ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка застосунку для ознайомлення зі слов'янською міфологією використанням JS, Angular, NestJs»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Андрій ПАНАСЮК
(підпис)

Виконав: здобувач вищої освіти групи ПД-44

Андрій ПАНАСЮК

Керівник: Тимур ДОВЖЕНКО
к.т.н.

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Панасюку Андрію Сергійовичу

1. Тема кваліфікаційної роботи: «Розробка застосунку для ознайомлення зі слов'янською міфологією використанням JS, Angular, NestJs»
керівник кваліфікаційної роботи к.т.н., доцент кафедри ІПЗ Тимур ДОВЖЕНКО, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про слов'янську міфологію, опис інструментів для створення інтерактивних елементів веб-додатку та бази даних для зберігання та обробки міфологічної інформації.

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

1. Аналіз слов'янської міфології та існуючих веб-додатків, які ознайомлюють з цією темою.
2. Проектування веб-додатку для інтерактивного вивчення слов'янської міфології з використанням фреймворків JS, Angular і NestJs.
3. Програмна реалізація та опис функціонування веб-додатку.
4. Тестування функціональності веб-додатку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Програмні засоби реалізації.
3. Діаграма використання.
4. Діаграма архітектури застосунку.
5. Схема бази даних.
6. Діаграма класів Article.
7. Діаграма класів модулів.
8. Екранні форми.
9. Апробація результатів дослідження.

6. Дата видачі завдання«28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Дослідження програмних засобів та фреймворків.	14.03-18.03.2024	
4	Проектування архітектури системи та бази даних.	19.03-29.03.2024	
5	Розробка функціоналу додатку. Створення інтерфейсу користувача	30.03-15.04.2024	
6	Тестування функціональності та користувацького інтерфейсу	16.04-26.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	27.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Андрій ПАНАСЮК

(підпис)

Керівник

кваліфікаційної роботи _____ Тимур ДОВЖЕНКО

(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 59 стор., 1 табл., 38 рис., 16 джерел.

Мета роботи – розширення засобів інформування про слов'янську міфологію через веб-застосунок.

Об'єкт дослідження – процес ознайомлення зі слов'янською міфологією.

Предмет дослідження – веб-застосунок для ознайомлення зі міфологією України.

Короткий зміст роботи: В роботі проаналізовано та програми засоби для візуалізації інформації про слов'янську міфологію в контексті задачі створення освітнього веб-додатку. Розроблено веб-додаток та програмно реалізовані ключові функціональні можливості, зокрема: внесення даних та збереження їх у базі даних системи, їх зміна, реєстрація користувачів, зміна ролей та доступу до функціоналу системи, перегляд статей та елементів місяцеліку, фільтрація контенту за категорією. Проведено тестування функціоналу додатку, перевірено роботу форм користувацького інтерфейсу. В роботі використано фреймворки NestJS для розробки серверної частини додатку, Angular для побудови користувацького інтерфейсу, базу даних PostgreSQL для зберігання інформації, бібліотеку ORM Sequelize, Postman для тестування запитів.

КЛЮЧОВІ СЛОВА: JavaScript, Angular, NestJS, PostgreSQL, TypeScript, Sequelize, CMS, Postman, додаток, веб-сайт.

ЗМІСТ

ВСТУП 11

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 12

1.1 Актуальність теми 12

1.2 Аналіз існуючих платформ 13

1.2.1 Аналіз платформи: Bestiary.us 13

1.2.2 Аналіз платформи: oldworldgods.com 15

1.2.3 Аналіз платформи: The Medieval Bestiary 16

1.3 Вибір засобів програмної реалізації додатку 19

1.3.1 Вибір мови програмування 20

1.3.2 Вибір фреймворків 22

1.4 Вибір бази даних 23

2 ПРОЕКТУВАННЯ СИСТЕМИ 25

2.1 Проектування структури системи 25

2.2 Визначення вимог до системи 29

2.3 Проектування структури бази даних 30

2.4 Проектування структури серверної частини додатку 37

2.5 Проектування структури клієнтської частини додатку 41

3 ОПИС РЕАЛІЗАЦІЇ СИСТЕМИ 46

3.1 Діаграма використання 46

3.2 Діаграма класів 48

4 ТЕСТУВАННЯ ПРОГРАМИ 52

ВИСНОВКИ 57

ПЕРЕЛІК ПОСИЛАНЬ 58

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 60

ДОДАТОК Б. ЛІСТИНГ КЛАСІВ МОДУЛІВ 68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API - Application Programming Interface

CSS - Cascading Style Sheets

HTML - Hyper Text Markup Language

MVC - Model View Controller

DOM - Document Oriented Model

HTTP - Hyper Text Transfer Protocol

HTTPS - Hyper Text Transfer Protocol Secure

SQL – Structured Query Language

БД – База даних

CMS - Content Management System

DI – Dependency Injection

ORM – Object-Relational Mapping

DTO - Data Transfer Object

JWT - JSON Web Token

UI - User Interface

ВСТУП

Однією з таких цікавих галузей є вивчення міфологічних систем різних народів, які відображають їхню світоглядну та культурну спадщину. У цьому контексті особливий інтерес представляє слов'янська міфологія, що містить в собі велику кількість унікальних образів, легенд та вірувань, які продовжують жити серед свідомості людей вже століттями.

Мета даного дипломного проекту полягає у створенні веб-додатку, який надасть користувачам можливість ознайомлення з основними аспектами слов'янської міфології. Додаток буде використовувати сучасні технології програмування, зокрема JavaScript, Angular та NestJs, що дозволить створити зручне та інтерактивне середовище для дослідження цієї цікавої галузі культурної спадщини.

У процесі розробки програмного забезпечення будуть використані найновіші підходи до створення користувацького інтерфейсу, забезпечуючи відмінну візуалізацію інформації про міфологічних персонажів, богів та звичаїв слов'янської культури. Такий підхід дозволить зробити процес дослідження цієї теми цікавішим та зрозумілішим.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність теми

Тематика ознайомлення зі слов'янською міфологією, в контексті глобалізації, не може бути недооцінена. Протягом останніх десятків років, процес глобалізації все більше розмиває культурні кордони між країнами. Через це з часом можлива втрата самобутності та ідентичності традицій народу. Перед суспільством стоять виклики в культурні та історичній сфері.

Одним з методів збереження культурної спадщини та традицій суспільства являється тематика слов’янської міфології. Вивчення міфів сприяє збереженню та розумінню культурної спадщини народу. Вона допомагає краще зрозуміти традиції та звичаї місцевого населення, розкриває унікальні аспекти їх світогляду, вірувань та менталітету. Крім того – це чудове джерело натхнення для сучасного мистецтва, літератури та кінематографу. Образи та мотиви можуть бути використані для створення нових творів, які розкриють унікальність української культури.

Елементами міфології українського народу являється: бестіарій міфічних створінь, пантеон язичницьких богів, обряди та оповіді, місяцелік тощо.

Місяцеліком вважають народним календарем, який містить багато обрядів та вірувань, спостережень природніх та суспільних явищ по днях, деталі про свята та їх значення. Хоч із плином часу та зі зміною укладу життя, значення Місяцеліка втратилося, але деякі його елементи збереглися у фольклорі та народних традиціях України.

Подальше дослідження і розробка веб-сайту з використанням фреймворку Angular та NestJS, є кроком для полегшення розуміння тематики. Використання сучасних технологій веб-розробки дозволить створити зрозумілий та функціональний веб-сайт, який забезпечить необхідну інформацію користувачам.

1.2 Аналіз існуючих платформ

Розвиток інтернету значно допоміг, та полегшив поширення інформації в людському суспільстві. Завдяки цьому для вивчення міфології різних народів, включаючи слов'янську. Зокрема, існує велика кількість веб-сторінок, присвячених міфології різних країн світу, які пропонують ресурси та можливості для кращого вивчення цієї тематики.

Вони представляють собою збірки інформації, від легенд та міфів до аналізу символіки та інтерпретації різних аспектів віровчень. Багато веб-ресурсів пропонують також можливість обговорення та обміну думками з іншими зацікавленими особами, що створює співтовариство. Вони створюють сприятливе середовище для спілкування фахівців, студентів, аматорів та просто зацікавлених осіб, щоб обговорювати та ділитися своїми дослідженнями, враженнями тощо.

1.2.1 Аналіз платформи: Bestiary.us

Bestiary.us - це онлайн-ресурс, який присвячений вивченню міфології та фольклору різних культур світу. Сайт має велику базу даних з інформацією про різноманітні аспекти міфології, включаючи слов'янську. На Bestiary.us можна знайти детальні описи різних персонажів, богів, створінь та історій з міфології різних народів світу.

Сайт також містить розділи, присвячені різним аспектам слов'янської міфології, таким як символіка, релігійні переконання та ритуали. Ці розділи допомагають краще зрозуміти контекст, в якому існують міфічні персонажі, а також поглибити знання про слов'янську культуру та історію.

Присутня можливість реєструватись та вносити нові дані до вже існуючого матеріалу, чи створювати свої власні статті. Є можливість коментування, якщо користувач зареєстрований та авторизований у системі додатку. Інформацій постійно оновлюється адміністраторами ресурсу, так зареєстрованими

користувачами. Це робить ресурс більш інтерактивним та актуальним в інформаційному просторі даної тематики.

Тут можна знайти теми про різних персонажів міфології, тлумачення символіки, порівняння з іншими міфологіями, а також обговорення сучасних тенденцій у вивченні міфології.

Рис 1.1 Головна сторінка bestiary.us

1.2.2 Аналіз платформи: oldworldgods.com

OldWorldsGods - це веб-сайт, який відданий вивченню та популяризації історій та легенд про богів різних народів світу. Цей ресурс надає користувачам унікальну можливість дізнатися більше про різноманітні аспекти пантеонів богів через розмаїття статей, ілюстрацій, аналітичних матеріалів та інших ресурсів. Кожна стаття містить детальну інформацію з посиланнями на їх першоджерело. На рисунку 1.2 можна побачити приклад однієї зі статей.

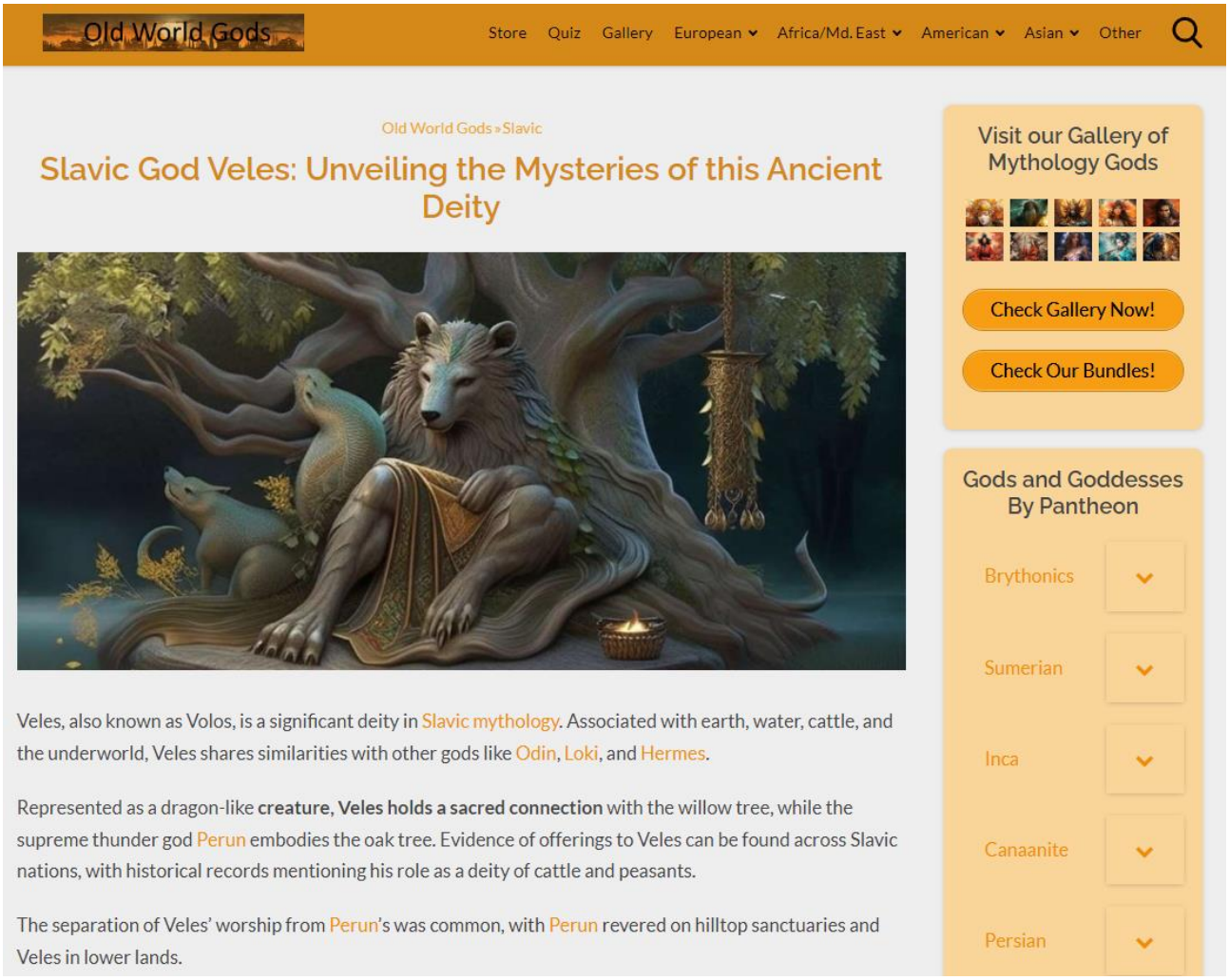


Рис. 1.2 Приклад статті

Головна сторінка сайту містить зручні посилання на пантеони богів різних національностей, що дозволяє користувачам швидко та зручно знайти інформацію, яка їх цікавить.

Одним із цікавих функціональних елементів сайту є тести які можна побачити на рисунку 1.3, які дозволяють користувачам перевірити свої знання про богів конкретної міфології. Ці тести можуть містити різну кількість та типів питань.

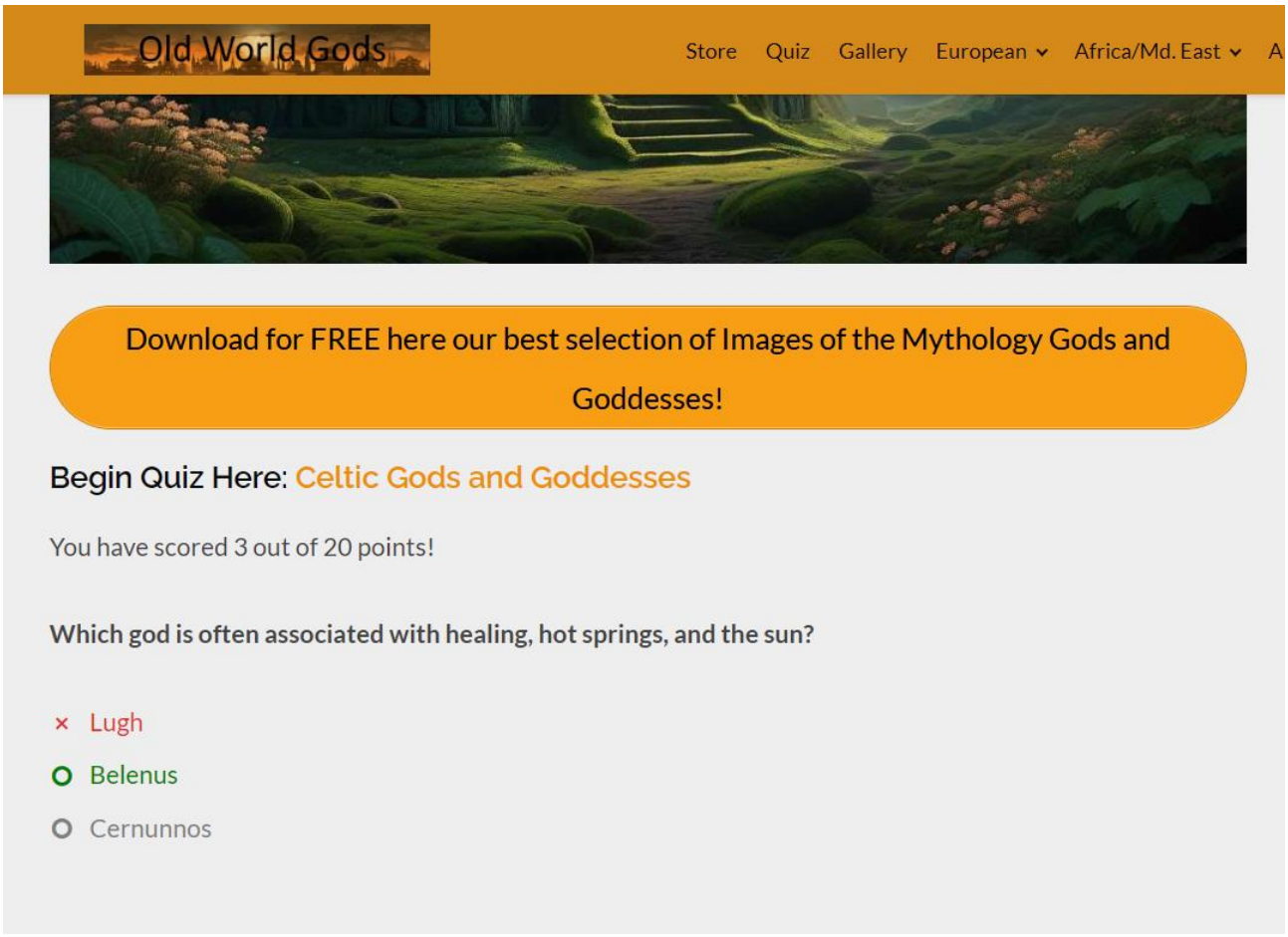


Рис. 1.3 Приклад тесту OldWorldsGods

На сайті також присутній магазин де можна придбати ексклюзивний пакет ілюстрацій якій постійно оновлюється художниками ресурсу.

