МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБДОДАТКУ ДЛЯ РЕЄСТРАЦІЇ НА КОНФЕРЕНЦІЇ З ВИКОРИСТАННЯМ LARAVEL TA VUEJS»**

Виконав: студент 4 курсу, групи 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Е.Р. Кестан
(ініціали та прізвище)

Керівник декан математичного факультету, професор, д.т.н. Гоменюк С.І.
(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 – 2024

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)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кестану Едему Рустемовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебдодатку для реєстрації на конференції з використанням Laravel та VueJs

керівник роботи Гоменюк Сергій Іванович, д.т.н., професор
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 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. Проєктування та розробка вебзастосунку реєстрації користувачів на конференцію.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 25.12.2023 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.	09.01.2024	
2.	Збір вихідних даних.	23.01.2024	
3.	Обробка методичних та теоретичних	27.02.2024	
	джерел.		
4.	Розробка першого та другого розділу.	12.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль		
	кваліфікаційної роботи бакалавра.	27.05.2024	
7.	Захист кваліфікаційної роботи.	17.06.2024	

Студент _____ Е.Р. Кестан
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.І. Гоменюк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебдодатку для реєстрації на конференції з використанням Laravel та VueJs»: 45 с., 15 рис., 10 джерел.

БАЗА ДАНИХ, ВЕБДОДАТОК, ВЕБЗАСТОСУНОК, КОМПОНЕНТ, КОНТРОЛЕР, LARAVEL, МОДЕЛЬ, ФРЕЙМВОРК, VUE.JS.

Об’єкт дослідження – процес розробки вебдодатку для реєстрації на конференції.

Мета роботи: розробка вебдодатку для реєстрації на конференції з використанням фреймворків Laravel та Vue.js.

Методи дослідження – методи об’єктно-орієнтованого програмування, методи програмної інженерії.

Предмет дослідження – розробка вебдодатку для реєстрації на конференції з використанням фреймворків Laravel та Vue.js.

У роботі наведено етапи розробки вебдодатку для реєстрації на конференції з використанням фреймворків Laravel та Vue.js, виявлено ключові проблеми та застосовано нові методи для збору та аналізу вимог до програмного забезпечення. Описано предметну область. Розроблено вебдодаток для реєстрації на конференції з використанням фреймворків Laravel та Vue.js, який використовується для зручної реєстрації учасників на конференції та управління подіями в реальному часі. Зростання популярності конференційних заходів вимагає ефективних інструментів для організації реєстрації та управління учасниками, що робить розробку вебдодатку для цієї цілі актуальною задачею.

SUMMARY

Bachelor’s qualifying paper “Development of a Web Application for Conference Registration Using Laravel and VueJs”: 45 pages, 15 figures, 10 references.

DATABASE, WEBDODATOK, WEBSTORE, COMPONENT, CONTROLLER, LARAVEL, MODEL, FRAMEWORK, VUE.JS.

Research object – the process of developing a web application for conference registration.

Objective of the study – development of a web application for conference registration using Laravel and Vue.js.

Research methods – object-oriented programming methods, software engineering methods.

Research subject – development of a web application for conference registration using Laravel and Vue.js.

The thesis outlines the stages of developing a web application for conference registration using Laravel and Vue.js, identifies key issues, and applies new methods for gathering and analyzing software requirements. The domain area is described, and a web application for conference registration and real-time event management is developed using Laravel and Vue.js. The increasing popularity of conference events demands effective tools for registration and participant management, making the development of such a web application a relevant task.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 7
1 Теоретичні основи 9
1.1 Основні концепції вебдодатків із застосуванням фреймворків 9
1.2 Вибір та використання баз даних 13
1.3 Архітектурні принципи та патерни проєктування вебдодатків 18
1.4 Основні можливості та переваги фреймворків 20
2 Моделювання та проєктування 24
2.1 Визначення вимог до системи 24
2.2 Проєктування структури бази даних 25
2.3 Проєктування інтерфейсу користувача 29
2.4 Діаграмма прецедентів та активностей 30
3 Реалізація та тестування 33
3.1 Розробка основної функціональності вебдодатку 33
3.2 Реалізація додаткових методів 36
3.3 Тестування функціональності та модульне тестування 38
3.4 Результат та висновок тестування та реалізації вебзастосунку 42
Висновки 43
Перелік посилань 45

ВСТУП

У сучасному інформаційному суспільстві розвиток вебдодатків для реєстрації на конференції визнається однією з найбільш актуальних та важливих напрямків розробки програмного забезпечення. Швидкий темп технологічного прогресу та зростаючі вимоги користувачів стимулюють постійне удосконалення цих інструментів, спрямованих на оптимізацію процесів реєстрації учасників та управління конференціями. Використання сучасних фреймворків, зокрема Laravel і Vue.js, в розробці вебдодатків дозволяє не лише підвищити ефективність програмного забезпечення, а й забезпечити його високу якість та відповідність сучасним стандартам.

Метою даної кваліфікаційної роботи є детальний аналіз та розробка вебдодатку для реєстрації на конференції з використанням фреймворків Laravel та Vue.js. Для досягнення цієї мети поставлені такі завдання:

- проаналізувати потреби та вимоги щодо функціональності вебдодатку;
- дослідити та вибрати оптимальний технологічний стек для реалізації проєкту;
- розробити архітектуру системи та моделі даних для ефективного зберігання і обробки інформації про учасників та події конференції;
- спроєктувати інтуїтивно зрозумілий інтерфейс користувача, що забезпечує зручну реєстрацію та управління подіями;
- реалізувати програмний код вебдодатку, використовуючи можливості Laravel та Vue.js для забезпечення швидкодії та масштабованості системи;
- провести тестування для впевненості у функціональності та надійності розробленого продукту.

Об’єктом дослідження є процес розробки вебдодатку для реєстрації на конференції, тоді як предметом є вивчення ключових аспектів та вирішення

викликів, що виникають під час його створення. Дослідження проводиться з використанням методів об’єктно-орієнтованого програмування та програмної інженерії, що дозволяє ефективно реалізувати поставлені завдання та досягти поставлених цілей.

