

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ BACK-END ЧАСТИНИ ІТ-ФОРУМУ

(назва теми)

SOFTWARE IMPLEMENTATION OF THE BACK-END PART OF AN
OF AN IT-FORUM

Виконав(ла): студент(ка) 4 курсу, групи КНТ-229

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

СТУПАКОВ К. В.

(ПРИЗВИЩЕ та ініціали)

Керівник ФЕДОРОНЧАК Т. В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТЕРЕЩЕНКО Е. В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

Сергій СУББОТІН

«_____» _____ 2023 року

**З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

СТУПАКОВА Кирила Вадимовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація back-end частини ІТ-форуму. Software Implementation of the Back-End Part of an IT-forum

керівник проєкту (роботи) к.т.н., доцент, ФЕДОРОНЧАК Тетяна Василівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 04 » квітня 20 23 року № 87

2. Строк подання студентом проєкту (роботи) 29 травня 2023 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Характеристика об'єкта проєктування. 2. Вибір і обґрунтування структури системи. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ФЕДОРОНЧАК Т. В., доцент		
Нормоконтроль	ДЕЙНЕГА Л. Ю., ст. викл.		

7. Дата видачі завдання «04» квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4, 5
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Кирило СТУПАКОВ
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна ФЕДОРОНЧАК
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 149 с., 42 рис., 8 табл., 3 дод., 15 джерел.

ІТ-ФОРУМ, САЙТ, ІНТЕРНЕТ, PHP, MVC, MySQL, OPEN SERVER, ПОШУК ІНФОРМАЦІЇ, LARAVEL, СЕРВЕР, МАРШРУТИЗАЦІЯ, ВАЛІДАЦІЯ ДАНИХ.

Об'єкт дослідження – Інтернет-форум, присвячений ІТ.

Предмет дослідження – backend частина програмного забезпечення форуму для ІТ-спеціалістів.

Мета роботи – розробка серверної частини програмного продукту, що реалізує базовий функціонал ІТ-форуму, взаємодіє з базою даних та надає перелік ендпоінтів для клієнтської частини.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, шаблон проєктування архітектури MVC, фреймворк Laravel, мова програмування PHP, база даних MySQL, пакетний менеджер Composer, програмне забезпечення локального сервера Open Server.

Результати. Розроблено кросплатформну серверну частину програмного забезпечення ІТ-форуму, яка надає можливість організації спілкування користувачів для обговорення професійних питань та пошуку вже існуючої інформації.

Висновки. Створено серверне програмне забезпечення для зручної взаємодії ІТ-спеціалістів різних галузей, що надає останнім можливість шукати, зберігати, обмінюватися, додавати та редагувати необхідну для них інформацію.

Галузь використання – побудова застосунків для організації ІТ-форумів, які можуть спростити роботу для програмістів, системних адміністраторів, тестувальників та спеціалістів інших ІТ-галузей.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 149 pages, 42 figures, 8 tables, 3 appendixes, 15 sources.

IT-FORUM, SITE, INTERNET, PHP, MVC, MySQL, OPEN SERVER, INFORMATION SEARCH, LARAVEL, SERVER, ROUTING, DATA VALIDATION.

The object of the research is an Internet forum dedicated to IT.

The subject of the research is the backend part of the forum software for IT specialists.

The purpose of the work is to develop the server part of the software product that implements the basic functionality of the IT forum, interacts with the database and is able to respond to requests from the client side.

Materials, methods and technical means: object-oriented programming, MVC architecture design pattern, Laravel framework, PHP programming language, MySQL database, Composer package manager, Open Server local server software.

The results. A cross-platform server part of the IT forum software has been developed, which provides an opportunity to quickly search for information and provides for its constant accumulation.

Conclusions. Server software has been created for the convenient interaction of IT specialists of various industries, which provides the latter with the opportunity to search, store, exchange, add and edit the information they need.

The field of use is the construction of applications for the organization of IT forums, which can simplify the work for programmers, system administrators, testers and specialists of other IT industries.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	9
Вступ.....	10
1 Характеристика об'єкта проектування	12
1.1 Поняття Інтернет-форуму	12
1.2 Структура Інтернет-форуму.....	14
1.2.1 Архітектура найпростішого форуму	14
1.2.2 Архітектура форуму з категоріями	15
1.2.3 Деревоподібна архітектура форуму	16
1.3 Функціональні можливості форуму	18
1.3.1 Базовий функціонал	18
1.3.2 Розширений функціонал.....	18
1.3.3 Засоби ідентифікації користувачів.....	19
1.3.4 Функціональні можливості ролей у межах форуму	20
1.4 Існуючі аналоги ІТ-форумів.....	21
1.4.1 Stack Overflow.....	21
1.4.2 Sololearn.....	23
1.5 Висновки за розділом	25
2 Вибір і обґрунтування структури системи	26
2.1 Вибір мови програмування	26
2.2 Вибір фреймворку	29
2.3 Вибір бази даних	33
2.4 Вибір системи керування версіями	38
2.5 Функціональна схема роботи ІТ-форуму	40
2.5.1 Функціональна схема форуму для ролі користувача	40
2.5.2 Функціональна схема форуму для ролі гостя	42
2.5.3 Функціональна схема форуму для ролі адміністратора.....	44
2.6 Висновки за розділом	45
3 Основні рішення реалізації компонентів системи.....	46

3.1 Основні відомості про розробку	46
3.2 Система маршрутизації	48
3.3 Структура бази даних	51
3.4 Розроблені класи контролерів.....	54
3.4.1 Клас BookmarksController.....	54
3.4.2 Клас CategoriesController	56
3.4.3 Клас CommentsController.....	57
3.4.4 Клас LikesController	59
3.4.5 Клас PostsController.....	60
3.4.6 Клас UsersController.....	62
3.5 Розроблені класи моделей	63
3.5.1 Клас Category	63
3.5.2 Клас Comment.....	65
3.5.3 Клас Like	66
3.5.4 Клас Post.....	67
3.5.5 Клас User	69
3.6 Розроблені класи валідації	71
3.6.1 Клас CategoriesRequest.....	71
3.6.2 Клас PostsRequest	72
3.6.3 Клас UsersRequest.....	72
3.7 Висновки за розділом	73
4 Керівництво програміста.....	74
4.1 Розгортання локального сервера розробки	74
4.2 Інсталяція вебфоруму	76
4.3 Розгортання структури бази даних.....	77
4.4 Заповнення бази даних тестовими даними.....	78
4.5 Висновки за розділом	80
5 Керівництво користувача	82
5.1 Загальні відомості	82
5.2 Запит на отримання списку всіх публікацій.....	82

5.3 Запит на отримання даних конкретної публікації	83
5.4 Запит пошуку публікацій	84
5.5 Запит на створення публікацій	85
5.6 Запит на отримання списку всіх категорій	86
5.7 Запит на отримання списку всіх користувачів.....	87
5.8 Висновки за розділом	88
Висновки	89
Перелік джерел посилання	90
Додаток А Технічне завдання	92
Додаток Б Текст програми	95
Додаток В Слайди презентації.....	142

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
MVC	– Model-View-Controller,
PHP	– Hypertext Preprocessor;
SQL	– Structured Query Language;
IT	– Information Technologies;
CRUD	– Create, Read, Update and Delete;
HTML	– HyperText Markup Language;
XML	– eXtensible Markup Language;
JSON	– JavaScript Object Notation;
REST	– Representational State Transfer;
IDE	– Integrated Development Environment;
URL	– Uniform Resource Locator;
UBB	– Ultimate Bulletin Board;
WWW	– World Wide Web;
ORM	– Object Relational Mapping
CAPTCHA	– Completely Automated Public Turing test to tell Computers and Humans Apart;
СМ	– система маршрутизації;
БД	– база даних;
СУБД	– система управління базами даних;
ПЗ	– програмне забезпечення.

ВСТУП

На сьогодні важко уявити певну сферу людської діяльності, яку б оминув стрімкий розвиток ІТ-індустрії. У медицині з'являються нові, більш точні електронні прилади, застосовується штучний інтелект для проведення операцій. На виробництві починають застосовувати складніші системи автоматизації людської праці. Такі звичайні речі, як доставлення їжі, виклик таксі, бронювання квитків або онлайн-замовлення якогось товару вже давно можна зробити в Інтернеті, що стимулює надавачів подібних послуг створювати комфортні сайти з осучасненим дизайном, прогресивних ботів, що автоматично підкажуть клієнту найвигіднішу пропозицію.

Все це, в свою чергу, вимагає певного обслуговування. Створені застосунки та розроблені системи повинні пройти тестування, бути належним чином захищені від зламу та кимось адмініструватись. Як результат, разом з ІТ-індустрією зросла роль найрізноманітніших ІТ-спеціалістів, починаючи від програмістів та закінчуючи системними адміністраторами, тестувальниками, професіоналами кібербезпеки.

Разом з тим з'являються нові засоби та технології для створення відповідного програмного забезпечення. Прикладом можуть бути мови програмування, кожна з яких має свій власний вектор застосувань та певні переваги й недоліки у порівнянні з іншими мовами. Розвиваються й деякі більш специфічні технології, наприклад HTML – мова розмітки, або SQL – мова запитів до реляційних баз даних. І кількість та різноманіття таких технологій тільки зростає, особливо беручи до уваги великий попит на них зі сторони багатьох сфер людської діяльності.

Оскільки інформація, що необхідна кваліфікованому ІТ-спеціалісту, постійно збільшується, зростає популярність відповідних Інтернет-ресурсів. Це документації, певні електронні посібники та книжки, форуми та тематичні сайти, відеоролики та вже створені шаблонні приклади. Але, незважаючи на різноманіття цих ресурсів, час від часу виникають певні труднощі з пошуком

інформації, її неактуальність або навіть інколи невірність. Все це значно ускладнює роботу ІТ-фахівцям та будь-які процеси, пов'язані з програмним забезпеченням, будь-то розробка, тестування чи його обслуговування.

У даній дипломній роботі було прийнято рішення створити серверну частину програмного забезпечення такого Інтернет-ресурсу, який би зміг найкращим чином задовольнити потреби ІТ-спеціалістів. В якості виду ресурсу було обрано форум, як місце, де структуровано може бути розміщено колосальний обсяг інформації та надано користувачам відповідні інструменти для керування нею. Окрім цього форум надає можливість комунікації користувачів, що стало ще одним аргументом у користь вибору саме такого виду інформаційного ресурсу.

Серверну частину ІТ-форуму буде розроблено мовою PHP з використанням фреймворку Laravel. У ролі БД обрано MySQL. Даний стек технологій надає широкий набір функціоналу для розробки і є дуже надійним, зважаючи на велику кількість створених за ним популярних застосунків [1].

1 ХАРАКТЕРИСТИКА ОБ'ЄКТА ПРОЄКТУВАННЯ

1.1 Поняття Інтернет-форуму

Інтернет-форум – це вид інформаційного ресурсу, розташованого у мережі Інтернет, що однією з основних функцій несе в собі функцію комунікації.

Інтернет-ресурс або вебресурс – це узагальнювальна назва інформаційних ресурсів, які доступні у мережі WWW та можуть бути відкриті у браузері.

Браузер – програмний засіб, що надає базовий функціонал для роботи з Інтернет-ресурсами та Інтернетом в цілому. Основними функціями такого програмного забезпечення є пошук та відображення інформації. З точки зору клієнт-серверної архітектури, браузер є оболонкою, яка виконує роль клієнтської частини [2].

Найперші Інтернет-форуми почали з'являтися у середині 1990-х років, майже одразу з появою та початком розвитку Інтернету та вебтехнологій. Вони склалися з примітивних HTML-сторінок та мали максимально простий дизайн. У 1997 році було створено Ultimate Bulletin Board – перше програмне забезпечення, що дозволяло дуже швидко розробляти власний форум та масштабувалось у залежності від потреб користувачів. За рахунок своєї унікальності, UBВ дуже швидко став популярним та зробив значний вклад у розвиток вебфорумів. Через деякий час були написані схожі додатки, націлені на спрощення процесу розробки форумів, та вдосконалені розширеним функціоналом (наприклад phpBB та vBulletin) [3].

Основна ідея Інтернет-форуму полягає у взаємодії користувачів за допомогою так званих розділів форуму – тем, постів або публікацій. Користувачі класичного форуму можуть створювати власні теми, переглядати та коментувати теми інших. Темі для обговорень можуть бути найрізноманітніші: починаючи з вільного спілкування та закінчуючи чимось вузькоспрямованим.

Будь-який вебфорум є різновидом віртуальної спільноти і має свою систему ролей користувачів. Основною є роль звичайного користувача, який зареєструвався та авторизувався на форумі. Такий тип користувачів є найчисельнішим та має стандартний набір прав та доступів. Другим типом користувачів є адміністратори форуму – особи, що виконують функції керування форумом, маючи при цьому найвищі рівні доступу та інколи – окремі панелі управління сайтом. Для слідкування за дотриманням внутрішніх правил форуму користувачами існує роль модератора, хоча інколи цим займаються адміністратори. У разі, якщо форум не є закритим, тобто для перегляду наявних публікацій не треба реєструватись, слід виділити ще один тип користувачів – це гості. Дана роль немає майже ніяких дозволів окрім перегляду інформаційного наповнення, контенту [4].

У якості внутрішніх правил Інтернет-форуму найчастіше бувають:

- заборона або часткове обмеження у виборі тем для обговорень;
- заборона або недоцільність виходу за межі теми під час її обговорення;
- заборона створення цілеспрямованих багаторазових публікацій або коментарів з метою деструктуризації або забруднення інформаційного простору форуму;
- заборона на використання ненормативної лексики;
- заборона на публікацію персональної інформації інших користувачів без їх дозволу.

За порушення правил можуть бути застосовані певні каральні міри з боку адміністраторів та модераторів, при чому адміністрації різних форумів можуть встановлювати різні покарання. В залежності від порушення та обставин, адміністратори здатні:

- заблокувати користувачу доступ до певних функцій, інформації або форуму в цілому на деякий час (або назавжди);
- видати попередження користувачу;

- видалити сліди діяльності користувача, які призвели до порушення правил;
- власноруч або програмно виправити дрібні порушення, пов'язані, наприклад, з публікацією певної інформації шляхом її редагування;
- ігнорувати зовсім несуттєві порушення.

Не зважаючи на те, що на сьогодні форуми поступово витісняються месенджерами та соціальними мережами, вони досі актуальні для певних груп користувачів. Основною перевагою вебфорумів є краща структуризація контенту, акцентування уваги на більш конкретизованих речах та звужені теми для обговорень у той час, коли соціальні мережі є менш структурні з точки зору розміщення та взаємозв'язків різних кластерів інформації. Другою суттєвою перевагою форумів є повнота розміщеної інформації, її цілеспрямована модерація, корегування та перевірка, що важко сказати про соціальні мережі в цілому [5].

1.2 Структура Інтернет-форуму

1.2.1 Архітектура найпростішого форуму

Інтернет-форум – це перш за все місце, де зберігається інформація, то ж воно обов'язково повинно мати певну структуру. Якість архітектури форуму напряму впливає на структуризацію інформації. Чим краще структурована інформація, тим зручніше користуватися самим форумом навіть тим користувачам, які тільки почали з ним знайомитися. Отже, можна зробити висновок, що рівень архітектури Інтернет-форуму визначає його популярність та актуальність у моменті вибору інформаційного ресурсу ІТ-спеціалістом.

Найпростіший форум складається з простих публікацій, які не мають власної ієрархії та знаходяться на одному структурному рівню. Користувачі такого форуму можуть швидко створювати публікації, не переймаючись їх структуризацією, що є однією з двох основних переваг даної архітектури.

Друга перевага полягає у простій реалізації, з точки зору створення програмного забезпечення. Головним недоліком є погано структурована інформація та, як наслідок, ускладнений пошук потрібного для користувачів.

Архітектура найпростішого виду Інтернет-форуму зображена на рис. 1.1.



Рисунок 1.1 – Архітектура найпростішого Інтернет-форуму

Це базова архітектура Інтернет-форуму, яка може бути розширена. У такому випадку, окрім тем чи постів, з'являється нове поняття – категорія [6].

1.2.2 Архітектура форуму з категоріями

Публікації, що схожі між собою за змістом, сенсом або тематикою, можуть бути об'єднані у категорію з метою покращення структуризації інформації на форумі. Таким чином, при наявності величезного обсягу публікацій, користувачам достатньо обрати необхідну категорію і вже там знайти відповідну публікацію, не фільтруючи увесь контент форуму. Архітектура Інтернет-форуму з категоріями зображена на рис. 1.2.

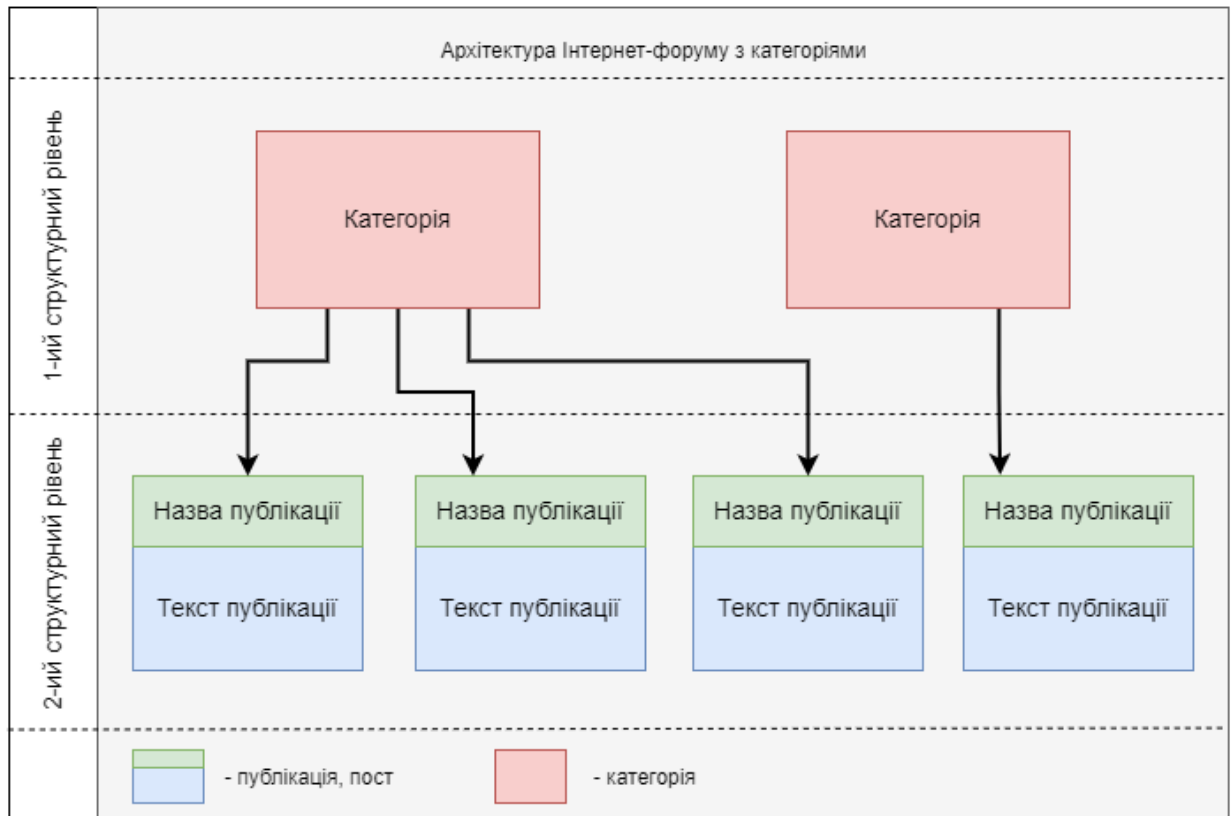


Рисунок 1.2 – Архітектура Інтернет-форуму з категоріями

Такий підхід є ефективним, але не гнучким, адже в певний момент кількість категорій збільшиться до величезних масштабів, що утворить проблему, подібну недоліку найпростішої архітектури Інтернет-форуму. Але, порівнюючи дану архітектуру з її попередником, можна з точністю сказати, що у загальному вигляді інформація буде краще структурованою при даній архітектурі за умови однакових обсягів інформації [6].

1.2.3 Деревоподібна архітектура форуму

Щоб уникнути попередньої проблеми та надати гнучкості структурі форуму, були створені підкатегорії. Підкатегорія – це одиниця, що структурно нижча за категорію та не може існувати без неї. Така модель формує деревоподібну архітектуру, де корневими елементами є категорії, «гілками» - підкатегорії, а «листочками», або кінцевими елементами – самі

публікації, пости. Це значно кращий варіант у порівнянні з попередніми двома, адже він майже повністю вирішує проблему накопичення та необхідності структуризації інформації.

Архітектура Інтернет-форуму з категоріями та підкатегоріями зображена на рис. 1.3.

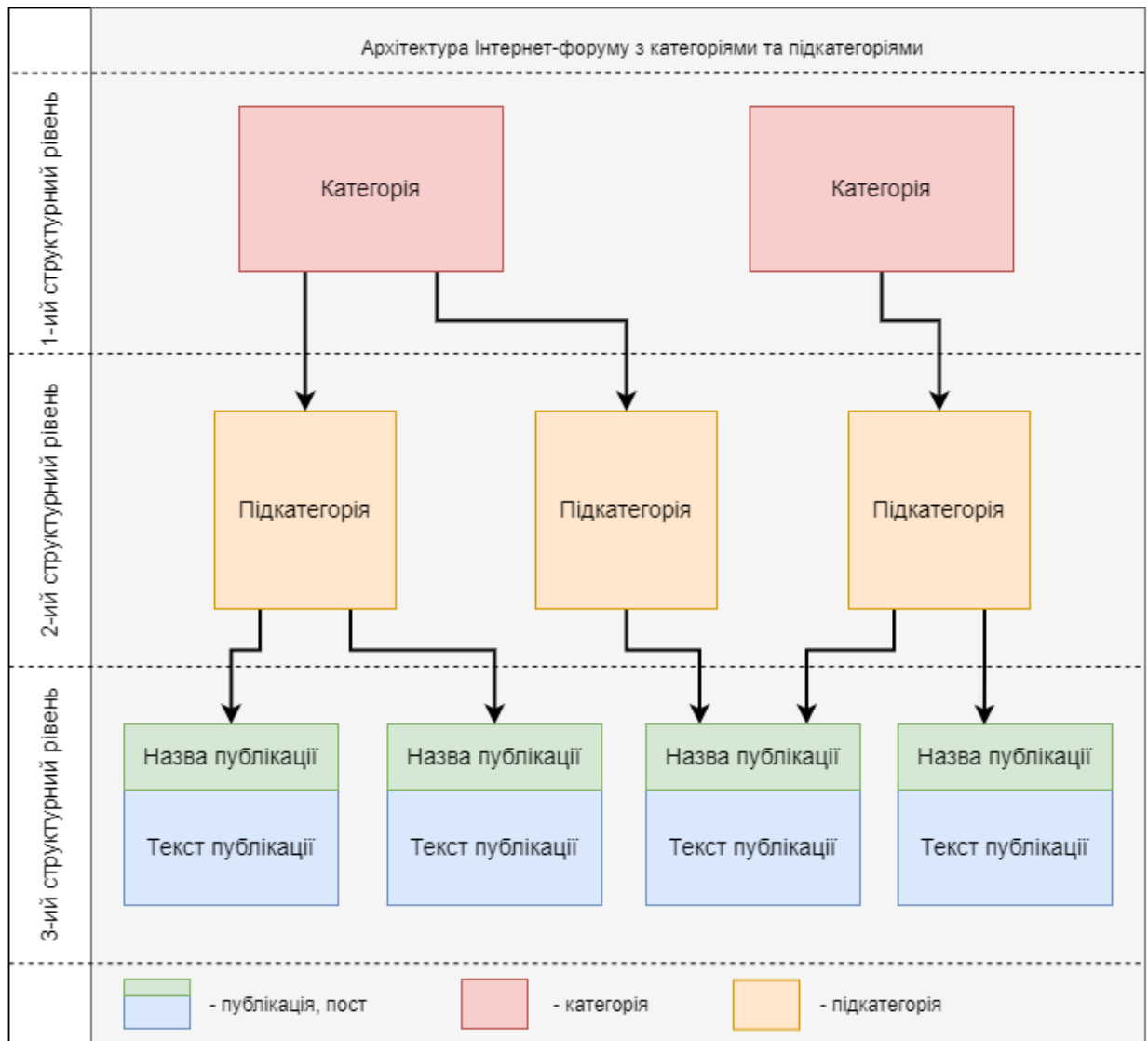


Рисунок 1.3 – Архітектура Інтернет-форуму з категоріями та підкатегоріями

Невеликим недоліком цієї архітектури є значно складніша програмна реалізація, адже слід враховувати розширення схеми бази даних.

Окремо слід зауважити наявність можливості надання посту підкатегорії, що належать до різних категорій. Це значно підвищує структурність контенту форуму та пришвидшує пошук необхідного [6]-[7].

1.3 Функціональні можливості форуму

1.3.1 Базовий функціонал

Центральними поняттями об'єкта дослідження є «публікації», «теми», «категорії» або «розділи», «обговорення» та «коментарі», адже жоден форум не може існувати без цього. Таким чином, основна функціональність декларується саме цими поняттями.

Користувачі форуму найчастіше мають наступні базові можливості:

- створювати нові публікації, коментарі, розділи;
- редагувати теми, коментарі та категорії, що вже існують;
- переглядати створені інформаційні об'єкти;
- видаляти публікації, коментарі, розділи.

Тобто основою роботи з інформацією є CRUD-принцип, який обов'язково має бути реалізований для базових потреб користувачів. Також необхідно зауважити, що даний принцип повинен бути виконаний стосовно усіх (або більшості) основних сутностей, формуючих архітектуру форуму. Так, наприклад, форум може мати функцію створення коментарів, не маючи можливості їх видалення або редагування, у той час, коли до публікацій недоречно застосовувати такі обмеження [7].

1.3.2 Розширений функціонал

Під розширеним функціоналом найчастіше мають на увазі модифікації базових CRUD-операцій. Операції створення, редагування, які

зазвичай застосовуються до одиничного екземпляру даних, можуть набувати множинного застосування. Наприклад:

- множинне створення даватиме змогу швидкого опублікування великої кількості постів;
- множинне видалення дозволяє вмиль прибрати всі коментарі до якоїсь теми, або видалити усі збережені публікації;
- множинне редагування є невід’ємною складовою функцій автокорекції даних.

