МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**КУРСОВИЙ РОБОТА**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Бази даних»

на тему:

**«БАЗА ДАНИХ ПЛАТФОРМИ ДЛЯ ПРОВЕДЕННЯ КУРСІВ»**

студента II курсу групи ІПЗ-20-4

спеціальності 121 «Інженерія програмного забезпечення»

Кормиша Романа Івановича

(прізвище, ім’я та по-батькові)

Керівник: к.т.н., доц. Сугоняк.І.І.

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Коротун .

(підпис) (прізвище та ініціали) \_\_\_С.М. Кравченко\_ .

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Чижмотря .

(підпис) (прізвище та ініціали)

Житомир – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»

В.о.зав. кафедри ІПЗ

А.В.Морозов

“\_\_\_\_” 2022 р.

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ СТУДЕНТУ

Кормишу Роману Івановичу

1. Тема роботи: Розробка бази даних платформи для проведення курсів

керівник роботи: к.т.н., доц. Сугоняк І.І. .

1. Строк подання студентом: “ ” 2022р.
2. Вихідні дані до роботи: Розробити базу даних для платформи для проведення курсів.
3. Зміст розрахунково-пояснювальної записки(перелік питань. Які підлягають розробці)

* Аналіз інформаційних потоків та особливостей предметної області дослідження
* Проектування бази даних за напрямком курсової роботи
* Реалізація підсистеми обробки даних за напрямком курсової роботи
* Адміністрування баз даних

1. Перелік графічного матеріалу(з точним зазначенням обов’язкових креслень)
2. Презентація до КР: .
3. Посилання на репозиторій: <https://github.com/Raimhal/ISkills.git>
4. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| Завдання видав | Завдання прийняв |
| 1,2,3 | Сугоняк І.І., к.т.н., доц. |  |  |
|  |  |  |  |

1. Дата видачі завдання “ 10 ” лютого 2022 р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсової роботи | Термін виконання етапів роботи | Примітки |
| 1 | Постановка задачі | 10.02.22 | Виконано |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 18.02.22 | Виконано |
| 3 | Формулювання технічного завдання | 19.02.22 | Виконано |
| 4 | Опрацювання літературних джерел | 22.02.22 | Виконано |
| 5 | Проектування структури | 23.02.22 | Виконано |
| 6 | Написання програмного коду | 24.02.22 –  16.06.22 | Виконано |
| 7 | Відлагодження | 16.06.22 –  8.07.22 | Виконано |
| 8 | Написання пояснювальної записки | 10.07.22 –  15.07.22 | Виконано |
| 9 | Захист |  |  |

**Студент***\_\_\_\_\_\_\_\_\_\_\_\_\_\_* Кормиш Р.І

(підпис) (прізвище та ініціали)

**Керівник проекту** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сугоняк І.І.

(підпис) (прізвище та ініціали)

**РЕФЕРАТ**

Завданням на курсовой проект (роботу) було розробити базу даних до платформи проведення курсів.

Пояснювальна записка до курсового проекту (роботи) на тему «Розробка бази даних платформи для проведення курсів» складається з вступу, чотирьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 59 сторінках друкованого тексту. Пояснювальна записка має 7 сторінку додатків. Список використаних джерел містить 10 найменувань і займає 1 сторінку. В роботі наведено 37 рисунків. Загальний обсяг роботи – 67 сторінок.

Ключові слова: POSTGRESQL, REACT, REDUX, ASP.NET, CLOUDINARY, КЛІЄНТ, АДМІН, АВТОРИЗАЦІЯ, КУРСИ, ІНТЕРНЕТ, РОЛІ, ДАНІ, МОДЕЛЬ ДАНИХ, БАЗА ДАНИХ, КЛЮЧОВЕ ПОЛЕ, ЗАПИТ.

**Зміст**

[**ВСТУП** 7](#_Toc108713464)

[**РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ** 8](#_Toc108713465)

[**1.1 Аналіз інформаційних потреб та визначення предметної області дослідження** 8](#_Toc108713466)

[**1.2 Архітектура та засоби реалізації бази даних** 10](#_Toc108713467)

[**Висновки до першого розділу** 19](#_Toc108713468)

[**РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ ПЛАТФОРМИ ДЛЯ ПРОВЕДЕННЯ КУРСІВ** 20](#_Toc108713469)

[**2.1 Аналіз структури інформаційних процесів адміністраторів** 20](#_Toc108713470)

[**2.2 Проектування бази даних платформи курсів** 24](#_Toc108713471)

[**2.3 Розробка математичної моделі та алгоритмів обробки інформації про платформу курсів** 29](#_Toc108713472)

[**Висновки до другого розділу** 31](#_Toc108713473)

[**РОЗДІЛ 3. РЕАЛІЗАЦЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПЛАТФОРМИ КУРСІВ** 32](#_Toc108713474)

[**3.1. Проектування інтерфейсу системи платформи курсів** 32](#_Toc108713475)

[**3.2. Реалізації операцій обробки даних в БД** 45](#_Toc108713476)

[**3.3. Організація звітності системи** 49](#_Toc108713477)

[**Висновки до третього розділу** 52](#_Toc108713478)

[**РОЗДІЛ 4. АДМІНІСТРУВАННЯ БАЗ ДАНИХ** 53](#_Toc108713479)

[**4.1. Розробка заходів захисту інформації в БД** 53](#_Toc108713480)

[**4.2. Налаштування параметрів роботи Postgresql.** 55](#_Toc108713481)

[**Висновки до четвертого розділу** 57](#_Toc108713482)

[**ВИСНОВКИ** 58](#_Toc108713483)

[**СПИСОК ЛІТЕРАТУРИ** 59](#_Toc108713484)

[**ДОДАТКИ** 60](#_Toc108713485)

[Додаток А 61](#_Toc108713486)

**Перелік умовних скорочень**

Рис – рисунок

БД – база даних

КР – курсова робота

CRUD – create, read, update, delete

ІС – інформаційна система

ТЗ – технічне завдання

СУБД – система управління базою даних

# **ВСТУП**

В даній курсовій роботі буде наведено процес побудови клієнтського додатку та розробка бази для платформи для проведення курсів.

**Актуальність теми.** Використання баз даних є однією з характерних рис більшості сучасних інформаційних систем. По своїй суті бази даних є тим, навколо чого і будується інформаційна система будь-якого підприємства. Тому теорії створення та практиці використання баз даних приділяється достатня увага протягом періоду функціонування ІС. Тривалий час основним типом були реляційні бази даних, які на сьогодні вже вважаються класичними. Однак класичність не означає легко та швидко. Хоча на практиці і використовується чимало різноманітних баз даних, але для більшості з них існує велика кількість шаблонних фрагментів коду що, робить створення додатку нудним і призводить до однакових помилок при розобці. Тому для спрощення процесу розробки додатку потрібно знаходити щось зручніше, легше та сучасніше. Цьому завданню і було приділену головну увагу при розробці.

**Метою роботи** є дослідження особливостей проектування та реалізації бази днаих за визначеною темою курсової роботи.

**Завданням на курсову роботу є :**

* аналіз теоретичних засад проектування та реалізації систем на основі баз даних;
* визначення інформаційних потреб предметної області дослідження;
* аналіз напрямку ризиків інформаційних потоків та їх структури;
* проектування бази даних за визначеною предметною областю;
* розробка математичної та алгоритмічної моделі функціонування системи на основі БД;
* реалізація БД та інтерфейсних засобів інформаційної системи.

**Об'єктом дослідження** є методи та засоби проектування баз даних за визначеними предметними областями.

**Предметом дослідження** можливості застосування концепції БД та СУБД для забезпечення інформаційних потреб предметної області.

# **РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ ПОТОКІВ ТА ОСОБЛИВОСТЕЙ ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ**

## **1.1 Аналіз інформаційних потреб та визначення предметної області дослідження**

В даному курсовому проекті за предметну область взято базу даних, яка буде зберігати усю необхідну інформацію про платформу курсів. Ця платформа надає можливість отримувати знання з різних спеціальностей за допомогою мережі Internet, тому деякі дання потрібно зберігати саме в мережі. Щоб користувачі мали змогу одержати для себе більш повну інформацію про курс та про платформу, необхідно створити базу даних, яка зберігатиме усі необхідні дані. А для амінів, потрібно зробити можливість контролю даних та отримання статистичної інформації.

Для коректного виконання запитів на отримання даних, дані повинні вноситись та змінюватись у БД. За це будуть відповідати деякий перелік операцій:

* Реєстрація користувача. Додаток повинен розрізняти користувачів, щоб давати йому доступ до певних даних та функцій, тому потрібно, створити таблицю, з користувачами. Для заповнення цієї таблиці потрібне створення нового користувача, за це і відповідатиме дана операція.
* Вхід користувача. Логічний опис використано у попередньому пункті (функція, що дозволить розрізняти користувачів).
* Реєстрація курсу. Для того, щоб система розуміла, що в ній є потрібно це щось, передавати їй і зберігати про нього інформацію. Таким чином, дана функція буде отримувати новий курс та зберігати його в БД.
* Редагування курсу. Може трапитись ситуація, що потрібно буде змінити якусь інформацію про курс, наприклад при некоректному заповненні даних, тому потрібно реалізувати дану функцію.
* Створення категорій та тем курсів, що дасть змого лекше орієнтуватися в тематиці курсів та надасть можливість зручного пошуку за інтересами.
* Зміна теми курсу, що забезпечить розшинення оновлення даних про курс, яке було наведено вище.
* Створення роздів курсу. Курси мають мати деякі ключові розділи, що користувач міг легко відсліджувати сівй прогрес.
* Редагування розділів та додання відео записів, з можливістю подальшого перегляду та редагування. Ця функція надасть можливість змінювати інформації про певний розділ та додавати відео-матеріал курсу, що поліпсить спийняття інформації.
* Створення коментаріїв. Ця опція необхідна, щоб користувач зміг поділитися важеннями про курс та вплинути як на рейтинг самого курсу, та і на рейтинг його автора.
* Купівля курсів. Щоб зробити можливим отринання коштів за певну інформацю та реалівувати платні курси, адже ми всі знаємо, що найцінніша інформація є платною, дана функція необхідна.
* Перегляд курсів. Дана функція потрібна для того, щоб користувач розумів, які курси наявні для вивчення.
* Перегляд профілю потрібен, щоб користувач міг переглянути або відредактувати дані, які про нього будуть бачитми інші користувачі.
* Пошук та відбір курсів по мультикритеріями (ціна, дата створення, рейтинг і т.д). Завдання даної платформи дозволяти користувачам онлайн отримувати знання по певним спеціальностям. Для цього користувачу портібно дати можливість шукати та аналізувати курси, які він хочу придбати або отримати. Для цього потрібний одночасний пошук по темі курсу та по параметру його назви. Також даний функціонал розширюється на сторінці користувача, де він може відслідковувати саме власностворені курси, та ті які він придбав.
* Статистика. Дуже важливим є аналізування інформації про прибуток за певний проміжок часу, саме цим і буде займатись дана функція, про кількість куплених курсів та про рейтинги авторів курсів в вспіввідношенні до їх кількості власностворених курсів. Також дана функція буде гарно візуалізувати дані, для зручного та швидкого аналізу.
* Створення бекапів. Адміністратору варто мати можливість зробити резервну копію бази щоб зберігти дані та потім мати змогу відновити її у випадку взлому, або пошкодження.
* Відновлення бази. Дана функція буде очищати таблиці поточної БД, та заповнювати їх даними з обраного бекапу.

## **1.2 Архітектура та засоби реалізації бази даних**

Аналіз та вибір СУБД проведемо з урахуванням того, що число клієнтських місць становить від 30 до 500, а доступ до даних має бути максимально ефективним. Обчислювальна техніка працюватиме під керівництвом ОС Windows і Linux. В таблицях наведено порівняльні характеристики СУБД.

Таблиця 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | Microsoft SQL Server | MongoDB | Postgresql |
| Робота під керівництвом ОС | Задовільно | Добре | Добре |
| Складність | Відмінно | Добре | Відмінно |
| Доступ до даних | Відмінно | Відмінно | Відмінно |
| Використання у  великих проектах | Відмінно | Добре | Відмінно |
| Використання у  малих проектах | Добре | Відмінно | Відмінно |
| Схема даних | Відмінно | Відмінно | Добре |
| Підтримувані мови  програмування | Відмінно | Відмінно | Відмінно |
| Підключення до  Web | Відмінно | Відмінно | Відмінно |
| Графічні інструменти | Відмінно | Відмінно | Відмінно |
| Тригери | Відмінно | Добре | Відмінно |
| Паралелізм | Відмінно | Відмінно | Відмінно |
| Одночасний  доступ декількох  користувачів | Добре | Добре | Відмінно |
| Обробка даних мультимедіа | Слабо | Добре | Відмінно |
| Пошук за регулярними виразами | Добре | Відмінно | Відмінно |
| Підтримка об’єктно-орієнтованої  парадигми | Відмінно | Відмінно | Відмінно |
| Засоби аналізу | Відмінно | Відмінно | Відмінно |

Пояснення таблиці:

MongoDB є провідною базою даних документів . Яка побудована на розподіленій масштабованій архітектурі та стала комплексною хмарною платформою для керування та доставки даних у програми. MongoDB обробляє транзакційні, операційні та аналітичні навантаження.

MongoDB зберігає дані як документи у двійковому представленні під назвою BSON (Binary JSON). Поля можуть відрізнятися від документа до документа; немає необхідності оголошувати системі структуру документів – документи описуються самі. Якщо до документа потрібно додати нове поле, це поле можна створити, не впливаючи на всі інші документи в колекції, не оновлюючи центральний системний каталог, оновлюючи ORM і не переводячи систему в автономний режим. За бажанням перевірку схеми можна використовувати для забезпечення контролю керування даними над кожною колекцією.

PostgreSQL є прикладом добре керованого проекту з відкритим кодом.

PostgreSQL називає себе системою об’єктно-реляційної бази даних з відкритим кодом. Це база даних SQL, яка має певні стратегії для обробки індексування, підвищення паралельності та впровадження оптимізацій і підвищення продуктивності, включаючи розширене індексування, розділення таблиць та інші механізми. Об’єктна частина PostgreSQL пов’язана з багатьма розширеннями, які дозволяють включати інші типи даних, такі як об’єкти даних JSON та XML.

MS SQL Server - cистема управління базами даних, яка розробляється корпорацією Microsoft. Як сервер даних виконує головну функцію по збереженню та наданню даних у відповідь на запити інших застосунків, які можуть виконуватися як на тому ж самому сервері, так і у мережі. Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

Модель даних документа MongoDB природним чином відображається на об’єктах у коді програми, що полегшує розробникам вивчення та використання. Документи дають можливість представляти ієрархічні зв’язки для зберігання масивів та інших складніших структур. Документи JSON можуть зберігати дані в полях, у вигляді масивів у вигляді вкладених під документів. Таким чином пов’язану інформацію можна зберігати разом для швидкого доступу до запитів за допомогою багатофункціональної та виразної мови запитів MongoDB .

Однією з найпотужніших функцій реляційних баз даних, що полегшує написання програм, є транзакції ACID. Деталі того, як визначаються та реалізуються транзакції ACID, містять багато підручників з інформатики. Велика частина дискусій у сфері ІТ стосується рівнів ізоляції в транзакціях бази даних . PostgreSQL за замовчуванням використовує рівень ізоляції для читання та дозволяє користувачам налаштовувати його до рівня ізоляції, який можна серіалізувати.

У реляційній базі даних дані, про які йде мова, моделюватимуться в окремих батьківських і дочірніх таблицях у табличній схемі. Це означає, що оновлення всіх записів одночасно потребуватиме транзакції. У певному сенсі базам даних документів легше реалізовувати транзакції, оскільки вони згруповують дані в документі, а запис і читання документа є операцією для якої не потрібна транзакція з кількома документами. Одне або кілька полів можна записати за одну операцію, включаючи оновлення кількох піддокументів і елементів масиву. MongoDB гарантує повну ізоляцію під час оновлення документа. Будь-які помилки ініціюватимуть операцію оновлення до відкату, скасовуючи зміни та гарантуючи, що клієнти отримають узгоджений вигляд документа.

MongoDB також підтримує транзакції бази даних у багатьох документах, тому фрагменти пов’язаних змін можна зафіксувати або відкотити. Завдяки можливості транзакцій з багатьма документами MongoDB є однією з небагатьох баз даних, яка поєднує в собі гарантії ACID традиційних реляційних баз даних зі швидкістю, гнучкістю та потужністю моделі документів.

Транзакції в MongoDB схожі на транзакції, які використтовуються в PostgreSQL. Транзакції в MongoDB складаються з кількох інструкцій із подібним синтаксисом (наприклад, starttransaction and committransaction), тому будь-кому, хто має попередній досвід роботи з транзакціями, легко додати такі конструкції до будь-якої програми.

Властивості ACID у SQL Server забезпечують цілісність даних під час транзакції. SQL ACID є акронімом для атомарності, узгодженості, ізоляції, міцності.

Модель реляційної бази даних, яку використовує PostgreSQL, базується на зберіганні даних у таблицях і подальшому використанні мови структурованих запитів (SQL) для доступу до бази даних.

Щоб це працювало, у PostgreSQL та інших базах даних SQL необхідно створити схему бази даних і встановити зв’язки даних до того, як база даних буде заповнена даними. Пов’язана інформація може зберігатися в окремих таблицях, але пов’язана за допомогою зовнішніх ключів і JOIN. Більшість змін у схемі вимагає процедури міграції, яка може перевести базу даних у автономний режим або знизити продуктивність програми під час її роботи.

Сильною стороною SQL є його потужна та широко відома мова запитів із великою системою інструментів. Проблема використання реляційної бази даних полягає в необхідності заздалегідь визначити її структуру. Змінити структуру після завантаження даних часто дуже складно , що потребує кількох команд розробників, адміністратора баз даних і операцій для тісної координації змін.

MongoDB не використовує SQL за замовчуванням. Натомість для роботи з документами в MongoDB і вилучення даних MongoDB надає власну мову запитів (MQL), яка пропонує майже таку саму потужність і гнучкість, як SQL. Наприклад, як і SQL, MQL дозволяє посилатися на дані з кількох таблиць, перетворювати та агрегувати ці дані, а також фільтрувати для конкретних результатів, які вам потрібні. На відміну від SQL, MQL працює ідіоматично для кожної мови програмування .

Продуктивність запитів у MongoDB можна прискорити, створивши індекси для полів у документах і піддокументах. MongoDB дозволяє індексувати будь-які поля документа, включно з тими, які глибоко вкладені в масиви та піддокументи, і ефективно надсилати запити.

Microsoft SQL Server як мову запитів використовує версію SQL, що отримала назву Transact-SQL. TSQL в основному використовується для написання цілої програми процедури блокової функції, яка визначає, як все має бути завершено, і створення програми, у якій кожна програма надсилає транзактний запит через SQL-сервер і немає взаємодії з базою даних. Він виконується як цілий блок із розширенням мови SQL. У T-SQL використовуються різні типи функцій T-SQL: скалярна функція, функція ранжирування, агрегатна функція, функція набору рядків

MongoDB створено для масштабування. Таким чином, випадки використання, які вимагають надшвидких запитів і величезних обсягів даних або обох, можуть бути оброблені шляхом створення все більших кластерів невеликих машин.