Для досягнення результату використано мову програмування JavaScript, що забезпечує універсальність і можливість розробки як серверної, так і клієнтської частини вебдодатку. В якості системи управління базами даних обрано MongoDB, яка відповідає потребам зберігання даних у форматі JSON і сприяє зручності роботи з неструктурованою інформацією.

Отримані результати демонструють не лише практичне застосування вибраних технологій, а й їхнє теоретичне значення у контексті розробки високоефективних та масштабованих вебдодатків для сучасної електронної комерції та організації подій. Розроблений вебдодаток може знайти застосування у різних сферах, де важлива швидкість реакції, надійність та зручність використання онлайн-сервісів.

1 ТЕОРЕТИЧНІ ОСНОВИ

1.1 Основні концепції вебдодатків із застосуванням фреймворків

Основні концепції вебдодатків із застосуванням фреймворків охоплюють ключові аспекти проєктування, реалізації та експлуатації вебдодатків з використанням сучасних фреймворків, таких як Laravel та Vue.js. Вони надають розробникам зручний інструментарій для швидкого створення та підтримки складних вебзастосунків [9].

Основні концепції вебдодатків із застосуванням фреймворків включають:

- MVC архітектура: Модель-Вид-Контролер (Model-View-Controller) є основною концепцією, що дозволяє розділити логіку додатку на компоненти моделі (дані), представлення (відображення) та контролери (логіка обробки запитів);
- роутінг і маршрутизація: фреймворки надають зручні механізми для визначення URL-адрес для різних частин додатку і направлення їх на відповідні контролери або обробники;
- шаблонізація: використання шаблонів дозволяє відділити представлення від логіки, що полегшує розробку, підтримку і модифікацію;
- ORM (Object-Relational Mapping): використання ORM дозволяє зручно взаємодіяти з базою даних через об’єктно-орієнтований підхід, уникнувши написання складних SQL-запитів;
- фронтенд і бекенд: чітке розділення між клієнтською та серверною частинами дозволяє створювати вебдодатки, які ефективно взаємодіють з користувачем і забезпечують потрібний функціонал;
- безпека: фреймворки надають інструменти для забезпечення безпеки додатків, такі як захист від XSS (міжсайтовий скриптінг), CSRF (підроблення міжсайтових запитів) та інших атак.

Розглядаючи конкретно для вашого проєкт, можна виділити кілька ключових концепцій вебдодатку реєстрації на конференцію на базі фреймворків Laravel та Vue.js:

– SPA (Single Page Application) підхід: використання Vue.js дозволяє створити односторінкове додаток, де основна частина інтерфейсу завантажується один раз, а подальша навігація і взаємодія з сервером відбуваються без перезавантаження сторінки – це покращує користувацький досвід і спрощує розробку;
– RESTful API: Laravel забезпечує легкість створення та підтримки RESTful API, які використовуються для взаємодії між клієнтською та серверною частинами додатку – це дозволяє ефективно передавати дані та забезпечувати масштабованість системи;
– компонентний підхід: використання Vue.js дозволяє розбивати користувацький інтерфейс на багато компонентів, які можуть бути перевикористані в різних частинах додатку – це полегшує структурування та підтримку коду;
– модульність: Laravel і Vue.js підтримують модульний підхід до розробки, де кожна частина функціоналу може бути відокремлена і покращена без впливу на інші частини системи;
– автентифікація та авторизація: Laravel має вбудовану систему автентифікації і авторизації, що спрощує реалізацію захищених ресурсів в додатку. Vue.js в свою чергу дозволяє зручно інтегрувати фронтенд частину цих механізмів.

Для більш гнучкого розуміння розглянемо кожну концепцію більш детальніше.

SPA (Single Page Application) – це тип вебдодатка, який взаємодіє з користувачем, перезавантажуючи тільки частину сторінки, вміст якої змінюється, замість того щоб завантажувати всю сторінку знову. Основні концепції та особливості SPA включають:

– асинхронне завантаження даних: SPA використовує AJAX (Asynchronous JavaScript and XML) для асинхронного завантаження

даних з сервера без необхідності перезавантаження всієї сторінки – це зменшує час очікування користувача та покращує відзивчивість додатка;

- односторінковий інтерфейс: весь інтерфейс додатка завантажується один раз при старті, і змінюється динамічно відповідно до взаємодії користувача без перезавантаження сторінки – це створює плавний інтерактивний досвід для користувача;
- маршрутизація на клієнті: SPA використовує маршрутизацію на клієнті для управління станами додатка та відображення відповідних виглядів частин сторінки, наприклад, зміна URL в браузері не призводить до перезавантаження сторінки, а викликає показ нового вигляду через JavaScript;
- фронтенд та бекенд розділені: SPA дозволяє розділити фронтенд (клієнтську частину) та бекенд (серверну частину) додатка, що полегшує розробку та підтримку – фронтенд часто взаємодіє з сервером через API (Application Programming Interface);
- фреймворки та бібліотеки: для розробки SPA часто використовуються сучасні JavaScript фреймворки та бібліотеки, такі як React.js, Angular, Vue.js – вони надають потужні інструменти для створення складних односторінкових додатків та дозволяють ефективно керувати станом додатка;
- кешування та локальне сховище: SPA можуть використовувати кешування та локальне сховище для зберігання даних на клієнтській стороні, що покращує продуктивність та забезпечує доступ до даних в автономному режимі.

SPA здатні підвищити продуктивність та зручність вебдодатків, особливо при розробці додатків зі складною взаємодією користувача та потужним функціоналом [6].

RESTful API (Representational State Transfer) – це архітектурний стиль для побудови вебсервісів, який забезпечує зручний і ефективний спосіб

взаємодії між різними системами за допомогою HTTP протоколу. Розглянемо, що включають основні концепції та особливості RESTful API.

Ресурси (Resources): у REST архітектурі, дані представляються як ресурси, до яких можна отримати доступ через унікальний ідентифікатор (URI). Ресурси можуть бути будь-якими об'єктами або інформаційними сутностями, які можна управляти за допомогою API.

HTTP методи: REST використовує стандартні HTTP методи (GET, POST, PUT, DELETE, PATCH) для виконання різних операцій з ресурсами:

– GET: отримання даних з сервера (часто використовується для читання ресурсів);
– POST: створення нового ресурсу на сервері;
– PUT: оновлення існуючого ресурсу на сервері;
– DELETE: видалення ресурсу з сервера;
– PATCH: часткове оновлення ресурсу.