1.2.3 Аналіз платформи: The Medieval Bestiary

The Medieval Bestiary – це онлайн ресурс для ознайомлення та вивчення свідчень про реальних та фантастичних тварин періоду Середньовіччя.

Сайт має велику базу знань про реальних та фантастичних тварин періоду Середньовіччя. Серед ресурсів веб-сайту є такі елементи як:

- Beasts: Цей розділ дозволяє дізнатися більше про тварин за індексом, категорією або типом. Детальніше можна побачити на рисунку 1.4.
- Manuscripts: Сайт посилається на багато середньовічних рукописів, переважно бестіарії та авіарії, а також інших, які включають повні або часткові бестіарії або містять зображення або текст. Рукописи поділяються на кілька типів, включаючи бестіарії, авіарії, енциклопедії, псалтирі.
- Bibliography: Кожен елемент у розділах Beasts, Manuscript та Encyclopedia має власну сторінку бібліографії з книгами та статтями, які стосуються тварини, рукопису або статті енциклопедії.
- Encyclopedia: Ця енциклопедія - це низька серія коротких статей на теми, пов'язані з бестіарієм та "тваринами в Середні віки" взагалі. Багато статей присвячені середньовічним людям, які якимось чином внесли вклад у середньовічну тваринну літературу; більшість з них - це автори бестіаріїв та енциклопедій та інші текстів

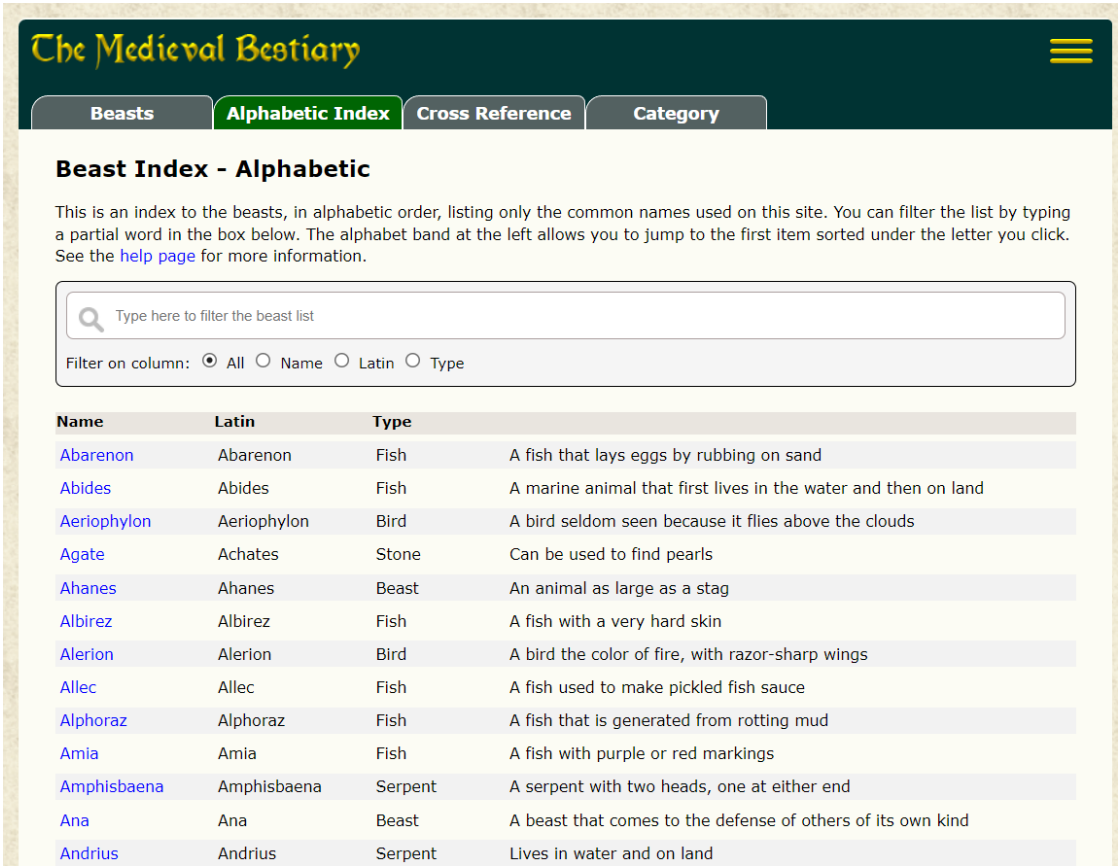


Рис. 1.4 Сторінка зі статтями

Таблиця 1.1

Порівняння існуючих аналогів

Особливості	**The Medieval Bestiary**	**OldWorldsGods**	**Bestiary.us**	**Власний додаток Velescape**
Розгорнута інформація	+	+	+	+
Фільтрація за назвою та категорією	-	+	+	+
Можливість коментувати	-	-	+	+
Інтерактивні тести	-	+	-	+
Місяцелік	-	-	-	+
Відповідність дизайну сучасним вимогам	-	+	-	+
Реєстрація та Авторизація	-	-	+	+
Наявність особистого кабінету	-	-	+	+

1.3 Вибір засобів програмної реалізації додатку

В залежності від заданих технічних задач які ставляться при розробці веб-сайту, можна обрати кілька шляхів його побудови. Найбільш поширеними методами являються: Веб-конструктори, CMS та “ручне” написання сайту на мовах програмування з використанням фреймворків. Кожен з цих варіантів має свої шляхи реалізації, переваги та недоліки. Розглянемо їх детальніше, щоб зрозуміти який метод розробки найкраще підходить.

Що таке Веб-конструктори? Це спеціальні онлайн-системи які дозволяють створити працюючий веб-сайт зі вже готових модулів та компонентів. Головною перевагою вважається простота у використанні, можна використовувати вже готові шаблони які надаються платформами. Також існує можливість створити свій власний шаблон, багато веб-конструкторів мають власні навчальні посібники та туторіали які допоможуть користувачеві створити все з “нуля”. Нажаль конструктори мають певні технічні обмеження і не можуть вирішити всі поставленні задачі та вимоги. Це обмежена гнучкість платформи, модулі та компоненти не завжди можуть реалізувати складний функціонал. Спроби скомпонувати модулі часто призводять до незадовільного та непередбачуваного результату, оскільки платформа може не підтримувати необхідні функції. Масштабованість, оновлення чи міграція проекту складний, а часто не можливий, бо модулі та функції є частиною платформи конструктора.

CMS (Content Management System) – це тип програмного забезпечення, який надає користувачам можливість створювати, редагувати, видаляти вміст веб-сайту без особливих технічних знань програмування. Ці системи покращили розробку веб-сайтів, стали більш доступнішими для людей, які не мають часу чи бажання вчитись програмуванню. CSM також має систему модулів та компонентів, але на відміну від конструкторів, їх можна переробити під потреби поставлених задач. Це надає більшої “гнучкості” при розробці, оскільки існує підтримка значного спектру функцій, плагінів та шаблонів. Цим значно користуються відомі компанії та

магазини. Такі сайти як TED Blog, Microsoft News, MTV News, Vogue побудовані на CMS – WordPress, що є найбільшим постачальником послуг CMS. Одним зі значних мінусів CMS являється низький рівень безпеки щодо хакерських атак та достатньо високий “learning curve” при побудові складного, ефективного веб-сайту.

Фреймворки веб-розробки - це набір інструментів, бібліотек та структурних шаблонів, які допомагають розробникам створювати веб-сайти та додатки ефективніше та швидше. Вони забезпечують базовий функціонал, який можна використовувати для створення різноманітних веб-рішень без необхідності писати код з нуля. Однією з основних переваг використання фреймворків є швидкість розробки. Замість того, щоб витрачати час на написання кожного елементу сайту вручну, розробник може скористатися готовими компонентами та функціями, які надає фреймворк. Це дозволяє значно прискорити процес створення веб-сайту та зосередитися на більш складних аспектах розробки. Ще однією важливою перевагою є стандартизація розробки. Фреймворки надають структуру та правила, які допомагають розробникам писати чистий та організований код. Це робить проект більш доступним для інших розробників, які можуть швидше зрозуміти структуру проекту та долучитися до розробки. Крім того, фреймворки часто надають різноманітні інструменти для підтримки безпеки, оптимізації швидкості завантаження сторінок, роботи з базами даних та іншими важливими аспектами веб-розробки.

1.3.1 Вибір мови програмування

Була обрана мова програмування JavaScript. JavaScript – це "високорівнева" мова програмування з динамічною типізацією, яка в основному використовується для написання скриптів. У сфері веб-розробки HTML відіграє роль основної "структури" веб-сайту, в той час як CSS відповідає за його стилізацію. Мова JavaScript додає інтерактивність та функціональність до сайту, роблячи його більш привабливим для користувачів. Використання цієї мови надає змогу розробнику створювати анімації, реагувати на дії користувачів та взаємодіяти з сервером без

перезавантаження сторінки. Однією з основних переваг JavaScript є те, що вона виконується безпосередньо в браузері користувача, що робить цю швидкою та ефективною. JavaScript також підтримує асинхронне програмування, що дозволяє виконувати декілька операцій паралельно одна до одної. Це зберігає час та ресурси, оскільки програма може продовжувати виконання інших завдань під час очікування завершення асинхронних операцій, таких як мережеві запити або робота з базою даних. Для цього використовуються такі методи як: call-back, promise та async/await. Асинхронний підхід у JavaScript дозволяє створювати відзивчиві та ефективні веб-додатки, які можуть взаємодіяти з користувачем та зовнішніми ресурсами швидко та ефективно.

Динамічна типізація в JavaScript - це властивість мови, що дозволяє змінювати типи даних змінних під час виконання програми. Це означає що під час оголошення змінної, вони не пов’язані з конкретним типом даних, тому такий тим може змінюватись залежно від значення, яке зберігається у змінній. Це може бути зручно для розробників, оскільки надає гнучкість під час роботи з даними. Але в інших може привести до неочікуваних помилок під час виконання програми, які складно виявити під час етапу компіляції коду. Також це знижує продуктивність не тільки програми, але й розробників, бо витрачається дорогоцінний ресурс на аналіз коду.

Проблему динамічної типізації вирішує розширення TypeScript. Це розширення додає статичну типізацію мови, яка дозволяє визначати тип даних для змінних, параметрів та об’єктів під час розробки. TypeScript трансформує та значуще покращує JavaScript, якщо раніше це було приємним розширенням до вже існуючого функціоналу, то зараз все більше компаній і розробників ставлять TypeScript за стандарт.

1.3.2 Вибір фреймворків

Angular – це фреймворк JavaScript з відкритим кодом, що використовується створення інтерфейсу користувача (UI). Цей фреймворк використовується для створення динамічних, інтерактивних та масштабованих веб-додатків.

На відміну від VueJS та ReactJS, які в основному є бібліотеками, Angular є повноцінним веб-фреймворком. Це означає, що він надає ширший набір вбудованих інструментів для побудови складних додатків, без необхідності завантажувати додаткові бібліотеки. Порівнюючи з Vue, який відзначається простішим синтаксисом та швидшою кривою вивчення, Angular пропонує розширений набір інструментів та більшу екосистему, що може бути корисним для управління проектами великого масштабу. ReactJS у свою чергу працює як бібліотека JavaScript. Хоча React популярний завдяки своїй функції віртуального DOM, яка підвищує продуктивність додатків, Angular використовує звичайний DOM разом з TypeScript для кращої читабельності та підтримки коду.

Angular використовує модульну систему організації, яка базується на різних функціональних частинах такі як: компонент, сервіс, модуль. За допомогою модульної системи, розробники можуть розділити додаток на невеликі функціональні блоки які можна повторно використовувати протягом всієї роботи над додатком. Завдяки цьому модулі можна повторно використовувати в різних частинах додатку, або навіть у кількох проєктах, що призводить до створення кодових баз. Angular має вбудований механізм **Dependency injection (DI),** він дозволяє визначати як код взаємодіє в компонентах модулю. Це дозволяє розробникам конфігурувати залежності, використовувати їх в інших модулях, робити модулі більш гнучкими для використання, полегшує тестування компоненту.

NestJS - це фреймворк JavaScript, що використовується для розробки серверних додатків на основі NodeJS. Цей фреймворк базується на TypeScript, і пропонує широкий набір інструментів та функцій, які полегшують розробку масштабованих та надійних серверних додатків. NestJS бере з Angular концепцію

модульності, де кожен модуль є самостійною одиницею, що дозволяє структурувати додаток на невеликі функціональні блоки. Також, NestJS використовує деякі інші концепції Angular, такі як Dependency Injection, що полегшує керування залежностями та робить додаток більш гнучким і підтримуваним. Крім того, NestJS має схожі з Angular деякі підходи до роботи з HTTP запитами та обробки подій.

1.4 Вибір бази даних

PostgreSQL - це реляційна база даних, що пропонує широкий спектр функцій та високу надійність. Реляційна база даних представляє себе у вигляді організованих таблиць. Кожна така таблиця має рядки та стовпці, де рядок представлений записом даних, а стовпець в цій таблиці відповідальний за властивість та тип атрибуту. Реляційна база даних використовує зв’язки між таблицями, щоб установити зв’язки між даними. Зв’язок між таблицями в реляційних базах даних відбувається за рахунок первинних та зовнішніх ключів таблиці. Первинний ключ визначає унікальний ідентифікатор запису, а зовнішній ключ встановлює зв’язок між таблицями. Це спеціальні поля, які посилаються на унікальний ключ в іншій таблиці, ці ключі мають бути однаковими. PostgreSQL є варіацією стандарту мови запитів SQL з більшими можливостями, такими як розширена система типів, успадкування, функції та правила виробництва. Ця система керування базою даних (СКБД) відповідає стандартам ACID. Завдяки цьому PostgreSQL пропонує має багато вбудованих типів даних, таких як: JSON, XML, HSTORE, PostGIS, IPv6; гнучкі індекси, включаючи композитні індекси, GiST, SP-GiST, GIN; повнотекстовий пошук, онлайн реорганізацію індексів; фонові процеси, такі як керований процес під назвою Mongress, який приймає запити MongoDB для взаємодії з даними Postgres; інтерфейс модулів участі: pgcrypto, HSTORE (дані без схеми); та обширну підтримку SQL.

Створення SQL-запитів може бути вимогливим завданням через його складність та часомісткість. Необхідність у глибокому розумінні SQL та структури бази даних стає ключовою для успішного складання запитів, особливо у випадку баз даних зі складною структурою та великою кількістю пов'язаних таблиць. Недостатній досвід або знання можуть ускладнити написання складних запитів та роблять процес розробки довгим.

Ці проблеми можна вирішити за допомогою бібліотек ORM, які спрощують процес взаємодії з базою даних.

ORM (Object-Relational Mapping) бібліотеки — це інструменти, які дозволяють розробникам працювати з базами даних, використовуючи об'єктно-орієнтований підхід. Вони дозволяють взаємодіяти з базою даних через об'єкти, що спрощує роботу з даними і забезпечує простіший спосіб взаємодії з базою даних для розробників. Вони автоматично перетворюють об'єкти програми в реляційні структури даних, що дозволяє використовувати підходи ООП без використання SQL-запитів. Ці бібліотеки також забезпечують відслідковування змін об'єктів, автоматичне вирішення конфліктів при збереженні даних та інші зручні функції, що полегшують роботу з базами даних. Однією з популярних ORM бібліотек є Sequelize.

Sequelize - це сучасна бібліотека для TypeScript і Node.js, яка дозволяє взаємодіяти з реляційними базами даних, такими як Oracle, Postgres, MySQL, SQLite та SQL Server. Вона надійно підтримує транзакції, асоціації та ліниве завантаження даних, а також розподіляти навантаження підвищуючи продуктивність.

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Проектування структури системи

Створення архітектури програми являється одним з найважливішим етапів розробки. Саме тут визначається, які взаємозв’язки компонентів системи треба створити, які фреймворки та бібліотеки підходять для розробки продукту. Додаток складається з клієнтської, серверної частини, бази даних та використовує архітектурний патерн MVC, що розділяє логіку програми на частини. Це допомагає розробникам працювати над кожним компонентом системи незалежно одне від одного. Такими компонентами являються:

- Model. Де зберігається логіка даних та сама інформація, яка зберігається додатком. Саме цей компонент визначає, які дані будуть відображатись користувачеві, та як вони будуть оброблятись програмою. Це може бути проста структура даних, така як масив або словник. Зазвичай модель зберігає інформацію про те, що сталося в одному або декількох подіях, таких як введення користувача або запити до бази даних.
- View. Це компонент який відображає інтерфейс користувача. Він створюється на основі даних з моделі, щоб створити за допомогою мови розмітки HTML та стилів CSS, веб сторінку.
- Controller. Завдання цього компоненту, забезпечити зв’язок між Model та View, визначити яку модель даних викликати, розпочати роботу відповідної функції на основі викликаних раніше даних, та передати для подальшого відображення. Контролер відповідає за обробку запитів клієнта та взаємодію елементів додатку. Саме тут обробляється «бізнес логіка» додатку.