MongoDB базується на розподіленій архітектурі, яка дозволяє користувачам масштабувати багато екземплярів, і доведено, що вона працює над величезними додатками, незалежно від того, вимірюється кількість користувачів або розмір даних . Стратегія масштабування базується на використанні більшої кількості менших машин. Цю стратегію можна розширити до сотень машин.

У PostgreSQL підхід до масштабування залежить від того, чи йдеться про запис чи читання даних. Для записів він базується на архітектурі масштабування, у якій одна основна машина, на якій працює PostgreSQL, має бути максимально потужною для масштабування. Для читання можна масштабувати PostgreSQL шляхом створення реплік, але кожна репліка повинна містити повну копію бази даних.

Техніка, яка робить MongoDB масштабованою, базується на ідеї інтелектуального розподілу (sharding) даних між примірниками в кластері. MongoDB не розбиває документи на частини, документи є незалежними одиницями, що полегшує їх розподіл між декількома серверами, зберігаючи локальність даних.

MongoDB реалізував сучасний набір елементів керування та інтеграції кібербезпеки як для локальної, так і для хмарної версії. Це включає потужні парадигми безпеки, як шифрування на рівні поля на стороні клієнта , що дозволяє шифрувати дані перед тим, як вони надсилаються через мережу до бази даних.

PostgreSQL має повний набір функцій безпеки, включаючи багато типів шифрування. PostgreSQL доступний у хмарі у всіх основних хмарних провайдерів. Хоча це та сама база даних, робочі інструменти та інструменти розробника відрізняються залежно від постачальника хмари, що ускладнює міграцію між різними хмарами. MongoDB Atlas однаково працює в усіх трьох основних хмарних провайдерах, спрощуючи міграцію та багатохмарне розгортання.

Оскільки PostgreSQL покладається на стратегію масштабування, він повинен максимально використовувати доступні обчислювальні ресурси. PostgreSQL робить це за допомогою різноманітних стратегій для індексування та паралельності.

PostgreSQL пропонує різноманітні потужні типи індексів, які найкраще відповідають заданому навантаженню запиту. На додаток до зрілого планувальника запитів і оптимізатора, PostgreSQL пропонує оптимізацію продуктивності, включаючи розпаралелювання запитів на читання, розділення таблиць і своєчасну (JIT) компіляцію виразів.

База даних відповідає широкому спектру стандартів безпеки та має численні функції для підтримки надійності, резервного копіювання та аварійного відновлення, як правило, за допомогою інструментів сторонніх розробників.

База даних MS SQL Server зазвичай виконує краще, ніж база даних Access, особливо з великою базою даних, що містить терабайт. Крім того, SQL Server обробляє запити, які значно швидші та ефективніше обробляють запити паралельно, використовуючи кілька власних потоків в одному процесі для обробки запитів користувачів. За допомогою надійного підключення SQL Server інтегрується з безпекою системи Windows, щоб надати єдиний інтегрований доступ до мережі та бази даних, використовуючи найкращу кількість систем безпеки. У MS SQL Server є дві функції підвищення безпеки на рівні сервера: режим автентифікації Windows і змішаний режим .

Модель безпеки MS SQL Server — це тісна інтеграція між режимом автентифікації Windows у Windows Server і базою даних. Режим автентифікації Windows працює краще в таких сценаріях: коли є контролер домену, з екземплярами бази даних SQL Server Express або LocalDB, коли і програма, і база даних знаходяться в одному машинному середовищі. Змішаний режим включає процес автентифікації за допомогою Windows Server і MS SQL Server. Тут база даних розгортає механізми Windows Password Policy. Розроблена складність Windows Password Policy спрямована на запобігання хакерським атакам.

Крім того, що Microsoft SQL Server та Postgresql зберігають дані реляційно, а MongoDb використовує сховище документів, дані субд мають інші різні характеристики, більше того, ці характеристики можна оцінити, що і зображено у таблиці номер 1.

У всіх СУБД користувачі отримують доступ лише туди, куди йому його надають, що забезпечує безпеку даних.

Схема даних (Схема баз даних — це структура системи баз даних описана мовою, яка підтримується системою керування баз даних і відноситься до організації даних для створення плану побудови бази даних з розподілом на таблиці.). Підтримка фіксованої схеми та розгортання без схеми з можливістю взаємодії між ними реалізована лише не підтримується ні в одній з БД, у той час як в SQL Server, Postgresql та MongoDb потрібно розгортати лише з описаною схемою даних.

По критерію підтримуваних мов програмування не можна виділити когось конкретного, тому що всі представлені СУБД є сучасними і підтримують доволі велику кількість мов програмування, така як: C#, Python, JavaScript, C, Java і т.д

Кожна з перечислених СУБД має додаток, або сайт за допомогою якого можна керувати базою та налаштовувати її(тобто є графічні інтерфейси), кожен з даних інтерфейсів є зручним та простим у використанні(однак іноді тяжко розібратись де знайти певний функціонал та як його заповнити, без допомоги інтернету чи інших джерел інформації).

Підтримка одночасного маніпулювання даними (паралелізм), реалізовано у всіх СУБД, що є великим плюсом для цих СУБД.

Кожна з даних СУБД підтримує одночасний доступ кільком користувачам, що є теж неабиким плюсом. Однак кожна з них має різні ліміти та різне розподілення ресурсів для кількох користувачів, в результаті аналізування цих критеріїв і було виставлено оцінки в таблиці 1. Обробка медіаданих реалізована у всіх СУБД, але в SQL SERVER для цього потрібно вручну створити таблицю, яка буде зберігати не лише сам файл, а й ім’я, щоб не втратити цю інформацію.

Щодо пошуку даних за певним значенням, що має зустрітись в тексті (регулярні вирази), то всі СУБД, є на одному рівні, адже дозволяють гнучно змінювати запити та обробляти їх.

Підртимка ООП, за допомогою мов програмування можна групувати отримані дані в об’єкти або класи. Багато інструментів керування даними покладаються на SQL і програмно генерують складні оператори SQL, щоб отримати потрібну колекцію даних із бази даних. PostgreSQL дуже добре працює в таких контекстах, оскільки це надійна реалізація корпоративного рівня, зрозуміла багатьом розробникам. Крім того, якщо у вас є плоска таблична модель даних, яка не буде часто змінюватися та не потребуватиме масштабування, реляційні бази даних і SQL можуть стати потужним вибором. Але передбачувані переваги SQL мають витрати, які слід враховувати. Недоліком PostgreSQL порівняно з MongoDB та MS SQL Server є те, що він покладається на реляційні моделі даних, які є недружніми до структур даних, з якими розробники працюють у коді, і які мають бути визначені заздалегідь, уповільнюючи прогрес щоразу, коли змінюються вимоги.

Найкращі засоби аналізу на мою думку має MongoDB, адже на їх сайті можна переглянути повну інформацію про БД, так як в SQL MS та Postgresql потрібно використовувати певні скрипти, що не є завжди зручним.

Отже можна зробити висновок, що MS SQL краще підходить для написання доволі складних проєктів, MongoDB – для невеликих та середніх, а PostgreSql можна використовувати в проєкті будь якого розміру.

## **Висновки до першого розділу**

В ході виконання першого розділу було проаналізовано три СУБД також було визначено переваги кожної з СУБД. Також було визначено головні функції програми та приблизну логіку доступу до даних. Також, під час виконання даного роздылу, було вирішено виконувати курсовий проект за допомогою СУБД PostgreSql. Це дозволить реалізувати БД якісно, надійно та швидко, без необхідності ручного налаштування серверу БД.

# **РОЗДІЛ 2. ПРОЕКТУВАННЯ БАЗИ ДАНИХ ПЛАТФОРМИ ДЛЯ ПРОВЕДЕННЯ КУРСІВ**

## **2.1 Аналіз структури інформаційних процесів адміністраторів**

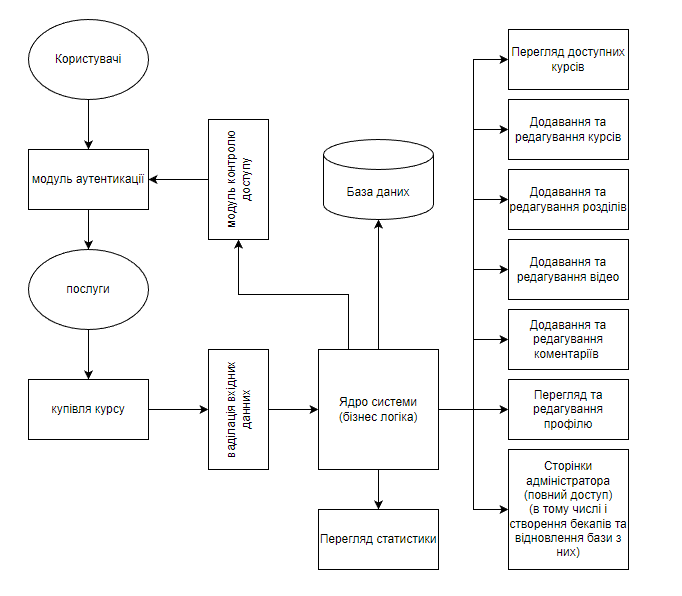


Рис 1. Структура автоматизованої інформаційної системи платформи курсів

Користувачі будуть мати змогу працювати з додатком як після аутентифікації, так і без неї. Наприклад, переглядати деякі дані про курси, коментарі можна буде і без аутентифікації, а ось отримувати якісь більші моливості, наприклад, для звичайного користувача – це буде можливість купівлі курсу, написання коментарію, створення курсу та перегляд куплених чи створених курсів користувача. Адмін матиме повний доступ до зміни будь яких данних, а також до перегляду статистики сайту та створення резурвних копій бази.

Далі наведено приблизні структурні схеми функцій системи.



Рис 2. Структурна схема кількох функцій системи, а саме – реєстрація, логінізація користувача та редагуваня профілю

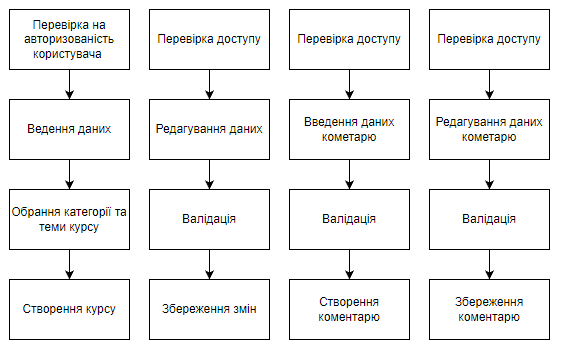


Рис 3. Структурна схема кількох функцій системи, а саме – створення курсу, оновлення курсу, написання коментарію та його редагування

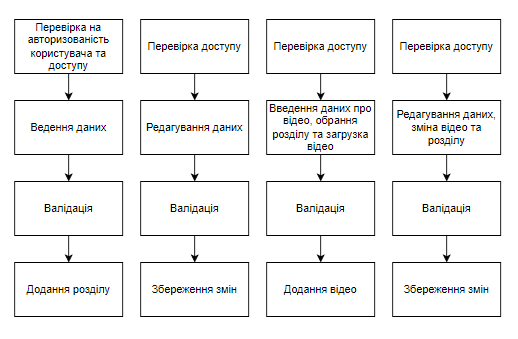


Рис 4. Структурна схема кількох функцій системи, а саме – створення розділу, оновлення розділу, додання відео та його редагування

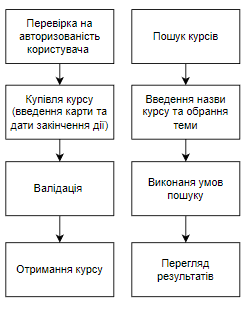


Рис 5. Структурна схема кількох функцій системи, а саме – купівля курсу та пошук курсів

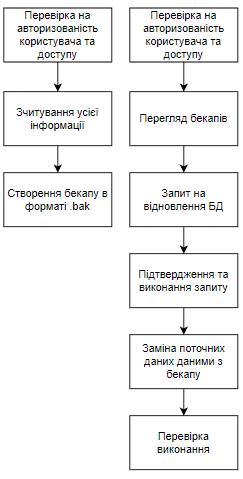


Рис 6. Структурна схема кількох функцій системи, а саме – створення бакапу та відновлення бази

## **2.2 Проектування бази даних платформи курсів**

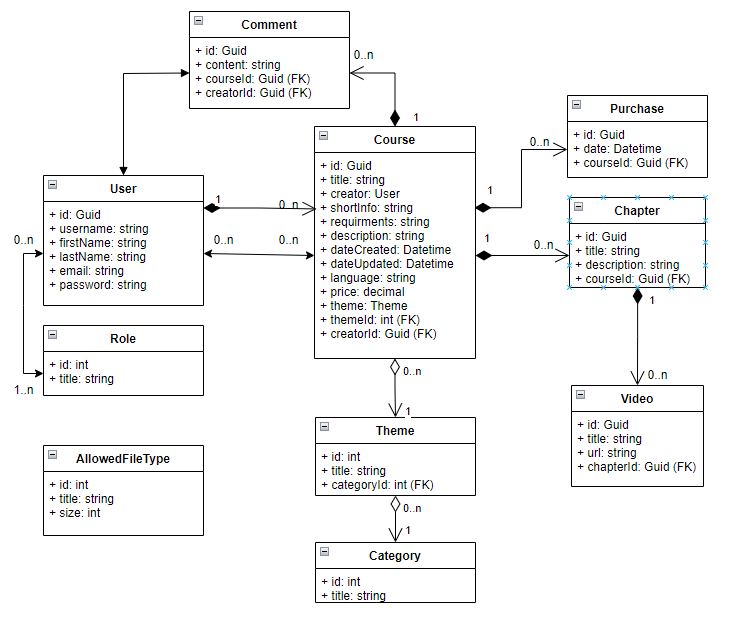


Рис 7. Діаграма “сутність-за’язок” (фізичний рівень) системи

Внаслідок проектування до бази даних включено таблиці:

* Users
* Roles
* Courses
* Themes
* Categories
* Purchases
* Chapters
* Videos
* Comments
* AllowedFileTypes

Для збереження користувачів призначена таблиця "Users ". В дану таблицю вводяться всі необхідні дані користувача. До них належать ім’я, пошта, ім’я, призвіще, псевдонім, рейтинг, хешувальна сіль та хешовавний пароль. Структура таблиці наведена нижче:

Таблиця 2

Структура таблиці “Users”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗК | Опис поля |
| id | Guid | + | - | Id користувача |
| username | string | - | - | Username користучава |
| firstName | string | - | - | Ім’я користучава |
| lastName | string | - | - | Прізвище користучава |
| email | string | - | - | Email користучава |
| password | string | - | - | Захешований пароль користувача |
| salt | string | - | - | Хешувальна сіль |
| rating | double | - | - | Рейтинг користувача |

Для збереження ролей призначена таблиця "Roles". В дану таблицю вводяться всі необхідні дані ролей. До них належить назва ролі. Структура таблиці наведена нижче:

Таблиця 3

Структура таблиці “Roles”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗК | Опис поля |
| id | int | + | - | Id ролі |
| title | string | - | - | Назва ролі |

Для збереження курсів призначена таблиця "Courses". В дану таблицю вводяться всі необхідні дані курсів. До них належать назва, посилання на автора, коротка інформація, опис, вимоги, дата створення, дана оновлення, мова курсу, ціна, посилання на тему. Структура таблиці наведена нижче:

Таблиця 4

Структура таблиці “Courses”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗК | Опис поля |
| id | Guid | + | - | Id курсу |
| title | string | - | - | Назва курсу |
| creatorId | Guid | + | + | Id автора (посилання на таблицю Users) |
| shortInfo | string | - | - | Короткий опис |
| description | string | - | - | Опис |
| requirements | string | - | - | Вимоги |
| dateCreated | DateTime | - | - | Дата створення |
| dateUpdated | DateTime | - | - | Дата оновлення |
| language | string | - | - | Мова курсу |
| price | decimal | - | - | Ціна курсу |
| themeId | int | + | + | Id теми (посилання на таблицю Themes) |
| rating | Double | - | - | Рейтинг курсу |

Для збереження тем призначена таблиця "Themes". В дану таблицю вводяться всі необхідні дані теми. До них належать назва, посилання на категорію. Структура таблиці наведена нижче:

Таблиця 5

Структура таблиці “Themes”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗК | Опис поля |
| id | Guid | + | - | Id теми |
| title | string | - | - | Назва теми |
| categoryId | int | + | + | Id категорії (посилання на таблицю Categories) |

Для збереження категорій призначена таблиця "Categories". В дану таблицю вводяться всі необхідні дані категорій. До них належать назва. Структура таблиці наведена нижче:

Таблиця 6

Структура таблиці “Categories”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗК | Опис поля |
| id | Guid | + | - | Id категорії |
| title | string | - | - | Назва категорії |

Для збереження покупок призначена таблиця "Purchases". В дану таблицю вводяться всі необхідні дані покупок. До них належать дата, посилання на курс. Структура таблиці наведена нижче:

Таблиця 7

Структура таблиці “Purchases”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗК | Опис поля |
| id | Guid | + | - | Id покупки |
| date | DateTime | - | - | Дата покупки |
| courseId | Guid | + | + | Id курсу (посилання на таблицю Courses) |

Для збереження розділів призначена таблиця "Chapters". В дану таблицю вводяться всі необхідні дані розділів. До них належать назва, опис, посилання на курс. Структура таблиці наведена нижче:

Таблиця 8

Структура таблиці “Chapters”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗК | Опис поля |
| id | Guid | + | - | Id розділу |
| title | string | - | - | Назва розділу |
| description | string | - | - | Опис |
| courseId | Guid | + | + | Id курсу (посилання на таблицю Courses) |

Для збереження відео призначена таблиця "Videos". В дану таблицю вводяться всі необхідні дані про відео. До них належать назва, посилання на відео, посилання на розділ. Структура таблиці наведена нижче:

Таблиця 9

Структура таблиці “Videos”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗК | Опис поля |
| id | Guid | + | - | Id відео |
| title | string | - | - | Назва відео |
| url | string | - | - | Посилання на выдео |
| chapterId | Guid | + | + | Id розділу (посилання на таблицю Chapters) |

Для збереження коментрарів призначена таблиця "Comments". В дану таблицю вводяться всі необхідні дані коментарів. До них належать коментар , посилання на користувача, посилання на курс. Структура таблиці наведена нижче:

Таблиця 8

Структура таблиці “Comments”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | ЗК | Опис поля |
| id | Guid | + | - | Id коментаря |
| content | string | - | - | Текс коментаря |
| creatorId | Guid | - | - | Id користувача (посилання на таблицю Users) |
| courseId | Guid | + | + | Id курсу (посилання на таблицю Courses) |

Для збереження доступних розширень призначена таблиця "AllowedFileTypes". В дану таблицю вводяться всі необхідні дані про дозволені розширення завантажуваних файлів та їх розміри. До них належать назва розширення та розмір в МБ. Структура таблиці наведена нижче:

Таблиця 9

Структура таблиці “AllowedFileTypes”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Тип даних | ПК | К | Опис поля |
| id | Guid | + | - | Id розширення |
| title | string | - | - | Назва розширення |
| size | int | - | - | Доволений розмір файлу в МБ |

## **2.3 Розробка математичної моделі та алгоритмів обробки інформації про платформу курсів**

Для збереження користувача в баз даних, юзеру доведеться заповнити відповідну форму реєстрації, де повнино бути коректно заповнено всі необхідні поля, а сам псевдонім, пошта, прізвище та ім'я (не є обов’язковими) та пароль. Після відправки форму, дані обробляються на бекенді, де додається користувацька роль та перевіряється вільність введеного юзернейму та пошти. Виконання даних операцій користувачу буде доступне операція логіну, де він вводить власні дані, а саме пошту та пароль, і після перевірки корекності даних, надається доступ до користувацького запису. Для оновлення даних потрібно буде перейти до сторінки користувача та змінити дані, там же можна буде змамінити фото профілю.