Стан представлення (State Representation): дані, пов'язані з ресурсами, передаються у форматі, який підтримується HTTP, такому як JSON або XML. Клієнтські програми можуть взаємодіяти з ресурсами, обмінюючись структурованою інформацією через HTTP запити та відповіді.

Безстановність (Statelessness): кожен запит до сервера у RESTful API містить всю необхідну інформацію для виконання операції. Сервер не зберігає стан клієнта між запитами, що покращує масштабованість і спрощує розробку.

Маршрутизація (Routing): у RESTful API, маршрутизація зазвичай використовується для визначення, які ресурси або колекції ресурсів обслуговуються певними HTTP методами на конкретних URL шляхах.

Серіалізація (Serialization): RESTful API використовує стандартні формати серіалізації даних, такі як JSON або XML, для обміну даними між клієнтом і сервером. Це дозволяє різним системам взаємодіяти незалежно від мов програмування чи технологій.

RESTful API є популярним інструментом для створення вебсервісів, оскільки він пропонує простий, зрозумілий і стандартизований спосіб інтеграції різних компонентів програмного забезпечення [2].

1.2 Вибір та використання баз даних

Вибір бази даних є критично важливим етапом при розробці будь-якого вебдодатку, включаючи додатки для реєстрації на конференції. Від правильного вибору залежить продуктивність, масштабованість і надійність системи. Розглянемо основні типи баз даних:

– реляційні бази даних (SQL);
– нереляційні бази даних (NoSQL).

Реляційні бази даних є основою багатьох інформаційних систем завдяки своїй структурованості, надійності та здатності до обробки складних запитів. Вони використовують реляційну модель даних, яка була вперше запропонована Едгаром Коддом у 1970 році. Розглянемо основні аспекти реляційних баз даних і їх використання в нашому проєкті [6].

Основні поняття реляційних баз даних.

Таблиці є основними структурами для зберігання даних в реляційних базах. Кожна таблиця складається з рядків (записів) і стовпців (полів). Наприклад, таблиця «користувачі» може мати стовпці «ID», «ім’я», «електронна пошта», «пароль».

Первинний ключ – це унікальний ідентифікатор для кожного запису в таблиці. Він гарантує, що кожен рядок є унікальним. Наприклад, у таблиці «користувачі» стовпець “ID” може бути первинним ключем.

Зовнішній ключ використовується для встановлення зв’язків між таблицями. Він посилається на первинний ключ в іншій таблиці. Наприклад, у таблиці «реєстрації» стовпець “user_id” може бути зовнішнім ключем, що посилається на “ID” таблиці «користувачі».

Нормалізація – це процес організації даних у базі даних для мінімізації надлишковості та запобігання аномаліям оновлення. Існує кілька нормальних форм, кожна з яких пропонує певний рівень нормалізації.

Реляційні бази забезпечують чітку структуру даних та їх цілісність завдяки використанню первинних і зовнішніх ключів [8].

Переваги реляційних баз даних:

- структурованість і цілісність даних: реляційні бази даних використовують таблиці, що дозволяє чітко структурувати дані; використання первинних та зовнішніх ключів гарантує цілісність даних і дозволяє уникнути надлишковості;
- підтримка складних запитів: SQL дозволяє виконувати складні запити, що включають операції JOIN, GROUP BY, ORDER BY, що дає можливість отримати необхідну інформацію з кількох таблиць;
- транзакційність і ACID властивості: реляційні бази даних підтримують транзакції, які забезпечують атомарність, консистентність, ізольованість і надійність (ACID), що гарантує надійність операцій з даними;
- підтримка стандартів: SQL є стандартною мовою, підтримуваною всіма основними реляційними СУБД, такими як MySQL, PostgreSQL, Oracle, Microsoft SQL Server, що забезпечує високу сумісність і портативність;
- широка підтримка і документованість: реляційні СУБД мають велику спільноту користувачів і розробників, а також багату документацію і навчальні матеріали;
- масштабованість у вертикальному напрямку: реляційні бази даних можуть легко масштабуватися вертикально (збільшення ресурсів сервера), що дозволяє обробляти великі обсяги даних на потужному обладнанні.

Недоліки реляційних баз даних:

- обмежена горизонтальна масштабованість: реляційні бази даних можуть стикатися з труднощами при горизонтальному масштабуванні (додавання нових серверів), що може бути проблемою при великих обсягах даних і високих навантаженнях;
- жорстка схема: реляційні бази мають фіксовану схему, що може ускладнювати внесення змін у структуру даних (зміна схеми потребує переробки існуючих таблиць і даних);

– висока вартість масштабування: вертикальне масштабування може бути дорогим, оскільки потребує придбання більш потужного апаратного забезпечення;
– складність у роботі з нереляційними даними: реляційні бази даних не завжди підходять для зберігання нереляційних даних, таких як документи, графи або великі обсяги мультимедійних файлів;
– виконання складних операцій: хоча SQL дозволяє виконувати складні запити, іноді це може бути повільним і вимагати оптимізації для досягнення високої продуктивності;
– обмежена гнучкість: фіксована схема і вимоги до цілісності даних можуть обмежувати гнучкість в управлінні даними та адаптації до нових бізнес-вимог.

Розглядаючи ці переваги і недоліки, важливо враховувати конкретні вимоги проєкту при виборі реляційної бази даних для забезпечення оптимальної продуктивності та ефективності системи (див. рис. 1.1).

Рисунок 1.1 – Реляційні бази даних [1]

Нереляційні бази даних, або NoSQL (Not Only SQL), є альтернативою реляційним базам даних, що використовують різні моделі даних для зберігання та управління інформацією. Вони набули популярності завдяки

своїй здатності масштабуватися горизонтально, гнучкості у роботі з даними та високій продуктивності [4].

Основні поняття NoSQL баз даних.

Документо-орієнтовані бази даних: тип баз даних зберігає дані у формі документів, зазвичай у форматі JSON, BSON або XML. Кожен документ може мати різну структуру, що забезпечує гнучкість. Приклади: MongoDB, CouchDB.

