoneyBuilder.build());
            sendMessage("Поповнення закінчено успішно");
        } catch (RuntimeException e) {
            e.printStackTrace();
            sendMessage("Поповнення закінчено з помилкою");
        } finally {
            endTransaction = true;
        }
    }
    private record PutMoneyEventItem(
        String message,
        Consumer<String> handle
    ) {

```

```

        void checkAndPut(String value) {
            this.handle.accept(value);
        }
    }
    private Queue<PutMoneyEventItem> itemQueueBuild() {
        LinkedList<PutMoneyEventItem> queue = new LinkedList<>();
        queue.add(new PutMoneyEventItem("Введіть номер карти у форматі -> XXXX XXXX XXXX XXXX",
putMoneyBuilder::cardNumber));
        queue.add(new PutMoneyEventItem("Введіть дату дійсності у форматі -> мм.рр",
putMoneyBuilder::expiryDate));
        queue.add(new PutMoneyEventItem("Введіть cvv2", putMoneyBuilder::cvv2));
        queue.add(new PutMoneyEventItem("Введіть назву карти", putMoneyBuilder::cardName));
        queue.add(new PutMoneyEventItem("Введіть суму поповнення", (v) -> {
            try {
                putMoneyBuilder.value(Long.valueOf(v));
            } catch (NumberFormatException e) {
                throw new PutMoneyException("Некоректне значення суми");
            }
        }));
        return queue;
    }
    public interface SandlerPutMoney {
        void sendPutMoney(PutMoney putMoney);
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        return o != null && getClass() == o.getClass();
    }
    @Override
    public int hashCode() {
        return PutMoneyEvent.class.hashCode();
    }
}

```

Файл ua.telegram.videoproject.events.SurveyEvent.java

```

@RequiredArgsConstructor
public class SurveyEvent implements Event {
    private final HttpCallBack callBack;
    private final TelegramLongPollingCommandBot bot;
    private final TelegramEvents events;
    private final HttpClient httpClient;
    private final HttpRequestParametersMapper requestParametersMapper;
    private Message message;
    private boolean destroyed;
    @Override
    public void emit(Update update) {
        if (isSkip(update)) {
            return ;
        }
        var httpRequestParameters = requestParametersMapper.toHttpRequestParameters(
            callBack,
            update.getCallbackQuery().getData()
        );
        if (RequestMethod.GET.equals(callBack.getMethod())) {
            httpClient.get(httpRequestParameters);
        } else {
            httpClient.post(httpRequestParameters);
        }
        completeSurvey();
    }
    private boolean isSkip(Update update){
        if (!update.hasCallbackQuery() ) {
            return true;
        }
        return
!update.getCallbackQuery().getMessage().getMessageId().equals(message.getMessageId());
    }
    @Override
    public Event setMessage(Message message) {
        this.message = message;
        return this;
    }
    @Override
    public void destroy() {
        try {
            bot.execute(DeleteMessage
                .builder()

```

```

        .chatId(message.getChatId())
        .messageId(message.getMessageId())
        .build());
    } catch (TelegramApiException e) {
        e.printStackTrace();
    } finally {
        destroyed = true;
    }
}
private void completeSurvey() {
    try {
        bot.execute(EditMessageText
            .builder()
            .chatId(message.getChatId())
            .messageId(message.getMessageId())
            .text(message.getText() + " ✅")
            .build());
    } catch (TelegramApiException e) {
        e.printStackTrace();
    }
    events.removeIf(message.getChatId(), this::equals);
    destroyed = true;
}
@Override
public boolean isDestroyed() {
    return destroyed;
}
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    SurveyEvent that = (SurveyEvent) o;
    if (!Objects.equals(callBack, that.callBack)) return false;
    return Objects.equals(message, that.message);
}
@Override
public int hashCode() {
    int result = callBack != null ? callBack.hashCode() : 0;
    result = 31 * result + (message != null ? message.hashCode() : 0);
    return result;
}
}

```

Файл ua.telegram.videoproject.NewTelegramBotApplication.java

```

@EnableCaching
@SpringBootApplication(exclude = {UserDetailsServiceAutoConfiguration.class })
public class NewTelegramBotApplication {
    public static void main(String[] args) {
        SpringApplication.run(NewTelegramBotApplication.class, args);
    }
}

```

Файл ua.telegram.videoproject.repository.JdbcTemplateUserRepository.java

```

public class JdbcTemplateUserRepository implements UserRepository {
    private final static String SELECT_USER = "SELECT chat_id as chatId,number as
phoneNumber,date_registration as dateRegistration,active FROM user ";
    private final static String SELECT_ONE = SELECT_USER + " WHERE " +
        " (:chatId IS NULL OR chat_id = :chatId) AND " +
        " (:phoneNumber IS NULL OR number = :phoneNumber) " +
        " limit 2";
    private final static String SELECT_ALL = SELECT_USER + " WHERE " +
        " (:chatIdsIsNull = true OR chat_id IN(:chatIds)) AND " +
        " (:phoneNumber IS NULL OR number LIKE CONCAT('%',:phoneNumber,'%')) AND " +
        " (:isActive IS NULL OR active = :isActive) AND " +
        " (:dateTo IS NULL OR date_registration < :dateTo) AND " +
        " (:dateFrom IS NULL OR date_registration > :dateFrom) ";
    private static final String UPDATE = "UPDATE user SET number = ?, active = ? WHERE chat_id =
?;";
    private static final String SAVE = "INSERT INTO user(chat_id,number,date_registration,active)
VALUES (?, ?, ?, ?);";
    private final NamedParameterJdbcTemplate namedParameterJdbc;
    private final RowMapper<User> userRowMapper;
    private final JdbcTemplate jdbc;
    public JdbcTemplateUserRepository(NamedParameterJdbcTemplate namedParameterJdbc, JdbcTemplate
jdbc) {
        this.namedParameterJdbc = namedParameterJdbc;
        this.jdbc = jdbc;
    }
}

```

```

        this.userRowMapper = new BeanPropertyRowMapper<>(User.class);
    }
    @Override
    public Optional<User> findOne(UserUniqueFieldFilter filter) {
        if (filter.areAllFieldsNull()) {
            throw new IllegalArgumentException("All params is null");
        }
        Map<String, Object> params = new HashMap<>();
        params.put("chatId", filter.getChatId());
        params.put("phoneNumber", filter.getPhoneNumber());
        return namedParameterJdbc.queryForStream(SELECT_ONE, params, userRowMapper).findFirst();
    }
    @Override
    public List<User> findAll(UserFilter filter) {
        Map<String, Object> params = new HashMap<>();
        params.put("chatIdsIsNull", CollectionUtils.isEmpty(filter.getChatIds()));
        params.put("chatIds", filter.getChatIds());
        params.put("phoneNumber", filter.getPhoneNumber());
        params.put("isActive", filter.getActive());
        params.put("dateTo", Range.to(filter.getDateOfCreation()));
        params.put("dateFrom", Range.from(filter.getDateOfCreation()));
        return namedParameterJdbc.query(SELECT_ALL, params, userRowMapper);
    }
    @Override
    public int update(User user) {
        return jdbc.update(UPDATE, user.getPhoneNumber(), user.isActive(), user.getChatId());
    }
    @Override
    public int save(User user) {
        return jdbc.update(SAVE, user.getChatId(), user.getPhoneNumber(),
            user.getDateRegistration(), user.isActive());
    }
}

```

Файл ua.telegram.videoproject.service.TelegramBotAuthorizationServiceImpl.java

```

@RequiredArgsConstructor
public class TelegramBotAuthorizationServiceImpl implements TelegramBotAuthorizationService {
    private final UserService userService;
    private final ContactReplyKeyboardBuilder contactButtonBuilder;
    @Override
    public boolean isRegisteredUser(Update update) {
        return userService.findOne(
            UserUniqueFieldFilter.builder()
                .chatId(UpdateUtils.getChatId(update))
                .build()
        ).isPresent();
    }
    @Override
    public SendMessage signUp(Update update) {
        if (update.getMessage().getContact() == null) {
            return SendMessage.builder()
                .chatId(update.getMessage().getChatId())
                .text("Please share contact")
                .replyMarkup(contactButtonBuilder.build()).build();
        }
        userService.save(new User(update.getMessage().getChatId(),
            update.getMessage().getContact().getPhoneNumber().replace("+", ""),
            LocalDateTime.now(), true));
        return SendMessage.builder()
            .chatId(update.getMessage().getChatId())
            .replyMarkup(new ReplyKeyboardRemove(true))
            .text("Registration is successful").build();
    }
}

```

Файл ua.telegram.videoproject.service.TelegramUserServiceImpl.java