Після логіну у користувача є 2 шляхи взаємодії з додатком. Перши шлях це переглянути наявні курси та придбати ті, які сподобалися. Для цього користувач має натиснути на відповідну кнопку, та ввести коректні дані карти, а саме номер карти та її срок дії. Це буде забезпечуватися за допомогою інтеграції з платіжною системою, а саме Braintree, яка надає такий функціонал. Для цього потрібно генерувати токен покупки на бекенді та передавати в форму на фронтенді.

Після купівлі курсу, користувач має можливість переглядати матеріал, а саме читати інформацію з розділів та переглядати відео-матеріали. Також матиме можливість залишити відгук про курс, що забезпечить відповідна форма. При написанні коментарю, користувач впливає як на рейтинг самого курсу, так і на рейтинг користувача .Також всі придбані курси користувач матиме змогу переглянути на сторінці профілю.

Іншим шляхом взаємодії буде створення власного курсу. Для цього потрібно заповнити відповідну форму та пройти валідацію. Після чого користувача переадресує на сторінку щойноствореного курсу, де він матиме змогу змнінити дані, за допомогою відповідної форми, додати розділи та відеоматеріали і змінити фото курсу. Всі створені курси користувач також матиме змогу переглянути на сторінці власного профілю.

Всі створенні курси має можливість видалити або їх автор або адмнім, після чого курс буде видалено із всіма розділами, відео та коментарями.

Далі розглянемо функції які доступні тільки адміну, а саме перегляд статистики платформи, додання категорій і тем, а також створення резервних копій бази та відновлення з них.

Для перегляду статистики, адмін повинен буде перейти до відповідної сторінки, де зможе переглянути сптатистику за певний період часу та відношення створених курсів до рейтингу їх авторів. Це досягається виконанням перних групувальних функцій які реалізовані на бекенді.

Для додання категорій, тем та дозволених типів файлів, адмін має перейти на відповідні сторінки та заповнити відповідні форми. Кожна форма має перевірку на унікальність поля назви та валідацію даних.

Також потрібно розглянути те, як буде відбуватипя процес створення бекапів. Для створення бекапу адмін має перейти на сторінку з бекапами та натиснути на відповідну кнопку по створенню бекапу. Далі запит надслиється на бекенд, де він виконується, емулюючи роботу з консоллю та виконуючи відповідні команди для створення бекапу бази даних. Після створення бекапу, адмін побачить його в списку доступних для відновлення.

Для відновлення з резервної копії потрібно обрати бекап та виконати віідповідну команду, запит надсилається на бекенд, де він оброблюється, емулюючи роботу з консоллю та виконуючи відповідні команди для рестору БД. Під час цього процесу база очищається та заповнюється даними з бекапу.

## **Висновки до другого розділу**

У результаті виконання даного розділу було проаналізовано інформаційні процеси, спроектовано структуру бази даних, описано приблизну логіку функцій для роботи з базою на клієнті, пояснено деяку логіку обробки, збереження та генерації даних для бази. Також було описано як буде відбуватись побудова та генерація статистичної інформації платформи.

# **РОЗДІЛ 3. РЕАЛІЗАЦЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПЛАТФОРМИ КУРСІВ**

## **3.1. Проектування інтерфейсу системи платформи курсів**

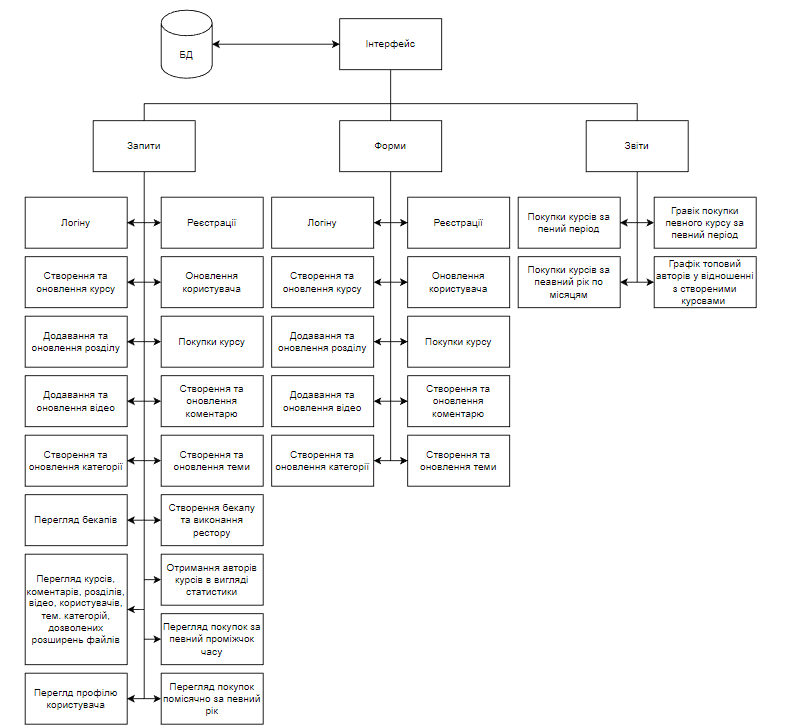
****

Рис 8. Структура інтерфейсу платформи курсів

На рисунку 8 було зображено загальну схему інтерфейсу платформи курсів. Далі будуть зображені форми та сторінки, що показують фінальний результат виконання проектування інтерфейсу додатку. Кожна з форм чи сторінок супроводжується розробкою функцій для роботи з БД.

У більшість випадків побудови форми редагування та додавання були об’єднані з метою зменшення коду. Реакт дозволяє робити умовні побудови первних елементів у компоненті, за рахунок чого і було реалізовано даний підхід розробки інтерфейсу.

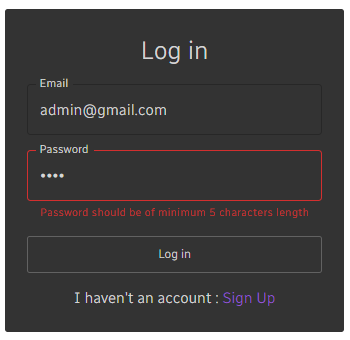


Рис 9. Форма логіну

Форма приймає логін та пароль. При відправці запиту на під’єднання користувача до додатку, відбуається перевірка чи співпадають пароль та логін користувача. Також, пароль має бути більшим ніж 5 символів.

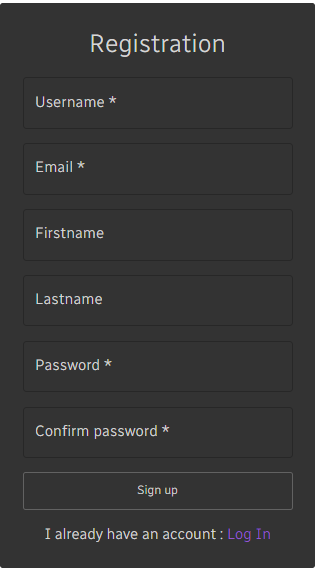


Рис 10. Форма логіну

Форма реєстрації приймає псевдонімі, ім’я, прізвище, пошту, та пароль і повторний пароль. Перераховані поля є обов’язковими, крім імені та прізвища. Перед відправкою форми на сервер відбувається перевірка того, чи співпадають паролі, якщо ж ні, то користувача повідомляють про це і попросять змінити дані, також паролі мають бути довші ніж 5 символів.

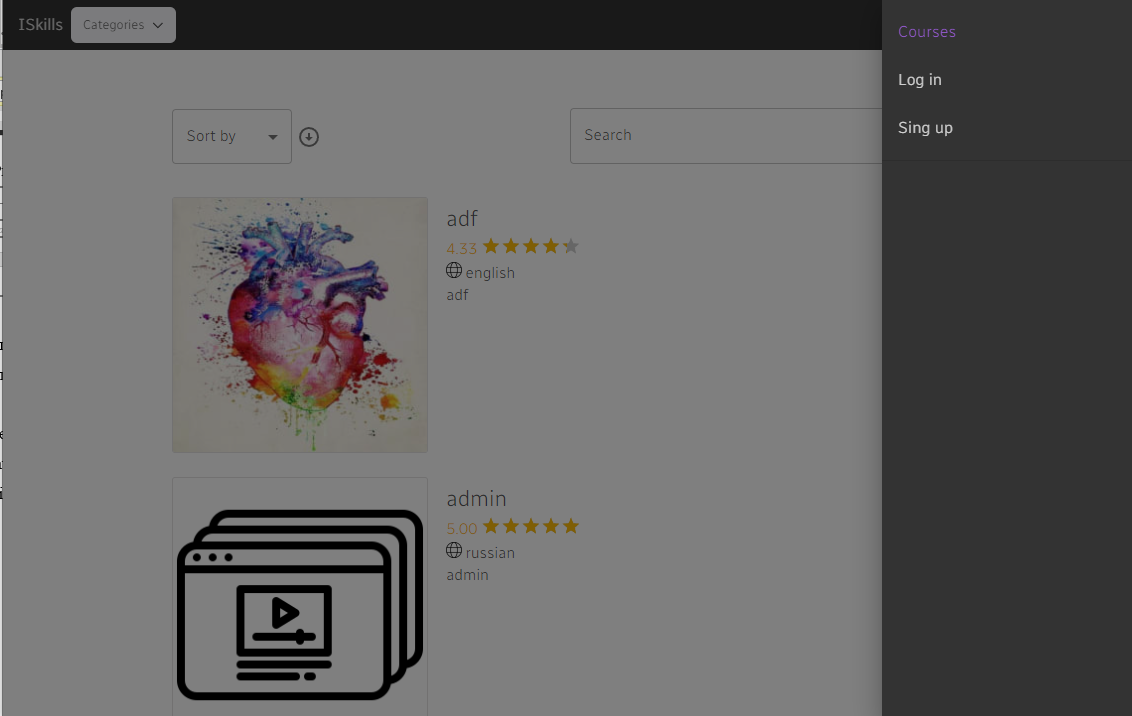


Рис 11. Початкове вікно неавторизованого юзера з боковим меню активованим меню

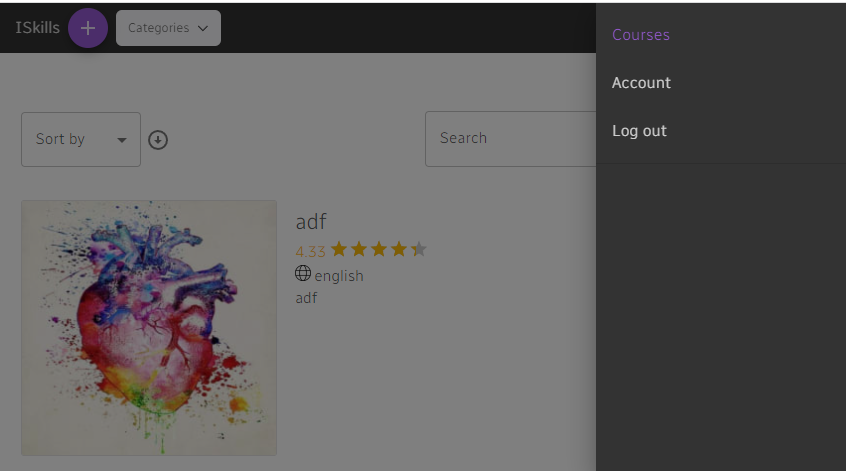


Рис 12. Початкове вікно авторизованого юзера з боковим меню активованим меню

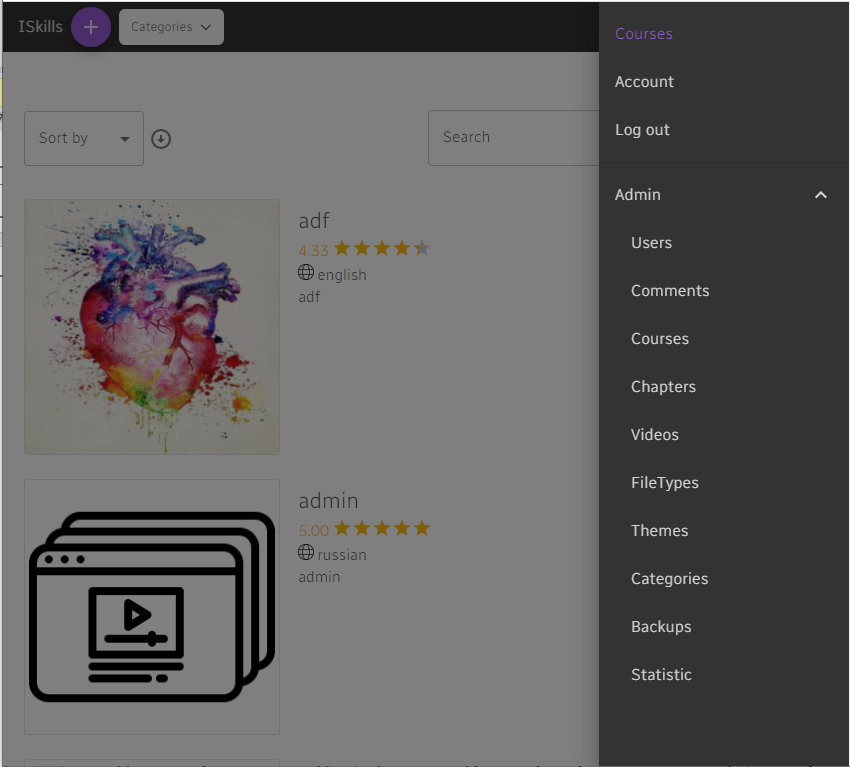
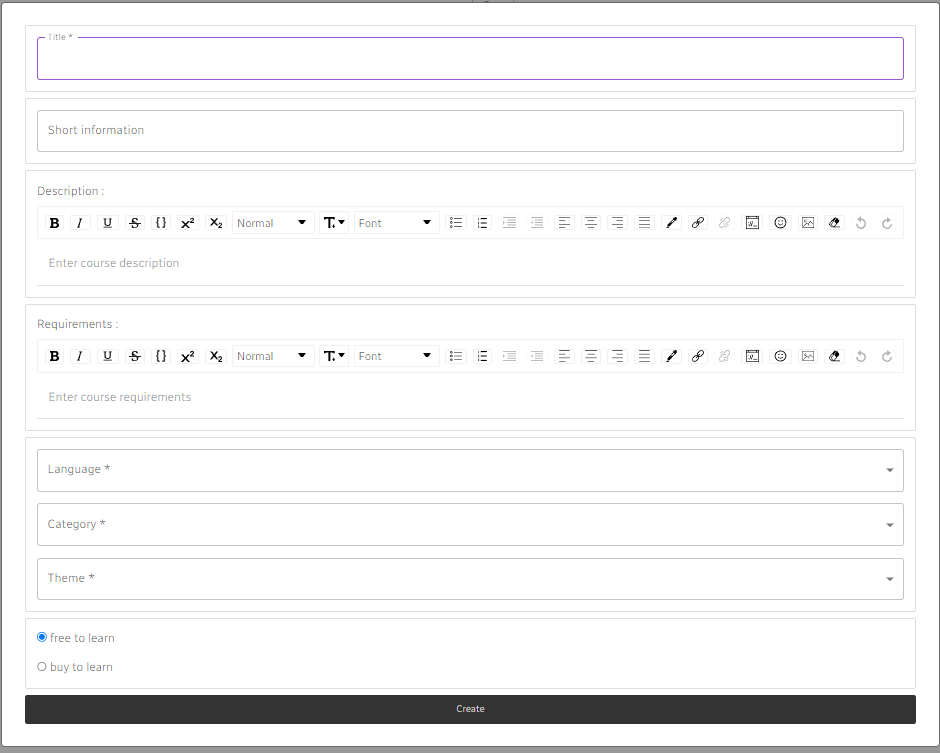


Рис 13. Початкове вікно для адміна з боковим меню активованим меню

З даних рисунків видно, що неавторизований користувач має тільки можливість перегляду курсів та логіну або реєстрації, а уже авторизований може повноцінно взаємодіяти з додатком, тоді як адміністратор має ще додаткові функції, а саме повний контроль над всіми даними в додатку.



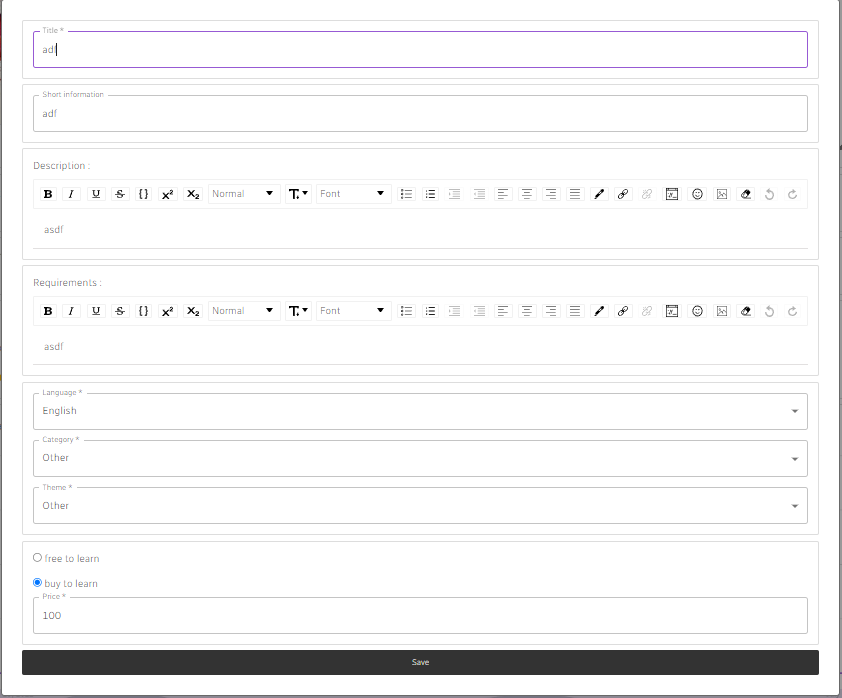


Рис 14-15. Форми створення та оновлення курсу

Дана форма обов’язково приймає назву, мову, категорію, тему та ціну при вказанні, що курс платний. Перед відправкою форми відбуваються перевірки на коректність даних. Також після створення курсу йому додається картинка по замовчування, яку користувач може змінити згодом.

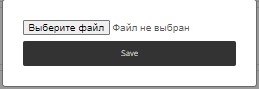


Рис 16. Форми зміни картинки

Дана форма використовується як і на сторінці курсу так і на сторінці користувача для зміни відображуваної картинки відповідного елемента.