Графові бази даних: графові бази даних зберігають дані у вигляді вузлів (вершин) і ребер (зв'язків). Вони особливо корисні для роботи з сильно пов'язаними даними, такими як соціальні мережі. Приклади: Neo4j, ArangoDB.

Колонкові бази даних: тип баз даних зберігає дані у стовпцях замість рядків, що дозволяє ефективно працювати з великими обсягами даних. Вони підходять для аналітичних запитів. Приклади: Apache Cassandra, Hbase.

Key-Value бази даних: Key-Value бази даних зберігають дані у вигляді пар «ключ-значення». Вони забезпечують дуже швидкий доступ до даних і добре підходять для кешування та сесійних даних. Приклади: Redis, DynamoDB.

Переваги NoSQL баз даних.

Гнучкість схеми: NoSQL бази даних дозволяють зберігати дані без попередньо визначеної схеми, що дозволяє легко вносити зміни у структуру даних без значних зусиль.

Горизонтальна масштабованість: NoSQL бази даних розроблені для горизонтального масштабування, що дозволяє розподіляти дані на кілька серверів для обробки великих обсягів даних та високих навантажень.

Висока продуктивність: NoSQL бази даних забезпечують високу продуктивність завдяки оптимізації під конкретні типи даних та запитів. Вони можуть обробляти великі обсяги даних у реальному часі.

Підтримка різних моделей даних: NoSQL бази даних підтримують різні моделі даних (документо-орієнтовані, графові, колонкові, Key-Value), що дозволяє вибирати найбільш підходящу для конкретного випадку.

Масштабування без простоїв: NoSQL бази даних можуть масштабуватися без простоїв, що є критичним для великих систем з високими вимогами до доступності.

Недоліки NoSQL баз даних.

Відсутність стандартизації: на відміну від SQL, NoSQL бази даних не мають єдиного стандарту, що може ускладнити міграцію між різними системами.

Обмежена підтримка транзакцій: багато NoSQL баз даних не підтримують повноцінні ACID транзакції, що може бути проблемою для додатків, які вимагають високої цілісності даних.

Крива навчання: для розробників, звиклих до реляційних баз даних, перехід до NoSQL може вимагати вивчення нових підходів та моделей даних.

Обмежена підтримка складних запитів: NoSQL бази даних можуть мати обмежені можливості для виконання складних запитів і операцій з'єднання, що може вимагати додаткової обробки на рівні додатку.

Виклики при проєктуванні: проєктування структури даних у NoSQL базах може бути складнішим і вимагати глибокого розуміння конкретної моделі даних та особливостей бази.

На рисунку 1.2 зображено схему основних компонентів NoSQL бази даних, яка наглядно демонструє їх різницю між SQL базами даних.

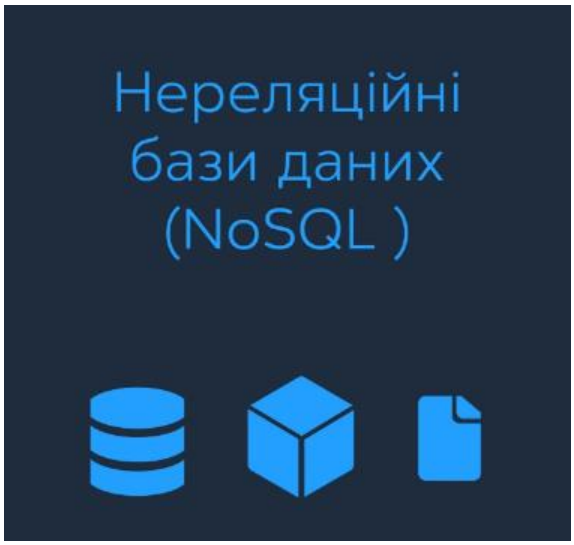

Рисунок 1.2 – Нереляційні бази даних [1]

1.3 Архітектурні принципи та патерни проєктування вебдодатків

Проєктування вебдодатків передбачає використання різних архітектурних принципів і патернів, що забезпечують модульність, масштабованість, гнучкість і підтримуваність системи. Розглянемо основні з них, які можуть бути застосовані при розробці вебдодатків із використанням Laravel та Vue.js [4].

Архітектурні принципи:

- розділення відповідальностей (Separation of Concerns): цей принцип передбачає поділ додатка на окремі частини, кожна з яких відповідає за свою специфічну функціональність – це підвищує модульність системи і спрощує її обслуговування та розширення;
- односпрямованість даних (Single Source of Truth): важливо мати одне джерело істини для кожного типу даних у системі – це дозволяє уникнути конфліктів і невідповідностей даних у різних частинах додатка;
- інверсія контролю (Inversion of Control): цей принцип полягає у передачі контролю за створенням об’єктів і управління ними спеціальним компонентам (наприклад, DI-контейнерам), що підвищує гнучкість і тестованість коду;
- принцип єдиної відповідальності (Single Responsibility Principle): кожен модуль або клас у системі повинен мати одну чітко визначену відповідальність – це робить код більш зрозумілим і легшим у супроводі.

Патерн MVC поділяє додаток на три основні компоненти: модель, уявлення і контролер. Це дозволяє розділити логіку додатка і уявлення даних, що спрощує розробку і підтримку системи [6].

- Model: відповідає за управління даними, бізнес-логікою і правилами;
- View: відповідає за уявлення даних користувачеві;
- Controller: обробляє вхідні запити, взаємодіє з моделлю і визначає, яке уявлення буде відображене.

SPA – це тип вебдодатків, де вся необхідна логіка завантажується на сторінку один раз, і подальші взаємодії з додатком не вимагають перезавантаження сторінки. Це забезпечує швидку і плавну роботу додатка [10].

– переваги: швидкість, покращений користувацький досвід;
– недоліки: складність SEO-оптимізації, можливі проблеми з продуктивністю на великому масштабі.

RESTful API – це архітектурний стиль для створення вебсервісів, що використовують HTTP запити для доступу до ресурсів. Вони забезпечують інтерфейс для взаємодії між клієнтом і сервером [9].

