МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБДОДАТКА ДЛЯ РЕЄСТРАЦІЇ НА КОНФЕРЕНЦІЇ З ВИКОРИСТАННЯМ LARAVEL, VUEJS, LARAVEL NOVA»**

Виконала: студентка 4 курсу, групи 6.1219-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
М.О.Суворова
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, PhD, Столярова А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ 07 ” 02 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Суворовій Марії Олександрівні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебдодатка для реєстрації на конференції з використанням Laravel, VueJs, Laravel Nova

керівник роботи Столярова Анастасія Валеріївна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка вебдодатка для реєстрації на конференції з використанням Laravel, VueJs, Laravel Nova

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 07.02.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.02.2023	
2.	Збір вихідних даних.	23.02.2023	
3.	Обробка методичних та теоретичних	15.03.2023	
	джерел.		
4.	Розробка першого та другого розділу.	12.04.2023	
5.	Розробка третього розділу.	17.05.2023	
6.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	23.06.2023	

Студент _____ М.О. Суворова
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.В. Столярова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебдодатка для реєстрації на конференції з використанням Laravel, VueJs, Laravel Nova»: 48 с., 22 рис., 8 джерел, 2 додатки.

КОНФЕРЕНЦІЯ, GOOGLE MAPS, LARAVEL, LARAVEL NOVA, STRIPE, VUEJS, ZOOM.

Об’єкт дослідження – вебдодаток для реєстрації на конференції.

Мета роботи: розробка функціонального та ефективного вебдодатока, який надає можливість реєстрації на конференції за допомогою Laravel, Vue.js та Laravel Nova.

Метод дослідження – аналітичні та порівняльні методи програмної інженерії.

Був розроблений додаток, що надає можливість реєстрації на конференцію, а також адміністрування системи. Його новизна полягає у використанні сучасних інструментів та технологій для розробки веб-додатків. Дана робота базується на попередніх дослідженнях з використання Laravel, Vue.js та Laravel Nova у веб-розробці. Вона доповнює та розширює існуючі знання в цій області. Розроблений веб-додаток може бути використаний організаторами конференцій для полегшення процесу реєстрації учасників.

Розробка веб-додатку для реєстрації на конференції є актуальною та значимою, оскільки спрощує процес реєстрації учасників та покращує організацію конференцій. Отже, розробка веб-додатку з використанням Laravel, Vue.js та Laravel Nova для реєстрації на конференції має важливе значення, а результати дослідження вказують на успішність та перспективи використання цього підходу у практиці.

SUMMARY

Bachelor’s qualifying paper «Development of a Web Application for Conference Registration using Laravel, VueJs, Laravel Nova»: 48 pages, 22 figures, 8 references, 2 supplements.

CONFERENCES, GOOGLE MAPS, LARAVEL, LARAVEL NOVA, STRIPE, VUEJS, ZOOM.

The object of the study is web application for conference registration.

The aim of the study is development of a functional and efficient web application that allows registration for conferences using Laravel, Vue.js, and Laravel Nova.

The method of research is software engineering methods.

An application was developed that allows conference registration as well as system administration. Its novelty lies in the use of modern tools and technologies for web application development. This work is based on previous research on the use of Laravel, Vue.js, and Laravel Nova in web development. It complements and expands existing knowledge in this field. The developed web application can be used by conference organizers to facilitate the registration process for participants.

The development of a web application for conference registration is relevant and significant as it simplifies the registration process for participants and improves conference organization. Therefore, the development of a web application using Laravel, Vue.js, and Laravel Nova for conference registration is of great importance, and the research results indicate the success and prospects of using this approach in practice.

ЗМІСТ

ВСТУП

Швидкий прогрес технологій кардинально змінив різні аспекти нашого життя, включаючи організацію конференцій та подій. У сучасну цифрову епоху важливо, щоб організатори конференцій мали ефективні та зручні веб-додатки для керування реєстрацією. Ця дипломна робота присвячена розробці веб-додатку, спеціально призначеного для реєстрації на конференції, з використанням потужних фреймворків Laravel, VueJs та Laravel Nova.

В даний час інтернет-технології стали необхідним інструментом для ефективного керування подіями, а також для забезпечення комфорту та задоволення учасників конференцій. Організаторам потрібні веб-додатки, які надають широкий спектр функцій, таких як реєстрація, обробка платежів, керування учасниками та надання детальної інформації про подію.

Мета цієї дипломної роботи - розробити веб-додаток для реєстрації на конференції, що відповідає вищезгаданим вимогам, з використанням фреймворків Laravel, VueJs та Laravel Nova. Laravel є одним з найпопулярніших фреймворків для розробки веб-додатків на PHP, відомим своєю простотою використання, гнучкістю та розширюваністю. VueJs - це молодий та потужний фреймворк JavaScript для розробки користувацького інтерфейсу, який забезпечує швидку та плавну взаємодію з додатком. Laravel Nova - це інструмент для швидкого створення адміністративних панелей, що дозволяє легко керувати даними та налаштуваннями додатку.

Цей документ розглядає весь процес розробки веб-додатку, починаючи з аналізу вимог, проєктування архітектури, програмування, тестування та розгортання. Результатом роботи буде функціональний веб-додаток, який забезпечить зручну реєстрацію на конференції та забезпечить організаторам потрібні інструменти для ефективного керування подією.

Цей дипломний проєкт важливий, оскільки він дозволить спростити процес реєстрації на конференції та полегшить організацію та керування

подіями. Крім того, він служить додатковим внеском у сферу розробки веб-додатків та використання сучасних технологій для покращення організаційних процесів.

1 ТЕОРЕТИЧНИЙ ОГЛЯД

1.1 Огляд фреймворка Laravel та його основних функцій і можливостей

Laravel – це фреймворк веб-додатків з виразним і елегантним синтаксисом. Laravel надаэ потужні функції, такі як повне впровадження залежностей, виразний рівень абстракції бази даних, черги та заплановані завдання, тестування на рівні одиниць та інтеграції та багато іншого [1]. Laravel – це потужний фреймворк на мові програмування PHP. Він надає розробникам зручні та ефективні інструменти для створення високоякісних та масштабованих додатків.

Ларавель пропонує просту та елегантну архітектуру, яка забезпечує легке використання та розширення. Фреймворк має широкий набір функціональних можливостей, які допомагають розробникам прискорити процес розробки та забезпечити високу якість коду [2].

Далі маю важливим зазначити основні переваги та недоліки Laravel.

Переваги Laravel.

Екосистема: Laravel має широку екосистему з багатьма готовими пакетами та інструментами. Це дозволяє розробникам швидше реалізувати функціональність та скоротити час розробки.

Простота використання: Laravel надає зрозумілий синтаксис та інтуїтивно зрозумілу структуру каталогів. Він має чистий і лаконічний код, що полегшує розробку й підтримку проєктів.

Маршрутизація: Laravel має потужну систему маршрутизації, яка дозволяє легко визначати URL-адреси та їх обробники. Це полегшує створення складних маршрутів та роботу з різними HTTP-методами.

Міграції баз даних: Laravel надає механізм міграцій, який дозволяє створювати та змінювати схему бази даних за допомогою коду. Це полегшує

управління версіями бази даних та спільну роботу над проєктом з іншими розробниками.

Аутентифікація та авторизація: Laravel має вбудовану систему аутентифікації, яка дозволяє легко впровадити функції реєстрації, входу та виходу з системи. Він також надає механізми для визначення рівнів доступу та авторизації користувачів.

Недоліки Laravel.

Швидкодія: порівняно з деякими іншими фреймворками, Laravel може бути трохи повільнішим. Це пов'язано з додатковими операціями, які виконуються фреймворком для підтримки його функціональності.

Навчання: якщо ви новачок у Laravel або PHP, то може знадобитися певний час для вивчення фреймворку. Він має певну кількість концепцій та патернів проєктування, які потребують розуміння.

Залежність від розширень: іноді для досягнення певної функціональності в Laravel може знадобитися встановлення розширень або додаткових пакетів. Це може вплинути на складність конфігурації та управління залежностями.

Варто зазначити, що багато з недоліків Laravel можна пом'якшити або усунути, використовуючи відповідні методи, пакети або розширення.

1.2 Вивчення фреймворка VueJs та його використання для розробки користувацького інтерфейсу

Vue (вимовляється як /vju:/, схоже на «в'ю») – це фреймворк JavaScript для побудови користувацьких інтерфейсів. Він ґрунтується на стандартних технологіях HTML, CSS і JavaScript та надає декларативну та компонентну модель програмування, що допомагає ефективно розробляти користувацькі інтерфейси, будь то прості або складні [3].

Vue.js – це прогресивний фреймворк JavaScript для розробки користувацького інтерфейсу веб-додатків. Він зосереджується на створенні

інтерактивних та динамічних веб-інтерфейсів, які реагують на дії користувача та забезпечують швидку та плавну взаємодію [4].

Далі маю важливим зазначити основні переваги та недоліки VueJs.

Переваги VueJs.

Легкість вивчення: Vue.js має простий та зрозумілий синтаксис, що робить його досить легким для вивчення, навіть для початківців у розробці веб-додатків. Це дозволяє швидко почати працювати з фреймворком і розвивати проєкти.

Прогресивність: Vue.js побудований на принципах прогресивного розширення. Ви можете почати з використання окремих частин фреймворку в існуючому проєкті, поступово розширюючи його функціональність. Це полегшує інтеграцію Vue.js з іншими проєктами.

Інтерактивність: Vue.js забезпечує декларативний підхід до роботи зі шаблонами, що дозволяє легко використовувати динамічні дані та створювати інтерактивні компоненти. Він також має зручні інструменти для реактивного оновлення елементів веб-сторінки.

Компонентний підхід: Vue.js базується на компонентній архітектурі, що сприяє повторному використанню коду, зручному управлінню структурою додатка та покращенню його скальованості. Ви можете легко створювати, комбінувати та вкладати компоненти, що спрощує розробку складних і багатофункціональних додатків.

Недоліки VueJs.

Розмір: одним з недоліків Vue.js є його розмір. У порівнянні з деякими іншими фреймворками, він може бути трохи більшим, що може вплинути на час завантаження додатку та продуктивність, особливо на мобільних пристроях з обмеженими ресурсами.

Екосистема: хоча екосистема Vue.js постійно розвивається, вона все ж не така розгалужена та розширена, як у випадку Angular або React. Це може призвести до того, що вам доведеться шукати альтернативні рішення або пакети для певного функціоналу, який може бути легко доступний в інших

фреймворках.

Сумісність зі старими браузерами: Vue.js використовує сучасні можливості веб-браузерів, що може призвести до проблем зі сумісністю зі старими версіями браузерів. Якщо ваша цільова аудиторія використовує застарілі браузери, можуть виникнути проблеми з роботою додатків на Vue.js.

Враховуючи ці переваги та недоліки, Vue.js залишається популярним вибором для розробки веб-додатків, особливо для менших та середніх проєктів, де простота вивчення та швидкість розробки мають важливе значення.

1.3 Представлення функціональних можливостей Laravel Nova та його значення для адміністративного керування

Nova – чудово оформлена панель адміністрування для Laravel. Ретельно розроблений творцями Laravel, щоб зробити вас найпродуктивнішим розробником у галактиці [5].