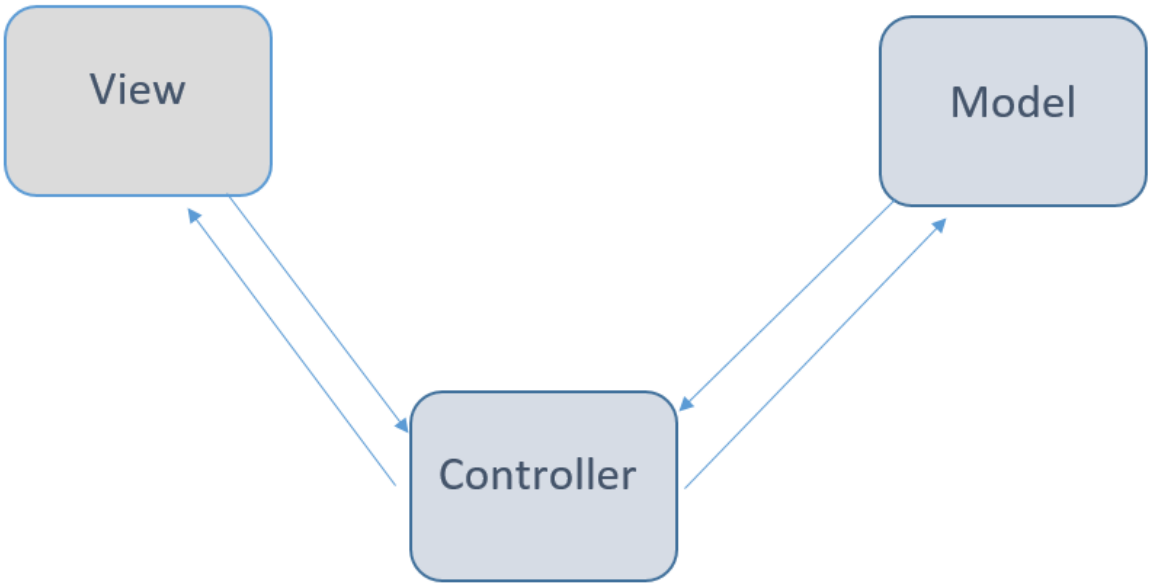


Рис.2.1.1 Схема MVC.

Архітектура MVC широко застосовується в програмуванні. Однак цей спосіб схематики проєкту має свої переваги та недоліки.

Серед переваг варто відзначити простоту в плануванні, обслуговуванні та модифікації додатку. За допомогою MVC розробнику легше відокремити логіку обробки даних від користувацького інтерфейсу, що дозволяє швидше оновити функції веб-сайту, або вносити зміни до UI без значних змін в програмі. Крім того, використання MVC полегшує розподіл завдань між розробниками, оскільки кожен може працювати над окремою частиною без втручання інших. Такий підхід сприяє покращенню якості програмного забезпечення.

Взаємодія між серверною та клієнтською чистинами додатку відбувається за принципом REST API та протокол запиту HTTP.

REST API – це абревіатура від Representational State Transfer Application Programming Interface, методологія за якою різні частини веб-додатків взаємодіють між собою. REST API слугує мостом між двома частинами додатку. Він дає змогу клієнту надсилати запити до серверу і отримувати дані звідти. Цей принцип був сформульований у 2000 році, Роєм Філдінгом і став чудовою заміною старішим

методам машинного спілкування. Вони все ще залишаються золотим стандартом для публічних API.

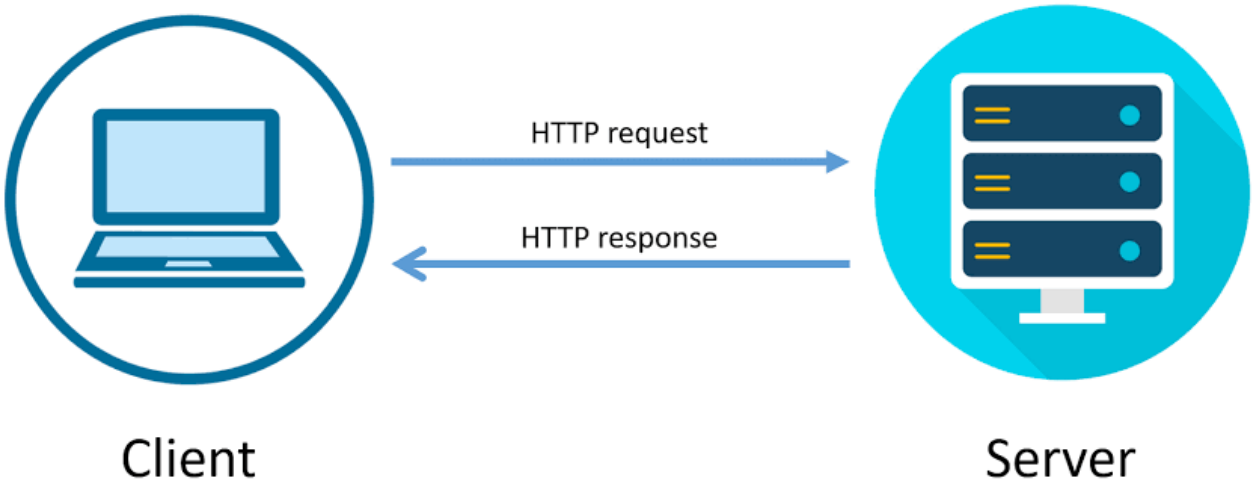


Рис.2.1.2 Схема взаємодії HTTP запитів

Протокол HTTP – використовується для передачі гіпертекстових даних через мережу. HTTP може передавати не лише текстові дані та сторінки HTML, але й зображення, аудіо- та відеофайли. Він також використовується для взаємодії з веб-сервісами, API та іншими програмами в Інтернеті.

Протокол був розроблений на основі протоколів TCP/IP, для зв'язку між веб-браузерами і серверами. Розглянемо які є типи запитів HTTP протоколу.

- GET: використовується для запиту певних даних з сервера.
- HEAD: аналогічний GET, але без тіла відповіді, використовується для отримання метаданих ресурсу.
- POST: створення та додавання нових даних або повідомлень до сервера.
- PUT: використовується для заміни або оновлення існуючого ресурсу на сервері. У випадку форм, оновлюються всі поля.
- DELETE: видалення ресурсу на сервері.
- TRACE: відстеження змін, що вносяться до веб-ресурсу.
- PATCH: для часткової зміни ресурсу на сервері.

- OPTIONS: йде запит на отримання списку доступних HTTP-методів для конкретного ресурсу.
- CONNECT: використовується для перетворення з'єднання запиту в прозорий TCP/IP тунель.

HTTP не зберігає жодної інформації, а лише передає її між клієнтом та сервером. Проте, безпека з'єднання через HTTP не надійна, оскільки дані, які передаються, не шифруються, це може призвести до перехоплення і зловживання даними. Для забезпечення безпеки передачі даних був розроблений протокол HTTPS, який використовує шифрування для захисту інформації, що передається між клієнтом і сервером. HTTPS використовує протокол SSL/TLS версії 1.3, для шифрування даних, що пересилаються, та забезпечує відправників і отримувачів відомостями про те, що їх з'єднання безпечне. На рисунку 2.2 показана спрощена схема шифрування HTTPS. Застосування HTTPS дозволяє забезпечити конфіденційність, цілісність та автентичність даних, що передаються через мережу, що робить його надійним варіантом для захисту веб-комунікацій.

Рис.2.1.3 Схема HTTPS з'єднання через SSL сертифікат

2.2 Визначення вимог до системи

Під час аналізу існуючих рішень в цій галузі були виявлені недоліки, що послужили основою для формулювання вимог до розробки цього веб-додатку. Однією з ключових вимог є забезпечення швидкого збереження, оновлення та відображення даних на веб-сторінці.

У рамках функціональних вимог, основною задачею є реєстрація користувачів і надання функціоналу "особистого кабінету", де користувач може оновлювати свої дані, такі як ім'я, електронна пошта і пароль. Можливість з легкістю переглядати, шукати та фільтрувати контент, пов'язаний із різними аспектами міфології. Додатково, додаток повинен включати засоби для коментування статтей, а також адміністративні інструменти для управління контентом та користувачами.

Також, адміністратор повинен мати можливість блокувати та розблокувати користувачів у випадках, коли вони порушують правила користування ресурсом. Блокування користувачів важливе для підтримання здорової атмосфери на платформі, забезпечення безпеки інших користувачів та відповідності веб-додатку законодавчим нормам щодо контенту.

Що стосується нефункціональних вимог, ключовими є продуктивність та швидкість роботи сайту, які забезпечуватимуть комфортне користування. Особисті паролі користувачів мають бути зашифровані за допомогою шифрувальних бібліотек.

2.3 Проектування структури бази даних

Для повноцінної роботи додатку необхідно створити базу даних, використовуючи PostgreSQL. Вона буде зберігати та організовувати дані веб-сайту.

База даних складається з семи реляційних таблиць, такі як: articles, calendar, comments, feedback, roles, users, user_roles.

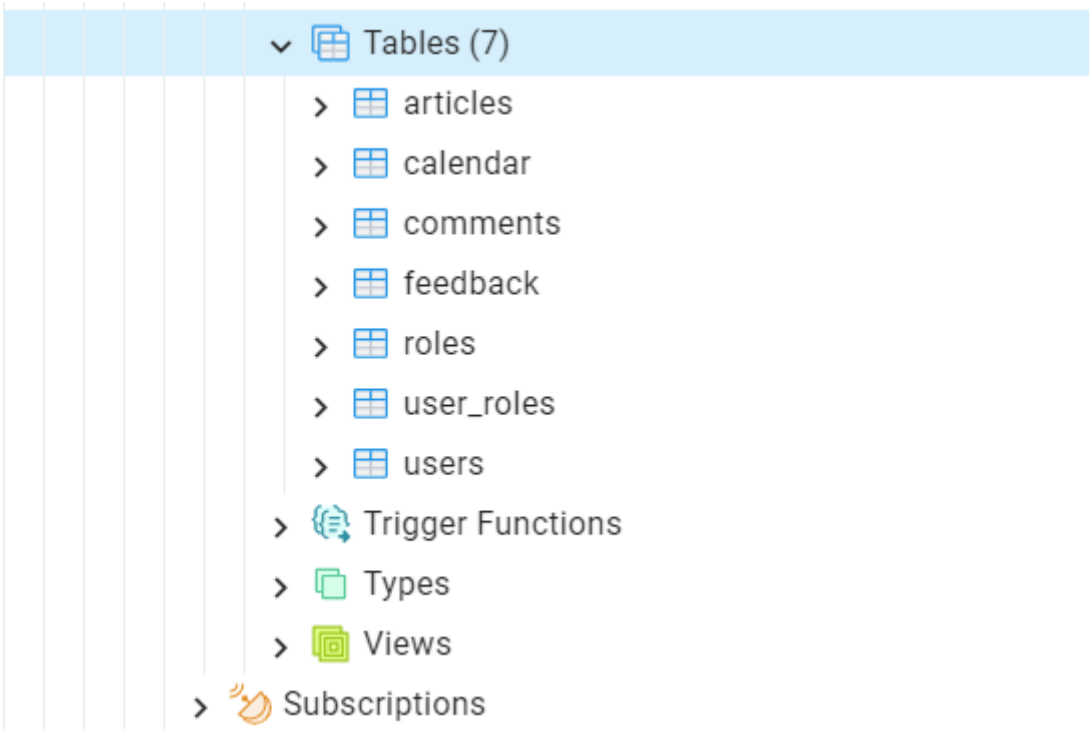


Рис.2.3.1 Структура таблиць PostgreSQL

Таблиця "articles" містить 15 колонок для зберігання різноманітної інформації. Деякі з цих колонок можуть містити значення NULL, тоді як інші, наприклад, "id", "title", "content", "category", обов'язкові для заповнення.

Розглянемо що кожний стовпець означає:

- ID: Це унікальний номер, що ідентифікує кожен запис у таблиці. Значення ID автоматично збільшується при додаванні нового запису за допомогою функції autoincrement, що дозволяє автоматично додавати елементи в цей рядок. Значення цього стовпця не може бути NULL.

- Title: Тут міститься назва статті, що є ключовою інформацією, оскільки вона відображається на веб-сторінці додатку для користувача. Значення цього стовпця не може бути NULL.
- Content: Текстове інформаційне наповнення статті.
- Image: Методика зберігати зображення в базі даних не є ефективною та безпечною. Вони займають багато пам’яті системи та ускладнюють резервне копіювання в разі проблем. Спеціалісти рекомендують зберігати зображення на файловій системі сервера, а в базі даних зберігати лише посилання на ці файли, для зручного управління та відображення. Саме в цьому стовпці зберігається посилання на шлях до файлової системи.
- Category: Тут зберігається інформація про категорія статті. Наприклад, вони можуть бути розподілені на: "Боги", "Створіння", "Духи", "Міфи" тощо, щоб користувачі могли легко знаходити ці статті за тематикою. Може мати значення NULL.
- Temper: Ця колонка відноситься до частини «бестіарій» і може бути відсутньою в інших статтях. Тут зберігається “вдача” створіння, наприклад: “Неоднозначний, бешкетник”, “Добрий, злий”. Може мати NULL значення.
- Location: Ця колонка відноситься до частини «бестіарій» і може бути відсутньою в інших статтях. Зберігаються дані про локацію чи регіон проживання, існування. Приклад: «Живе у непрохідних лісових хащях, у власній хаті укритій золотим мохом». Може мати значення NULL.
- Appointment: Ця колонка відноситься до частини «бестіарій» і може бути відсутньою в інших статтях. Зберігаються дані, що саме дух чи створіння робить, яке призначення. Прикладом може бути: «Допомагає пережити холодну зиму лісовим мешканцям. Береже та дбає про ліс узимку». Може мати значення NULL.
- Amulet: Ця колонка відноситься до частини «бестіарій» і може бути відсутньою в інших статтях. Обереги чи предмети. Приклад зберігання даних

в цю колонку: «Предмети з металу. Ніж, монети, ключі. Не займає первістків та мізинців». Може мати значення NULL.

- Fairing: Ця колонка відноситься до частини «бестіарій» і може бути відсутньою в інших статтях. Гостинець, чим можна задобрити. Може мати значення NULL.
- MagicalItem: Ця колонка відноситься до частини «бестіарій» і може бути відсутньою в інших статтях. Наявність будь-яких чарівних чи міфічних предметів пов’язані зі створінням бестіарію. Може мати значення NULL.
- Origin: Ця колонка відноситься до частини «бестіарій» і може бути відсутньою в інших статтях. Цей стовпчик вказує на походження або історію створіння. Він може містити інформацію про те, як створіння з'явилося у міфологічному світі, його походження або легенди, пов'язані з його появою. Може мати значення NULL.
- UserId: Містить ідентифікатор посилання на користувача який створив або змінив статтю.
- createdAt: Зберігається час створення рядку з даними.
- updatedAt: Стовпець містить дату та час останнього оновлення запису.

Таблиця "users" містить 8 колонок, та відповідає за зберігання даних зареєстрованих користувачів веб-сайту. У цій таблиці можуть бути записи для користувачів з обмеженим доступом, вони мають обмежений набір функцій додатку, а також для адміністраторів, які мають повний доступ до всіх можливостей веб-сайту.

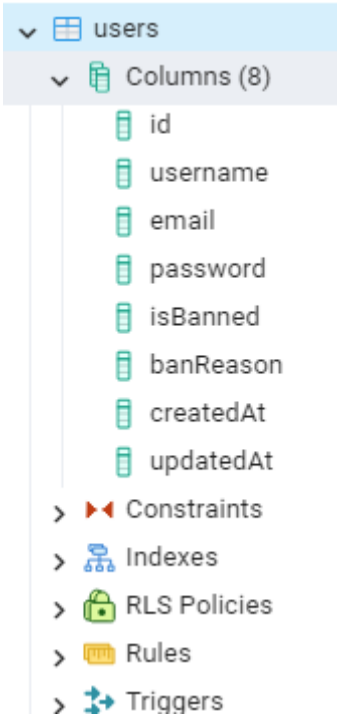


Рис. 2.3.2 Схема таблиці users

- ID: Це унікальний номер, ідентифікації таблиці. Він не може бути значення NULL, та автоматично створюється при створенні рядку.
- Username: Тут зберігається ім’я чи псевдонім користувача/адміністратора. Це значення необхідне для реєстрації та авторизації профілю, тому воно не може набувати значення NULL. Кожне ім'я має бути унікальним, тому при створенні застосовується умова (unique: true).
- Email: Зберігається електронна пошта користувача/адміністратора. Умови створення та зберігання ідентичні до username.
- Password: Захешований пароль користувача додатку. Він є унікальним та не може бути NULL.
- isBanned: Цей стовпчик має значення типу Boolean, та відповідає за стан користувача. Коли значення є TRUE, користувач забанений і не може використовувати додаток. У такий спосіб адміністратори може контролювати доступ додатку для всіх користувачів.

- banReason: При бані користувача, адміністратор може вписати в цей стовпчик текстове значення та пояснити, чому користувач був забанений на ресурсі. Це значення буде відображене у особистому кабінеті.

Таблиця “roles” відповідає за ролі які можливі у додатку. Вона складається з 5 стовпців (рис.2.3.3):

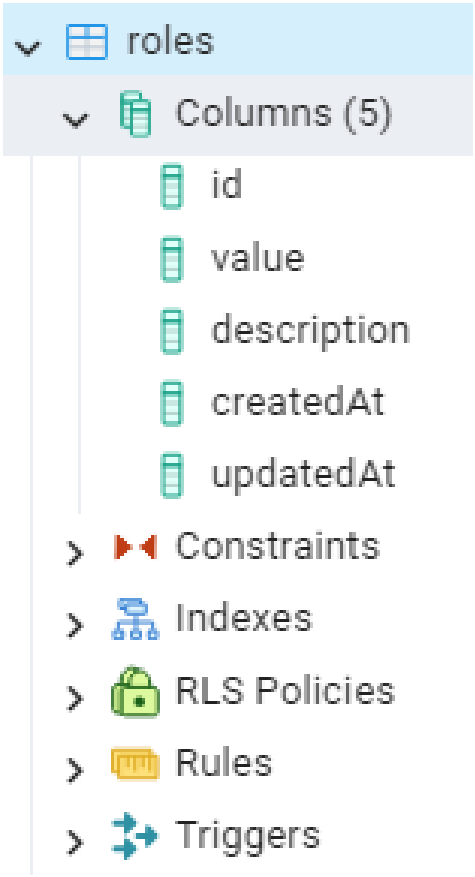


Рис. 2.3.3 Схема таблиці roles

- ID: Унікальний ключ ідентифікатор.
- Value: Значення цього рядку, може приймати значення admin та user.
- Description: Текстовий опис ролі.
- createdAt: Зберігається час створення рядку з даними.
- updatedAt: Стовпець містить дату та час останнього оновлення запису.

	id [PK] integer	value character varying (255)	description character varying (255)	createdAt timestamp with time zone	updatedAt timestamp with time zone
1	1	admin	Admin of application	2023-12-18 09:33:51.523+02	2023-12-18 09:33:51.523+02
2	2	user	User of application. Some functions are forbidden	2023-12-18 09:39:13.27+02	2023-12-18 09:39:13.27+02

Рис.2.3.4 Заповнена таблиця roles

Таблиця “user_roles” є проміжною таблицею між “users” та “roles. Використовується для зв'язку користувачів з ролями. Кожен рядок у цій таблиці містить зв'язок між конкретним користувачем і конкретною роллю. Наприклад, якщо користувач має роль "адміністратора", то в таблиці "user_roles" буде посилання на roleId = 1, що вказує на те, що цей користувач має роль "адміністратора". Якщо користувач має роль “user”, то в таблиці буде посилання на roleId = 2. Така структура дозволяє змінювати ролі користувачів без необхідності зміни основної таблиці користувачів.

	id [PK] integer	roleId integer	userId integer
1	9	1	2
2	10	2	3

Рис. 2.3.5 Приклад таблиці user_roles

Зв’язок між сервером та базою даних відбувається з використання Node.JS бібліотеки Sequelize. Розглянемо як ORM бібліотека працює, на прикладі методів users.