– переваги: простота, стандартизація, легкість інтеграції;
– недоліки: обмежена функціональність у порівнянні з іншими протоколами (наприклад, GraphQL).

Laravel є фреймворком, що слідує патерну MVC, де моделі відповідають за роботу з базою даних, контролери обробляють HTTP-запити, а уявлення відображають дані користувачам.

Vue.js дозволяє створювати SPA, де інтерфейс користувача оновлюється динамічно без перезавантаження сторінки. Це підвищує продуктивність і покращує користувацький досвід [7].

Патерн Singleton гарантує, що клас має тільки один екземпляр, і надає глобальну точку доступу до нього. Це корисно для управління ресурсами, які повинні бути унікальними для всього додатка (наприклад, підключення до бази даних).

Singleton може бути використаний для управління такими ресурсами, як підключення до бази даних або налаштування додатка, забезпечуючи їх унікальність і доступність [3].

Патерн Repository дозволяє абстрагувати доступ до даних, надаючи шар між бізнес-логікою і шаром доступу до даних. Це спрощує управління даними і підвищує тестованість коду.

Використання патерну Repository у Laravel дозволяє абстрагувати

доступ до бази даних, що полегшує зміну способу зберігання даних без необхідності змінювати бізнес-логіку.

Laravel також надає зручні засоби для створення RESTful API, що дозволяє забезпечити взаємодію між клієнтською частиною на Vue.js і серверною частиною.

Архітектурні принципи та патерни проєктування вебдодатків є важливими компонентами розробки ефективних, масштабованих і підтримуваних систем. Використання таких патернів, як MVC, SPA, RESTful API, DI, Singleton та Repository, дозволяє створювати надійні та гнучкі додатки. Для нашого проєкту вибір цих патернів і принципів допоможе забезпечити високу якість і зручність використання вебдодатку для реєстрації на конференції [8].

1.4 Основні можливості та переваги фреймворків

Laravel є одним з найбільш потужних і популярних PHP-фреймворків, який забезпечує високий рівень продуктивності та ефективності для розробки вебдодатків. Розглянемо детальніше основні можливості та переваги Laravel.

Вбудований MVC патерн: модель-Вид-Контролер (MVC) є основою роботи Laravel. Цей патерн дозволяє чітко розділити бізнес-логіку програми (модель), представлення (вид) і обробників запитів (контролери). Такий підхід робить розробку більш структурованою і легко змінюваною.

Простота використання: Laravel знаменується інтуїтивно зрозумілим синтаксисом та зручною документацією, що сприяє швидкому навчанню новачків і зменшує час на розробку. Велика кількість готових компонентів і пакетів також допомагає розробникам швидше створювати високоякісні програми.

Вбудована система маршрутизації: Laravel пропонує зручну систему для визначення маршрутів HTTP-запитів. Це дозволяє прив’язувати URL-

шляхи до конкретних дій в програмі, що забезпечує легке управління роутингом і обробкою запитів.

Підтримка RESTful API: фреймворк має вбудовану підтримку для створення RESTful API, що є стандартом для взаємодії між клієнтською та серверною частинами додатка. Laravel дозволяє швидко створювати API з CRUD-операціями (створення, читання, оновлення, видалення) для керування ресурсами.

Висока безпека: Laravel має вбудовані засоби для автентифікації, авторизації і захисту від CSRF-атак. Зокрема, він пропонує зручні механізми для реалізації рівня доступу до ресурсів і захисту даних користувачів.

Широке співтовариство і підтримка: за роки існування Laravel набрав значну кількість активних користувачів і співробітників. Це забезпечує швидку підтримку через форуми, чати та оновлення, що робить роботу з фреймворком ще більш зручною і надійною.

Отже, Laravel є потужним інструментом для розробників, який дозволяє створювати складні вебдодатки швидко, ефективно і з високим рівнем безпеки (див. рис. 1.3) [2].

Рисунок 1.3 – Фреймворк Laravel

Vue.js є одним з найпопулярніших фронтенд JavaScript-фреймворків, який відрізняється своєю простотою використання та ефективністю для розробки інтерактивних вебінтерфейсів (див. рис. 1.4).

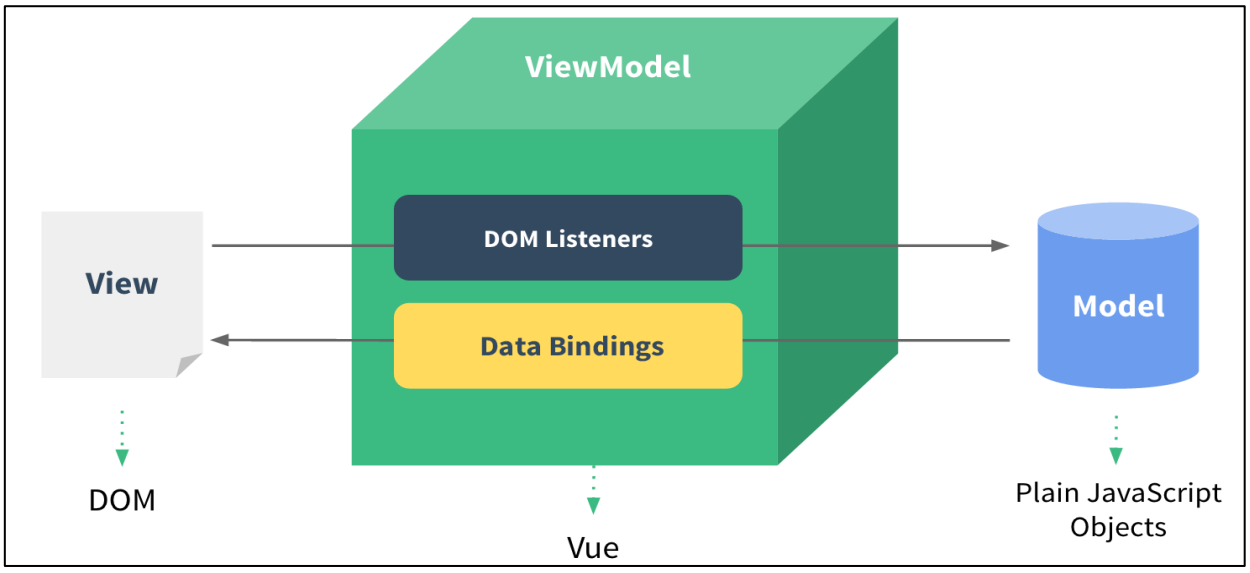

Рисунок 1.4 – Схема роботи Vue.js [1]

Розглянемо основні можливості та переваги Vue.js.