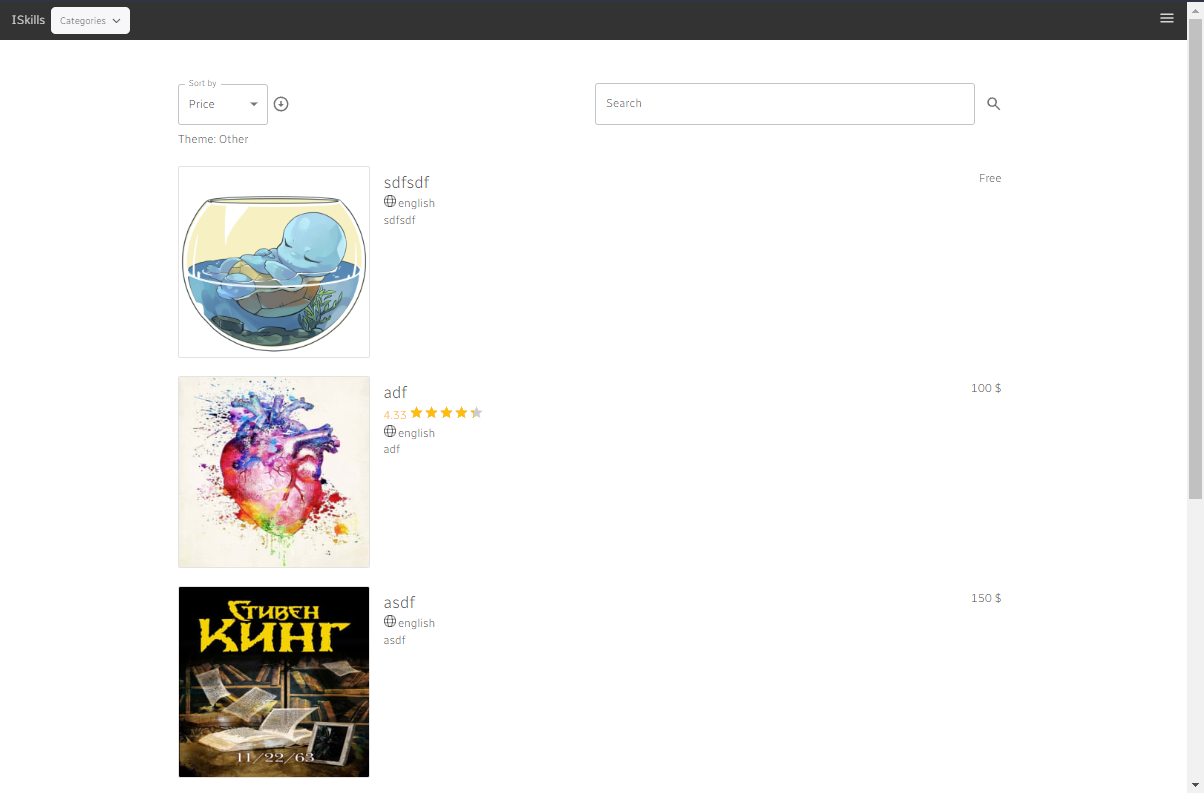
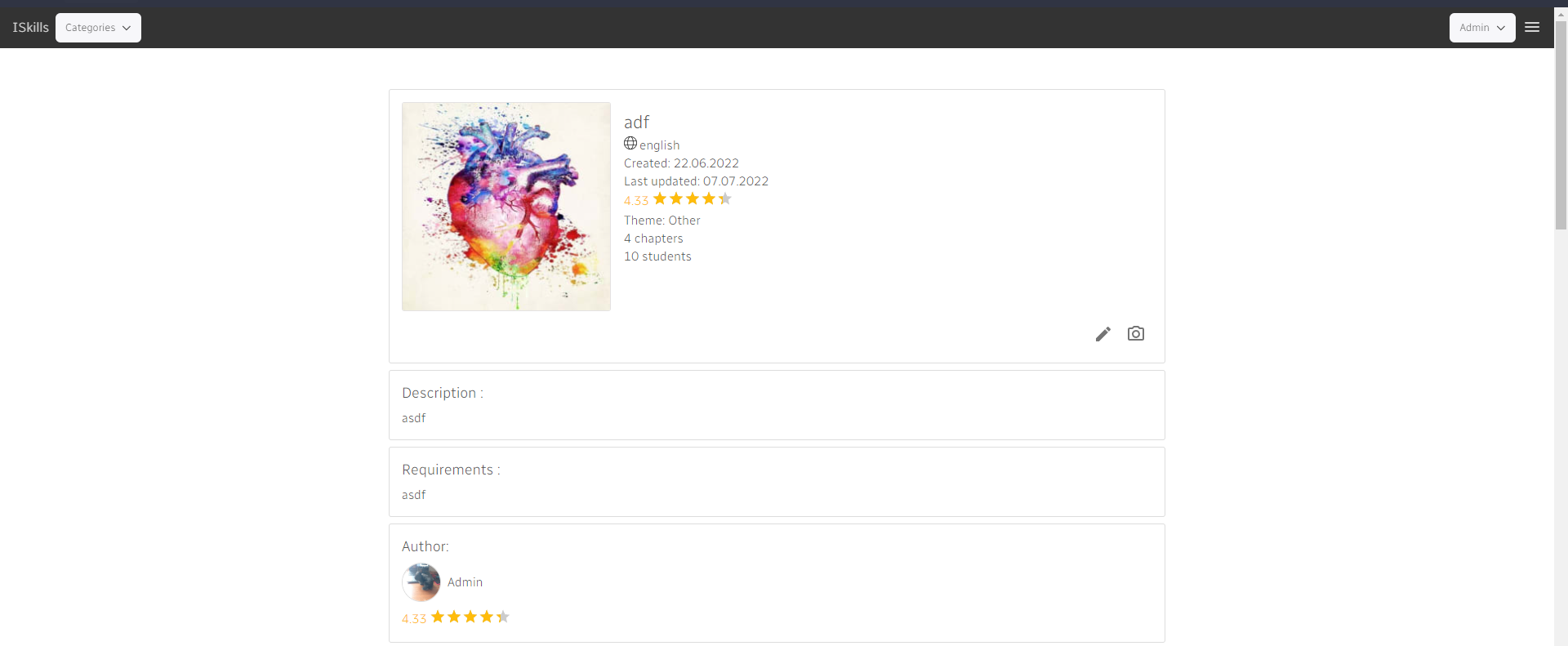
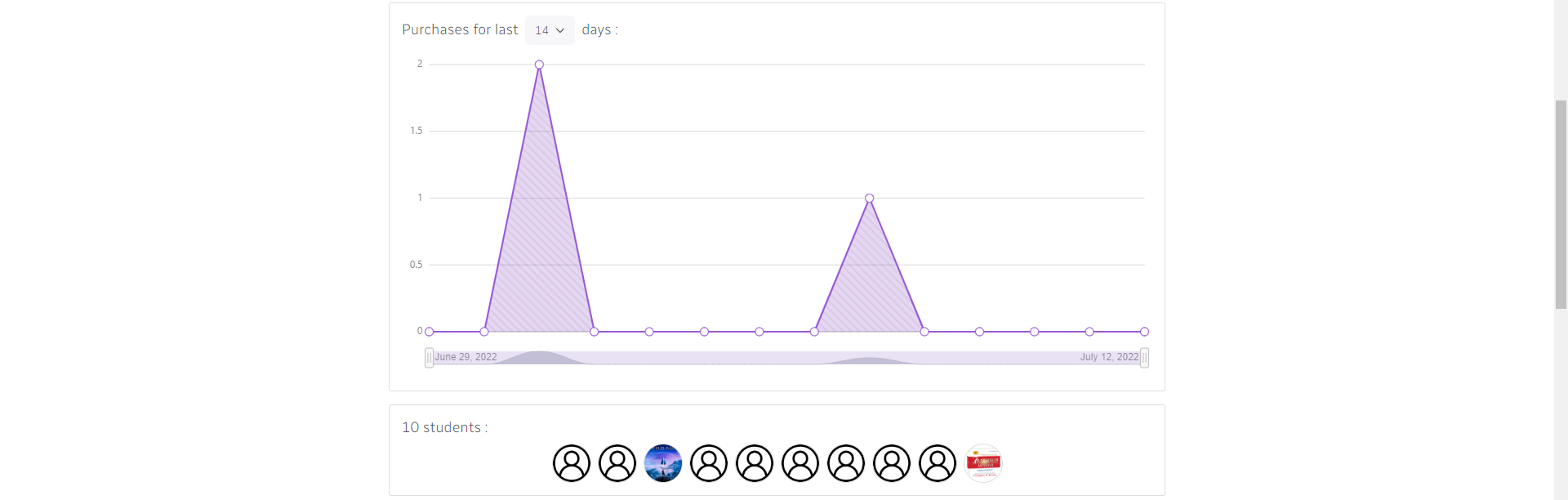


Рис 17. Сторінка для перегляду наявних курсів

На даній сторінці корситувач моде перглядати курси, шукати по їх по назві та по темі одночасно, також має можливість сортувати по деяким параметрам в різному порядку. Також на даній сторінці адмін або автор курсу з може видалити певний курс або курси.





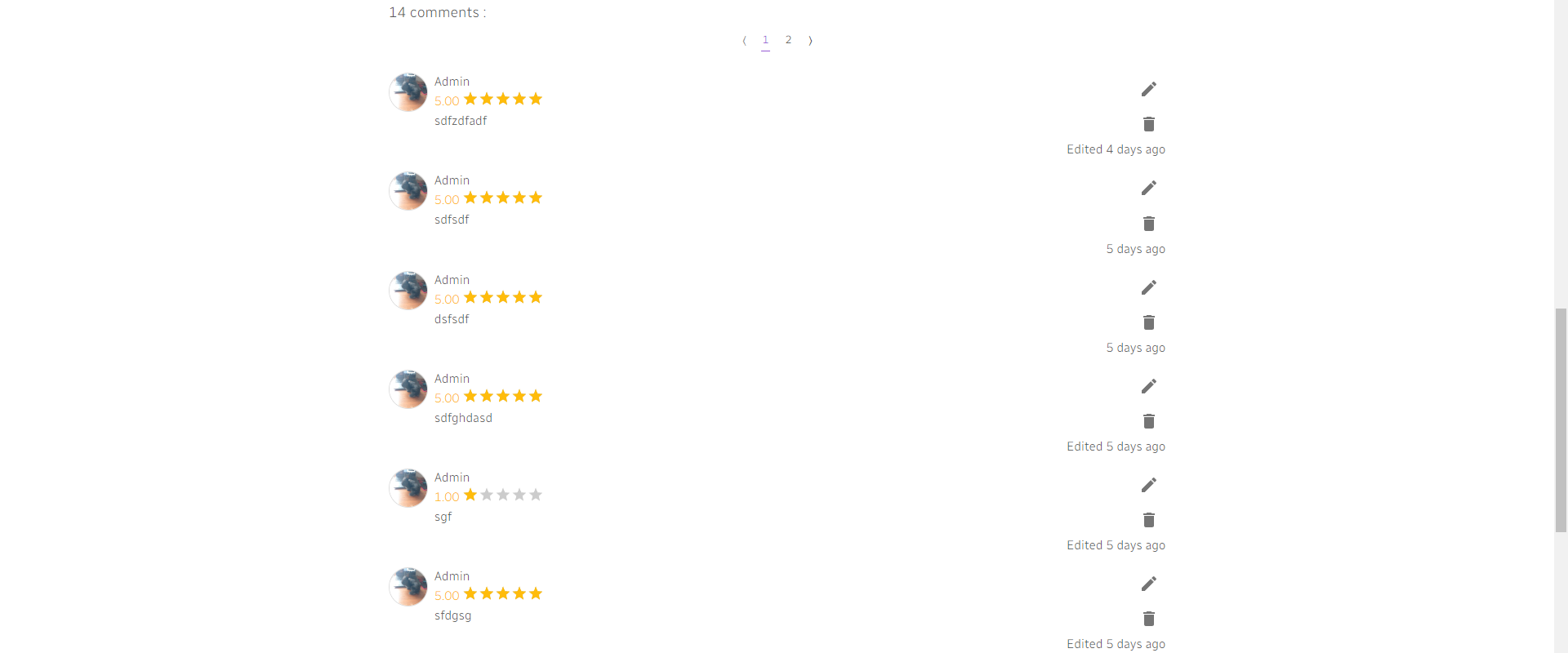
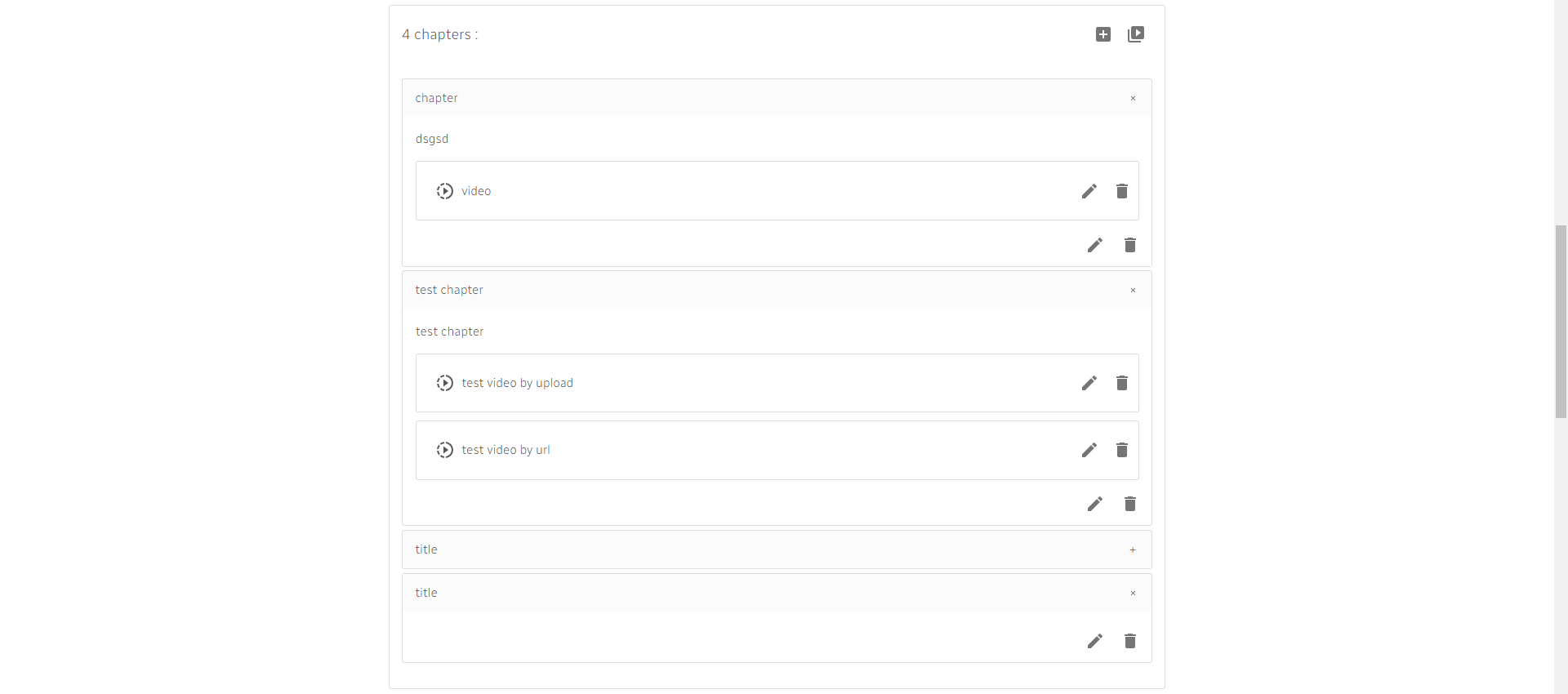
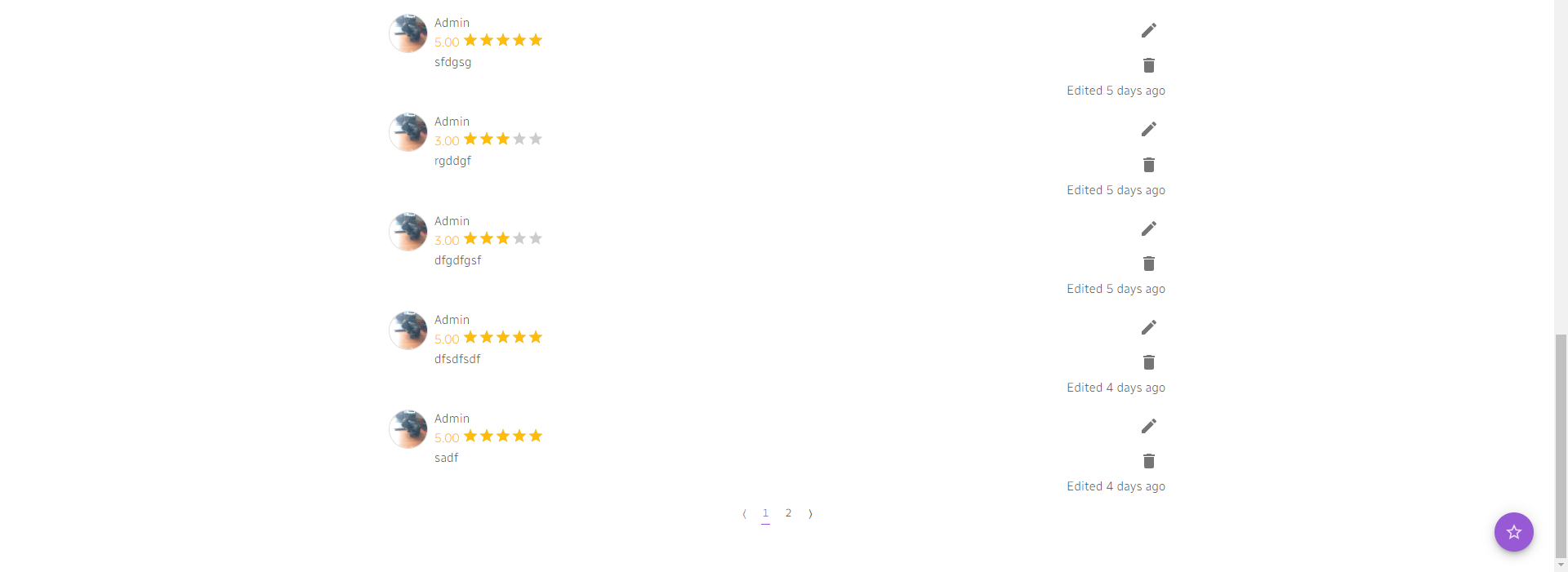
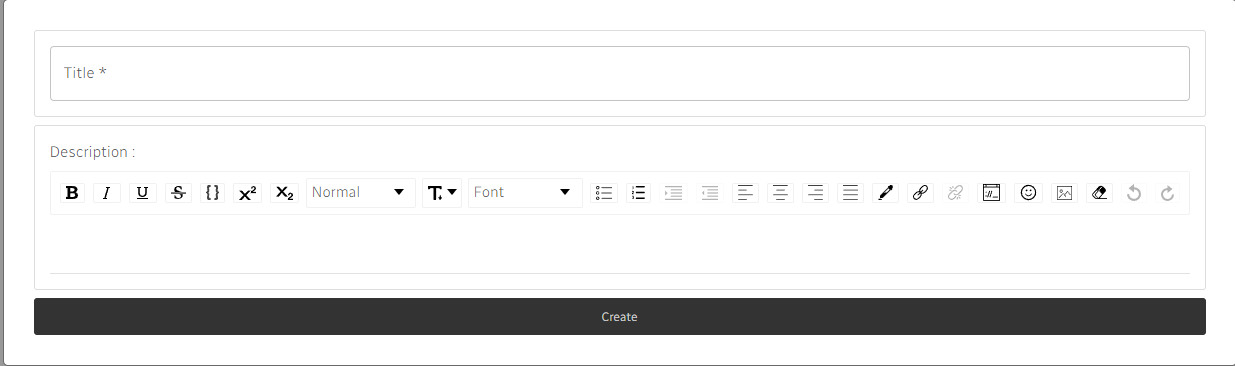
 

Рис (комплексний) 18. Приклад сторінки курсу

На даній сторінці користувач може переглядати матеріали курсу та писти коментарі, а автор курсу може доповнювати новими розділами та відеоматеріалами. адміністратор має такі ж права як і автор тільки ще може видаляти коентарі інших користувачів.



