МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Факультет математики і інформатики

Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

бакалавр

на тему «Проектування та розробка серверної частини для сайту Студабр»

Виконав: студент 4 курсу, групи МФ-41

спеціальність 122 «Комп’ютерні науки»

освітньо-професійна програма «Теоретична і прикладна інформатика»

Фартушний В. С.

Керівник Бережна Н. І.

Рецензент

Харків – 2024 року

ЗМІСТ

ВСТУП……..3

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГ ДО СЕРВЕРНОЇ ЧАСТИНИ САЙТУ СТУДАБР..……..5

1.1. Формулювання задачі та обґрунтування актуальності теми ……..5

1.2. Аналіз технологій, що використовувались ……...6

1.3. Аналіз функціоналу, який потрібно розробити ……..10

2. ПРОЕКТУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ САЙТУ СТУДАБР ……...11

2.1. Архітектурні рішення та структура системи ……...11

2.2. Опис бази даних ……...13

2.3. Опис кінцевих точок ……16

3. РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ САЙТУ СТУДАБР ……18

3.1. Вибір та налаштування середовища розробки ……18

3.2. Опис архітектури..……..20

3.3. Реалізація основних модулів та функцій ……27

3.4. Реалізація системи безпеки ……...28

4. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ СЕРВЕРНОЇ ЧАСТИНИ САЙТУ СТУДАБР ……...29

4.1. Методи та інструменти тестування ……..29

4.2. Результати тестування ……30

4.3. Впровадження та розгортання серверної частини ……..31

ВИСНОВОК ……..32

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ……....35

ВСТУП

У сучасному цифровому світі інформаційні технології відіграють ключову роль у забезпеченні доступу до знань і ресурсів. Розвиток інтернету та його можливостей дозволяє створювати різноманітні платформи для обміну інформацією, що сприяє розвитку інтелектуального потенціалу суспільства. Однією з таких платформ є сайт Студабр, призначений для допомоги студентам у різних питаннях. На сайті публікуватимуться статті на різні тематики, написані іншими студентами, що можуть бути корисними для читачів у їхньому навчанні та повсякденному житті.

Мета і завдання дослідження

Метою цієї роботи є розробка серверної частини сайту Студабр, яка забезпечить стабільне і ефективне функціонування платформи для публікації та взаємодії зі статтями. Основні завдання дослідження включають:

1. Аналіз предметної області та вимог до серверної частини сайту Студабр.
2. Проектування серверної частини сайту Студабр.
3. Розробка серверної частини сайту Студабр.
4. Тестування та впровадження серверної частини сайту Студабр.

Актуальність теми

Актуальність теми зумовлена зростаючою потребою в освітніх ресурсах та платформах для самоосвіти. В умовах постійного розвитку інформаційного суспільства доступ до якісної інформації стає дуже важливим. Студабр задовольнить цю потребу, надаючи зручний інструмент для створення та поширення знань.

Методи дослідження

Для досягнення поставленої мети в роботі використано такі методи дослідження:

1. Аналіз літературних джерел: Вивчення наукових публікацій та технічної документації.
2. Проектування системи: Використання методів об'єктно-орієнтованого проектування для створення архітектури серверної частини.
3. Розробка програмного забезпечення: Використання мови програмування TypeScript і фреймворку NestJS для реалізації серверної логіки.
4. Тестування: Застосування методів модульного та інтеграційного тестування для перевірки коректності роботи серверної частини.

Застосування цих методів дозволило забезпечити достовірність отриманих результатів і висновків, а також забезпечити надійне функціонування серверної частини сайту Студабр.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГ ДО СЕРВЕРНОЇ ЧАСТИНИ САЙТУ СТУДАБР

1.1. Формулювання задачі та обґрунтування актуальності теми

Задачею даної дипломної роботи є розробка серверної частини сайту Студабр, яка забезпечить стабільне і ефективне функціонування платформи для публікації та взаємодії зі статтями та іншими користувачами. Сайт має стати інструментом для обміну знаннями між користувачами, що потребує надійної інфраструктури для обробки даних, управління користувачами та забезпечення безпеки інформації.

Актуальність теми зумовлена зростаючою потребою в якісних освітніх ресурсах, безкоштовних платформах для самоосвіти та полегшенню життя студентів.

1.2. Аналіз технологій, що використовувались

Розробка серверної частини сайту Студабр потребує використання сучасних технологій та інструментів, що забезпечують високий рівень продуктивності, масштабованості та безпеки. Основні технології, що використовуються, включають:

1. **Мова програмування та фреймворки:** Для розробки серверної логіки використано мову програмування TypeScript, котра конвертується в JavaScript, разом із фреймворком NestJS, що забезпечує асинхронну обробку запитів і високу продуктивність.

   Альтернативою могли б бути мови та фреймворки, такі як Python з Django, або Java з SpringBoot. Хоча Django є потужними та гнучкими фреймворками, вони не забезпечують такого рівня асинхронності, який є критичним для високонавантажених систем. Java з SpringBoot теж потужний фреймворк для розробки серверної частини, але вони не підходять для даного сервера, так як розробка на ній складніше та більше ресурсомістке в порівнянні з NestJS.

   Висновок: Тому вирішено використати TypeScript разом із NestJS, так як він забезпечує кращу продуктивність і масштабованість завдяки асинхронній обробці запитів.

2. **Бази даних:** Для зберігання інформації про користувачів, статті та взаємодії між ними використано нереляційну базу даних MongoDB.

   Альтернативою могла б бути реляційна база даних, наприклад, MySQL. Реляційні бази даних мають свої переваги в забезпеченні цілісності даних і підтримці складних запитів, але вони можуть бути менш ефективними при роботі з великими обсягами нереляційних даних.

   Переваги MongoDB в порівнянні з реляційними базами даних:

   1. Гнучкість даних: MongoDB дозволяє зберігати дані у форматі JSON, що надає велику гнучкість. Реляційні ж бази даних потребують суворого дотримання схем даних, що обмежує динамічність.