Laravel Nova – це адміністративна панель адміністрування, розроблена для Laravel, яка надає широкий набір функціональних можливостей для легкого та ефективного керування веб-додатком.

Далі маю важливим зазначити основні переваги та недоліки Laravel Nova.

Переваги Laravel Nova.

Швидке створення адміністративних панелей: Laravel Nova надає готові шаблони та інструменти, які дозволяють швидко створювати адміністративні панелі для управління вашими даними. Ви можете швидко налаштувати моделі, поля, фільтри та інші компоненти, не витрачаючи багато часу на рутинну роботу.

Легка інтеграція з Laravel: Laravel Nova розроблений спеціально для фреймворку Laravel, тому він інтегрується дуже легко. Ви можете використовувати всю потужну функціональність Laravel, таку як міграції, маршрутизацію, автентифікацію та авторизацію, в адміністративних панелях

Laravel Nova.

Гнучкість та розширюваність: Laravel Nova надає можливість розширювати його функціональність за допомогою власних розширень. Ви можете створювати свої власні поля, фільтри, метрики та інші компоненти, щоб відповідати специфічним потребам вашого додатку.

Елегантний та сучасний дизайн: Адміністративний інтерфейс Laravel Nova має елегантний та сучасний дизайн, що дозволяє створювати професійно виглядящі адміністративні панелі. Він має зручний інтерфейс користувача, що дозволяє з легкістю навігувати та виконувати різні завдання.

Недоліки Laravel Nova.

Вартість: Laravel Nova є комерційним продуктом, і для його використання потрібно придбати ліцензію. Це може бути недоліком для розробників з обмеженим бюджетом або для проєктів з низькими фінансовими можливостями.

Обмеження функціональності: хоча Laravel Nova надає широкі можливості для швидкої розробки адміністративних панелей, він все ж може бути обмежений у порівнянні з власноруч розробленими інтерфейсами. Якщо вам потрібні специфічні функції або налаштування, можливо, вам доведеться вносити додаткові зміни в код Laravel Nova або писати власний код.

Залежність від Laravel: Laravel Nova побудований на основі фреймворку Laravel і повністю залежить від нього. Це означає, що ви повинні користуватися Laravel для використання Nova, і він не може бути використаний з іншими фреймворками або технологіями. Це може бути недоліком, якщо ви бажаєте використовувати інший фреймворк або вам потрібно інтегрувати Nova з існуючою системою, побудованою на іншій технології.

Загалом, Laravel Nova є потужним інструментом для адміністративного керування вашим Laravel-додатком. Він дозволяє легко та ефективно керувати даними, налаштовувати доступ та забезпечувати високу продуктивність вашого веб-додатку. Його гнучкість та широкий функціонал роблять його цінним ресурсом для розробників та адміністраторів, що шукають потужні засоби для

адміністрування.

1.4 Огляд зовнішніх API: Google Maps, Stripe, Zoom

API (інтерфейс програмування додатків) – це набір правил та протоколів, які дозволяють різним програмним додаткам взаємодіяти між собою. Він визначає, які запити можна робити до системи та які відповіді отримувати. API встановлює комунікаційний міст між різними програмними компонентами, дозволяючи їм обмінюватися даними та виконувати певні функції.

API може бути реалізовано у вигляді набору веб-сервісів, бібліотек, протоколів або інтерфейсу командного рядка. Через API, розробники можуть використовувати функціональність інших додатків або сервісів без необхідності знати всі деталі їх внутрішньої реалізації. API дозволяє створювати додатки, які використовують можливості інших систем, розширюючи функціонал та забезпечуючи взаємодію з ними.

API може бути публічним, доступним для широкої громадськості, або приватним, призначеним для обмеженого кола користувачів. Він може мати різні методи, параметри та формати даних для обміну інформацією між додатками. API використовується для інтеграції різних систем та створення розширень, додатків або сервісів, що взаємодіють з іншими програмами.

1.4.1 Google Maps API

Google Maps API – це набір програмних інтерфейсів, наданих компанією Google, які дозволяють взаємодіяти з картографічними сервісами та місцезнаходженнями, що надаються Google Maps [7]. Цей API дозволяє розробникам інтегрувати функціональність карт Google Maps в свої веб-додатки.

За допомогою Google Maps API, ви можете отримувати доступ до різних функцій, таких як відображення карт, розташування об'єктів на карті, отримання маршрутів та вказівок про рух, пошук місць за адресою або назвою, відображення інформації про погоду та багато іншого [6].

Google Maps API також надає можливість розробникам створювати власні мітки, накладати шари на карту, взаємодіяти з географічними об'єктами та надавати користувачам інтерактивний досвід. Він також підтримує мобільні платформи, що дозволяє інтегрувати функціональність Google Maps у мобільні додатки.

Google Maps API є потужним інструментом для розробників, які потребують географічних функцій у своїх веб-додатках. Він дозволяє створювати зручні та інтерактивні картографічні рішення, що покращують користувацький досвід та надають доступ до різноманітної географічної інформації.

1.4.2 Stripe API

Stripe API – це набір програмних інтерфейсів, розроблений компанією Stripe, який дозволяє розробникам інтегрувати систему електронних платежів Stripe в свої веб-додатки [8].

За допомогою Stripe API, розробники можуть забезпечити прийом різних видів платежів в своїх додатках, включаючи кредитні та дебетові картки, мобільні платежі, електронні гаманці та інші способи оплати. Це дозволяє підприємствам легко та безпечно обробляти платежі в Інтернеті.

Stripe API надає розробникам доступ до різних функцій, таких як створення та управління клієнтами, створення та обробка платежів, відправлення рахунків, відстеження транзакцій та багато іншого. Він також забезпечує безпеку платежів, включаючи захист від шахрайства та додаткові заходи безпеки для збереження конфіденційності клієнтських даних.

Stripe API підтримує різні програмні мови та платформи, що дозволяє розробникам інтегрувати Stripe в будь-який тип веб-додатків. Він також надає документацію, приклади коду та різні інструменти, щоб полегшити розробку та інтеграцію платежної системи Stripe.

Stripe API є потужним інструментом для розробників, які потребують обробки платежів у своїх веб-додатках. Він дозволяє легко приймати платежі в Інтернеті та створювати зручні рішення для електронної комерції та інших платіжних сценаріїв.

1.4.3 Zoom API

Zoom API – це набір програмних інтерфейсів, розроблений компанією Zoom, який дозволяє розробникам інтегрувати функціональність відеозв'язку та конференцій Zoom у свої веб-додатки та сервіси.

За допомогою Zoom API, розробники можуть створювати зручні та інтерактивні додатки, які надають можливість проводити відеоконференції, вебінари, чати та багато іншого. Це дозволяє користувачам взаємодіяти в реальному часі, обмінюватися даними та спілкуватися віртуально, незалежно від їх місця перебування.

Zoom API надає розробникам доступ до різних функцій, таких як створення та керування зустрічами, управління користувачами, надання прав доступу, отримання потокового відео та аудіо, робота з чатом та багато іншого. Він також підтримує можливості інтеграції з іншими сервісами та додатками, що дозволяє розширити функціонал Zoom у своїх проєктах.

Zoom API підтримує різні програмні мови та платформи, що дозволяє розробникам інтегрувати функціональність Zoom у різноманітні веб-додатки та сервіси. Він надає документацію, приклади коду та різні інструменти, які полегшують розробку та інтеграцію з Zoom.

Zoom API є потужним інструментом для розробників, які потребують

відеозв'язку та віртуальних зустрічей у своїх веб-додатках. Він дозволяє створювати зручні та інноваційні рішення для комунікації та співпраці в режимі онлайн, забезпечуючи високу якість відео та аудіо передачі.

1.5 Висновки до розділу 1

У даному розділі було детально розглянуто ключові технологічні компоненти, які використовуються в розробці нашого веб-додатку для реєстрації на конференції. Огляд фреймворка Laravel надав нам розуміння його потужних функцій та можливостей, що дозволяють зосередитися на створенні вражаючих функцій, не витрачаючи зайвий час на деталі реалізації.

Дослідження фреймворка VueJs розкрило його роль у створенні користувацького інтерфейсу, забезпечуючи гнучкість та ефективність в розробці інтерактивних компонентів. Це надає можливість створювати зручний та привабливий інтерфейс для користувачів нашого веб-додатку.

Функціональні можливості Laravel Nova дозволяють легко управляти та адмініструвати наш веб-додаток, забезпечуючи зручний інтерфейс для адміністраторів. Це значно спрощує процес керування та підтримки нашого додатку, дозволяючи нам зосередитися на його розвитку та поліпшенні.

Нарешті, використання зовнішніх API, зокрема Google Maps, Stripe та Zoom, відкриває нові можливості для нашого веб-додатку. Ці API дозволяють нам інтегрувати потужні функціональність, такі як мапи, платежі та відеоконференції, що розширює можливості та забезпечує зручну взаємодію для користувачів.

В результаті, завдяки використанню Laravel, VueJs, Laravel Nova та зовнішніх API, ми маємо потужний інструментарій для розробки веб-додатку, який не тільки забезпечує зручну реєстрацію на конференцію, але й надає інноваційні та привабливі функції для задоволення потреб користувачів.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Постановка задачі

На сайті буде розроблена система, яка дозволить користувачам виконувати різні ролі: адміністратор, ведучий та слухач. Головна функціональність сайту буде пов'язана з організацією та участю в конференціях.

Кожна конференція матиме свої основні атрибути: назву, дату проведення та місце проведення. Користувачі зможуть зареєструватись на конференцію і вибрати роль – ведучого або слухача.

Якщо користувач обирає роль ведучого, він отримує можливість створити звіт про свою презентацію. В цьому звіті він зможе вказати назву конференції, час проведення та завантажити файл з презентацією, також матиме можливість вказувати буде проведена конференція очно, чи вона потребує створення zoom конференції. Існуватимуть різні плани для користувачів, з базовим планом при реєстрації та додатковими платними планами, які надають розширені функції.

Крім того, користувачі матимуть можливість залишати коментарі до звітів ведучих та додавати конференції до свого списку обраних подій. Сторінка конференцій також буде містити вбудовану карту Google, на якій буде відображено місце проведення конференції.

Адміністратор сайту матиме повний доступ до редагування конференцій. Він зможе змінювати назву, дату та місце проведення конференцій, а також вставляти карту Google з точним місцем проведення.

Загалом, цей веб-сайт надасть можливість різним типам користувачів ефективно спілкуватись, організовувати та брати участь у конференціях, обмінюватись думками та ідеями, а також збагатить досвід участі в наукових заходах.

2.2 Створення ER-діаграми та діаграми прецедентів

ER-діаграма (сутнісно-зв'язкова діаграма) є потужним інструментом для моделювання та візуалізації структури даних у базах даних. Тож створимо діаграмму згідно до опису завдання (рис. 2.1).

