НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Кафедра автоматизації проектування енергетичних процесів і систем

КУРСОВА РОБОТА

з дисципліни: «Основи web-програмування - 1»
(назва дисципліни)
на тему: Розробка сайту-форуму

Студента(-ки) групи ТІ-81
напряму підготовки ***Програмна інженерія***
_____Соломаха О.О._
(прізвище та ініціали)
Керівник _____Титенко С.В._____
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна оцінка _____
Кількість балів: _____Оцінка: ECTS _____

Члени комісії _____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)
_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)
_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ- 2020 рік

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет
_ТЕПЛОЕНЕРГЕТИЧНИЙ_____
(повна назва)

Кафедра _автоматизації проектування енергетичних процесів та систем_____
(повна назва)

Напрям підготовки _Програмна інженерія_____
(шифр і назва)

З А В Д А Н Н Я
НА КУРСОВУ РОБОТУ СТУДЕНТУ

_____Соломасі Олександру Олександровичу_____
(прізвище, ім’я, по батькові)

1.Темароботи_____
Керівник курсової роботи – ст.вик. Титенко С.В._____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2.Строк подання студентом роботи _____

3. Вихідні дані до проекту (роботи): середовище розробки – _____

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Написання ТЗ	31.03.2020	
2	Розробка дизайну сайту	22.04.2020	
3	Розробка бета-версії сайту	06.05.2020	
4	Оформлення пояснювальної записки	26.05.2020	
5	Презентація курсової роботи	29.05.2020	
6			
7			

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник курсової роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

При виконанні курсової роботи на тему: «Розробка сайту-форуму», командою будо прийнято рішення розподілити роботу на 3 частини (дизайн, фронт-енд, бек-енд), при чому кожен з членів команди відповідав за свою частину, але усі головні рішення повинні обговорюватися та прийматися усім складом команди.

Склад команди:
Андрєєв Д.М. – фронт-енд
Нестеров М.А. – дизайн
Соломаха О.О. – бек-енд
У пояснювальній записці описується поетапне створення сайту для курсової роботи.

ANOTATION

During the course work on the topic: "Development of the site-forum", the team had decided to divide the work into 3 parts (design, front-end, back-end), with each member of the team responsible for their part, but all the main decisions should be discussed and accepted by the entire team.

Team members:
Andreev D.M. – front-end
Nesterov M.A. – design
Solomaha O.O. – back-end
The explanatory note describes the step-by-step creation of a site for course work.

ЗМІСТ

Вступ ...……..…...5
1. Теоретичні відомості …….....6
1.1. Основні етапи розробки сайту……...6
1.2. Інструменти використані для розробки……8
2. Тема проекту……...…....12
2.1. Постановка завдання……12
2.2. Інтернет Форум……14
2.3. Функціонал сайту…….…….……15
2.4. Створення Дизайну…….……..17
2.5. Опис засобів виконання індивідуальної роботи……21
2.6. Опис результатів загальної веб системи…….24
Висновок……..…..…25
Список використаних джерел……..……26
Додаток 1. Текст програми……...……27

Вступ

З часу створення першого сайту минуло чимало часу, і навколишній світ достатньо змінився. Сталося це завдяки розвитку технологій, появі нових язиків програмування та фреймворків, які давали змогу робити сайти кращими ніж вони були. По мірі розвитку технологій стали збільшуватися і бажання людей. Їм було недостатньо веб-сторінки, яка виглядає лише як документ. Стали з’являтися дизайни, інтернет подолав великий шлях з чорно – білого до того, яким ми його знаємо зараз. З появою смартфонів, люди й самі того не помітили як вони перейшли до того, що зараз більшість інтернет трафику йде з мобільних пристроїв. Так з’явилася ще більша потреба у адаптивній верстці і розробці мобільних версій сторінок, які нічим не уступають десктопним.

Основна частина

1.ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Основні етапи розробки сайту

Створення технічного завдання

Складанням технічного завдання для фахівців займається менеджер проекту. Робота з замовником починається з заповнення брифу, в якому замовник викладає свої побажання щодо візуального представлення і структури сайту, вказує на помилки в старій версії сайту, наводить приклади сайтів конкурентів. Виходячи з брифа, менеджер складає технічне завдання, враховуючи можливості програмних і дизайнерських засобів. Етап закінчується після затвердження технічного завдання замовником.

Дизайн основний і типових сторінок сайту

Починається робота зі створення дизайну, зазвичай в графічному редакторі. Дизайнер створює один або кілька варіантів дизайну, відповідно до технічного завдання. При цьому окремо створюється дизайн головної сторінки, і дизайни типових сторінок (наприклад: статті, новини, каталог продукції). Власне «дизайн сторінки» представляє собою графічний файл, листковий малюнок, що складається з найбільш дрібних картинок-шарів елементів загального малюнка.

При цьому дизайнер повинен враховувати обмеження стандартів HTML (не створювати дизайн, який потім не зможе бути реалізований стандартними засобами HTML). Виняток становить Flash-дизайн.

Кількість ескізів і порядок їх надання обмовляється з проект-менеджером. Також менеджер проекту здійснює контроль термінів. У великих веб-студіях в процесі бере участь арт-директор, який контролює якість графіки. Етап також закінчується затвердженням ескізу замовником.

HTML - верстка

Затверджений дизайн передається HTML-верстальщику, який «нарізає» графічну картинку на окремі малюнки, з яких згодом складає HTML-сторінку. В результаті створюється код, який можна переглядати за допомогою браузера. А типові сторінки згодом будуть використовуватися як шаблони програмування.