Операція перегляду розширюється певними умовами розділення й відбору даних, фільтрами та сортуванням. На базі даної модифікації створюються:

- внутрішні пошукові системи форуму;
- групи фільтрів та правил сортувань інформації;
- елементи контролю пагінації, що ділять величезні масиви даних на рівномірні шматочки та запобігають їх одночасному відправленню на сторону клієнта (наприклад вивід по 50 постів на сторінці, коли у БД їх кількість може сягати декільком тисячам).

Окрім зазначених, форум може мати менш помітні функції, що працюють в автоматичному або напіваавтоматичному режимі:

- функції автопідрахунку статистичних даних, кількості коментарів, вподобань, публікацій тощо;
- функції автокорекції даних, автовиявлені вживання ненормативної лексики користувачами або автоблокування порушників правил форуму;
- функції автопідказок та авто виправлень;
- автогенерація рекомендацій контенту для перегляду [8].

1.3.3 Засоби ідентифікації користувачів

У багатьох форумів на сьогодні є можливість ідентифікації користувачів. Ідентифікація користувача – це процес розпізнавання певного

користувача серед групи інших за певним критерієм чи ознакою. В основному це застосовують з метою створення системи ролей та отриманні переваг, що вона надає.

Також інколи ідентифікацію використовують для слідкування за діяльністю користувача, щоб визначити його інтереси, передбачити можливі пошукові запити та запропонувати відповідний контент.

Для повноцінної та коректної ідентифікації система повинна мати:

- модуль реєстрації нових користувачів;
- модуль авторизації вже зареєстрованих користувачів;
- функції валідації даних,
- методи захисту персональних даних, що використовуються для входу в систему (наприклад пароль, пошта або номер телефону).

З метою відбору своєї цільової аудиторії, форум, як і будь-який вебресурс, може ускладнювати етапи реєстрації, розділяючи її на підпункти, додаючи додаткові поля для збору даних та ставлячи питання користувачу у процесі.

Це дозволяє відсіяти потенційно-небажаних користувачів для форуму, які можуть нести шкоду для його порядку або аудиторії.

Під час реєстрації або авторизації, користувач також може зустріти CAPTCHA. Дана технологія дозволяє відрізнити людину від комп'ютера. Основною ціллю використання цього тесту є захист вебресурсу від діяльності стороннього програмного забезпечення [8].

1.3.4 Функціональні можливості ролей у межах форуму

Більшість сучасних форумів мають систему ролей, в якій кожен користувач має власний набір допусків до того чи іншого функціоналу вебресурсу.

Порівняльна характеристика ролей за доступом до функціоналу наведена у табл. 1.1.

Таблиця 1.1 – Функціональні можливості ролей

Можливості	Роль				
	Гість	Користувач	Заблокований користувач	Модератор	Адмін
Операції з власним акаунтом	-	так	ні	так	так
Керування іншими акаунтами	-	ні	ні	частково	так
Перегляд контенту	так	так	обмежено	так	так
Створення контенту	ні	так	ні	так	так
Редагування контенту	ні	тільки власний	ні	так	так
Доступ до панелі керування	ні	ні	ні	частково	так
Доступ до додаткових інструментів	ні	так	ні або обмежено	так	так

Така система розгалужень робить форум гнучким та ще більш структурованим, адже ролі вносять розбалансування між користувачами та, як наслідок, утворюють ієрархію. Даний метод є широковживаним, бо дає змогу підтримувати порядок на форумі та відкриває можливості тонкого керування [8].

1.4 Існуючі аналоги ІТ-форумів

1.4.1 Stack Overflow

Одним з найяскравіших існуючих аналогів форумів, присвячених ІТ, є stack overflow. Найчастіше його використовують саме для запитань, описуючи свою проблему та очікуючи підказки від користувачів, або просто шукають відповіді у схожих публікаціях. Даний вебресурс має архітектуру 2-

го типу, бо має окрім постів ще й категорії, але немає підкатегорій. На форумі є можливість реєстрації та входу до акаунту, що свідчить про наявність системи ролей.

На рис. 1.4 зображено форум зі списком публікацій.

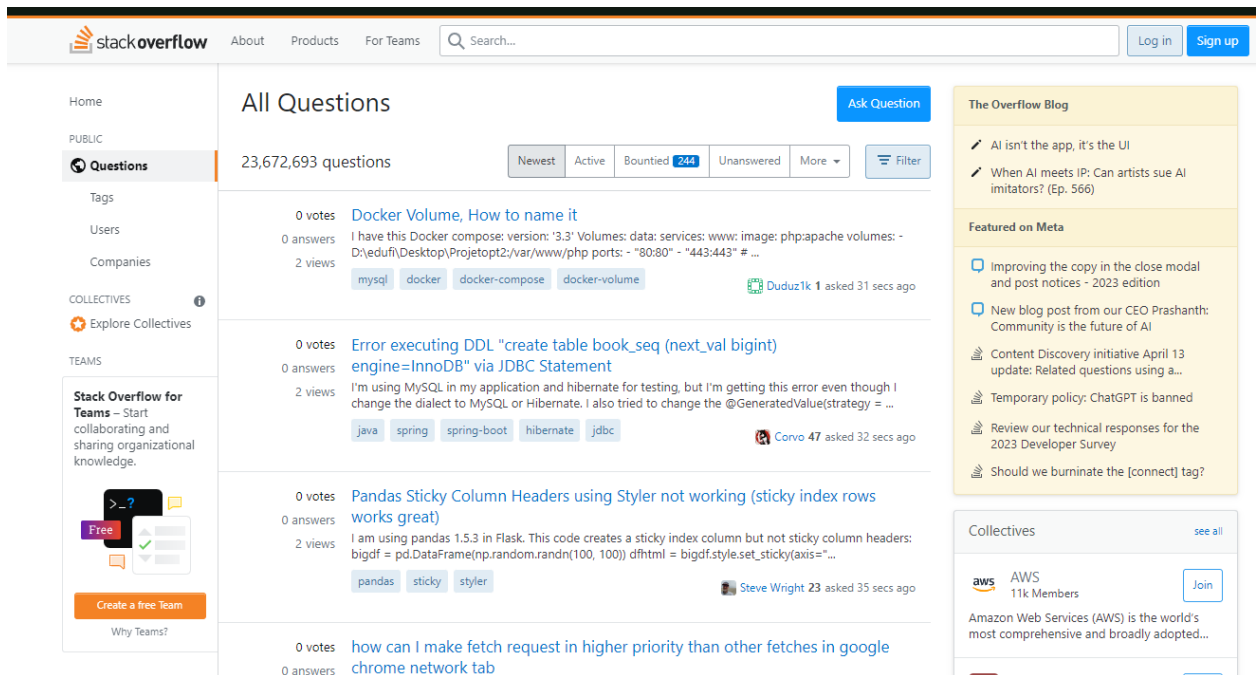


Рисунок 1.4 – Список публікацій форуму stack overflow

Перевагами форуму є:

- популярність та велика кількість активних користувачів;
- колосальність обсягу корисної інформації;
- наявність системи оцінювання якості тих порад, що пишуть користувачі у коментарях;
- широкий функціонал сайту.

До недоліків даного вебфоруму можна віднести порівняно слабку архітектуру, якщо брати до уваги кількість публікацій. Попри присутність категорій, їх просто не вистачає, щоб розв'язати задачу підвищення структуризації.

На рис. 1.5, на елементі пагінації сторінки з категоріями, можна побачити суть описаного недоліку.

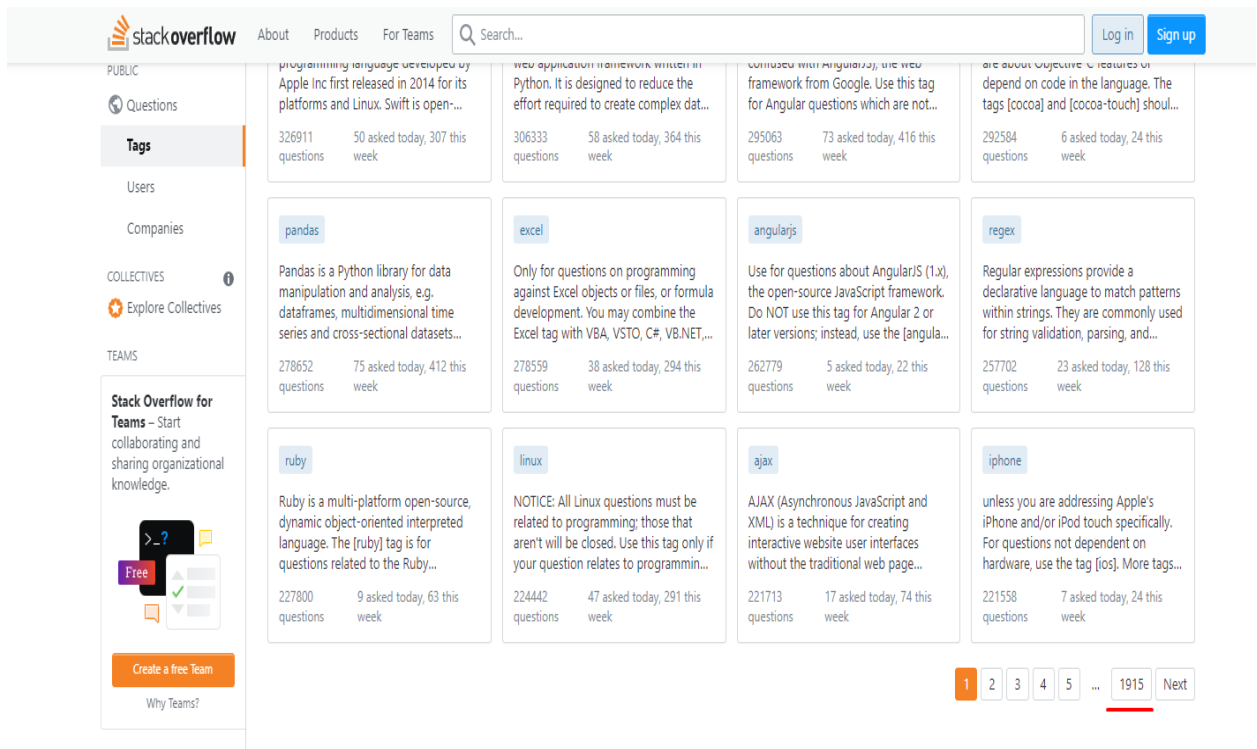


Рисунок 1.5 – Кількість сторінок з категоріями на форумі stack overflow

Враховуючи те, що на одній сторінці одночасно відображається 36 категорій, а існуючих сторінок вже 1915, можна зробити висновок про наявність, приблизно, 68 940 категорій.

Даний нюанс є суттєвим, бо впливає на зручність пошуку необхідної інформації [9].

1.4.2 Sololearn

В якості ще одного прикладу можна навести менш популярний ІТ-форум sololearn. Тут також можна задавати питання, шляхом їх публікації та присвоєнню їм категорій. Загальна кількість користувачів невелика, як і кількість публікацій.

Основною відмінною рисою форуму є вбудований онлайн-компілятор коду, що підтримує 17 різних мов програмування та розмітки.

На рис. 1.6 зображено загальний вигляд форуму.

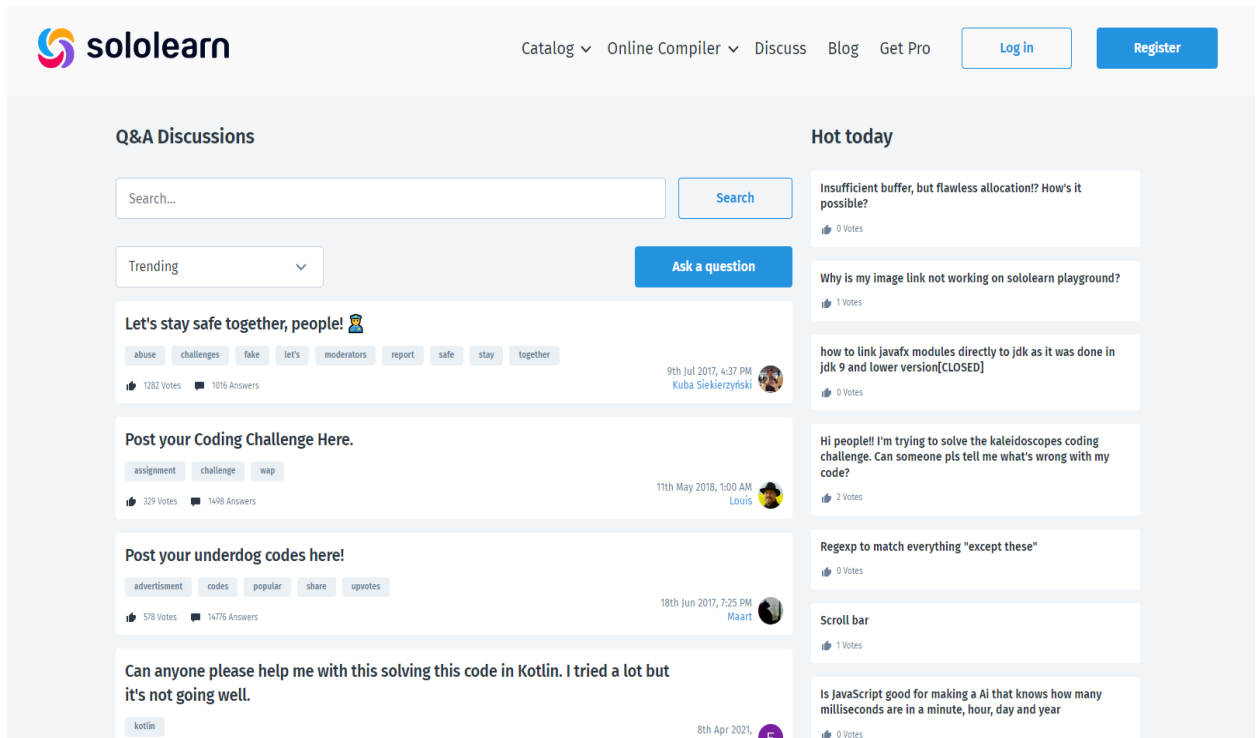


Рисунок 1.6 – Загальний вигляд форуму sololearn

Основними перевагами форуму є:

- мінімалізм, що дозволяє швидко орієнтуватись на сторінках;
- наявність потужного онлайн-компілятора коду, що дає змогу перевіряти код на помилки, не виходячи з сайту;
- наявність системи оцінювання коментарів.

До недоліків слід віднести:

- невелику кількість користувачів та активності на форумі загалом;
- складність пошуку необхідної інформації;
- маленькі обсяги створених публікацій;
- надвелику кількість та розсіяність категорій, що ускладнює роботу з форумом;
- відсутність розширеної пошукової системи, є лише звичайний пошук з одним фільтром [10].

1.5 Висновки за розділом

Проведено дослідження предметної області, окреслено базові поняття, що стосуються об'єкта дослідження. Визначено основну суть об'єкта дослідження, його провідну ідею та цілі.

Наведено детальний опис характеристик Інтернет-форумів. Описано принципи взаємодії користувачів форуму один з одним та вплив цих процесів на сам форум.

Наведено класифікацію вебфорумів на основі параметрів структуризації інформації та функціональної гнучкості. За значеннями параметрів, форуми були розбиті на групи. До кожної групи форумів наведено їх загальну структуру, визначено архітектурні та функціональні відмінності. Наведено ряд переваг та недоліків кожної архітектури.

Виконано детальний розбір функціонала об'єкта дослідження шляхом розбиття на основні компоненти та їх подальшого опису. Виділено найосновніші функції форуму та їх можливі модифікатори. Окремо розібрано систему ролей, проведено порівняння їх між собою в контексті функціональних можливостей.

Ґрунтуючись на проведеному дослідженні предметної області та самому об'єкті дослідження, було вирішено створити Інтернет-форум з урахуванням недоліків вже існуючих форумів, роблячи акцент на якісній структуризації інформації та відповідній архітектурі. З метою досягнення гнучкості функціонала, до форуму буде додано систему ролей та відповідні програмні модулі.

2 ВИБІР І ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ

2.1 Вибір мови програмування

Сучасна розробка будь-яких вебдодатків найчастіше відбувається за допомогою однієї з наступних мов: Python, Ruby, PHP, JavaScript, C# та Java. Ці мови є високорівневими та мають спеціальний набір модулів (або повністю орієнтовані) для веброзробки.

Порівняльна характеристика мов наведена у табл. 2.1.

Таблиця 2.1 – Порівняльна характеристика мов для веброзробки

№	Критерій порівняння	PHP	Python	C#	Java
1	2	3	4	5	6
1	Площина застосування	Розробка вебдодатків	Універсальна. Даною мовою може бути створено будь-що	Веброзробка, розробка ігор	Мобільна розробка, розробка ігор
2	Складність використання	Середня, вимагає певних знань	Незначна, можна дуже легко створювати свої рішення	Складна, вимагає поглиблених знань	Умовно-середня
3	Рівень документації	Детальна у достатньому обсязі, постійно оновлюється	Документація на високому рівні, оновлюється	Порівняно малий обсяг документації	Достатній обсяг
4	Синтаксис	Простий, C-подібний	Дуже простий та зрозумілий	Складний, C-подібний	Середній за складністю
5	Виконання коду	Інтерпретовані мови. Займають більше ресурсів.		Компілюються до проміжного коду. Працюють швидше.	

Продовження таблиці 2.1

1	2	3	4	5	6
6	Простота застосування у WEB-розробці	Дуже легка, цілеспрямовано розв'язує задачі WEB-розробки	Середня, вимагає використання додаткових модулів	Складний, С-подібний	Середній за складністю
7	Швидкість взаємодії з базами даних	Дуже висока		Середня	
8	Складність вивчення мови	Дуже легко вивчається	Вимагає певних зусиль для вивчення	Складна	Складна
9	Складність рефакторингу коду	Умовно проста	В залежності від обсягу проєкту	Складна. Має більше обмежень в зміні коду	Складна. Необхідно компілювати після кожної зміни у коді

ІТ-форум було розроблено за допомогою мови програмування PHP версії 8.0, яка зарекомендувала себе як дуже стабільна та надійна версія. PHP – це скриптова мова програмування, основною задачею якої є динамічне створення HTML-сторінок на серверній стороні.

Скриптова мова – це така мова програмування, яка призначена для написання коротких програм – скриптів, що можуть виконуватись без компіляції на вебсторінці або комп'ютері. Скрипти застосовуються для взаємодії з користувачем, обробкою даних, роботою з базами даних та розробкою вебсторінок.

Як видно з табл. 2.1, двома найкращими мовами для вирішення даної задачі були PHP та Python, але обрано саме PHP в міру наступних причин:

- Python має меншу підтримку стандарту веброботки, що виражається у меншій кількості готових бібліотек та інструментів;
- PHP є більш швидкою, особливо тоді, коли операції стосуються обробки рядків;

- PHP переважає у різноманітні фреймворків для веброзробки;
- Python більш складний для вивчення, через універсальність свого застосування [11].

Основними особливостями PHP є:

- її безкоштовність та відкрите програмне забезпечення, що дозволяє розробникам вносити зміни до вихідного коду за необхідності;
- мова підтримує роботу з багатьма базами даних;
- може використовуватись на більшості операційних систем, таких як Linux, Mac OS та Windows;
- є підтримка об'єктно-орієнтованого програмування, яка відкриває можливості створення більш просунутого та структурованого програмного забезпечення;
- має велику кількість розширень та модулів, що спрощують розробку.

Для спрощення роботи з проєктом, що має залежності, в PHP існує спеціальний пакетний менеджер – Composer. Він дозволяє автоматизувати процес встановлення, оновлення та видалення бібліотек, необхідних для роботи проєкту.

Використання Composer зменшує час на налаштування проєкту та збільшує його стабільність та безпеку. Його часто використовують, особливо тоді, коли розробка йде у команді [11]- [12].

Даною мовою пишуться найрізноманітніші вебдодатки та вебсайти. До найвідоміших вебресурсів, розроблених на PHP можна віднести:

- Facebook – популярна соціальна мережа;
- Wikipedia - електронна енциклопедія;
- WordPress - вебфреймворк для розробки блогів та сайтів, що базуються на контенті;
- Yahoo - пошукова система та провайдер вебпослуг;
- MailChimp - сервіс електронної пошти;

- Slack - вебдодаток, що використовується командами розробників для комунікації;

- Etsy - онлайн-магазин, що продає унікальні товари, ручну роботу та вантажні речі.

Перелічені приклади демонструють актуальність та успішність використання РНР у різноманітних вебпроектах, що мають велику кількість користувачів та обсягів даних.

Як і будь-яка мова, РНР має ряд недоліків:

- складність підтримки через велику кількість наявних у ньому функцій та бібліотек;

- низька швидкість виконання в міру інтерпретованості мови;

- є слабо типізованою мовою, через що, змінні не мають явно визначеного типу, що у свою чергу може привести до помилок та неочікуваних результатів.

Незважаючи на дані недоліки, РНР залишається однією з найпопулярніших мов веброзробки, що має велику кількість різноманітних фреймворків [11]- [12].

2.2 Вибір фреймворку

На сьогодення важко уявити розробку якогось серйозного вебдодатка без використання фреймворку. Фреймворки використовуються для спрощення та прискорення процесу розробки програмного забезпечення. Вони надають готові компоненти та бібліотеки для вирішення стандартних задач, таких як обробка HTTP-запитів, робота з базами даних та валідація форм.

Фреймворки полегшують створення додатків, спрощуючи структурування коду та надаючи шаблони й правила для організації проєкту. Вони також можуть підвищити рівень безпеки програм, адже мають

вбудовані механізми захисту від широко розповсюджених загроз, таких як SQL-ін'єкція.

Використання фреймворків також позитивно впливає на ефективність роботи розробників, бо вони можуть сконцентруватися на рішенні складних задач, не витрачаючи зусилля на написання стандартного коду.

До найпопулярніших PHP-фреймворків входять:

- Laravel – найпопулярніший фреймворк, який використовується для створення вебдодатків високого рівня;
- Symfony – модульний фреймворк, що використовують для написання складних проєктів;
- CodeIgniter – легкий фреймворк, за допомогою якого створюють швидкі вебдодатки;
- Zend Framework – потужний фреймворк для складних задач;
- Slim – легкий фреймворк для створення RESTful вебсервісів;
- Phalcon Micro – мінімалістичний фреймворк, на якому пишуть швидкі вебдодатки;
- FuelPHP – швидкий та гнучкий фреймворк для створення вебсайтів.

Порівняльна характеристика деяких найпопулярніших фреймворків наведена у табл. 2.2.

Таблиця 2.2 – Порівняльна характеристика PHP-фреймворків

№	Критерій порівняння	Laravel	Symfony	Slim	FuelPHP
1	2	3	4	5	6
1	Функціонал	Великий набір функцій та компонентів	Достатній набір різноманітних компонентів	В основному базові функції	Порівняно середній обсяг функцій
2	Синтаксис	Дуже простий	Середній	Порівняно середній	Простий

Продовження таблиці 2.2

1	2	3	4	5	6
3	Основна перевага розробки	Якість, масштабованість	Продуктивність, швидкодія	Простота використання	Висока продуктивність
4	Вбудована СМ	Присутня	Ні	Ні	Присутня
5	Вбудований механізм міграцій БД	Присутній	Присутній	Ні	Присутній
6	Вбудована технологія ORM	Присутня	Ні	Ні	Присутня
7	Рівень документації	Документація на дуже високому рівні	Вище за середній	Достатній	Середній

В якості фреймворку для розробки ІТ-форуму було вирішено обрати Laravel версії 9.19, як найбільш якісний та мультимодульний. Даний фреймворк має ряд переваг:

- чистий та лаконічний синтаксис;
- мінімум налаштувань, що дозволяє зосередитись на розробці програми, а не на налаштуванні сервера або інфраструктури;
- наявність великої кількості розширень та пакетів, що дає можливість розробникам дуже швидко додавати необхідний функціонал;
- MVC-архітектура, за допомогою якої відбувається розділення бізнес-логіки та інтерфейсу користувача;
- підтримка вбудованої аутентифікації, яка підвищує безпеку програми;
- вбудовані засоби тестування, що дають змогу швидко знаходити та виправляти помилки;
- швидкість розробки;

– підтримка багатьох популярних баз даних (наприклад MySQL, PostgreSQL та SQLite), що забезпечує універсальність та гнучкість при розробці вебсайтів.

Одним з головних параметрів при виборі Laravel стала наявність MVC-архітектури, адже саме вона робить проєкт більш структурованим, пришвидшуючи цим розробку та майбутню підтримку продукту [13].

MVC – це підхід до розробки програмного забезпечення, який розділяє додаток на три основні компоненти: модель (Model), вид (View), та контролер (Controller).

Схема взаємодії компонентів MVC представлена на рис. 2.1.

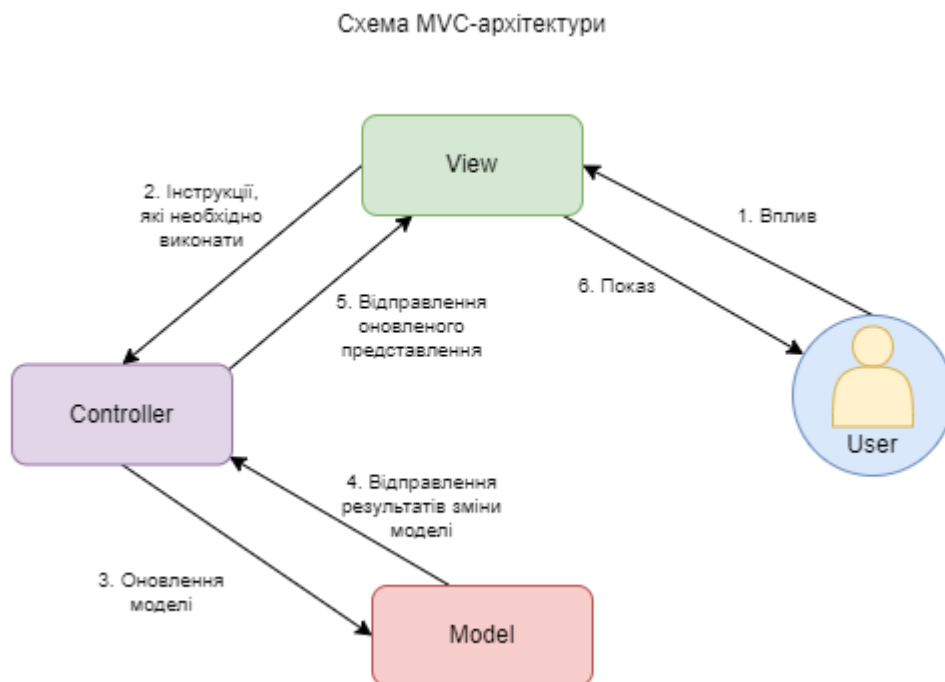


Рисунок 2.1 – Схема MVC-архітектури

Модель являє собою об'єкт або колекцію об'єктів, які зберігають дані та забезпечують доступ до них. Також вона може містити бізнес-логіку додатка.