```

@RequiredArgsConstructor
public class TelegramUserServiceImpl implements TelegramUserService {
    private final static String PHONE_NUMBER_TEXT = "Phone number -> ";
    private final TelegramLongPollingCommandBot bot;
    private final UserService userService;
    private final TelegramEvents events;
    private final ContactReplyKeyboardBuilder contactKeyboardBuilder;
    @Override
    public void editUser(Update update) {
        try {
            var chatId = UpdateUtils.getChatId(update);

```

```

        bot.execute(
            SendMessage.builder()
                .chatId(chatId)
                .text("Please share contact")
                .replyMarkup(contactKeyboardBuilder.build()).build()
        );
        events.put(
            chatId,
            new EditUserEvent(bot, userService, events)
        );
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}

@Override
public void showPhoneNumber(Update update) {
    var chatId = UpdateUtils.getChatId(update);
    var phoneNumber = userService
        .findOne(
            UserUniqueFieldFilter
                .builder()
                .chatId(chatId)
                .build()
        )
        .map(u -> u.getPhoneNumber())
        .orElse("not found");

    try {
        bot.execute(phoneNumberMessage(phoneNumber, chatId));
    } catch (TelegramApiException e) {
        throw new RuntimeException(e);
    }
}

private SendMessage phoneNumberMessage(String phoneNumber, Long chatId) {
    return SendMessage
        .builder()
        .text(PHONE_NUMBER_TEXT + phoneNumber)
        .entities(boldStyle(phoneNumber.length()))
        .chatId(chatId)
        .build();
}

private List<MessageEntity> boldStyle(int textLength) {
    return List.of(
        MessageEntity
            .builder()
            .type("bold")
            .offset(PHONE_NUMBER_TEXT.length())
            .length(textLength)
            .build();
    )
}
}

```

Файл ua.telegram.videoproject.service.TokenAuthenticationService.java

```

public class TokenAuthenticationService implements ApiAuthenticationService {
    private final static String TOKEN_PREFIX = "Bearer";
    private final static String HEADER_STRING = "Authorization";
    private final String secret;
    private final Gson gson;

    public TokenAuthenticationService(String secret) {
        this.secret = secret;
        this.gson = new Gson();
    }

    public Authentication getAuthentication(HttpServletRequest request) {
        String token = request.getHeader(HEADER_STRING);
        if (token != null) {
            // parse the token.
            String userJson = JwtToken
                .parser()
                .setSigningKey(secret)
                .parseClaimsJws(removeTokenPrefix(token))
                .getBody()
                .getSubject();

            // parse the json.
            AuthenticationUser authUser = gson.fromJson(userJson, AuthenticationUser.class);
            return new UsernamePasswordAuthenticationToken(
                authUser,
                null,
                List.of(authUser.getService()));
        }
    }
}

```

```

        return null;
    }
    private String removeTokenPrefix(String token) {
        return token.replace(TOKEN_PREFIX, "");
    }
}

```

Файл ua.telegram.videoproject.service.UserServiceImpl.java

```

@RequiredArgsConstructor
public class UserServiceImpl implements UserService {
    private final UserRepository repository;
    @Override
    @Caching(
        cacheable = {
            @Cacheable(cacheNames = {"users"}, key = "#filter.phoneNumber", condition =
"#filter.phoneNumber!=null"),
            @Cacheable(cacheNames = {"users"}, key = "#filter.chatId", condition =
"#filter.chatId!=null")
        }
    )
    public Optional<User> findOne(UserUniqueFieldFilter filter) {
        return repository.findOne(filter);
    }
    @Override
    public List<User> findAll(UserFilter filter) {
        return repository.findAll(filter);
    }
    @Override
    @CacheEvict(value = "users", key = "#user.phoneNumber")
    public boolean update(User user) {
        return repository.update(user) != 0;
    }
    @Override
    @Caching(
        evict = {
            @CacheEvict(value = "users", key = "#user.phoneNumber"),
            @CacheEvict(value = "users", key = "#user.chatId")
        }
    )
    public boolean save(User user) {
        return repository.save(user) != 0;
    }
}

```

Файл

ua.videoproject.authservice.authorization.service.AuthorizationServiceApplication.java

```

@SpringBootApplication(
    exclude = UserDetailsServiceAutoConfiguration.class
)
@EnableCaching
public class AuthorizationServiceApplication {
    public static void main(String[] args) {
        SpringApplication.run(AuthorizationServiceApplication.class, args);
    }
}

```

Файл

ua.videoproject.authservice.authorization.service.repositories.AuthCodeRepository.java

```

public interface AuthCodeRepository extends
    Repository<AuthCodeEntity, Long>,
    JpaSpecificationExecutor<AuthCodeEntity> {
    AuthCodeEntity save(AuthCodeEntity e);
    default AuthCodeEntity update(AuthCodeEntity e) {
        return save(e);
    }
    @Modifying
    @Transactional
    @Query(value = "UPDATE AuthCodeEntity SET " +
        "active = CASE WHEN :#{#e.active} IS NULL THEN active ELSE :#{#e.active} END," +
        "dateOfCreation = CASE WHEN :#{#e.dateOfCreation} IS NULL THEN dateOfCreation ELSE
:#{#e.dateOfCreation} END, " +
        "value = CASE WHEN :#{#e.value} IS NULL THEN value ELSE :#{#e.value} END, " +
        "userId = CASE WHEN :#{#e.userId} IS NULL THEN userId ELSE :#{#e.userId} END " +

```

```

        "WHERE id = :#{e.id}")
    int merge(@Param("e") AuthCodeEntity e);
}

```

Файл ua.videoproject.authservice.authorization.service.repositories.UserRepository.java

```

public interface UserRepository extends Repository<UserEntity, Long>,
JpaSpecificationExecutor<UserEntity> {
    @CachePut(value = "userEntities", key = "#user.id", condition = "#user.id != null",
        cacheManager = "constantLifetimeCacheManager")
    UserEntity save(@NonNull UserEntity user);
    @CachePut(value = "userEntities", key = "#user.id", condition = "#user.id != null",
        cacheManager = "constantLifetimeCacheManager")
    default UserEntity update(@NonNull UserEntity user) {
        return save(user);
    }
    @Cacheable(value = "userEntities", key = "#id", condition = "#id != null",
        cacheManager = "constantLifetimeCacheManager")
    Optional<UserEntity> findById(@NonNull Long id);
    @CachePut(value = "userEntities", key = "#oldUser.id", condition = "#oldUser.id != null",
        cacheManager = "constantLifetimeCacheManager")
    default UserEntity merge(@NonNull UserEntity newUser, @NonNull UserEntity oldUser) {
        UserEntity user = new UserEntity();
        user.setId(Utils.ifNullElse(UserEntity::getId, newUser, oldUser));
        user.setPhoneNumber(Utils.ifNullElse(UserEntity::getPhoneNumber, newUser, oldUser));
        user.setPassword(Utils.ifNullElse(UserEntity::getPassword, newUser, oldUser));
        user.setNickname(Utils.ifNullElse(UserEntity::getNickname, newUser, oldUser));
        user.setFirstName(Utils.ifNullElse(UserEntity::getFirstName, newUser, oldUser));
        user.setLastName(Utils.ifNullElse(UserEntity::getLastName, newUser, oldUser));
        user.setBirthDate(Utils.ifNullElse(UserEntity::getBirthDate, newUser, oldUser));
        user.setActive(Utils.ifNullElse(UserEntity::getActive, newUser, oldUser));
        user.setSelectedCurrency(Utils.ifNullElse(UserEntity::getSelectedCurrency, newUser,
oldUser));
        user.setUserState(Utils.ifNullElse(UserEntity::getUserState, newUser, oldUser));
        user.setEmail(Utils.ifNullElse(UserEntity::getEmail, newUser, oldUser));
        user.setDateRegistration(Utils.ifNullElse(UserEntity::getDateRegistration, newUser,
oldUser));
        user.setRole(Utils.ifNullElse(UserEntity::getRole, newUser, oldUser));
        return save(user);
    }
}

```

Файл ua.videoproject.authservice.authorization.service.services.AuthServiceImpl.java

```

@Builder
@RequiredArgsConstructor
public class AuthServiceImpl implements AuthService {
    private static final String ERROR_LOGIN_MESSAGE = "Phone number or password is not correct";
    private static final Object SIGN_UP_MONITOR = new Object();
    private final TelegramClient telegramClient;
    private final CodeGenerator codeGenerator;
    private final TokenService tokenService;
    private final UserRepository userRepository;
    private final AuthCodeRepository codeRepository;
    private final UserEntityMapper<SignUpUser> entityMapper;
    private final PasswordEncoder passwordEncoder;
    private final LoginResultUserMapper loginUserMapper;
    private final String confirmUrl;
    @Override
    public boolean signUp(SignUpUser u) {
        if (isBooked(u.getPhoneNumber(), u.getEmail())) {
            return false;
        }
        UserEntity user;
        synchronized (SIGN_UP_MONITOR) {
            if (isBooked(u.getPhoneNumber(), u.getEmail())) {
                return false;
            }
            var userForSave = entityMapper.toUserEntity(u);
            userForSave.setPassword(passwordEncoder.encode(userForSave.getPassword()));
            user = userRepository.save(userForSave);
            if (user.getId() == null) {
                return false;
            }
        }
        var code = codeRepository.save(buildCode(user.getId()));
        if (code.getId() == null) {
            return false;
        }
    }
}

```