Далі готові HTML-файли передають програмісту.

Завершальним етапом розробки сайту є тестування

Процес тестування може включати в себе найрізноманітніші перевірки: вид сторінки зі збільшеними шрифтами, при різних розмірах вікна браузера, при відсутності флеш-плеєра і багато інших. Також – юзабіліті-тестування. Виявлені помилки відправляються на виправлення до тих пір, поки не будуть усунуті. Терміни контролює менеджер проекту. Також, на цьому етапі залучають до роботи дизайнера, щоб він провів авторський нагляд.

Розміщення сайту в Інтернеті

Файли сайту розміщують на сервері провайдера (хостингу) і виробляють потрібні налаштування. На цьому етапі сайт поки закритий для відвідувачів.

Сайт наповнюють контентом – текстами, зображеннями, файлами для скачування і т. Д. Іноді тексти складаються фахівцем студії, іноді контентом займається відповідальна особа з боку замовника. Це вирішується на етапі складання технічного завдання. У разі якщо контент складається представником студії, то це відбувається і затверджується паралельно з іншими етапами проекту.

1.2 Інструменти використані для розробки

React JS

React - JavaScript-бібліотека з відкритим вихідним кодом для розробки призначених для користувача інтерфейсів. React розробляється і підтримується Facebook, Instagram і співтовариством окремих розробників і корпорацій. React може використовуватися для розробки односторінкових і мобільних додатків.

React JS був створений у 2015 році компанією facebook, і з тих пір він набув значної популярності

Superagent

Бібліотека Superagent підходить для сучасних браузерів. Вона надає розробнику просте і зрозуміле API, з яким зручно працювати. Дана бібліотека має дуже велику кількість сильних сторін і майже позбавлена недоліків. До плюсів використання superagent можна додати:

- Піддається конфігурації.
- Має приємний інтерфейс для виконання HTTP-запитів.
- Підтримує об'єднання в ланцюжок декількох викликів для виконання запитів.
- Підтримує індикацію прогресу для вивантаження і завантаження даних.
- Підтримує механізм chunked-transfer encoding.
- Підтримує коллбекі.
- Для цієї бібліотеки розроблено безліч плагінів.

PHP

PHP - це широко використовувана мова сценаріїв загального призначення з відкритим вихідним кодом. Говорячи простіше, PHP це мова програмування, спеціально розроблений для написання web-додатків (сценаріїв), що

виконуються на Web-сервері. Абревіатура PHP означає "Hypertext Preprocessor (Препроцесор Гіпертексту)".

PHP був створений у 1994 році Расмусом Лердорфом коли той намагався дізнатися хто з роботодавців все ж таки читає його резюме. Почалося с того, що Лердорф розсилав потенційним роботодавцям своє міні-резюме з URL посиланням на його повну версію. Для того щоб зібрати інформацію о переходах на сторінку він створив CGI скрипт на мові програмування Perl і вставив його як тег до HTML коду своєї сторінки з резюме. З метою справлення враження на потенційних працедавців, Лердорф дозволив будь-якому відвідувачу переглядати зібрану статистику відвідувань. Цей код для збору статистики він назваа «PHP-Tools for Personal Home Page», оскільки сам використовував його на своїй персональній домашній сторінці. Декілька компаній були зацікавлені у отримані такого інструменту, і творець скріпту погодився надати його іншим особам. Так бере свій початок одна з найпопулярніших та найефективніших мов програмування для web-додатків – PHP.

До сильних сторін цієї мови можна віднести:

- Вивчення PHP не вимагає багато часу
- Кроссплатформенность. PHP може бути запущений в будь-якій операційній системі
- Підтримка веб-серверів. Складно знайти той, який би не працював з PHP
- Безкоштовне розповсюдження. Можливо, PHP не був такий популярний для створення web-додатків, якби не був безкоштовним, як і більшість інструментів для роботи з ним. Аналоги, які, в основному, виконують ту ж роботу, як правило коштують дорожче.
- Наявність великої кількості навчальних матеріалів.
- Неперервний розвиток

Laravel

Laravel - безкоштовний веб-фреймворк з відкритим кодом, призначений для розробки з використанням архітектурної моделі MVC. Laravel випущений під ліцензією MIT. Вихідний код проекту розміщується на GitHub. Також Laravel є найпопулярнішим серед php фреймворків, в свою чергу php є найпопулярнішою мовою веб-програмування, якщо не найпопулярнішою серед всіх мов програмування а не тільки веб. Це робить Laravel найпопулярнішим фреймворком на сьогодні. Сьогодні найбільше проектів, що розробляються за допомогою фреймворків, створюються саме з використанням Laravel. При розміщенні вакансій PHP-розробників, веб-студії все частіше включать знання фреймворка Laravel як обов'язкову умову. Такої популярності Laravel досяг через:

- високий рівень безпеки
- Підвищена продуктивність. Кешування
- Міграції баз даних
- MVC-архітектура
- Помилки і виключення

І це навіть не половина з того, за що програмісти обирають саме Laravel.

PHPStorm

Підтримка популярних фреймворків, підсвітка синтаксису, автодоповнення коду, автоматичний рефактор, аналазітор коду та багато іншого. Зменшує кількість часу розробника на розробку, пошук документації та дебагінгу

Figma

Figma – це онлайн-сервіс для розробки інтерфейсів і прототипування з можливістю організації спільної роботи в режимі реального часу. Сервіс має широкі можливості для інтеграції з корпоративним месенджером Slack і інструментом для високорівневого прототипирования Framer. Дизайнеру

важливі швидкість роботи над проектом і комунікація в процесі. Ці та багато інших завдань дозволяє вирішити онлайн-сервіс Figma. У Figma низький поріг входження і потужний потенціал при уявній простоті.