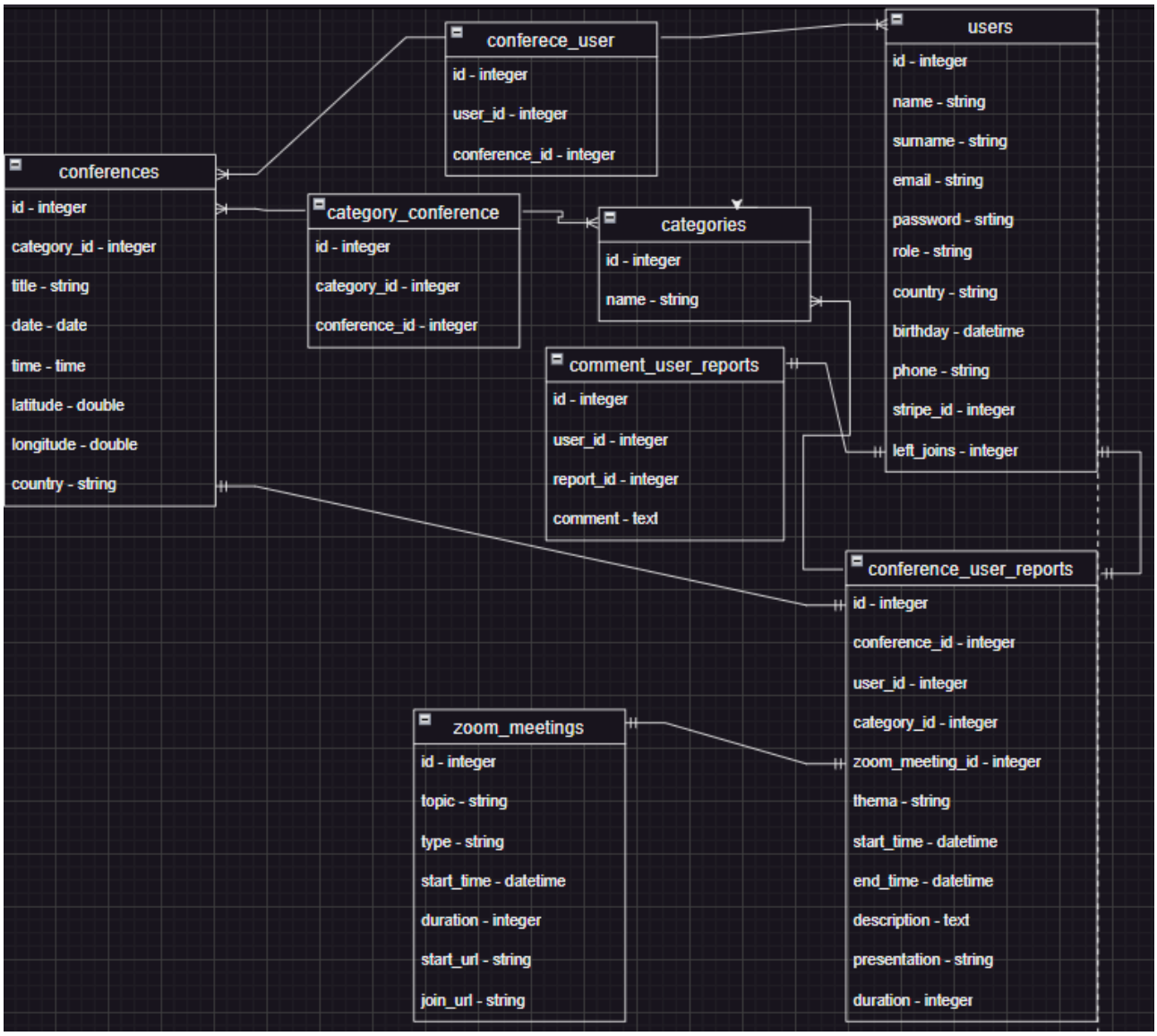


Рисунок 2.1 – ER-діаграма

Також у даному підрозділі створимо діаграму прецедентів, що представляє із себе ключовий інструмент у розробці програмного забезпечення, що допомагає зрозуміти та візуалізувати взаємодію між системою та її користувачами або зовнішніми акторами. Вона описує функціональність

системи з точки зору зовнішніх стейкхолдерів та використовується для аналізу, проєктування та документування вимог до системи (рис. 2.2).

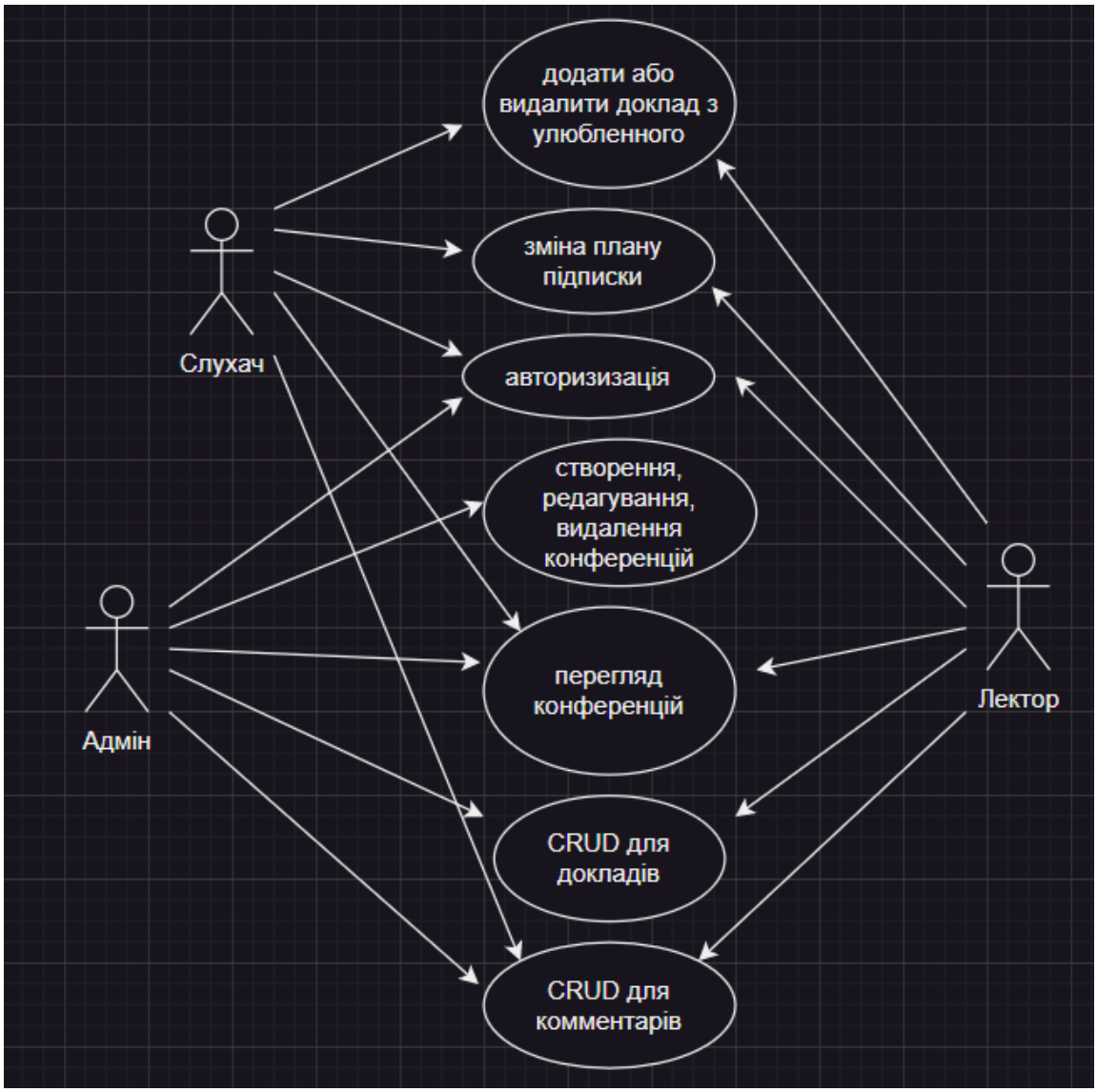


Рисунок 2.2 – Діаграма прецедентів

2.3 Створення діаграми класів

У даній роботі вважаю запотрібне використовувати підхід MVC, так як він є основополягаючим у архітектурі Laravel застосунків. Надалі представлена частина діаграми классів для розуміння структури застосунку (див. рис. 2.3 та 2.4).