```

        var message = ConfirmMessage
            .builder()
            .code(code.getValue())
            .phoneNumber(user.getPhoneNumber())
            .callbackUrl(confirmUrl)
            .build().toMessage();
        return telegramClient.sendSurveyMessage(message);
    }
    public AuthCodeEntity buildCode(Long userId) {
        return new AuthCodeEntity(null, codeGenerator.create(), LocalDateTime.now(), true,
            userId);
    }
    @Override
    public LoginResult login(LoginUser u) {
        var user = userRepository.findOne(
            UserSearchCriteria
                .builder()
                .userState(UserState.REGISTERED)
                .phoneNumberEq(u.getPhoneNumber())
                .active(true)
                .build()
        ).orElseThrow(() -> new LoginException(ERROR_LOGIN_MESSAGE));
        if (!passwordEncoder.matches(u.getPassword(), user.getPassword())) {
            throw new LoginException(ERROR_LOGIN_MESSAGE);
        }
        var token = tokenService.generate(Token.User
            .builder()
            .id(user.getId())
            .phoneNumber(user.getPhoneNumber())
            .build());
        return LoginResult.builder().token(token)
            .user(loginUserMapper.toLoginUser(user))
            .build();
    }
    @Override
    public boolean isRegistered(String phoneNumber) {
        return userRepository.exists(
            UserSearchCriteria.builder()
                .phoneNumberEq(phoneNumber)
                .userState(UserState.REGISTERED)
                .build()
        );
    }
    @Override
    public boolean isBooked(String phoneNumber, String email) {
        return
            userRepository.exists(BookedUser.builder().email(email).phoneNumber(phoneNumber).build());
    }
    @Override
    public boolean confirmSignUp(String phoneNumber, String codeValue) {
        var codeOpt = codeRepository.findOne(
            AuthCodeSearchCriteria
                .builder()
                .userPhoneNumberEq(phoneNumber)
                .valueEq(codeValue)
                .active(true)
                .dateOfCreation(Range.ofFrom(LocalDateTime.now().minusMinutes(15)))
                .build()
        );
        if (codeOpt.isEmpty()) {
            return false;
        }
        var code = codeOpt.get();
        code.setActive(false);
        codeRepository.update(code);
        var user = userRepository.findById(code.getUserId()).orElseThrow();
        user.setUserState(UserState.REGISTERED);
        return userRepository.update(user) != null;
    }
}

```

Файл ua.videoproject.authservice.authorization.service.services.UserServiceImpl.java

```

@RequiredArgsConstructor
public class UserServiceImpl implements UserService {
    private final UserMapper userMapper;
    private final UserSearchCriteriaMapper criteriaMapper;
    private final UserRepository userRepository;
    @Override

```

```

public Page<User> findBy(UserFilter f) {
    return userRepository.findAll(
        criteriaMapper.toUserSearchCriteria(f),
        criteriaMapper.toPageable(f)
    ).map(userMapper::toUser);
}
@Override
public Optional<User> findById(Long id) {
    return userRepository.findById(id).map(userMapper::toUser);
}
}

```

Файл ua.videoproject.authservice.authorization.service.services.UuidCodeGenerator.java

```

@Builder
@RequiredArgsConstructor
public class UuidCodeGenerator implements CodeGenerator {
    private final RandomGenerator ran;
    private final int codeLength;
    @Override
    public String create() {
        StringBuilder builder = new StringBuilder();
        do {
            builder.append(UUID.randomUUID().toString().replaceAll("-", ""));
        } while (builder.length() < codeLength);
        ran.ints()
            .limit(Math.round(codeLength * 35.0 / 100.0))
            .map(i -> Math.abs(i % codeLength))
            .distinct()
            .forEach(i -> builder.setCharAt(i, Character.toUpperCase(builder.charAt(i))));
        return builder.substring(0, codeLength);
    }
}

```

A.3 Основний код інтерфейсу користувача

Файл ActorPage.jsx

```

import {useParams} from "react-router-dom";
import ActorDetails from "../components/actors/ActorDetails";
import {useEffect, useState} from "react";
import Center from "../components/layout/Center";
import Loader from "../components/util/Loader";
import {fetchActors, fetchVideoContent} from "../clients/video.content.client";
import {VideoContentFilter} from "../clients/objects/VideoContentFilter";
import VideoContentNames from "../components/video/VideoContentNames";

const containerStyle = {
    display: "flex",
    justifyContent: "flex-start",
    flexDirection: "column",
    rowGap: "10px"
}

const ActorPage = () => {
    const {actorId} = useParams()
    const [actor, setActor] = useState(null);
    const [actorFilms, setActorFilms] = useState(null);

    useEffect(() => {
        async function getActors(id) {
            const respond = await fetchActors({

```

```

        actorIds: id
      });

      respond.ifNotError(d => setActor(d.content[0]))
    }

    getActors(actorId)
  }, [actorId])

  useEffect(() => {
    async function getFilmsByActorId(id) {
      const res = await
fetchVideoContent (VideoContentFilter.builder().withActorIds(id).build());

      res.ifNotError(d => setActorFilms(d.content))
    }

    getFilmsByActorId(actorId)
  }, [actorId])

  return (
    <Center useFreeHeightSpace={true}>
      <div style={containerStyle}>
        {
          actor ?
            <ActorDetails {...actor}/>
          :
            <Loader/>
        }
        {
          actorFilms?
            <VideoContentNames films={actorFilms} text={"Відомий за: "}/>
          :
            <Loader/>
        }
      </div>
    </Center>
  )
}
export default ActorPage

```

Файл FindVideoContentPage.jsx

```

import SearchWithFilter from "../components/util/SearchWithFilter";
import VideoContentFilter from "../components/filters/VideoContentFilter";
import formDataUtils from "../utils/form.data.utils";
import _ from "lodash";
import {useEffect, useMemo, useRef, useState} from "react";
import Modal from "../components/modals/Modal";
import VideoPosterContainer from "../components/video/VideoPosterContainer";
import {fetchVideoContent} from "../clients/video.content.client";
const initState = []
const initFilter = {
  init: 1

```

```

}

const FindVideoContentPage = () => {
  const [videoContent, setVideoContent] = useState(initState)
  const [isOpenFilter, setOpenFilter] = useState(false)

  const openFilter = () => {
    setOpenFilter(true)
  }

  const find = (f) => {
    fetchVideoContent(f)
      .then(r => r.ifNotError(d => {
        console.log(d.content)
        setVideoContent(d.content);
      })))
  }

  useEffect(() => {
    find(initFilter)
  }, [])

  const handleSubmit = (e) => {
    e.preventDefault()
    let filter = _.pickBy(formDataUtils.toObject(e.target) ?? {}, Boolean)
    find(filter)

    console.log(filter)
  }

  return (
    <form onSubmit={handleSubmit} >

      <SearchWithFilter openFilter={openFilter} text={"Назва"} name={'vcName'} />

      <VideoPosterContainer videoContents={videoContent} />

      <Modal active={isOpenFilter} setActive={setOpenFilter}>
        <VideoContentFilter />
      </Modal>
    </form>
  )
}

export default FindVideoContentPage

```

Файл LoginPage.jsx

```

import Center from "../components/layout/Center";
import Login from "../components/auth/Login";

const LoginPage = () => {
  return (
    <Center useFreeHeightSpace={true}>
      <Login />
    </Center>
  )
}

export default LoginPage

```

Файл OfficePage.jsx

```
import {useSelector} from "react-redux";
import Loader from "../components/util/Loader";
import {useEffect, useState} from "react";
import {fetchVideoContent} from "../clients/video.content.client";
import {VideoContentFilter} from "../clients/objects/VideoContentFilter";
import BoughtVideoContent from "../components/video/BoughtVideoContent";

const container = {
  display: 'flex',
  justifyContent: 'flex-start',
  flexDirection: 'column',
  rowGap: "10px",
  alignItems: 'center',
  paddingTop: '10px'
}

const notBoughtContentMessage = {
  color: 'var(--main-blue-color)',
  fontSize: "32px"
}

const OfficePage = () => {
  const {vcIds} = useSelector(s => s.boughtVcIds)
  const [boughtContent, setBoughtContent] = useState(null);

  useEffect(() => {
    if(vcIds.length){
      fetchVideoContent(VideoContentFilter.builder().withVcIds(vcIds).build())
        .then(d => d.ifNotError(d => setBoughtContent(d.content)))
    }else {
      setBoughtContent([])
    }
  },[vcIds])

  return (
    <div style={container}>
      {
        getContent(boughtContent)
      }
    </div>
  )
}

export default OfficePage
function getContent(boughtContent){
  if(!boughtContent){
    return <Loader/>;
  }
  if(!boughtContent.length){
    return <p style={notBoughtContentMessage}>Немає куплиного контенту</p>;
  }
}
```