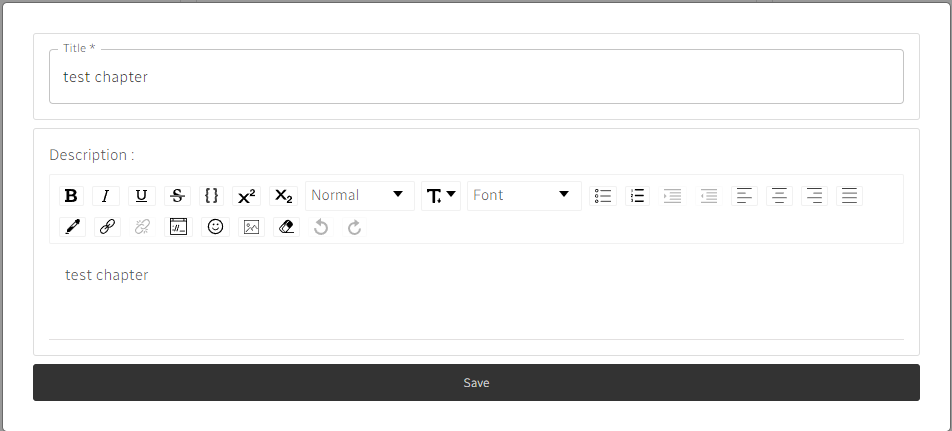


Рис 19, 20. Форми додавання та оновлення розділу

Логіка даних форм схожа до попередньо-розглянутих форм, але в даному випадку обовязковим полем є тільки назва, а опи опціональний.

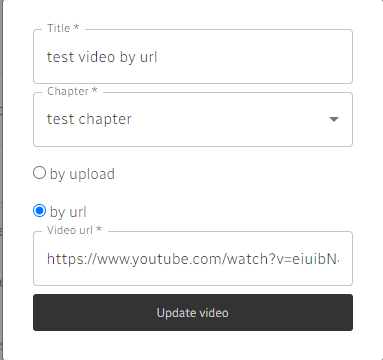
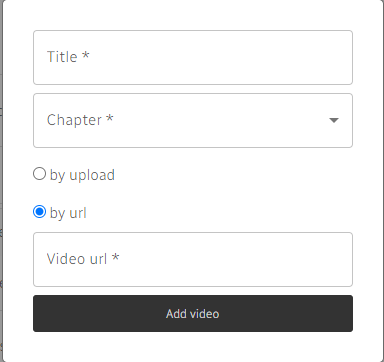


Рис 21, 22. Форми додавання та оновлення відео

Логіка всіх форм схожа, тому вирішено білне не акцентувати на цьому увагу. На даній формі додавання відео автор або адмін повинні ввести коректні дані про назву відео, обрати розділ та або передати url до відео, або загрузити його як файл. Після нажаття на відповідну кнопку виконується додання або оновлення відео. Як бачимо із рисунків, після додання відео, цого можна перемістити в інший розділ.

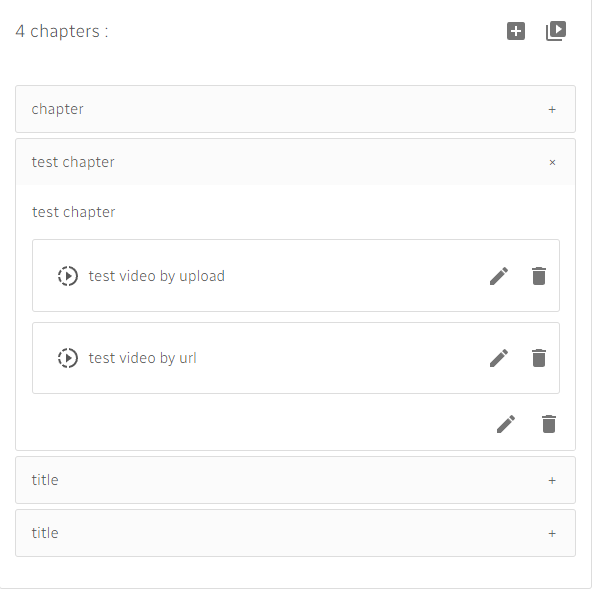


Рис 23. Приклад додавання декількох розділів та відео

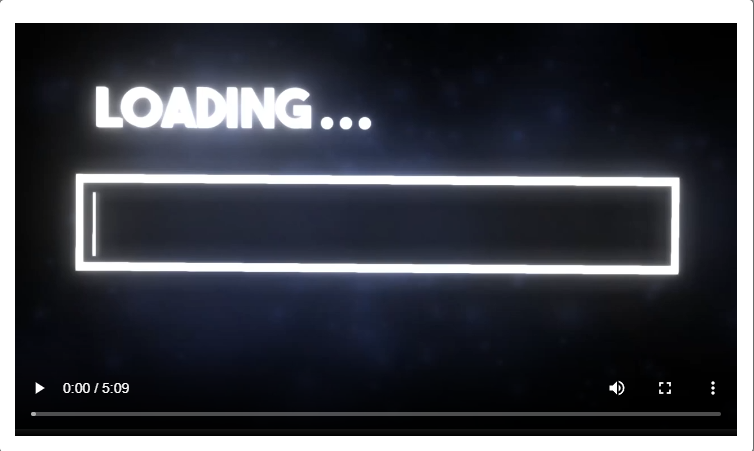
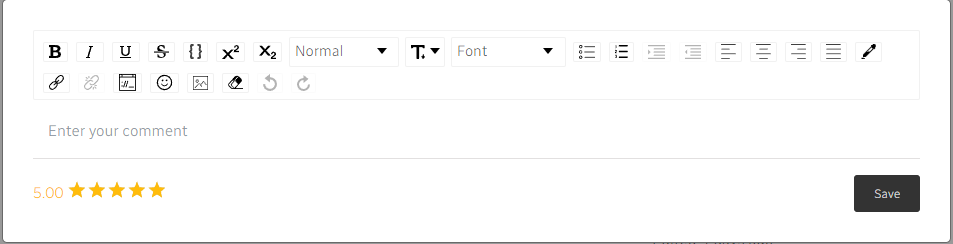


Рис 24. Приклад додавання декількох розділів та відео

На рисунку 23 наведено приклад реальної частини курсу із розділами та відео, які можна переглядати, що і наведено на рисунку 24.



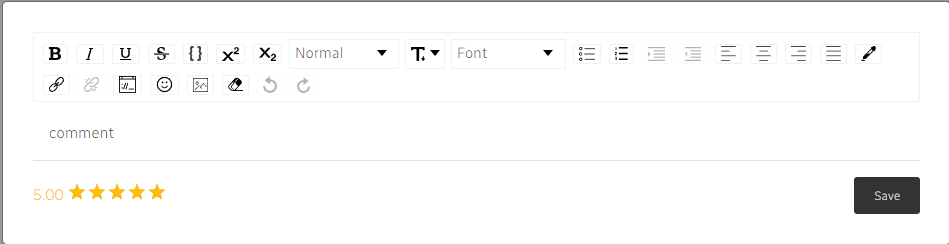
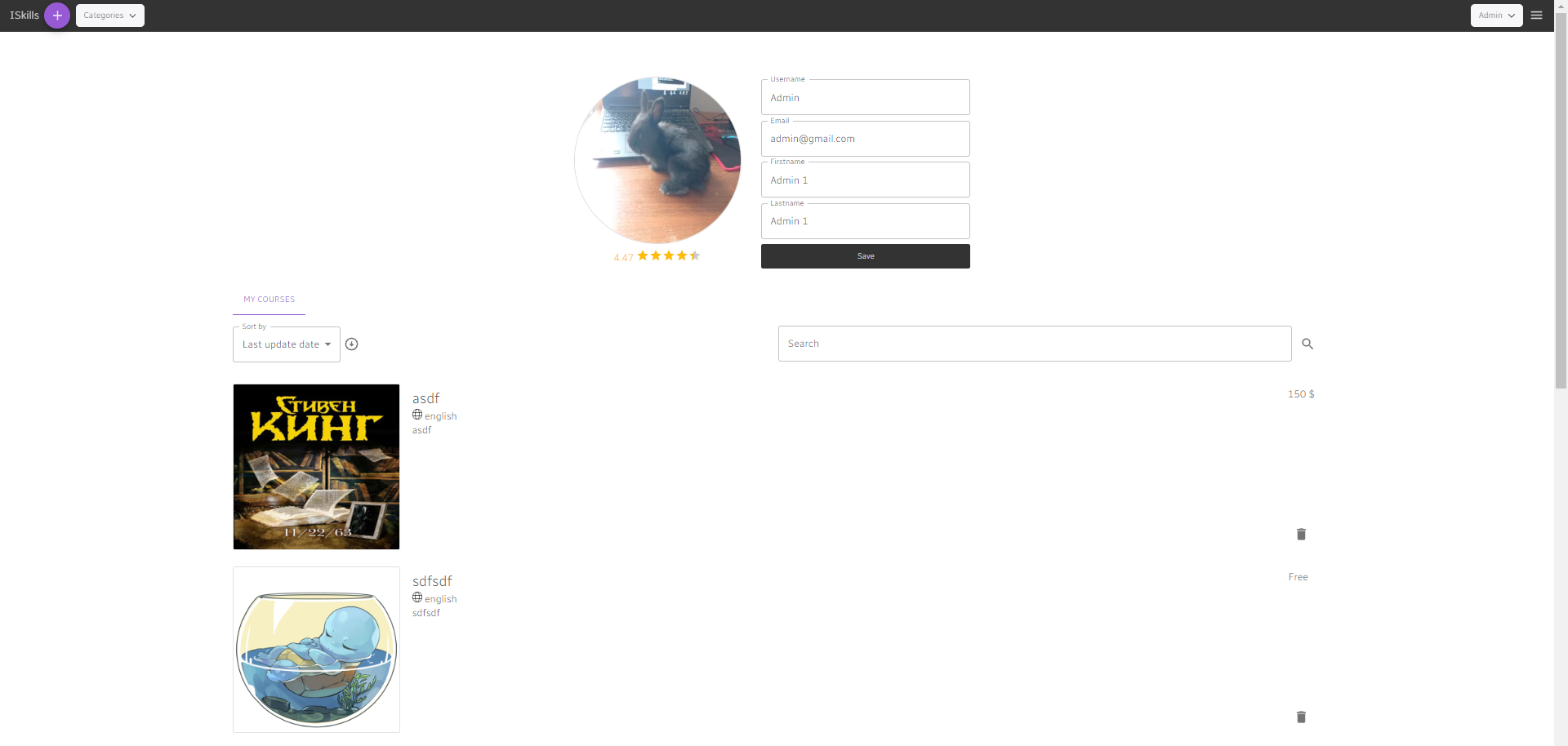


Рис 25, 26. Форми створення та редагування коментарю

В даних формах користувач має заповнити текст коментаря і обрати оцінку курсу, після чого після проходження валідації коментар додається до курсу.



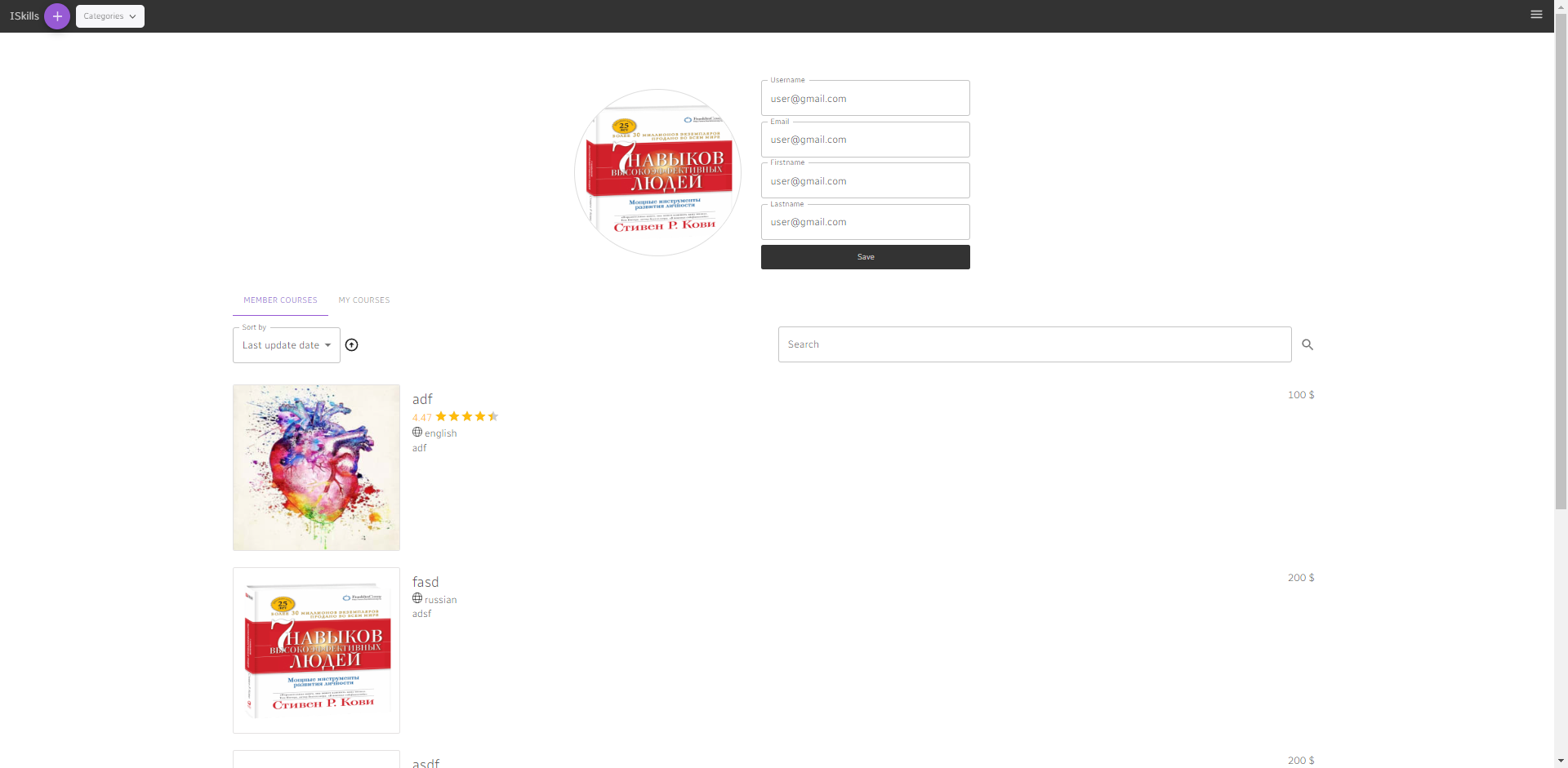
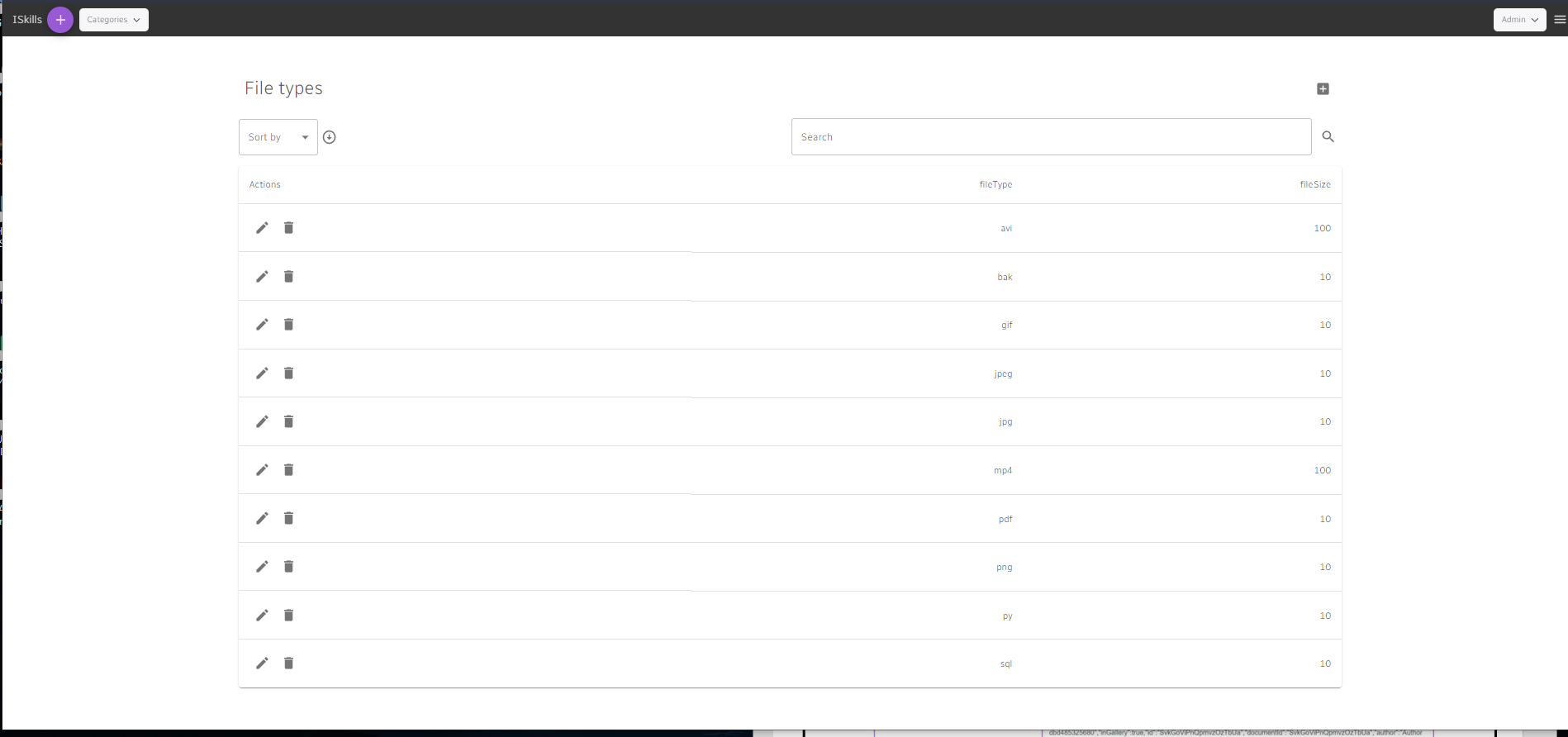


Рис 27, 28. Сторінка профілю адміна та сторінка профілю звичайного користувача

Будь-який корситувач на даній сторінці може змінти інформацію про себе, а також подивитися ті курси які він створив, також можна переглянути ті курси, які були придбані, але адміністратор не має такої можливості, бо він і там має доступ до всіх курсів, тому йому такий функціонал не потрібно, а також адмнім не може рахуватися як струдент курсу. Якщо користувач створить хоча б один курс і хтось його оцінить, то він отримає показник рейтингу в профілі.