2. ТЕМА ПРОЕКТУ

2.1. Постановка завдання

Для виконання курсової роботи пропонувалось обрати собі команду з 2-3 людей та обрати тему, яка була би цікава для всіх членів команди. Таким чином була обрана тема розробки – «інтернет форум», і зібраний такий склад команди розробників:

- Андрєєв Д.М.
- Нестеров М.А.
- Соломаха О.О.

Після обговорення та обмірковування теми, було створено технічне завдання, яке би відображало усі основні задачі та вимоги до майбутнього сайту (веб-додатку). Технічне завдання було презентовано та узгоджено з керівником курсової роботи. До основних вимог сайту входило: розробка концептуальної моделі сайту, створення дизайну за розробленою моделлю, верстка сторінок за дизайном, розміщення кінцевої версії сайту на сервері(хостингу). До технічних вимог були віднесені такі аспекти, як адаптивність програмного продукту під різні масштаби та співвідношення екрану, а також його підтримка більшою кількістю сучасних веб браузерів (Google Chrome, Opera, Mozilla firefox).

Окрім цього, були описані функціональні можливості користувачем на різних сторінках веб-додатку. Так для розділу «усі питання», який за сумісністю також є і головною сторінкою, було визначено таке: має містити перелік усіх питань, які були створені на сайті, та можливість відсортувати їх за кількістю переглядів, або датою створення; має містити навігаційне меню по сайту, строку пошуку запитань за їх назвою, кнопку для створення питання. Для розділу із авторизацією користувачів було описане таке: можливість реєстрації (створення аккаунту) або логіну до вже існуючого аккаунту за допомогою електронної пошти та паролю, авторизація користувачів необхідна задля можливості створення своїх питань, або

коментування питань інших; звичайний користувач, що війшов як гість має можливість тільки переглядати контент без можливості створення або редагування. На сторінці профілю передбачено дії: зміни паролю, видалення існуючого аккаунту, зміни аватару, написання інформації про себе, зміна пошти та нікнейму. За головну мету поставлено досягнення найбільшої зручності у використанні продукту кінцевим користувачем.

2.2. ІНТЕРНЕТ ФОРУМ

Суть роботи форуму полягає в створенні користувачами (відвідувачами форуму) своїх тем з їх подальшим обговоренням, шляхом розміщення повідомлень всередині цих тем. Користувачі можуть коментувати заявлену тему, задавати питання по ній і отримувати відповіді, а також самі відповідати на питання інших користувачів форуму і давати їм поради. Питання і відповіді зберігаються в базі даних форуму, і в подальшому можуть бути корисні як учасникам форуму, так і будь-яким користувачам мережі Інтернет, які можуть зайти на форум, знаючи адресу сайту, або отримавши його від пошукових систем при пошуку інформації. Тематика форумів може бути найрізноманітнішою, охоплюючи всі сфери життя, і визначається або власниками форуму або його адміністрацією. У випадку із, створюваним нашою командою, форумом була обрана тематика програмування. Тобто сайт орієнтований на програмістів, при чому не на всіх, а тільки на Android, Java та web розробників.

2.3. Функціонал сайту

Приймаючи до уваги те, що тематика розроблюваного сайту – інтернет форум, наша команда вирішила перед розробкою визначити головний функціонал який повинен бути реалізован:

- Можливість створити питання
- Можливість залишати відповіді
- Створення аккаунту
- Авторизація
- Зміна паролю
- Видалення аккаунту
- Сортування питань за кількістю переглядів, датою створення, популярністю
- Пошук питань
- Можливість редагування вже написаних відповіді та питань
- Можливість надання привілеїв адміну для редагування або видалення відповідей та питань, інформації о користувачах та їх аккаунтах.
- Додання аватарів до профілю

Усі ці можливості було успішно додано до сайту, а також протестовано. Таким чином сайт задовольняє усім необхідним потребам користувачів. Реалізовано це було за допомогою бази даних:

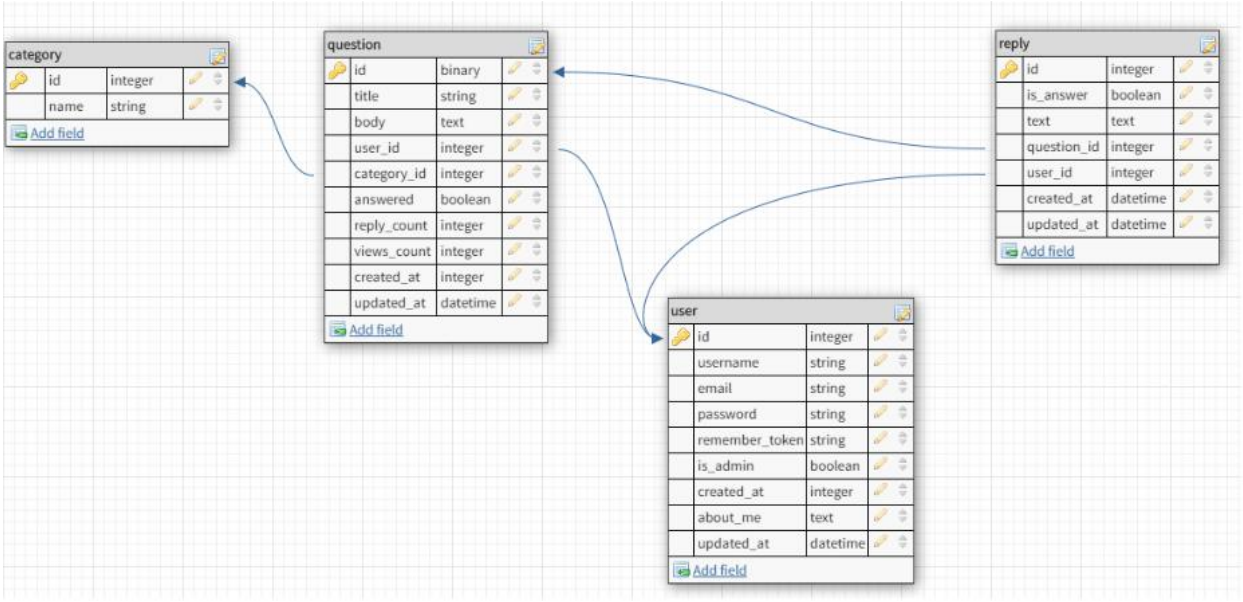

Рисунок 2.3.1 База даних сайту

2.4 Створення Дизайну

Одним з перших етапів розробки, після написання ТЗ, було створення дизайну сайту, а саме його головної сторінки, сторінки питання та сторінки профілю у обох форматах (як мобільний так і десктопний). Дизайн було узгоджено усією командою. Створено дизайн було у графічному редактору Figma.

Через те що основною функцією сайту є створення та обговорення питань а також пошук інформації, було вирішено зробити мінімалістичний дизайн без яскравих кольорів. Для цього було обрано 3 основні кольори (білий, сіро-чорний, сіро-синій).

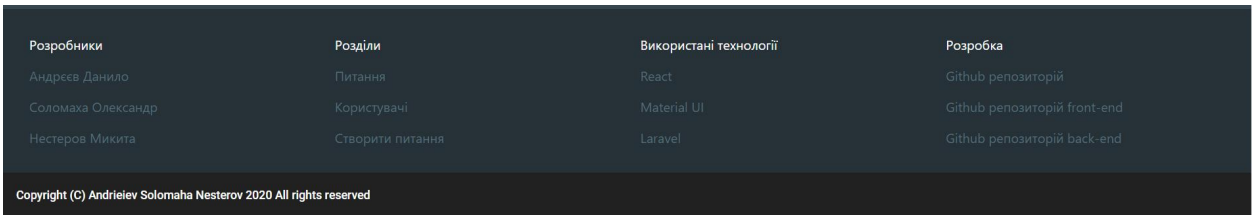

Рисунок 2.4.1 Футер сайту для демонстрації кольорової палітри

Для зручної навігації по сайту зроблено меню зліва.

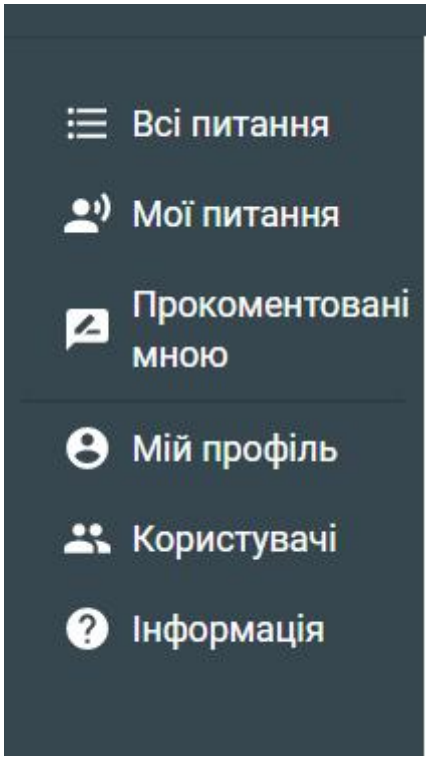

Рисунок 2.4.2 Меню сайту

Як видно з рисунку меню складається з шести пунктів:

- Всі питання – веде на головну стоінку, де можна побачити перелік питань, які були задані
- Мої питання – сторінка, що відображає питання створені користувачем
- Прокоментовані мною – сторінка, де знаходяться питання на які користувач залишив відповіді
- Мій профіль – сторінка, що містить інформацію о користувачеві
- Користувачі – перелік всіх зареєстрованих юзерів
- Інформація – коротко про проект

Слід зазначити, що не всі функції доступні неавторизованим користувачам, наприклад вони не можуть створювати та коментувати питання. Тому навігаційне меню відрізняється для авторизованих та неавторизованих юзерів. На рисунку віще було зображено повне меню (Для перегляду обмеженого меню дивитись рисунок 2.3.3).

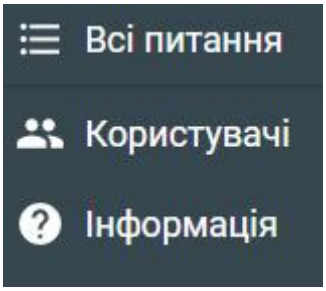

Рисунок 2.4.3 Меню для не авторизованих користувачів

Для сайту був зроблений логотип за допомогою графічного редактору Adobe Photoshop і обрано назву «Readit» співзвучну з назвою вже популярного форуму «Reddit». Даний вибір зумовлен тим, що така назва повинна підвищити популярність сайту.

Рисунок 2.4.4. Логотип сайту

Логотип знаходиться як у хедері сайту так і на інформаційній сторінці. Також у хедері можна побачити кнопку «logout» ім’я користувача та аватар (якщо авторизований), або кнопки «Sign up» та «Login» (якщо користувач не увійшов до свого профілю)

Рисунок 2.4.5. Хедер для авторизованого користувача

Рисунок 2.4.6. Хедер для невідомого користувача

При натисканні на кнопку «Sign up» або «Login» з’явиться вікно з полями для введення даних.