Вид відповідає за представлення даних користувачу та обробку його дій. Вид може бути у форматі HTML, XML або JSON.

Контролер взаємодіє з моделлю та видом, обробляє запити користувачів та відповідає на них. Він також може містити логіку, яка не пов'язана з моделлю або видом.

Застосування підходу MVC забезпечує легшу розширюваність, тестування та підтримку програмного забезпечення. Кожен компонент може бути змінений або замінений без впливу на інші компоненти.

Як і будь-який фреймворк, Laravel має ряд незначних недоліків:

- складність засвоєння для тих, хто вперше працює з цим фреймворком;
- залежність від пакетного менеджера Composer;
- складність налаштування;
- дуже висока частота оновлень, що може привести до проблем сумісності зі старими розробками [13].

2.3 Вибір бази даних

Будь-який вебресурс, що має значний обсяг даних, буде зберігати їх у базі даних. Даний проєкт по розробці IT-форуму не став винятком. Під час аналізу предметної області було виявлено ряд сутностей, які неодмінно вказують на наявність великої кількості інформації, що робить використання БД обов'язковим.

Laravel підтримує підключення до різних систем управління базами даних. Найчастіше використовують наступні:

- MySQL – найпопулярніша відкрита СУБД, яка підтримує багато різноманітних операцій з БД;
- PostgreSQL – ще одна відкрита СУБД з високим показником надійності та масштабованості;
- SQLite – легка вбудована СУБД, що зберігає бази даних в одному файлі, ідеально підходить для невеликих проєктів або локальної розробки;
- SQL Server – популярна комерційна СУБД від Microsoft;

– Oracle – також комерційна СУБД, яка використовується в багатьох великих корпоративних проєктах.

Порівняльна характеристика деяких найпопулярніших СУБД наведена у таблиці 2.3.

Таблиця 2.3 – Порівняльна характеристика СУБД

№	Критерій порівняння	MySQL	PostgreSQL	SQLite
1	2	3	4	5
1	Функціональність	Повноцінний сервер БД з багатим функціоналом	Сервер БД з достатнім функціоналом	Є тільки мінімум базових функцій
2	Продуктивність за умови невеликих навантажень	Середня	Середня	Висока за рахунок швидкодії
3	Продуктивність за умови високих навантажень	Умовно висока	Умовно висока	Низька
4	Надійність	Не має особливих переваг, середня	Забезпечує більш високий рівень цілісності та стійкості даних	Не має особливих переваг, середня
5	Розмір	Більш важкий, краще підходить для великих проєктів	Умовно середній	Легка БД, краще проявляє себе на невеликих проєктах
6	Підтримка	Має велику активну спільноту та відкритий код	Відкритий код, менша спільнота	Найменший рівень підтримки
7	Легкість застосування	Умовно середня	Вимагає певних зусиль	Дуже легка в застосуванні БД

Для кожної з СУБД, що представлені в таблиці 2.3, Laravel має вбудований драйвер, який дозволяє працювати з будь-якою з них без усіляких проблем та додаткових модулів.

Враховуючи особливості проєкту – потреба у постійній масштабованості та гнучкості IT-форуму, постійному накопиченні великого обсягу даних, було обрано MySQL в якості БД для розробки.

MySQL – це відкрите програмне забезпечення для керування реляційними базами даних. MySQL використовує мову запитів SQL для взаємодії з даними у базі даних. Також дана СУБД підтримує багато функцій, включаючи транзакції, внутрішні ключі, збережені процедури, тригери та індекси. MySQL доступний на різних платформах, у тому числі й на популярних, таких як Windows, Linux та Mac OS X [14].

Реляційна база даних - це тип бази даних, який зберігає дані у вигляді таблиць з рядками та стовпцями. Кожна таблиця містить набір записів (рядків), які містять інформацію про певний об'єкт або сутність, а кожен стовпець представляє різні атрибути або властивості цих об'єктів.

Реляційна база даних забезпечує структуроване зберігання та організацію даних, що дозволяє виконувати ефективні запити та забезпечує збереження цілості даних. Вона забезпечує можливість встановлювати зв'язки між різними таблицями, що дозволяє виконувати запити з використанням даних з різних таблиць.

Основна ідея реляційної бази даних полягає в тому, що дані зберігаються у вигляді таблиць, а зв'язки між таблицями встановлюються за допомогою ключів. Такі ключі можуть бути використані для пошуку, оновлення та видалення даних. Реляційна база даних є одним з найпоширеніших типів баз даних і використовується в багатьох програмах та сервісах, що працюють з великими обсягами даних.

Головними перевагами обраної СУБД є:

- висока швидкість роботи з великим обсягом даних та швидкодія обробки запитів;
- має вбудований механізм забезпечення надійності даних, такий як перевірка цілісності даних та автоматичне відновлення після аварійних ситуацій;

- має простий у використанні інтерфейс та відкритий код, що дозволяє легко налаштовувати та розширювати базу даних;
- підтримує багато різних операційних систем та мов програмування, що дозволяє легко інтегрувати базу даних з різноманітними додатками;
- використовується великою кількістю розробників, що постійно розвивають та підтримують базу даних, а також надають безкоштовну технічну підтримку;
- працює на різному обладнанні, від потужних серверів до звичайних ПК, що дозволяє використовувати базу даних навіть на старих комп'ютерах;
- надає високий рівень захисту бази даних та даних користувачів за допомогою шифрування, аутентифікації та авторизації.

Схема роботи MySQL базується на клієнт-серверній архітектурі, де клієнтські додатки звертаються до сервера MySQL для взаємодії з базою даних [14]. Суть такої взаємодії зображена на рисунку 2.2.

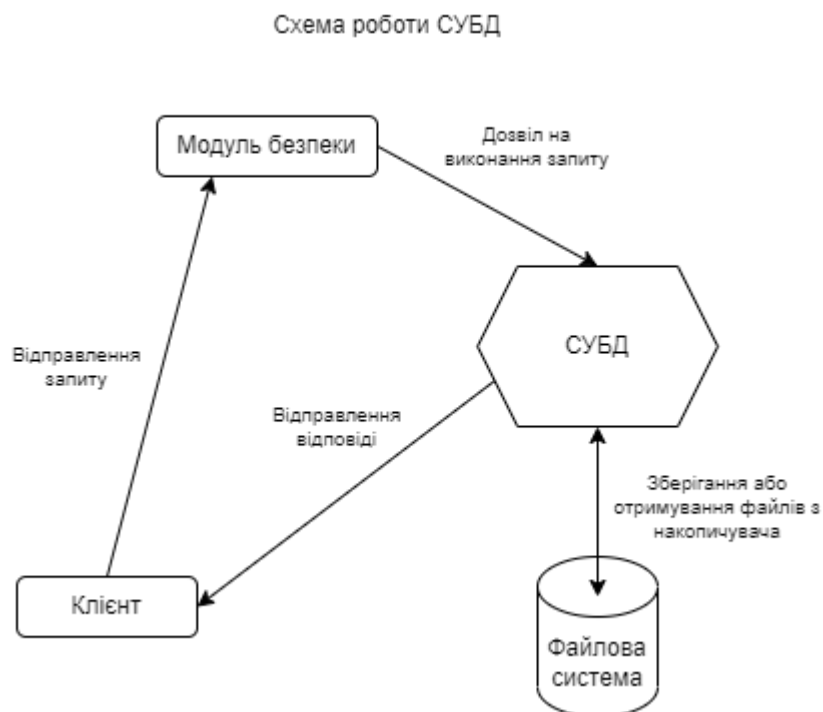


Рисунок 2.2 – Схема роботи СУБД

Сервер MySQL слухає з'єднання на певному порті та очікує запити від клієнтів. Будь-який запит, перш ніж бути виконаним, очікує перевірки прав доступу клієнта, котрий надіслав цей запит.

Перевірка відбувається у модулі безпеки. Клієнтські додатки можуть бути написані на різних мовах програмування (PHP, Java, Python та інші). Для взаємодії з БД, клієнти можуть використовувати SQL-запити [14].

MySQL також підтримує багатопоточну обробку запитів, що дозволяє обробляти багато запитів одночасно з різних клієнтів.

Окрім цього він має підтримку транзакцій, що дозволяє групувати декілька операцій з даними в одну транзакцію для забезпечення атомарності та цілісності даних.

СУБД може взаємодіяти з файловою системою операційної системи, щоб зберігати та отримувати файли з накопичувача. Це може бути корисним у випадках, коли потрібно зберігати файли, такі як зображення, відео або документи, в базі даних, замість зберігання їх окремо на файловій системі. Зазвичай, це виконується за допомогою спеціального типу даних (binary large object), який зберігає вміст файлу у вигляді байтів в базі даних.

Основними недоліками MySQL є:

- низька продуктивність при виконанні комплексних запитів (коли виконуються складні запити, продуктивність MySQL може несуттєво знизитися, особливо при великому обсязі даних);
- може мати проблеми з підтримкою при роботі зі старими версіями;
- багато додаткових функцій для MySQL, таких як резервне копіювання або шифрування, коштують додатково і можуть бути недоступні для менших компаній або розробників [14].

Перелічені недоліки є несуттєвими та повністю компенсуються перевагами обраної СУБД, що робить її ідеальним варіантом для вирішення поточної задачі розробки форуму.

2.4 Вибір системи керування версіями

Оскільки розробка вебфоруму виконувалась каскадно, з поступовим нарощуванням функціонала, було вирішено використовувати систему контролю версій.

Система контролю версій – це програмний інструмент, який дозволяє зберігати та керувати різними версіями файлів та папок, що змінюються з часом. Вона зберігає копії файлів, які зберігаються в спеціальних сховищах (репозиторіях), дозволяючи легко відстежувати, хто та коли вносив зміни в конкретний файл.

Системи контролю версій можуть бути локальними, коли репозиторій знаходиться на локальному комп'ютері, або розподіленими, коли репозиторій розміщується на сервері та доступний для кількох користувачів. До найбільш поширених систем контролю версій відносяться Git, Subversion та Mercurial. Їх порівняльна характеристика наведена у табл. 2.4.

Таблиця 2.4 – Порівняльна характеристика систем контролю версій

№	Критерій порівняння	Git	Subversion	Mercurial
1	2	3	4	5
1	Обов'язковий доступ до сервера для збереження змін	Ні	Так	Ні
2	Рівень засобів гілкування та злиття	Потужний	Порівняно слабкий	Середній
3	Доступний інструментарій та набір плагінів	Дуже широкий	Середній	Мінімальний
4	Гнучкість налаштування	Висока	Середня	Обмежена
5	Складність налаштування	Найскладніша	Легка	Дуже легка
6	Складність використання	Середня	Порівняно низька	Дуже легка

Продовження таблиці 2.4

1	2	3	4	5
7	Краще підходить для	Великих проєктів, над якими працюють у команді	Проєктів, середніх за обсягами	Невеликих та простих проєктів
8	Популярність	Найпопулярніша	Порівняно середня	Менш популярна

Зважаючи на складність розробки та загальний обсяг проєкту, було обрано Git. В основі вибору стали наступні переваги даної системи контролю версій:

- кожен клон репозиторію має повну копію історії проєкту, що дозволяє працювати над проєктом офлайн і без доступу до центрального сервера;
- Git дає можливість вибрати будь-який режим роботи з гілками, який найкраще підходить для проєкту;
- має оптимізований алгоритм збереження даних, тому дуже швидкий;
- легкий в освоєнні та надає багато графічних інтерфейсів для роботи з репозитарієм;
- Git має велику спільноту користувачів, яка постійно розвивається та працює над покращенням корисних інструментів системи;
- підтримує популярні сервіси хостингу, такі як GitHub, GitLab, Bitbucket;
- дає можливість легко об'єднувати зміни з різних гілок та репозиторіїв в одному проєкті, що дозволяє зручно працювати над проєктом у команді [15].

До недоліків Git можна віднести:

- можуть виникати проблеми зі злиттям гілок, особливо якщо вони значно відрізняються;
- незручне керування гілками;

– необхідність використовувати засоби шифрування або інші заходи для захисту даних, якщо треба зберігати конфіденційну інформацію.

В цілому, зазначені недоліки не заважають якісно використовувати Git, а його переваги роблять дану систему контролю версій кращим варіантом для використання у розробці [15].

2.5 Функціональна схема роботи ІТ-форуму

2.5.1 Функціональна схема форуму для ролі користувача

Беручи до уваги те, що ІТ-форум має систему ролей, його функціонал буде різним для кожної ролі, як було з'ясовано під час дослідження предметної області. Наймасовішою роллю є роль звичайного користувача, який зареєструвався та успішно авторизувався на форумі.

На рис. 2.3 зображено функціональну схему форуму для користувача.

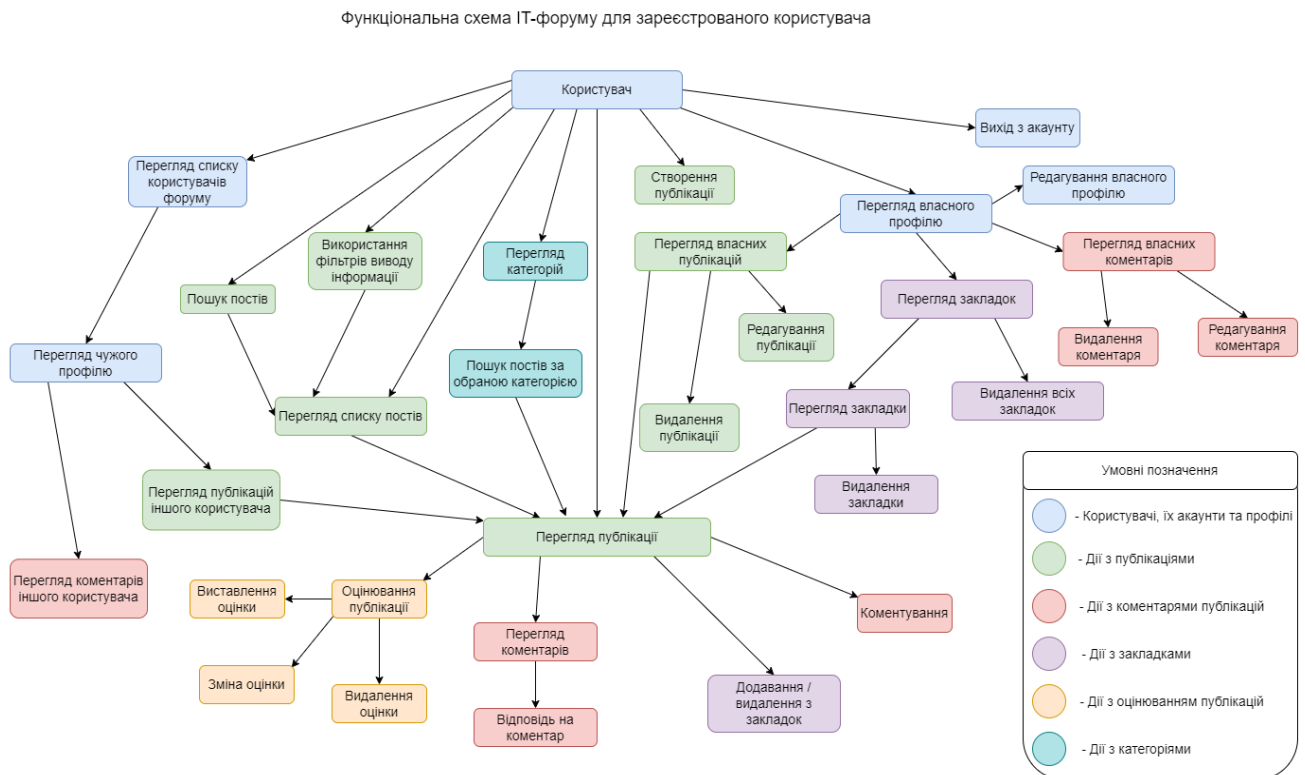


Рисунок 2.3 – Функціональна схема форуму для користувача

Таким чином, основні сутності, з якими може взаємодіяти користувач, можна сформулювати так: «Профілі користувачів / користувачі», «Публікації», «Коментарі», «Оцінки», «Закладки» та «Категорії».

Користувач може взаємодіяти з сутністю «Публікації» у такі способи:

- створення публікацій;
- редагування своїх публікацій;
- видалення своїх публікацій;
- перегляд будь-яких публікацій;
- пошук публікацій за текстовим запитом;
- пошук публікацій за категоріями;
- накладання фільтрів на список публікацій;
- коментування публікації;
- оцінювання публікації;
- додавання або видалення публікації з закладок.

Користувач може взаємодіяти з сутністю «Профілі користувачів / користувачі» наступним чином:

- перегляд списку користувачів форуму;
- редагування та перегляд власного профілю;
- перегляд профілю будь-якого користувача;
- перегляд списку постів будь-якого користувача;
- перегляд списку коментарів будь-якого користувача;
- вихід з акаунту.

Користувач може взаємодіяти з сутністю «Закладки», а саме:

- перегляд списку своїх закладок;
- видалення одразу всіх закладок;
- видалення однієї закладки;
- додавання закладки.

Користувач може взаємодіяти з сутністю «Оцінки» у такі способи:

- створення (виставлення) оцінки публікації;
- зміна вже існуючої оцінки публікації;

– видалення вже існуючої оцінки публікації.

Користувач може взаємодіяти з сутністю «Коментарі» наступним чином:

- створювати нові коментарі;
- редагувати свої коментарі;
- видаляти свої коментарі;
- переглядати існуючі коментарі;
- відповідати на вже існуючі коментарі.

Користувач може взаємодіяти з сутністю «Категорії», а саме:

- переглядати список категорій;
- обирати категорію, список публікацій якої він хотів би переглянути.

2.5.2 Функціональна схема форуму для ролі гостя

Роль гостя – це роль для тих користувачів, які не зареєструвались та не авторизувались на форумі. Функціонал такого типу користувачів набагато менший, ніж у попередньої ролі. На рис. 2.4 зображено функціональну схему форуму для ролі гостя.

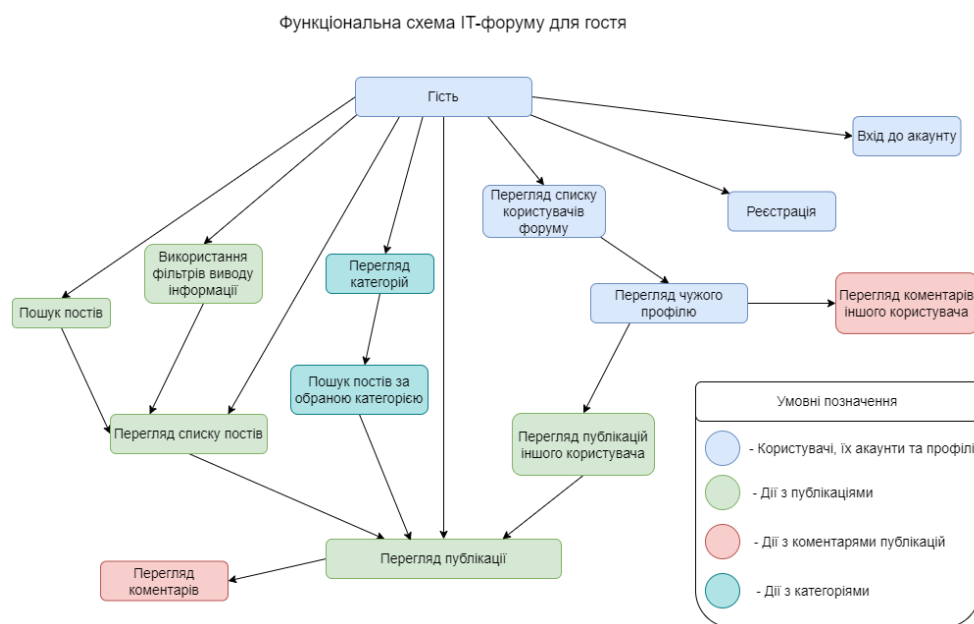


Рисунок 2.4 – Функціональна схема роботи форуму для гостя

Для ролі гостя немає таких понять, як «Закладки» та «Оцінки», тож і на схемі вони відсутні. Основний функціонал гість може отримати тільки після проходження авторизації.

Гість може взаємодіяти з сутністю «Публікації» у такі способи:

- перегляд будь-яких публікацій;
- пошук публікацій за текстовим запитом;
- пошук публікацій за категоріями;
- накладання фільтрів на список публікацій.

Гість може взаємодіяти з сутністю «Профілі користувачів / користувачі» наступним чином:

- перегляд списку користувачів форуму;
- перегляд профілю будь-якого користувача;
- перегляд списку постів будь-якого користувача;
- перегляд списку коментарів будь-якого користувача;
- вхід до акаунту;
- реєстрація.

Гість може взаємодіяти з сутністю «Коментарі» тільки в режимі перегляду існуючих коментарів.

Гість може взаємодіяти з сутністю «Категорії», а саме:

- переглядати список категорій;
- обирати категорію, список публікацій якої він хотів би переглянути.

Таким чином, для гостей залишене ядро пошуку з майже тим же функціоналом, що й у зареєстрованих користувачів. Це зроблено з метою підтримки користувачів, які принципово, або через якісь обставини не можуть зареєструватися та увійти до акаунту. Даний підхід дозволить підвищити відвідуваність форуму, та сприятиме його популяризації у майбутньому.

- видаляти публікації будь-яких користувачів;
- редагувати коментарі будь-яких користувачів;
- видаляти коментарі будь-яких користувачів.

Саме такий набір функцій дає можливість коректування інформації, що підвищує загальний порядок на форумі. Окрім цього, саме адміністратору надана можливість керування категоріями, щоб уникнути самовільного створення великої кількості категорій звичайними користувачами. Таке рішення зберігає кращу структуру інформації, спрощує її адміністрування та суттєво зменшує шанс появи дублікатів.

2.6 Висновки за розділом

В даному розділі було визначено основні компоненти розробки, які необхідні для рішення поставленого завдання.

В якості мови програмування обрано скриптову мову PHP версії 8.0. Для встановлення додаткових модулів та бібліотек вирішено використовувати пакетний менеджер Composer.

Для пришвидшення та покращення процесу розробки й структури всього проєкту обрано фреймворк Laravel версії 9.19.

Для роботи з базами даних обрано СУБД MySQL.

Щоб підвищити якість каскадної розробки форуму, було використано систему контролю версій Git.

Також в даному розділі визначено функціональні схеми IT-форуму для таких ролей користувачів, як гість, зареєстрований користувач та адміністратор. Передбачено основний набір функцій, які повинні мати кожна з зазначених ролей та обґрунтовано саме такий розподіл.

3 ОСНОВНІ РІШЕННЯ РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Основні відомості про розробку

Форум дає можливість ІТ-спеціалістам різних галузей знаходити необхідну інформацію та покращує продуктивність їх роботи.

Backend частина ІТ-форуму розроблена мовою PHP з використанням фреймворку Laravel. Для роботи з залежностями використано Composer, а в якості системи контролю версій – Git. Робота з БД відбувається через MySQL.

Загальна файлова структура застосунку наступна:

- файл системи маршрутів;
- файли класів контролерів;
- файли класів моделей;
- файли класів валідацій;
- файли налаштувань та конфігурацій;
- файли міграцій бази даних;
- файли сідів бази даних.

У таблиці 3.1 наведено детальну файлову структуру з вказанням імен.

Таблиця 3.1 – Детальна файлова структура проекту

№	Тип файлу	Назва файлу
1	2	3
1	Система маршрутів	web.php
2	Класи контролерів	BookmarksController.php
		CategoriesController.php
		CommentsController.php
		HomeController.php
		LikesController.php
		PostsController.php
		UsersController.php

Продовження таблиці 3.1

1	2	3
3	Класи моделей	Bookmark.php
		Category.php
		Comment.php
		Like.php
		Post.php
		User.php
4	Класи валідацій	CategoriesRequest.php
		PostsRequest.php
		UsersRequest.php
5	Налаштування та конфігурації	.env
		DatabaseSeeder.php
6	Міграції бази даних	2014_10_12_000000_create_users_table.php
		2023_01_04_173553_create_posts_table.php
		2023_01_07_191258_create_permission_tables.php
		2023_02_02_222044_create_comments_table.php
		2023_02_14_233309_create_likes_table.php
		2023_04_08_144433_create_bookmarks_table.php
		2023_04_12_125003_create_categories_table.php
		2023_04_12_135738_create_category_post_table.php
7	Сіди бази даних	BookmarksTableSeeder.php
		CategoriesTableSeeder.php
		Category_postTableSeeder.php
		CommentsTableSeeder.php
		LikesTableSeeder.php
		Model_has_rolesTableSeeder.php
		PostsTableSeeder.php
		RolesTableSeeder.php
		UsersTableSeeder.php

Система маршрутів дозволяє керувати вхідними запитами на сервер, створюючи спеціальні адреси для кожного з них. Функції класів контролерів викликаються в залежності від маршруту, по якому прийшов запит. Основною задачею даних класів є відправка модифікації у класи моделей та отримання з них даних. Отримані дані формуються у спеціальному форматі, та надсилаються в якості відповіді з сервера.

Для різноманітних маніпуляцій з базами даних використовуються файли налаштувань та конфігурацій. Щоб автоматично створити схему бази

даних, використовуються файли міграцій. Автоматичне наповнення утвореної структури бази даних реалізовано за допомогою файлів сідів БД.

3.2 Система маршрутизації

Робота будь-якого вебзастосунку базується на системі маршрутизації. Система маршрутизації – це набір адрес, які сервер прослуховує на наявність запитів, в результаті чого, виконує певні дії та надсилає відповідь. Це є основою клієнт-серверної архітектури.

Запити бувають 4 типів та відповідають CRUD-принципу:

- GET – перегляд записів (read);
- POST – створення запису (create);
- PATCH – оновлення запису (update);
- DELETE – видалення запису (delete).

У табл. 3.2 наведено перелік основних розроблених маршрутів та вказано їх роль та тип.