На рисунку 2.3.6 ми бачимо три функції, які створюють запит до бази даних та повертають значення user.

У асинхронній функції getAllUsers() метод за отримує всіх користувачів з PostgreSQL. Використовується метод findAll, який повертає масив даних.

Додаткова опція include: {all: true} зобов’язує бібліотеці отримати всі дані які пов’язані з user.

До асинхронної функції getUserByEmail(email:string) вноситься параметр email типу string. Він використовує метод findOne, який повертає один відповідний результат пошуку. Опціями пошуку вказується: where. { email }, include:{all:true}. Це означає що буде відбуватись в базі даних пошук користувача, де стовпець електронної пошти відповідає параметру email. Як і метод getAllUsers, він також отримує всі пов'язані дані з user.

Функція getUserById(id:string). Цей асинхронний метод отримує користувача з бази даних за їхнім ID. Він використовує метод findByPk, який означає "знайти за первинним ключем".

Кожен з цих методів використовує ключове слово await для тимчасового призупинення виконання функції, доки Promise, повернений методом Sequelize, не буде виконаний. Це спрощує написання асинхронного коду, оскільки його можна структурувати у синхронному стилі, що полегшує зрозуміння.

```
async getAllUsers() {
  const users = await this.userRepository.findAll({ include: { all: true } });
  return users;
}

async getUserByEmail(email: string) {
  const user = await this.userRepository.findOne({
    where: { email },
    include: { all: true },
  });

  return user;
}

async getUserById(id: string) {
  const user = await this.userRepository.findByPk(id, {
    include: { all: true },
  });

  return user;
}
```

Рис.2.3.6 Демонстрація методів users

2.4 Проектування структури серверної частини додатку

Структура середовища серверної чистини проілюстровано на рисунку 2.4.1.

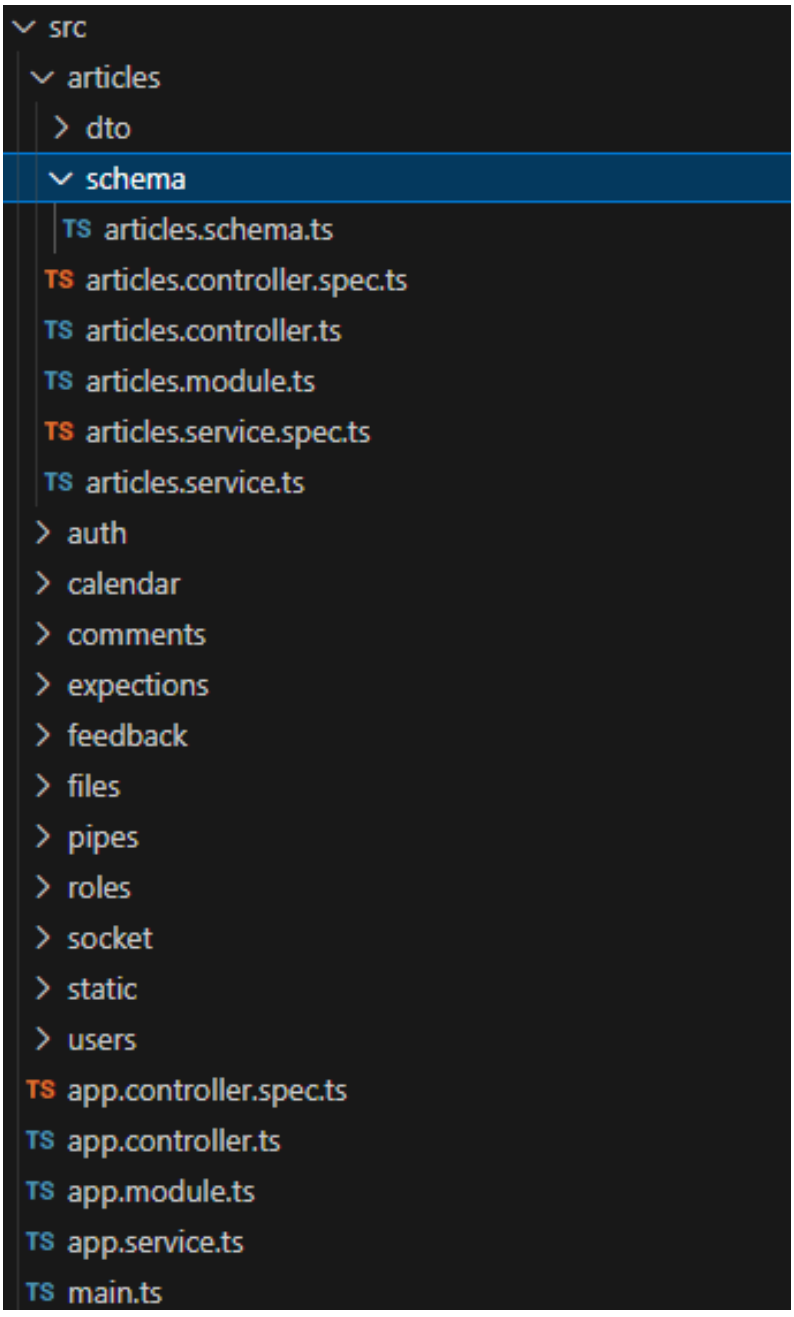


Рис. 2.4.1 Структура директорій проекту

Кожна папка складається з таких основних компонентів як: module, service, controller.

- Module групує компоненти для подальшого використання.
- Controller відповідає за обробку HTTP-запитів та повернення відповідей.

- Service містить бізнес-логіку та виконує різноманітні завдання, такі як взаємодія з базою даних чи іншими сервісами.
- Моделі (schema) відображають структуру даних.

Папка dto має файли DTO (Data Transfer Object) об’єктів, що використовується сервісами для передачі даних. Ці об’єкти зменшують кількість запитів на передачу даних, завдяки цьому вони часто використовуються для передачі даних між клієнтом та сервером.

Одним з необхідним елементів роботи додатку є аутентифікація та авторизація користувача. Процес перевірки даних особи називається – аутентифікація. Це може бути перевірка імені та паролю, розпізнавання особистих рис обличчя людини, голосове підтвердження особи. Процес авторизації - це наступний етап після аутентифікації. Після успішної аутентифікації система визначає, які дії конкретний користувач може виконувати в рамках системи.

Авторизація зазвичай базується на ролях або правах доступу, які надаються користувачеві після аутентифікації. Наприклад, адміністратор може мати доступ до всіх функцій додатку, тоді як звичайний користувач може мати обмежений доступ.

Оскільки NestJS працює по принципу модульності – це значно полегшує впровадження практик безпеки, таких як використання middleware для різних функцій проекту. Спеціальними класами в NestJS є Guards, які використовуються для захисту маршрутизації в додатку. Вони дозволяють контролювати доступ до функцій через наявність або відсутність аутентифікації. Основними видами Guards є [3]:

- AuthGuard: Використовується для захисту маршрутів, які потребують аутентифікації. Цей Guard перевіряє, чи користувач має дійсний токен аутентифікації (наприклад, JWT), і, якщо так, дозволяє доступ до маршруту.
- RolesGuard: Дозволяє контролювати доступ до маршрутів на основі ролей користувачів. Наведемо приклад: на рисунку 2.4.2 показано, що лише користувач з роллю «Адміністратор» може оновити інформацію про статтю.

- JwtAuthGuard: Спеціальний варіант AuthGuard, призначений для роботи з JWT-токенами. Цей Guard перевіряє валідність та наявність JWT-токенів у запиті перед доступом до маршруту.

```
@Roles('admin')
@UseGuards(RolesGuard)
@Patch(':id')
updateArticle(@Param('id') id: number, @Body() updateDTO: UpdateArticleDto) {
  return this.articleService.updateArticle(id, updateDTO);
}
```

Рис. 2.4.2 Демонстрація методу оновлення статті

Додаток використовує процес авторизації на основі JWT токену. При заповненні відповідної форми реєстрації, йде процес хешування паролю за допомогою криптографічної функції bcrypt. Далі йде процес збереження інформації користувача до відповідної таблиці бази даних. Варто зазначити що зберігається лише хеш паролю, тобто щоб отримати пароль системі потрібно буде розшифрувати його, взявши хеш з бази даних.

Модуль JwtModule використовується для налаштування та обробки JWT токенів в додатках NestJS. Після реєстрації або авторизації користувача, створюється JWT токен, який містить інформацію про нього та його роль доступу. Цей токен зазвичай зберігається на стороні клієнта, а серверна частина додатку використовує JwtModule для генерації та перевірки цих токенів. Серед налаштувань можна обрати коли токен буде не дійсним. Наприклад токен Velescape буде не дійсним через 24 години після створення, після чого користувач повинен знову авторизуватись Рис 2.4.3.

```
@Module({
  controllers: [AuthController],
  providers: [AuthService],
  imports: [
    forwardRef(() => UsersModule),
    JwtModule.register({
      secret: process.env.PRIVATE_KEY || 'SECRET',
      signOptions: {
        expiresIn: '24h',
      },
    }),
  ],
  exports: [AuthService, JwtModule],
})
export class AuthModule {}
```

Рис. 2.4.3. Конфігурація параметрів модулю AuthModule

Файли «app.module.ts» та «main.ts» є ключовими файлами, завдяки ним відбувається підключення до бази даних та запуск серверу веб-сайту. У файлі «app.module.ts» проводиться конфігурація бази даних та ORM бібліотеки, підключення моделей для бази даних, а також функціональних модулів з усіх папок.

```
@Module({
  imports: [
    ConfigModule.forRoot({
      envFilePath: '.env',
    }),
    ServeStaticModule.forRoot({
      rootPath: path.join(__dirname, '..', 'static'),
    }),
    SequelizeModule.forRoot({
      dialect: 'postgres',
      host: process.env.POSTGRES_HOST,
      port: Number(process.env.POSTGRESS_PORT),
      username: process.env.POSTGRES_USER,
      password: process.env.POSTGRES_PASSWORD,
      database: process.env.POSTGRES_DB,
      models: [
        User,
        Role,
        UserRoles,
        Article,
        Calendar,
        Message,
        Comment,
        Feedback,
      ],
      autoLoadModels: true,
      synchronize: true,
    }),
    UsersModule,
    RolesModule,
    AuthModule,
    ArticlesModule,
    SocketModule,
    CommentsModule,
    CalendarModule,
    FeedbackPostModule,
    FeedbackAdminModule,
  ],
  controllers: [AppController],
  providers: [AppService, SocketGateway],
})
export class AppModule {}
```

Рис. 2.4.4 Конфігурація головного модуля додатку на NestJS

2.5 Проектування структури клієнтської частини додатку

Клієнтський інтерфейс (User Interface) – це візуальна частина веб-сайту, яка в архітектурі MVC відповідає за компонент View. Важливо, щоб інтерфейс був зрозумілим та зручним у використанні, це допоможе залучити більше користувачів до вивчення міфології. Розглянемо які основні елементи необхідні для додатку.

- Головна сторінка.
- Navbar. Навігаційний компонент додатку. До елементів навігації входять такі посилання маршрутизації, як "Головна", "Статті", "Календар", "Логін", "Профіль" (якщо користувач авторизувався) Рис.2.5.1, і "Адмін" (посилання відображається лише для адміністраторів). Цей компонент є фіксованим, та є присутнім на кожній сторінці. Рис.2.5.2.

Рис.2.5.1 Навігаційна панель

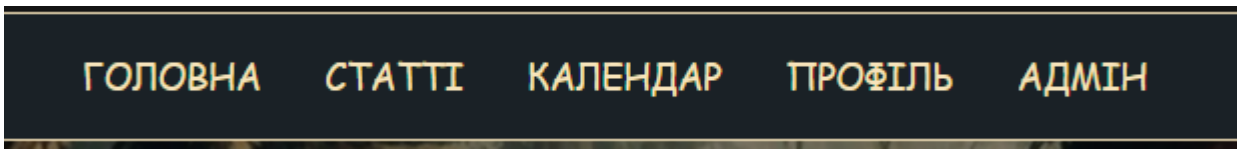


Рис.2.5.2 Навігаційна панель для адміністратора

- Сторінка з статтями продемонстрована на рисунку 2.5.3

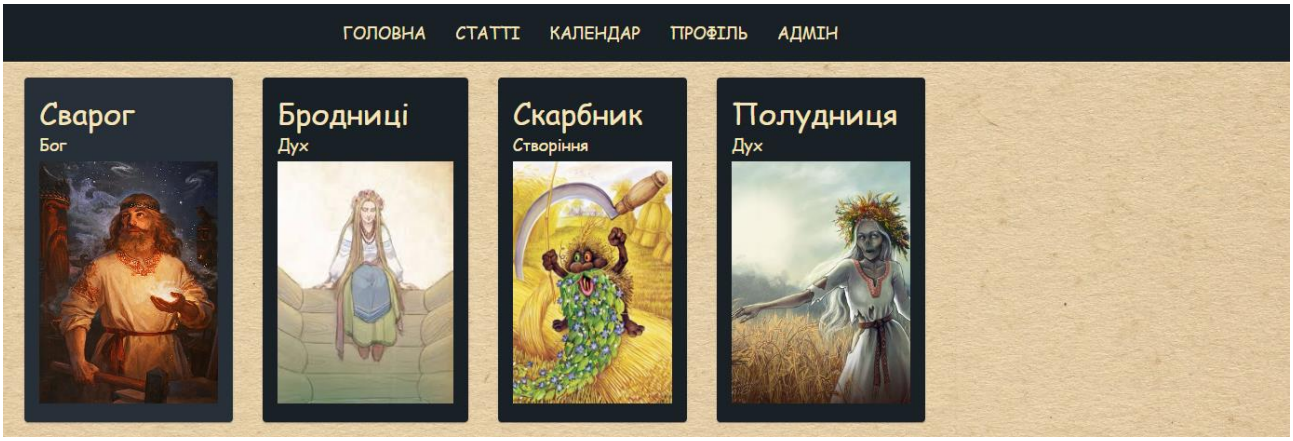


Рис.2.5.3 Сторінка з картками статей

- Поле пошук по імені. Користувач може ввести назву статті, результатом буде виведення відповідної статті. Рисунок.2.5.4

Рис.2.5.4 Пошукове поле

- Список кнопок фільтрації по категоріям. Рис.2.5.5

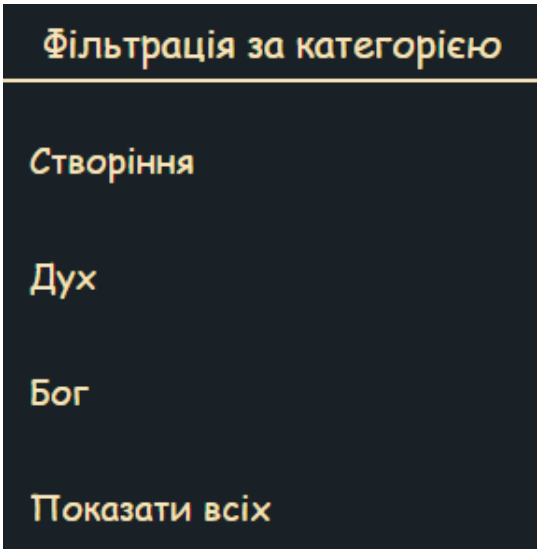


Рис. 2.5.5 Фільтри для вибору категорій статей

- Картки з інформацією про статтю продемонстровано на рисунку 2.5.6. Показує детальну картку однієї з статей, зокрема зображення та назву.

Рис.2.5.6 Інформаційна картка статті

- Сторінка стаття. Відображається вся інформація про статтю. Кнопка "Коментарі". Рис.2.5.7

Рис.2.5.7 Кнопка коментарів

- Компонент "Коментарі" з полям для введення тексту коментаря, кнопкою підтвердження форми та відображенням всіх коментарів статті на екрані. Для автора коментаря доступні кнопки редагування та видалення власного коментаря. Рис. 2.5.8

Рис.2.5.8 Коментарі

- Форма реєстрації та авторизації.
- Профіль. Виведення інформації про користувача, його ім’я, email, статус. Кнопка «Log Out». Форма оновлення даних користувача.
- Сторінка календар з графічними блоками, які містять список посилань на детальну інформацію про кожен день Рис.2.5.9.