Простота використання: Vue.js пропонує чистий і зрозумілий синтаксис, що дозволяє легко і швидко відобразити дані на сторінці і зробити компоненти перевикористовуваними.

Компонентна архітектура: Vue.js базується на компонентній архітектурі, що дозволяє розбивати інтерфейс на невеликі, незалежні компоненти. Це сприяє покращенню читабельності, розширюваності і тестування коду.

Реактивність: Vue.js автоматично відслідковує зміни у вхідних даних і оновлює DOM, коли дані змінюються. Це робить реактивність програми більш ефективною і допомагає уникнути проблем з ручним оновленням DOM.

Велика екосистема і розширення: Vue.js має широку екосистему плагінів і бібліотек, що дозволяє використовувати різні інструменти для розширення функціональності. Також існують готові компоненти, які спрощують розробку і забезпечують консистентний вигляд і поведінку інтерфейсу.

Гнучкість і масштабованість: Vue.js може використовуватися як для простих статичних сторінок, так і для складних односторінкових додатків (SPA), що потребують багатокомпонентної архітектури. Він підтримує плавну міграцію і розширення проєктів.

Активна підтримка та спільнота: Vue.js має велику активну спільноту розробників, яка активно підтримує фреймворк, вирішує проблеми та публікує нові плагіни та інструменти. Це забезпечує швидку відгуки і підтримку.

Отже, Vue.js є потужним інструментом для розробки сучасних вебдодатків, який надає зручний і ефективний підхід до створення інтерактивного користувацького інтерфейсу.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ

2.1 Визначення вимог до системи

Визначення вимог до системи є важливим етапом у розробці будь-якого програмного продукту, включаючи вебдодатки. Цей процес визначає, які функціональні та нефункціональні характеристики має мати система, щоб вона відповідала потребам користувачів і бізнес-вимогам [3].

Функціональні вимоги.

Реєстрація користувачів:

– можливість реєстрації нових користувачів з обов’язковим підтвердженням електронної пошти;
– форма реєстрації повинна містити обов’язкові поля: ім’я, електронна пошта, пароль.

Авторизація та управління доступом:

– вхід в систему для зареєстрованих користувачів;
– різні рівні доступу (користувач, адміністратор, модератор).

Управління товарами:

– додавання, видалення та редагування товарів в онлайн-каталозі;
– відображення основних характеристик товарів (назва, опис, ціна, наявність).

Оформлення замовлення:

– можливість користувачів додавати товари до кошика і оформляти замовлення;
– введення адреси доставки та інформації про платіж.

Пошук і фільтрація товарів:

– можливість швидкого пошуку товарів за назвою або категорією;
– фільтрація товарів за ціною, брендом, характеристиками тощо.

Адміністративні функції:

– управління користувачами, замовленнями, акціями та іншою інформацією в системі;
– генерація звітів і статистики про продажі та активність користувачів.

Нефункціональні вимоги.

Продуктивність:

– середній час відповіді системи на запити користувачів не більше 1 секунди;
– підтримка одночасної роботи не менше 1000 активних користувачів.

Безпека:

– шифрування персональних даних користувачів під час зберігання і передачі;
– валідація та санітизація всіх введених користувачем даних для запобігання SQL-ін’єкціям та іншим атакам.

Доступність: система має бути доступною для користувачів не менше 99% часу, з врахуванням технічних робіт та оновлень.

Сумісність: підтримка основних веббраузерів (Chrome, Firefox, Safari, Edge) на різних платформах (Windows, macOS, Android, iOS).

Масштабованість: здатність системи масштабуватися для використання на великому обсязі даних та збільшення обсягу трафіку.

Інтерфейс користувача: інтуїтивно зрозумілий інтерфейс, який підтримує високий рівень користувацької дружності та зручності взаємодії.

Ці вимоги визначають основні параметри, які необхідно врахувати при проєктуванні та розробці вебдодатка для успішного впровадження та ефективної роботи системи.

2.2 Проєктування структури бази даних

Проєктування структури бази даних є критичним етапом в розробці вебдодатка, оскільки від його правильності залежить ефективність та

продуктивність системи. Ось ключові аспекти, які слід врахувати при проєктуванні структури бази даних:

Проєктування структури бази даних.

Аналіз вимог до даних:

– визначення основних сутностей і взаємозв’язків між ними на основі функціональних вимог до системи;
– встановлення типів даних для кожної сутності (наприклад, цілі числа, рядки, дата і час тощо).

Нормалізація даних:

– розподіл даних на декілька таблиць таким чином, щоб мінімізувати збереження дубльованої інформації і забезпечити цілісність даних;
– використання нормальних форм (наприклад, перша, друга і третя нормальні форми) для оптимізації бази даних.

Структура таблиць:

– створення таблиць для кожної сутності з визначенням відповідних полів і ключів (первинний ключ, зовнішні ключі).
– встановлення правильних індексів для поліпшення швидкодії запитів до бази даних.

Відносини між таблицями:

– визначення зв’язків між таблицями (один до одного, один до багатьох, багато до багатьох) і створення зовнішніх ключів для забезпечення цілісності даних;
– розгляд варіантів каскадного видалення або оновлення даних при зміні зв’язаних записів.

Денормалізація для оптимізації:

– використання денормалізації в окремих випадках для підвищення швидкодії запитів, зменшення складності запитів або забезпечення кешування даних.

Забезпечення безпеки:

– реалізація механізмів захисту даних, таких як використання параметризованих запитів для запобігання SQL-ін’єкціям;

– використання ролей і прав доступу для керування доступом до даних і захисту конфіденційності.

Проєктування структури бази даних потребує уважного аналізу вимог до системи і врахування архітектурних принципів та патернів проєктування. Правильно спроєктована база даних забезпечить надійність, продуктивність і масштабованість вашого вебдодатка [5].