2. Масштабованість: MongoDB розроблено з урахуванням горизонтальної масштабованості (додавання нових серверів для розподілу навантаження), що дозволяє легко розподіляти дані по кількох серверах і підтримувати високу продуктивність при збільшенні обсягів даних. Реляційні бази даних зазвичай краще підходять для вертикальної масштабованості (додавання потужності існуючим серверам), яка може бути дорожчою та менш гнучкою.
3. Швидкість обробки: MongoDB оптимізовано для швидкої обробки великих обсягів нереляційних даних, що робить її більш ефективною для багатьох операцій введення-виведення. Реляційні бази даних можуть бути менш ефективними при роботі з великими нереляційними даними через накладні витрати на підтримку цілісності даних та індексацію.

Висновок: Використання MongoDB забезпечує гнучкість і масштабованість при роботі з великими обсягами даних і складними структурами, що робить її оптимальним вибором для цього проекту.

3. **API:** Для взаємодії між клієнтською та серверною частинами буде розроблено RESTful API, що дозволяє здійснювати CRUD-операції (створення, читання, оновлення, видалення) з даними. Альтернативою є використання GraphQL, який надає більшу гнучкість у запитах до серверної частини та зменшує обсяг даних, що передаються.

   Переваги RESTful API порівняно з GraphQL:

   1. Підтримка: У RESTful API є широка підтримка у спільноті розробників. Багато бібліотек та інструментів підтримують RESTful API, що спрощує інтеграцію та забезпечує стабільність.
   2. Простота реалізації: RESTful API базується на стандартних HTTP методах (GET, POST, PUT, DELETE), що робить його простим у реалізації. GraphQL, хоча і надає більшу гнучкість, може бути складнішим у налаштуванні та оптимізації.
   3. Відповідність потребам: RESTful API добре підходить для сценаріїв, де потрібні чітко визначені кінцеві точки для різних ресурсів. GraphQL може бути надмірним для простих CRUD-операцій, які є основними для даного сайту.

Висновок: Використання RESTful API забезпечує стабільність і простоту інтеграції, що є важливим для забезпечення надійної взаємодії між клієнтською та серверною частинами сайту Студабр.

4. **Безпека:** Для забезпечення безпеки даних користувачів буде застосовано технології шифрування та захисту від SQL-ін'єкцій, а також реалізовано механізми аутентифікації та авторизації. Для аутентифікації використовується JWT-токен (JSON Web Tokens), що є одним із найпопулярніших і безпечних стандартів.

   Переваги JWT-токен порівняно з іншими механізмами аутентифікації:

   1. Легкість у використанні: JWT-токени мають просту структуру і легко використовуються для аутентифікації між клієнтом та сервером. Вони можуть бути закодовані та декодовані за допомогою простих бібліотек.
   2. Безпека: JWT-токени підписуються за допомогою секретного ключа або RSA-ключа, що забезпечує їх цілісність і захист від підробки. Токени можуть бути захищені від перехоплення шляхом використання HTTPS.
   3. Незалежність від стану: JWT-токени є самодостатніми і не вимагають збереження сесій на сервері. Вся інформація, необхідна для аутентифікації та авторизації, міститься у самому токені.
   4. Масштабованість: Завдяки незалежності від стану, JWT легко масштабуються на різні сервери, що робить їх зручними для використання в розподілених системах.

   Використання JWT для аутентифікації

   Процес аутентифікації:

   Користувач надсилає свої облікові дані (ім'я користувача/електронну пошту та пароль) на сервер.

   Сервер перевіряє облікові дані і, якщо вони правильні, створює JWT-токен з інформацією про користувача (ID, ролі тощо).

   Токен підписується секретним ключем і повертається клієнту.

   Клієнт зберігає токен, наприклад, у localStorage або в кукі, та надсилає його з кожним наступним запитом на сервер.

Процес авторизації:

Сервер отримує запит з токеном від клієнта.

Токен розшифровується та перевіряється на цілісність та коректність.

Якщо токен дійсний, сервер надає доступ до захищених ресурсів відповідно до ролей та прав, що містяться в токені.

Висновок: Використання JWT-токенів забезпечує високий рівень безпеки та зручність для користувачів, роблячи процес аутентифікації та авторизації на сайті Студабр простим та надійним. Цей підхід також сприяє масштабованості системи та полегшує захист даних користувачів.

Таким чином, аналіз технологій дозволяє визначити оптимальні рішення для реалізації серверної частини сайту Студабр, що забезпечать його надійне та ефективне функціонування. Вибір TypeScript з NestJS, MongoDB, RESTful API, та JWT-токен обґрунтований їхньою продуктивністю, масштабованістю, гнучкістю та рівнем контролю, що є ключовими для успішного розвитку проекту.

1.3. Аналіз функціоналу, який потрібно розробити

Для забезпечення повноцінного функціонування сайту Студабр потрібно розробити наступний функціонал:

1. Реєстрація та аутентифікація користувачів: Створення системи для реєстрації нових користувачів та подальший вхід у систему.
2. Публікація статей: Надання можливості користувачам маніпулювати своїми статтями, а саме створювати, редагувати та видаляти.
3. Лайки та збереження статей в свій список: Реалізація функції взаємодії з контентом через лайки та збереження статей до особистого списку статей.
4. Підписка на авторів: Створення системи підписки на улюблених авторів, щоб отримувати сповіщення про їх нові публікації.
5. Ієрархія статей: Можливість зв'язувати статті між собою, створюючи логічні ланцюги інформації.
6. Адмін панель: Надання адміністраторам можливості керувати контентом сайту та правами користувачів.

Цей функціонал забезпечує виконання основних задач сайту Студабр.

2. ПРОЕКТУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ САЙТУ СТУДАБР

2.1. Архітектурні рішення та структура системи

Серверна частина сайту Студабр використовує мікросервісний архітектурний стиль. Мікросервіси забезпечують гнучкість, масштабованість та легкість в обслуговуванні системи, оскільки кожна частина функціоналу розділена на сервіси. Основні компоненти системи включають:

- Auth Service (Сервіс аутентифікації): Відповідає за реєстрацію, логін та управління користувацькими сесіями, використовуючи JWT-токен для забезпечення безпечного доступу.
- User Service (Сервіс користувачів): Обробляє інформацію про користувачів, їх профілі та взаємодію з іншими користувачами (наприклад, фоловінг).
- Article Service (Сервіс статей): Відповідає за створення, редагування, видалення та пошук статей, а також управління вподобаннями.
- Follow Service (Сервіс фоловерів): Обробляє дані про взаємодію користувачів у контексті підписки один на одного.