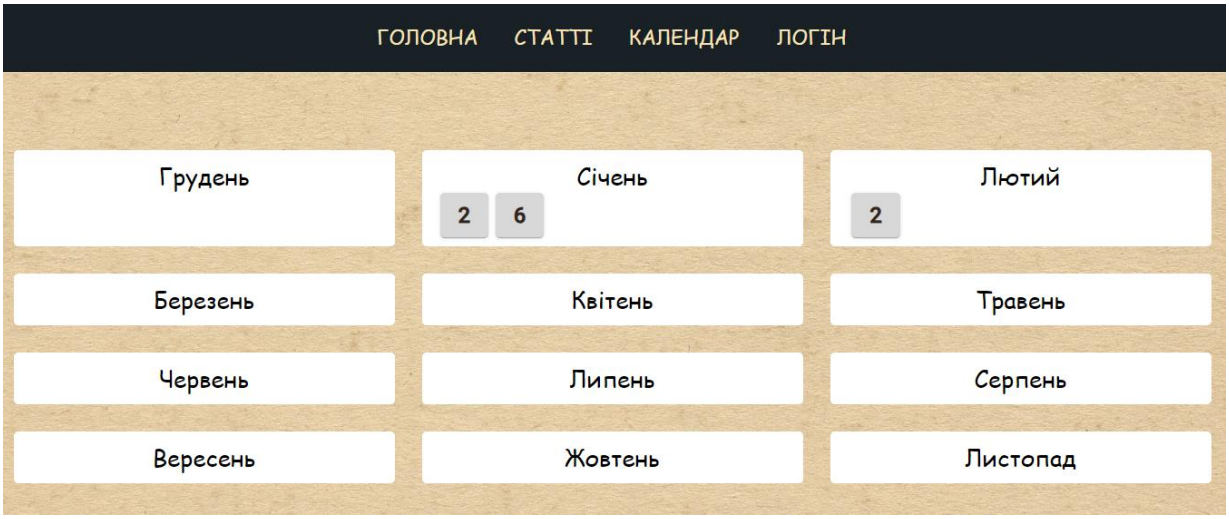


Рис.2.5.9 Сторінка з місяцеліком

- Сторінка адміністратора з бічною навігацією (sidebar) для переходу між функціоналом. Можливі шляхи переходу: "Зворотній зв’язок", "Користувачі", "Статті", "Календар". Рис.2.5.10

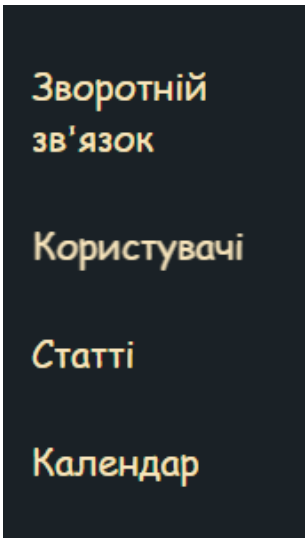


Рис.2.5.10 Адміністративний інтерфейс з боковим меню

- Сторінка "Зворотній зв’язок" з випадаючим списком елементів, що містить інформацію про повідомлення від користувачів, як тематика проблеми, відправник та детальна інформація про проблему. На сторінці є кнопки "Відзначити як виконано" та "Видалити".
- Сторінка "Користувачі" має список зареєстрованих користувачів додатку. Поле перегляду інформації про користувача. Кнопки "забанити користувача", "змінити роль".
- Сторінка "Статті" відображається список статей додатку Рис.2.5.11. Присутнє поле для пошуку статті за назвою Рис.2.5.12. Кнопка "Додати", яка відкриває форму для заповнення статті Рис.2.5.13.

Рис. 2.5.11

Рис. 2.5.12

Рис. 2.5.13

3 ОПИС РЕАЛІЗАЦІЇ СИСТЕМИ

3.1 Діаграма використання

Діаграми використання описують функції на високому рівні та обсяг системи. Ці діаграми також визначають взаємодії між системою та її акторами. Використані випадки та актори в діаграмах використання описують, що робить система та як актори її використовують, але не описують, як система працює всередині [9]. Діаграма використання додатку представлена на Рис.3.1.1

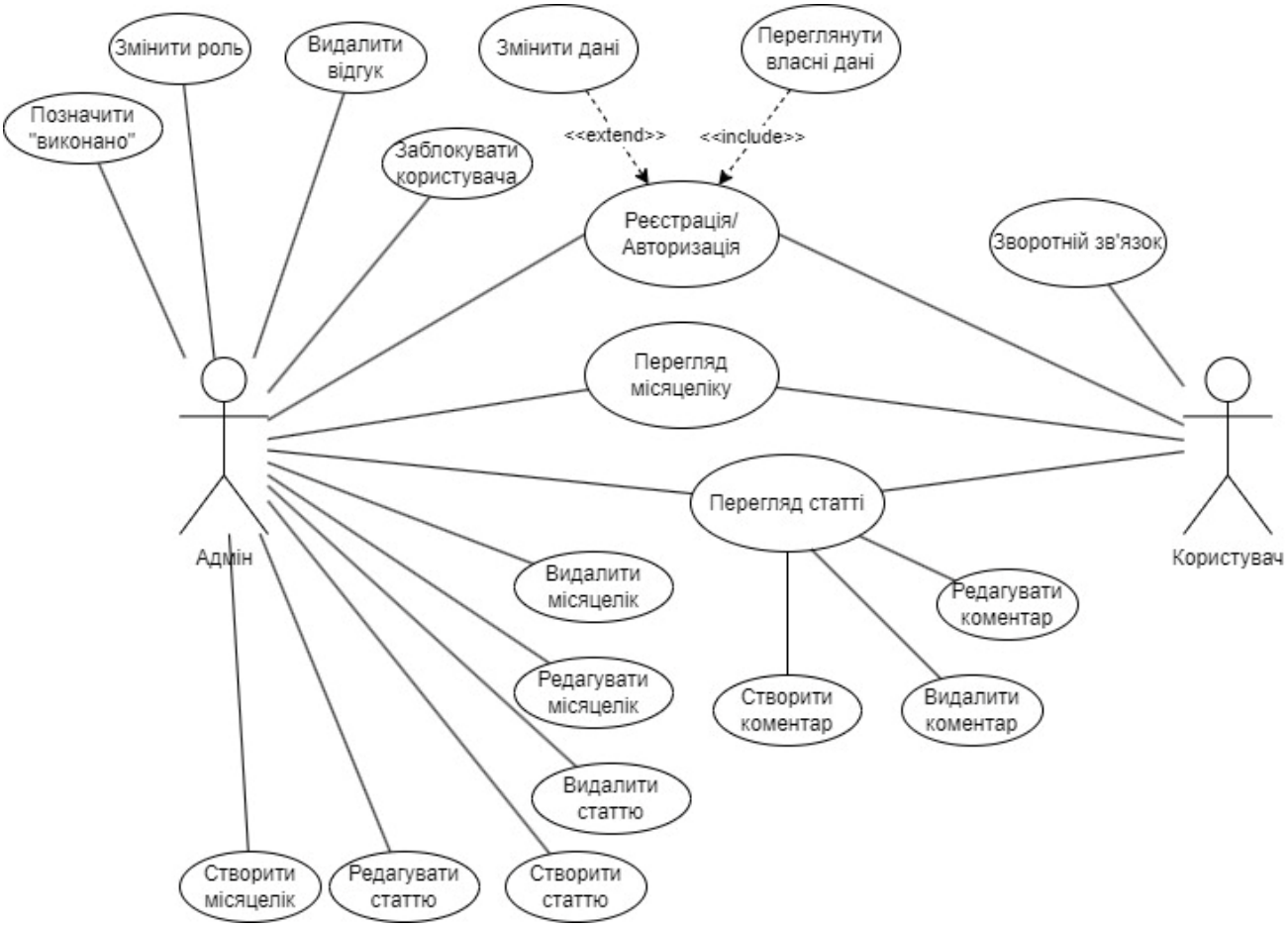


Рис. 3.1.1 Діаграма використання веб-застосунку

На цій use-case діаграмі описано різні випадки взаємодії користувачів (звичайних та адміністраторів) із системою. Ось ключові елементи діаграми:

Користувач має доступ до таких функцій:

- Перегляд статей.
- Написання feedback - можливість залишити відгук.
- Перегляд Місяцеліку.
- Профіль (з можливістю змінювати дані) - перегляд та редагування особистих даних.
- Коментарі - взаємодія з коментарями, включаючи створення, перегляд, редагування та видалення своїх написаних коментарів.

Адміністратор (Admin) має такі можливості:

- Управління таблицями користувачів, адміністраторів, статтями, відгуками.
- Перегляд таблиці користувачів.
- Регулювання прав користувачів - зміна прав користувачів або адміністраторів.
- Блокування користувачів - можливість блокувати або розблоковувати користувачів.
- Управління статтями (Створити, Видалити, Редагувати) - керування статтями.
- Управління контентом календаря (Створити, Видалити, Редагувати).
- Перегляд таблиці feedback. Можливість поставити позначку на “Виправлено”. Видалити повідомлення.

Звичайний користувач та адміністратор мають можливість зареєструватись та авторизуватись. Обом доступна можливість оновити персональні дані в Особистому кабінеті.

3.2 Діаграма класів

Діаграма класів являється однією з шести структурних діаграм UML, які відображають архітектуру програмного додатку. Діаграми класів є своєрідними кресленнями системи або її підсистем. З їх допомогою можна моделювати об'єкти, які складають систему, відображати взаємозв'язки між об'єктами, а також описувати функції об'єктів і послуги, які вони надають.[10].

В залежності від масштабованості та складності проєкту, діаграму можна поділити на кілька субдіаграм, для більш зручнішого та зрозумілішого користування. У випадку веб-додатку Velescape, було вирішено створити діаграми класів до кожного модуля NestJs.

Розглянемо таблицю діаграм Articles Рис.3.2.1. Ми маємо шість таблиць з класами.

- Article. Цей клас є моделлю, використовуваною у Sequelize.
- ArticleService: Має методи для обробки даних. Тут присутні такі функції, як:
    - getArticleById отримує статтю з бази даних за її ID.
    - Метод getAllArticles отримує всі статті з бази даних.
    - Функція updateArticle оновлює існуючу статтю. Спочатку він отримує статтю за її ID, потім оновлює її властивості зі значеннями у updateDTO та зберігає оновлену статтю у базу даних.
    - Метод deleteArticle видаляє статтю за її ID.
    - Метод searchArticle шукає статті, заголовок яких містить заданий рядок запиту. Він використовує оператор Op.like з Sequelize для виконання запиту LIKE у базі даних.
- ArticleController: Зберігає HTTP запити та методи керування ними. Має такі фукнції як:
    - createArticle. В тіло запиту використовується декоратор dto, та image, для завантаження графічного зображення.

  - deleteArticle видалення статті. Приймає ідентифікаційний номер.
  - Метод deleteComment який видаляє певний коментар з певної статті. Має параметри Id та commentId.
  - Метод filterArticles обробляє GET-запит використовуючи декоратор @Query('title'), для фільтрації статей за параметром title
  - getArticle, використовує декоратор @Param('id').
  - Отримання коментарів за ідентифікатором статті getCommentsByArticleId
  - updateArticle використовує параметр ID та updateDTO. Ідентифікаційний номер для того щоб визначити яку статтю оновити, dto для визначення яких даних.
  - updateComment. Використовує параметри ID та commentId для ідентифікації номера коментаря, а файл dto для даних коментаря.
- CreateArticleDto - об'єкт призначений для передачі інформації для створення нової статті.
- UpdateArticleDto - об'єкт застосовується для надсилання даних про оновлення змісту статті.
- FileService – клас який відповідальний за збереження графічних елементів статті. Функція createFile приймає параметр file, завантажуючи зображення до файлової системи додатку.

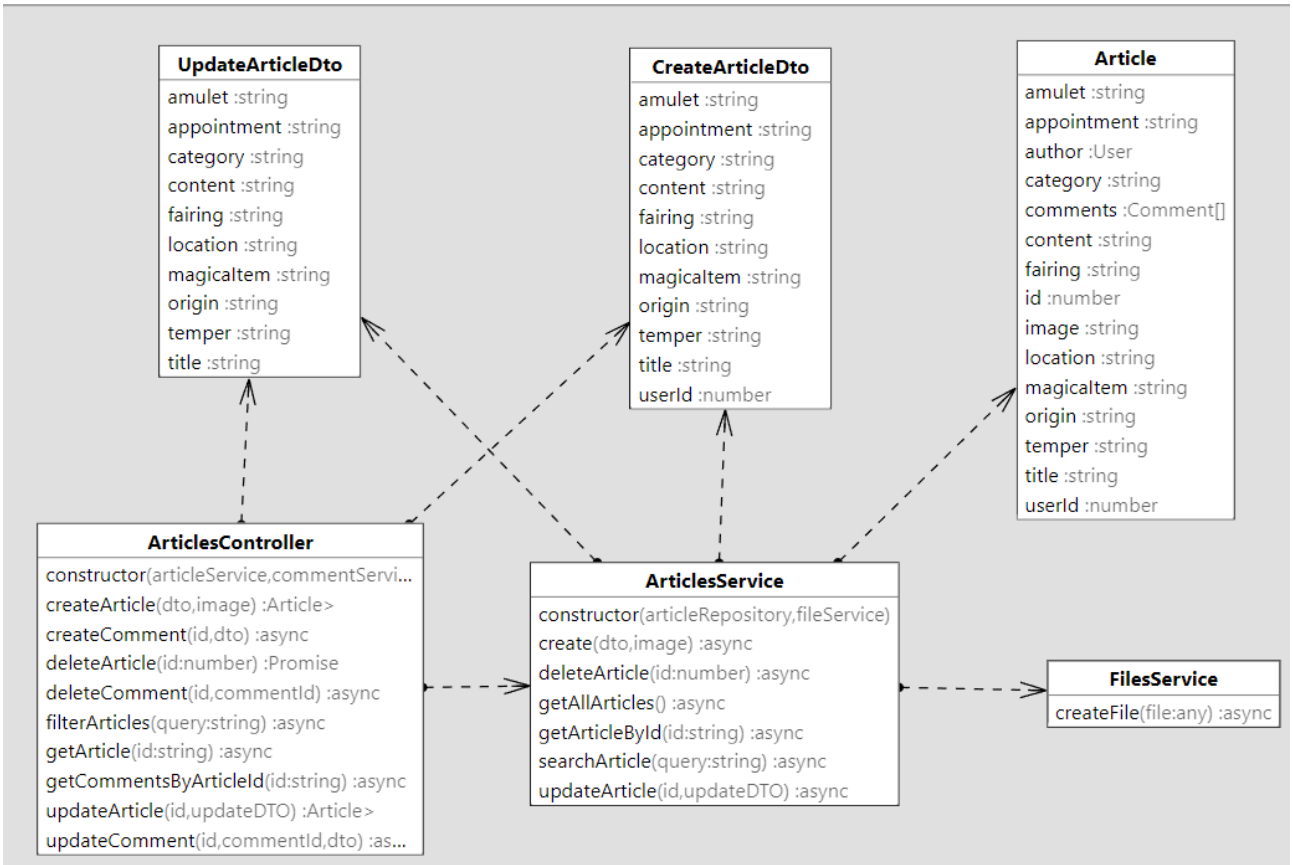


Рис.3.2.1 Діаграма класів Articles

На діаграмі модулів Рис.3.2.2, можна переглянути як модулі додатку взаємопов’язані між собою структурною архітектурою NestJS. Кожен модуль відповідає за свій специфічний аспект функціональності веб-додатку. Ось ключові елементи та їх ролі:

- AuthModule – відповідає за аутентифікацію та авторизацію користувачів. Він є одним з ключових модулів системи, бо забезпечує захист неавторизованого доступу до системи.
- ArticlesModule - модуль, призначений для управління статтями на сайті. Цей модуль має зв’язки з такими функціональними модулями, як FilesModule для роботи із зображеннями та CommentsModule для коментарів статті.
- UsersModule та RolesModule - ці модулі керують користувацькими профілями та ролями в системі, забезпечуючи можливість зміни доступу та функціоналу в залежності від ролі користувача.

- CalendarModule – модуль який відповідальний за керування наповненням Місяцеліку.
- FeedbackAdminModule та FeedbackPostModule - ці модулі управляють зворотним зв'язком від користувачів.
- FilesModule - відповідає за завантаження фотографій та картинок до файлової системи.
- CommentsModule - забезпечує функціональність додавання, редагування, та видалення коментарів користувачів.

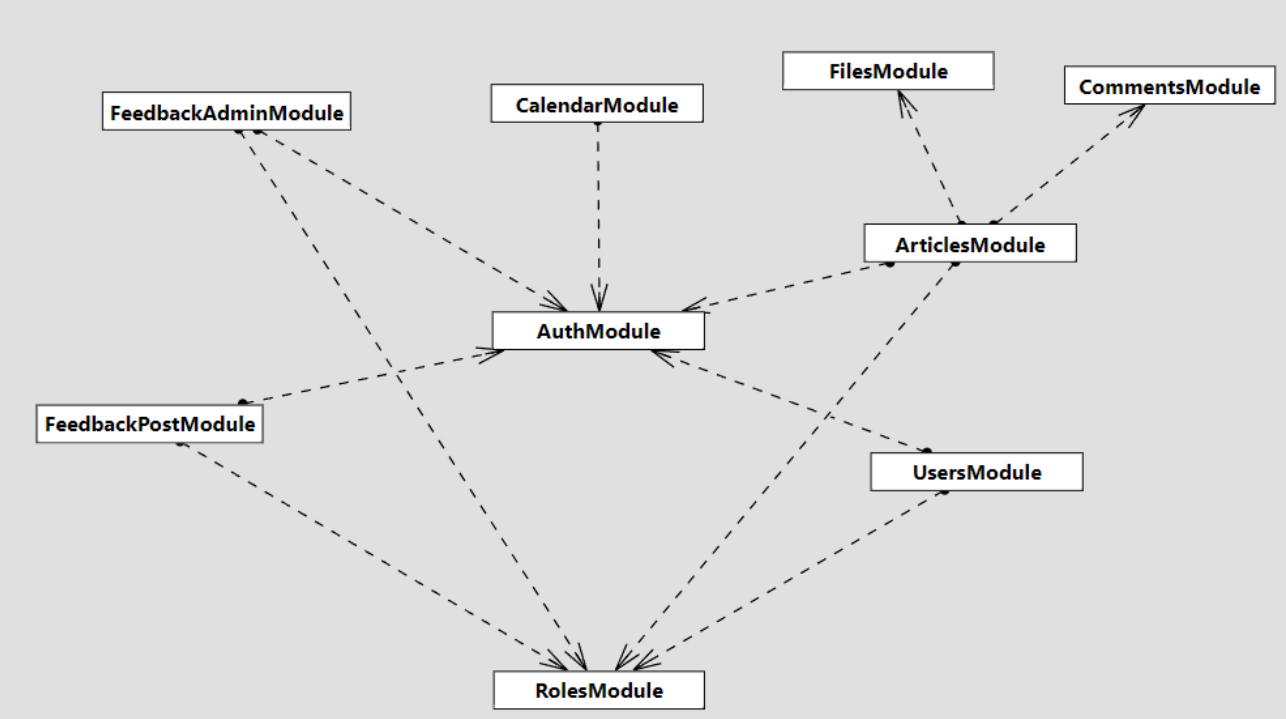


Рис.3.2.2 Діаграма модулів додатку

4 ТЕСТУВАННЯ ПРОГРАМИ

Для тестування серверної та клієнтської частини додатку було використано програму Postman для перевірки HTTP запитів API. Postman є ефективним інструментом, який дозволяє імітувати запити до сервера та перевіряти відповіді в режимі реального часу, що забезпечує можливість глибоко аналізувати та перевіряти взаємодію між клієнтом і сервером. На рисунку 4.1 представлено запит POST, який надсилає поля email та password по шляху /auth/login/ на перевірку аутентифікації. При невдалій перевірці одного з двох елементів запиту, система надсилає повідомлення у консоль, де попереджає користувача, про неправильний email чи password.