```

    return boughtContent.map(c => <BoughtVideoContent {...c} key={c.id}/>);
  }

```

Файл ProfilePage.jsx

```

import React, {useEffect, useState} from 'react';
import Center from "../components/layout/Center";
import Profile from "../components/profile/Profile";
import {useSelector} from "react-redux";
import {fetchWallet} from "../clients/video.content.client";

function ProfilePage() {
  const user = useSelector(s => s.auth.user);
  const token = useSelector(s => s.auth.tokenValue);
  const [wallet, setWallet] = useState(null)

  useEffect(() => {
    fetchWallet(token)
      .then(r => r.ifNotError(d => setWallet(d)))
  }, [])

  return (
    <Center useFreeHeightSpace={true} >
      <Profile user={user} wallet={wallet}/>
    </Center>
  );
}
export default ProfilePage;

```

Файл SignUpPage.jsx

```

import SignUp from "../components/auth/SignUp";
import Center from "../components/layout/Center";
import ConfirmSignUp from "../components/auth/ConfirmSignUp";
import {useEffect, useState} from "react";
import {fetchIsRegistered} from "../clients/auth.client";
import {useNavigate} from "react-router-dom";

const TEN_SECONDS = 10000;
const SignUpPage = () => {
  const [phoneNumber, setPhoneNumber] = useState(null)
  const navigate = useNavigate()

  const nextComponent = (phoneNumber) => {
    setPhoneNumber(phoneNumber)
  }

  useEffect(() => {
    if(phoneNumber === null){
      return
    }
    const interval = setInterval(() => {
      pinkRegistered(phoneNumber)
    }, TEN_SECONDS)
  }, [phoneNumber])
}

```

```

        .then(r => r === true ? navigate('../authorization/login', {replace: true}) :
null)

        }, TEN_SECONDS);
        return () => clearInterval(interval);
    },[navigate, phoneNumber])

    return (
        <Center useFreeHeightSpace={true}>
            {
                !phoneNumber ?
                    <SignUp next={nextComponent}/> :
                    <ConfirmSignUp/>
            }
        </Center>
    )
}
export default SignUpPage
async function pinkRegistered(phoneNumber){
    const res = await fetchIsRegistered(phoneNumber)
    return !res.isError && `${res.data}` === 'true';
}

```

Файл VideoContentPage.jsx

```

import VideoInfo from "../components/video/VideoInfo";
import {useParams} from "react-router-dom";
import {useEffect, useState} from "react";
import {fetchVideoContentById} from "../clients/video.content.client";
import Modal from "../components/modals/Modal";
import BuyVideoContent from "../components/video/BuyVideoContent";
import Loader from "../components/util/Loader";
import ErrorAlert from "../components/modals/ErrorAlert";
import EstimateVideoContent from "../components/video/EstimateVideoContent";

const boxStyle = {
    display: 'flex',
    justifyContent: 'space-around',
    marginTop: '6px' ,
    paddingBottom: '6px'
}

const VideoContentPage = () => {
    const {id} = useParams();
    const [isError, setError] = useState(false)
    const [errorMessage, setErrorMessage] = useState(false)
    const [isOpenBuyModal, setOpenBuyModal] = useState(false)
    const [isOpenEstimateModal, setOpenEstimateModal] = useState(false)
    const [vc, setVc] = useState(null)
    const showError = (message) => {
        setErrorMessage(message)
        setError(true)
    }

    const handleModalBuy = () => {
        setOpenBuyModal(true)
    }

```

```

    }
    const openEstimate = () =>{
      setOpenEstimateModal(true)
    }
    useEffect(() => {
      const getVc = async () => {
        const res = await fetchVideoContentById(id)
        if (!res.isError && res.data) {
          setVc(res.data)
        } else {
          showError(res.data)
        }
      }
      getVc();
    }, [])

    return (
      <div style={boxStyle}>
        {
          waitFetch(vc,handleModalBuy,openEstimate)
        }
        <ErrorAlert active={isError} setActive={setError}>
          {errorMessage}
        </ErrorAlert>
        <Modal setActive={setOpenBuyModal} active={isOpenBuyModal} modalStyle={{
          backgroundColor: "#00000000"
        }}>
          <BuyVideoContent name={vc?.name} price={vc?.price} id={vc?.id} showError={ (m) =>
            {
              setOpenBuyModal(false)
              showError(m)
            }} onClose={() => {setOpenBuyModal(false)}}
          </>
        </Modal>
        <Modal setActive={setOpenEstimateModal} active={isOpenEstimateModal} modalStyle={{
          backgroundColor: "#00000000",
          overflow:"hidden"}}>
          <EstimateVideoContent vcId={id} close={() => setOpenEstimateModal(false)}>
        </Modal>
      </div>
    )
  }
}
export default VideoContentPage
function waitFetch(vc,handleBuy,openEstimate) {
  if (!vc) {
    return <Loader/>
  }
  return <VideoInfo buyHelper={handleBuy} openEstimate={openEstimate} {...vc}/>
}

```

Файл WatchRoomPage.jsx

```

import {useLocation} from "react-router-dom";
import watchRoomStore from "../stores/watchRoomStore";

```

```

import {Provider, useSelector} from "react-redux";
import VideoContentPlayer from "../components/video/VideoContentPlayer";
import {useEffect, useState} from "react";
import Menu from "../components/watchroom/Menu";
import styles from './watch.room.page.module.css'
import {fetchVideoContent} from "../clients/video.content.client";
import {VideoContentFilter} from "../clients/objects/VideoContentFilter";
import PutVideoContent from "../components/watchroom/PutVideoContent";
import Modal from "../components/modals/Modal";
import queryString from 'query-string';
import Center from "../components/layout/Center";
import Loader from "../components/util/Loader";
import {playVideoContent} from "../reducers/watch.room.reducer";
import {types} from "../utils/vc.utils";
import History from "../components/watchroom/History";

export const sizePlayer = {
  width: '1120px',
  height: '680px'
}

const WatchRoomPage = () => {
  const { search } = useLocation();
  const { start } = queryString.parse(search);
  const {tokenValue} = useSelector(s => s.auth)
  const {vcIds} = useSelector(s => s.boughtVcIds);
  const [startUrl, setStartUrl] = useState(null)
  const [isOpenPutVC, setOpenPutVC] = useState(false)

  useEffect(() => {
    const findId = vcIds.includes(Number(start)) ? Number(start) : vcIds[0];
    if(!findId){
      return
    }
    fetchVideoContent(VideoContentFilter.builder().withVcIds(findId).build())
      .then(r => {
        r.ifNotError(d => {
          console.log(d.content[0])
          const video = getVideo(d.content[0].videos);

          const vcName = `${d.content[0].name} ${video.number ? `episode`
            `${video.number}` : ''}`.trim()
          watchRoomStore.dispatch(playVideoContent({
            type: types.VIDEO_SERVICE,
            content: {
              id: d.content[0].id,
              url: video.fileUrl,
              duration: video.duration,
              mainImage: d.content[0].mainImage,
              name : vcName
            }
          }))