Рисунок 2.4.7 Вікно реєстрації

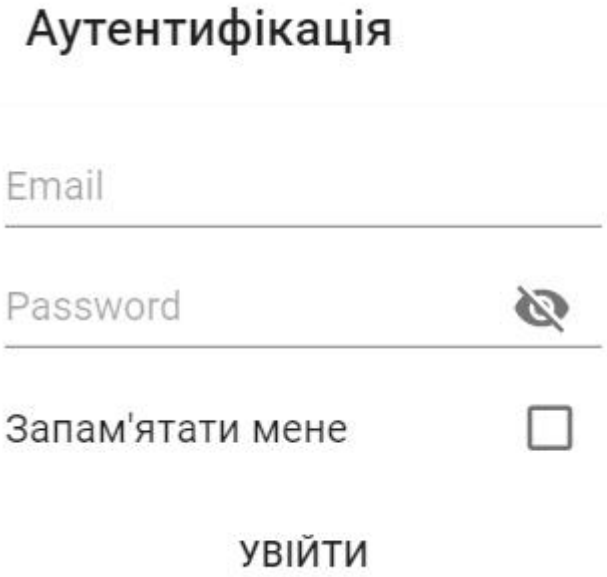

Рисунок 2.4.8 Вікно авторизації

2.5. Опис засобів виконання індивідуальної роботи

Після узгодження дизайну, інтерфейсу та функціоналу сайту було розроблено попередню структуру REST-API з вказанням роутів, методів, вхідних та вихідних даних у вигляді DTO. Кінцевий варіант виглядає наступним чином:

Method	URI	Action
POST	api/auth/login	App\Http\Controllers\Auth\LoginController@login
POST	api/auth/logout	App\Http\Controllers\Auth\LoginController@logout
POST	api/auth/register	App\Http\Controllers\Auth\RegisterController@register
GET|HEAD	api/categories	App\Http\Controllers\CategoryController@index
GET|HEAD	api/categories/{category}	App\Http\Controllers\CategoryController@show
POST	api/questions	App\Http\Controllers\QuestionController@store
GET|HEAD	api/questions	App\Http\Controllers\QuestionController@index
GET|HEAD	api/questions/commented	App\Http\Controllers\QuestionController@commented
GET|HEAD	api/questions/my	App\Http\Controllers\QuestionController@my
GET|HEAD	api/questions/top10	App\Http\Controllers\QuestionController@top10
GET|HEAD	api/questions/{question}	App\Http\Controllers\QuestionController@show
PUT|PATCH	api/questions/{question}	App\Http\Controllers\QuestionController@update
DELETE	api/questions/{question}	App\Http\Controllers\QuestionController@destroy
GET|HEAD	api/questions/{question}/replies	App\Http\Controllers\ReplyController@index
POST	api/questions/{question}/replies	App\Http\Controllers\ReplyController@store
POST	api/questions/{question}/replies/{reply}/markAsAnswer	App\Http\Controllers\ReplyController@markAsAnswer
PUT|PATCH	api/replies/{reply}	App\Http\Controllers\ReplyController@update
DELETE	api/replies/{reply}	App\Http\Controllers\ReplyController@destroy
POST	api/users	App\Http\Controllers\UserController@store
GET|HEAD	api/users	App\Http\Controllers\UserController@index
GET|HEAD	api/users/me	App\Http\Controllers\UserController@me
DELETE	api/users/{user}	App\Http\Controllers\UserController@destroy
PUT|PATCH	api/users/{user}	App\Http\Controllers\UserController@update
GET|HEAD	api/users/{user}	App\Http\Controllers\UserController@show
POST	api/users/{user}/avatar	App\Http\Controllers\UserController@avatar

Рисунок 2.5.1 Кінцевий список роутів REST API

Задля універсальності та уніфікованості інтерфейсу спілкування використано саме архітектурний стиль REST API.

Було розроблено базовий REST застосунок для демонстрації бета-версії із максимальним використанням вбудованого у Laravel функціоналу. Наприклад, автоматична генерація коду за допомогою CLI-застосунку Artisan:

- одночасно міграцій та моделі;
- сидери (seeders) для бази даних;
- проміжні обробники (middleware)
- API-ресурси та контролери

Протоколом спілкування фронтенду та бекенду було обрано JSON як найпопулярніший формат. Для форсування JSON запитів та відповідей було

створено middleware під назвою ForceJsonResponse. Усі помилки, в тому числі помилки валідації відправляються у форматі JSON з відповідним до стандарту HTTP статус-кодом.

Схему відповідей сервера описують та контролюють файли API-ресурсів, що вбудовані у фреймворк Laravel та надають легкий інтерфейс контролю даних. Для кожної сутності існує по 2 файли, наприклад, User та ViewUser, для того, щоб віддавати повну інформацію про користувача та пов'язані сутності лише при окремому запиті.

Для аутентифікації було використано шаблонні файли, що генеруються командою під час створення нового Laravel проект за допомогою прапора –auth. Нажаль, згенеровані контролери повністю не вирішують цю проблему, тому було перезавантажено необхідні методи.

Після успішної авторизації сервер встановлює http-only secure cookie. Таке рішення суперечить постулатам REST API, але було обране через більший захист до XSS атак. Таким чином, злочинець, отримавши доступ до JS на стороні клієнта (що наразі суттєва проблема через низьку якість node-modules) не зможе змінити ці cookies. До того ж, в майбутньому, можна легко додати аутентифікацію за допомогою JWT, наприклад, для мобільного застосунку чи іншого фронтенду.