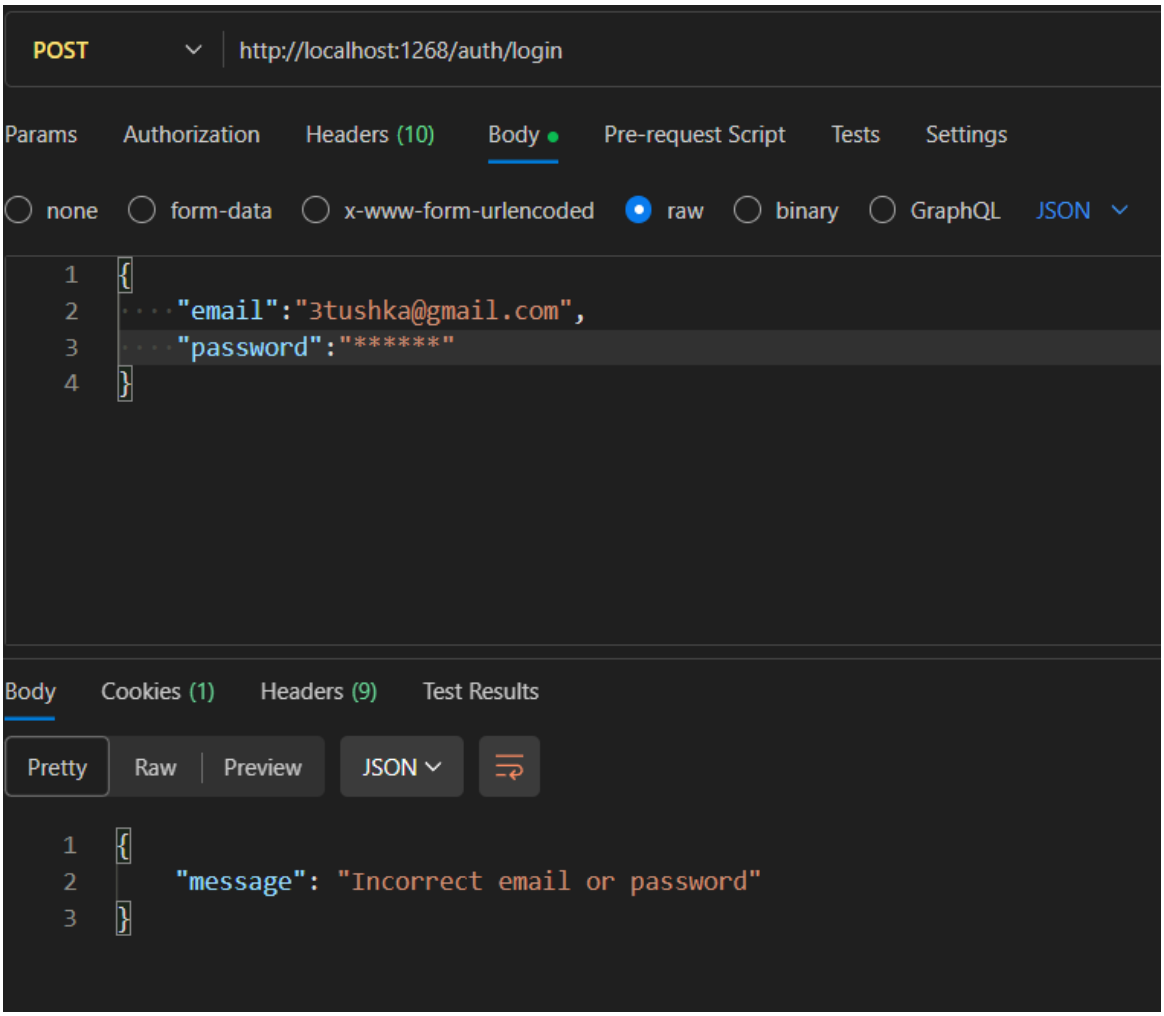


Рис.4.1 Приклад запиту авторизації

На рисунку 4.2 показано правильний запит, який повертає JWT token для сесії користувача. Цей токен використовується в системі для подальшої авторизації користувача і надає йому доступ до авторизованих функцій системи.

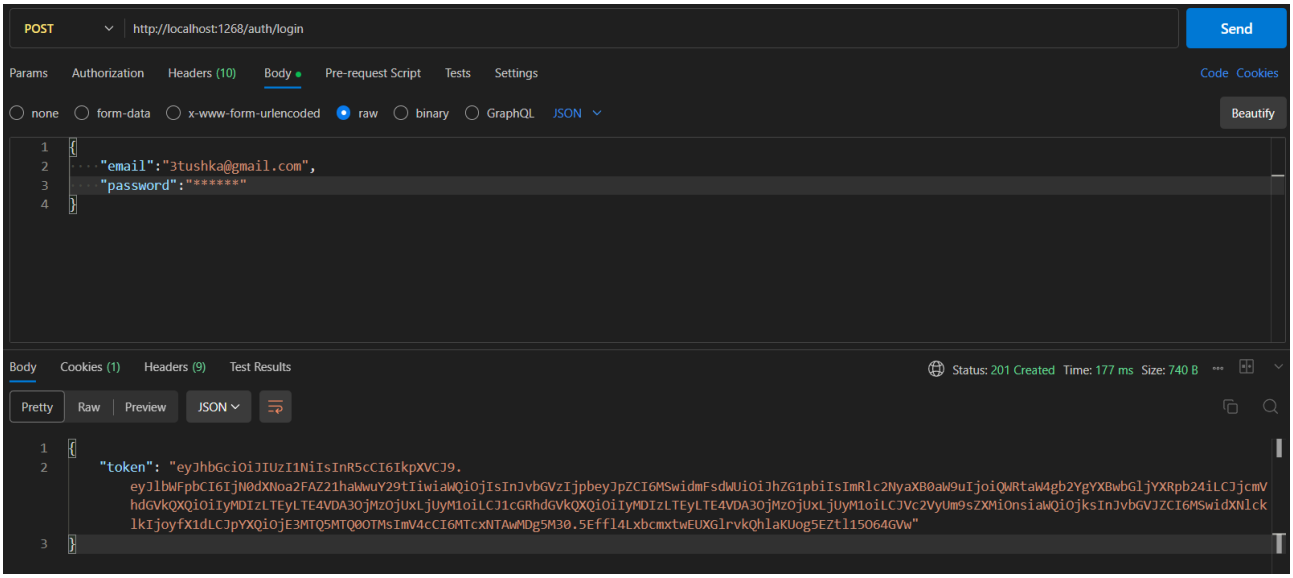


Рис. 4.2 Вдала запит авторизації

Розглянемо тестування внесення даних до місяцеліку без авторизації користувача. На Рисунку 4.3 зображено дані, які спробували внести в базу даних через Postman. Важливо зауважити відсутність Bearer токена в параметрах авторизації. Без цього токену неможливо продовжити та підтвердити внесення даних. Відсутність токену блокує подальші дії та видає помилку statusCode:403, це означає, що користувач немає доступу до цього функціоналу. Параметр message: “Not auth” скорочена версія помилки неавторизованого доступу до цієї функції.

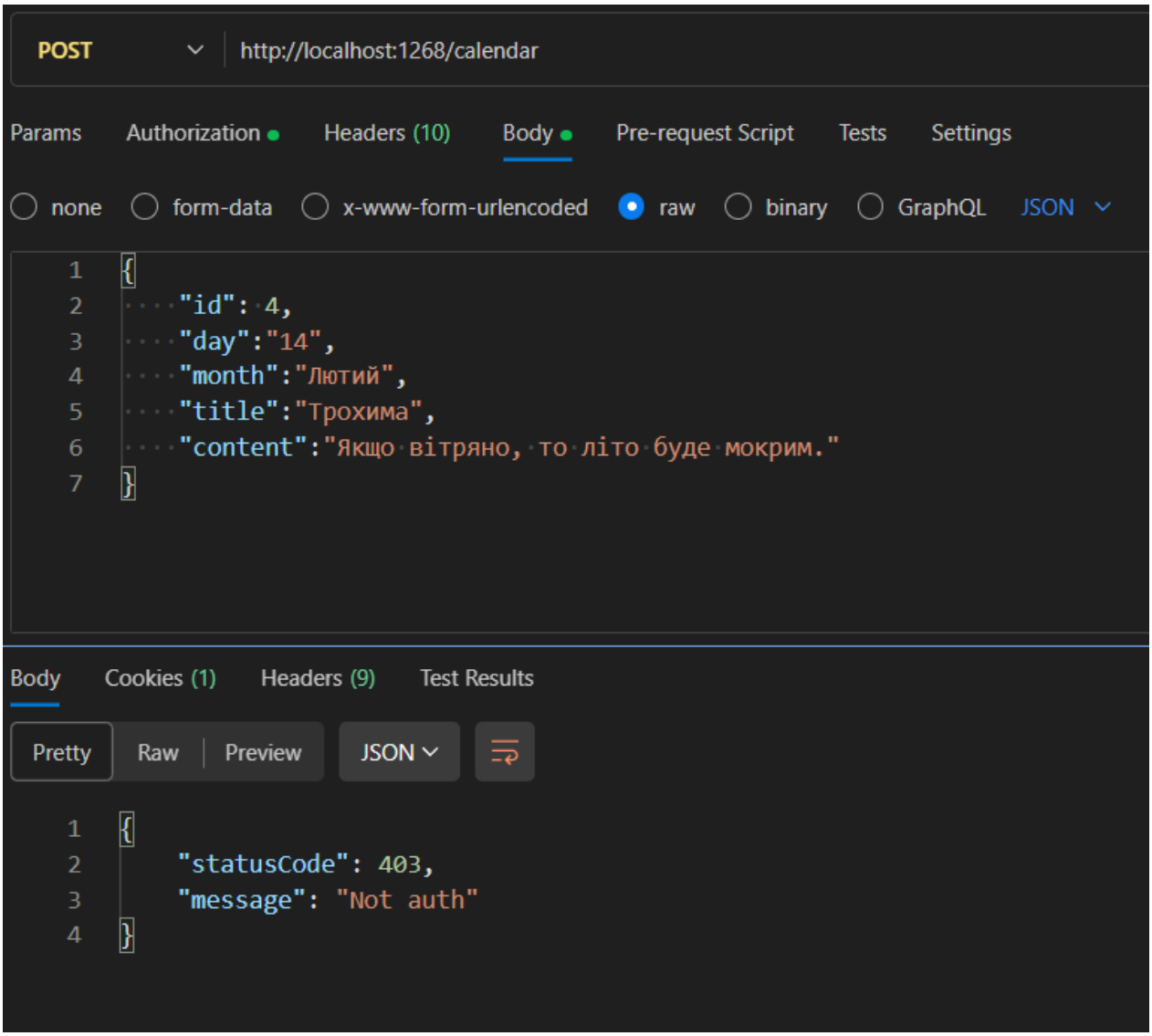


Рис. 4.3 Приклад внесення даних

На Рисунку 4.4 був доданий JWT token до параметру авторизації. Представлено успішне додавання даних через запит POST.

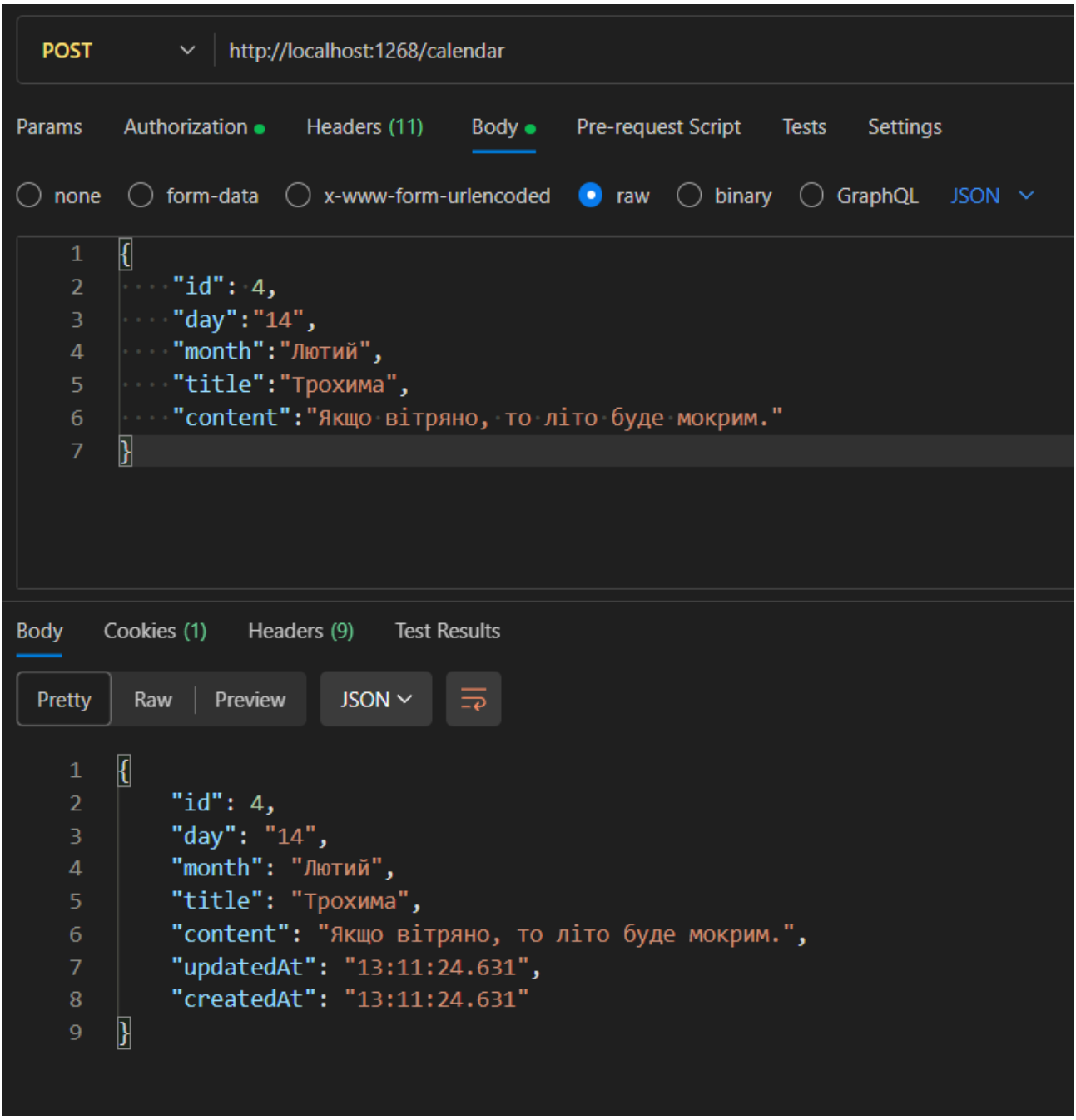


Рис.4.4 Приклад вдалого внесення даних

На рисунку 4.5 зображено інтерфейс форми для входу у веб-додаток, що включає поля для введення електронної пошти та пароля. Форми були зроблені з використання бібліотеки Angular Material, що полегшила розробку форм та інтерфейсу користувача загалом. За допомогою вбудованих функцій mat-raised-button:[disabled], кнопка підтвердження форми залишається неактивною, поки поля

форми не заповнені або містять помилки валідації, забезпечуючи таким чином додаткову перевірку даних перед їхнім відправленням.