          setStartUrl(video.fileUrl)
        })
      })
  })
}

```

```

        })
      })
    }, [vcIds, start])
    const handleOpenPutVideoContent = () => {
      setOpenPutVC(true)
    }
    const handleClosePutVideoContent = () => {
      setOpenPutVC(false)
    }
    return (
      <Provider store={watchRoomStore}>
        {
          startUrl ?
            <div className={styles.container}>
              <div className={styles.contentBox}>
                <VideoContentPlayer sizePlayer={sizePlayer} startUrl={startUrl}/>
                <History />
              </div>
              <Menu openPutVc={handleOpenPutVideoContent}
                token={tokenValue}/>
            </div>
            :
            <Center useFreeHeightSpace={true}><Loader/></Center>
          }
          <Modal active={isOpenPutVC} setActive={setOpenPutVC} modalStyle={{
            backgroundColor: '#00000000'
          }}>
            <PutVideoContent onClose={handleClosePutVideoContent} vcIds={vcIds}/>
          </Modal>
        </Provider>
      )
    }
  }
  export default WatchRoomPage
  function getVideo(videos, lang = 'EN'){
    return videos.filter(v => v.language === lang)[0]
  }
}

```

ДОДАТОК Б

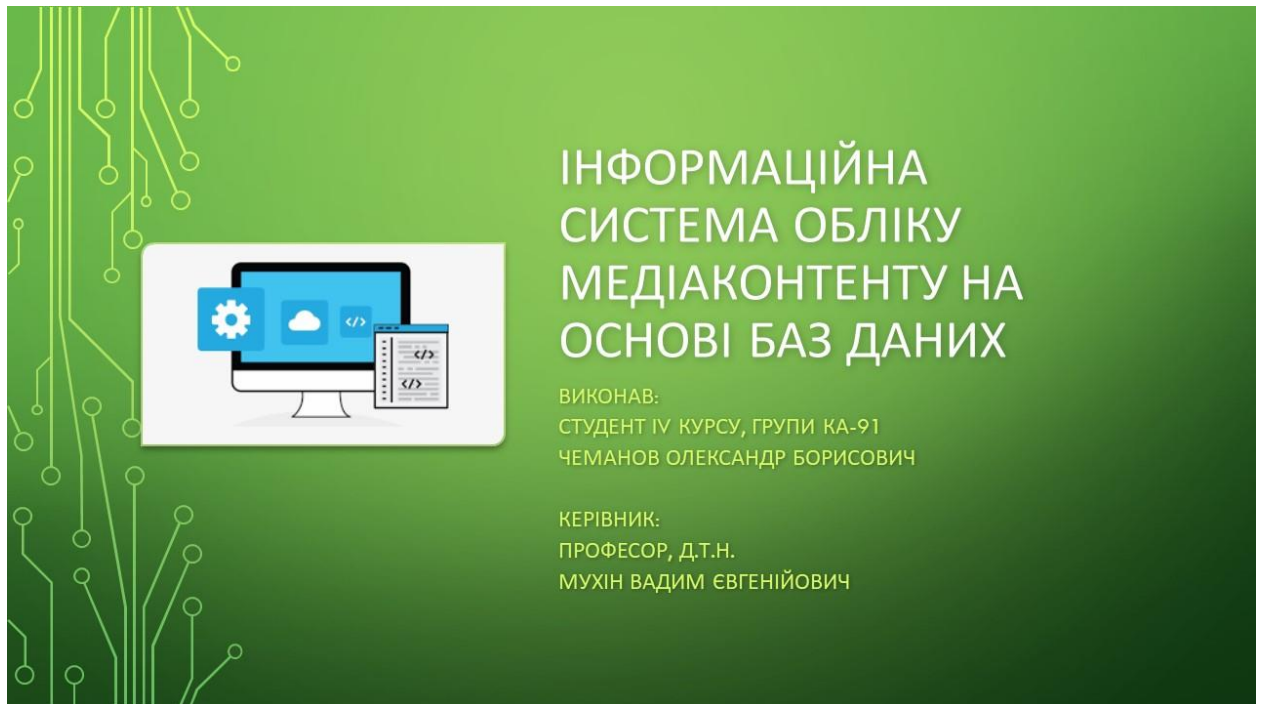


Рисунок Б.1 – Слайд 1

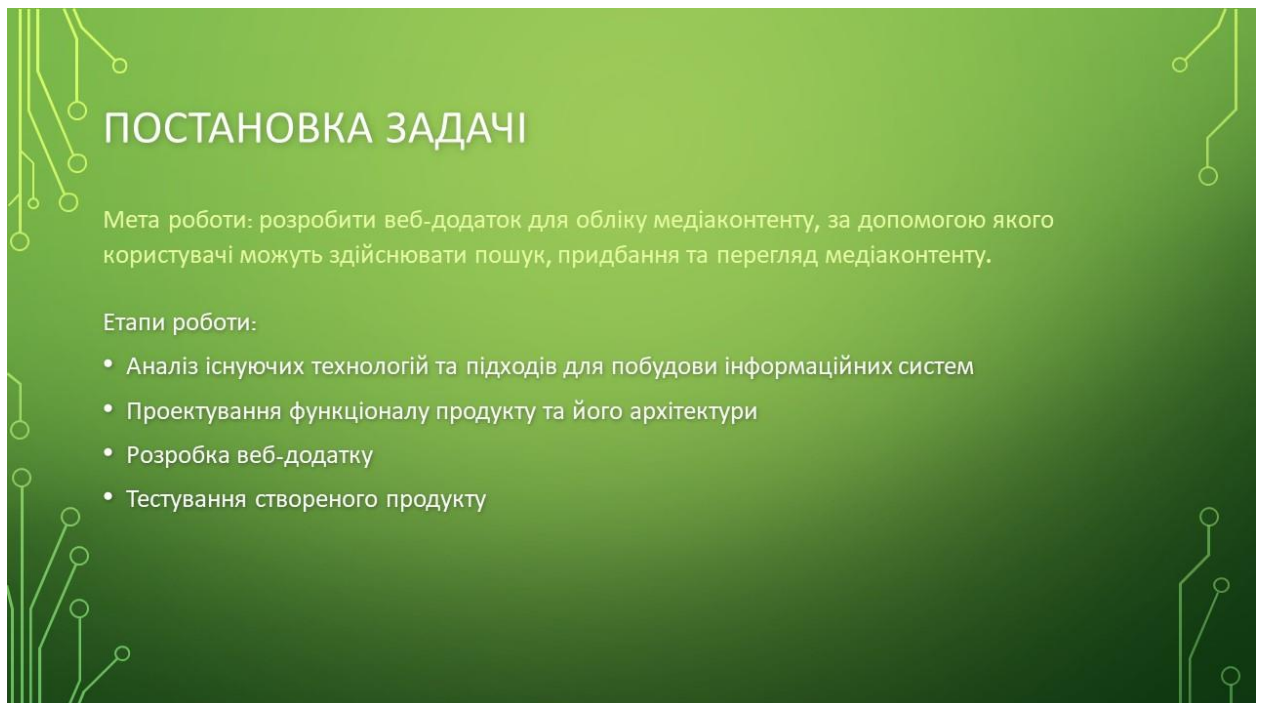


Рисунок Б.2 – Слайд 2

АКТУАЛЬНІСТЬ ПРОБЛЕМИ

Однією з найбільш поширених проблем у сфері медіаконтенту є великий обсяг інформації, який потрібно зберігати, організовувати і швидко знаходити. До медіаконтенту, що є популярним серед користувачів мережі Інтернет можна віднести відеоблоги, фільми, серіали, мультсеріали та подібне.