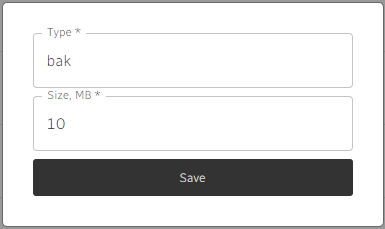
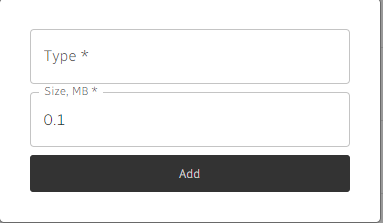


Рис 29, 30, 31. Сторінка дозволених розширень та відповідні форми створення та редагування

На даній сторінці адміністратор може переглянути, які є дозволені для загрузки типи файлів є та їх обмеження на розмір. Також на даній сторінці є відповідні форми, де адміністратор повинен вказати назву розширення та розмір. В даній курсовій роботі використовується хмарний сервіс cloudinary, який дозволяє зберігати медіа-файли, але з пеними обмеженнями, а саме для файлів та картинок це 10 МБ, а для відео – 100 МБ.

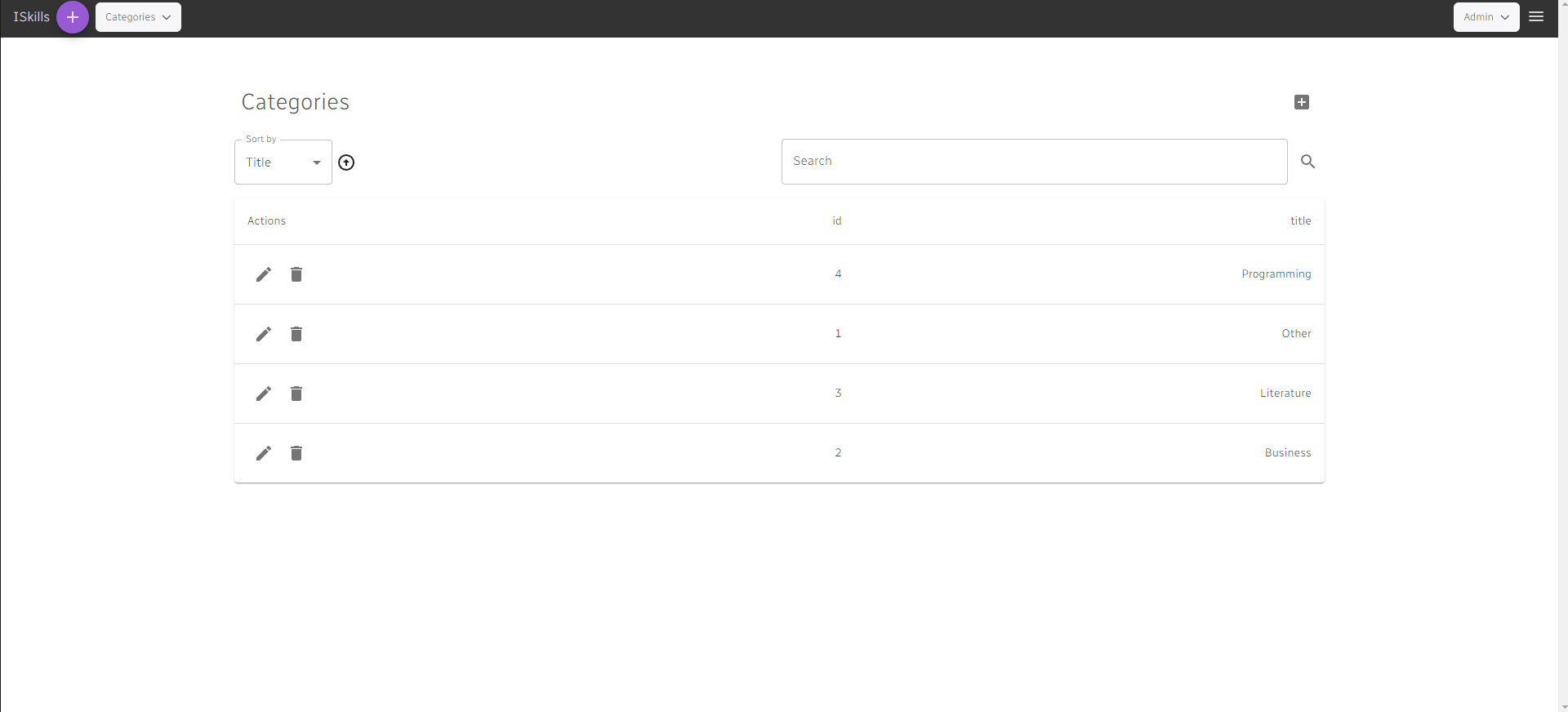


Рис 32. Сторінка категорій

Функціонал даної сторінки аналогічний до попередньої сторінки, за вийнятком контексту про категорії, а у формах потрібно вказувати тільки назву, яка має бути унікальною.

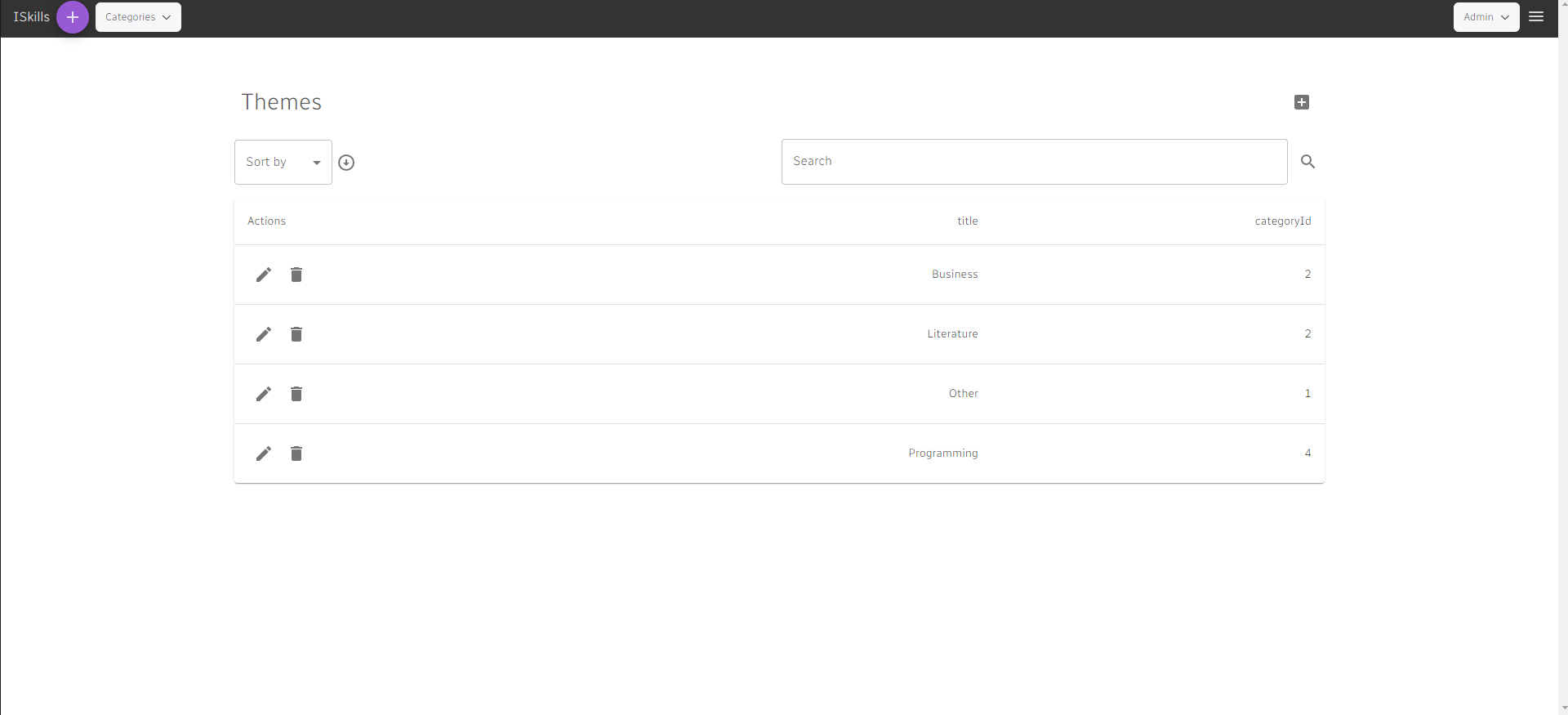


Рис 33. Сторінка тем

Функціонал даної сторінки аналогічний до попередньої сторінки, за вийнятком контексту про теми, а у формах потрібно вказувати назву та обрати категорію, назва також має бути унікальною.

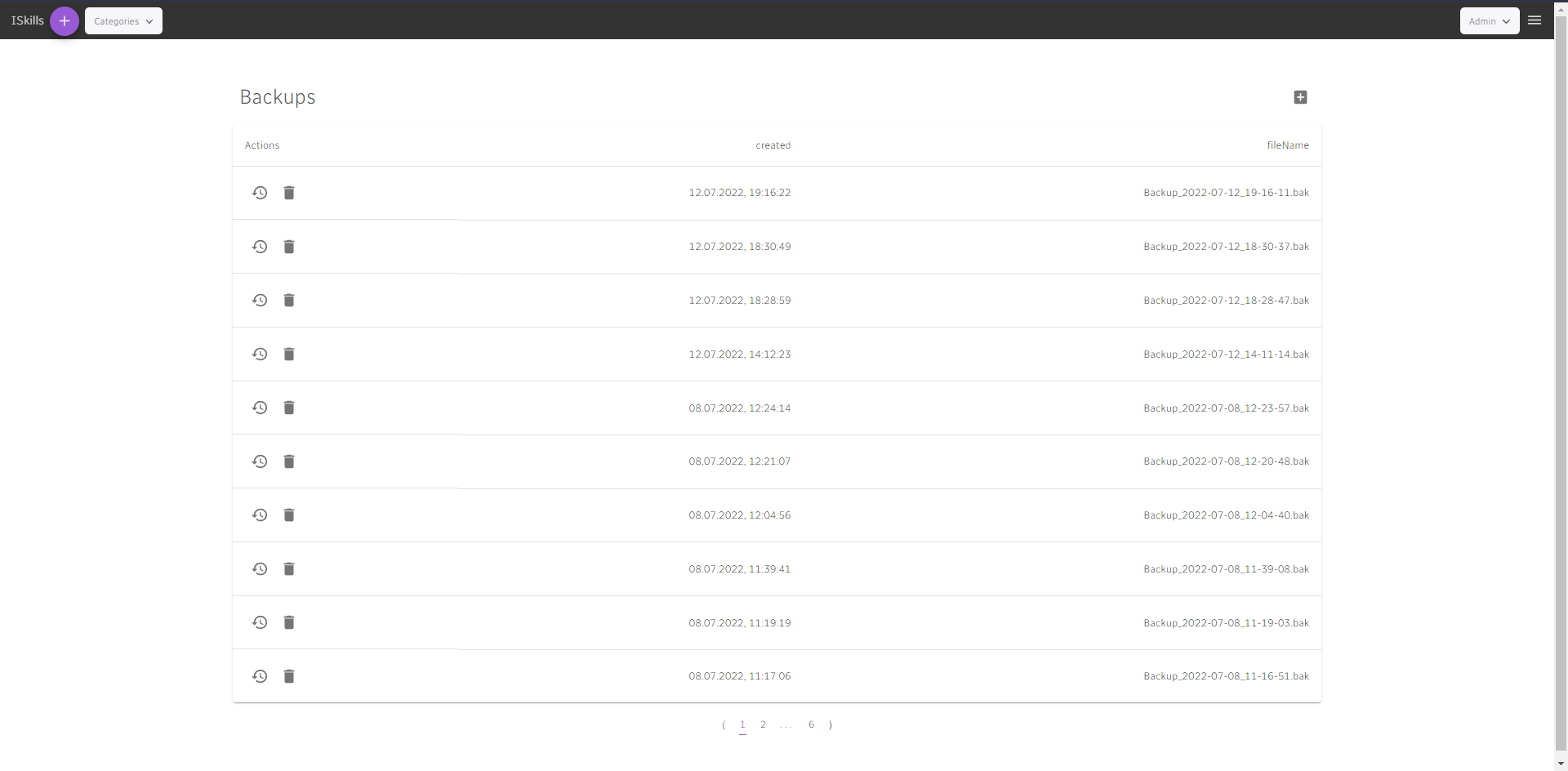
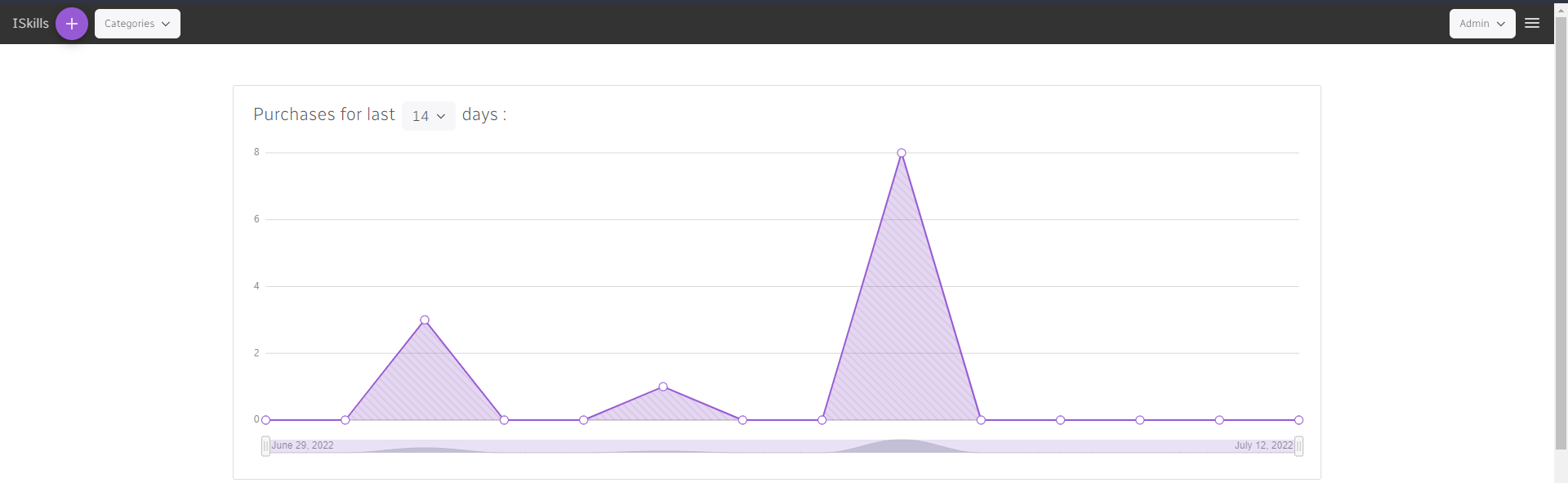
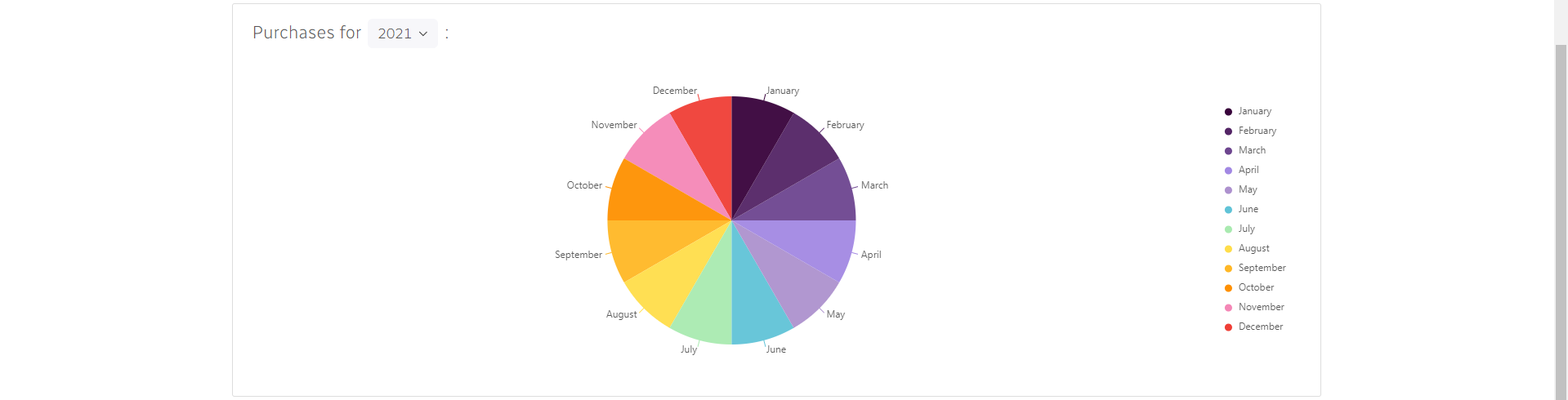


Рис 34. Сторінка бекапів

На даній сторінці адмін може переглянути доступні бекапи, зробити рестор та створити новий бекап.