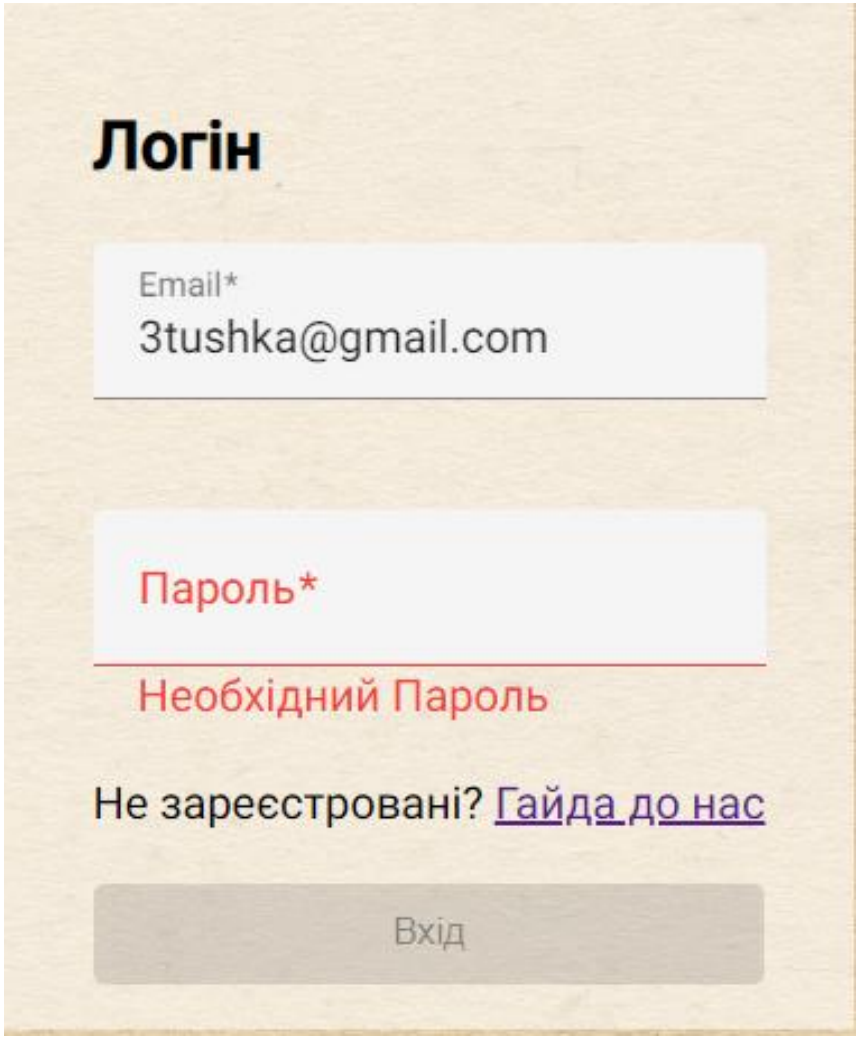


Рис. 4.5 Форма авторизації

ВИСНОВКИ

В ході виконання цієї дипломної роботи було спроектовано та реалізовано веб-додаток, призначений для надання інформації про слов'янську міфологію та її культурні аспекти.

1. Проведено аналіз вже існуючих аналогів додатку для визначення переваг та недоліків. Було вирішено розробити веб-додаток, оскільки такий підхід забезпечує кращу масштабованість, дозволяючи легко розширювати функціональність і доступність додатку для більшої кількості користувачів з різних регіонів і пристроїв.

2. Вибір технологічного стеку. Були досліджені технічні засоби, такі як сайти конструктори, CMS та фреймворки. Було обрано JavaScript, Angular, та NestJs як основні інструменти розробки через їх гнучкість та потужні можливості у створенні динамічних веб-додатків.

3. Проаналізовано і вибрано інструменти, які оптимально впораються з реалізацією серверної та клієнтської частин додатку. Зокрема, було вибрано середовище розробки Visual Studio Code та систему управління базами даних PostgreSQL.

4. В процесі розробки веб-додатку були реалізовані ключові компоненти: спроектовано функціональні модулі серверної частини, які забезпечують основу для обробки даних та взаємодії з користувачами. Була реалізована структура бази даних, що дозволяє зберігати і ефективно обробляти необхідну інформацію. Побудований користувацький інтерфейс.

5. Виконано тестування запитів додатку з допомогою програми Postman. Проведено перевірку обмежень функціоналу для користувача. Проведено тестування користувацьких форм, включаючи форми реєстрації, входу, та інші форми взаємодії в додатку.

6. Робота пройшла апробацію на Всеукраїнській науково-технічній конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, ДУІКТ та «Всеукраїнській науково-практичній конференції молодих учених», 16 травня 2024 р., Київ, КСУІМГ. За результатами участі опубліковано тези доповідей [15, 16].

ПЕРЕЛІК ПОСИЛАНЬ

1. State-of-Art Frameworks for Front-end and Back-end Web Development [Електронний ресурс] // Department of Information and Communication Technology, South Eastern University of Sri Lanka, Sri Lanka. – 2022. – Режим доступу до ресурсу: http://ir.lib.seu.ac.lk/bitstream/123456789/6339/3/62-67.pdf.

2. BESTIARY.US. [Електронний ресурс] Режим доступу до ресурсу: https://www.bestiary.us

3. OLDWORLDSGODS. [Електронний ресурс] Режим доступу до ресурсу: https://oldworldgods.com

4. THE MEDIEVAL BESTIARY. [Електронний ресурс] Режим доступу до ресурсу: https://bestiary.ca/index.html

5. Офіційна документація Angular [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io.

6. Robinson S. Content Management System (CMS) [Електронний ресурс] / S. Robinson, S. Amsler, F. Churchville – Режим доступу до ресурсу: https://www.techtarget.com/searchcontentmanagement/definition/content-management-system-CMS.

7. Документація NestJS [Електронний ресурс] – Режим доступу до ресурсу: https://nestjs.com.

8. PostgreSQL Офіційна документація [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/docs/16/intro-whatis.html.

9. Sequelize Офіційна документація [Електронний ресурс] – Режим доступу до ресурсу: https://sequelize.org/docs/v6/getting-started/.

10. The Good and the Bad of Angular Development [Електронний ресурс] – Режим доступу до ресурсу: https://www.altexsoft.com/blog/the-good-and-the-bad-of-angular-development/.

11. Use-case diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-use-case.

12. Class diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-class.

13. JSON Web Tokens [Електронний ресурс] – Режим доступу до ресурсу: https://auth0.com/docs/secure/tokens/json-web-tokens.

14. Westerveld D. API Testing and Development with Postman: A practical guide to creating, testing, and managing APIs for automated software testing / Dave Westerveld., 2021. – 340 с. – (Packt Publishing Ltd).

15. Панасюк А.С., Довженко Т.П., РОЗРОБКА RESTFUL API З ВИКОРИСТАННЯ ФРЕЙМОРКУ NESTJS. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». 24 квітня 2024 р., Київ, Державний університет інформаційно комунікаційних технологій, Збірник тез. К.: ДУІКТ, 2024. С. 95.

16. Панасюк А.С., Довженко Т.П., Sequelize як інструмент ORM для забезпечення доступу реляційної бази даних PostgreSQL. «Всеукраїнської науково–практичної конференції молодих учених». 16 травня 2024 р., Київ, КСУІМГ, Збірник тез. К.: КСУІМГ, 2024. С.169.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«Розробка застосунку для ознайомлення зі слов'янською міфологією з використанням JS, Angular, NestJs»

Виконав студент 4 курсу
групиПД-44
Панасюк Андрій Сергійович

Керівник роботи
к.т.н., доцент кафедриІПЗ Довженко Тимур Павлович

Київ - 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - розширення засобів інформування про слов'янську міфологію через веб-застосунок.
- **Об'єкт дослідження** процес ознайомлення зі слов'янською міфологією.
- **Предмет дослідження** веб-застосунокдля ознайомлення зі міфологією України.

ЗАДАЧІДИПЛОМНОЇ РОБОТИ

1. Провести аналіз існуючихвеб-додатків на тему слов'янської міфології, визначити їх переваги та недоліки .
2. Визначити вимогидо веб-додатку на основі результатів аналізу існуючих рішень .
3. Провести огляд технічних рішень та засобів розробки для побудови веб-застосунку про слов'янську міфологію.
4. Спроектувати архітектуру системи застосунку.
5. Створити базу даних, розробити серверну та клієнтську частину додатку на основі визначених вимог.
6. Провести тестування функціональності системи та користувацького інтерфейсу.

3

АНАЛІЗ АНАЛОГІВ

Особливості	The Medieval Bestiary	OldWorldsGods	Bestiary.us	Власний додаток
Розгорнута інформація	+	+	+	+
Фільтрація за назвою та категорією	-	+	+	+
Можливість коментувати	-	-	+	+
Інтерактивні тести	-	+	-	+
Місяцелік	-	-	-	+
Відповідність дизайну сучасним вимогам	-	+	-	+
Реєстрація та Авторизація	-	-	+	+
Наявність особистого кабінету	-	-	+	+

4

ВИМОГИ ДОДОДАТКУ

Функціональніможливості

1. Можливістьпереглянути статті.
2. Фільтрація та пошук статті за назвою чи категорією.
3. Користувачі мають змогу коментувати статті.
4. Переглядмісяцеліку.
5. Реєстрація та авторизація користувача.
6. Зміна ролей користувачів адміністратором ресурсу.
7. Можливість блокування та розблокування користувачів адміністратором.
8. Проходження тестів для перевірки знань.

Нефункціональнівимоги:

1. Паролі користувачів мають бутизашифорвані
2. Авторизація та аутентифікація користувача за ролями за допомогоюJWT-токенів.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

JavaScript

TypeScript

Angular

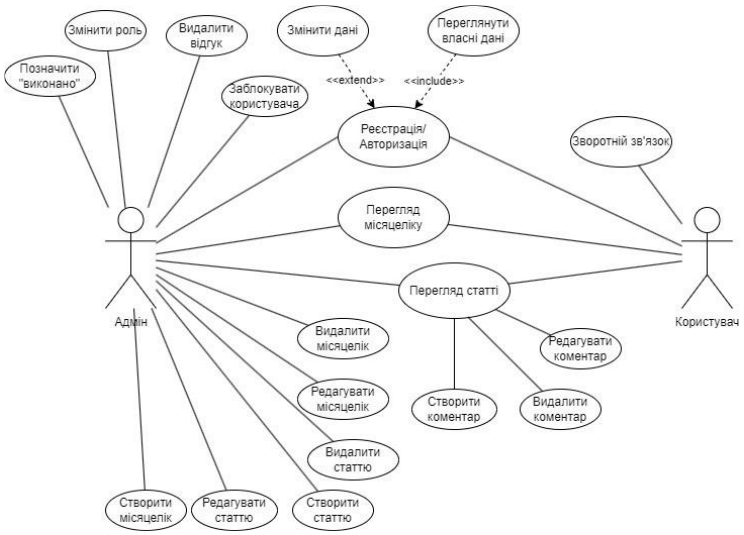
PostgreSQL

Sequelize

NestJS

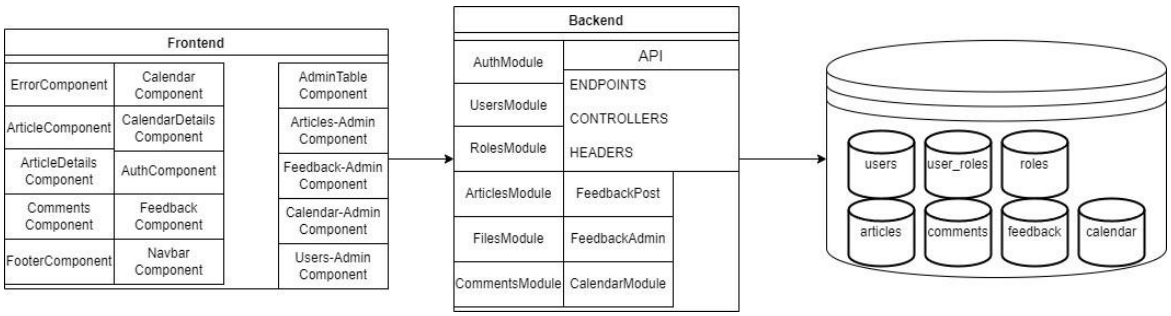
6

ДІАГРАМА ВИКОРИСТАННЯ



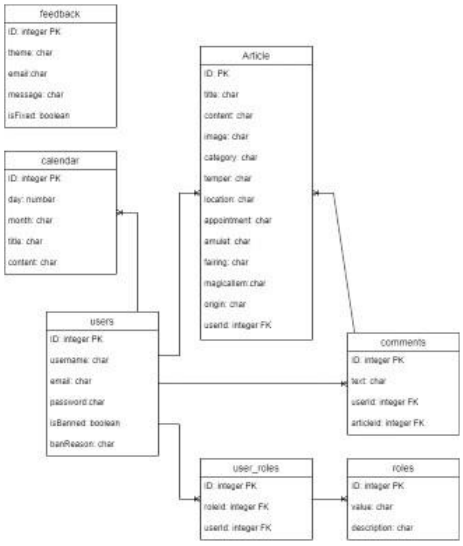
7

СХЕМА АРХІТЕКТУРИ ЗАСТОСУНКУ



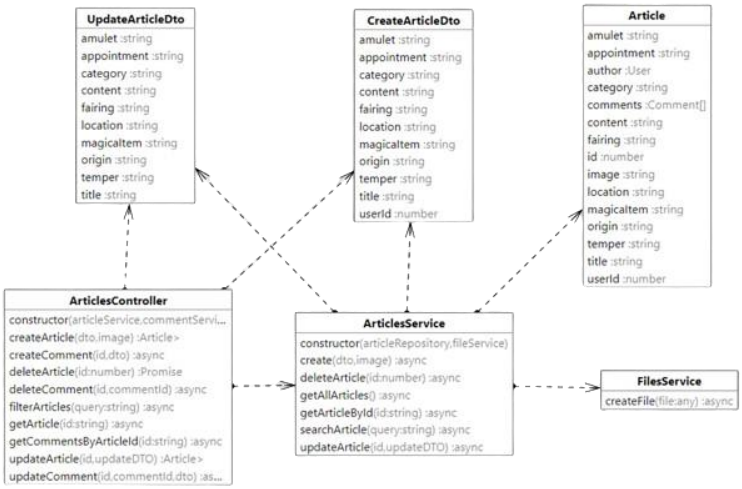
8

СХЕМА БАЗИ ДАНИХ



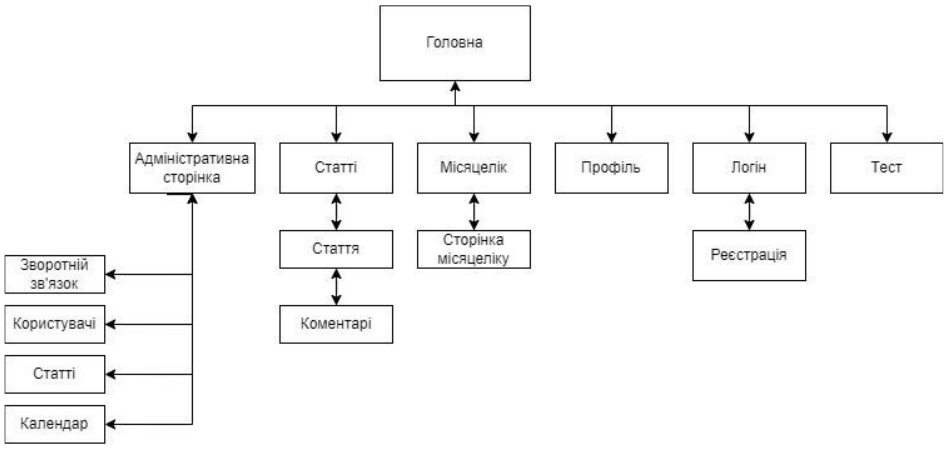
9

ДІАГРАМА КЛАСІВ ARTICLE



10

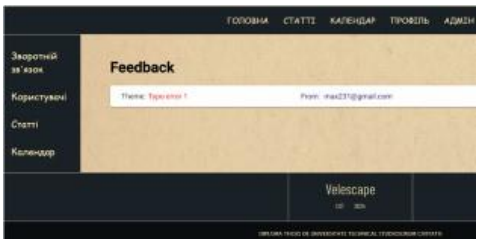
МАПА ВЕБ -САЙТУ



11

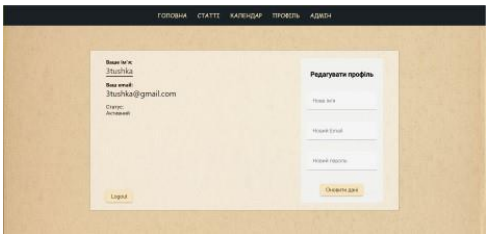
ЕКРАННІ ФОРМИ

Сторінка зі статтями



Сторінка адміністратора

Сторінка з місяцеліком



Профіль

12

ЕКРАННІ ФОРМИ

Приклад статті

13

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

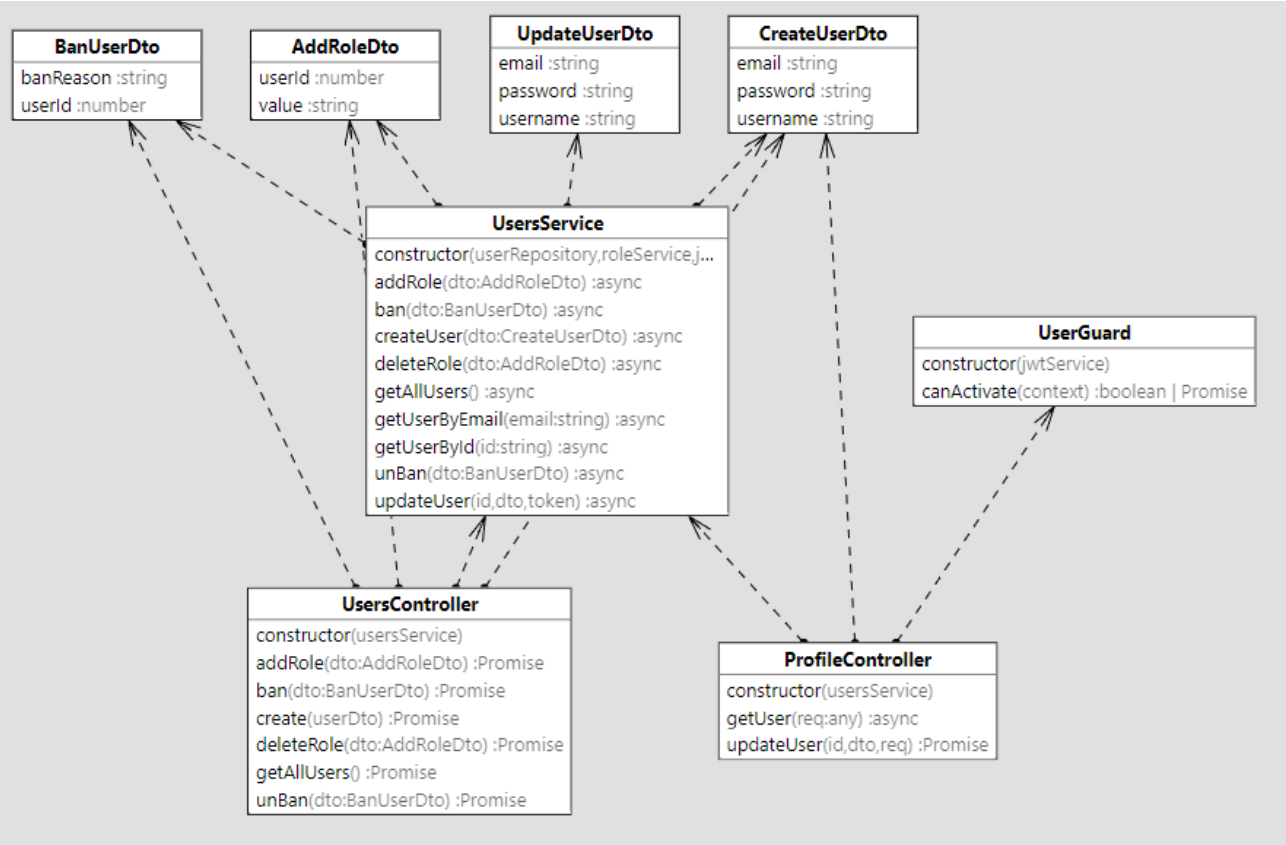
1. Панасюк А.С., Довженко Т.П., РОЗРОБКА RESTFUL API З ВИКОРИСТАННЯ ФРЕЙМОРКУ NESTJS . Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». Збірник тез. 24 квітня 2024 року, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. С. 95
2. Панасюк А.С., Довженко Т.П., Sequelize як інструмент ORM для забезпечення доступу реляційної бази даних PostgreSQL. «Всеукраїнської науково–практичної конференції молодих учених». Збірник тез 16 травня 2024 року, КСУІМГ, м. Київ. К.: КСУІМГ, 2024. С.169