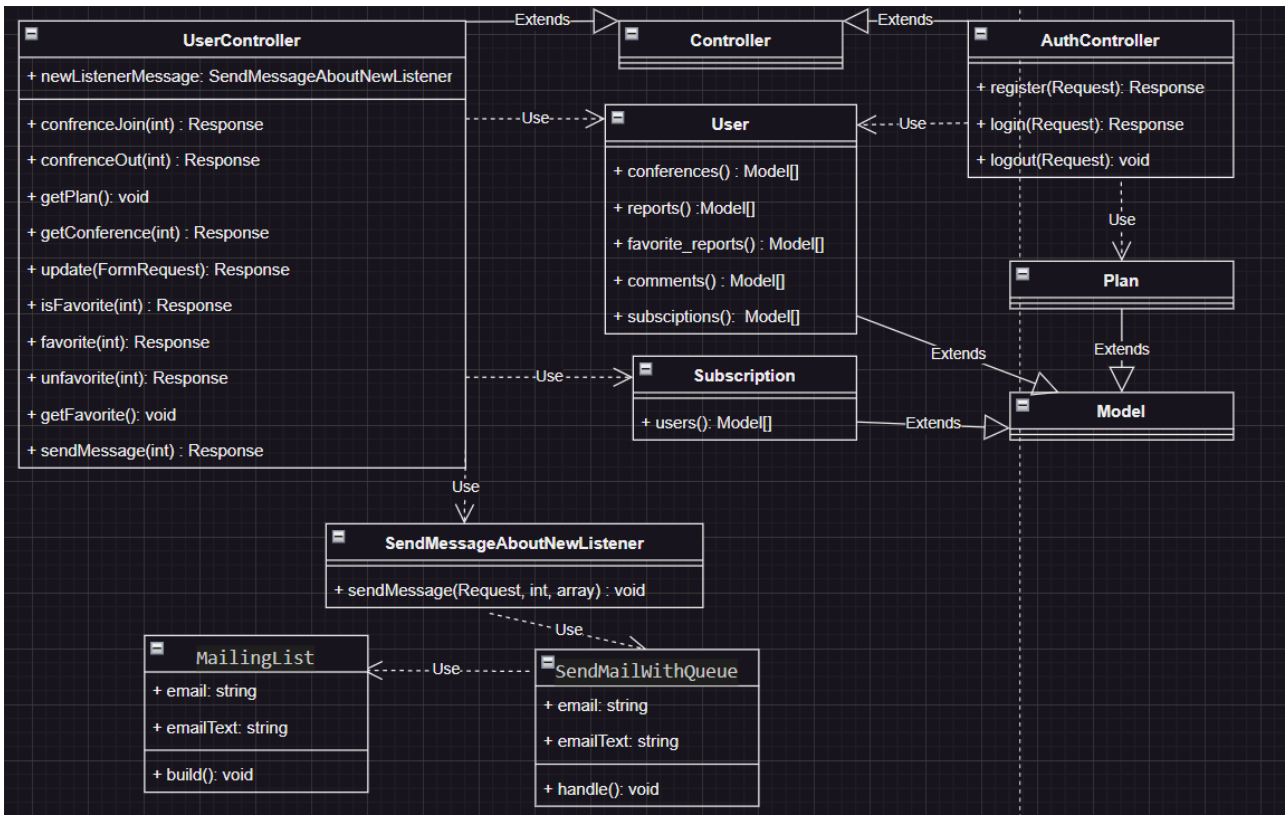


Рисунок 2.3 – Фрагмент діаграми классів

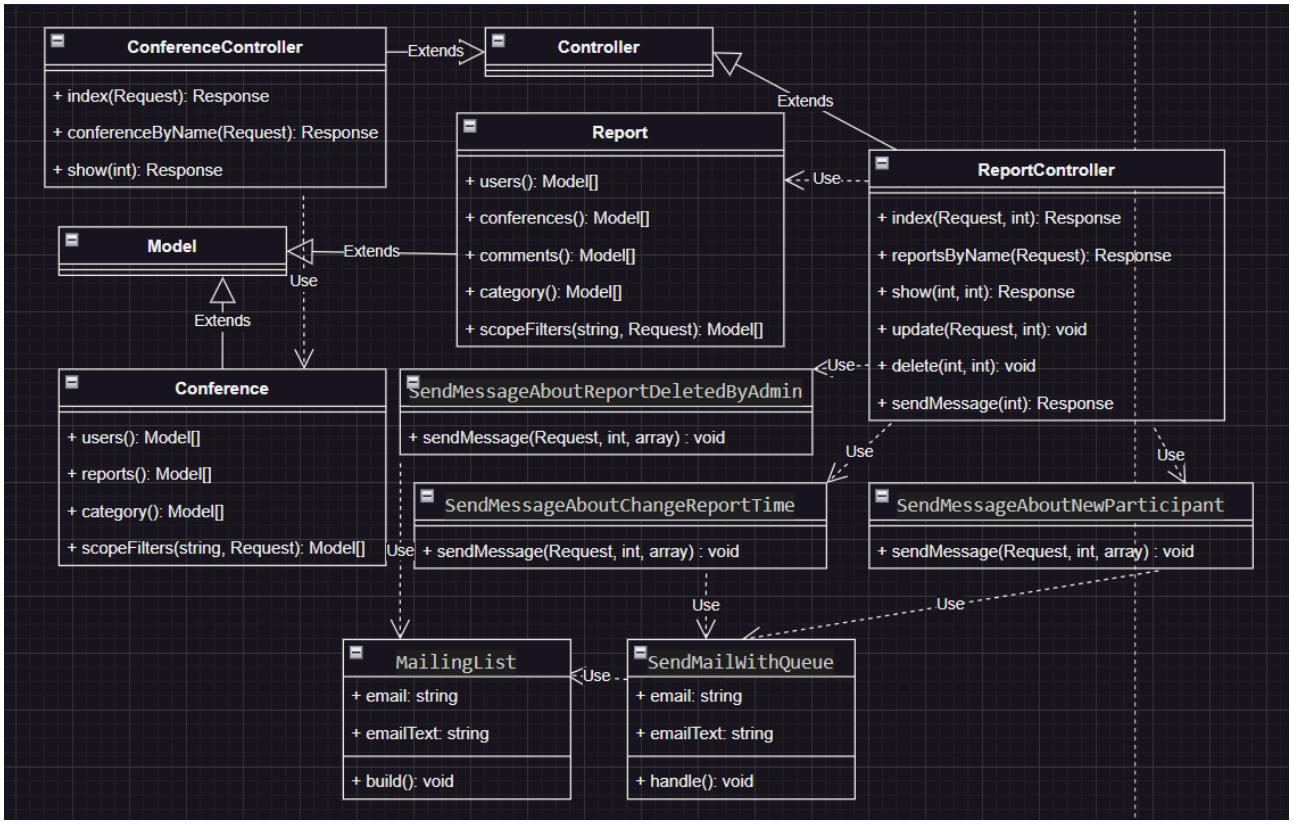


Рисунок 2.4 – Фрагмент діаграми класів 2

MVC (Model-View-Controller) – це архітектурний підхід, що широко використовується в Laravel. Використовуючи цей підхід разом з Vue.js, можна створити потужну комбінацію для розробки додатків з динамічним інтерфейсом користувача.

У звичайному MVC-підході Laravel, представлення (View) відповідають за відображення даних та інтерфейсу користувача. Однак, з використанням Vue.js, можна замінити традиційні представлення на компоненти Vue, які забезпечують динамічну реактивність та взаємодію з користувачем.

Контролери (Controller) в Laravel відповідають за обробку запитів та координацію між моделями та представленнями. Вони обробляють запити користувача та передають необхідні дані до відповідних Vue компонентів для відображення та взаємодії з ними.

Моделі (Model) в Laravel зберігають бізнес-логіку та відповідають за доступ до даних. Вони можуть бути використані незмінним чином з Vue.js, якщо зв'язок з представленням та контролером відбувається через API-запити. Моделі можуть надсилати та отримувати дані з сервера за допомогою AJAX-запитів, що забезпечує реактивність та оновлення інтерфейсу без перезавантаження сторінки.

Завдяки MVC-підходу, код додатку стає більш організованим та керованим. Модульність архітектури дозволяє змінювати окремі компоненти без впливу на решту системи. Крім того, цей підхід полегшує співпрацю між розробниками, оскільки кожен може працювати над своїми відповідальними областями.

2.4 Висновки до розділу 2

В результаті аналізу та проєктування системи було розроблено детальну архітектуру та компоненти, необхідні для її реалізації. Були визначені взаємозв'язки між цими компонентами та способи взаємодії між ними.

Проєктування системи допомагає забезпечити ефективне та структуроване виконання розробки, зменшує ризики та забезпечує високу якість результуючої системи. Цей процес визначає основні принципи, архітектурні рішення та структуру, які використовуються для побудови системи.

Результатом проєктування системи є докладний план, що включає опис компонентів системи, їх функціональність та взаємозв'язки, а також детальні технічні специфікації. Цей план є основою для подальшої розробки та імплементації системи з використанням відповідних технологій та інструментів.

В цьому розділі були представлені результати проєктування системи, що будуть використані для подальшої розробки та реалізації системи. Вони включають діаграми архітектури, опис компонентів, взаємодію між ними та інші важливі аспекти, які допоможуть зрозуміти та втілити задуману систему.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

У цьому розділі розглядається реалізація та тестування розробленої системи, що становить ключову частину дипломної роботи. Після проведення аналізу, проєктування та моделювання бази даних, прийшов час перейти до фази реалізації, де концепції ідеї перетворюються в живий функціональний продукт.

У цьому розділі детально описано процес розробки програмного забезпечення, включаючи вибір технологій, розробку коду, налаштування середовища розробки та побудову архітектури системи. Кожен крок реалізації детально розглядається з метою забезпечення стабільності, ефективності та надійності розробленої системи.

Окрім того, в розділі описано процес тестування системи для перевірки її функціональності, відповідності вимогам та виявлення помилок. Різні методи тестування використовуються для перевірки як окремих компонентів системи, так і їх взаємодії, з метою забезпечення якісного та безперебійного функціонування.

У цьому розділі будуть представлені результати реалізації системи, включаючи детальний опис архітектури, використані технології, принципи розробки та інструменти, що використовувалися. Також буде представлений огляд проведених тестів, їх результати та висновки.

Мета даного розділу полягає у представленні реалізації та валідації розробленої системи, що демонструє успішне втілення поставлених завдань та досягнення поставлених цілей.

3.1 Серверна частина

У цьому підрозділі детально розглядається реалізація серверної частини системи, використовуючи фреймворк Laravel. Серверна частина є ключовим

компонентом архітектури системи, вона відповідає за обробку запитів, взаємодію з базою даних та надання необхідних ресурсів для клієнтської частини.

Фреймворк Laravel, завдяки своїм потужним функціональним можливостям та зручному синтаксису, використовується для швидкої та ефективної розробки серверної частини. Його простота використання, гнучкість та багата екосистема допомагають розробникам створювати стабільні та масштабовані серверні додатки.

У цьому підрозділі будуть описані основні компоненти серверної частини, включаючи маршрутизацію, контролери, моделі та міграції бази даних, які використовуються для реалізації функцій і логіки системи. Також будуть наведені приклади коду та пояснення їх ролі в контексті системи.

Метою цього підрозділу є детальне представлення реалізації серверної частини з використанням фреймворку Laravel, що дозволяє досягти швидкості, надійності та розширюваності системи.

3.1.1 Міграції

Міграції в Laravel – це механізм, який дозволяє керувати структурою бази даних в процесі розробки та розгортання веб-додатків. Вони представляють собою спосіб описати зміни, які потрібно внести до бази даних, та забезпечити автоматичну міграцію даних безпосередньо з коду.

Міграції Laravel базуються на системі контролю версій бази даних (Database Version Control), що дозволяє зберігати та керувати історією змін бази даних. Кожна міграція представляє собою окремий файл, який містить інструкції для створення, зміни або видалення таблиць, стовпців та інших об'єктів бази даних.

Завдяки міграціям, розробники можуть зручно вносити та зберігати зміни в базу даних, що спрощує колаборацію в команді та підтримку бази даних в

актуальному стані. Крім того, міграції дозволяють легко відновити базу даних до попереднього стану або перейти на нову версію без необхідності вручну виконувати SQL-запити.

Загальний процес використання міграцій включає створення нових міграційних файлів, визначення необхідних змін у методах "up" та "down", запуск команди міграції для виконання змін та, за потреби, відкат змін за допомогою команди відкату.

Міграції в Laravel є потужним інструментом для керування базою даних, який спрощує процес розробки та підтримки веб-додатків, забезпечуючи консистентність та керовану еволюцію бази даних.

Міграція в Laravel має загальну структуру, яка включає наступні елементи.

Назва міграції: кожна міграція має унікальну назву, яка зазвичай відображає її мету або зміст. Наприклад, «create_users_table» або «add_email_column_to_users».

Метод «up»: цей метод містить код, який виконується при виконанні міграції. В ньому описуються зміни, які потрібно внести до бази даних, такі як створення таблиці, додавання стовпців або індексів.

Метод «down»: цей метод містить код, який виконується при відкаті міграції. Він визначає зміни, які необхідно відмінити або відкотити, щоб повернути базу даних до попереднього стану.

Метод «create» або «table» (необов'язковий): цей метод використовується для створення нової таблиці або редагування існуючої. В ньому визначаються назва таблиці та її структура, включаючи назви стовпців, типи даних, обмеження та індекси.

Інші методи: Laravel надає різні методи, які можна використовувати в міграціях для виконання специфічних змін. Наприклад, «addColumn», «dropColumn», «renameTable» та інші.

Це загальний вид міграції в Laravel. Використовуючи цю структуру, розробники можуть описати зміни в базі даних та ефективно керувати ними під

час розробки та підтримки веб-додатків.