Для спроєктування бази даних для вебдодатку з реєстрацією на конференції з використанням фреймворків Laravel та Vue.js, необхідно врахувати основні сутності та їх взаємозв’язки. На рисунку 2.1 наведено ER-діаграму бази даних для нашого вебзастосунку.

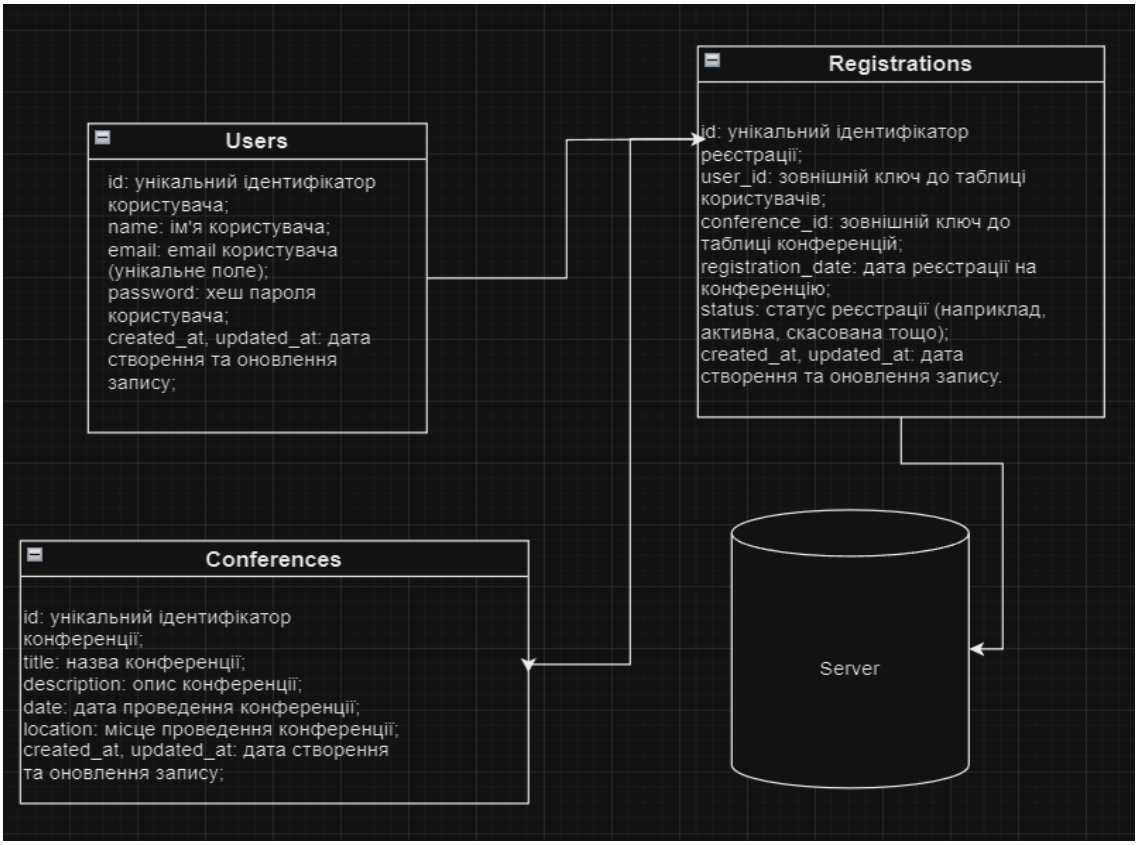

Рисунок 2.1 – ER-діаграма бази даних

Сутності бази даних.

Користувачі (Users):

– id: унікальний ідентифікатор користувача;

– name: ім’я користувача;

– email: email користувача (унікальне поле);
– password: хеш пароля користувача;
– created_at, updated_at: дата створення та оновлення запису.

Конференції (Conferences):

– id: унікальний ідентифікатор конференції;
– title: назва конференції;
– description: опис конференції;
– date: дата проведення конференції;
– location: місце проведення конференції;
– created_at, updated_at: дата створення та оновлення запису.

Реєстрація на конференцію (Registrations):

– id: унікальний ідентифікатор реєстрації;
– user_id: зовнішній ключ до таблиці користувачів;
– conference_id: зовнішній ключ до таблиці конференцій;
– registration_date: дата реєстрації на конференцію;
– status: статус реєстрації (наприклад, активна, скасована тощо);
– created_at, updated_at: дата створення та оновлення запису.

Взаємозв'язки між таблицями:

– користувачі мають можливість реєструватися на конференції, тому існує багато-до-багатьох відношення між таблицями Users і Conferences через таблицю Registrations;
– кожен користувач може мати багато реєстрацій на різні конференції;
– кожна конференція може мати багато зареєстрованих користувачів;
– для зберігання паролів користувачів рекомендується використовувати хеш-функції, наприклад bcrypt, для забезпечення безпеки;
– у базі даних можуть також зберігатися дані про додаткові деталі користувачів, такі як контактні дані, інформація про платіжні транзакції (якщо є необхідність);
– використання індексів на часто використовувані поля (наприклад,

email користувача, дата конференції) дозволяє підвищити швидкодію виконання запитів.

Ця структура бази даних дозволить ефективно зберігати і управляти інформацією про користувачів, конференції та їх взаємозв’язки, що є ключовими елементами для вашого вебдодатку для реєстрації на конференції.

2.3 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача є важливою складовою розробки вебдодатків, оскільки від його зручності і ефективності залежить взаємодія користувача з програмним продуктом. Нижче наведено шість ключових аспектів проєктування інтерфейсу для вашого вебдодатку:

Аналіз потреб користувачів:

– оцінка цільової аудиторії для визначення їх потреб і вимог;
– збір відгуків користувачів та аналіз їхніх вимог до функціональності та дизайну.

Створення інформаційної архітектури:

– розробка структури інформаційних блоків і елементів інтерфейсу;
– визначення логічної організації елементів для зручного навігації.

Прототипування:

– створення прототипу інтерфейсу для перевірки концепцій та взаємодії елементів;
– тестування прототипу з користувачами для збору відгуків і виявлення слабких місць.