Таблиця 3.2 – Перелік основних розроблених маршрутів

№	Адреса	Тип	Роль
1	2	3	4
1	/	GET	Виведення списку публікацій
2	/posts/create	GET	Виведення сторінки для створення публікації
3	/posts/{post}/edit	GET	Виведення сторінки для редагування публікації
4	/posts/{post}	PATCH	Оновлення публікації
5	/posts/{post}	DELETE	Видалення публікації
6	/posts/deleteInProfile/{post}	DELETE	Видалення публікації через профіль
7	/posts/{post}/comment	POST	Створення нового коментаря
8	/posts/{post}/comment/{comment}	PATCH	Редагування коментаря

Продовження таблиці 3.2

1	2	3	4
9	/posts/{post}/comment/{comment}	DELETE	Видалення коментаря
10	/posts/{post}/like	POST	Виставлення оцінки публікації
11	/posts/{post}/like/{like}	PATCH	Зміна оцінки публікації
12	/posts/{post}/like/{like}	DELETE	Видалення оцінки публікації
13	/users/{user}/edit	GET	Виведення сторінки редагування профілю користувача
14	/users/{user}	PATCH	Редагування профілю
15	/users/{user}/bookmarks	GET	Виведення списку закладок
16	/users/{user}/bookmarks/{bookmark}	DELETE	Видалення закладки через профіль користувача
17	/posts/{post}/bookmarks	POST	Створення закладки на публікації
18	/posts/{post}/bookmarks/{bookmark}	DELETE	Видалення закладки через сторінку публікації
19	/users/{user}/allDeleteBookmarks	DELETE	Видалення усіх закладок користувача
20	/categories/create	GET	Виведення сторінки для створення категорії
21	/categories/create	POST	Створення категорії
22	/categories/{category}	DELETE	Видалення категорії
23	/categories/{category}/edit	GET	Виведення сторінки для редагування категорії
24	/categories/{category}/edit	PATCH	Редагування категорії
25	/categories/controlPanel	GET	Виведення сторінки з панеллю керування категоріями
26	/posts/{post}	GET	Виведення інформації про публікацію
27	/post	POST	Створення публікації
28	/search	GET	Виведення публікацій, що відповідають пошуковому запиту
29	/users	GET	Виведення списку користувачів
30	/users/{user}	GET	Вивід інформації про користувача
31	/users/{user}/comments	GET	Виведення списку коментарів користувача (у його профілі)
32	/categories	GET	Виведення списку категорій
33	/categories/{category}/posts	GET	Виведення публікацій обраної категорії

Деякі маршрути об'єднані у групу «auth», що автоматично робить їх недоступними для неавторизованих користувачів. Даний метод є одним з обов'язкових кроків у створенні системи ролей форуму.

Нижче наведено програмний код зазначених маршрутів, взятий з файлу маршрутизації web.php.

```
Route::get('/', [PostsController::class, 'index']);
Route::group(['middleware' => ['auth']], function () {
    Route::get('/posts/create', [PostsController::class, 'create']);
    Route::get('/posts/{post}/edit', [PostsController::class, 'edit']);
    Route::patch('/posts/{post}', [PostsController::class, 'update']);
    Route::delete('/posts/{post}', [PostsController::class, 'destroy']);
    Route::delete('/posts/deleteInProfile/{post}', [PostsController::class,
'destroyInUserProfile']);
    Route::post('/posts/{post}/comment', [CommentsController::class, 'store']);
    Route::patch('/posts/{post}/comment/{comment}', [CommentsController::class,
'update']);
    Route::delete('/posts/{post}/comment/{comment}', [CommentsController::class,
'destroy']);
    Route::post('/posts/{post}/like', [LikesController::class, 'store']);
    Route::patch('/posts/{post}/like/{like}', [LikesController::class, 'update']);
    Route::delete('/posts/{post}/like/{like}', [LikesController::class, 'destroy']);
    Route::get('/users/{user}/edit', [UsersController::class, 'edit']);
    Route::patch('/users/{user}', [UsersController::class, 'update']);
    Route::get('/users/{user}/bookmarks', [BookmarksController::class, 'index']);
    Route::delete('/users/{user}/bookmarks/{bookmark}', [BookmarksController::class,
'destroy']);
    Route::post('/posts/{post}/bookmarks', [BookmarksController::class, 'store']);
    Route::delete('/posts/{post}/bookmarks/{bookmark}', [BookmarksController::class,
'destroyInPost']);
    Route::delete('/users/{user}/allDeleteBookmarks', [BookmarksController::class,
'allDestroy']);
    Route::get('/categories/create', [CategoriesController::class, 'create']);
    Route::post('/categories/create', [CategoriesController::class, 'store']);
```

```

Route::delete('/categories/{category}', [CategoriesController::class, 'destroy']);
Route::get('/categories/{category}/edit', [CategoriesController::class, 'edit']);
Route::patch('/categories/{category}/edit', [CategoriesController::class, 'update']);
Route::get('/categories/controlPanel', [CategoriesController::class, 'editList']);
});

Route::get('/posts/{post}', [PostsController::class, 'show']);
Route::post('/post', [PostsController::class, 'store']);
Route::get('/search', [PostsController::class, 'search']);
Route::get('/users', [UsersController::class, 'index']);
Route::get('/users/{user}', [UsersController::class, 'show']);
Route::get('/users/{user}/comments', [UsersController::class, 'showUserComments']);
Route::get('/categories', [CategoriesController::class, 'index']);
Route::get('/categories/{category}/posts', [CategoriesController::class, 'show']);

```

Дані маршрути були розроблені з метою досягнення гнучкості та ефективності роботи серверної частини застосунку. Вони можуть бути використані та застосовані у клієнтському додатку.

3.3 Структура бази даних

Розроблений ІТ-форум використовує базу даних, що складається з двох основних частин: структури, що реалізує систему ролей, та структури, яка містить в особі всю основну інформацію форуму.

Для реалізації структури системи ролей, було використано такі сутності:

- «Ролі»;
- «Дозволи»;
- «Роль має дозволи»;
- «Модель має дозволи»;
- «Модель має ролі».

На рис. 3.1 зображено схему БД для системи ролей.

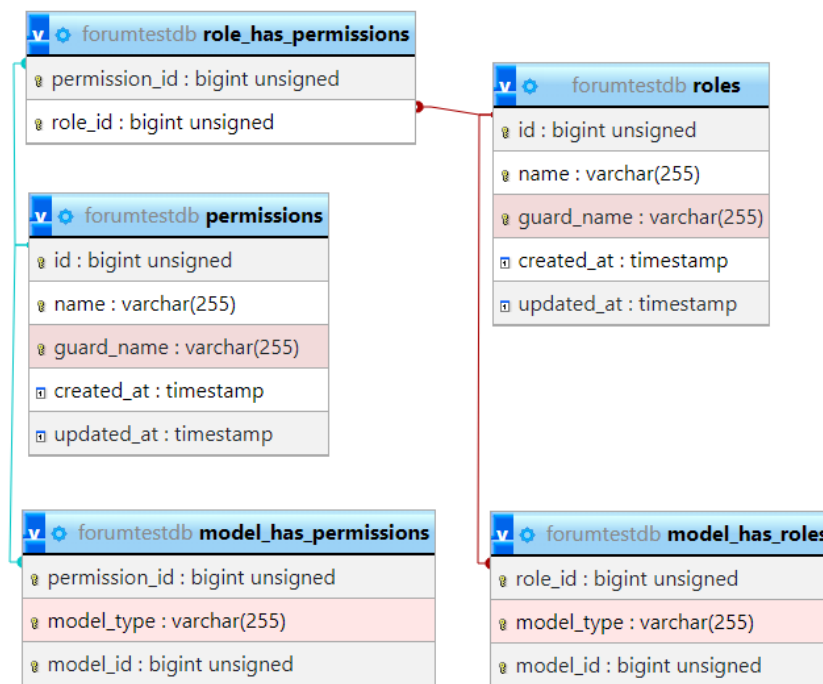


Рисунок 3.1 – Схема БД, що реалізує систему ролей на форумі

Сутність «Ролі» необхідна для збереження переліку ролей, які можуть мати користувачі форуму. «Дозволи» - це набір доступів до тих чи інших дій на форумі. Для того, щоб кожна роль мала свій власний набір дозволів, створено сутність «Роль має дозволи». Оскільки основна бізнес-логіка міститься у моделях, їм також видано переліки ролей та доступів, якими вони мають керуватися. Цей крок реалізують сутності «Модель має дозволи» та «Модель має ролі», а також відповідні їм таблиці.

Для реалізації основної інформаційної структури форуму, було використано такі сутності:

- «Користувачі»;
- «Публікації»;
- «Категорії»;
- «Коментарі»;
- «Категорія поста»;
- «Оцінки»;
- «Закладки».

На рис. 3.2 зображено схему БД для основної інформаційної структури форуму.

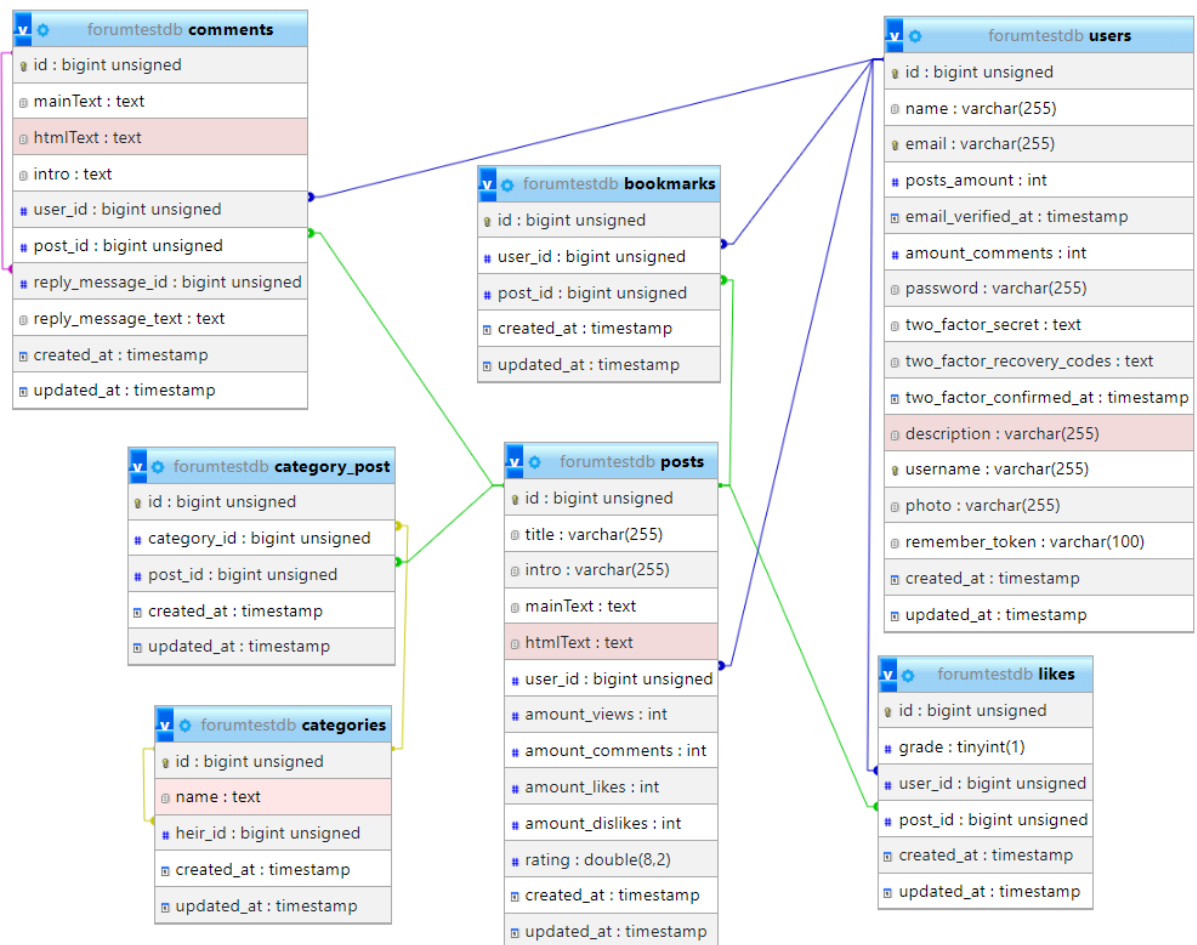


Рисунок 3.2 – Схема БД, що реалізує основну інформаційну структуру

У даному випадку схема має декілька зв'язків між сутностями. Сутність «Користувач» пов'язана з сутностями:

- «Публікації», як 1 до багатьох, адже один користувач може створювати безліч публікацій;
- «Закладки», як 1 до багатьох, адже один користувач може додавати безліч публікацій до закладок;
- «Коментарі», як 1 до багатьох, адже один користувач може писати безліч коментарів;
- «Оцінки», як 1 до багатьох, адже один користувач може ставити безліч оцінок.

Сутність «Публікації» пов’язана з сутностями:

- «Закладки», як 1 до багатьох, адже одна і та сама публікація може бути додано у закладки різними користувачами;
- «Оцінки», як 1 до багатьох, адже одна і та сама публікація може бути оцінений багатьма користувачами;
- «Коментарі», як 1 до багатьох, адже одна і та сама публікація може бути багаторазово прокоментований;
- «Категорія посту» , як 1 до багатьох, адже одна і та сама публікація може мати декілька категорій.

Сутність «Категорії» пов’язана з сутностями:

- «Категорія посту», як 1 до багатьох, адже одна й та сама категорія може бути надана багатьом публікаціям;
- «Категорії» (тобто сама з собою), як 1 до багатьох. Даний зв’язок необхідний для утворення сутності «Підкатегорії», яка не виділяється як окрема в міру того, що має таку ж структуру як і сутність «Категорії».

Сутність «Коментарі» пов’язана сама з собою, як 1 до багатьох, щоб утворити сутність «Відповіді на коментарі».

3.4 Розроблені класи контролерів

3.4.1 Клас BookmarksController

Даний клас відповідає за керування закладками користувачів. Для виведення списку закладок використовується функція, наведена нижче.

```
public function index(User $user, Request $req)
{
    if($req->user()->id == $user->id)
    {
        $bookmarks = $user->bookmarks()->paginate(5);
        return view('users.userBookmarks', compact('user', 'bookmarks'));
    } else {
```

```

        return redirect(url()->previous());
    }
}

```

Для видалення однієї закладки використовується функція `destroy`:

```

public function destroy(User $user, Bookmark $bookmark)
{
    $bookmark->delete();
    return redirect(url()->previous());
}

```

Для створення закладки використовується функція `store`:

```

public function store(Post $post, Request $req)
{
    Like::decViewsAmountsForPost($post);
    $post->bookmarks()->attach($post->id, Bookmark::getDataArrayForInsert($req));
    return redirect(url()->previous());
}

```

Для того, щоб користувач міг видалити одразу всі свої закладки, створено функцію `allDestroy`:

```

public function allDestroy(User $user, Request $req)
{
    if($req->user()->id == $user->id)
    {
        Bookmark::where('user_id', $user->id)->delete();
    }
    return redirect(url()->previous());
}

```

3.4.2 Клас CategoriesController

Даний клас відповідає за керування категоріями публікацій. Для виведення списку категорії використовується функція index:

```
public function index()
{
    $categories = Category::getHeirs();
    return view('categories.index', compact('categories'));
}
```

Функція create відправляє список базових категорій на сторінку створення нової категорії:

```
public function create(Request $req)
{
    if($req->user()->hasRole('admin'))
    {
        $basicCategories = Category::getBasicCategories();
        return view("categories.create", compact('basicCategories'));
    } else {
        return redirect(url()->previous());
    }
}
```

Для створення нової категорії використовується функція store:

```
public function store(CategoriesRequest $req)
{
    Category::create(Category::getArrData($req));
    return redirect('/categories');
}
```

Для видалення категорії використовується функція destroy:

```
public function destroy(Category $category, Request $req)
{
    if($req->user()->hasRole('admin'))
    {
        $category->delete();
    }
    return redirect('/categories/controlPanel');
}
```

Для редагування категорії використовується функція update:

```
public function update(Category $category, CategoriesRequest $req)
{
    $category->update(['name' => $req->name]);
    return redirect('/categories/controlPanel');
}
```

Для виведення списку публікацій, що має обрану категорію, використовується функція show:

```
public function show(Category $category, Request $req)
{
    $posts = $category->getPostsByCategory($req);
    $filters = Post::getFilterParams($req);
    return view('index', compact('posts', 'filters'));
}
```

3.4.3 Клас CommentsController

Даний клас відповідає за керування коментарями публікацій. Для створення нового коментаря використовується функція store:

```
public function store(Post $post, Request $req)
```

```

{
    $post->comments()->attach($post->id, Comment::getDataArrayForInsert($req));
    $post->increment('amount_comments');
    User::find($req->user()->id)->increment('amount_comments');
    return redirect('/posts/'.$post->id);
}

```

Для видалення коментаря використовується функція `destroy`:

```

public function destroy(Post $post, Comment $comment, Request $req)
{
    if($req->user()->id == $comment->user_id || $req->user()->hasRole('admin'))
    {
        // transformation reply-comments into comments
        $answers = Comment::getReplyComments($comment->id);
        foreach ($answers as $answer){
            $answer->update(
                [
                    'reply_message_text' => NULL,
                    'reply_message_id' => NULL
                ]
            );
        }
        $comment->delete();
        $post->decrement('amount_comments');
        User::find($req->user()->id)->decrement('amount_comments');
        return redirect('/posts/'.$post->id);
    } else {
        return redirect('/');
    }
}

```

Для редагування коментаря використовується функція `update`:

```

public function update(Post $post, Comment $comment, Request $req)

```

```

{
    if($req->user()->id == $comment->user_id)
    {
        // update intro in reply-message
        $answers = Comment::getReplyComments($comment->id);
        foreach ($answers as $answer){
            $answer->update(
                [
                    'reply_message_text' => Comment::generateIntro($req->mainText)
                ]
            );
        }
        $comment->update(request(['htmlText', 'mainText'])) +
        Comment::getReplyParamsForUpdate($req));
        return redirect('/posts/'.$post->id.'#'.$comment->id);
    } else {
        return redirect('/');
    }
}
}

```

3.4.4 Клас LikesController

Даний клас відповідає за керування оцінками публікацій. Для створення нової оцінки використовується функція store:

```

public function store(Post $post, Request $req)
{
    $post->likes()->attach($post->id, Like::getDataArrayForInsert($req));
    Like::incrementLikeAmounts($post, $req->grade);
    Like::decViewsAmountsForPost($post);
    return redirect('/posts/'.$post->id);
}

```

Для редагування оцінки використовується функція update:

```
public function update(Post $post, Like $like, Request $req)
{
    $like->update(Like::getDataArrayForInsert($req));
    Like::editLikeAmounts($post, $req->grade);
    Like::decViewsAmountsForPost($post);
    return redirect('/posts/'.$post->id);
}
```

Для видалення оцінки використовується функція `destroy`:

```
public function destroy(Post $post, Like $like)
{
    $like->delete();
    Like::decrementLikeAmounts($post, $like->grade);
    Like::decViewsAmountsForPost($post);
    return redirect('/posts/'.$post->id);
}
```

3.4.5 Клас **PostsController**

Даний клас відповідає за керування публікаціями. Для виведення списку всіх публікацій використовується функція `index`:

```
public function index(Request $request)
{
    $posts = Post::getPosts($request, 'all');
    $filters = Post::getFilterParams($request);
    return view('index', compact('posts', 'filters'));
}
```

Для виведення публікації використовується функція `show`:

```
public function show(Post $post)
```

```

{
    $post->increment('amount_views');
    $post->updateRating();
    $commentators = $post->comments->unique('id');
    if(Auth::check())
    {
        $bookmark = Bookmark::where('post_id', $post->id)->where('user_id',
Auth::id())->first();
        $userLike = Like::where('post_id', $post->id)->where('user_id', Auth::id())-
>first();
        return view('posts.show', compact('post', 'commentators', 'userLike',
'bookmark'));
    }
    return view('posts.show', compact('post', 'commentators'));
}

```

Для створення публікації використовується функція store:

```

public function store(PostsRequest $req)
{
    $user = User::find($req->user()->id);
    $post = $user->posts()->create(Post::getArrPostData($req));
    $post->setCategoriesByCreatePost($req);
    $user->increment('posts_amount');
    return redirect('/');
}

```

Для редагування публікації використовується функція update:

```

public function update(Post $post, PostsRequest $req)
{
    DB::table('category_post')->where('post_id', $post->id)->delete();
    $post->setCategoriesByCreatePost($req);
    $post->update(Post::getArrPostData($req));
    return redirect('/');
}

```

```
}
```

Для видалення публікації використовується функція `destroy`:

```
public function destroy(Post $post, Request $req)
{
    self::deletePost($post, $req);
    return redirect('/');
}
```

Для пошуку публікацій використовується функція `search`:

```
public function search(Request $request)
{
    $posts = Post::getPosts($request, 'search');
    $searchText = $request->s;
    $filters = Post::getFilterParams($request);
    return view('index', compact('posts', 'searchText', 'filters'));
}
```

3.4.6 Клас `UserController`

Даний клас відповідає за керування акаунтами користувачів. Для виведення списку всіх користувачів форуму використовується функція `index`:

```
public function index()
{
    $users = User::paginate(28);
    return view('users.index', compact('users'));
}
```

Для виведення інформації конкретного користувача використовується функція `show`:

```
public function show(User $user)
{
    $posts = Post::query()
        ->where('user_id', $user->id)
        ->paginate(3);
    return view('users.userPosts', compact('user', 'posts'));
}
```

Для виведення коментарів конкретного користувача використовується функція `showUserComments`:

```
public function showUserComments(User $user)
{
    $comments = $user->comments()->paginate(3);
    return view('users.userComments', compact('user', 'comments'));
}
```

Для редагування даних акаунту використовується функція `update`:

```
public function update(User $user, UsersRequest $req)
{
    $user->updatePhoto($req);
    $user->update(User::getArrUserData($req));
    return redirect('/users/'.$user->id);
}
```

3.5 Розроблені класи моделей

3.5.1 Клас `Category`

Клас `Category` є моделлю, яка відповідає сутності «Категорії». Для отримання списку базових категорій модель має функцію `getBasicCategories`:

```
public static function getBasicCategories()
{
```

```

        return self::query()->whereNull('heir_id')->get();
    }

```

Для отримання списку підкатегорій модель має функцію `getHeirs`:

```

public static function getHeirs()
{
    $categories = [];
    $basicCategories = self::getBasicCategories();
    foreach ($basicCategories as $basicCategory){
        $categories[] = [
            'id' => $basicCategory->id,
            'name' => $basicCategory->name,
            'heirs' => self::query()->where('heir_id', $basicCategory->id)->get()
        ];
    }
    return $categories;
}

```

Для отримання списку постів, що містять обрану категорію з врахуванням фільтрів, модель має функцію `getPostsByCategory`:

```

public function getPostsByCategory($request)
{
    if($request->type == null) {
        $request->type = 'rating';
    }
    if($request->growth == null) {
        $request->growth = 'desc';
    }
    return $this->posts()->orderBy($request->type, $request->growth)->paginate(6);
}

```

3.5.2 Клас Comment

Клас Comment є моделлю, яка відповідає сутності «Коментарі». Для генерації скороченої назви коментаря модель має функцію generateIntro:

```
public static function generateIntro($text)
{
    if (strlen($text) > 30)
    {
        $text = mb_strimwidth($text, 0, 30, "...");
    }
    return $text;
}
```

Для формування даних перед створенням коментаря модель має функцію getDataArrayForInsert:

```
public static function getDataArrayForInsert($request)
{
    $dataForInsert = [
        'htmlText' => $request->htmlText,
        'mainText' => $request->mainText,
        'intro' => Comment::generateIntro($request->mainText),
        'user_id' => $request->user()->id
    ];
    if($request->reply_message_id != null)
    {
        $dataForInsert['reply_message_id'] = $request->reply_message_id;
        $dataForInsert['reply_message_text'] = $request->reply_message_text;
    }
    return $dataForInsert;
}
```

Для отримання коментарів, які є відповідями на інші коментарі, модель має функцію `getReplyComments`:

```
public static function getReplyComments($comment_id)
{
    return Comment::where('reply_message_id', '=', $comment_id)->get();
}
```

3.5.3 Клас Like

Клас `Like` є моделлю, яка відповідає сутності «Оцінки». Для збільшення лічильника оцінок та рейтингу модель має функцію `incrementLikeAmounts`:

```
public static function incrementLikeAmounts(Post $post, $grade)
{
    if($grade == True) {
        $post->increment('amount_likes');
        $post->increment('rating');
    } else {
        $post->increment('amount_dislikes');
    }
}
```

Для зменшення лічильника оцінок та рейтингу модель має функцію `decrementLikeAmounts`:

```
public static function decrementLikeAmounts(Post $post, $grade)
{
    if($grade == True) {
        $post->decrement('amount_likes');
        $post->decrement('rating');
    } else {
```

```

        $post->decrement('amount_dislikes');
    }
}

```

Для корегування лічильника оцінок та рейтингу модель має функцію `editLikeAmounts`:

```

public static function editLikeAmounts(Post $post, $grade)
{
    if($grade == True) {
        $post->increment('amount_likes');
        $post->decrement('amount_dislikes');
        $post->increment('rating');
    } else {
        $post->decrement('amount_likes');
        $post->increment('amount_dislikes');
        $post->decrement('rating');
    }
}

```

Для корегування лічильника переглядів та рейтингу модель має функцію `decViewsAmountsForPost`:

```

public static function decViewsAmountsForPost(Post $post)
{
    $post->decrement('amount_views');
    $post->update(['rating' => $post->rating - 0.25]);
}

```

3.5.4 Клас Post

Клас `Post` є моделлю, яка відповідає сутності «Публікації».