Тепер знаючи це, створюємо міграції відповідні до описаної у другому розділі ER-діаграми (рис. 3.1).

```php
<?php

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

return new class extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->string('role');
            $table->string('surname');
            $table->string('birthday');
            $table->string('country');
            $table->string('phone');

            $table->rememberToken();
            $table->timestamps();
        });
    }

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

Рисунок 3.1 – Вигляд міграції в laravel на прикладі користувача

3.1.2 Роутінг

Роутинг в Laravel – це механізм, що дозволяє встановлювати зв'язок між URL-шляхами та функціями або методами контролерів, які обробляють запити

веб-додатку. Основна структура роутів в Laravel має наступний вигляд.

Визначення роута: використовуючи метод Route, визначаються роути для різних HTTP-методів, таких як GET, POST, PUT, DELETE і т.д.

URL-шлях: вказується шлях або шаблон URL-адреси, за яким буде доступний роут. Він може містити статичні та динамічні сегменти.

Дія: вказується функція або метод контролера, які виконуються при збігу роута. Це може бути звичайна функція або метод контролера, що виконує потрібну логіку обробки запиту.

Назва роута (необов'язково): для кожного роута можна встановити унікальну назву, що дозволяє посилатись на нього в інших частинах додатку.

Middleware (необов'язково): можна визначити middleware, які будуть застосовуватись до роута, що дозволяє контролювати доступ до нього та виконувати певні перевірки перед обробкою запиту.

Групування роутів: роути можуть бути згруповані, що спрощує організацію і структурування маршрутів у великих додатках.

Це загальний опис роутингу в Laravel. Використовуючи цей механізм, розробники можуть зручно визначати URL-шляхи та пов'язані з ними дії, що дозволяє ефективно обробляти запити веб-додатку.

Відповідно до цього створимо роутінг для нашої роботи (рис. 3.2).

```
30    Route::get('/meeting/{id}', [MeetingController::class, 'show']);
31    Route::post('/meeting', [MeetingController::class, 'store']);
32    Route::put('/meeting/{id}', [MeetingController::class, 'update']);
33    Route::get('/meeting', [MeetingController::class, 'index']);
34    Route::delete('/meeting/{id}', [MeetingController::class, 'destroy']);
```

Рисунок 3.2 – Основні типи роутів на прикладі роутів для зустрічі

3.1.3 Моделі

Моделі в Laravel – це класи, які представляють таблиці в базі даних і забезпечують доступ до даних. Основні характеристики моделей в Laravel.

Визначення моделі: кожна модель в Laravel представлена окремим класом, який наслідує базовий клас Illuminate\Database\Eloquent\Model.

Встановлення зв'язків: моделі можуть мати встановлені різні типи зв'язків з іншими моделями, такі як один-до-одного, один-до-багатьох, багато-до-багатьох. Це дозволяє виконувати зручні операції зв'язування та отримання пов'язаних даних.

Валідація даних: моделі можуть використовувати правила валідації для перевірки та забезпечення правильності даних перед збереженням у базу даних.

Робота з запитами: моделі надають зручний спосіб виконання запитів до бази даних, використовуючи методи, такі як find, create, update, delete та інші.

Масове присвоєння атрибутів: моделі дозволяють масово присвоювати значення атрибутів з масиву даних, що спрощує процес створення та оновлення записів.

Мутатори та аксесори: моделі можуть мати методи-мутатори, які автоматично обробляють значення атрибутів перед збереженням, а також методи-аксесори, які дозволяють отримувати значення атрибутів з моделі з власною логікою.

М'яке видалення: Laravel надає можливість використовувати «м'яке видалення», коли запис не видаляється фактично з бази даних, а лише отримує статус «видалений». Це дозволяє відновлювати видалені записи та забезпечує безпеку даних.

Це загальний огляд моделей в Laravel. Використовуючи моделі, розробники можуть зручно працювати з даними, виконувати запити до бази даних і забезпечувати цілісність даних у своєму Laravel-додатку.

Опираючись на дану вище інформацію створимо моделі, необхідні для подальшої роботи (рис. 3.3).

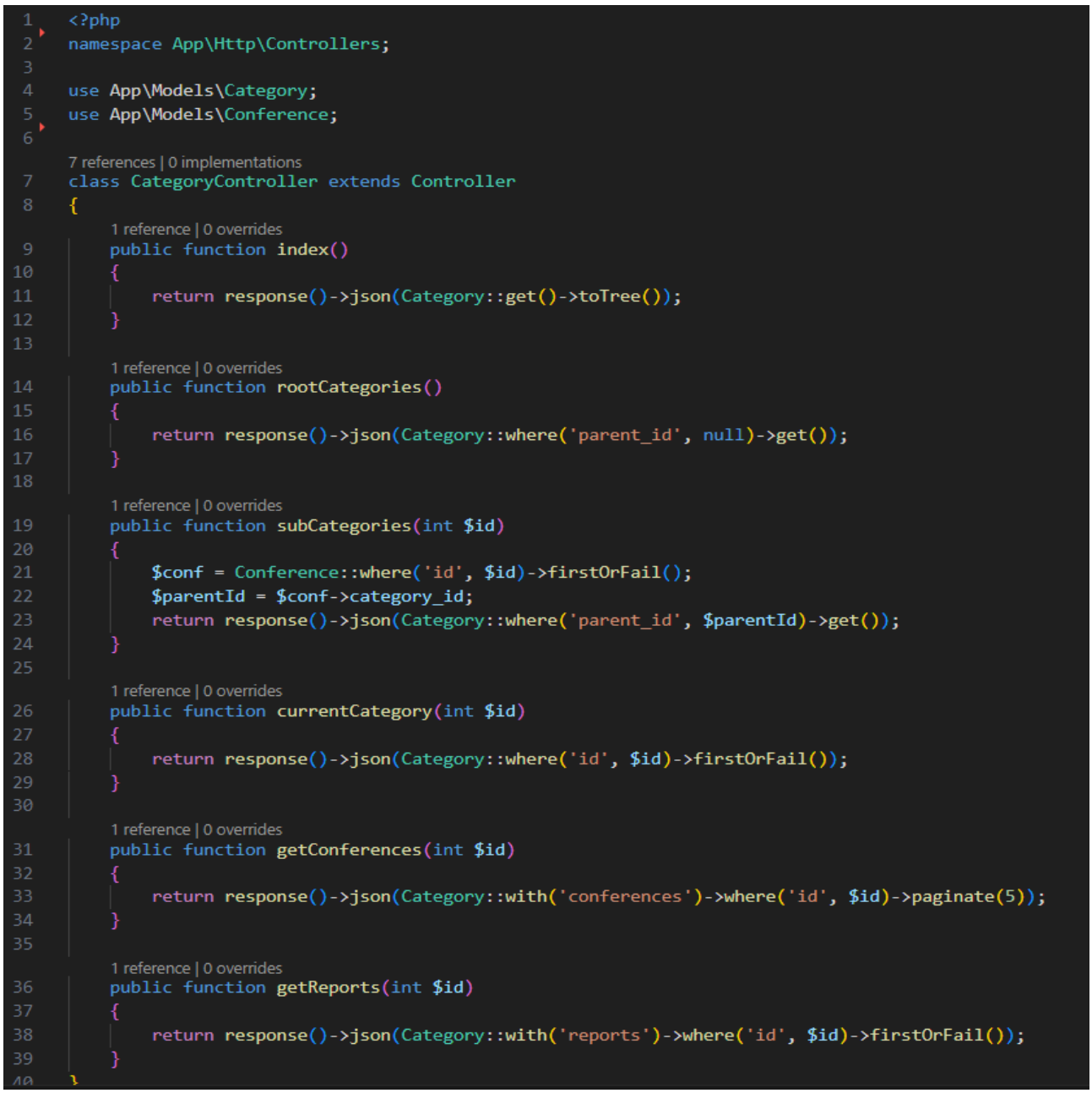
```php
<?php

namespace App\Models;

use App\Models\Conference;
use App\Models\Report;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Kalnoy\Nestedset\NestedSet;
use Kalnoy\Nestedset\NodeTrait;

30 references | 0 implementations
class Category extends Model
{
    use HasFactory, NodeTrait;

    0 references
    protected $fillable = [
        'id',
        'name'
    ];

    0 references | 0 overrides
    public function categories()
    {
        return $this->hasMany(Category::class);
    }

    0 references | 0 overrides
    public function reports()
    {
        return $this->hasMany(Report::class);
    }

    0 references | 0 overrides
    public function conferences()
    {
        return $this->hasMany(Conference::class);
    }
}
```

Рисунок 3.3 – Вигляд коду моделі на прикладі моделі категорії

3.1.4 Контролери

Використовуючи контролери, розробники можуть ефективно розділити логіку обробки запитів та бізнес-логіку свого додатку. На рисунку 3.4 представлено створені контролери, які необхідні будуть в подальшій роботі.

Створенню цих контролерів передував загальний огляд контролерів в Laravel. Наведемо нижче основні моменти.



```php
<?php
namespace App\Http\Controllers;

use App\Models\Category;
use App\Models\Conference;

7 references | 0 implementations
class CategoryController extends Controller
{
    1 reference | 0 overrides
    public function index()
    {
        return response()->json(Category::get()->toTree());
    }

    1 reference | 0 overrides
    public function rootCategories()
    {
        return response()->json(Category::where('parent_id', null)->get());
    }

    1 reference | 0 overrides
    public function subCategories(int $id)
    {
        $conf = Conference::where('id', $id)->firstOrFail();
        $parentId = $conf->category_id;
        return response()->json(Category::where('parent_id', $parentId)->get());
    }

    1 reference | 0 overrides
    public function currentCategory(int $id)
    {
        return response()->json(Category::where('id', $id)->firstOrFail());
    }

    1 reference | 0 overrides
    public function getConferences(int $id)
    {
        return response()->json(Category::with('conferences')->where('id', $id)->paginate(5));
    }

    1 reference | 0 overrides
    public function getReports(int $id)
    {
        return response()->json(Category::with('reports')->where('id', $id)->firstOrFail());
    }
}
```

Рисунок 3.4 – Вигляд коду контролера на прикладі контролера для категорій

Контролери в Laravel – це класи, які містять методи для обробки запитів і виконання бізнес-логіки у веб-додатку. Основні характеристики контролерів в Laravel.

Визначення контролера: кожен контролер в Laravel представлений окремим класом, який може містити різні методи для обробки різних запитів.

Маршрутизація запитів: контролери пов'язані з маршрутами (routes) в додатку. Визначаючи маршрути, можна вказати, який метод контролера буде виконуватись при збігу запиту з URL-шляхом.

Обробка запитів: контролери містять методи, які обробляють запити і виконують певні дії. Наприклад, метод index() може повертати список елементів, метод store() може зберігати новий запис у базі даних тощо.

Взаємодія з моделями: контролери можуть спілкуватись з моделями для отримання та збереження даних. Вони можуть використовувати методи моделей для виконання запитів до бази даних та маніпуляцій з даними.

Передача даних у вигляд (views): контролери можуть передавати дані у вигляд (views) для відображення інформації на веб-сторінках. Вони можуть використовувати методи для передачі даних та відображення відповідного шаблону.

Middleware: контролери можуть бути пов'язані з middleware, що дозволяє виконувати певні перевірки аутентифікації, авторизації або інші обробки перед обробкою запиту.

Розширення функціональності: Laravel надає можливість розширити функціональність контролерів шляхом використання трейтів або власних класів для додаткової логіки.

3.1.5 Middlewares

Мідлвари (Middleware) в Laravel – це проміжні компоненти, які оброблюють запити до вашого додатку перед тим, як вони потрапляють до маршрутів або після обробки відповідей перед їх поверненням клієнту. Основні характеристики мідлварів в Laravel.

Обробка запиту: мідлвари можуть виконувати певні операції або перевірки перед обробкою запиту. Наприклад, вони можуть перевіряти аутентифікацію користувача, перевіряти права доступу або проводити валідацію даних.

Послідовність виконання: мідлвари виконуються у певному порядку, який визначається в файлі App\Http\Kernel. Це дозволяє контролювати

послідовність виконання різних мідлварів.

Можливість перехопити запит та відповідь: мідлвари можуть перехопити запит та відповідь, що дозволяє модифікувати їх або додавати додаткові дані до них перед їх обробкою або поверненням.

Фільтрація і обробка помилок: мідлвари можуть бути використані для фільтрації запитів, відстеження помилок або обробки винятків, що спрощує керування помилками у додатку.

Групування мідлварів: мідлвари можуть бути груповані, що дозволяє застосовувати їх до групи маршрутів або контролерів.

Створення власних мідлварів: розробники можуть створювати свої власні мідлвари для виконання специфічних завдань або перевірок у своєму додатку.

Доступ до об'єктів запиту та відповіді: мідлвари мають доступ до об'єкта запиту та відповіді, що дозволяє виконувати операції з ними або модифікувати їх.

Мідлвари в Laravel дозволяють розробникам зручно контролювати потік запитів у своєму додатку, забезпечуючи розширення та перевикористання логіки обробки (рис. 3.5).