Однією з проблем, з якою зіткнулися під час розробки та деплою - CORS обмеження. Було вирішено додаванням певного `access-control-allow-origin` до заголовків у відповіді серверу. Також, через використання авторизації за допомогою cookie, додаємо заголовок `access-control-allow-credentials: true`.

У користувачів можливо 2 ролі: звичайний користувач та адміністратор. Для авторизації використано функціонал Laravel під назвою Policies. Наприклад, обмеження для редагування, видалення питання, коментарю чи профілю є лише у автора та адміністратора. Laravel дозволяє без великої кількості коду прив'язати Policy деякої моделі до Resource контролеру та зручно проводити авторизацію, не змінюючи код екшену контролера.

Значну кількість часу та коду економить Eloquent ORM, що використовуємо для опису моделей, їх зв’язку, міграцій та слідкуванням за цілісністю даних.

Для заповнення бази даних тестовими даними використовували database seeders. Вони дозволяють заповнити базу, наприклад, десятьма категоріями, тридцятьма питаннями та по п’ять питань до кожного питання.

Після завершення основної частини завдання, було вирішено задеплоїти серверну частину на власний сервер з операційною системою Ubuntu. В якості веб серверу обрано highload-сервер nginx. Для середовища PHP використовується php-fpm. Статичні файли, а саме аватари користувачів, віддаються напряму, в обхід PHP та з правильними заголовками для кешування.

2.6. Опис результатів загальної веб системи

Веб система являє собою клієнт-серверний додаток, в якому клієнтську частину реалізує браузер, який здійснює діалог з користувачем і відображення інформації, а серверну частину - веб-сервер і сервер додатків, які реалізують основну логіку системи.

У розробленій веб системі, реалізовано усі функції які вже були описані вище у документі (див. розділ 2.3 Функціонал сайту). У системі є свого роду ієрархія користувачів, яка полягає у тому, що кожен користувач залежно від його статусу на форумі може або навпаки не має можливості виконувати ті чи інші дії. Таким чином є три групи користувачів: гісті, авторизовані користувачі (надалі юзери) та адміністратори. Якщо говорити про цю ієрархію як про наслідування класів в об’єктно орієнтованому програмуванні, то тоді можна уявити гостя (неопізнаного користувача) як батьківській клас, тоді юзер буде похідним класом від гостя, а адміністратор в свою чергу – похідним від юзера.

У гостя на нашому сайті є можливості переглядати публічний контент (питання, створені юзерами, та відповіді на них; перелік усіх юзерів сайту; інформація у профілі; інформація про сайт) та зареєструватися або авторизуватися (увійти у вже створений акаунт). У юзера додаються до цих можливостей ще такі, як створення питань, залишання коментарів (відповідей) на питання інших або власні, редагування свого профілю. Що стосується адміну, то він може змінювати будь-яку інформацію створену юзерами або видаляти її, також має можливість видалити акаунт іншого користувача.