Для отримання списку публікацій з урахуванням фільтрів модель має функцію `getPosts`:

```
public static function getPosts($request, $mode)
{
    if($request->type == null) {
        $request->type = 'rating';
    }
    if($request->growth == null) {
        $request->growth = 'desc';
    }
    if($mode === 'all') {
        return self::orderBy($request->type, $request->growth)->paginate(6);
    } elseif ($mode === 'search') {
        return self::where('title', 'LIKE', "%{$request->s}%")->orderBy($request->type,
        $request->growth)->paginate(6);
    }
}
```

Для отримання списку доступних фільтрів модель має функцію `getFilterParams`:

```
public static function getFilterParams($request) {
    $filters = [];
    if($request->type == null) {
        $filters['sortingType'] = 'rating';
    } else {
        $filters['sortingType'] = $request->type;
    }
    if($request->growth == null) {
        $filters['sortingGrowth'] = 'desc';
    } else {
        $filters['sortingGrowth'] = $request->growth;
    }
}
```

```

    return $filters;
}

```

Для встановлення категорій при створенні публікації модель має функцію `setCategoriesByCreatePost`:

```

public function setCategoriesByCreatePost($request)
{
    $categAttributes = array_filter($request->all(), function($key) {
        return strpos($key, 'category') !== false;
    }, ARRAY_FILTER_USE_KEY);
    foreach ($categAttributes as $attribute){
        $this->categories()->attach($attribute);
    }
}

```

3.5.5 Клас User

Клас `User` є моделлю, яка відповідає сутності «Користувачі». Для отримання нікнейму користувача модель має функцію `getUsername`:

```

public static function getUsername($str)
{
    if(is_null($str)) {
        return null;
    } else {
        return '@'.$str;
    }
}

```

Для оновлення фотографії профілю користувача модель має функцію `updatePhoto`:

```

public function updatePhoto($request)

```

```

{
    if($request->hasFile('photo'))
    {
        $this->clearOldPhoto($this->id);
        $photo = $request->file('photo');
        $filename = time().'.'.$photo->getClientOriginalExtension();
        Image::make($photo)->fit(300, 300)->save(public_path($this->getPhotoPath($this->id)).$filename);
        $this->photo = $filename;
        $this->save();
    } elseif ($request->flagPhoto === "deleted") {
        $this->clearOldPhoto($this->id);
        $this->photo = null;
        $this->save();
    }
}
}

```

Для отримання адреси розташування файлу фотографії профілю користувача у файловій системі модель має функцію `getPhotoPath`:

```

public function getPhotoPath($user_id)
{
    $path = "uploads/usersPhotos/id{$user_id}";
    if(!file_exists($path)) {
        if (!mkdir($path, 0777, true) && !is_dir($path)) {
            throw new \RuntimeException(sprintf('Directory "%s" was not created',
                $path));
        }
    }
    return "$path/";
}

```

Для видалення старої фотографії профілю користувача з файлової системи модель має функцію `clearOldPhoto`:

```

public function clearOldPhoto($user_id)
{
    $path = "uploads/usersPhotos/id{$user_id}";
    if(file_exists(public_path("/$path")))
    {
        foreach(glob(public_path("/$path/*")) as $photo)
        {
            unlink($photo);
        }
    }
}

```

3.6 Розроблені класи валідації

3.6.1 Клас CategoriesRequest

Даний клас відповідає за валідацію даних, пов'язаних з категоріями. Для валідації використовується функція правил (rules) та повідомлень (messages):

```

public function rules()
{
    return [
        'name' => 'required'
    ];
}

public function messages(){
    return [
        'name.required' => 'Поле імені не може бути порожнім',
    ];
}

```

3.6.2 Клас PostsRequest

Даний клас відповідає за валідацію даних, пов'язаних з публікаціями. Для валідації використовується функція правил (rules) та повідомлень (messages):

```
public function rules()
{
    return [
        'title' => 'required|min:2',
        'mainText' => 'required'
    ];
}

public function messages(){
    return [
        'title' => 'Назва публікації повинна мати не менше 2-ох символів',
        'mainText' => 'Текст публікації не може бути порожнім'
    ];
}
```

3.6.3 Клас UsersRequest

Даний клас відповідає за валідацію даних, пов'язаних з користувачами. Для валідації використовується функція правил (rules) та повідомлень (messages):

```
public function rules()
{
    return [
        'description' => 'max:50',
        'username' => ['nullable','sometimes','max:15','regex:/^[a-zA-Z0-9_]+$/'],
        Rule::unique('users')->ignore(Auth::user()->id)],
```

```

        'photo' => 'mimes:jpg,bmp,png',
        'name' => 'required|max:15'
    ];
}

public function messages(){
    return [
        'description' => 'Занадто багато символів в описі профілю (допустиме
значення: не більше 50)',
        'username.max' => 'Занадто багато символів в юзернеймі (допустиме
значення: не більше 15)',
        'username.unique' => 'Такий юзернейм вже зайнятий',
        'regex' => 'Юзернейм може містити тільки букви, цифри та символ "_"',
        'photo' => 'Неправильний тип файлу (допустимі формати: jpg, png, bmp)',
        'name.required' => 'Поле імені не може бути порожнім',
        'name.max' => 'Поле імені містить забагато символів (допустиме значення:
не більше 15)'
    ];
}

```

3.7 Висновки за розділом

В даному розділі було наведено основні відомості про розробку backend частини ІТ-форуму. Детально описано створену систему маршрутизації сайту.

Наведено структуру розробленої бази даних та сутностей, які її сформували. Описані та обґрунтовані зв'язки між таблицями у БД.

Наведено перелік базових структурних класів проєкту – класів валідації, моделей та контролерів. Описано основні розроблені функції цих класів, наведено їх код.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Розгортання локального сервера розробки

Для подальшого вдосконалення створеного ІТ-форуму рекомендується використовувати безкоштовне програмне забезпечення під назвою Open Server Panel. Даний програмний продукт забезпечує розгортання локального сервера та надає ряд інструментів для роботи з ним.

На рис. 4.1 показано приклад меню управління Open Server Panel та його запуск.

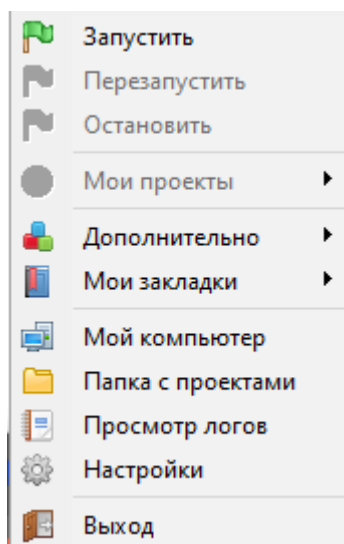


Рисунок 4.1 – Меню управління Open Server Panel

Під час запуску сервера буде змінюватись колір прапорця, що свідчить про успішність роботи програми. На рис. 4.2 зображено опції швидкого доступу до вже встановлених проєктів.

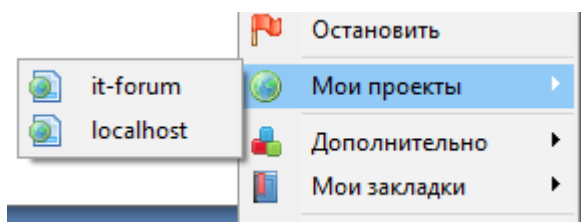


Рисунок 4.2 – Опції швидкого доступу до проєктів

Open Server Panel дозволяє вести розробку одразу декількох проєктів та запобігає конфліктів між ними. Для їх розміщення є спеціальна системна директорія «domains», в якій мають бути усі проєкти. В іншому випадку програмне просто не зможе розгорнути локальний сервер для них. У меню керування є відповідна опція, що дозволяє швидко потрапити у дану директорію.

На рис. 4.3 зображено опцію швидкого доступу в директорію проєктів та саму директорію.

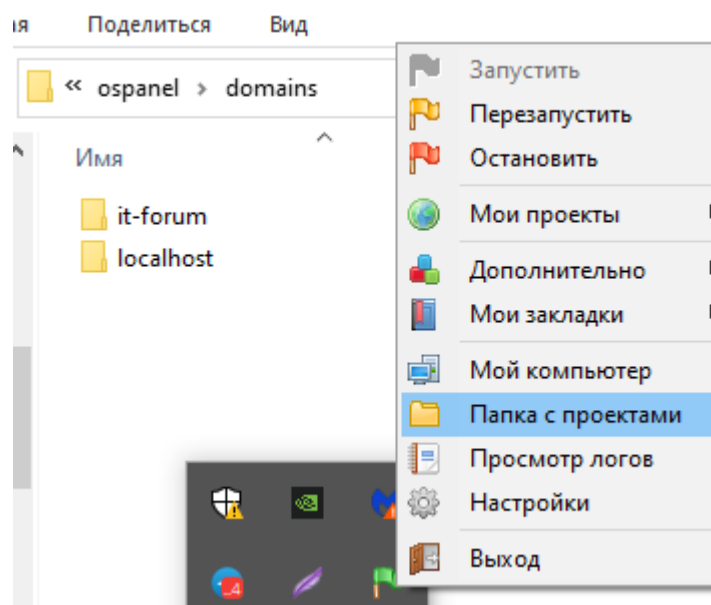


Рисунок 4.3 – Опція швидкого доступу до директорії проєктів

Для налаштування програми необхідно використати відповідну опцію. У вікні налаштувань треба обрати вкладинку «Модулі» для керування модулями сервера, баз даних та мови програмування.

Для зміни портів необхідно відкрити вкладинку «Сервер» та встановити вільні порти для обраних модулів. Рекомендується вносити зміни тільки за крайньої потреби.

На рис. 4.4 зображено відкрите вікно налаштувань портів модулів.

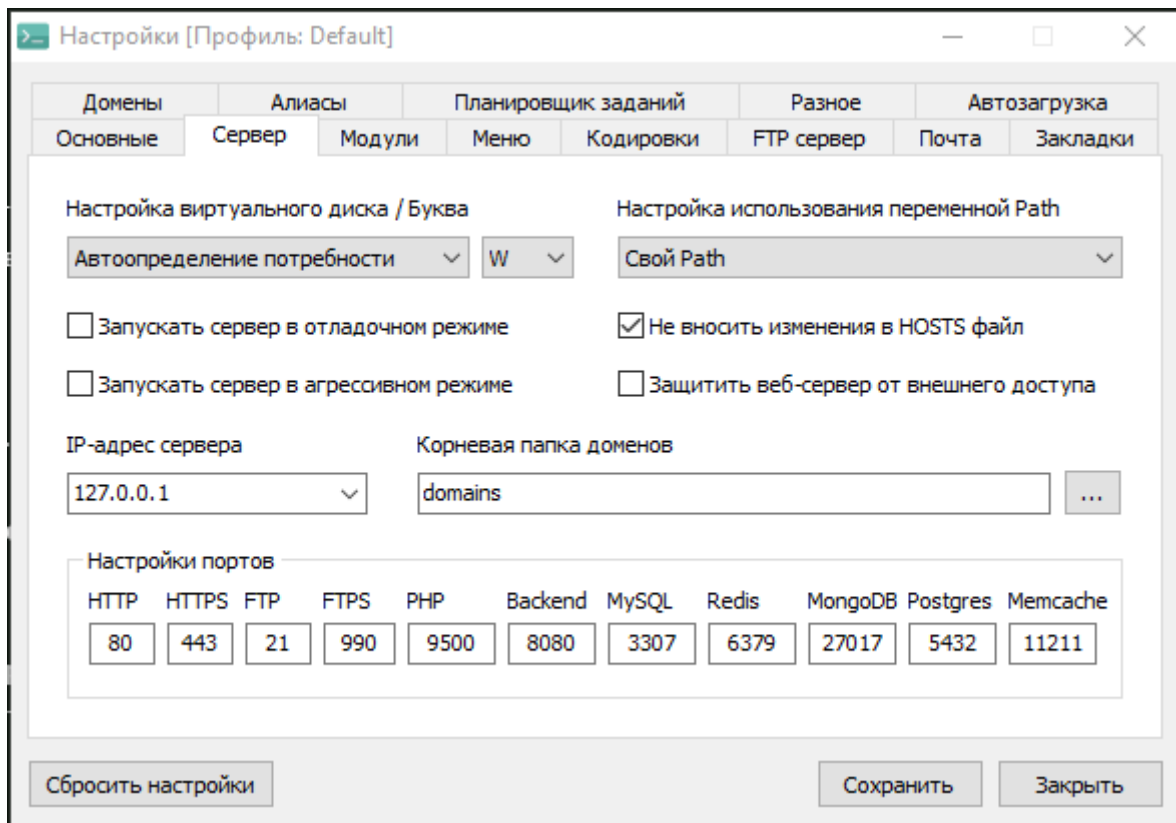


Рисунок 4.4 – Вікно налаштувань портів модулів

4.2 Інсталяція вебфоруму

Проект розробленої backend частини ІТ-форуму поставляється в форматі архіву «it-forum.zip». Також проєкт може бути підтягнутий з віддаленого репозитарію при наявності доступу.

Для інсталяції форуму необхідні:

- встановлений PHP версії 8.0;
- встановлений Laravel версії 9.19;
- встановлений пакетний менеджер Composer.

Першим кроком необхідно розпакувати вміст архіву «it-forum.zip» у файлову систему та перейти у консолі в директорію «it-forum», що утвориться в результаті розпакування.

Далі необхідно виконати команду «composer install». Команда автоматично встановить всі залежності проєкту.

Наступним кроком треба виконати команду «`npm i`». Команда автоматично оновить модулі проєкту до необхідних версій.

Після даних кроків форум готовий до подальших маніпуляцій. Усі необхідні команди також містяться в описовому файлі проєкту «`README.md`».

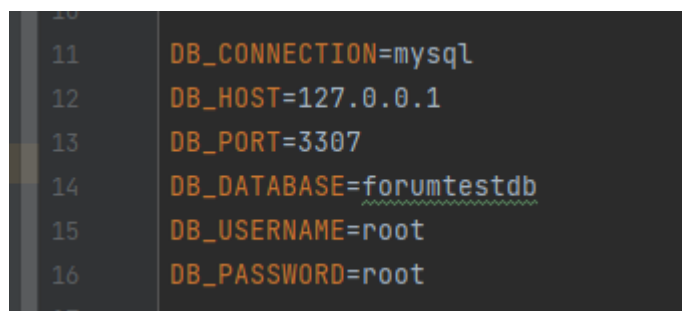
4.3 Розгортання структури бази даних

Щоб швидко розгорнути структуру бази даних, існує інструмент міграцій. Він дозволяє автоматизувати цей процес та не створювати таблиці вручну.

Для виконання даного кроку повинні бути виконані наступні умови:

- розгорнутий та ввімкнутий сервер розробки;
- імпортовано проєкт вебфоруму;
- встановлено всі компоненти розробки;
- встановлено всі додаткові залежності та модулі.

Далі необхідно задати конфігурацію підключення до бази даних. Для цього слід відкрити файл налаштувань «`.env`» та знайти у ньому рядки, показані на рис. 4.5.



```
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3307
14 DB_DATABASE=forumtestdb
15 DB_USERNAME=root
16 DB_PASSWORD=root
```

Рисунок 4.5 – Частина файлу налаштувань «`.env`», що відповідає за підключення до БД

Необхідно задати тип бази даних, хост, на яку знаходиться сервер з БД, порт, через який виконуватиметься підключення, назву бази даних та поля доступу.

Тепер треба запустити механізм міграцій. Це можна зробити за допомогою команди «php artisan migrate» у командному рядку середовища розробки. Ця команда утворить таблиці у БД, відповідні файлам міграцій, зображених на рис. 4.6.

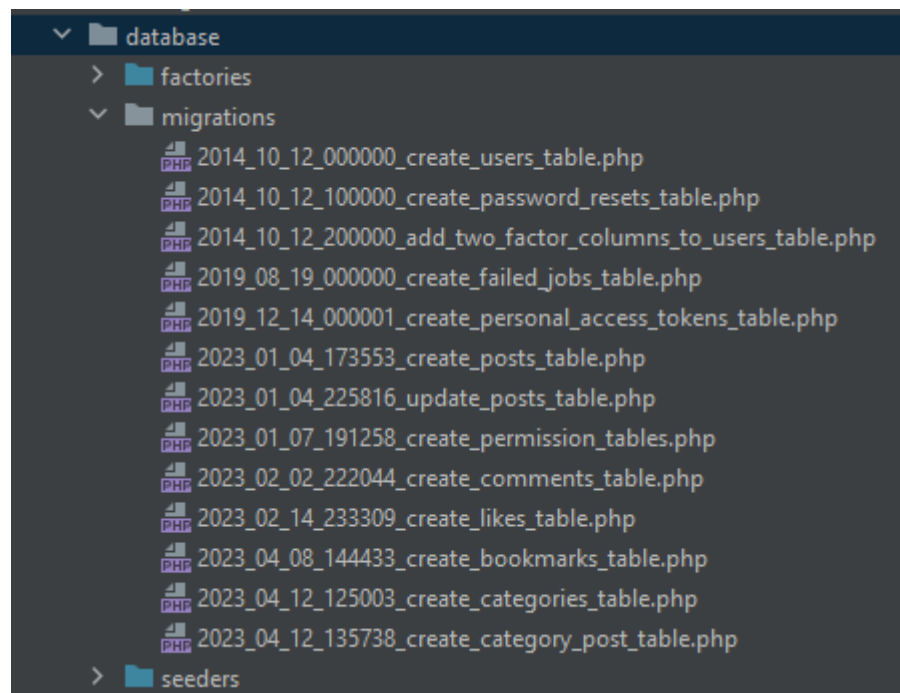


Рисунок 4.6 – Файли міграцій

На даному етапі буде утворено порожню базу даних (згенеруються таблиці, їх поля та зв'язки між ними, але не їх наповнення), структура якої повністю відповідає тій, що була використана у розробці форуму.

4.4 Заповнення бази даних тестовими даними

Для пришвидшення тестування та розробки форуму можуть знадобитись певні дані з БД. Для їх автоматичної генерації існує відповідний

механізм – сіди. На рис. 4.7 зображено розташування сідів у структурі проєкту.

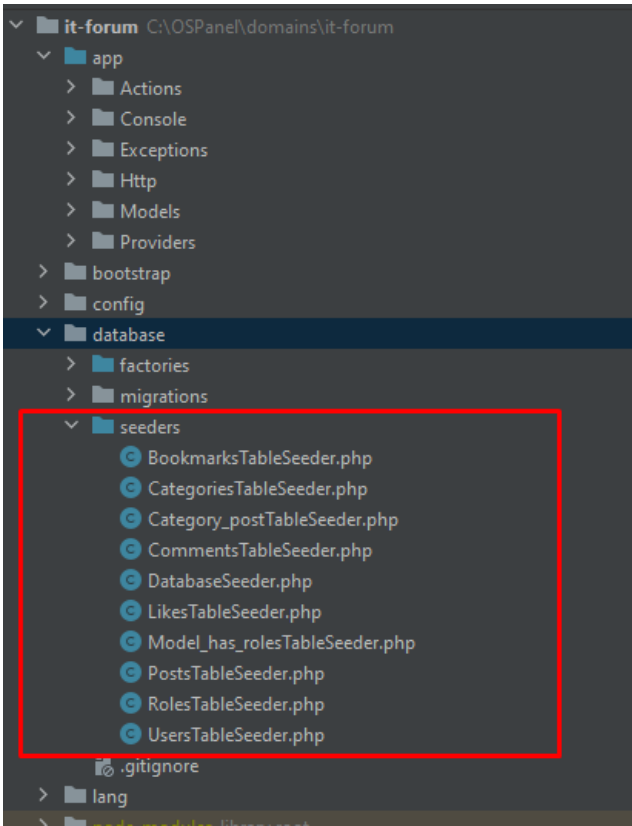


Рисунок 4.7 – Розташування файлів сідів у проєкті

У табл. 4.1 наведено перелік всіх існуючих сідів та їх роль.

Таблиця 4.1 – Перелік основних розроблених маршрутів

№	Файл	Роль
1	2	3
1	BookmarksTableSeeder.php	Закладки користувачів
2	CategoriesTableSeeder.php	Всі категорії та підкатегорії публікацій
3	Category_postTableSeeder.php	Належність публікації до категорії
4	CommentsTableSeeder.php	Усі коментарі всіх публікацій
5	LikesTableSeeder.php	Всі оцінки публікацій
6	Model_has_rolesTableSeeder	Приналежність ролей моделям
7	PostsTableSeeder.php	Дані публікацій
8	RolesTableSeeder.php	Перелік ролей користувачів форуму
9	UsersTableSeeder.php	Дані акаунтів користувачів

Після створення нових сідів, їх обов'язково слід зареєструвати у файлі «DatabaseSeeder.php». На рис. 4.8 зображено приклад реєстрації сідів на основі існуючих.



```
public function run()
{
    $this->call( class: UsersTableSeeder::class);
    $this->call( class: PostsTableSeeder::class);
    $this->call( class: RolesTableSeeder::class);
    $this->call( class: Model_has_rolesTableSeeder::class);
    $this->call( class: CommentsTableSeeder::class);
    $this->call( class: LikesTableSeeder::class);
    $this->call( class: BookmarksTableSeeder::class);
    $this->call( class: CategoriesTableSeeder::class);
    $this->call( class: Category_postTableSeeder::class);
}
```

Рисунок 4.8 – Приклад реєстрації сідів у системі Laravel

Якщо сід буде незареєстрованим, помилку в роботі механізму сідування це не викличе. Такий сід буде просто проігноровано.

Щоб запустити сідування, треба виконати команду «php artisan db:seed» у командному рядку.

Якщо при цьому попередньо не виконано міграції, виникне помилка. Щоб виконати міграції та сіди разом, існує команда «php artisan migrate:refresh --seed». Рекомендується проводити оновлення бази даних саме цією командою.

Після виконання сідів, база даних буде повністю готова до роботи.

4.5 Висновки за розділом

В даному розділі були розглянуті основні кроки, необхідні для подальшої розробки та вдосконалення вебфоруму.

Описано принцип розгортання сервера розробки та налаштувань відповідного програмного забезпечення. Розглянуто приклад налаштування портів та компонентів сервера.

Наведено детальну інструкцію з інсталяції форуму, його правильного розташування у файловій системі, налаштування, а також інсталяцію усіх залежностей та модулів за допомогою пакетного менеджера.

Описано механізми розгортання структури бази даних, її налаштування та підключення до неї.

Наведено принцип роботи механізму автоматичного заповнення даними БД. Додано описову характеристику вже створених модулів генерації даних. Продемонстровано приклад підключення нових модулів.

5 КЕРІВНИЦТВО КОРИСТУВАЧА

5.1 Загальні відомості

Оскільки розроблений застосунок є лише серверною частиною, звичайний користувач не зможе скористатися їм без відповідного клієнтського додатка. Тому далі під користувачем матиметься на увазі саме клієнтське програмне забезпечення, що здатне надсилати запити на сервер (до розробленого додатка) та отримуватиме відповіді від нього.

Розроблена backend частина вебдодатка надає можливість користуватися ІТ-форумом сучасного зразку з відповідною архітектурою та основними інструментами керування інформацією. Форум має систему ролей користувачів та базовий захист персональної інформації.

Основний вид взаємодії зі створеним програмним забезпеченням – надсилення запитів. Для відправлення запитів на сервер, розроблена спеціальна система маршрутів, яка утворює зручне API.

Далі будуть розглянуті найосновніші види запитів та дані, які надсилає сервер в якості відповіді.

5.2 Запит на отримання списку всіх публікацій

Щоб отримати загальний список усіх публікацій, необхідно звернутись на сервер за адресою «/». Запит повинен мати тип «GET». Ніяких додаткових атрибутів та параметрів не треба.

У відповідь сервер надішле 2 колекції:

- «posts» - колекція публікацій;
- «filters» - колекція фільтрів.

Також буде відправлено дані пагінації, якщо постів занадто багато.

На рис. 5.1 зображено приклад такого запиту та відповідь з детальним описом всіх атрибутів колекцій.

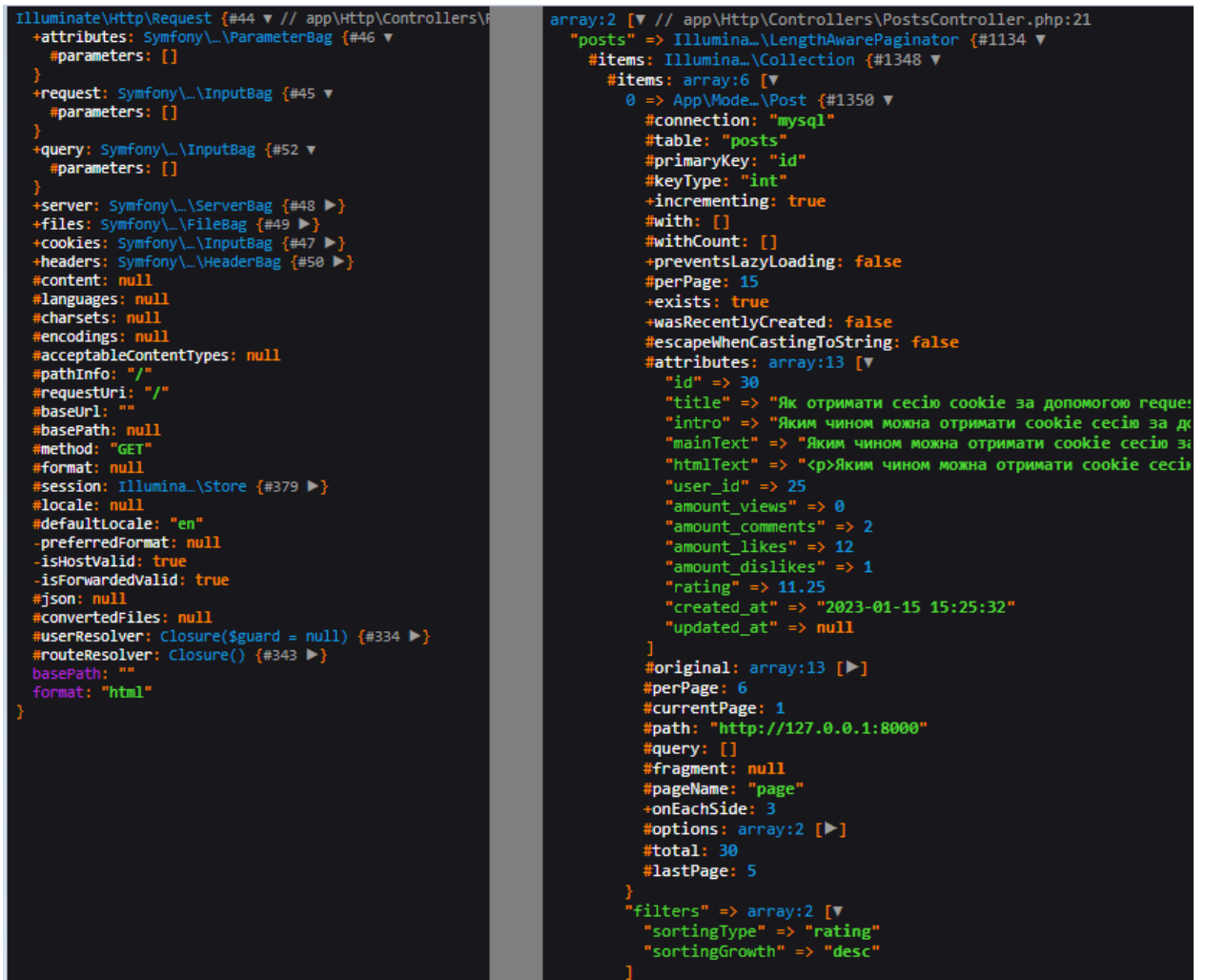


Рисунок 5.1 – Приклад запиту (ліворуч) та відповіді (праворуч)

5.3 Запит на отримання даних конкретної публікації

Для отримання даних конкретної публікації, необхідно звернутись на сервер за адресою «/posts/{post}», де «{post}» - динамічна частина адреси, що містить у собі ідентифікатор публікації, яку сервер повинен надіслати у відповідь. Запит повинен мати тип «GET».