```php
<?php

declare (strict_types = 1);

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Auth, Exception;

1 reference | 0 implementations
class isRigthAnnouncer
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    0 references | 0 overrides
    public function handle(Request $request, Closure $next)
    {
        if (!Auth::check() || Auth::user()->role !== 'announcer') {
            abort(403, 'Access denide');
        }
        return $next($request);
    }
}

```

Рисунок 3.5 – Вигляд мідлвару у коді на прикладі того, що перевіряє користувача на роль лектора

3.2 Клієнтська частина

У даному підрозділі розглядається клієнтська частина розробки веб-додатка, заснованого на фреймворку Laravel. Клієнтська частина відповідає за візуальне відображення даних та взаємодію користувача з додатком.

Основна мета клієнтської частини – створення зручного та привабливого користувацького інтерфейсу, який забезпечить комфортну навігацію та інтерактивність веб-додатка. Для досягнення цієї мети використовуються сучасні веб-технології, такі як HTML, CSS та JavaScript, а також фреймворки та бібліотеки, включаючи Vue.js, React або Angular.

У даному розділі буде розглянуто використання Vue.js як клієнтського фреймворка, оскільки він надає зручні інструменти для створення динамічних та реактивних користувацьких інтерфейсів. Будуть описані основні концепції роботи з Vue.js, а також його інтеграція з Laravel для забезпечення плавної та ефективної взаємодії між клієнтською та серверною частинами додатка.

Крім цього, буде розглянуто використання AJAX-запитів для взаємодії з сервером та отримання та оновлення даних без перезавантаження сторінки. Це дозволяє створювати більш плавну та реактивну користувацьку взаємодію та покращує загальний досвід використання додатка.

У цьому підрозділі будуть надані практичні приклади та пояснення роботи з клієнтською частиною веб-додатка на основі Laravel та Vue.js.

3.2.1 Ajax-запити

AJAX-запити (Asynchronous JavaScript and XML) – це технологія, що дозволяє взаємодіяти з сервером без перезавантаження сторінки. Використовуючи AJAX, можна асинхронно відправляти запити на сервер і отримувати від нього дані у форматі JSON, XML або інших.

Головна перевага використання AJAX-запитів полягає в тому, що користувач може взаємодіяти з веб-додатком, не очікуючи повного перезавантаження сторінки. Це дозволяє створювати більш плавні та

інтерактивні користувацькі інтерфейси.

Для виконання AJAX-запитів у веб-додатках зазвичай використовуються JavaScript-бібліотеки, такі як jQuery, або фреймворки, такі як Vue.js або React. За допомогою цих інструментів можна легко виконувати запити на сервер, обробляти результати та оновлювати вміст веб-сторінки без перезавантаження.

AJAX-запити дозволяють створювати динамічні веб-додатки з реактивною взаємодією, покращуючи продуктивність та зручність використання для користувачів.

Для реалізації запитів будемо використовувати axios. Axios – це JavaScript-бібліотека, яка дозволяє легко виконувати HTTP-запити з використанням Promise API. Вона широко використовується для звернення до сервера та отримання або відправлення даних у форматі JSON.

Основні переваги використання Axios включають простоту використання, можливість перехоплювати помилки, автоматичну серіалізацію та десеріалізацію даних, а також можливість налаштувати заголовки та параметри запиту.

За допомогою Axios можна виконувати різні типи запитів, такі як GET, POST, PUT, DELETE та інші. Вона також надає зручні інструменти для обробки результатів запиту, зокрема роботу з відповідями у форматі JSON.

Завдяки своїй простоті та потужним можливостям, Axios став популярним вибором для виконання HTTP-запитів в багатьох веб-додатках на основі JavaScript та фреймворків, таких як Vue.js або Reac (рис. 3.6).

```
112         ajaxGetCurrentPlan({ commit }) {
113             let token = "Bearer " + localStorage.getItem("Authorized");
114             return axios({
115                 method: "get",
116                 url: "api/plan",
117                 headers: {
118                     Authorization: token,
119                     "Content-type": "application/json; charset=UTF-8",
120                 },
121             }).then((response)=>{
122                 commit("setCurrentPlan", response.data);
123             })
124         },
```

Рисунок 3.6 – Запит до бази даних з клієнтської частини для отримання дійсного плану користувача

3.2.2 Роутінг

Маршрутизація в Vue здійснюється за допомогою Vue Router, який є офіційним маршрутизатором для Vue.js. Vue Router дозволяє організовувати навігацію в веб-додатку та встановлювати відповідні маршрути для кожного компонента.

Для використання Vue Router потрібно спочатку встановити його через npm або інший пакетний менеджер. Після цього можна імпортувати Vue Router у своєму проєкті і налаштувати маршрути.

Основною концепцією Vue Router є створення об'єкту маршрутизації, в якому описуються шляхи (routes) та компоненти, які відповідають кожному шляху. Кожен шлях має URL-шаблон і пов'язаний з ним компонент.

У Vue Router є кілька методів, які дозволяють виконувати навігацію між сторінками, такі як router-link, який створює посилання на іншу сторінку, і методи програмної навігації, такі як router.push() або router.replace().

Крім того, Vue Router надає можливість передавати параметри та опціональні сегменти у шляхах, що дозволяє створювати динамічні маршрути.

Усе це дозволяє ефективно організувати маршрутизацію в Vue-додатку, дозволяючи користувачам навігуватися між сторінками та відображати відповідні компоненти в залежності від маршруту (рис. 3.7).

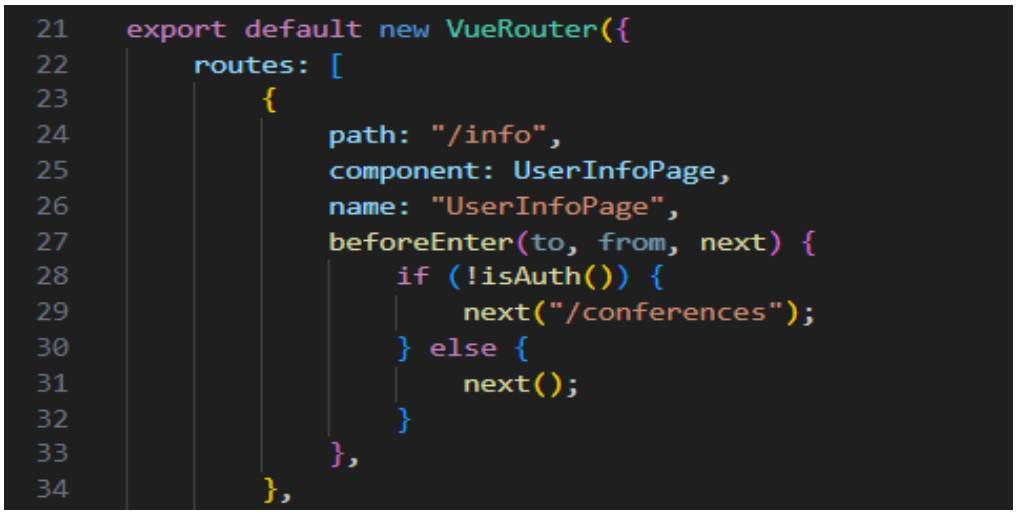


Рисунок 3.7 – Приклад компоненту роутінга

3.2.3 Верстка

Верстка в Vue з використанням Vuetify - це процес створення і організації вигляду і розміщення елементів на сторінці з використанням Vuetify, що є бібліотекою компонентів і стилів для Vue.js.

Vuetify надає набір готових компонентів, які можна використовувати для побудови інтерфейсу користувача. Ці компоненти мають заздалегідь визначений стиль і функціонал, що дозволяє швидко і зручно створювати привабливі і сучасні веб-інтерфейси.

Для використання Vuetify потрібно спочатку встановити його за допомогою npm або іншого пакетного менеджера. Після цього можна імпортувати Vuetify у свій проєкт і використовувати його компоненти.

Основними концепціями Vuetify є використання сітки (grid system) для розміщення компонентів, використання готових стилів та класів для оформлення, а також можливість налаштовувати компоненти за допомогою властивостей і подій.

Vuetify також надає можливість розширювати і налаштовувати компоненти, що дозволяє створювати власні компоненти на основі наявних.

З використанням Vuetify можна швидко та зручно створювати естетичний інтерфейс користувача зі стандартними компонентами, що значно спрощує процес верстки в Vue (рис. 3.8).

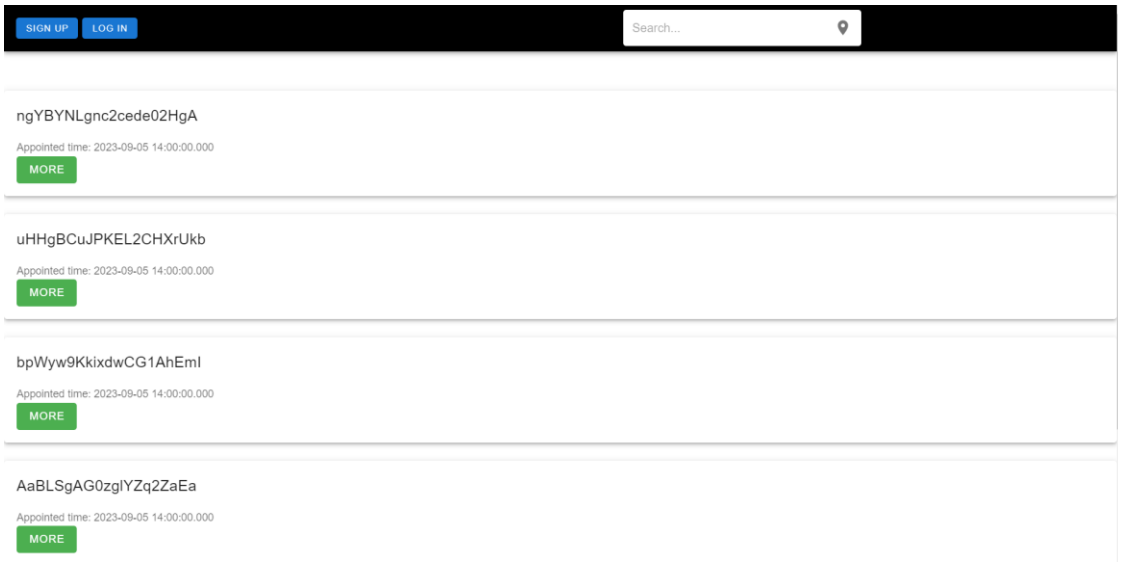


Рисунок 3.8 – Приклад сторінки створеної с використанням Vuetify

3.3 Адміністративна частина

У цьому підрозділі я бажаю описати, як була застосована Laravel Nova для реалізації адміністративної частини веб-сайту. Використовуючи стандартні функції та моделі Laravel, було здійснено інтеграцію Laravel Nova для швидкого та зручного управління адміністративними функціями.

За допомогою Laravel Nova були створені адміністративні моделі, які відображали основні сутності сайту, такі як користувачі, статті, замовлення тощо (рис. 3.9). За допомогою стандартних полів та відношень між моделями, було забезпечено зручний інтерфейс для редагування та керування цими даними.