Мікросервісна архітектура дозволяє кожному сервісу бути незалежним, що полегшує їх розгортання, тестування та масштабування. Кожен сервіс має власну базу даних, що дозволяє уникнути конфліктів та покращує продуктивність.

USE CASE діаграма

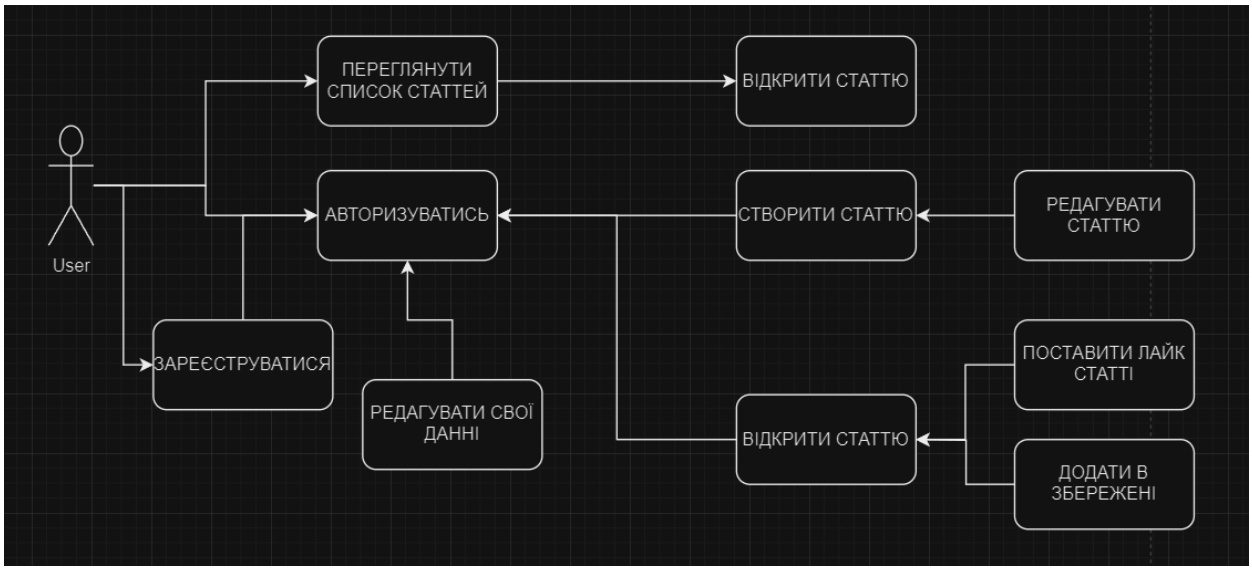

Рисунок 1. Діаграма варіантів використання для ролі Користувач

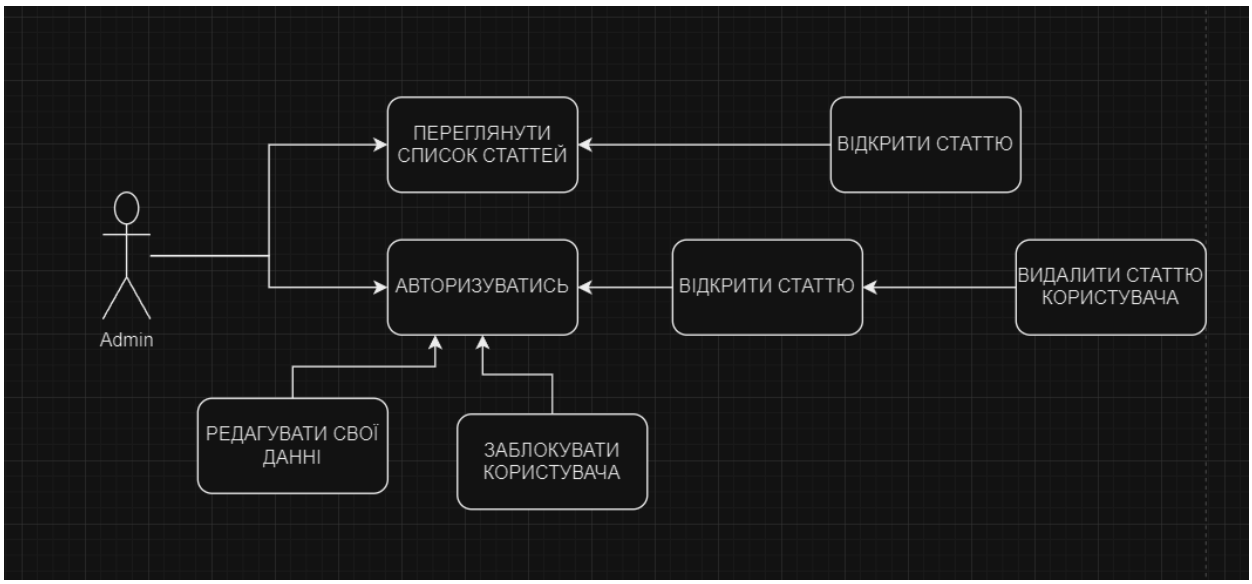

Рисунок 2. Діаграма варіантів використання для ролі Адмін

2.2. Опис бази даних

Для зберігання даних використовується нереляційна база даних MongoDB, що забезпечує гнучкість у роботі з динамічними структурами даних і високий рівень продуктивності. Основні колекції та їх структура:

- Користувачі (Users):

```
@Schema()
export class User {
  @Prop() id: number;
  @Prop() slug: string;
  @Prop() email: string;
  @Prop() username: string;
  @Prop() bio: string;
  @Prop() image: string;
  @Prop() password: string;
  @Prop({ type: [{ type: mongoose.Schema.Types.String, ref: 'Article' }] })
  articles?: Article[];
  @Prop({ type: [{ type: mongoose.Schema.Types.String, ref: 'Article' }] })
  favorites?: string[];
}
```

  У цій колекції зберігається інформація про користувачів, включаючи їх статті та вподобані статті.