У відповідь сервер надішле 2 колекції:

- «post» - колекція даних публікації;
- «commentators» - колекція коментарів публікації.

На рис. 5.2 зображено приклад такого запиту та відповіді з детальним описом всіх атрибутів колекцій.

```

Illuminate\Http\Request {#44 ▾ // app\Http\Controllers
  attributes: Symfony\Component\HttpFoundation\ParameterBag {#46 ▾
    #parameters: []
  }
  request: Symfony\Component\HttpFoundation\InputBag {#45 ▾
    #parameters: []
  }
  query: Symfony\Component\HttpFoundation\InputBag {#52 ▾
    #parameters: []
  }
  server: Symfony\Component\HttpFoundation\ServerBag {#48 ▹}
  files: Symfony\Component\HttpFoundation\FileBag {#49 ▹}
  cookies: Symfony\Component\HttpFoundation\CookieBag {#47 ▹}
  headers: Symfony\Component\HttpFoundation\HeaderBag {#50 ▹}
  content: null
  languages: null
  charsets: null
  encodings: null
  acceptableContentTypes: null
  pathInfo: "/posts/30"
  requestUri: "/posts/30"
  baseUrl: ""
  basePath: null
  method: "GET"
  format: null
  session: Illuminate\Session\Store {#387 ▹}
  locale: null
  defaultLocale: "en"
  preferredFormat: null
  isHostValid: true
  isForwardedValid: true
  json: null
  convertedFiles: null
  userResolver: Closure($guard = null) {#334 ▹}
  routeResolver: Closure() {#343 ▹}
  basePath: ""
  format: "html"
}

array:2 [▾ // app\Http\Controllers\PostsController
  "post" => App\Models\Post {#739 ▾
    #attributes: array:13 [▾
      "id" => 30
      "title" => "Як отримати сесію cookie за"
      "intro" => "Яким чином можна отримати сесію"
      "mainText" => "Яким чином можна отримати сесію"
      "htmlText" => "<p>Яким чином можна отримати сесію"
      "user_id" => 25
      "amount_views" => 3
      "amount_comments" => 2
      "amount_likes" => 12
      "amount_dislikes" => 1
      "rating" => 12.0
      "created_at" => "2023-01-15 15:25:32"
      "updated_at" => "2023-05-11 11:32:02"
    ]
  }
  "commentators" => Illuminate\Database\Eloquent\Collection {#1141 ▾
    #items: array:2 [▾
      0 => App\Models\User {#1361 ▾
        #attributes: array:16 [▾
          "id" => 11
          "name" => "Nikita"
          "email" => "nekit@gmail.com"
          "posts_amount" => 0
          "email_verified_at" => null
          "amount_comments" => 0
          "password" => "$2y$10$UZ678SNqiTBMVV4pW"
          "two_factor_secret" => null
          "two_factor_recovery_codes" => null
          "two_factor_confirmed_at" => null
          "description" => null
          "username" => "@nekit"
          "photo" => null
          "remember_token" => null
          "created_at" => "2023-04-22 17:20:04"
          "updated_at" => null
        ]
      }
    ]
  }
]

```

Рисунок 5.2 – Приклад запиту (ліворуч) та відповіді (праворуч)

5.4 Запит пошуку публікацій

Для виконання пошуку публікацій, необхідно звернутись на сервер за адресою «/search». Запит повинен мати тип «GET». Для пошуку необхіден додатковий параметр, що містить пошуковий запит. У даному випадку це параметр «s». Повний зразок адреси: «/search?s=Ваш+запит».

У відповідь сервер надішле 3, вже відомі колекції:

- «posts» - колекція публікацій;
- «filters» - колекція фільтрів;
- «searchText» - текст пошукового запиту.

На рис. 5.3 зображено приклад такого запиту та відповідь з детальним описом всіх атрибутів колекцій.

```

Illuminate\Http\Request {#44 ▾ // app\Http\Controllers\PostsController.php
  +attributes: Symfony\Component\HttpFoundation\ParameterBag {#46 ▾
    #parameters: []
  }
  +request: Symfony\Component\HttpFoundation\InputBag {#45 ▾
    #parameters: []
  }
  +query: Symfony\Component\HttpFoundation\InputBag {#52 ▾
    #parameters: array:1 [▾
      "s" => "Python"
    ]
  }
  +server: Symfony\Component\HttpFoundation\ServerBag {#48 ▸}
  +files: Symfony\Component\HttpFoundation\FileBag {#49 ▸}
  +cookies: Symfony\Component\HttpFoundation\InputBag {#47 ▸}
  +headers: Symfony\Component\HttpFoundation\HeaderBag {#50 ▸}
  #content: null
  #languages: null
  #charsets: null
  #encodings: null
  #acceptableContentTypes: null
  #pathInfo: "/search"
  #requestUri: "/search?s=Python"
  #baseUrl: ""
  #basePath: null
  #method: "GET"
  #format: null
  #session: Illuminate\Session\Store {#388 ▸}
  #locale: null
  #defaultLocale: "en"
  -preferredFormat: null
  -isHostValid: true
  -isForwardedValid: true
  #json: null
  #convertedFiles: null
  #userResolver: Closure($guard = null) {#334 ▸}
  #routeResolver: Closure() {#343 ▸}
  basePath: ""
  format: "html"
}

array:3 [▾ // app\Http\Controllers\PostsController.php
  "posts" => Illuminate\Pagination\LengthAwarePaginator {#1143 ▾
    #items: Illuminate\Collection {#1357 ▾
      #items: array:6 [▾
        0 => App\Models\Post {#1359 ▸}
        1 => App\Models\Post {#1360 ▸}
        2 => App\Models\Post {#1361 ▸}
        3 => App\Models\Post {#1362 ▸}
        4 => App\Models\Post {#1363 ▸}
        5 => App\Models\Post {#1364 ▸}
      ]
    }
    #escapeWhenCastingToString: false
  }
  #perPage: 6
  #currentPage: 1
  #path: "http://127.0.0.1:8000/search"
  #query: []
  #fragment: null
  #pageName: "page"
  +onEachSide: 3
  #options: array:2 [▸]
  #total: 8
  #lastPage: 2
}
  "searchText" => "Python"
  "filters" => array:2 [▾
    "sortingType" => "rating"
    "sortingGrowth" => "desc"
  ]
]

```

Рисунок 5.3 – Приклад запиту (ліворуч) та відповіді (праворуч)

5.5 Запит на створення публікацій

Для створення нової публікації, необхідно звернутись на сервер за адресою «/post». Запит повинен мати тип «POST».

Ніяких додаткових атрибутів та параметрів не треба. В якості відповіді, сервер надішле інструкцію редіректу на сторінку з усіма публікаціями.

На рис. 5.4 зображено приклад такого запиту та детальний опис тих атрибутів, що має містити запит для успішного додавання публікації до БД.

```

App\Http\Requests\PostsRequest {#406 ▾ // app\Http\Controllers\PostsCo
+attributes: Symfony\...\ParameterBag {#408 ►}
+request: Symfony\...\InputBag {#405 ▾
  #parameters: array:6 [▾
    "_token" => "JzpenhCHYC4LXTdRr0XeebJ810oJEje64WopjcIX"
    "title" => "Нова публікація"
    "category_1" => "3"
    "category_0" => "11"
    "htmlText" => ""
    "mainText" => "Текст нової публікації. Текст нової публікації. Т
  ]
}
+query: Symfony\...\InputBag {#407 ►}
+server: Symfony\...\ServerBag {#411 ►}
+files: Symfony\...\FileBag {#410 ►}
+cookies: Symfony\...\InputBag {#409 ►}
+headers: Symfony\...\HeaderBag {#412 ►}
#content: "_token=JzpenhCHYC4LXTdRr0XeebJ810oJEje64WopjcIX&title=%D0

```

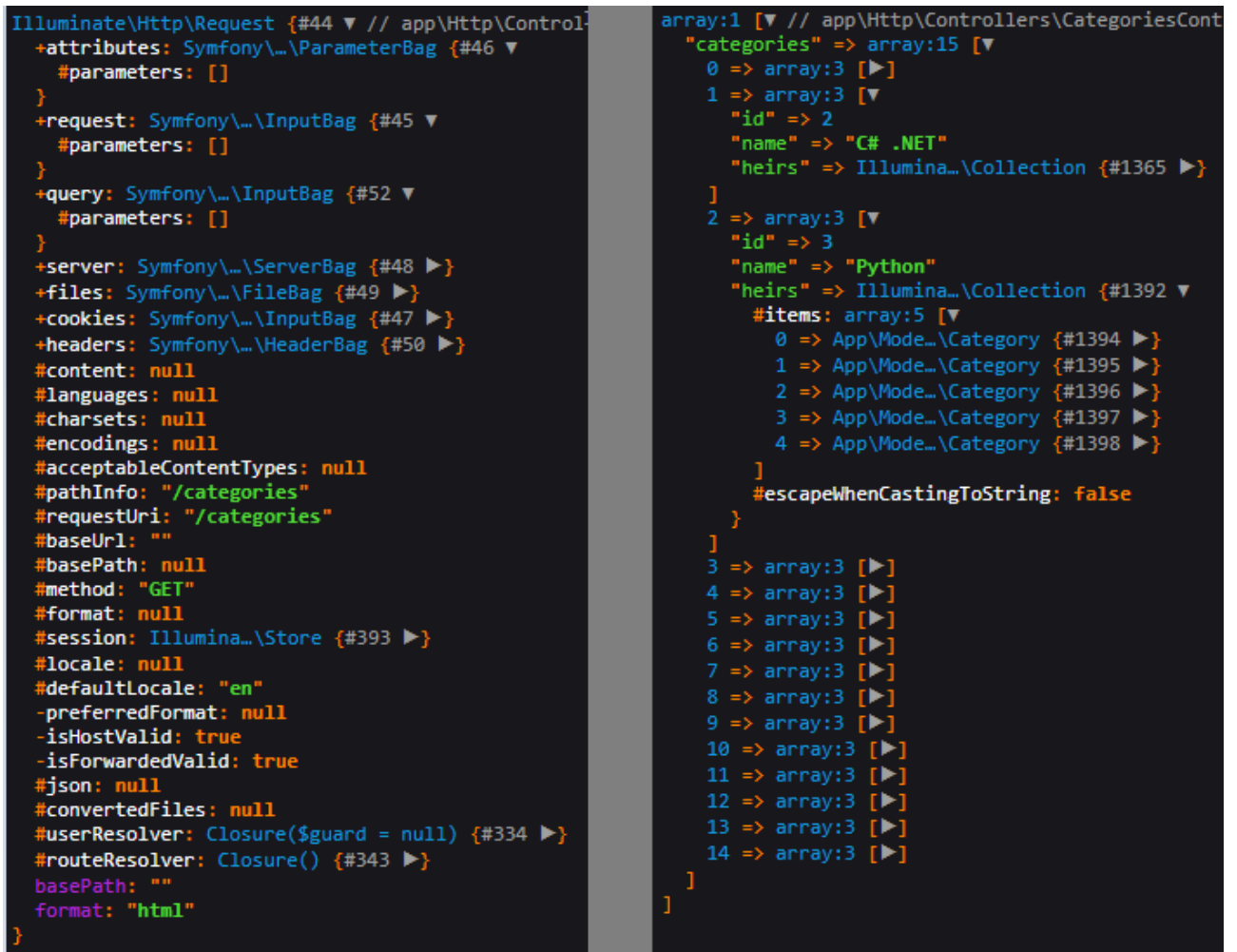
Рисунок 5.4 – Приклад запиту на створення публікації

5.6 Запит на отримання списку всіх категорій

Щоб отримати список усіх категорій, необхідно звернутись на сервер за адресою «/categories». Запит повинен мати тип «GET». Ніяких додаткових атрибутів та параметрів не треба.

У відповідь сервер надішле колекцію «categories», що міститиме масив категорій. Кожна з них матиме внутрішню колекцію «heirs», де знаходиться перелік підкатегорій та відповідні дані.

На рис. 5.5 зображено приклад такого запиту та відповідь з детальним описом всіх атрибутів колекцій.



```

Illuminate\Http\Request {#44 ▾ // app\Http\Control
  +attributes: Symfony\...\ParameterBag {#46 ▾
    #parameters: []
  }
  +request: Symfony\...\InputBag {#45 ▾
    #parameters: []
  }
  +query: Symfony\...\InputBag {#52 ▾
    #parameters: []
  }
  +server: Symfony\...\ServerBag {#48 ▸}
  +files: Symfony\...\FileBag {#49 ▸}
  +cookies: Symfony\...\InputBag {#47 ▸}
  +headers: Symfony\...\HeaderBag {#50 ▸}
  #content: null
  #languages: null
  #charset: null
  #encodings: null
  #acceptableContentTypes: null
  #pathInfo: "/categories"
  #requestUri: "/categories"
  #baseUrl: ""
  #basePath: null
  #method: "GET"
  #format: null
  #session: Illuminate\Store {#393 ▸}
  #locale: null
  #defaultLocale: "en"
  -preferredFormat: null
  -isHostValid: true
  -isForwardedValid: true
  #json: null
  #convertedFiles: null
  #userResolver: Closure($guard = null) {#334 ▸}
  #routeResolver: Closure() {#343 ▸}
  basePath: ""
  format: "html"
}

array:15 [▾ // app\Http\Controllers\CategoriesCont
  "categories" => array:15 [▾
    0 => array:3 [▸]
    1 => array:3 [▸]
    "id" => 2
    "name" => "C# .NET"
    "heirs" => Illuminate\Collection {#1365 ▸}
  ]
  2 => array:3 [▾
    "id" => 3
    "name" => "Python"
    "heirs" => Illuminate\Collection {#1392 ▾
      #items: array:5 [▾
        0 => App\Mode\...\Category {#1394 ▸}
        1 => App\Mode\...\Category {#1395 ▸}
        2 => App\Mode\...\Category {#1396 ▸}
        3 => App\Mode\...\Category {#1397 ▸}
        4 => App\Mode\...\Category {#1398 ▸}
      ]
      #escapeWhenCastingToString: false
    }
  ]
  3 => array:3 [▸]
  4 => array:3 [▸]
  5 => array:3 [▸]
  6 => array:3 [▸]
  7 => array:3 [▸]
  8 => array:3 [▸]
  9 => array:3 [▸]
  10 => array:3 [▸]
  11 => array:3 [▸]
  12 => array:3 [▸]
  13 => array:3 [▸]
  14 => array:3 [▸]
]

```

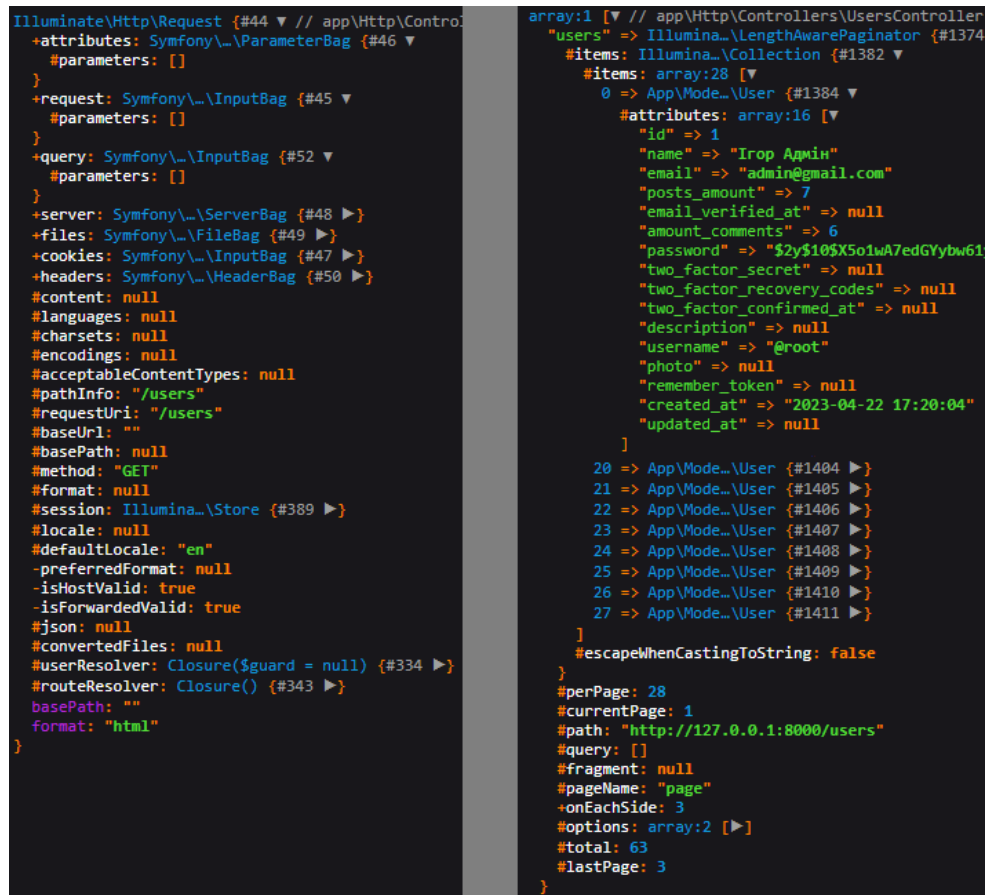
Рисунок 5.6 – Приклад запиту (ліворуч) та відповіді (праворуч)

5.7 Запит на отримання списку всіх користувачів

Щоб отримати список усіх користувачів, необхідно звернутись на сервер за адресою «/users». Запит повинен мати тип «GET». Ніяких додаткових атрибутів та параметрів не треба.

У відповідь сервер надішле колекцію «users», та дані для пагінації у разі, якщо користувачів у колекції буде забагато.

На рис. 5.7 зображено приклад такого запиту та відповідь з детальним описом всіх атрибутів колекцій.



```

Illuminate\Http\Request {#44 ▾ // app\Http\Controllers\UsersController
+attributes: Symfony\Component\HttpFoundation\ParameterBag {#46 ▾
#parameters: []
}
+request: Symfony\Component\HttpFoundation\InputBag {#45 ▾
#parameters: []
}
+query: Symfony\Component\HttpFoundation\InputBag {#52 ▾
#parameters: []
}
+server: Symfony\Component\HttpFoundation\ServerBag {#48 ▾
+files: Symfony\Component\HttpFoundation\FileBag {#49 ▾
+cookies: Symfony\Component\HttpFoundation\InputBag {#47 ▾
+headers: Symfony\Component\HttpFoundation\HeaderBag {#50 ▾
#content: null
#languages: null
#charsets: null
#encodings: null
#acceptableContentTypes: null
#pathInfo: "/users"
#requestUri: "/users"
#baseUrl: ""
#basePath: null
#method: "GET"
#format: null
#session: Illuminate\Session\Store {#389 ▾
#locale: null
#defaultLocale: "en"
-preferredFormat: null
-isHostValid: true
-isForwardedValid: true
#json: null
#convertedFiles: null
#userResolver: Closure($guard = null) {#334 ▾
#routeResolver: Closure() {#343 ▾
#basePath: ""
#format: "html"
}
}
}

array:1 [ ▾ // app\Http\Controllers\UsersController
"users" => Illuminate\Pagination\LengthAwarePaginator {#1374
#items: Illuminate\Collection {#1382 ▾
#items: array:28 [ ▾
0 => App\Models\User {#1384 ▾
#attributes: array:16 [ ▾
"id" => 1
"name" => "Irop Admin"
"email" => "admin@gmail.com"
"posts_amount" => 7
"email_verified_at" => null
"amount_comments" => 6
"password" => "$2y$10$X5o1wA7edGYybW61
"two_factor_secret" => null
"two_factor_recovery_codes" => null
"two_factor_confirmed_at" => null
"description" => null
"username" => "@root"
"photo" => null
"remember_token" => null
"created_at" => "2023-04-22 17:20:04"
"updated_at" => null
]
]
20 => App\Models\User {#1404 ▾
21 => App\Models\User {#1405 ▾
22 => App\Models\User {#1406 ▾
23 => App\Models\User {#1407 ▾
24 => App\Models\User {#1408 ▾
25 => App\Models\User {#1409 ▾
26 => App\Models\User {#1410 ▾
27 => App\Models\User {#1411 ▾
]
#escapeWhenCastingToString: false
}
#perPage: 28
#currentPage: 1
#path: "http://127.0.0.1:8000/users"
#query: []
#fragment: null
#pageName: "page"
+onEachSide: 3
#options: array:2 [ ▾
#total: 63
#lastPage: 3
}
]
}

```

Рисунок 5.7 – Приклад запиту (ліворуч) та відповіді (праворуч)

5.8 Висновки за розділом

Таким чином, в розділі було розглянуто основні принципи звернення клієнтського програмного забезпечення до розробленого застосунку.

Наведено декілька прикладів запитів до серверної частини та її відповіді. Проілюстровано структури даних, що використовуються у даній клієнт-серверній взаємодії з детальним описом атрибутів та параметрів.

Описано стислі інструкції щодо адрес, на яких розроблений застосунок веде прослуховування, та вказано, які саме типи запитів очікуються.

Дані інструкції повнні враховуватись під час розробки клієнтського додатка для його успішної взаємодії з сервером.

ВИСНОВКИ

У процесі виконання дипломної роботи було розроблено backend частину IT-форуму з деревоподібною архітектурою.

Розроблений застосунок має ряд переваг:

- налаштовану систему маршрутизації, що утворює зручне API для сторонніх клієнтських додатків;
- систему ролей користувачів та відповідні механізми керування доступами до функціонала форуму;
- сучасний зразок інформаційної архітектури додатка, що забезпечує гнучкість, масштабованість та структурність даних;
- проєкт розроблено за сучасними принципами та стандартами, що дозволяє дуже легко модифікувати додаток, вносячи в нього нові зміни та нарощуючи функціонал.

Після тестування додатка не виявлено помилок або нестабільності у роботі, що свідчить про коректне функціонування вебфоруму. Також виконано перевірку на предмет захисту даних.

Для розробки було використано мову програмування PHP з використанням фреймворку Laravel та СУБД MySQL. Проєктування структури додатка відбувалось за принципами MVC.

Розроблений вебфорум дозволить IT-спеціалістам з різних галузей шукати, створювати та обмінюватися корисною інформацією, що підвищить їх продуктивність роботи.

Форум враховує всі базові потреби у роботі з даними та надає відповідний набір інструментів користувачам. Має зручну систему пошуку та сортування публікацій, а також надає додаткові засоби для їх фільтрації. Надає можливість створювати публікації, коментувати, зберігати та каталогізувати їх.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Молчанов В. П. Технології розробки WEB-ресурсів / В. П. Молчанов, О. К. Пандорін. – Харків: ХНЕУ ім. С. Кузенця, 2019. – 130 с.
2. Субботін С. О. Архітектури та технології WEB-служб / С. О. Субботін. – Запоріжжя: ЗНТУ, 2016. – 23 с.
3. UBB Documentation [Electronic resource]. – Access mode: <https://www.ubbcentral.com/docs.php>
4. Пелешишин А. М. Аналіз існуючих типів віртуальних спільнот у мережі Інтернет та побудова моделі віртуальної спільноти на основі веб-форуму / А. М. Пелешишин, Р. Б. Кравець, Ю. О. Сєров. – Л.: НУЛП, 2011. – 11 с.
5. Сєров Ю. О. Аналіз комунікативних процесів у Веб-спільнотах середовища Веб 2.0 / Ю. О. Сєров, А. М. Пелешишин, К. О. Слобода. – Харків: Східно-Європейський журнал передових технологій, 2009. – № 1/2. – С. 38–41.
6. Возна Н. Я. Структуризація поліфункціональних даних: Теорія, методи та засоби / Н. Я. Возна. – Тернопіль.: ТНЕУ, 2018. – 378 с.
7. Грицунов О. В. Інформаційні системи та технології / О. В. Грицунов. – Х.: ХНАМГ, 2010. – 222 с.
8. Бойко Н. І. Моделювання WEB-орієнтованих систем та напрямків розвитку WEB-ресурсів / Н. І. Бойко. – Л.: ЛКА, 2012. – 10 с.
9. Stack Overflow All Questions [Electronic resource]. – Access mode: <https://stackoverflow.com/questions>
10. Sololearn Q&A Discussions [Electronic resource]. – Access mode: <https://www.sololearn.com/Discuss>
11. PHP Documentation [Electronic resource]. – Access mode: <https://www.php.net/manual/en/>
12. Composer Documentation [Electronic resource]. – Access mode: <https://getcomposer.org/doc/>

13. Laravel 9.x Documentation [Electronic resource]. – Access mode:
<https://laravel.com/docs/9.x>

14. MySQL Documentation [Electronic resource]. – Access mode:
<https://dev.mysql.com/doc/>

15. Git Documentation [Electronic resource]. – Access mode: <https://git-scm.com/doc>

ДОДАТОК А
Технічне завдання

A.1 Підстава для розробки

Підставою для розробки послужило завдання на дипломну роботу на тему «Програмна реалізація back-end частини IT-форуму», затверджений 04 квітня 2023 року № 87 по Національному університету «Запорізька політехніка».

A.2 Призначення розробки

Мета роботи - розробка серверної частини програмного продукту, що реалізує базовий функціонал IT-форуму, взаємодіє з базою даних та надає перелік ендпоінтів для клієнтської частини.

A.3 Основні вимоги до розроблювальної програми

A.3.1 Вимоги до надійності

Програма повинна забезпечувати коректну обробку помилок у наступних ситуаціях:

- при введенні користувачем неприпустимих значень;
- при невірній роботі з файлами (спробі доступу до файлу неприпустимого формату, спробі запису на захищений від запису диск та ін.).

A.3.2 Умови експлуатації

Під час роботи програми не рекомендується перезавантажувати ПК або вимикати живлення. Вихід із програми необхідно здійснювати послідовною зупинкою клієнтського й серверного сервісів.