```php
class Category extends Resource
{
    0 references
    public static $model = \App\Models\Category::class;

    0 references
    public static $title = 'id';

    0 references
    public static $search = [
        'id', 'name', 'parent_id', 'created_at', 'updated_at',
    ];

    1 reference
    private function getParentId()
    {
        $list = array();
        foreach (\App\Models\Category::where('parent_id', null)->get() as $category) {
            $list[$category['id']] = $category['name'];
        }
        return $list;
    }

    0 references
    private function get()
    {
        return \App\Models\Category::get();
    }

    0 references | 0 overrides
    public function fields(NovaRequest $request)
    {
        return [
            Text::make('Name', 'name')
            ->sortable()
            ->rules('required', 'max:255'),
            Select::make('Patend ID', 'parent_id')->options($this->getParentId()),
            Text::make('Created at', 'created_at')
            ->sortable()
            ->rules('required', 'max:255')
            ->exceptOnForms(),
            Text::make('Updated at', 'updated_at')
            ->sortable()
            ->rules('required', 'max:255')
            ->exceptOnForms(),
        ];
    }
```

Рисунок 3.9 – Частина коду Laravel Nova для моделі категорій

Крім того, Laravel Nova надає можливості фільтрації, сортування та пошуку даних, що значно спрощує навігацію та пошук в адміністративній частині сайту. За допомогою цих функцій, адміністратор сайту може швидко знаходити необхідні записи та здійснювати необхідні зміни.

Крім того, Laravel Nova дозволяє налаштовувати права доступу до адміністративних функцій для різних користувачів. Це дозволяє обмежити доступ до певних дій або моделей тим користувачам, які мають відповідні права.

Загалом, використання Laravel Nova дозволило ефективно реалізувати адміністративну частину веб-сайту, надаючи зручний інтерфейс та широкі можливості для управління даними та адміністративними функціями. Це сприяло покращенню ефективності роботи адміністраторів та забезпечило гнучкість для майбутнього розширення функціоналу сайту.

3.4 Тестування

У цьому підрозділі проводиться функціональне тестування розробленої системи з метою перевірки її відповідності функціональним вимогам і очікуванням користувачів. Функціональне тестування дозволяє виявити потенційні проблеми та дефекти в роботі системи, а також переконатися, що всі функції працюють належним чином. Метою функціонального тестування є забезпечення роботи системи відповідно до очікувань користувачів, виявлення потенційних дефектів та покращення її якості. У данній роботі я буду використовувати фічер тести (рис. 3.10).

Фічер-тести є важливою складовою частиною тестування в Laravel і дозволяють перевірити роботу окремих функціональних можливостей (фічерів) системи. Вони базуються на популярній бібліотеці для тестування PHP-додатків – PHPUnit.