- Статті (Articles):

```
@Schema()
export class Article {
```

```
  @Prop() id: number;
  @Prop() slug: string;
  @Prop() title: string;
  @Prop() subtitle: string;
  @Prop() content: string;
  @Prop() mainImageLink: string;
  @Prop() createdAt: Date;
  @Prop() updatedAt: Date;
  @Prop([String]) tagList: string[];
  @Prop() favoritesCount: number;
  @Prop({ type: User, ref: 'User' }) author: any;
}
```

У цій колекції зберігається інформація про статті, включаючи їх авторів та кількість вподобань.

- Фоловери (Followers):

```
@Schema()
export class Follow {
  @Prop() id: number;
  @Prop({ type: String, required: true }) followerId: string;
  @Prop({ type: String, required: true }) followingId: string;
}
```

У цій колекції зберігається інформація про підписки користувачів один на одного.

Схема бази даних

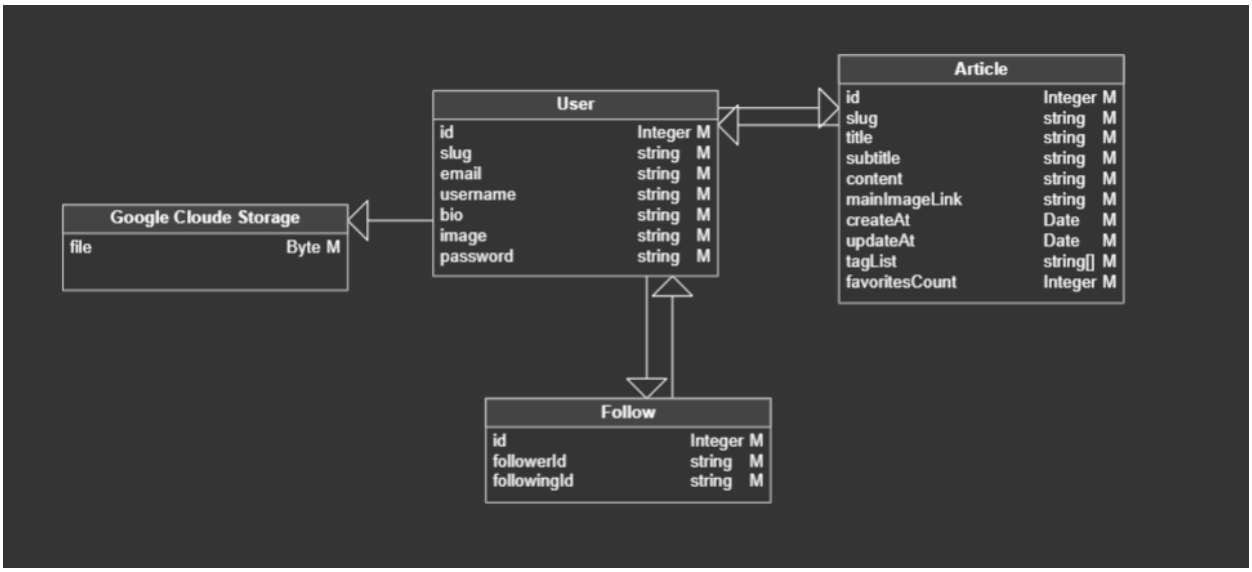

Рисунок 3. Схема бази даних

2.3. Опис кінцевих точок

Кінцеві точки (API endpoints) серверної частини сайту Студабр реалізовані з використанням фреймворку NestJS. Основні контролери та їх кінцеві точки включають:

- Article Controller:

  GET /articles/get-all: Повертає всі статті.

  POST /articles/create: Створює нову статтю (вимагає аутентифікації).

  GET /articles/get-single/:slug: Повертає конкретну статтю за її slug.

  DELETE /articles/delete/:slug: Видаляє статтю (вимагає аутентифікації).

  PUT /articles/update/:slug: Оновлює статтю (вимагає аутентифікації).

  POST /articles/:slug/favorite: Додає статтю до вподобаних (вимагає аутентифікації).

  DELETE /articles/:slug/unfavorite: Видаляє статтю з вподобаних (вимагає аутентифікації).

  GET /articles/search-by-keyword: Пошук статей за ключовими словами.

- User Controller:

  POST /user/create: Створює нового користувача.

  POST /user/login: Логін користувача.

  GET /user/current-user: Повертає поточного користувача (вимагає аутентифікації).

  PUT /user/update-current: Оновлює дані поточного користувача (вимагає аутентифікації).

  GET /user/get-single/:slug: Повертає дані конкретного користувача за його slug.

- Profile Controller:

  GET /profiles/get-profile/:username: Повертає профіль користувача (вимагає аутентифікації).

  POST /profiles/:username/follow: Підписується на користувача (вимагає аутентифікації).

  DELETE /profiles/:username/unfollow: Відписується від користувача (вимагає аутентифікації).

- Follow Controller:

  POST /follow: Створює новий запис про підписку.

  DELETE /unfollow/:id: Видаляє запис про підписку.

- GCS(google cloud storage) Controller:

  POST /storage/upload: кладе файл на сервер google cloud storage.

  GET /storage/get/:filename: дістаємо файл із сервера google cloud storage за його назвою.

  DELETE /storage/delete/:filename: видаляє файл із сервера google cloud storage.

  POST /storage/create-bucket: створення нового бакета (директорії) для зберігання в ньому файлів.

Для забезпечення безпеки використовується JWT-токен для аутентифікації та авторизації користувачів. Кожен запит до захищених кінцевих точок перевіряється за допомогою токена доступу, який видається після успішної аутентифікації користувача.

3. РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ САЙТУ СТУДАБР

3.1. Вибір та налаштування середовища розробки

1. Середовище розробки:
    - Використовуване середовище: Visual Studio Code
    - Плагіни та розширення для полегшення роботи з кодом:
        - Auto Close Tag: автоматично закриває HTML та XML теги.
        - Auto Rename Tag: автоматично перейменовує відповідний тег, коли один змінюється.
        - ESLint: інтеграція з ESLint для виявлення та виправлення проблем у TypeScript коді.
        - Prettier - Code formatter: автоматичне форматування коду для забезпечення консистентності.
        - WSL: підтримка Windows Subsystem for Linux для розробки у Linux середовищі на Windows.
2. Мова програмування та інструменти:
    - Основна мова програмування: TypeScript, яка конвертується в JavaScript.
    - Інструменти та бібліотеки:
        - Node.js: серверна платформа для запуску JavaScript.
        - NestJS: фреймворк для створення масштабованих та ефективних серверних додатків на Node.js.
        - MongoDB: база даних NoSQL для зберігання даних.