Експлуатація програмного продукту здійснюється відповідно до експлуатаційної документації на розроблену програму, що відповідає стандартам і містить інформацію, необхідну для освоєння й експлуатації.

A.3.3 Вимоги до складу й параметрів технічних засобів

Необхідними вимогами до складу й параметрів технічних засобів, які повинні бути застосовані для роботи Інтернет-магазин, є:

- Оперативна пам'ять - не менш 4 Гб;
- HDD (жорсткий диск)- не менш 80 Гб;
- відео адаптер: DirectX 9 або пізніша версія з драйвером WDDM 1.0;
- процесор - не менш 1 ГГц
- мережний адаптер.

A.3.4 Вимоги до маркування й пакування

Програма може бути записана як на Flash-накопичувач, так і на компакт-диску. На пакуванні повинна бути назва програми - «Програмна реалізація back-end частини ІТ-форуму»

A.3.5 Вимоги до транспортування й зберігання

Вимоги до транспортування й зберігання аналогічні вимогам, які пропонуються до носія, на якому записана програма.

A.3.6 Вимоги до програмної документації

Програма повинна поставлятися з "Керівництвом користувача".

ДОДАТОК Б
Текст програми

Б.1 Текст модуля **BookmarksController.php**

```

<?php

namespace App\Http\Controllers;

use App\Models\Bookmark;
use App\Models\Like;
use App\Models\Post;
use App\Models\User;
use Illuminate\Http\Request;

class BookmarksController extends Controller
{
    public function index(User $user, Request $req)
    {
        if($req->user()->id == $user->id)
        {
            $bookmarks = $user->bookmarks()->paginate(5);

            return view('users.userBookmarks', compact('user', 'bookmarks'));
        } else {
            return redirect(url()->previous());
        }
    }

    public function destroy(User $user, Bookmark $bookmark)
    {
        $bookmark->delete();
        return redirect(url()->previous());
    }

    public function destroyInPost(Post $post, Bookmark $bookmark)
    {
        Like::decViewsAmountsForPost($post);
        $bookmark->delete();
        return redirect(url()->previous());
    }

    public function store(Post $post, Request $req)
    {
        Like::decViewsAmountsForPost($post);
        $post->bookmarks()->attach($post->id, Bookmark::getDataArrayForInsert($req));
        return redirect(url()->previous());
    }

    public function allDestroy(User $user, Request $req)
    {
        if($req->user()->id == $user->id)
        {
            Bookmark::where('user_id', $user->id)->delete();
        }
    }
}

```

```

    }
    return redirect(url()->previous());
}
}

```

Б.2 Текст модуля CategoriesController.php

```

<?php

namespace App\Http\Controllers;

use App\Http\Requests\CATEGORIESRequest;
use App\Models\Category;
use App\Models\Post;
use Illuminate\Http\Request;

class CategoriesController extends Controller
{
    public function index()
    {
        $categories = Category::getHeirs();
        return view('categories.index', compact('categories'));
    }

    public function create(Request $req)
    {
        if($req->user()->hasRole('admin'))
        {
            $basicCategories = Category::getBasicCategories();
            return view("categories.create", compact('basicCategories'));
        } else {
            return redirect(url()->previous());
        }
    }

    public function store(CATEGORIESRequest $req)
    {
        Category::create(Category::getArrData($req));

        return redirect('/categories');
    }

    public function destroy(Category $category, Request $req)
    {
        if($req->user()->hasRole('admin'))
        {
            $category->delete();
        }
        return redirect('/categories/controlPanel');
    }
}

```

```

public function editList()
{
    $categories = Category::getHeirs();
    return view('categories.controlPanel', compact('categories'));
}

public function edit(Category $category, Request $req)
{
    if($req->user()->hasRole('admin'))
    {
        return view('categories.edit', compact('category'));
    } else {
        return redirect(url()->previous());
    }
}

public function update(Category $category, CategoriesRequest $req)
{
    $category->update(['name' => $req->name]);
    return redirect('/categories/controlPanel');
}

public function show(Category $category, Request $req)
{
    $posts = $category->getPostsByCategory($req);
    $filters = Post::getFilterParams($req);
    return view('index', compact('posts', 'filters'));
}
}

```

Б.3 Текст модуля CommentsController.php

```

<?php

namespace App\Http\Controllers;

use App\Models\Comment;
use App\Models\Post;
use App\Models\User;
use Illuminate\Http\Request;

class CommentsController extends Controller
{
    public function store(Post $post, Request $req)
    {
        $post->comments()->attach($post->id, Comment::getDataArrayForInsert($req));

        $post->increment('amount_comments');
        User::find($req->user()->id)->increment('amount_comments');

        return redirect('/posts/'.$post->id);
    }
}

```

```

public function destroy(Post $post, Comment $comment, Request $req)
{
    if($req->user()->id == $comment->user_id || $req->user()->hasRole('admin'))
    {
        // transformation reply-comments into comments
        $answers = Comment::getReplyComments($comment->id);
        foreach ($answers as $answer){
            $answer->update(
                [
                    'reply_message_text' => NULL,
                    'reply_message_id' => NULL
                ]
            );
        }

        $comment->delete();
        $post->decrement('amount_comments');
        User::find($req->user()->id)->decrement('amount_comments');
        return redirect('/posts/'.$post->id);
    } else {
        return redirect('/');
    }
}

public function update(Post $post, Comment $comment, Request $req)
{
    if($req->user()->id == $comment->user_id)
    {
        // update intro in reply-message
        $answers = Comment::getReplyComments($comment->id);
        foreach ($answers as $answer){
            $answer->update(
                [
                    'reply_message_text' => Comment::generateIntro($req->mainText)
                ]
            );
        }

        $comment->update(request(['htmlText', 'mainText']))
        Comment::getReplyParamsForUpdate($req));
        return redirect('/posts/'.$post->id.'#'.$comment->id);
    } else {
        return redirect('/');
    }
}
}

```

Б.4 Текст модуля Controller.php

```

<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Routing\Controller as BaseController;

class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
}

```

Б.5 Текст модуля LikesController.php

```

<?php

namespace App\Http\Controllers;

use App\Models\Like;
use App\Models\Post;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class LikesController extends Controller
{
    public function store(Post $post, Request $req)
    {
        $post->likes()->attach($post->id, Like::getDataArrayForInsert($req));
        Like::incrementLikeAmounts($post, $req->grade);
        Like::decViewsAmountsForPost($post);
        return redirect('/posts/'.$post->id);
    }

    public function update(Post $post, Like $like, Request $req)
    {
        $like->update(Like::getDataArrayForInsert($req));
        Like::editLikeAmounts($post, $req->grade);
        Like::decViewsAmountsForPost($post);
        return redirect('/posts/'.$post->id);
    }

    public function destroy(Post $post, Like $like)
    {
        $like->delete();
        Like::decrementLikeAmounts($post, $like->grade);
        Like::decViewsAmountsForPost($post);
        return redirect('/posts/'.$post->id);
    }
}

```

```
}
```

Б.6 Текст модуля PostsController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Bookmark;
use App\Models\Category;
use App\Models\Like;
use App\Models\User;
use Illuminate\Http\Request;
use DB;
use App\Models\Post;
use App\Http\Requests\PostsRequest;
use Illuminate\Support\Facades\Auth;

class PostsController extends Controller
{
    public function index()
    {
        $posts = Post::getPosts($request, 'all');
        $filters = Post::getFilterParams($request);
        return view('index', compact('posts', 'filters'));
    }

    public function show(Post $post)
    {
        $post->increment('amount_views');
        $post->updateRating();
        $commentators = $post->comments->unique('id');

        if(Auth::check())
        {
            $bookmark = Bookmark::where('post_id', $post->id)->where('user_id',
Auth::id())->first();
            $userLike = Like::where('post_id', $post->id)->where('user_id', Auth::id())-
>first();
            return view('posts.show', compact('post', 'commentators', 'userLike',
'bookmark'));
        }
        return view('posts.show', compact('post', 'commentators'));
    }

    public function create()
    {
        $categories = Category::all();
        return view("posts.create", compact('categories'));
    }
}
```

```

public function store(PostsRequest $req)
{
    $user = User::find($req->user()->id);
    $post = $user->posts()->create(Post::getArrPostData($req));
    $post->setCategoriesByCreatePost($req);
    $user->increment('posts_amount');

    return redirect('/');
}

public function edit(Post $post, Request $req)
{
    if($req->user()->id == $post->user['id'] || $req->user()->hasRole('admin'))
    {
        $categories = Category::all();
        return view("posts.edit", compact('post', 'categories'));
    } else {
        return redirect('/');
    }
}

public function update(Post $post, PostsRequest $req)
{
    DB::table('category_post')->where('post_id', $post->id)->delete();
    $post->setCategoriesByCreatePost($req);
    $post->update(Post::getArrPostData($req));
    return redirect('/');
}

public static function deletePost(Post $post, Request $req)
{
    if($req->user()->id == $post->user['id'] || $req->user()->hasRole('admin'))
    {
        User::find($post->user_id)->decrement('posts_amount');
        $post->delete();
    }
}

public function destroy(Post $post, Request $req)
{
    self::deletePost($post, $req);
    return redirect('/');
}

public function destroyInUserProfile(Post $post, Request $req)
{
    self::deletePost($post, $req);
    return redirect(url()->previous());
}

public function search(Request $request)

```

```

    {
        $posts = Post::getPosts($request, 'search');
        $searchText = $request->s;
        $filters = Post::getFilterParams($request);
        return view('index', compact('posts', 'searchText', 'filters'));
    }
}

```

Б.7 Текст модуля UsersController.php

```

<?php

namespace App\Http\Controllers;

use App\Http\Requests\UsersRequest;
use App\Models\Post;
use App\Models\User;
use Illuminate\Http\Request;

class UsersController extends Controller
{
    public function index()
    {
        $users = User::paginate(28);
        return view('users.index', compact('users'));
    }

    public function show(User $user)
    {
        $posts = Post::query()
            ->where('user_id', $user->id)
            ->paginate(3);

        return view('users.userPosts', compact('user', 'posts'));
    }

    public function showUserComments(User $user)
    {
        $comments = $user->comments()->paginate(3);

        return view('users.userComments', compact('user', 'comments'));
    }

    public function edit(User $user, Request $req)
    {
        if($req->user()->id == $user->id)
        {
            return view('users.edit', compact('user'));
        } else {
            return redirect(url()->previous());
        }
    }
}

```

```

    }
}

public function update(User $user, UsersRequest $req)
{
    $user->updatePhoto($req);
    $user->update(User::getArrUserData($req));
    return redirect('/users/'.$user->id);
}
}

```

Б.8 Текст модуля CategoriesRequest.php

```

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class CategoriesRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, mixed>
     */
    public function rules()
    {
        return [
            'name' => 'required'
        ];
    }

    public function messages(){
        return [
            'name.required' => 'Поле імені не може бути порожнім',
        ];
    }
}

```

Б.9 Текст модуля PostsRequest.php

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class PostsRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, mixed>
     */
    public function rules()
    {
        return [
            'title' => 'required|min:2',
            'mainText' => 'required'
        ];
    }

    public function messages(){
        return [
            'title' => 'Назва публікації повинна мати не менше 2-ох символів',
            'mainText' => 'Текст публікації не може бути порожнім'
        ];
    }
}
```

Б.10 Текст модуля UsersRequest.php

```
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Auth;
```

```

use Illuminate\Validation\Rule;

class UsersRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, mixed>
     */
    public function rules()
    {
        return [
            'description' => 'max:50',
            'username' => ['nullable','sometimes','max:15','regex:/^[a-zA-Z0-9_]+$/'],
            Rule::unique('users')->ignore(Auth::user()->id),
            'photo' => 'mimes:jpg,bmp,png',
            'name' => 'required|max:15'
        ];
    }

    public function messages(){
        return [
            'description' => 'Занадто багато символів в описі профілю (допустиме
значення: не більше 50)',
            'username.max' => 'Занадто багато символів в юзернеймі (допустиме
значення: не більше 15)',
            'username.unique' => 'Такий юзернейм вже зайнятий',
            'regex' => 'Юзернейм може містити тільки букви, цифри та символ "_"',
            'photo' => 'Неправильний тип файлу (допустимі формати: jpg, png, bmp)',
            'name.required' => 'Поле імені не може бути порожнім',
            'name.max' => 'Поле імені містить забагато символів (допустиме значення:
не більше 15)'
        ];
    }
}

```

Б.11 Текст модуля Bookmark.php

```
<?php
```

```

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Bookmark extends Model
{
    use HasFactory;

    protected $fillable = ['user_id', 'post_id'];

    public static function getDataArrayForInsert($request)
    {
        return [
            'user_id' => $request->user()->id
        ];
    }
}

```

Б.12 Текст модуля Category.php

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    use HasFactory;
    protected $fillable = ['name', 'heir_id'];

    public function posts()
    {
        return $this->belongsToMany(Post::class, 'category_post', 'category_id', 'post_id');
    }

    public static function getBasicCategories()
    {
        return self::query()->whereNull('heir_id')->get();
    }

    public static function getHeirs()
    {
        $categories = [];
        $basicCategories = self::getBasicCategories();
        foreach ($basicCategories as $basicCategory){
            $categories[] = [
                'id' => $basicCategory->id,

```

```

        'name' => $basicCategory->name,
        'heirs' => self::query()->where('heir_id', $basicCategory->id)->get()
    ];
}
return $categories;
}

public static function getArrData($request)
{
    return [
        'name' => $request->name,
        'heir_id' => $request->heir_id
    ];
}

public function getPostsByCategory($request)
{
    if($request->type == null) {
        $request->type = 'rating';
    }

    if($request->growth == null) {
        $request->growth = 'desc';
    }

    return $this->posts()->orderBy($request->type, $request->growth)->paginate(6);
}
}

```

Б.13 Текст модуля Comment.php

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Comment extends Model
{
    use HasFactory;

    protected $fillable = ['user_id', 'post_id', 'mainText', 'htmlText', 'intro',
'reply_message_id', 'reply_message_text'];

    public static function generateIntro($text)
    {
        if (strlen($text) > 30)
        {

```

```

        $text = mb_strimwidth($text, 0, 30, "...");
    }
    return $text;
}

public static function getDataArrayForInsert($request)
{
    $dataForInsert = [
        'htmlText' => $request->htmlText,
        'mainText' => $request->mainText,
        'intro' => Comment::generateIntro($request->mainText),
        'user_id' => $request->user()->id
    ];

    if($request->reply_message_id != null)
    {
        $dataForInsert['reply_message_id'] = $request->reply_message_id;
        $dataForInsert['reply_message_text'] = $request->reply_message_text;
    }

    return $dataForInsert;
}

public static function getReplyParamsForUpdate($request)
{
    $dataForUpdate = [
        'reply_message_id' => $request->reply_message_id,
        'reply_message_text' => $request->reply_message_text,
        'intro' => Comment::generateIntro($request->mainText)
    ];

    if($request->reply_message_id == null)
    {
        $dataForUpdate['reply_message_id'] = NULL;
        $dataForUpdate['reply_message_text'] = NULL;
    }

    return $dataForUpdate;
}

public static function getReplyComments($comment_id)
{
    return Comment::where('reply_message_id', '=', $comment_id)->get();
}
}

```

Б.14 Текст модуля Like.php

<?php

```

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Like extends Model
{
    use HasFactory;

    protected $fillable = ['user_id', 'post_id', 'grade'];

    public static function getDataArrayForInsert($request)
    {
        return [
            'user_id' => $request->user()->id,
            'grade' => $request->grade
        ];
    }

    public static function incrementLikeAmounts(Post $post, $grade)
    {
        if($grade == True) {
            $post->increment('amount_likes');
            $post->increment('rating');
        } else {
            $post->increment('amount_dislikes');
        }
    }

    public static function decrementLikeAmounts(Post $post, $grade)
    {
        if($grade == True) {
            $post->decrement('amount_likes');
            $post->decrement('rating');
        } else {
            $post->decrement('amount_dislikes');
        }
    }

    public static function editLikeAmounts(Post $post, $grade)
    {
        if($grade == True) {
            $post->increment('amount_likes');
            $post->decrement('amount_dislikes');
            $post->increment('rating');
        } else {
            $post->decrement('amount_likes');
            $post->increment('amount_dislikes');
            $post->decrement('rating');
        }
    }
}

```

```

    public static function decViewsAmountsForPost(Post $post)
    {
        $post->decrement('amount_views');
        $post->update(['rating' => $post->rating - 0.25]);
    }
}

```

Б.15 Текст модуля Post.php

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    use HasFactory;

    protected $fillable = ['title', 'mainText', 'htmlText', 'user_id', 'intro'];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function comments()
    {
        return $this->hasManyMany(User::class, 'comments', 'post_id', 'user_id')->
            withPivot('id', 'mainText', 'htmlText', 'intro', 'reply_message_id',
'reply_message_text')->withTimestamps();
    }

    public static function generateIntro($text)
    {
        if (strlen($text) > 175)
        {
            $text = mb_strimwidth($text, 0, 175, "...");
        }
        return $text;
    }

    public static function getArrPostData($request)
    {
        return [
            'title' => $request->title,
            'mainText' => $request->mainText,
            'htmlText' => $request->htmlText,

```

```

        'user_id' => $request->user()->id,
        'intro' => Post::generateIntro($request->mainText)
    ];
}

public function likes()
{
    return $this->belongsToMany(User::class, 'likes', 'post_id', 'user_id')-
>withPivot('id', 'grade');
}

public function updateRating()
{
    $this->update(['rating' => $this->rating += 0.25]);
}

public static function getPosts($request, $mode)
{
    if($request->type == null) {
        $request->type = 'rating';
    }

    if($request->growth == null) {
        $request->growth = 'desc';
    }

    if($mode === 'all') {
        return self::orderBy($request->type, $request->growth)->paginate(6);
    } elseif ($mode === 'search') {
        return self::where('title', 'LIKE', "%{$request->s}%")->orderBy($request->type,
$request->growth)->paginate(6);
    }
}

public static function getFilterParams($request) {
    $filters = [];

    if($request->type == null) {
        $filters['sortingType'] = 'rating';
    } else {
        $filters['sortingType'] = $request->type;
    }

    if($request->growth == null) {
        $filters['sortingGrowth'] = 'desc';
    } else {
        $filters['sortingGrowth'] = $request->growth;
    }

    return $filters;
}

```

```

        public function bookmarks()
        {
            return $this->belongsToMany(User::class, 'bookmarks', 'post_id', 'user_id')-
>withPivot('id');
        }

        public function categories()
        {
            return $this->belongsToMany(Category::class, 'category_post', 'post_id',
'category_id');
        }

        public function setCategoriesByCreatePost($request)
        {
            $categAttributes = array_filter($request->all(), function($key) {
                return strpos($key, 'category') !== false;
            }, ARRAY_FILTER_USE_KEY);

            foreach ($categAttributes as $attribute){
                $this->categories()->attach($attribute);
            }
        }
    }
}

```

Б.16 Текст модуля User.php

```

<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Spatie\Permission\Traits\HasRoles;
use Image;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable, HasRoles;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
    ]
}

```

```

        'email',
        'password',
        'posts_amount',
        'description',
        'username'
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    public function posts()
    {
        return $this->hasMany(Post::class);
    }

    public function comments()
    {
        return $this->belongsToMany(Post::class, 'comments', 'user_id', 'post_id')->
            withPivot('id', 'mainText', 'htmlText', 'intro', 'reply_message_id',
'reply_message_text')->withTimestamps();
    }

    public function likes()
    {
        return $this->belongsToMany(Post::class, 'likes', 'user_id', 'post_id')->
withPivot('id', 'grade');
    }

    public static function getArrUserData($request)
    {
        return [
            'name' => $request->name,
            'description' => $request->description,
            'username' => self::getUsername($request->username)
        ];
    }
}

```

```

public static function getUsername($str)
{
    if(is_null($str)) {
        return null;
    } else {
        return '@'.$str;
    }
}

public function updatePhoto($request)
{
    if($request->hasFile('photo'))
    {
        $this->clearOldPhoto($this->id);

        $photo = $request->file('photo');
        $filename = time().'.'.$photo->getClientOriginalExtension();

        Image::make($photo)->fit(300, 300)->save(public_path($this-
>getPhotoPath($this->id)).$filename);

        $this->photo = $filename;
        $this->save();
    } elseif ($request->flagPhoto === "deleted") {
        $this->clearOldPhoto($this->id);
        $this->photo = null;
        $this->save();
    }
}

public function getPhotoPath($user_id)
{
    $path = "uploads/usersPhotos/id{$user_id}";

    if(!file_exists($path)) {
        if (!mkdir($path, 0777, true) && !is_dir($path)) {
            throw new \RuntimeException(sprintf('Directory "%s" was not created',
$path));
        }
    }

    return "$path/";
}

public function clearOldPhoto($user_id)
{
    $path = "uploads/usersPhotos/id{$user_id}";

    if(file_exists(public_path("$path")))
    {
        foreach(glob(public_path("$path/*")) as $photo)

```

```

        {
            unlink($photo);
        }
    }
}

public function bookmarks()
{
    return $this->belongsToMany(Post::class, 'bookmarks', 'user_id', 'post_id')-
>withPivot('id');
}
}

```

Б.17 Текст модуля 2014_10_12_000000_create_users_table.php

```
<?php
```

```

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->integer('posts_amount')->default(0);
            $table->timestamp('email_verified_at')->nullable();
            $table->integer('amount_comments')->default(0);
            $table->string('password');
            $table->string('description')->nullable();
            $table->string('username')->nullable()->unique();
            $table->string('photo')->nullable();
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
}

```

```

    */
    public function down()
    {
        Schema::dropIfExists('users');
    }
};

```

Б.18**Текст****модуля****2014_10_12_100_create_password_resets_table.php**

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('password_resets', function (Blueprint $table) {
            $table->string('email')->index();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('password_resets');
    }
};

```

Б.19**Текст****модуля****2014_10_12_200000_add_two_factor_columns_to_users_table.php**

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
use Laravel\Fortify\Fortify;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->text('two_factor_secret')
                ->after('password')
                ->nullable();

            $table->text('two_factor_recovery_codes')
                ->after('two_factor_secret')
                ->nullable();

            if (Fortify::confirmsTwoFactorAuthentication()) {
                $table->timestamp('two_factor_confirmed_at')
                    ->after('two_factor_recovery_codes')
                    ->nullable();
            }
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn(array_merge([
                'two_factor_secret',
                'two_factor_recovery_codes',
            ], Fortify::confirmsTwoFactorAuthentication() ? [
                'two_factor_confirmed_at',
            ] : []));
        });
    }
};

```

Б.20 Текст модуля 2019_08_19_000000_create_failed_jobs_table.php

```

<?php

```

```

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('failed_jobs', function (Blueprint $table) {
            $table->id();
            $table->string('uuid')->unique();
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')->useCurrent();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('failed_jobs');
    }
};

```

Б.21**Текст****модуля****2019_12_14_000001_create_personal_access_tokens_table.php**

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.