14

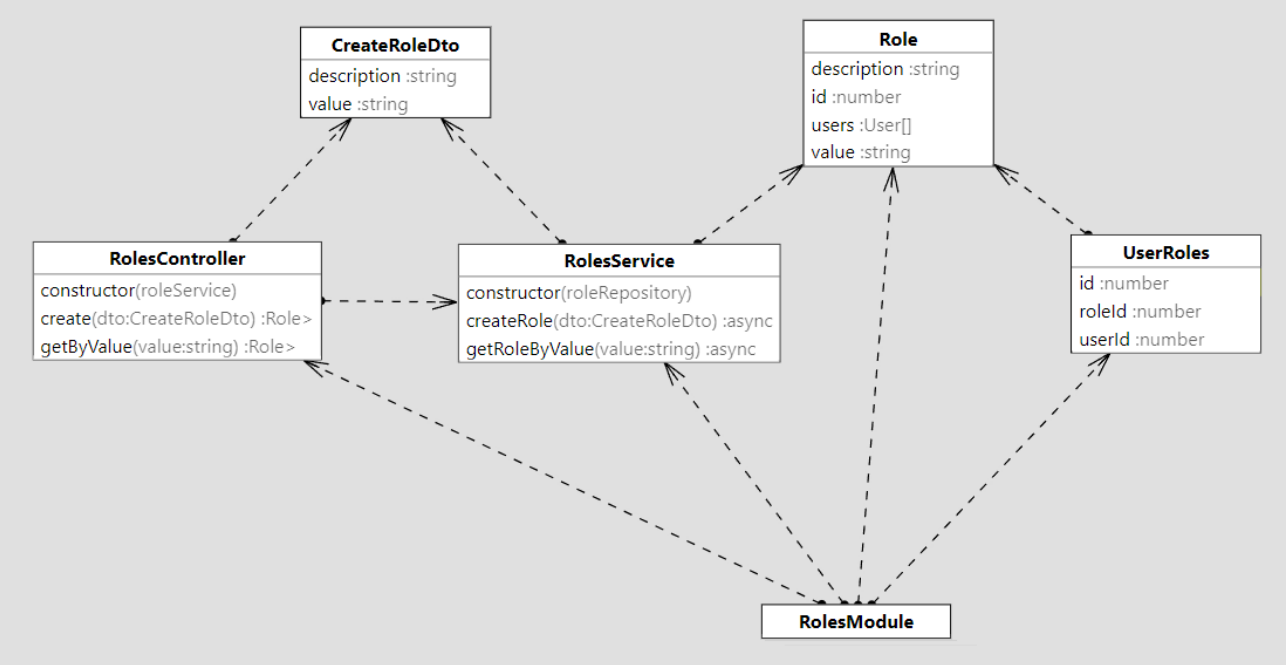
ВИСНОВКИ

1. Проведено аналіз існуючих аналогів для ознайомлення зі слов'янською міфологією, визначено їх переваги та недоліки.
2. Визначено вимоги до веб -застосунку на основі результатів аналізу.
3. Проведено огляд технічних рішень та засобів розробки для побудови веб - застосунку.
4. Спроектовано архітектуру веб -застосунку.
5. Було створено базу даних, розроблено серверну та клієнтську частину додатку на основі визначених вимог, з використанням зазначених технологій розробки веб застосунків.
6. Проведено тестування: запитів додатку з допомогою програми Postman, обмежень функціоналу для користувача, користувацьких форм, включаючи форми реєстрації, входу, та інші форми взаємодії в додатку.

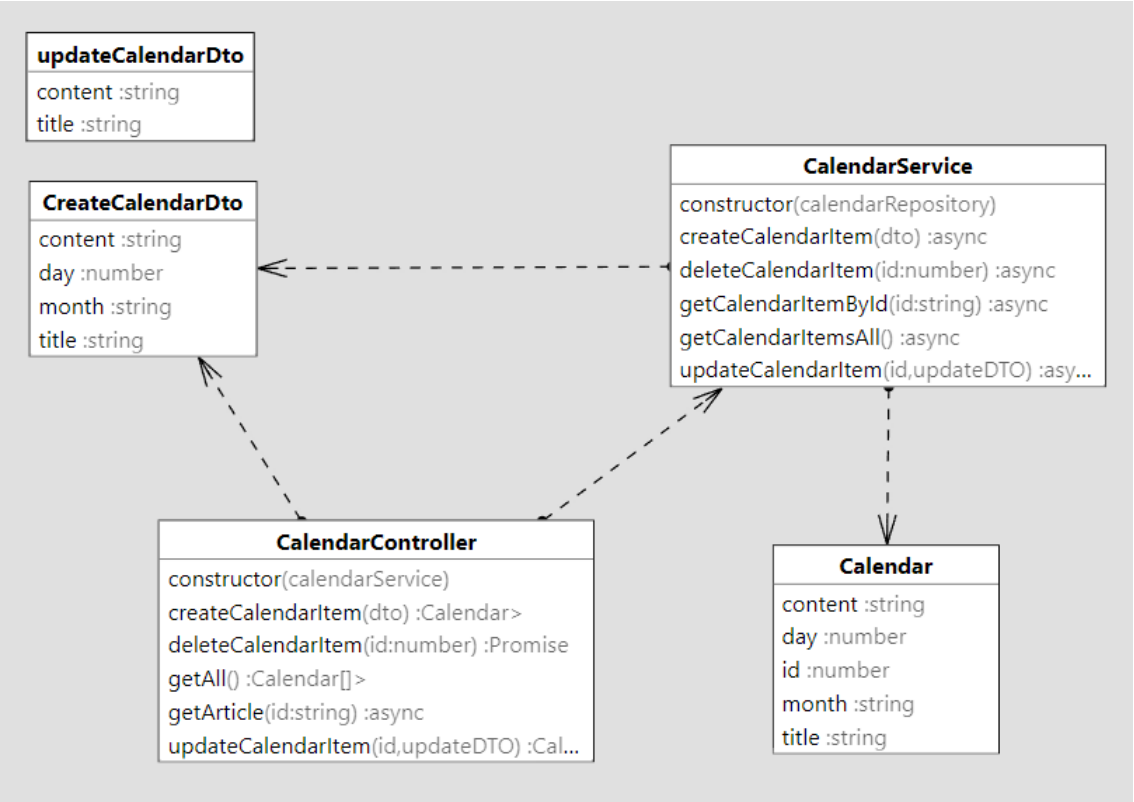
ДОДАТОК Б. ЛІСТИНГ КЛАСІВ МОДУЛІВ



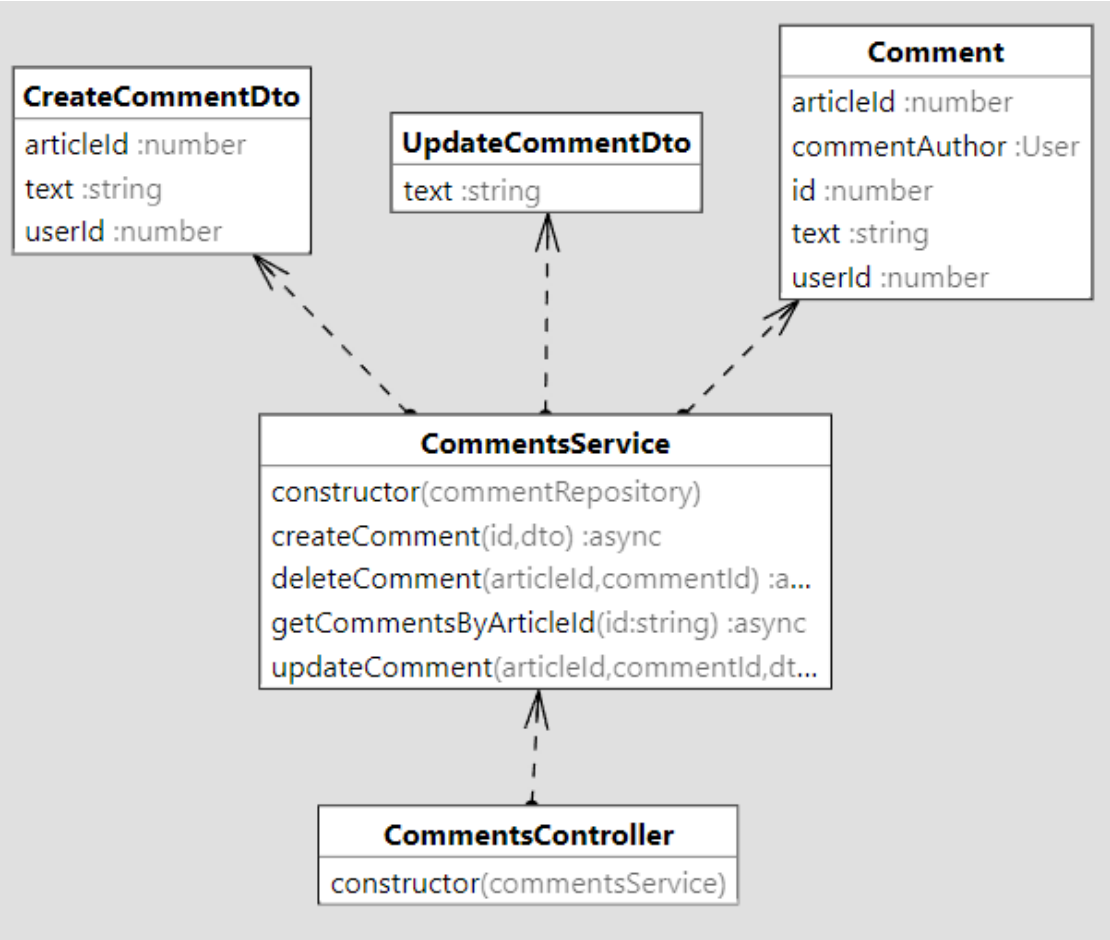
Діаграма класів модулю Users



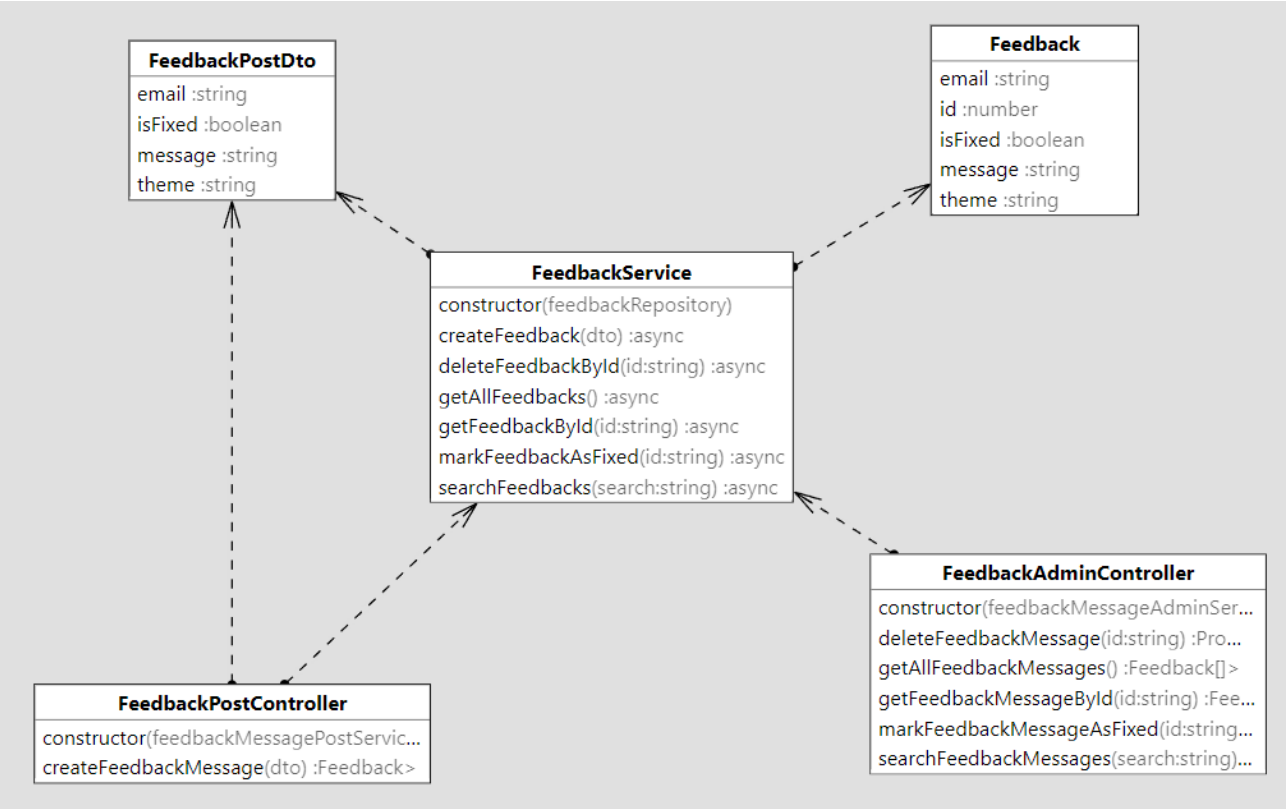
Діаграма класів модулю Roles



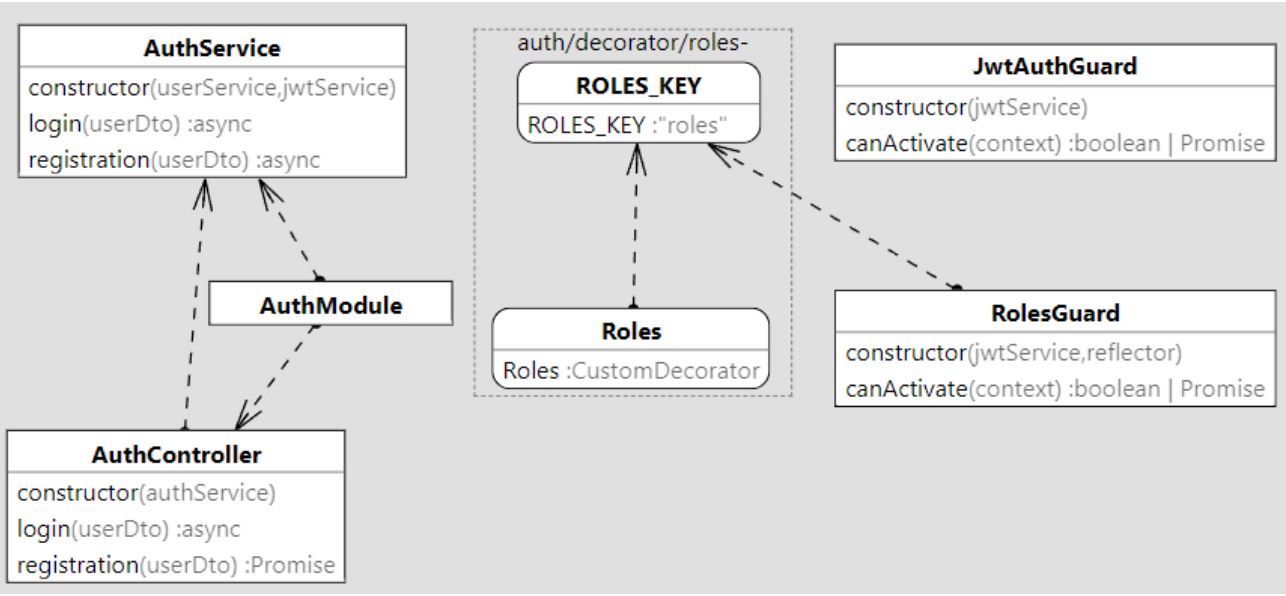
Діаграма класів Місяцеліку



Діаграма класів модулю Comments



Діаграма класів модулю Feedback



Діаграма класів модулю авторизації та аутентифікації