3. Система контролю версій:

    - Використовувана система контролю версій: Git
    - Платформи для хостингу репозиторіїв: GitHub

4. Налаштування локального середовища:

    - Інструкції з налаштування локального середовища:
        - Встановити Node.js з офіційного сайту.
        - Запустити сервер MongoDB на офіційному сайті.
        - Клонувати репозиторій з GitHub: git clone <URL>
        - Перейти в директорію проекту: cd <project-directory>

    - Встановити необхідні бібліотеки: pnpm install
    - Запустити сервер: pnpm start:dev
- Налаштування конфігураційних файлів:
    - Знайти в корні проекту директурію common. В цій директорії знайти файл constants.ts.
    - Додати необхідні конфігураційні змінні, такі як:
    - MONGODB_URI=mongodb://localhost:<your_port>/studabr
    - JWT_SECRET=<your_jwt_secret>
    - BUCKET_NAME=<bucket_name>
    - KEY_GCS=<path_to_your_key>

3.2. Опис архітектури

Classes diagram

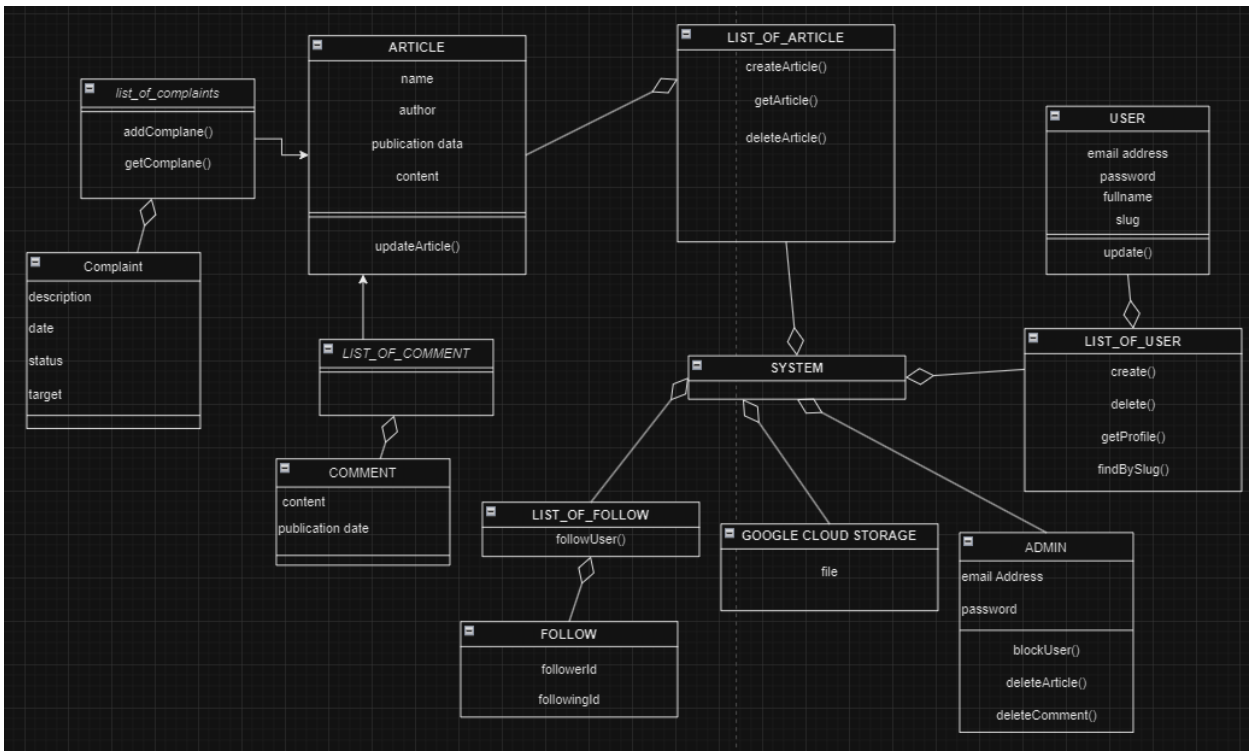

Рисунок 4. Діаграма класів

Activity diagram

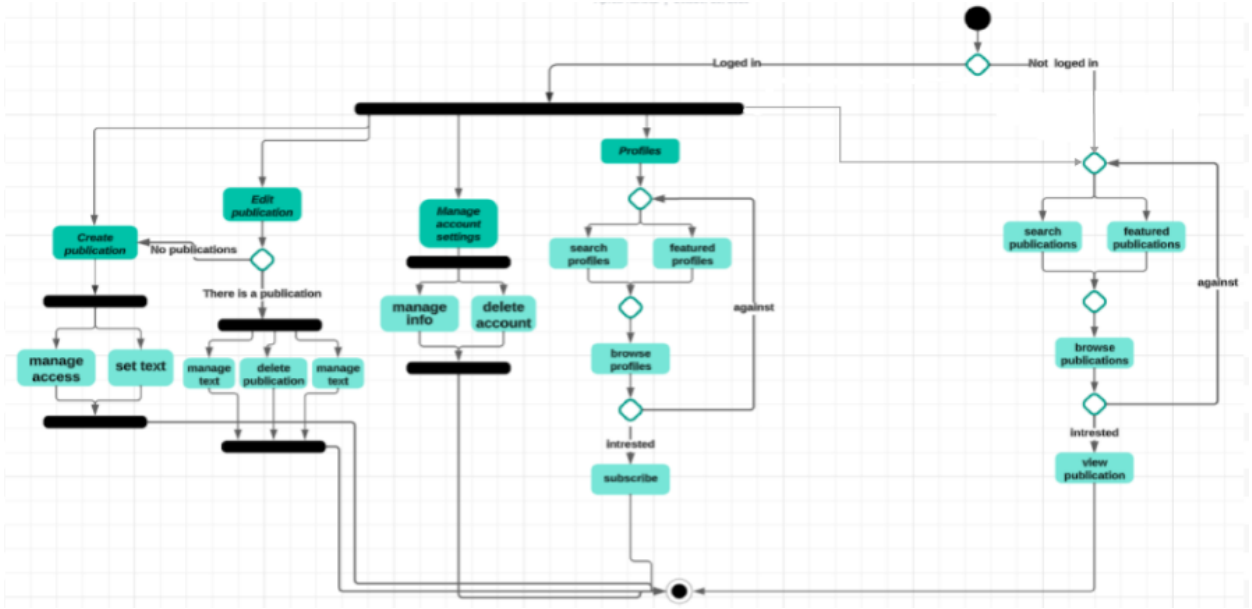

Рисунок 5. Актівіті діаграма

Опис прецедентів

1. Опублікувати статтю

Користувач може опублікувати статтю на нашому сайті.