```

```

*
* @return void
*/
public function up()
{
    Schema::create('personal_access_tokens', function (Blueprint $table) {
        $table->id();
        $table->morphs('tokenable');
        $table->string('name');
        $table->string('token', 64)->unique();
        $table->text('abilities')->nullable();
        $table->timestamp('last_used_at')->nullable();
        $table->timestamp('expires_at')->nullable();
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('personal_access_tokens');
}
};

```

Б.22 Текст модуля 2023_01_04_173553_create_posts_table.php

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->string('intro');
            $table->text('mainText');
            $table->text('htmlText');

```

```

        $table->foreignId('user_id')->constrained();
        $table->integer('amount_views')->default(0);
        $table->integer('amount_comments')->default(0);
        $table->integer('amount_likes')->default(0);
        $table->integer('amount_dislikes')->default(0);
        $table->float('rating')->default(0.0);
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('posts');
}
};

```

Б.23 Текст модуля 2023_02_02_222044_create_comments_table.php

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('comments', function (Blueprint $table) {
            $table->id();
            $table->text('mainText');
            $table->text('htmlText');
            $table->text('intro');
            $table->foreignId('user_id')->constrained();
            $table->foreignId('post_id')->constrained()->onDelete('cascade');
            $table->foreignId('reply_message_id')->nullable()->references('id')-
            >on('comments');
            $table->text('reply_message_text')->nullable();
            $table->timestamps();
        });
    }
};

```

```

    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('comments');
    }
};

```

Б.24 Текст модуля 2023_02_14_233309_create_likes_table.php

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('likes', function (Blueprint $table) {
            $table->id();
            $table->boolean('grade');
            $table->foreignId('user_id')->constrained();
            $table->foreignId('post_id')->constrained()->onDelete('cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('likes');
    }
};

```

Б.25 Текст модуля 2023_04_08_144433_create_bookmarks_table.php

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('bookmarks', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained();
            $table->foreignId('post_id')->constrained()->onDelete('cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('bookmarks');
    }
};

```

Б.26 Текст модуля 2023_04_12_125003_create_categories_table.php

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *

```

```

* @return void
*/
public function up()
{
    Schema::create('categories', function (Blueprint $table) {
        $table->id();
        $table->text('name');
        $table->foreignId('heir_id')->nullable()->references('id')->on('categories')-
>onDelete('cascade');
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('categories');
}
};

```

Б.27**Текст****модуля****2023_04_12_135738_create_category_post_table.php**

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('category_post', function (Blueprint $table) {
            $table->id();
            $table->foreignId('category_id')->constrained()->onDelete('cascade');
            $table->foreignId('post_id')->constrained()->onDelete('cascade');
            $table->timestamps();
        });
    }
}

```

```

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('category_post');
}
};

```

Б.28 Текст модуля DatabaseSeeder.php

```

<?php

namespace Database\Seeders;

// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        // \App\Models\User::factory(10)->create();

        // \App\Models\User::factory()->create([
        //     'name' => 'Test User',
        //     'email' => 'test@example.com',
        // ]);
        $this->call(UsersTableSeeder::class);
        $this->call(PostsTableSeeder::class);
        $this->call(RolesTableSeeder::class);
        $this->call(Model_has_rolesTableSeeder::class);
        $this->call(CommentsTableSeeder::class);
        $this->call(LikesTableSeeder::class);
        $this->call(BookmarksTableSeeder::class);
        $this->call(CategoriesTableSeeder::class);
        $this->call(Category_postTableSeeder::class);
    }
}

```

Б.29 Текст модуля RolesTableSeeder.php

```

<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class RolesTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        DB::table('roles')->insert(
            [
                [
                    'name' => "user",
                    'guard_name' => "web"
                ],
                [
                    'name' => "admin",
                    'guard_name' => "web"
                ]
            ]
        );
    }
}

```

Б.30 Текст модуля UsersTableSeeder.php

```

<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;

class UsersTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
}

```

```

*/
public function run()
{
    DB::table('users')->insert(
        [
            [
                'name' => "Ігор Адмін",
                'email' => "admin@gmail.com",
                'username' => "@root",
                'posts_amount' => 7,
                'amount_comments' => 6,
                'created_at' => "2023-04-22 17:20:04",
                'password' => Hash::make("12345678")
            ],
            [
                'name' => "Дарина Гулько",
                'email' => "dashalyk@gmail.com",
                'username' => "@dasha12",
                'posts_amount' => 5,
                'amount_comments' => 7,
                'created_at' => "2023-04-22 17:20:04",
                'password' => Hash::make("qwerty123")
            ],
            [
                'name' => "Ваня Поняв",
                'email' => "ivan@gmail.com",
                'username' => null,
                'posts_amount' => 6,
                'amount_comments' => 7,
                'created_at' => "2023-04-22 17:20:04",
                'password' => Hash::make("11111111")
            ],
            [
                'name' => "Ваня НеПоняв",
                'email' => "ivannot@gmail.com",
                'username' => "@van11",
                'posts_amount' => 4,
                'amount_comments' => 5,
                'created_at' => "2023-04-22 17:20:04",
                'password' => Hash::make("11111111")
            ],
            [
                'name' => "Тетяна Кравчук",
                'email' => "iamtanya@gmail.com",
                'username' => null,
                'posts_amount' => 7,
                'amount_comments' => 5,
                'created_at' => "2023-04-22 17:20:04",
                'password' => Hash::make("www1111")
            ],
            [
                'name' => "Neo",

```

```

    'email' => "neo@gmail.com",
    'username' => "@qwert5",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("12345678")
  ],
  [
    'name' => "RudyGirl",
    'email' => "rigan@gmail.com",
    'username' => "@nokif",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("12312399")
  ],
  [
    'name' => "Tatatiana",
    'email' => "tinatina@gmail.com",
    'username' => "@polli",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("qwer1234")
  ],
  [
    'name' => "Zakhar",
    'email' => "foxy@gmail.com",
    'username' => "@ret567",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("asd12345")
  ],
  [
    'name' => "Andrey",
    'email' => "withcer@gmail.com",
    'username' => null,
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("zxc12345")
  ],
  [
    'name' => "Nikita",
    'email' => "nekit@gmail.com",
    'username' => "@nekit",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("11122233")
  ],
],

```

```
[
  'name' => "Juljaa",
  'email' => "julia@gmail.com",
  'username' => "@jue",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("dq12ee34")
],
[
  'name' => "Tonya",
  'email' => "toxa@gmail.com",
  'username' => null,
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("assa3993")
],
[
  'name' => "Akela",
  'email' => "akela@gmail.com",
  'username' => "@akela777",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("69966996")
],
[
  'name' => "Ketrin",
  'email' => "ketrin@gmail.com",
  'username' => "@joket4",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("010010011")
],
[
  'name' => "Bagheera",
  'email' => "bagheera@gmail.com",
  'username' => "@rokki",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("45563216")
],
[
  'name' => "Baloo",
  'email' => "baloo@gmail.com",
  'username' => "@asdf12",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
```

```

    'password' => Hash::make("86524673")
  ],
  [
    'name' => "Kaa",
    'email' => "kaa@gmail.com",
    'username' => "@apple",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("78963214")
  ],
  [
    'name' => "Shere Khan",
    'email' => "sherekhan@gmail.com",
    'username' => "@norri",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("93654782")
  ],
  [
    'name' => "Baloo",
    'email' => "baloo11@gmail.com",
    'username' => "@sss3",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("6543dfd21")
  ],
  [
    'name' => "Shere Khan",
    'email' => "shere_khan@gmail.com",
    'username' => null,
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("pfdffsword")
  ],
  [
    'name' => "Rikki-Tikki-Tavi",
    'email' => "rikki-tikki-tavi@gmail.com",
    'username' => null,
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("paewffword")
  ],
  [
    'name' => "Kaa",
    'email' => "kaa11@gmail.com",
    'username' => "@terter3",
    'posts_amount' => 0,

```

```

    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("Abc1ewfewf23")
  ],
  [
    'name' => "Chil",
    'email' => "chil@gmail.com",
    'username' => "@y6999",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("qwerweffty")
  ],
  [
    'name' => "Darzee",
    'email' => "darzee@gmail.com",
    'username' => null,
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("Passw0rd")
  ],
  [
    'name' => "Nagaina",
    'email' => "nagaina@gmail.com",
    'username' => "@kurt",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("password123")
  ],
  [
    'name' => "Rama",
    'email' => "rama@gmail.com",
    'username' => "@ramma",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("letmein")
  ],
  [
    'name' => "Duga",
    'email' => "duga@gmail.com",
    'username' => "@dosi4",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("g54gedsfr")
  ],
  [
    'name' => "Ohiohi",
    'email' => "ohiohi@gmail.com",

```

```

'username' => "@orioni",
'posts_amount' => 0,
'amount_comments' => 0,
'created_at' => "2023-04-22 17:20:04",
'password' => Hash::make("effgertgfre")
],
[
'name' => "Zed",
'email' => "zxc@gmail.com",
'username' => "@zed",
'posts_amount' => 0,
'amount_comments' => 0,
'created_at' => "2023-04-22 17:20:04",
'password' => Hash::make("dfdfdf3")
],
[
'name' => "Катя Краповець",
'email' => "mp41510@gmail.com",
'username' => "@gogo",
'posts_amount' => 0,
'amount_comments' => 0,
'created_at' => "2023-04-22 17:20:04",
'password' => Hash::make("56u67us45yt465uj")
],
[
'name' => "Сергій Сергійович",
'email' => "bukceline@gmail.com",
'username' => null,
'posts_amount' => 0,
'amount_comments' => 0,
'created_at' => "2023-04-22 17:20:04",
'password' => Hash::make("56u67us45yt465uj")
],
[
'name' => "Матвій Чепуренко",
'email' => "zarlengod@gmail.com",
'username' => null,
'posts_amount' => 0,
'amount_comments' => 0,
'created_at' => "2023-04-22 17:20:04",
'password' => Hash::make("56u67us45yt465uj")
],
[
'name' => "Константин Грушко",
'email' => "sandeep@gmail.com",
'username' => null,
'posts_amount' => 0,
'amount_comments' => 0,
'created_at' => "2023-04-22 17:20:04",
'password' => Hash::make("56u67us45yt465uj")
],
[

```

```

    'name' => "Дора",
    'email' => "noramfuller@gmail.com",
    'username' => "@lololo",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Якийсь хлопець",
    'email' => "bangtoyib2@gmail.com",
    'username' => "@dor3d",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Самсобі Андрій",
    'email' => "andersonasantos@gmail.com",
    'username' => "@rtg6",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Єва Солодун",
    'email' => "aubreybyfield@gmail.com",
    'username' => "@kikij7",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Аміна",
    'email' => "alesganz@gmail.com",
    'username' => "@zxcvb4",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Паляниця",
    'email' => "mcgrenaghan@gmail.com",
    'username' => "@redha",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ]

```

```

],
[
  'name' => "Ярмак Бова",
  'email' => "sanquel21@gmail.com",
  'username' => null,
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Калыш",
  'email' => "atkins618@gmail.com",
  'username' => "@artist",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Dosam Ali",
  'email' => "28youngpills@gmail.com",
  'username' => "@coder",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Lokki",
  'email' => "armanddarkmoon08@gmail.com",
  'username' => "@popo",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Igor",
  'email' => "wood0987@gmail.com",
  'username' => "@lokki",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Cat Cat",
  'email' => "bloodsik@gmail.com",
  'username' => "@wer344",
  'posts_amount' => 0,
  'amount_comments' => 0,

```

```

'created_at' => "2023-04-22 17:20:04",
'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Siao",
  'email' => "sainath@gmail.com",
  'username' => null,
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Dodoj",
  'email' => "athrun@gmail.com",
  'username' => null,
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Foggi",
  'email' => "got2scrap@gmail.com",
  'username' => null,
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Поля",
  'email' => "bharath@gmail.com",
  'username' => "@hohu",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Чайник Чаю",
  'email' => "rahgirl08@gmail.com",
  'username' => "@terra1",
  'posts_amount' => 0,
  'amount_comments' => 0,
  'created_at' => "2023-04-22 17:20:04",
  'password' => Hash::make("56u67us45yt465uj")
],
[
  'name' => "Дід Іван",
  'email' => "hardcorekicks@gmail.com",
  'username' => "@terra2",

```

```

    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Просто Дід",
    'email' => "maggiebit@gmail.com",
    'username' => null,
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Дід Панас",
    'email' => "carcynsmommy2008@gmail.com",
    'username' => "@okitoki33",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Дід Тарас",
    'email' => "shengyin1377@gmail.com",
    'username' => "@null1",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Арсен",
    'email' => "revitn1@gmail.com",
    'username' => null,
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Олексій Арестович",
    'email' => "ra1@gmail.com",
    'username' => null,
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Кчел",

```

```

    'email' => "watin@gmail.com",
    'username' => null,
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Самвел",
    'email' => "ivanbouchot@gmail.com",
    'username' => "@yuio",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Totok",
    'email' => "vcbnvbnvbbouchot@gmail.com",
    'username' => "@trtete",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Falisom",
    'email' => "tytyujtyouchot@gmail.com",
    'username' => "@uiuytu",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Петрович",
    'email' => "ivgfgfgfgt@gmail.com",
    'username' => "@afdf",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ],
  [
    'name' => "Dora Myra",
    'email' => "julio@gmail.com",
    'username' => "@etetet3",
    'posts_amount' => 0,
    'amount_comments' => 0,
    'created_at' => "2023-04-22 17:20:04",
    'password' => Hash::make("56u67us45yt465uj")
  ]
]

```

```

    });
}
}

```

Б.31 Текст модуля README.md

```

## About this project
This is a graduation project for students of the KNT-229 group. This is the
implementation of an IT forum that solves a number of shortcomings of existing forums.

```

```

</br>

```

```

</br>

```

```

</br>

```

```

## Installation and start

```

```

Run the following commands one by one before starting work on the project.

```

```

</br>

```

```

</br>

```

```

Install updated dependencies:

```

```

```bash

```

```

composer install

```

```

```

```

```

Install updated modules:

```

```

```bash

```

```

npm i

```

```

```

```

```

Database update:

```

```

```bash

```

```

php artisan migrate:refresh --seed

```

```

```

```

```

Start project:

```

```

```bash

```

```

php artisan serve

```

```

```

```

Б.32 Текст модуля .env

```

APP_NAME=Laravel

```

```

APP_ENV=local

```

```

APP_KEY=base64:7DoiNsKz6voj4+/ctVGxuPyjunJAa/kniBxeTOnKDSU=

```

```

APP_DEBUG=true

```

```

APP_URL=http://localhost

```

```

LOG_CHANNEL=stack

```

```

LOG_DEPRECATIONS_CHANNEL=null

```

```

LOG_LEVEL=debug

```

```

DB_CONNECTION=mysql

```

```
DB_HOST=127.0.0.1
DB_PORT=3307
DB_DATABASE=forumtestdb
DB_USERNAME=root
DB_PASSWORD=root
```

```
BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
```

```
MEMCACHED_HOST=127.0.0.1
```

```
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379
```

```
MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"
```

```
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false
```

```
PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_HOST=
PUSHER_PORT=443
PUSHER_SCHEME=https
PUSHER_APP_CLUSTER=mt1
```

```
VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
VITE_PUSHER_HOST="${PUSHER_HOST}"
VITE_PUSHER_PORT="${PUSHER_PORT}"
VITE_PUSHER_SCHEME="${PUSHER_SCHEME}"
VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Б.33 Текст модуля web.php

```
<?php
```

```

use App\Http\Controllers\BookmarksController;
use App\Http\Controllers\CategoriesController;
use App\Http\Controllers\LikesController;
use Illuminate\Support\Facades\Route;
use Illuminate\Support\Facades\DB;
use App\Http\Controllers\PostsController;
use App\Http\Controllers\UsersController;
use App\Http\Controllers\CommentsController;

Route::get('/', [PostsController::class, 'index']);

Route::group(['middleware' => ['auth']], function () {
    Route::get('/posts/create', [PostsController::class, 'create']);

    Route::get('/posts/{post}/edit', [PostsController::class, 'edit']);

    Route::patch('/posts/{post}', [PostsController::class, 'update']);

    Route::delete('/posts/{post}', [PostsController::class, 'destroy']);

    Route::delete('/posts/deleteInProfile/{post}', [PostsController::class,
'destroyInUserProfile']);

    Route::post('/posts/{post}/comment', [CommentsController::class, 'store']);

    Route::patch('/posts/{post}/comment/{comment}', [CommentsController::class,
'update']);

    Route::delete('/posts/{post}/comment/{comment}', [CommentsController::class,
'destroy']);

    Route::post('/posts/{post}/like', [LikesController::class, 'store']);

    Route::patch('/posts/{post}/like/{like}', [LikesController::class, 'update']);

    Route::delete('/posts/{post}/like/{like}', [LikesController::class, 'destroy']);

    Route::get('/users/{user}/edit', [UsersController::class, 'edit']);

    Route::patch('/users/{user}', [UsersController::class, 'update']);

    Route::get('/users/{user}/bookmarks', [BookmarksController::class, 'index']);

    Route::delete('/users/{user}/bookmarks/{bookmark}', [BookmarksController::class,
'destroy']);

    Route::post('/posts/{post}/bookmarks', [BookmarksController::class, 'store']);
    Route::delete('/posts/{post}/bookmarks/{bookmark}', [BookmarksController::class,

```

```

'destroyInPost']));

        Route::delete('/users/{user}/allDeleteBookmarks',      [BookmarksController::class,
'allDestroy']);

        Route::get('/categories/create', [CategoriesController::class, 'create']);
        Route::post('/categories/create', [CategoriesController::class, 'store']);
        Route::delete('/categories/{category}', [CategoriesController::class, 'destroy']);
        Route::get('/categories/{category}/edit', [CategoriesController::class, 'edit']);
        Route::patch('/categories/{category}/edit', [CategoriesController::class, 'update']);

        Route::get('/categories/controlPanel', [CategoriesController::class, 'editList']);
    });

    Route::get('/posts/{post}', [PostsController::class, 'show']);
    Route::post('/post', [PostsController::class, 'store']);
    Route::get('/search', [PostsController::class, 'search']);

    Route::get('/users', [UsersController::class, 'index']);

    Route::get('/users/{user}', [UsersController::class, 'show']);
    Route::get('/users/{user}/comments', [UsersController::class, 'showUserComments']);

    Route::view('/description', 'description.descriptionPage');

    Route::get('/categories', [CategoriesController::class, 'index']);

    Route::get('/categories/{category}/posts', [CategoriesController::class, 'show']);

    Auth::routes();

    Route::get('/home',      [App\Http\Controllers\HomeController::class,      'index'])-
>name('home');

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра

ПРОГРАМНА РЕАЛІЗАЦІЯ BACK-END ЧАСТИНИ ІТ-ФОРУМУ

Виконав
ст. гр. КНТ-229

Кирило СТУПАКОВ

Керівник
к. т. н., доцент

Тетяна ФЕДОРОНЧАК

Рисунок В.1 – Слайд вступу

Об'єкт та мета роботи

- Об'єкт розробки – backend частина Інтернет-форуму, присвяченого ІТ.
- Мета роботи – розробка серверної частини програмного продукту, що реалізує базовий функціонал ІТ-форуму, взаємодіє з базою даних та надає перелік ендпоінтів для клієнтської частини.

Рисунок В.2 – Слайд мети роботи

Задачі

- Аналіз предметної області;
- Розробка структури системи;
- Розробка схеми бази даних;
- Розробка системи маршрутизації;
- Розробка системи ролей користувачів;
- Розробка основних програмних компонентів;
- Тестування та рефакторинг розробленого додатка.

Рисунок В.3 – Слайд задач

Обґрунтування вибору мови програмування

| Критерій порівняння мов програмування | Мови програмування | | |
|---|--------------------|----|------|
| | PHP | C# | Java |
| Зручність використання у WEB-розробці | + | +- | + |
| Простота вивчення | +- | - | +- |
| Швидкість виконання коду | + | - | - |
| Різноманіття фреймворків для WEB-розробки | + | +- | +- |
| Підтримка багатьох баз даних | +- | + | +- |
| Підтримка більшістю операційних систем | + | +- | + |

Рисунок В.4 – Слайд вибору мови програмування

Обґрунтування вибору фреймворку

| Критерій порівняння PHP-фреймворків | PHP-фреймворки | | |
|---|----------------|---------|------|
| | Laravel | Symfony | Slim |
| Різноманітність функціоналу | + | +- | - |
| Простота використання | +- | - | + |
| Вбудований механізм міграцій бази даних | + | + | - |
| Вбудовані засоби системи маршрутизації | + | - | - |
| Велика кількість розширень та пакетів | + | + | - |
| Простота вивчення | - | +- | + |

Рисунок В.5 – Слайд вибору фреймворку

Концепція інформаційної архітектури форуму

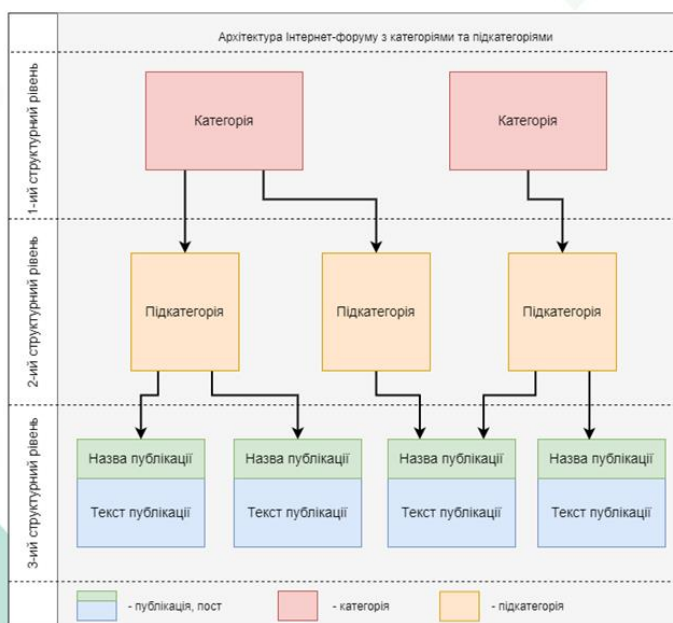


Рисунок В.6 – Слайд візуалізації інформаційної структури форуму

Схема взаємодії основних компонентів системи

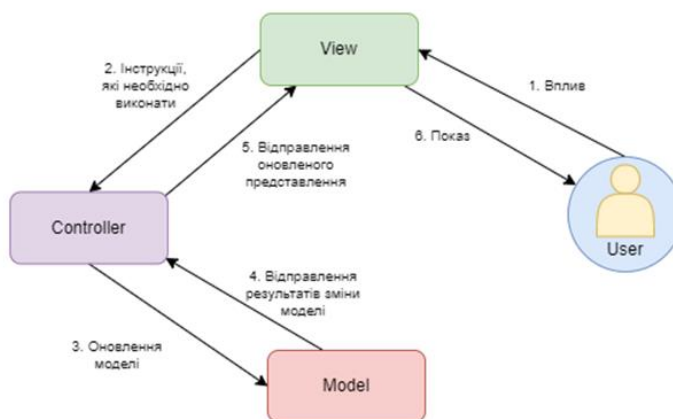


Рисунок В.7 – Слайд схеми взаємодії основних компонентів системи

Функціональна схема форуму для гостя



Рисунок В.8 – Слайд функціональної схеми форуму для гостя



Рисунок В.9 – Слайд функціональної схеми форуму для користувача



Рисунок В.10 – Слайд функціональної схеми форуму для адміністратора

Схема бази даних, що реалізує систему ролей на форумі

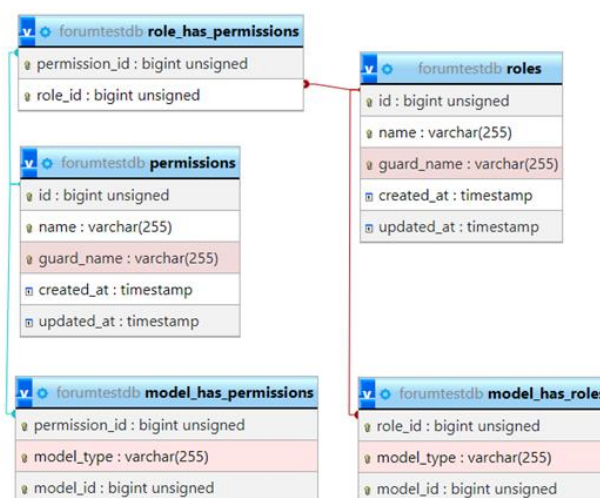


Рисунок В.11 – Слайд схеми бази даних, що реалізує СМ на форумі

Схема бази даних, що реалізує основну інформаційну структуру

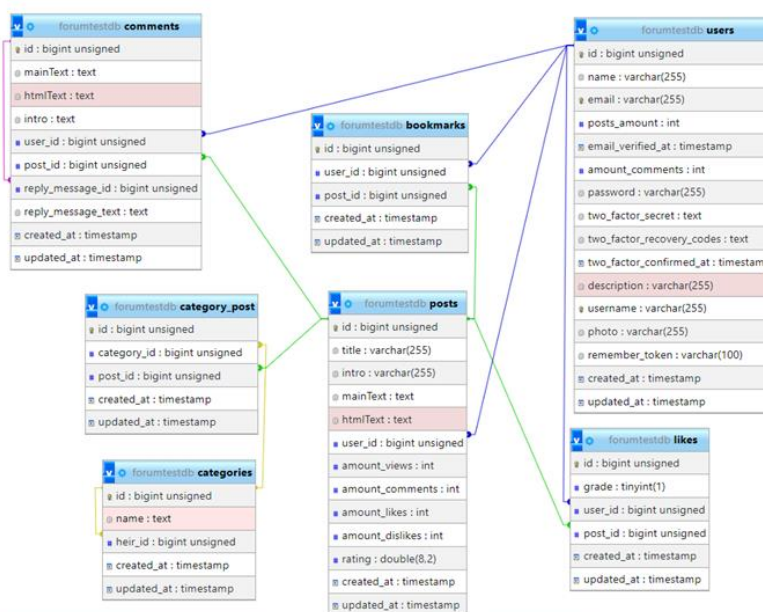


Рисунок В.12 – Слайд схеми бази даних, що реалізує основну інформаційну структуру

Приклад взаємодії з розробленим додатком, шляхом відправлення запиту

```
Illuminate\Http\Request {#444 ▼ // app\Http\Controllers\
  +attributes: Symfony\Component\HttpFoundation {#446 ▼
    +parameters: []
  }
  +request: Symfony\Component\HttpFoundation {#445 ▼
    +parameters: []
  }
  +query: Symfony\Component\HttpFoundation {#452 ▼
    +parameters: []
  }
  +server: Symfony\Component\HttpFoundation {#448 ▼
    +files: Symfony\Component\HttpFoundation {#449 ▼
    +cookies: Symfony\Component\HttpFoundation {#447 ▼
    +headers: Symfony\Component\HttpFoundation {#450 ▼
    +content: null
    +languages: null
    +charset: null
    +encoding: null
    +acceptableContentTypes: null
    +pathInfo: "/"
    +requestUri: "/"
    +baseUrl: ""
    +basePath: null
    +method: "GET"
    +format: null
    +session: Illuminate\Session {#379 ▼
    +locale: null
    +defaultLocale: "en"
    +preferredFormat: null
    +isHostValid: true
    +isForwardValid: true
    +json: null
    +convertedFiles: null
    +userResolver: Closure($guard = null) {#334 ▼
    +routeResolver: Closure() {#343 ▼
    +basePath: ""
    +format: "html"
  }
}
```

```
array:2 [▼ // app\Http\Controllers\PostsController.php:21
  "posts" => Illuminate\LengthAwarePaginator {#1134 ▼
    +items: Illuminate\Collection {#1348 ▼
      +items: array:6 [▼
        0 => App\Models\Post {#1350 ▼
          +connection: "mysql"
          +table: "posts"
          +primaryKey: "id"
          +keyType: "int"
          +incrementing: true
          +with: []
          +withCount: []
          +preventsLazyLoading: false
          +perPage: 15
          +exists: true
          +wasRecentlyCreated: false
          +escapeWhenCastingToString: false
          +attributes: array:13 [▼
            "id" => 30
            "title" => "Як отримати сесію cookie за допомогою request"
            "intro" => "Яким чином можна отримати cookie сесію за допомогою"
            "mainText" => "Яким чином можна отримати cookie сесію за допомогою"
            "htmlText" => "Яким чином можна отримати cookie сесію за допомогою"
            "user_id" => 25
            "amount_views" => 0
            "amount_comments" => 2
            "amount_likes" => 12
            "amount_dislikes" => 1
            "rating" => 11.25
            "created_at" => "2023-01-15 15:25:32"
            "updated_at" => null
          ]
        ]
      ]
    }
  ]
  +original: array:13 [▼
  ]
  +perPage: 6
  +currentPage: 1
  +path: "http://127.0.0.1:8000"
  +query: []
  +fragment: null
  +pageName: "page"
  +onEachSide: 3
  +options: array:2 [▼
    +total: 30
    +lastPage: 5
  ]
  +filters" => array:2 [▼
    "sortBy" => "rating"
    "sortByOrder" => "desc"
  ]
]
```

Рисунок В.13 – Слайд прикладу взаємодії з розробленим додатком

Висновки

Проведено дослідження предметної області, окреслено базові поняття, що стосуються об'єкта дослідження. Визначено основну суть об'єкта дослідження, його провідну ідею та цілі.

Обрано та обґрунтовано компоненти розробки. Спроектовано структуру системи на основі виявлених сутностей та функціональних потреб.

Створено повноцінне серверне програмне забезпечення з урахуванням сучасних принципів розробки, яке здатне ефективно взаємодіяти з клієнтськими додатками.

Застосунок передбачає всі базові потреби у роботі з даними та надає відповідний набір інструментів користувачам. Має зручну систему пошуку та сортування публікацій, а також містить додаткові засоби для їх фільтрації.

Рисунок В.14 – Слайд висновків