```
55      public function test_with_no_payment_token_to_basic_success_plan_change()
56      {
57          $user = User::factory()->create();
58          Sanctum::actingAs(
59              $user,
60              ['*']
61          );
62          $response = $this->postJson('/api/subscribe', [
63              'plan' => 'Basic',
64          ]);
65          Subscription::where('user_id', $user->id)->delete();
66          $user->delete();
67          $response
68              ->assertStatus(200);
69      }
```

Рисунок 3.10 – Приклад коду тесту

Основні особливості фічер-тестів в Laravel.

Структурованість: фічер-тести організовані у вигляді методів класу тесту. Кожен метод відповідає окремому функціоналу або фічеру системи, що дозволяє зручно організувати тести та підтримувати чистоту коду.

Симуляція взаємодії з системою: фічер-тести дають можливість симулювати взаємодію користувача з системою шляхом написання коду, що виконує запити до маршрутів або взаємодіє з контролерами системи.

Перевірка очікуваних результатів: у фічер-тестах можна перевірити правильність роботи функціоналу, порівнюючи отримані результати з очікуваними. Для цього використовуються різноманітні перевірки, такі як перевірка рядків, масивів, HTTP-статусів тощо.

Підготовка тестових даних: Laravel надає зручні механізми для підготовки тестових даних, які можна використовувати у фічер-тестах. Це дозволяє створювати потрібні дані для виконання тестових сценаріїв та перевірки функціоналу.

Інтеграція з іншими тестами: фічер-тести можна поєднувати з іншими видами тестів, такими як одиницеві тести та прийомочні тести, для комплексного перевірки функціоналу системи.

Фічер-тести в Laravel є потужним інструментом для тестування окремих функцій та фічерів системи. Вони дозволяють перевірити коректність роботи коду, перевірити очікувані результати та забезпечити якість функціоналу

системи.

Рисунок 3.11 – Результат виконання тестів

3.5 Висновки до розділу 3

В рамках розділу «Реалізація та тестування» було виконано розробку та тестування системи. Процес реалізації включав в себе створення серверної та клієнтської частини з використанням Laravel і Vue.js відповідно. Були реалізовані основні функції, такі як реєстрація користувачів, управління конференціями, генерація звітів та коментування.

Для забезпечення якості розробленої системи були використані функціональні тести з використанням фічер-тестів в Laravel. Ці тести дозволили перевірити коректність роботи окремих функціональних можливостей системи. Також були виконані інтеграційні тести для перевірки взаємодії між компонентами системи. В процесі тестування були виявлені та виправлені деякі помилки та проблеми, що допомогло покращити якість системи.

У результаті реалізації та тестування було створено функціональну систему, яка задовольняє поставлені вимоги та забезпечує коректну роботу відповідно до специфікацій. Дана система готова до подальшого розгортання та використання в реальних умовах.

ВИСНОВКИ

В рамках дипломної роботи була проведена розробка та реалізація вебдодатку для управління конференціями. Робота включала аналіз вимог, проєктування системи, реалізацію функціональності, тестування та оцінку якості. Нижче наведений висновок щодо результатів дипломної роботи:

Основні результати:

– було розроблено функціональний вебдодаток, який забезпечує можливість реєстрації користувачів, управління конференціями, генерацію звітів та коментування;
– вебдодаток був реалізований з використанням Laravel у серверній частині та Vue.js у клієнтській частині, що дозволяє забезпечити зручний та ефективний інтерфейс користувача;
– були використані різні технології та інструменти, такі як база даних MySQL, мова програмування PHP, фреймворк Laravel, бібліотека Vue.js та інші, для досягнення поставлених цілей проєкту.

Оцінка якості:

– в процесі тестування було виявлено та виправлено деякі помилки та проблеми, що сприяло поліпшенню якості системи;
– були застосовані функціональні тести для перевірки коректності роботи системи та взаємодії між її компонентами.

В цілому, дипломна робота була успішно завершена, мета проєкту була досягнута, а розроблений веб-додаток готовий до використання. Результати роботи можуть бути використані для подальшого розгортання та використання системи в реальних умовах.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документації Laravel. URL: https://laravel.com/docs/10.x (дата звернення: 03.04.2023).
2. He R. Y. Design and implementation of web based on Laravel framework. 2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014). Atlantis Press, 2015. P. 301-304.
3. Документація VueJS. URL: https://vuejs.org/guide/introduction.html (дата звернення: 04.04.2023).
4. Nelson B. Getting to Know Vue. Js. New York : Apress, 2018. 488 p.
5. Документація Laravel Nova. URL: https://nova.laravel.com (дата звернення: 03.04.2023).
6. Svennerberg G. Beginning google maps API 3. New York : Apress, 2010. 330 p.
7. Документація Гугл. URL: https://developers.google.com/maps?hl=en (дата звернення: 25.04.2023).
8. Документація Страйп. URL: https://stripe.com/docs/api (дата звернення: 25.04.2023).

ДОДАТОК А

Сторінки вебдодатка

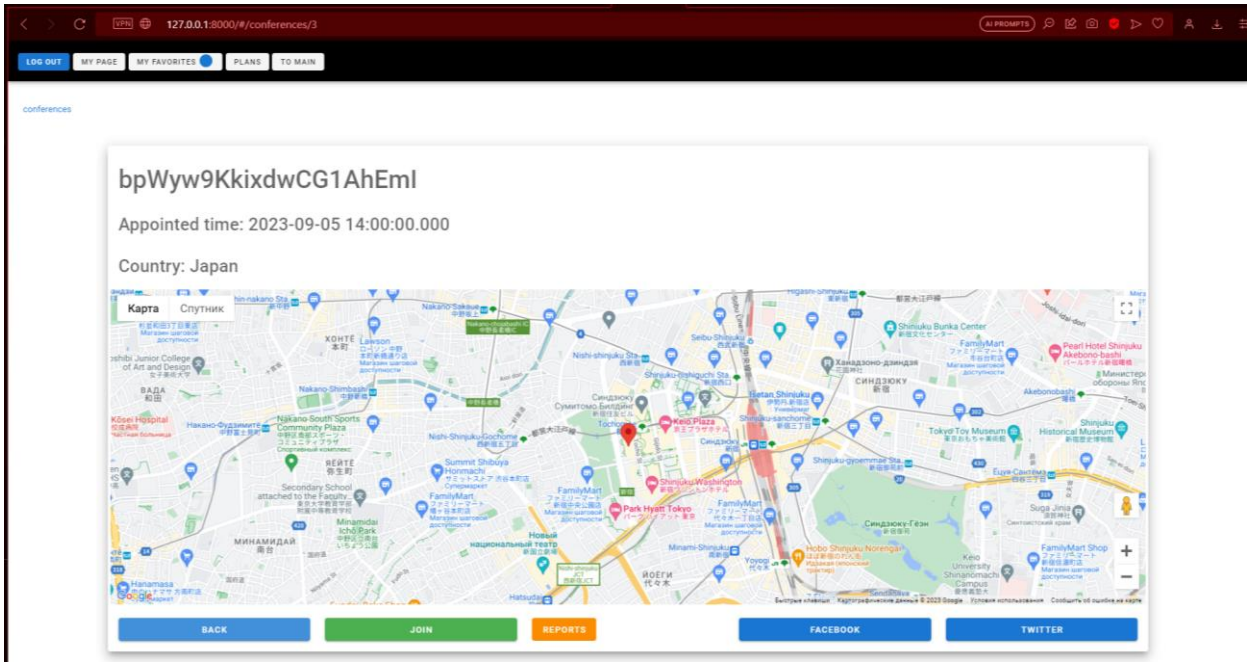


Рисунок А.1 – Сторінка перегляду конференції

Рисунок А.2 – Головна сторінка

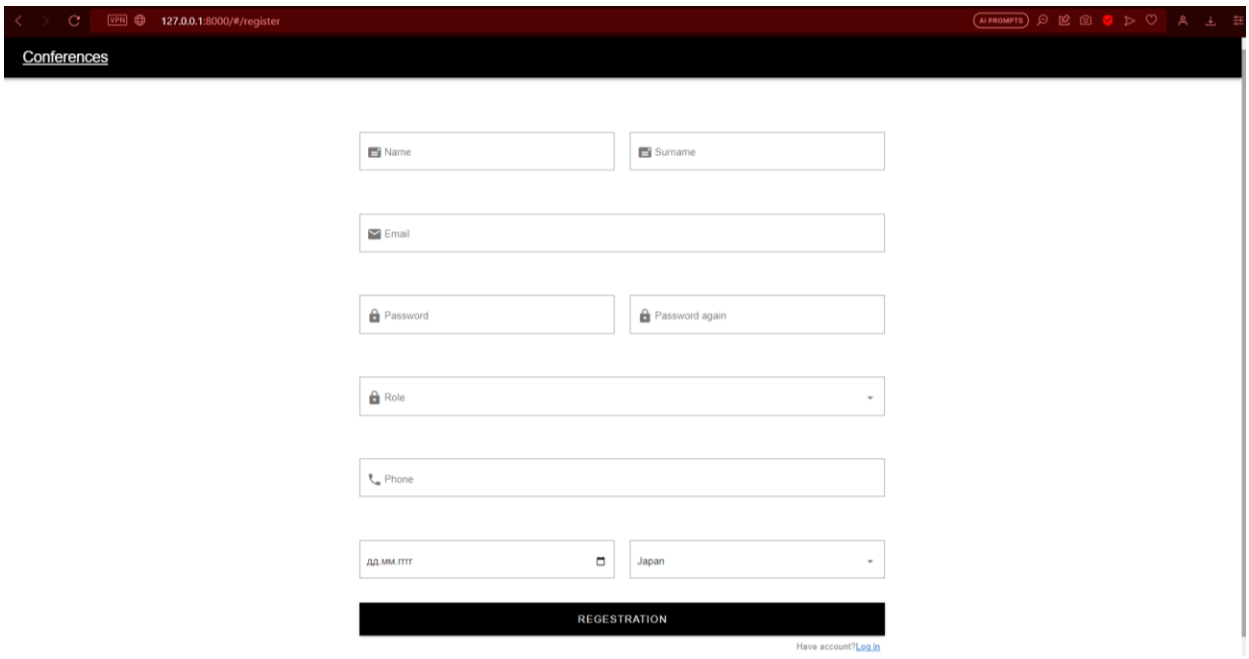


Рисунок А.3 – Сторінка реєстрації

Рисунок А.4 – Сторінка докладу

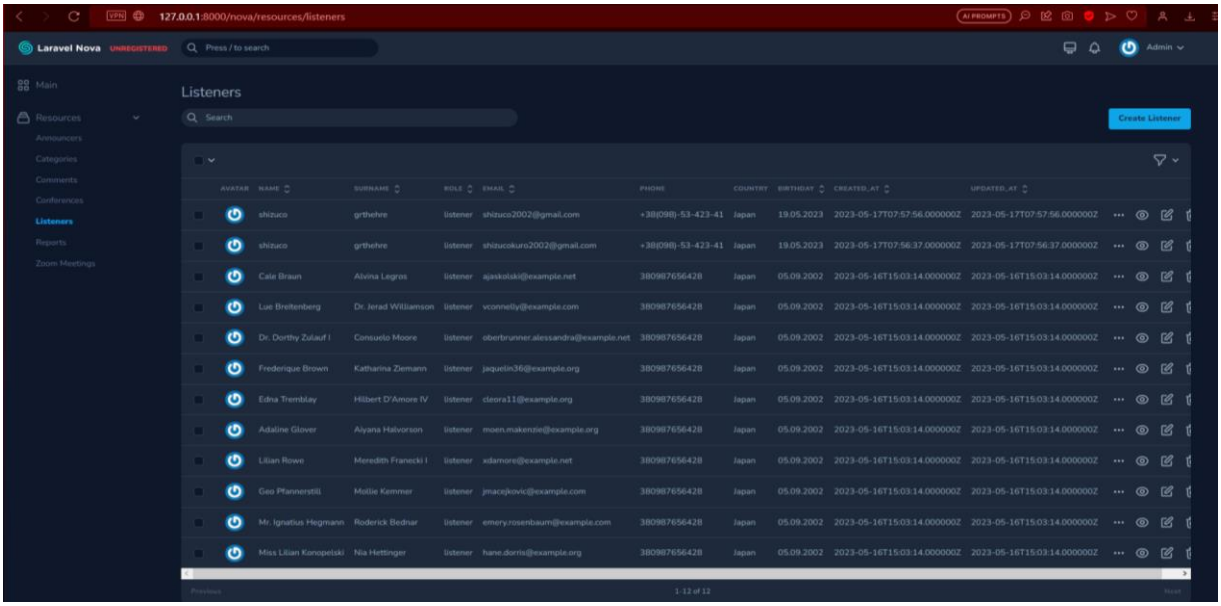


Рисунок А.5 – Вид стандартної таблиці в нові

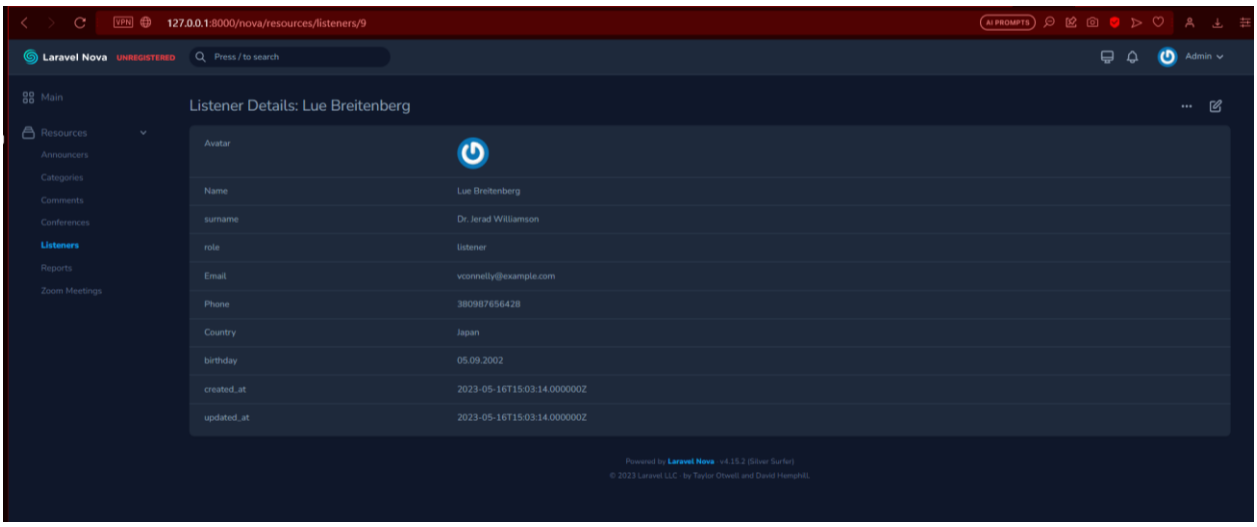


Рисунок А.6 – Вигляд сторінки деталей у нові

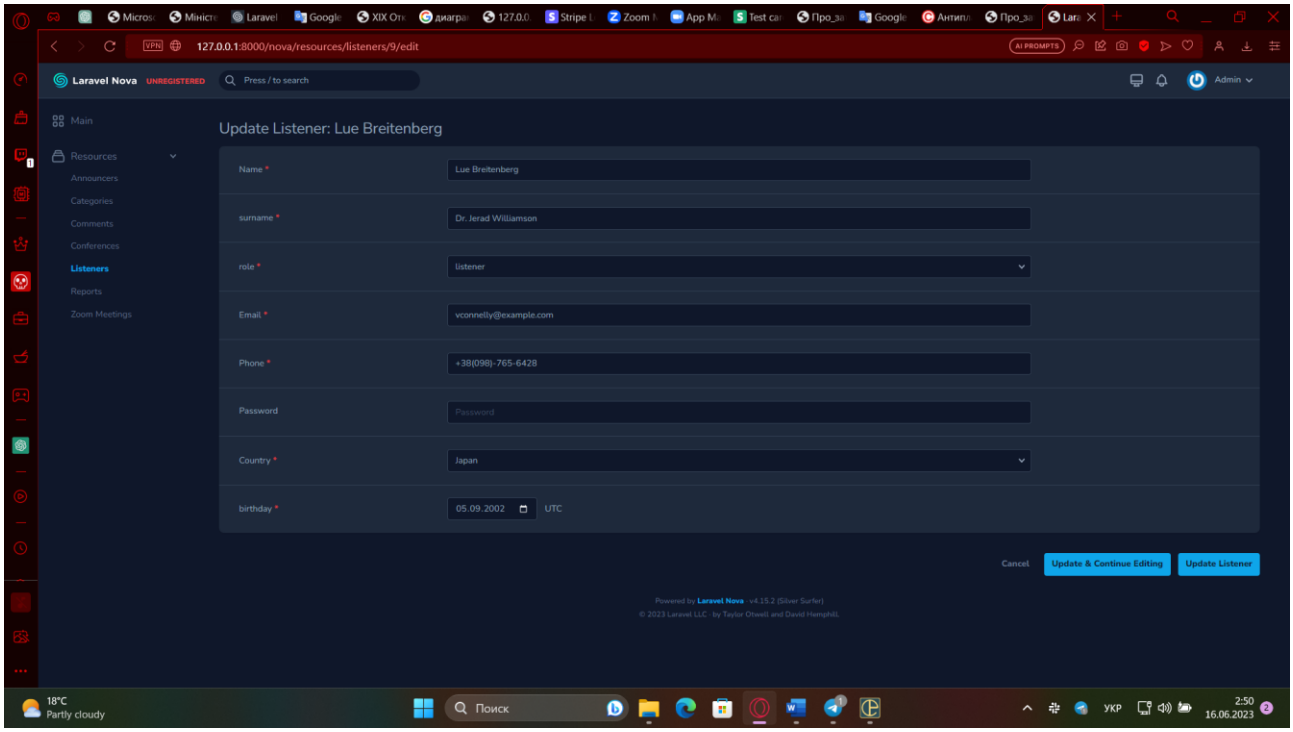


Рисунок А.7 – Сторінка редактування у нові

ДОДАТОК Б

Код розробленного додатку

Посилання на GitHub: https://github.com/Shizuco/conference-site-laravel-nova.