Головною дійовою особою в цьому потоці є користувач.

Для цієї дії користувач повинен бути зареєстрованим.

Основний потік виглядає наступним чином:

Користувач: Натискає кнопку «Створити статтю».

Система: Відкриває сторінку редактора.

Умовні потоки:

Користувач: позначає частину тексту як код.

Система: Змінює візуальне представлення тексту до стилю коду.

Користувач: натискає кнопку «Змінити доступ».

Система: Відображає екран, на якому користувач може змінити налаштування доступу.

Кінцевим станом буде публікація статті на веб-сайті, зі вставками коду або без них, зі змінами доступу або без них.

2. Пошук статей за ключовими словами

Користувач може шукати статті за ключовими словами на нашій платформі.

Головною дійовою особою в цьому потоці є користувач.

Для цієї дії немає передумов.

Основний потік виглядає наступним чином:

Користувач: вводить ключові слова в пошуковий рядок і натискає кнопку «Search».

Система: Отримує список статей, що відповідають ключовим словам.

Умовні потоки:

Користувач: Вибирає статтю з результатів пошуку і вирішує переглянути її.

Система: Відображає вибрану статтю.

Користувач: додає коментарі до переглянутої статті.

Системний: Дозволяє користувачеві вводити і надсилати коментарі, якщо він зареєстрований.

Користувач: Переглядає статті і вирішує поскаржитися на одну з них.

Система: Надає користувачеві можливість подати скаргу на статтю, якщо він зареєстрований.

Кінцевий стан може бути різним:

Перегляд статей призведе до того, що користувач прочитає вибрані статті.

Додавання коментарів додасть коментарі користувача до обраної статті.

Подання скарги на статтю ініціює процес розгляду скарги, і стаття може підлягати перегляду або модерації.

3. Пошук профілів за іменем/прізвищем

Користувач може шукати профілі за іменем або прізвищем на нашій платформі.

Головною дійовою особою в цьому потоці є користувач.

Для цієї дії немає ніяких конкретних передумов.

Основний потік виглядає наступним чином:

Користувач: Вводить ім'я в пошуковий рядок і натискає кнопку «Search».

Система: Отримує список профілів, що відповідають введеному імені.

Умовні потоки:

Користувач: Вибирає профіль з результатів пошуку і вирішує переглянути його.

Система: Відображає вибраний профіль.

Користувач: Скаржиться на профіль.

Система: Надає користувачеві можливість подати скаргу на обраний профіль.

Умовні потоки:

Користувач: обирає профіль з результатів пошуку і натискає кнопку «Переглянути».

Система: Відображає вибраний профіль.

Користувач: підписується на профіль.

Система: Дозволяє користувачеві підписатися на вибраний профіль, якщо користувач зареєстрований.

Кінцевий стан може бути різним:

Перегляд профілів дозволяє користувачеві побачити вибрані профілі.

Подання скарги на профіль ініціює процес розгляду скарги, який може включати подальший розгляд або модерацію.

Підписка на профіль додає профіль до списку підписок користувача.

4. Керування обліковим записом

Користувач може керувати налаштуваннями свого облікового запису на нашій платформі.

Головною дійовою особою в цьому процесі є користувач.

Для цього ви повинні бути зареєстрованим користувачем.

Основний потік виглядає наступним чином:

Користувач: Переходить до розділу управління акаунтом.

Система: Надає опції для управління акаунтом, такі як редагування профілю або видалення акаунту.

Умовні потоки:

Користувач: Вибирає видалення акаунта.

Система: Проводить користувача через процес видалення акаунта.

Користувач: Вибирає редагування профілю.

Система: Надає користувачеві інструменти та опції для редагування інформації в його профілі.

Кінцевий стан може бути різним:

Видалення облікового запису призведе до видалення облікового запису користувача з видаленням пов'язаних з ним даних.

Редагування профілю дозволить користувачеві оновити або змінити інформацію свого профілю.

5. Auth/Reg (Аутентифікація/Реєстрація)

Користувач може виконувати дії, пов'язані з аутентифікацією та реєстрацією на нашій платформі.

Головною дійовою особою в цьому потоці є користувач.

Ніяких специфічних передумов для цих дій немає.

Основний потік виглядає наступним чином:

Користувач: Або входить (аутентифікується) до свого існуючого облікового запису, або реєструє новий обліковий запис.

Система: Надає можливості для аутентифікації (входу) та реєстрації.

Кінцевий стан може бути різним:

Аутентифікація дозволить користувачеві отримати доступ до існуючого облікового запису.

Реєстрація створює новий обліковий запис для користувача, зазвичай вимагаючи від нього надання необхідної інформації, такої як ім'я користувача, пароль та адреса електронної пошти.

6. Розгляд скарг

Адміністратор може розглядати скарги, подані користувачами на нашій платформі.

Головною дійовою особою в цьому процесі є адміністратор.

Для цієї дії немає ніяких конкретних передумов.

Основний потік виглядає наступним чином:

Адміністратор: Заходить в розділ розгляду скарг.

Система: Надає список скарг для розгляду.

Умовні потоки:

Адміністратор: Вибирає видалення скарг.

Система: Дозволяє адміністратору видалити вибрані скарги.

Видалити скарги

Адміністратор може видалити скарги, які були розглянуті.

Головною дійовою особою в цьому потоці є адміністратор.

Для цієї дії немає ніяких конкретних передумов.