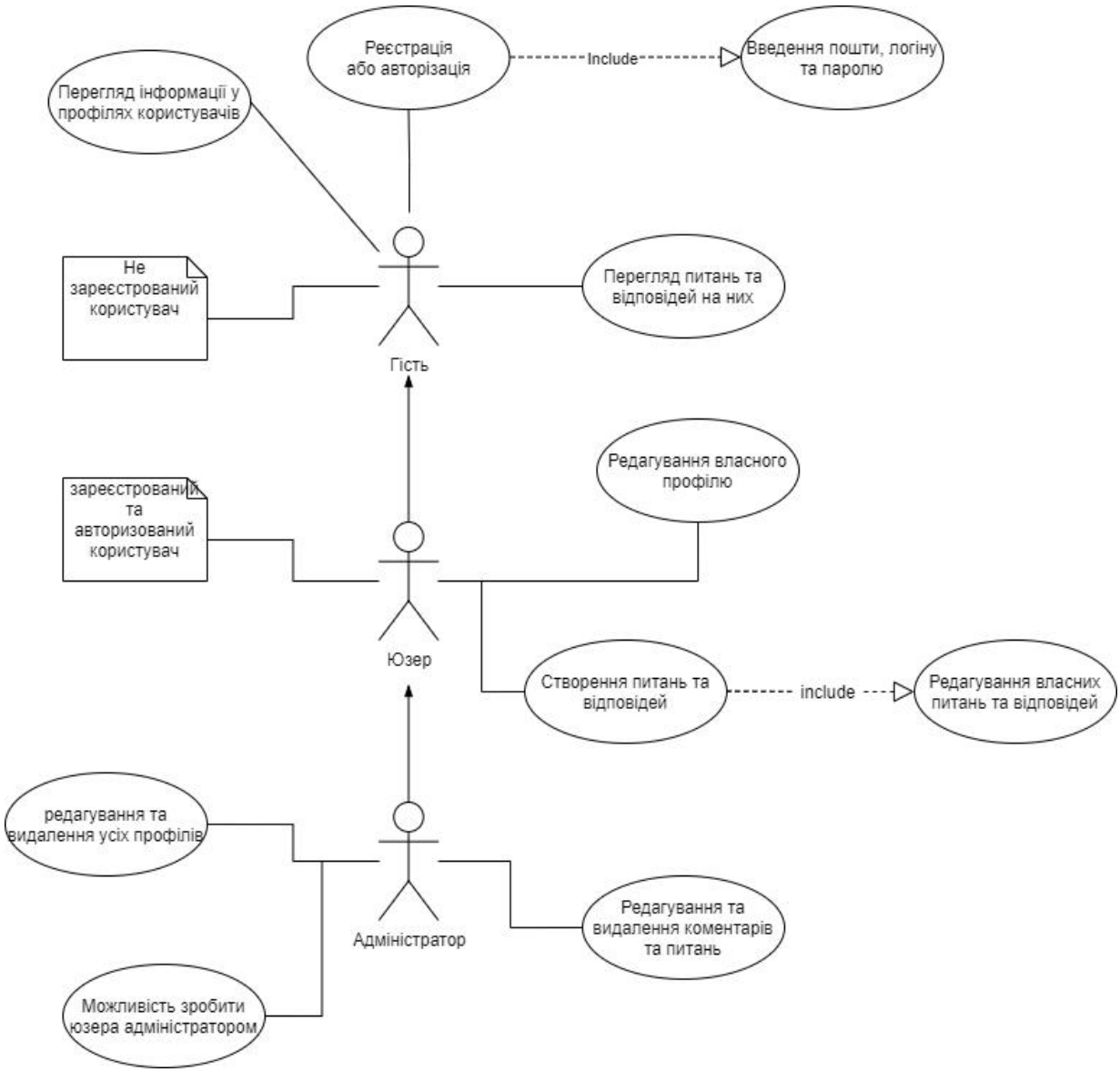

Рисунок 2.6.1 Ієрархія користувачів

ВИСНОВОК

В результаті проведеної роботи було закріплено матеріал, вивчений в курсі дисципліни «Основи web-програмування».

Мета курсової роботи створення повноцінного веб-сайту, в моєму випадку сайту-форуму. Розроблений сайт задовольняє всім вимогам, поставленим на етапі створення технічного завдання. Як і планувалося на етапі постановки завдання, сайт містить всі необхідні структурні і навігаційні елементи: форму пошуку, навігаційне меню і так далі.

Було проведено тестування отриманого продукту та результатів його роботи. Форум є повністю готовим програмним продуктом і може бути використаний не тільки для навчальної діяльності а й для комерційної.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «PHP, MySQL и DreamweaverMX 2004. Разработка интерактивных Web-сайтов.» ДроновВ. А. — СПб.: БХВ-Петербург, 2005. — 448 с :ил.
2. «PHP and MySQL Web Development (4th Edition)», Luke Welling, Laura Thomson 848 стр., сил.;ISBN 978-5-8459-1574-0, 978-0-672-32916-6.
3. Мардан Азат. React быстро. Веб-приложения на React, JSX, Redux и GraphQL. — СПб.: «Питер», 2019. — С. 560. — ISBN 978-5-4461-0952-4
4. Shawn McCool. Laravel Starter. — Packt Publishing, 2012. — 64 p. — ISBN 978-1-78216-091-5.
5. Зандстра М. PHP. Объекты, шаблоны и методики программирования. — 5-е изд.. — СПб.: «Диалектика», 2019. — С. 736. — ISBN 978-5-907144-54-5.
6. Кузнецов М., Симдянов И. Самоучитель PHP 7. — 2-е изд.. — СПб., 2018. — С. 448. — ISBN 978-5-9775-3817-6.

ДОДАТОК 1. КОД ПРОГРАМИ

ForceJsonResponse.php Middleware

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class ForceJsonResponse
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle(Request $request, Closure $next)
    {
        $request->headers->set('Accept', 'application/json');

        return $next($request);
    }
}
```

QuestionController.php

```
<?php

namespace App\Http\Controllers;

use App\Http\Resources\Question as QuestionResource;
use App\Http\Resources\ViewQuestion;
use App\Question;
use Illuminate\Http\Request;
use Illuminate\Database\Eloquent\Builder;

class QuestionController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth')->except('index', 'show', 'top10');
        $this->authorizeResource(Question::class, 'question');
    }

    /**
     * Display a listing of the resource.
     *
     * @param \Illuminate\Http\Request $request
     */
```

```
    public function index(Request $request)
    {
        $search = $request->get('search');
        $sort = $request->get('sort');

        return QuestionResource::collection(
            Question::when($search, function ($query, $search) {
                return $query->where('title', 'like', '%' . $search .
'%');
            })
                ->when($sort, function ($query, $sort) {
                    return $query->orderBy($sort, 'desc');
                }, function ($query) {
                    return $query->orderBy('created_at', 'desc');
                })
                ->paginate()
        );
    }

    /**
     * Display a listing of the resource.
     *
     * @param \Illuminate\Http\Request $request
     */
    public function my(Request $request)
    {
        $search = $request->get('search');
        $sort = $request->get('sort');

        return QuestionResource::collection(
            Question::when($search, function ($query, $search) {
                return $query->where('title', 'like', '%' . $search .
'%');
            })
                ->when($sort, function ($query, $sort) {
                    return $query->orderBy($sort, 'desc');
                }, function ($query) {
                    return $query->orderBy('created_at', 'desc');
                })
                ->where('user_id', $request->user()->id)
                ->paginate()
        );
    }

    /**
     * Display a listing of the resource.
     *
     * @param \Illuminate\Http\Request $request
     */
    public function commented(Request $request)
    {
```

```
        $search = $request->get('search');
        $sort = $request->get('sort');

        return QuestionResource::collection(
            Question::when($search, function ($query, $search) {
                return $query->where('title', 'like', '%' . $search .
'%');
            })
                ->when($sort, function ($query, $sort) {
                    return $query->orderBy($sort, 'desc');
                }, function ($query) {
                    return $query->orderBy('created_at', 'desc');
                })
                ->whereHas('replies', function (Builder $query) use
($request) {
                    $query->where('user_id', $request->user()->id);
                })
                ->paginate()
        );
    }