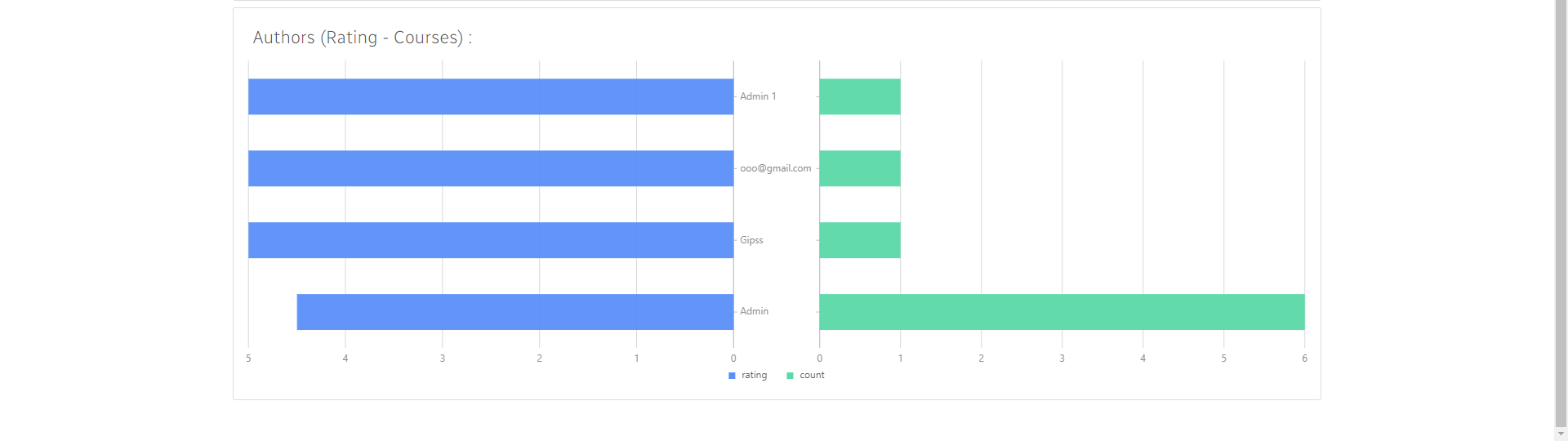


Рис (комплексний) 35. Сторінка статистики

На даній сторінці адміністратор може переглянути покупки за певний період в вигляді графіка, покупки помісячно за певний рік в вигляді кругової діаграми та топових авторів у відношенні до їх курсів у вигдяду дуальної гістограми. Також статистика присутня на сторінці кожного курсу, де будь-який користувач може пеглянути історію покупок за певний період. А також елементом статистики думаю можна вважати рейтинги як курсу, так і користувача.

## **3.2. Реалізації операцій обробки даних в БД**

Оскільки будь-яка програма розпочинається з аутентифікації та контролю доступу, то потрібно розглянути, як саме користувач реєструється та входить в додаток під своїм логіном.

public async Task<Guid> CreateAsync(RegisterUserModel model, CancellationToken cancellationToken)

{

if (await \_userContext.Users.AnyAsync(u => u.Email == model.Email, cancellationToken))

throw new AlreadyExistsException(nameof(User), nameof(model.Email), model.Email);

if (await \_userContext.Users.AnyAsync(u => u.UserName == model.UserName, cancellationToken))

throw new AlreadyExistsException(nameof(User), nameof(model.UserName), model.UserName);

var user = \_mapper.Map<User>(model);

user.Id = Guid.NewGuid();

var role = await \_roleContext.Roles

.FirstOrDefaultAsync(r =>

r.Name == "User", cancellationToken);

user.Roles = new List<Role> { role };

user.Rating = default;

user.Salt = Hasher.GenerateSalt(saltSize);

user.Password = Hasher.GetSaltedHash(user.Password, user.Salt);

await \_userContext.Users.AddAsync(user, cancellationToken);

await \_userContext.SaveChangesAsync(cancellationToken);

return user.Id;

}

Дана функція приймає об’єкт, що містить пароль, пошту, псевдонім, ім’я, прізвище. Далі шукаємо чи користувач з даним емайло адо юзернеймом уже існує, якщо так то викидаємо помилку, в іншому випадку продовжуємо реестрацію, хешуючи пароль і створюючи хешовану сіль. Далі зберігаємо користувача в базі.

Тепер розглянемо метод для оновлення інформації про користувача

public async Task UpdateAsync(Guid id, RegisterUserModel model, CancellationToken cancellationToken)

{

var user = await \_userContext.Users.GetAsync(\_mapper,

u => u.Id == id, new() { }, cancellationToken);

if (!string.IsNullOrEmpty(model.Email) && model.Email != user.Email)

{

if (await \_userContext.Users.AnyAsync(u => u.Email == model.Email, cancellationToken))

throw new AlreadyExistsException(nameof(User), nameof(model.Email), model.Email);

user.Email = model.Email;

}

if (!string.IsNullOrEmpty(model.UserName) && model.UserName != user.UserName)

{

if (await \_userContext.Users.AnyAsync(u => u.UserName == model.UserName, cancellationToken))

throw new AlreadyExistsException(nameof(User), "Username", model.UserName);

user.UserName = model.UserName;

}

user.FirstName = model.FirstName;

user.LastName = model.LastName;

user.UserName = model.UserName;

if (!string.IsNullOrEmpty(model.Password))

user.Password = Hasher.GetSaltedHash(model.Password, user.Salt);

\_userContext.Users.Update(user);

await \_userContext.SaveChangesAsync(cancellationToken);

}

В даному методі отримуємо тей самий обект, що і при реєстрації і потім шукаємо по id користувача, дані якого згодом оновлюємо.

Далі розглянемо авторизацію, що забезпечую логінізацію користувача.

public async Task<AuthenticateResponse> Authenticate(AuthenticateRequest model, string ip, CancellationToken cancellationToken)

{

var user = await \_userContext.Users.GetAsync(\_mapper,

u => u.Email == model.Email,

new() { u => u.Roles, u => u.RefreshTokens }, cancellationToken);

if (!model.Password.AreEqual(user.Salt, user.Password))

throw new ConflictException(message: "Email or password is incorrect");

RefreshToken refreshToken;

if (!user.RefreshTokens.Any(t => t.IsActive == true))

{

var refreshTokenDto = GenerateRefreshToken(ip);

refreshToken = \_mapper.Map<RefreshToken>(refreshTokenDto);

user.RefreshTokens.Add(refreshToken);

await \_userContext.SaveChangesAsync(cancellationToken);

}

else

refreshToken = user.RefreshTokens

.FirstOrDefault(t => t.IsActive);

var jwtToken = GenerateJwtToken(user);

return new AuthenticateResponse(jwtToken, refreshToken.Token);

}

В даному методі ми отримуємо об’єкт, який містить пароль та пошту. Далі перевіряємо на співпадіння в базі, створюємо або отримуємо токен оновлення, в залежності від умов та генеруємо токен доступу, в результаті повертаємо токен доступу та токен оновлення.

Розглянемо далі метод створення курсу.

public async Task<Guid> CreateAsync(CreateCourseDto model, CancellationToken cancellationToken)

{

var user = await \_userContext.Users.GetAsync(\_mapper,

u => u.Id == model.CreatorId, new () { }, cancellationToken);

var theme = await \_themeContext.Themes.GetAsync(\_mapper,

t => t.Id == model.ThemeId, new() { }, cancellationToken);

var course = \_mapper.Map<Course>(model);

course.Id = Guid.NewGuid();

course.DateCreated = DateTime.UtcNow;

course.DateUpdated = DateTime.UtcNow;

course.Students = new List<User>();

course.Comments = new List<Comment>();

course.Chapters = new List<Chapter>();

course.Rating = default;

await \_courseDbContext.Courses.AddAsync(course, cancellationToken);

await \_courseDbContext.SaveChangesAsync(cancellationToken);

return course.Id;

}

Даний метод приймає об’єкт зі всіма необхідними даними про курс, а саме назва, опис, тема і т.д.. Далі робимо перевірки та зберігаємо його.

Далі розглянемо метод оновлення курсу, який приймає тей самий об’єкт, що і попередній метод , але з Id.

public async Task UpdateAsync(Guid id, CreateCourseDto model, CancellationToken cancellationToken)

{

var course = await \_courseDbContext.Courses.GetAsync(\_mapper,

v => v.Id == id, new () { x => x.Theme }, cancellationToken);

var theme = await \_themeContext.Themes.GetAsync(\_mapper,

t => t.Id == model.ThemeId, new () { }, cancellationToken);

course.Title = model.Title;

course.ShortInfo = model.ShortInfo;

course.Requirements = model.Requirements;

course.Description = model.Description;

course.Language = model.Language;

course.Price = model.Price;

course.Theme = theme;

course.DateUpdated = DateTime.UtcNow;

\_courseDbContext.Courses.Update(course);

await \_courseDbContext.SaveChangesAsync(cancellationToken);

}

Далі проводеимо перевірки і зберігаємо зміни.

Далі розглянемо метод отримання курсів.

public async Task<PaginationList<CourseDto>> GetList(int skip, int take, string query,

string sortOption, bool reverse, CancellationToken cancellationToken, params object[] dynamics)

{

int? themeId = (int?) dynamics[0];

Guid? creatorId = (Guid?) dynamics[1];

return await \_courseDbContext.Courses.GetListAsync<Course, CourseDto>(

\_mapper,

skip,

take,

c => c.Title.Contains(query.ToLower().Trim())

&& (themeId == null || c.ThemeId == themeId)

&& (creatorId == null || c.CreatorId == creatorId),

sortOption,

reverse,

new() { },

cancellationToken);

}

Даний метод приймає такі параметри, як: query, skip, take, параметер сортування та порядок сортування. Тут формуємо відповідні лямбда вирази, щоб отримати базані курси. За кількість курсі які будуть пропущені і отримані, відповідають параметри skip і take відповідно. За сотрування відповідає sortOption і reverse.

Далі розглянемо метод для видалення курсу.

public async Task DeleteByIdAsync(Guid id, CancellationToken cancellationToken)

{

var course = await \_courseDbContext.Courses.GetAsync(\_mapper,

c => c.Id == id, new() { }, cancellationToken);

var creator = await \_userContext.Users.GetAsync(

\_mapper, x => x.Id == course.CreatorId, new() { }, cancellationToken);

\_courseDbContext.Courses.Remove(course);

await \_courseDbContext.SaveChangesAsync(cancellationToken);

creator.Rating = await \_courseDbContext.Courses

.GetAvarage(x => x.CreatorId == creator.Id && x.Rating != default, x => x.Rating);

await \_userContext.SaveChangesAsync(cancellationToken);

}

Даний метод приймає id курсу, потім шукає відповідний курс в базі та видаляє його, перераховуючи рейтинг його автора, що ми можемо бачити вище в коді.

Дані методи є базовими для платформи курсів і інші необхідні методи будуть написані зі схожою поведенкою тільки в іншому контексті.

## **3.3. Організація звітності системи**

У даному підрозділі буде розглянуто лише ті звітності, що були розроблені спеціально для додатку, а саме статистика по покупкам за певний період, по покупкам за певний рік по місяцям, по покупка конкретного курсу за певний період і по авторам у відношенні рейтингу та кількості курсів.

Першим методом є метод GetGroupedPurchases. Лістинг наведено нижче :

public async Task<List<DayGroupedItem>> GetGroupedPurchases(string sortOption,

bool reverse, CancellationToken cancellationToken, params object[] dynamics)

{

var courseId = (Guid?)dynamics[0];

var startDate = (DateTime?)dynamics[1];

var endDate = (DateTime?)dynamics[2];

return await \_purchaseDbContext.Purchases.CustomGroupByAsync(

p => (courseId == null || p.CourseId == courseId)

&& (startDate == null || p.Date >= startDate)

&& (endDate == null || p.Date <= endDate),

sortOption,

reverse,

p => p.Date.ToString(),

group => new DayGroupedItem { Day = group.Key.ToString(), Amount = group.Count()},

cancellationToken);

}

Даний метод є асинхронним, він приймає параметер сортування, яким має бути поле таблиці в базі даних, параметер який відповідає за порядок сортування, токен відміни та опціональні параметри, а саме id курсу, щоб групувати тільки конкретного курсу, початкову дату та кінцеву дату, щоб вказати період часу. Даний метод потрібен, щоб отрати статистику за певний період по всім курса чи по конкретному. Сам метод викликає метод який є розширенням для таблиці бази даних, лістинг якого наведено нижче :

public static async Task<List<TResult>> CustomGroupByAsync<T, TResult>(this IQueryable<T> context, Expression<Func<T, bool>> expression, string sortOption, bool reverse,Expression<Func<T, string>> groupByExpression, Expression<Func<IGrouping<string, T>, TResult>> selectExpression,CancellationToken cancellationToken = default)

where T : class

where TResult : class

=> await context

.Where(expression)

.OrderBy(sortOption, reverse)

.GroupBy(groupByExpression)

.Select(selectExpression)

.AsNoTracking()

.ToListAsync(cancellationToken);

Даний метол приймає вираз пошуку, поле сортування, порядок сортування, вираз групування, вираз вибірки після групування та токен відмініми, який є необовязковим. В ході виконання даного методу, спочатку в відповідній таблиці бази даних відбираються записи, які відповідають заданому виразу, потім відбувається сортування по переданому полу та порядку, далі ці записи групуються по виразу групування, а вже потім вони конвертуються в потрібний обєкт. Даний метон написаний так, що його можна було використовувати для будь-якої таблиці БД.

Далі розглянемо метод для отримання статистики за певний рік згрупований по місяцям, а саме GetGroupedYearPurchases, лістинг якого наведено нижче :

public async Task<List<MonthGroupedItem>> GetGroupedYearPurchases(string sortOption,

bool reverse, int year, CancellationToken cancellationToken, params object[] dynamics)

{

var courseId = (Guid?)dynamics[0];

return await \_purchaseDbContext.Purchases.CustomGroupByAsync(

p => (courseId == null || p.CourseId == courseId) && p.Date.Year == year,

sortOption,

reverse,

p => p.Date.Month.ToString(),

group => new MonthGroupedItem { Month = group.Key.ToString(), Amount = group.Count() },

cancellationToken);

}

Як можемо бачити даний метод схожий до попереднього, бо викликає тей самий метод розширення але з іншими параметрами, що є великим плюсом, бо так уникається дублювання коду. Даний метод приймає замість дат номер року і тоді уже відбирає записи, які були зроблені в даному році і групує по місяцям.

Далі розглянемо метод для отримання статистики по авторам, а саме GetTopUsers, лістинг якого наведено нижче :

public async Task<List<UserStatisticModel>> GetTopUsers(CancellationToken cancellationToken)

=> await \_userContext.Users

.Where(u => \_courseDbContext.Courses.Any(c => u.Id == c.CreatorId))

.OrderBy("rating", true)

.Select(x => new UserStatisticModel

{

UserName = x.UserName,

Count = \_courseDbContext.Courses

.Where(c => x.Id == c.CreatorId).Count(),

ImageUrl = x.ImageUrl,

Email = x.Email,

Rating = x.Rating

})

.AsNoTracking()

.ToListAsync(cancellationToken);

Даний метод відбирає користувачів, які створювали курси, та сортує їх за рейтингом та відобрає кількість створених курсів цими корисувачами, що дає змогу зробити дуальну гістограму.

Всі статистичні методи також доповнюються на фронтенді, так як якщо в базі даних немає записів, наприкла за конкретний день, то і даних про цей день не буде, для цього потрібно доповнювати графіки на фронтенді.

## **Висновки до третього розділу**

У ході виконання даного розділу відбулось написання повного коду для взаємодії користувача з БД. Було спроектовано інтерфейс платформи курсів. Були розроблені форми для створення курсів, користувачів, категорій, тем, коментарів, дозволених типів файлів з обмеженнями розміру, розділів, відео та їх редагувань. Також були створені сторінки з можливістю перегляду, виконання запитів на фільтрації даних, виконання запитів на підгрузку даних. Крім цього, було створено можливість зберігати поточний стан бази, та відновлювати його з резервної копії його.

Також при виконання даного розділу було написано та описано логіку та код виконання звітності про систему за кількома типами, а саме – отримання покупок за певний період або за рік згрупованим по місяцям по всім курсам платформи або по конкретному курсу, а також статистика по авторам курсів, а саме відношення рейтингу до кількості курсів.

# **РОЗДІЛ 4. АДМІНІСТРУВАННЯ БАЗ ДАНИХ**

## **4.1. Розробка заходів захисту інформації в БД**

Отже, розглянемо категорії користувачів інформаційної системи платформи курсів: головний адміністратор, звичайні користувачі та гості. Найбільші права доступу до даних має головний адміністратор, що необхідно для організації роботи. Він можє переглядати будь-які дані даної платформи. Також головний адміністратор повинен мати змогу робити бекапи та можливість відновлювати базу даних, для яких потрібні доступи до усіх таблиць. Тому доцільно створити окрему роль "Адміністратор" для забезпечення надання доступу до цих даних. Більш детально перелік об’єктів БД, доступ до яких надано ролі " Адміністратор " наведено в табл. 10. На перетині рядків і стовпців зазначено дії, які може виконувати користувач даного типу ролі: 1 – перегляд даних, 2 – редагування, 3 – видалення, 4 – повний доступ.

Таблиця 10

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Courses | Chapters | Videos | Users | Comments | Categories | Themes | AllowedFileTypes | Purchases |
| Admin | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 1 |

Окрему групу користувачів системи складають авторизовані користувачі та просто гості платформи. Авторизовані користувачі мають змогу створювати власні курси і повністю з ними взаємодіяти, а також купувати курси інших користувачів. Гості в свою чергу можуть тільки переглядіти доступні курси, без пожливості їх купівлі. Детальніше ролевий доступ зображений в таблиці 11.

Таблиця 11

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Courses | Chapters | Videos | Users | Comments | Categories | Themes | AllowedFileTypes | Purchases |
| Authorized users (тільки власні курси) | 4 | 4 | 4 | 1 | 1, 2,3  (тільки власні коментарі) | 1 | 1 | 1 | 1 |
| Authorized users  (тільки куплені курси) | 1 | 1 | 1 | 1 | 1, 2,3  (тільки власні коментарі) | 1 | 1 | 1 | 1 |
| Guestes | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

Коли авторизований користувач не придбав певний курс або не створий його, він має тіж самі доступи що і гість, тільки з можливістю придбання курсу.