Основний потік виглядає наступним чином:

Адміністратор: Вибирає скарги для видалення.

Система: Видаляє вибрані скарги з платформи.

Умовні потоки:

Адміністратор: Вирішує видалити статтю на основі скарги.

Система: Видаляє статтю з платформи.

Адміністратор: Вирішив заблокувати акаунт на підставі скарги.

Система: Блокує обліковий запис користувача, пов'язаний зі скаргою.

Кінцевий стан призведе до видалення вибраних скарг, статей або блокування облікових записів користувачів відповідно до рішень адміністратора під час процесу розгляду.

7. Управління всіма публікаціями

Адміністратор може керувати всіма публікаціями на нашій платформі.

Головною дійовою особою в цьому потоці є адміністратор.

Для цієї дії немає ніяких конкретних передумов.

Основний потік виглядає наступним чином:

Адміністратор: Отримує доступ до розділу управління публікаціями.

Система: Надає список всіх публікацій на платформі для перегляду та управління.

Умовні потоки:

Адміністратор: Вибирає видалення статті.

Система: Видаляє вибрану статтю з платформи.

Кінцевий стан призведе до видалення вибраної статті, як визначено рішенням адміністратора під час процесу управління.

3.3. Реалізація основних модулів та функцій

1. Опис основних модулів:
    - Модуль користувачів:
        - Реєстрація користувачів, логін, оновлення інформації про користувачів.
    - Модуль статей:
        - Створення, редагування, видалення, перегляд статей, додавання статей до улюблених, скарги.
    - Модуль фоловерів:
        - Підписка на користувачів, скасування підписки, перегляд профілів користувачів.
    - Модуль Google Cloud Storage:
        - Завантаження, збереження та видалення файлів у Google Cloud Storage.

2. Ключові функції:
    - Модуль статей:
        - Створення статті.
        - Видалення статті.
        - Збереження статті до улюблених.
    - Модуль користувачів:
        - Реєстрація користувача.
        - Авторизація користувача.
    - Модуль Google Cloud Storage:
        - Завантаження файлу.
        - Отримання файлу зі сховища.

3.4. Реалізація системи безпеки

1. Методи аутентифікації та авторизації:
    - Використання JWT-токен для аутентифікації.
    - Реалізація авторизації на основі ролей користувачів:
        - Неавторизовані користувачі можуть лише переглядати статті.
        - Зареєстровані користувачі можуть створювати, редагувати, видаляти статті, зберігати статті в улюблені, підписуватись на інших авторів, отримувати сповіщення про нові статті улюблених авторів, скаржитись на статті.
        - Адміністратори мають повний доступ до всіх функцій сайту.
2. Захист даних:
    - Шифрування паролів:
        - Паролі користувачів шифруються за допомогою бібліотеки bcrypt перед збереженням у базі даних.
    - Захист від SQL-ін'єкцій:
        - Використання ORM (Object-Relational Mapping) для взаємодії з базою даних (MongoDB), що зменшує ризик SQL-ін'єкцій.
3. Інші заходи безпеки:
    - Регулярні оновлення залежностей:
        - Залежності регулярно оновлюються для запобігання використанню вразливих бібліотек.
    - Використання SSL/TLS:
        - Захист передаваних даних за допомогою SSL/TLS.
    - Автоматизовані сканери безпеки:
        - Використання інструментів для виявлення та виправлення вразливостей у коді та конфігураціях.

4. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ СЕРВЕРНОЇ ЧАСТИНИ САЙТУ СТУДАБР

4.1. Методи та інструменти тестування

1. Методи тестування:

   Юніт-тестування: Тестування окремих модулів або функцій у відриві від інших частин системи.

   Інтеграційне тестування: Тестування взаємодії між різними модулями та компонентами.

   E2E (end-to-end) тестування: Тестування всієї системи від початку до кінця, щоб перевірити реальний робочий процес.

2. Інструменти тестування:

   Jest: Основний інструмент для юніт та інтеграційного тестування.

   Supertest: Використовується для тестування HTTP запитів до API.

   Postman: Інструмент для тестування API, що дозволяє виконувати ручне тестування та автоматизацію тестування через Postman Collections.

   MongoDB Memory Server: Використовується для створення в пам'яті екземпляра бази даних MongoDB для тестування.

4.2. Результати тестування

1. Юніт-тести:

   Всі основні функції (реєстрація користувачів, створення статей, підписка на користувачів) успішно пройшли юніт-тести.

2. Інтеграційні тести:

   Проведено інтеграційне тестування для перевірки взаємодії між різними модулями (користувачі, статті, підписки).

3. Результати тестування:

   Усі тести пройдено успішно.

   Виявлені та виправлені кілька незначних помилок.

   Всі функції працюють згідно з очікуваннями.

4.3. Впровадження та розгортання серверної частини

1. Підготовка до розгортання:
    - Очищення коду: Видалення непотрібного коду, коментарів та тимчасових файлів.
    - Оновлення документації: Актуалізація документації для розробників та користувачів.
    - Підготовка середовища: Налаштування серверного середовища, включаючи встановлення необхідних залежностей та налаштування бази даних.
2. Процес розгортання:
    - Створення Docker контейнерів.
    - Створення Dockerfile для додатка.
    - Налаштування CI/CD.
    - Використання GitHub Actions для автоматизації процесу деплою.
3. Моніторинг та підтримка:
    - Налаштування інструментів моніторингу, таких як Prometheus та Grafana.
    - Використання логування для відслідковування роботи додатка та виявлення помилок, наприклад, через Winston.
    - Регулярне оновлення та підтримка додатка, включаючи виправлення багів та оновлення залежностей.

ВИСНОВОК

Під час виконання дипломного проекту була реалізована серверна частина для сайту Студабр. Було обрано сучасні технології та інструменти, такі як NestJS та MongoDB, які забезпечують високу продуктивність, гнучкість та масштабованість системи. У цьому розділі будуть розглянуті основні досягнення проекту, виклики, з якими стикнулися під час його виконання, а також перспективи подальшого розвитку.

Вибір технологій:

NestJS було обрано як основний фреймворк для розробки серверної частини завдяки його модульній архітектурі, підтримці TypeScript та інтеграції з іншими сучасними інструментами. NestJS дозволяє будувати структуровані та масштабовані додатки, що є важливим для довготривалої підтримки та розвитку проекту.