Системи обліку медіаконтенту дозволяють не тільки швидко знаходити потрібний контент, а ще й переглядати його, виказувати враження за допомогою відгуків та оцінок, отримувати загальну інформацію про контент. До того ж усі види придбаного чи збереженого контенту знаходяться в одному місці, що економить час користувачів при повторному пошуку.

Метою роботи було створити просту, зручну та ефективну систему, яка на відміну від існуючих рішень буде виконувати свої функції без перенавантаження користувача непотрібним функціоналом, а також яка не буде створювати нав'язливі чи відволікаючі рекомендації для користувача.

Рисунок Б.3 – Слайд 3

ОГЛЯД ТЕХНОЛОГІЙ, ПІДХОДІВ ТА РІШЕНЬ

АРХІТЕКТУРИ СИСТЕМ

- Файл-серверна архітектура
- Клієнт-серверна архітектура
- Багаторівнева архітектура

СУБД

- SQLite
- MySQL
- PostgreSQL

ІСНУЮЧІ РІШЕННЯ

- YouTube
- Megogo
- IMDb
- Netflix

Рисунок Б.4 – Слайд 4



Рисунок Б.5 – Слайд 5



Рисунок Б.6 – Слайд 6

ПОРІВНЯННЯ РЕЛЯЦІЙНИХ СУБД ТА ОБ'ЄКТНО-ОРІЄНТОВАНИХ

Переваги реляційних СУБД • Реляційні бази даних є більш популярними, а тому й більш підтримуваними • Для реляційних баз даних існує чіткий математичний апарат, а також існують чіткі стандарти • Існує багато стійких реалізацій реляційного підходу • Швидкість роботи	Переваги об'єктно-орієнтованих СУБД • Дозволяють легко зберігати структури даних у вигляді об'єктів, особливо складних • Підтримка складних ієрархічних структур даних