Для контролю доступу використовуєтьються методи репозиторію AccessRepository, лістинг яких наведено нижче:

public async Task<bool> HasAccessToUser(Guid userId, Guid id, CancellationToken cancellationToken)

=> await HasAccessToEntity(userId, \_userContext.Users,

x => x.Id == id, cancellationToken);

public async Task<bool> HasAccessToCourse(Guid userId, Guid id, CancellationToken cancellationToken)

=> await HasAccessToEntity(userId, \_courseContext.Courses,

x => x.CreatorId == userId && x.Id == id, cancellationToken);

public async Task<bool> HasAccessToComment(Guid userId, Guid id, CancellationToken cancellationToken)

=> await HasAccessToEntity(userId, \_commentContext.Comments,

x => x.CreatorId == userId && x.Id == id, cancellationToken);

public async Task<bool> HasAccessToChapter(Guid userId, Guid id, CancellationToken cancellationToken)

=> await HasAccessToEntity(userId, \_chapterContext.Chapters,

x => x.Course.CreatorId == userId && x.Id == id, cancellationToken);

public async Task<bool> HasAccessToVideo(Guid userId, Guid id, CancellationToken cancellationToken)

=> await HasAccessToEntity(userId, \_videoContext.Videos,

x => x.Chapter.Course.CreatorId == userId && x.Id == id, cancellationToken);

public async Task<bool> HasAccessToCreateComment(Guid userId, Guid id, CancellationToken cancellationToken)

=> await HasAccessToEntity(userId, \_courseContext.Courses,

x => (x.CreatorId == userId || x.Students.Any(x => x.Id == userId)) && x.Id == id, cancellationToken);

private async Task<bool> IsAdmin(Guid id, CancellationToken cancellationToken)

=> await \_userContext.Users

.AnyAsync(u => u.Id == id

&& u.Roles.Any(r => r.Name == AdminRoleName), cancellationToken);

private async Task<bool> HasAccessToEntity<T>(Guid userId, DbSet<T> context,

Expression<Func<T, bool>> expression, CancellationToken cancellationToken)

where T : class

=> await IsAdmin(userId, cancellationToken) || await context.AnyAsync(expression, cancellationToken);

Також контроль доступу забезпечується використання атрибутів [Authorize] в контролерах з указанням дозволених ролей: приклад використання наведено нижче :

[Authorize(Roles = "Admin")]

[HttpGet]

[Route("api/users")]

public async Task<ActionResult<List<UserDto>>> GetUsers(CancellationToken cancellationToken = default, int skip = 0, int take = 10, string query="", string sortOption = "Email", bool reverse = false, Guid? courseId = null)

{

var content = await \_userRepository.GetList(skip, take, query, sortOption, reverse, cancellationToken, courseId);

Response.Headers.Add("X-Total-Count", content.TotalCount.ToString());

return Ok(content.List);

}

## **4.2. Налаштування параметрів роботи Postgresql.**

Для підклювення Postgresql до ASP.NET потрібно отримати з віддаленого сервера Posgresql стрічку підклювення, далі передаєму її в зміні оточення проекту, а потім уже робимо переконстрюювання в стрічка, задовільняє потреби метода бібліотеки, яка дає змогу підключення до БД. Код переконструювання наведено нижче :

public static string GetConnectionString(IConfiguration configuration)

{

string connectionString = Environment.GetEnvironmentVariable("DATABASE\_URL");

if (string.IsNullOrEmpty(connectionString))

connectionString = configuration.GetConnectionString("DbConnection");

connectionString = connectionString.Split("//")[1];

string user = connectionString.Split(':')[0];

connectionString = connectionString.Replace(user, "").Substring(1);

string password = connectionString.Split('@')[0];

connectionString = connectionString.Replace(password, "").Substring(1);

string server = connectionString.Split(':')[0];

connectionString = connectionString.Replace(server, "").Substring(1);

string port = connectionString.Split('/')[0];

string database = connectionString.Split('/')[1];

connectionString = $"Host={server};Port={port};Database={database};Username={user};Password={password};";

if (server != "localhost")

connectionString = $"{connectionString}Sslmode=Require;Trust Server Certificate=true";

return connectionString;

}

Пояснення :

Спочатку ми розбираємо отриману стрічку підклювення на user, password, server, port та databaseName змінні, а потім складаємо по шаблону в відповідну стрічку підключення, також для коректної роботи з віддаленою базою додаємо підтримку мертифікатів безпеки. Після чого уже можемо передати цю стрічку в діповідний метод, лістинг якого наведено нижче :

services.AddDbContext<IskillsContext>(options =>

options.UseNpgsql(IskillsContext.GetConnectionString(configuration)));

Для створення віддаленої бази postgresql потрібно зареєструватися на heroku, після цього творити проект та додати доповнення Heroku Postgres.

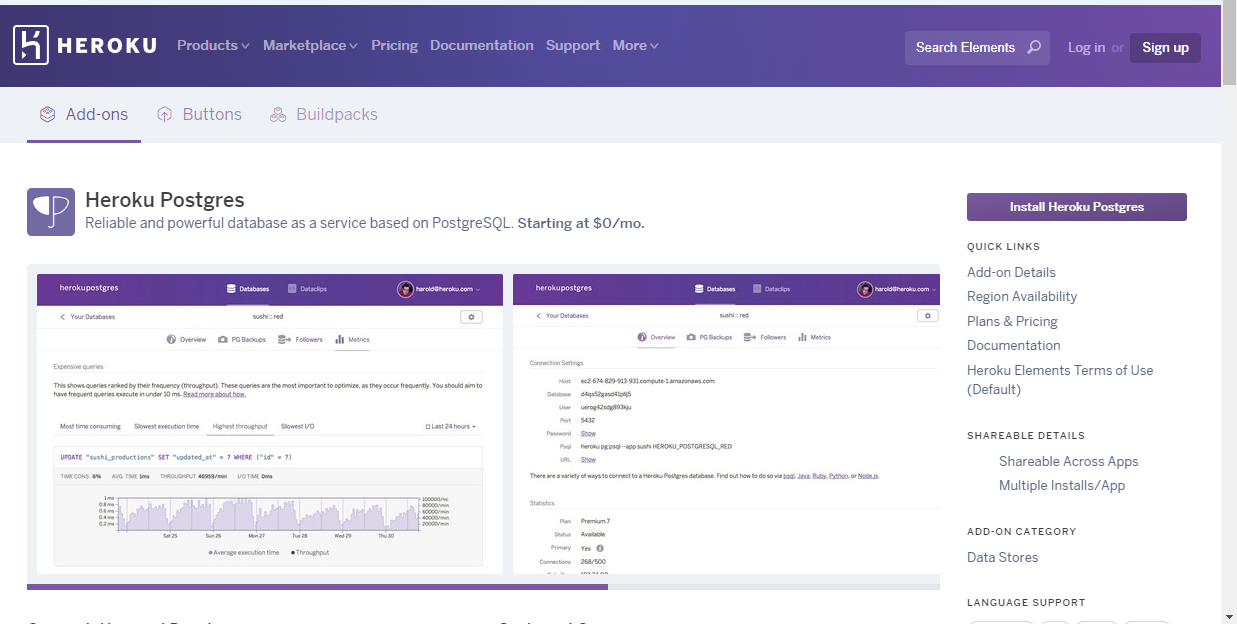


Рис 36. Heroku Postgres add-on

Далі переходимо на сторінку створеного проекту на heroku, переходимо в налаштування та бачимо, що уже додалася стрічка підклювення до БД.

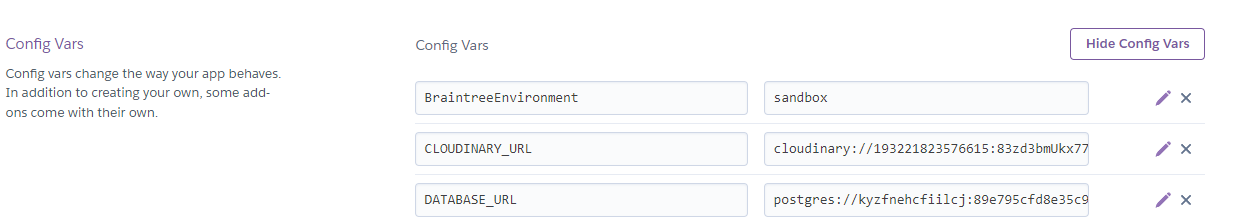


Рис 37. Пошук стрічки підклювення для віддаленого сервера Postgresql

Також варти мати на увазі, оскільки дана база є безкоштовною то має свої обмеження, а саме максимальню кількість записів 10 тисяч.

## **Висновки до четвертого розділу**

У ході виконання даного розділу проведено налаштування прав доступу користувача до бази, а саме контроль за допомогою ролей та певних контролюючих доступ методів. Також у ході виконання цього розділу було показано, як саму відбулось підключення додатку до БД, створення та наштування віддаленої бази даних Postgresql.

# **ВИСНОВКИ**

Під час написання даної курсової роботи було отримано навички роботи з віддаленими базами даних, створення резервних копій та відновлення з них даних баз даних.

В першому розділі курсової роботи, було проаналізовано три СУБД, визначено переваги кожної з СУБД та обрано одну для реалізації додатку. Також було визначено головні функції програми та приблизну логіку доступу до даних.

В другому розділі було проаналізовано інформаційні процеси, спроектовано структуру бази даних, описано приблизну логіку функцій для роботи з базою на клієнті, пояснено деяку логіку обробки, збереження та генерації даних для бази. Також було описано як буде відбуватись побудова та генерація статистичної інформації платформи.

В третьому розділі відбулось написання повного коду для взаємодії користувача з БД та спроектовано інтерфейс платформи курсів.. Крім цього, було створено можливість зберігати поточний стан бази, та відновлювати його з резервної копії. Також було написано та описано логіку та код виконання звітності системи.

В четвертому розділі проведено налаштування прав доступу користувача до бази, а саме контроль за допомогою ролей та певних контролюючих доступ методів. Також у ході виконання цього розділу було показано, як саму відбулось створення, наштування та підключення додатку до віддаленої бази даних Postgresql.

В результаті виконання курсової роботи отримано платформу для проведення курсів, яка повністю відповідає запланованому функціоналу, умовам та вимогам, поставленим на початку проектування, підтримуючи резервне копіювання бази даних та певну статистику.

# **СПИСОК ЛІТЕРАТУРИ**

1. Документація по C# [електронний ресурс]. Режим доступу : <https://docs.microsoft.com/ru-ru/dotnet/csharp/>
2. Документація по ASP.NET (5 та 6) [електронний ресурс]. Режим доступу : <https://docs.microsoft.com/ru-ru/aspnet/core>
3. Документація по Postgresql [електронний ресурс]. Режим доступу : <https://www.postgresql.org/docs/>
4. ASP.NET CORE WITH POSTGRESQL AND ENTITY FRAMEWORK CORE [електронний ресурс] – 2016. Режим доступу : <https://damienbod.com/2016/01/11/asp-net-5-with-postgresql-and-entity-framework-7/>
5. Документація по Cloudinary [електронний ресурс]. Режим доступу : <https://cloudinary.com/documentation/dotnet_integration>
6. Документація по Braintree [електронний ресурс]. Режим доступу : <https://developer.paypal.com/braintree/docs/start/hello-client>
7. Документація по React [електронний ресурс]. Режим доступу : <https://uk.reactjs.org/docs/getting-started.html>
8. Документація по React + Redux [електронний ресурс]. Режим доступу : <https://react-redux.js.org/introduction/getting-started>
9. Документація по Mui [електронний ресурс]. Режим доступу : <https://mui.com/material-ui/getting-started/overview/>
10. Документація по LINQ [електронний ресурс]. Режим доступу : <https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/linq/>

# **ДОДАТКИ**

## Додаток А

**Технічне завдання**

**1. Загальне положення**

**1.1 Найменування програмного засобу**

Повне найменування програмної системи: "База даних платформи для проведення курсів". (надалі “програма”)

**1.2 Призначення розробки та область застосування**

Програмна система "База даних платформи для проведення курсів" призначена для збору, обробки та відображення даних про курси, що є на платформі. В програмі передбачено використання механізму пошуку по параметрам доступних курсів та інших сушностей бази даних. За результатами обробки даних програма має формувати звітність за певний період.

**1.3.Мета**

Програмна система "База даних платформи курсів" дає можливість використовувати повноцінну платформу курсів, можливість пошуку курсів для користувача та перегляду певної інформації про наявні курси, також забезпечить керівництво підприємства своєчасною та повною інформацією про актуальний стан роботи платформи у повному обсязі.

**1.4. Найменування розробника та замовника.**

Розробник даного продукту - студент групи ІПЗ-20-4 Кормиш Роман (надалі “розробник”).

Замовник програмного продукту – кафедра інженерії програмного забезпечення Житомирського державного університету “Житомирська політехніка”, в межах виконання курсової з дисципліни «Бази даних» (надалі замовник).

**2. Підстава для розробки**

**2.1. Документ на підставі якого ведеться розробка**

Робота ведеться на підставі навчального плану за напрямом 6.050201 «Інженерія программного забезпечення» наказу про закріплення тем курсових робіт за студентами.

**3. Вимоги до програми**

**3.1. Вимоги до функціональних характеристик.**

**3.1.1. Загальні вимоги**

Програмна система має забезпечувати:

* можливість дистанційної роботи з робочих станцій локальної та глобальної мережі підприємства;
* інтерфейс користувача, що не залежить від операційної системи;
* постійний доступ користувачів до БД;
* оптимальне збереження даних (за обсягом та структурою);
* аутентифікацію користувачів та захист інформації від несанкціонованого доступу;
* надійне збереження даних та можливість відновлення даних у випадку непередбачуваних збоїв системи;
* можливість модернізації системи через зміну функціональних потреб користувача або модернізацію обладнання;

**3.1.1 Склад виконуваних функцій**

Розробити базу даних платформи курсів, що підтримує виконання наступних транзакцій:

1. Реєстрацію користувачів;

2. Додання курсу, розділу, відео, коментарю;

3. Придбання курсу;

4. Зміна даних про курс, розділ, відео, коментар.

5. Пошук та відбір курсів по мультикритеріями (назва, тема, автор).

6. Форммування звітності щодо курсів :

* статистика покупов за певний період по всім курсам
* статистика покупов за певний період по конкретному курсу
* звітність про авторів курсів, а саме віднощення рейтингу та кількості створених курсів

**3.1.2. Організація вхідних і вихідних даних**

Вхідними даними є дані про курси(назва, автор, ціна і т.д.).

Організація вхідних і вихідних даних повинна відповідати інформаційній структурі виконуваних операцій, вхідним та вихідним паперовим документами. Введення оперативних даних повинно виконуватися з використанням діалогових екранних форм, побудованих на основі візуальних компонентів. Введення даних виконується на основі затверджених форм документів: заява на додання курсу, заява на додання розділу чи відео, форма з покупкою курсу і т.д. Будь який звіт повинен мати своє представлення.

**3.1.3. Часові характеристики і розмір пам'яті, необхідної для роботи програми.**

Час реакції програми на дії користувача (маніпуляції з пристроями введення даних) не повинен перевищувати 0,25 с. Час виконання команд меню не більше 1 с.

Відображення масивів даних за запитами не більше 3 хвилин.

Доступність БД – 90% цілодобово.

Операції з’єднання з БД не більше 1 хвилини.

Обсяг оперативної пам'яті, необхідний для роботи програми не менше 1Гб.

Дисковій простір, необхідний для збереження програми і файлів даних не більше 300 Мбат для робочої станції та 20 Гбайт.

Інсталяційний пакет програми, що містить у складі БД не повинні перевищувати 100 Мбайт.

**3.2. Вимоги до надійності.**

**3.2.1. Вимоги до надійного функціонування**

Програма повинна нормально функціонувати при безперебійній роботі ПК.

Доступність

БД 90% при одночасному доступі 30 користувачів.

При апаратних збоях, відновлення нормальної роботи програми повинне виконуватися після:

1. апаратні збої сервера - перезавантаження ОС сервера, запуск сервера БД (запуску резервного сервера, використання технологій RAID для збереження даних);
2. апаратні збої робочої станції – перезавантаження ОС ПК запуск виконуваного файлу програми.
3. БД повинна повертатись в найближчий несуперечний стан – передбачити точки відновлення.

При збоях програмного забезпечення:

1. система повинна забезпечувати можливість відновлення даних та фіксацію і «відкат» транзакцій.
2. в системі має бути реалізована коректна обробка виняткових ситуацій.

**3.2.2.Контроль вхідної і вихідної інформації**

Для контролю коректності вхідної інформації та захисту від помилок оператора:

* Перевірка відповідності даних доменам інформаційних атрибутів;
* Використанням механізму авто заповнення та вибору за переліком для зв’язаних даних;
* Захист від помилок оператора (залипань, випадкових символів тощо).

Визначені некоректні дії повинні супроводжуватись повідомленнями про помилку і блокуванням операцій оновлення даних. В системі має бути передбачений захист від загального блокування.

Для вихідної інформації передбачити:

* відсутність блокування даних через багато користувацький доступ;
* постійне оновлення даних у відображених на екрані звітних формах.
* повідомлення про використання режиму монопольного доступу супервайзерами БД

**3.2.3. Час відновлення після відмови.**

Час відновлення після відмови, не пов'язаною з роботою програми, повинен складатися із: часу перезапуску операційної системи; часу запуску сервера БД (підключення до сервера) запуску виконуваного файлу, часу повторного введення або зчитування даних.

**3.3. Умови експлуатації і збереження**

Програма використовується у багато користувацькому середовище. Регламенті операції проводити за графіком:

* створення резервних копій даних: 1 раз на тиждень;
* збереження резервної копії журналу транзакцій – 1 раз на тиждень;
* архівування даних – 1 раз на місяць;
* обслуговуючі функції (прибирання “сміття”, де фрагментація тощо) - щоденне;
* оновлення системи ідентифікації та аутентифікації користувачів – 1 раз на місяць та за потребою.

Збереження архівних копій - на визначених носіях. Збереження резервних копій – на визначеному диску без перезапису. Знищення копій за регламентом.

**3.4. Вимоги до інформаційної і програмної сумісності.**

**3.4.1. Вимоги до інформаційних структур на вході і виході**

Формат відображення даних має дозволяти імпорт даних в додатки MS Office для редагування та перегляду.

**3.4.2. Вимоги до методів рішення і мов програмування**

Вибір методів рішення здійснюється розробникам без узгодження з замовником. СУБД обирається у відповідності до характеристик визначених в п.3. З замовником погоджується вибір варіанту за вартісною ознакою.

**3.4.3. Вимоги до системи програмних засобів.**

Вимоги до програмного забезпечення сервера:

ОС – серверна версія ОС Windows не нижче Windows 2000 Server, IIS (або інший сервер), СУБД визначається встановлюється та налаштовується розробником (ліцензування СУБД виконується замовником).

Вимоги до програмного забезпечення робочої станції:

ОС - родина Windows не нижче Windows XP.

Драйвери периферійних пристроїв - введення/виводу визначаються та встановлюються при встановленні ОС в залежності від конфігурації робочої станції.

Пакет додатків MS Office.

Робоча станція є клієнтом мереж Microsoft.

**3.5. Вимоги до складу і параметрів технічних засобів.**

Вимоги до складу технічних засобів:

1. Сервер:
   * сервер у базовій конфігурації із підтримкою RAID;
   * з’ємний запам’ятовуючий пристрій для архівування даних;
   * принтер для друку;
   * засоби для під’єднання до локальної мережі.
2. Робоча станція:
   * ПК на базі процесорів Intel, AMD у стандартній комплектації;
   * периферійні пристрої друку;
   * засоби для під’єднання до локальної мережі. засоби для під’єднання до локальної мережі.
   * Pentium III з тактовою частотою 750 МГц;
   * оперативна пам’ять - 128Мбайт;
   * об’єм дискової пам’яті - 20Гбайт.

**4. Вимоги до програмної документації**

Програмна документація повинна включати наступні відомості:

1. "Інструкція по інсталяції (встановленню) програми", складається з опису інсталяційного пакету, переліку етапів інсталяції та їх послідовності, опису встановлених програмних компонентів та режимів їх роботи після інсталяції. Під час оформлення пояснювальної записки до курсової роботи дані відомості містяться у 4 розділі.

2. Керівництво користувача" складається з опису послідовності завантаження програми, основних режимів роботи, опису основних екранних форм, переліку виняткових ситуацій та реакції користувача на них, порядку виконання завдань в системі. Під час оформлення пояснювальної записки до курсової роботи дані відомості містяться в 3 розділі курсової роботи.

3. "Керівництво адміністратора баз даних", складається з опису складу таблиць бази даних та правил доступів до них, опису та послідовності робіт по обслуговуванню бази даних (архівування, резервне копіювання, з вказівкою періодичності виконання та засобів, що для цього використовуються), порядку налаштування серверу та клієнтських додатків. Під час оформлення пояснювальної записки до курсової роботи дані відомості містяться в 2,3 та 4 розділах курсової роботи згідно плану.