MongoDB була обрана як база даних завдяки її гнучкій схемі даних, високій продуктивності та можливості горизонтального масштабування. MongoDB добре підходить для додатків, які працюють з великою кількістю різнорідних даних та потребують швидкого доступу до них.

Google Cloud Storage було обрано для зберігання файлів завдяки його надійності, високій продуктивності та легкій інтеграції з іншими сервісами Google Cloud. Це дозволяє ефективно зберігати та обробляти великі обсяги файлів.

Основні досягнення проекту:

Розробка модулів для управління статтями та користувачами:

Створення та управління статтями, включаючи функції створення, редагування, видалення та отримання статей.

Реалізація можливості підписки на інших користувачів та відстеження їх діяльності.

Зберігання та обробка зображень і файлів у Google Cloud Storage.

Автентифікація та авторизація:

Реєстрація та логін користувачів з використанням bcrypt для хешування паролів та JWT для створення токенів автентифікації.

Захист маршрутів за допомогою охоронців (guards), що забезпечує доступ до ресурсів лише авторизованим користувачам.

Тестування та налагодження:

Автоматизоване тестування з використанням Jest, включаючи тести для контролерів, сервісів та моделей.

Відлагодження коду за допомогою логів, консольних повідомлень та інструментів відладки в IDE.

Моніторинг та логування:

Використання бібліотеки Winston для логування, що дозволяє зберігати логи у файлах або відправляти їх у зовнішні сервіси.

Інтеграція з Prometheus та Grafana для моніторингу продуктивності та збору метрик.

Виклики та їх подолання:

Інтеграція з Google Cloud Storage: Налаштування автентифікації та прав доступу для коректної роботи з Google Cloud Storage вимагало детального вивчення документації та налаштування сервісних акаунтів.

Забезпечення безпеки даних: Хешування паролів та створення токенів автентифікації вимагало ретельного підходу для запобігання можливим вразливостям та забезпечення безпеки користувачів.

Оптимізація продуктивності: Застосування кращих практик для роботи з базою даних MongoDB, включаючи використання індексів та оптимізацію запитів, дозволило значно підвищити продуктивність додатка.

Перспективи розвитку:

Інтеграція з соціальними мережами: Реалізація авторизації через соціальні мережі (Google, Facebook, Twitter), що спростить процес реєстрації та входу для користувачів.

Оптимізація та масштабування: Проведення оптимізації продуктивності додатка та підготовка його до масштабування, включаючи розподілену архітектуру та використання додаткових інструментів для балансування навантаження.

Ці вдосконалення допоможуть зробити сайт Студабр більш зручним, функціональним та привабливим для користувачів, а також забезпечити його довготривалу підтримку та розвиток.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Documentation | NestJS. URL: https://docs.nestjs.com/ (дата доступу: 30.05.2024)
2. Google Cloud Documentation. URL: https://cloud.google.com/docs (дата доступу: 02.06.2024)
3. MongoDB Support Portal. URL: https://support.mongodb.com/?_ga=2.54296941.430311340.1717677399-1737728074.1713440476 (дата доступу: 21.05.2024)
4. Node.js Documentation. URL: https://nodejs.org/docs/latest/api/ (дата доступу: 26.05.2024)
5. Інструкція Git для новачків: що це таке, як він працює та які є основні команди. URL: https://dan-it.com.ua/uk/blog/instrukcija-git-dlja-novachkiv-shho-ce-take-jak-vin-pracjuie-ta-jaki-ie-osnovni-komandi/ (дата доступу: 23.03.2024)
6. Git - Book. URL: https://www.git-scm.com/book/uk/v2 (дата доступу: 04.05.2024)
7. About GitHub and Git - GitHub Docs. URL: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git (дата доступу: 23.03.2024)
8. TypeScript: JavaScript With Syntax For Types. URL: https://www.typescriptlang.org/ (дата доступу: 10.04.2024)
9. Що таке rest api: основні принципи та практики. URL: https://foxminded.ua/shcho-take-rest-api/ (дата доступу: 20.01.2024)
10. Sairyss/backend-best-practices. URL: https://github.com/Sairyss/backend-best-practices (дата доступу: 10.02.2024)
11. Key Principles and Best Practices for Effective Backend Development. URL: https://www.linkedin.com/pulse/key-principles-best-practices-effective-backend-srilekha-senthilkumar/ (дата доступу: 04.03.2024)
12. The 5 SOLID principles of object-oriented design explained. URL: https://www.techtarget.com/searchapparchitecture/feature/An-intro-to-the-5-SOLID-principles-of-object-oriented-design (дата доступу: 25.04.2024)
13. Алгоритми та структури даних — від «десь чув» до «ефективно застосовую». URL: https://dou.ua/forums/topic/40645/ (дата доступу: 20.11.2023)

14. АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ. URL: https://ela.kpi.ua/server/api/core/bitstreams/0db974f9-16fa-459c-9f19-fab0021222ed/content (дата доступу: 27.03.2024)
15. How to Design a Backend Architecture for Your Web Application. URL: https://medium.com/@queenskisivuli/how-to-design-a-backend-architecture-for-your-web-application-2b7188a497ab (дата доступу: 13.01.2024)
16. Best NestJS Practices and Advanced Techniques. URL: https://dev.to/drbenzene/best-nestjs-practices-and-advanced-techniques-9m0 (дата доступу: 20.01.2024)
17. Testing | NestJS - A progressive Node.js framework. URL: https://docs.nestjs.com/fundamentals/testing (дата доступу: 20.05.2024)
18. How to write a NestJS Dockerfile optimized for production. URL: https://www.tomray.dev/nestjs-docker-production (дата доступу: 13.05.2024)
19. 7 етапів розробки сайту: як правильно розробити сайт. URL: https://brander.ua/blog/kh-etapiv-rozrobki (дата доступу: 07.03.2024)
20. JSON Web Token. URL: https://uk.wikipedia.org/wiki/JSON_Web_Token (дата доступу: 04.01.2024)
21. Encryption and Hashing | NestJS. URL: https://docs.nestjs.com/security/encryption-and-hashing (дата доступу: 09.03.2024)