---	---

Рисунок Б.7 – Слайд 7

ПОРІВНЯЛЬНИЙ ОГЛЯД РЕЛЯЦІЙНИХ СУБД

		
SQLite Переваги • Малий розмір • Портативність • Просте налаштування Недоліки • Обмежений паралелізм • Низький рівень безпеки • Вплив користувача	MySQL Переваги • Простота • Популярність • Безпека • Ефективність та швидкість • Масштабованість Недоліки • Деякі функціональні обмеження • Повільний розвиток	PostgreSQL Переваги • Повна підтримка стандартів SQL • Розширюваність • Активна спільнота розробників Недоліки • Проблеми з пам'яттю • Зменшення популярності

Рисунок Б.8 – Слайд 8

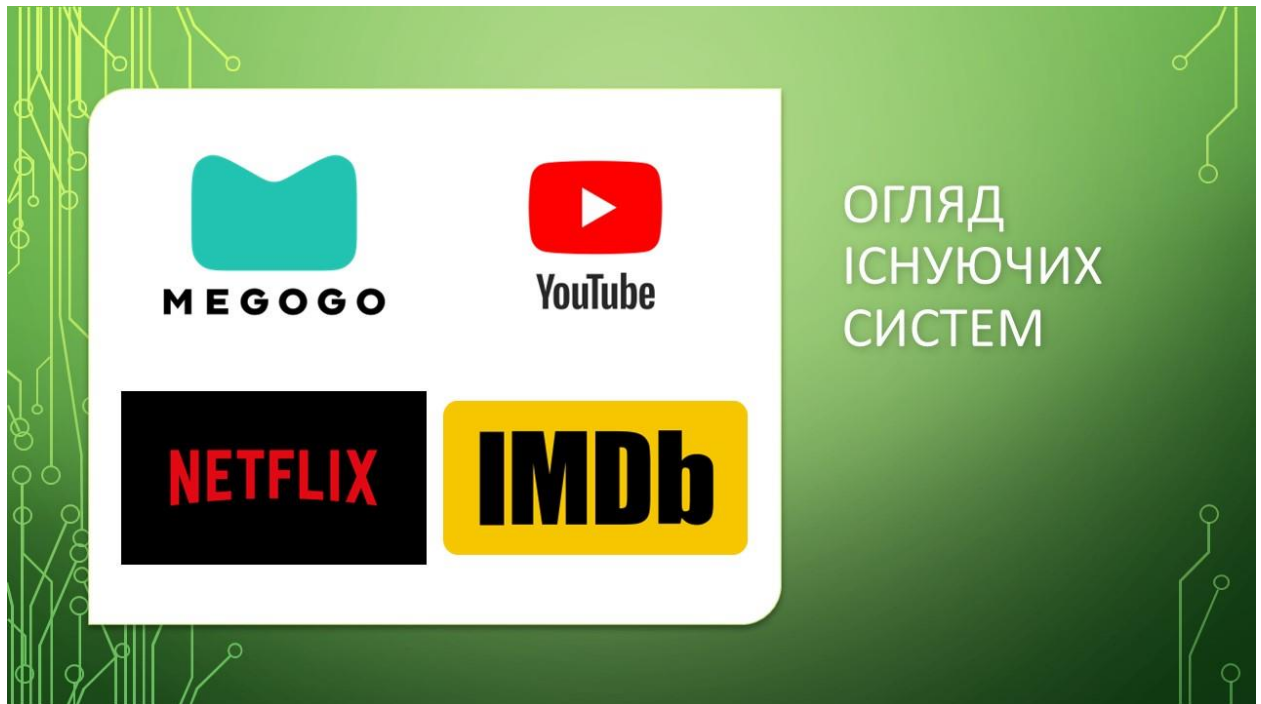


Рисунок Б.9 – Слайд 9

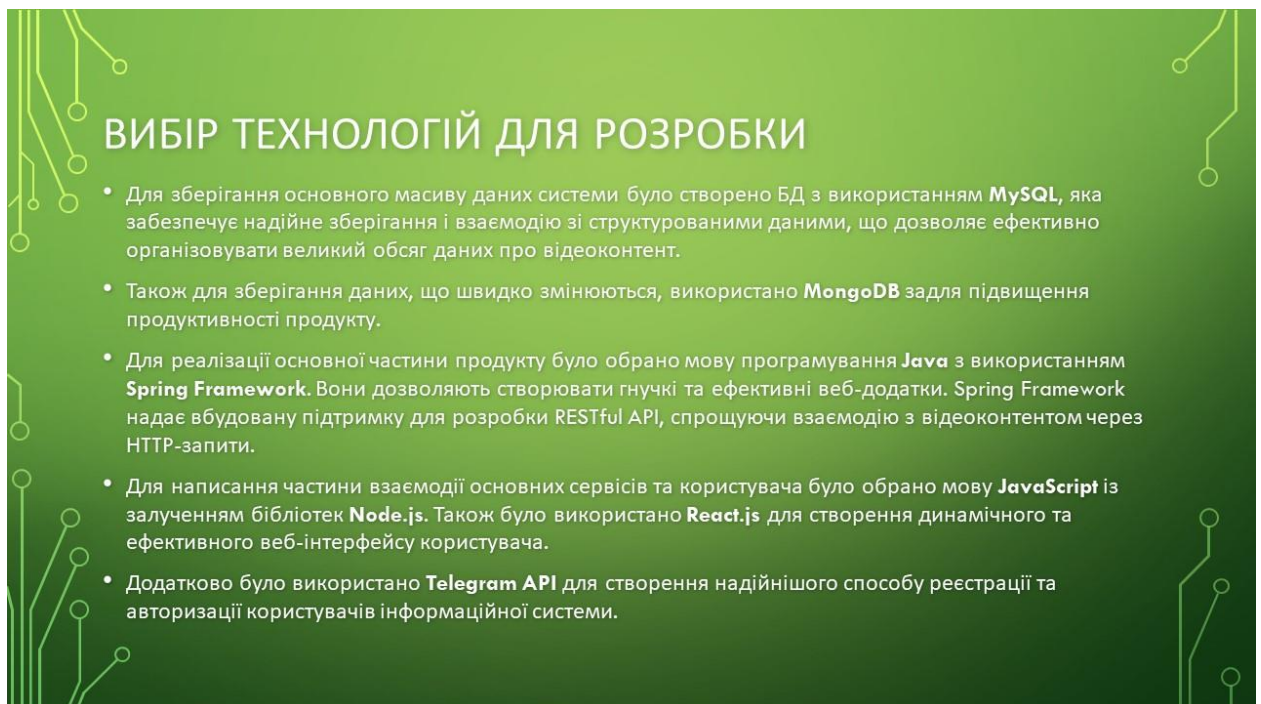


Рисунок Б.10 – Слайд 10

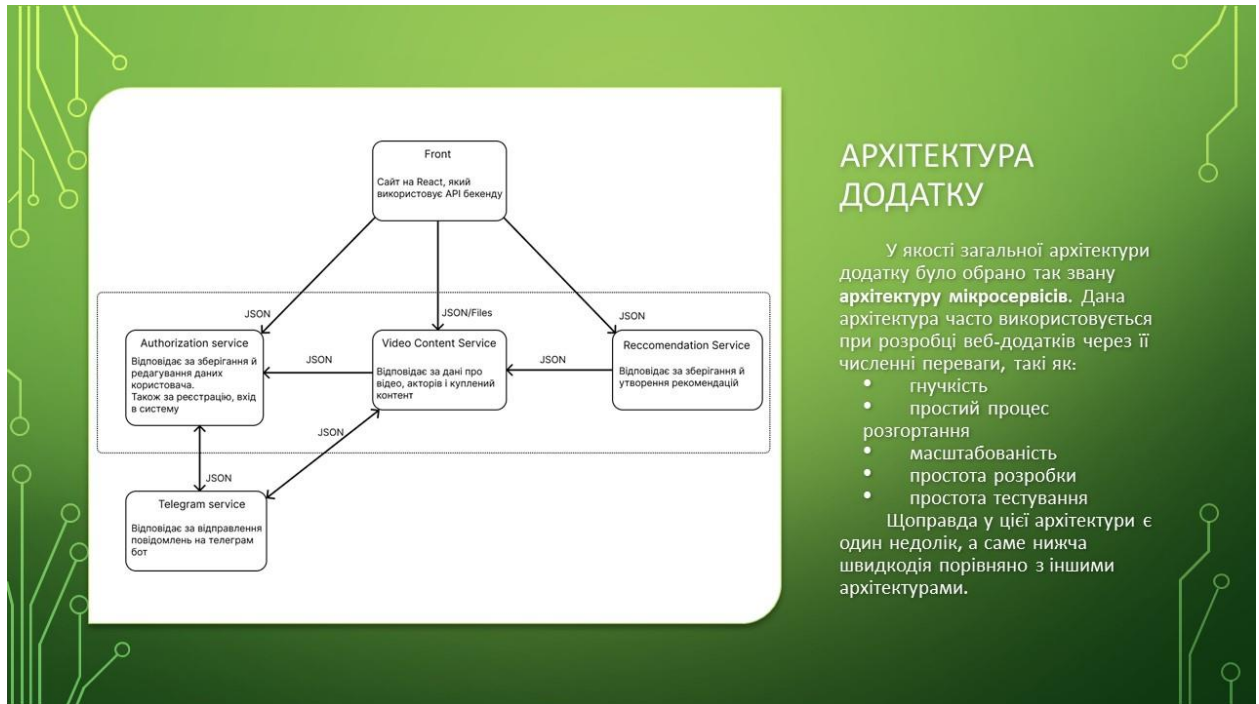


Рисунок Б.11 – Слайд 11

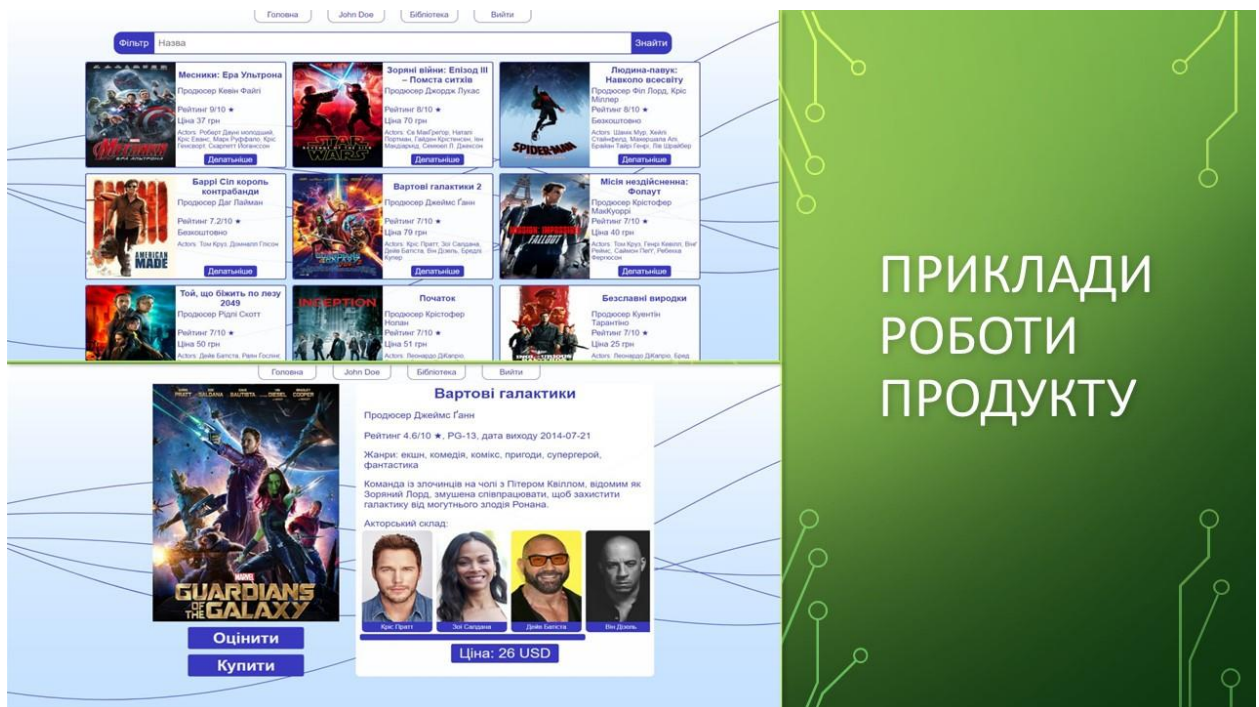


Рисунок Б.12 – Слайд 12

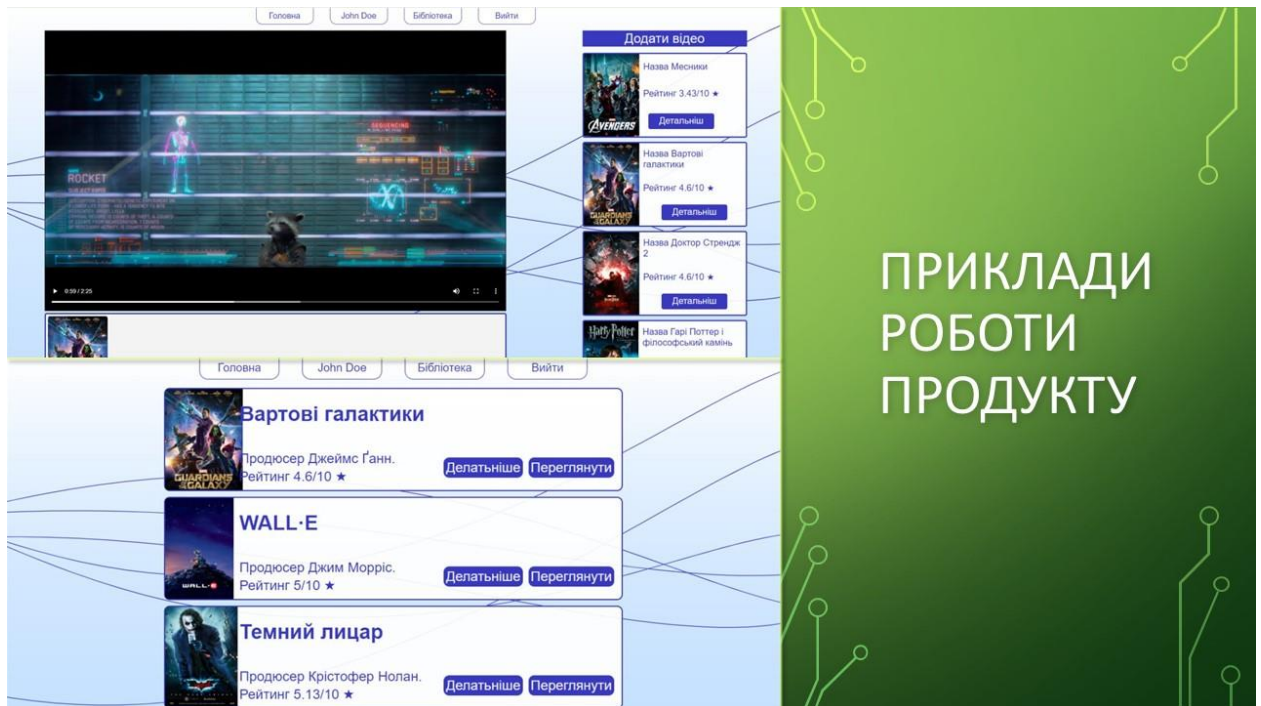


Рисунок Б.13 – Слайд 13

ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ДЛЯ ВИЗНАЧЕННЯ ВАГОВИХ КОЕФІЦІЄНТІВ В СИСТЕМІ ОБРОБЦІ ЗАПИТІВ БД

У цьому експериментальному дослідженні ми зосередимося на системі обробки запитів у розподіленій базі даних з різноманітними компонентами. Наша основна мета полягає в тому, щоб встановити вагові коефіцієнти запитів, які будуть використовуватися в цій системі.

Для проведення експериментів ми використовуємо три таблиці `video\_content`, `purchased\_video\_content`, `actors`. Перша таблиця `video\_content` містить дані про відео контент, друга таблиця `purchased\_video\_content` про куплений відео контент клієнтом, третя таблиця `actors` містить дані про акторів.