    /**
     * Display a listing of the resource.
     *
     * @param \Illuminate\Http\Request $request
     */
    public function top10(Request $request)
    {
        return QuestionResource::collection(
            Question::orderBy('reply_count', 'desc')->limit(10)->get()
        );
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     */
    public function store(Request $request)
    {
        $question = Question::create([
            'user_id' => $request->user()->id,
            'category_id' => $request->category_id,
            'title' => $request->title,
            'body' => $request->body,
        ]);

        return new ViewQuestion($question);
    }

    /**
```

```
     * Display the specified resource.
     *
     * @param \App\Question $question
     */
    public function show(Question $question)
    {
        $question->views_count++;
        $question->save();

        return new ViewQuestion($question);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @param \App\Question $question
     */
    public function update(Request $request, Question $question)
    {
        $question->update($request->only(['title', 'body',
'category_id']));

        return new ViewQuestion($question);
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param \App\Question $question
     * @throws \Exception
     */
    public function destroy(Question $question)
    {
        $question->delete();

        return response()->json(null, 204);
    }
}
```

UserPolicy.php

```
namespace App\Policies;
     use App\User;
     use Illuminate\Auth\Access\HandlesAuthorization;

     class UserPolicy
     {
         use HandlesAuthorization;

         public function before($user, $ability)
         {
             if ($user->isAdmin()) {
```

```
                return true;
            }

            if ($user->is_deleted) {
                return false;
            }
        }
```

```
public function viewAny(?User $user)
    {
        return true;
    }
public function view(?User $user, ?User $model)
    {
        return !$model->is_deleted;
    }
public function create(User $user)
    {
        return false;
    }
public function update(User $user, User $model)
```

```
    {
        return $user->id === $model->id;
    }
public function delete(User $user, User $model)
    {
        return $user->id === $model->id;
    }
}
```

QuestionController.php

```
<?php
namespace App\Http\Controllers;
use App\Http\Resources\Question as QuestionResource;
use App\Http\Resources\ViewQuestion;
use App\Question;
use Illuminate\Http\Request;
use Illuminate\Database\Eloquent\Builder;

class QuestionController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth')->except('index', 'show',
'top10');
        $this->authorizeResource(Question::class, 'question');
    }
public function index(Request $request)
    {
        $search = $request->get('search');
        $sort = $request->get('sort');
        return QuestionResource::collection(
            Question::when($search, function ($query, $search) {
                return $query->where('title', 'like', '%' .
$search . '%');
            })
```

```
                ->when($sort, function ($query, $sort) {
                    return $query->orderBy($sort, 'desc');
                }, function ($query) {
                    return $query->orderBy('created_at', 'desc');
                })
                ->paginate()
        );
    }
public function my(Request $request)
    {
        $search = $request->get('search');
        $sort = $request->get('sort');

        return QuestionResource::collection(
            Question::when($search, function ($query, $search) {
                return $query->where('title', 'like', '%' .
$search . '%');
            })
                ->when($sort, function ($query, $sort) {
                    return $query->orderBy($sort, 'desc');
                }, function ($query) {
                    return $query->orderBy('created_at', 'desc');
                })
                ->where('user_id', $request->user()->id)
                ->paginate()
        );
    }
public function commented(Request $request)
    {
        $search = $request->get('search');
        $sort = $request->get('sort');

        return QuestionResource::collection(
            Question::when($search, function ($query, $search) {
```

```
                return $query->where('title', 'like', '%' .
$search . '%');
            })
                ->when($sort, function ($query, $sort) {
                    return $query->orderBy($sort, 'desc');
                }, function ($query) {
                    return $query->orderBy('created_at', 'desc');
                })
                ->whereHas('replies', function (Builder $query)
use ($request) {
                    $query->where('user_id', $request->user()-
>id);
                })
                ->paginate()
        );
    }
```

User.php

```
<?php

namespace App;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

/**
 * App\User
 *
 * @property int $id
 * @property string $name
 * @property string $email
 * @property \Illuminate\Support\Carbon|null $email_verified_at
 * @property string $password
 * @property string|null $remember_token
 * @property boolean $is_admin
```

```
 * @property boolean $is_deleted
 * @property \Illuminate\Support\Carbon|null $created_at
 * @property \Illuminate\Support\Carbon|null $updated_at
 * @property-read
\Illuminate\Notifications\DatabaseNotificationCollection|\Illumi
nate\Notifications\DatabaseNotification[] $notifications
 * @property-read int|null $notifications_count
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User newModelQuery()
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User newQuery()
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User query()
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User
whereCreatedAt($value)
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User
whereEmail($value)
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User
whereEmailVerifiedAt($value)
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User whereId($value)
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User whereName($value)
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User
wherePassword($value)
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User
whereRememberToken($value)
 * @method static
\Illuminate\Database\Eloquent\Builder|\App\User
whereUpdatedAt($value)
```

```
 * @mixin \Eloquent
 */
class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name',
        'email',
        'password',
        'is_admin',
        'about_me',
        'is_deleted',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
```

```
    protected $casts = [
        'email_verified_at' => 'datetime',
        'is_admin' => 'boolean',
    ];

    public function isAdmin()
    {
        return $this->is_admin;
    }
}
```

Весь код доступний за посиланням:

https://github.com/DanilAndreev/starscream_project