До кожної таблиці виконуються 4 серії можливих типів запитів: SELECT, INSERT, UPDATE, DELETE. Для кожного типу вимірюється час запиту, а потім визначається середній час. Потім визначається середній час для типу за усіма таблицями та обчислюється ваговий коефіцієнт для відповідного типу.

Рисунок Б.14 – Слайд 14

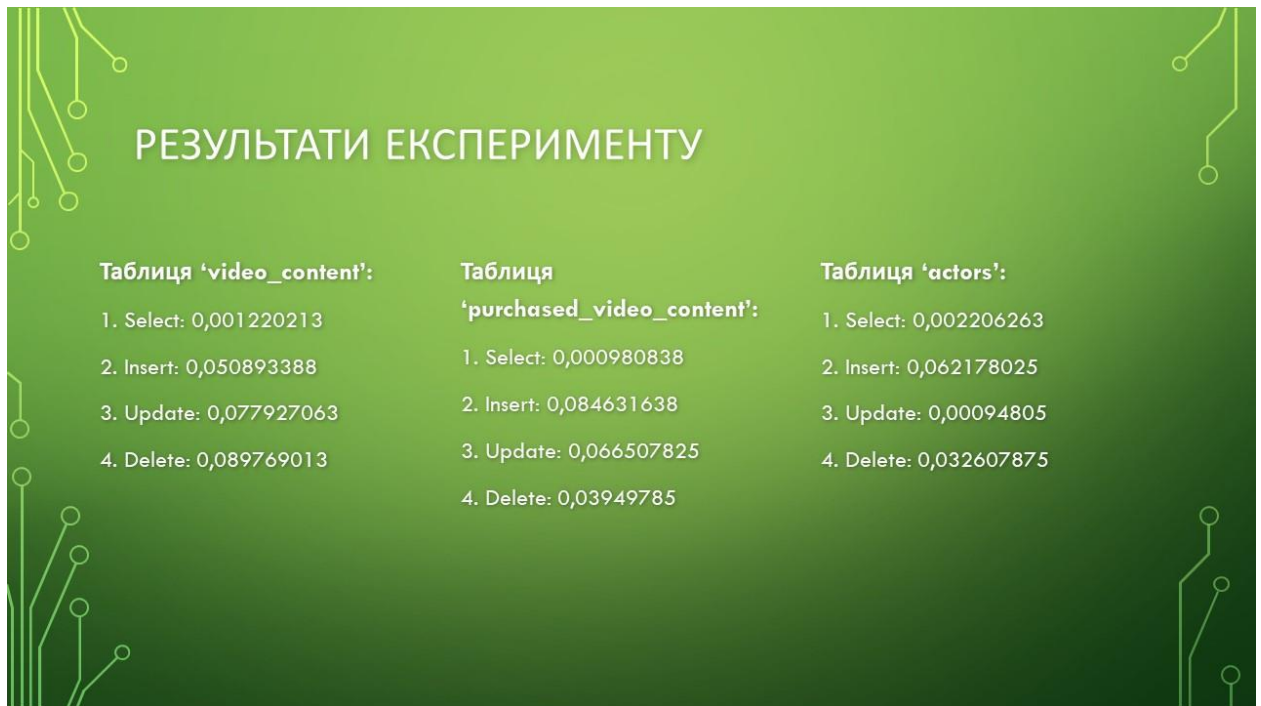


Рисунок Б.15 – Слайд 15

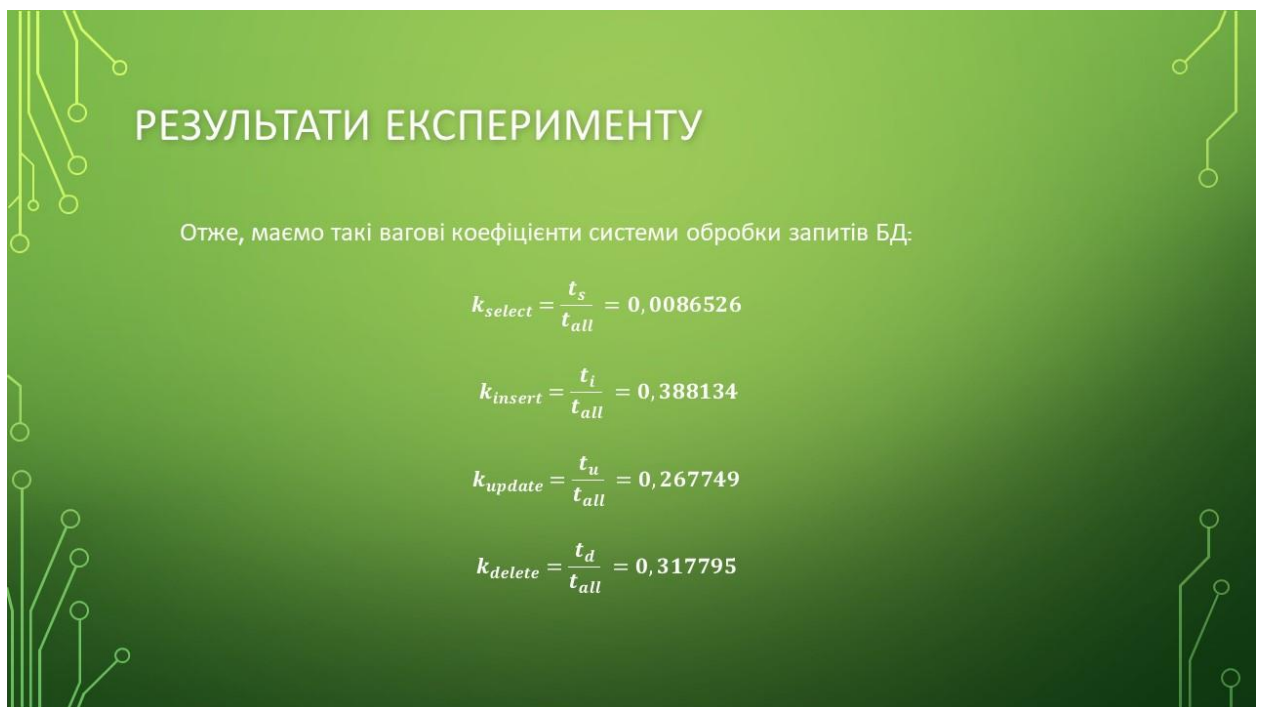


Рисунок Б.16 – Слайд 16

МОЖЛИВІ ВДОСКОНАЛЕННЯ ТА ПОДАЛЬШИЙ РОЗВИТОК

До подальших вдосконалень системи можна віднести наступне:

- збільшення доступних видів медіаконтенту
- реалізація можливості коментування контенту, окрім можливості виставлення оцінок
- розгортання системи на реальних серверних системах
- інтеграція з реальними сервісами оплати для зарахування коштів користувача

Рисунок Б.17 – Слайд 17

ВИСНОВКИ

У ході даної роботи було розроблено систему зручну інформаційну систему, яка надає користувачам зручний доступ до медіаконтенту. Дана система створена з урахуванням існуючих потреба користувачів на основі досвіду схожих існуючих систем.

Було проведено тестування функціоналу системи та її складових елементів. Проведені експериментальні дослідження, які підтвердили ефективність системи обробки запитів гетерогенної розподіленої бази даних, яка використовується даною інформаційною системою.

Рисунок Б.18 – Слайд 18

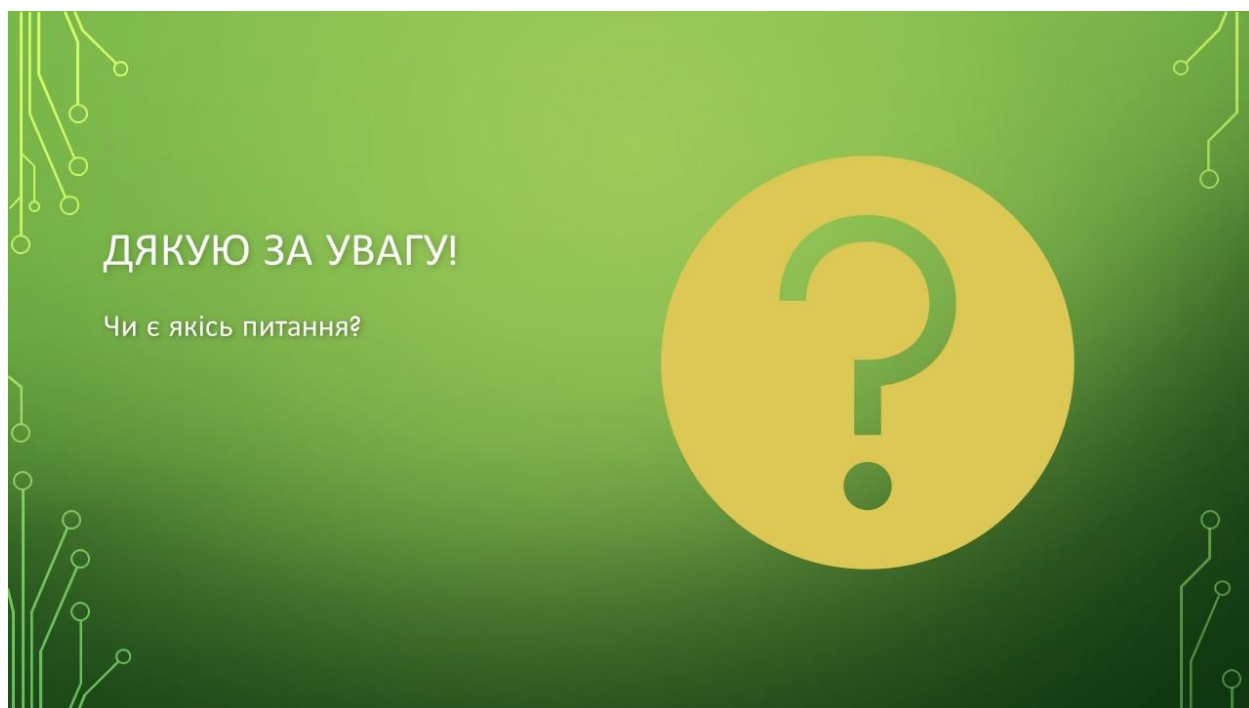


Рисунок Б.19 – Слайд 19