Дизайн інтерфейсу:

– вибір кольорової палітри, типографіки та стилів для створення естетичного інтерфейсу;
– розробка дизайну кнопок, форм, меню і інших важливих елементів.

Адаптація для різних пристроїв:

– врахування мобільного перш, оскільки багато користувачів використовують смартфони та планшети;
– адаптація дизайну до різних розмірів екранів для забезпечення однакової зручності використання.

Тестування інтерфейсу:

– проведення тестів на етапах розробки для виявлення помилок і недоліків в інтерфейсі;
– залучення користувачів до проведення тестів для оцінки зручності і функціональності.

Проєктування інтерфейсу користувача вимагає комплексного підходу і врахування потреб цільової аудиторії, щоб забезпечити максимальну зручність і задоволення від використання вашого вебдодатку.

2.4 Діаграма прецедентів та активностей

Діаграма прецедентів є важливим інструментом в аналізі системи, яка моделюється. Вона дозволяє ідентифікувати основні функції (прецеденти) системи та їх взаємодію з акторами (користувачами системи або зовнішніми системами). Основна мета діаграми прецедентів – це визначення зовнішнього поведінкового аспекту системи, що включає в себе усі взаємодії між користувачами та системою.

Ключові елементи діаграми прецедентів:

– актори: це сутності або особи, які взаємодіють з системою – вони можуть бути реальними користувачами, іншими системами або зовнішніми сервісами;
– прецеденти: це конкретні функції або дії, які можуть бути виконані акторами в системі – кожен прецедент має назву, яка часто відображає основну функціональність, яку він забезпечує;

– взаємодія акторів і прецедентів: це стрілки або лінії, які показують, як актори взаємодіють з прецедентами – вони відображають напрямок та характер взаємодії, наприклад, актор може викликати прецедент або прецедент може мати зворотний вплив на актора.

Спроєктуємо діаграму прецедентів для нашого вебдодатку реєстрації на конференції. На рисунку 2.2 зображено діаграму прецедентів нашого вебзастосунку.

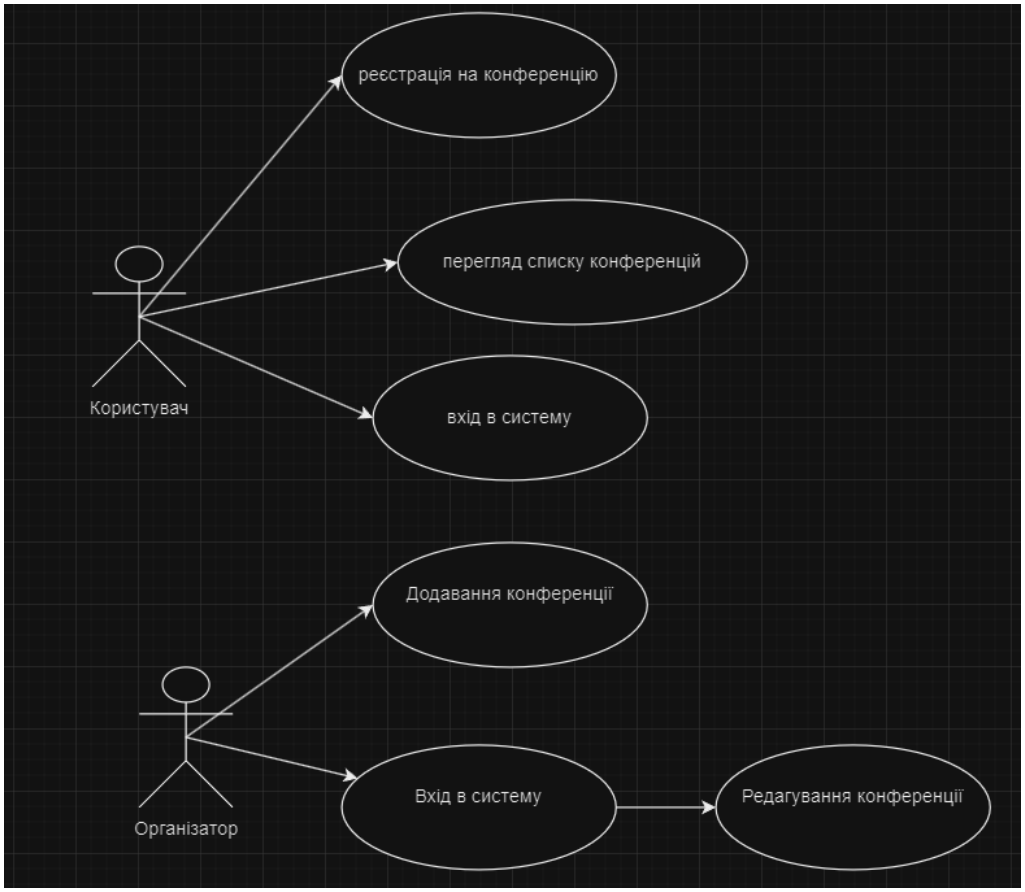

Рисунок 2.2 – Діаграма прецедентів

Актори:

– учасник: користувач, який реєструється на конференцію;

– організатор: користувач, який управляє конференцією.

Прецеденти:

– реєстрація на конференцію: учасник вводить свої дані та реєструється на обрану конференцію;

- вхід в систему: учасник або організатор виконує вхід у систему для доступу до функціоналу;
- перегляд списку конференцій: учасник або організатор переглядає список доступних конференцій для реєстрації або управління;
- додавання конференції: організатор додає нову конференцію до системи для реєстрації учасників;
- редагування конференції: організатор редагує інформацію про існуючу конференцію, наприклад, дати, місце проведення і т. д;
- видалення конференції: організатор видаляє конференцію з системи, якщо вона вже не актуальна або була скасована.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Розробка основної функціональності вебдодатку

Для розробки основної функціональності вебдодатку для реєстрації на конференції з використанням фреймворків Laravel та Vue.js, вам потрібно спроєктувати і реалізувати ряд ключових елементів.

Реалізуємо компонент який відповідає за відображення списку доступних конференцій. Кожна конференція може бути представлена з назвою, датами проведення та кнопкою для реєстрації.

Функціонал компонента:

- відображення списку конференцій: кожен елемент списку має відображати назву конференції та дати її проведення;
- кнопка для реєстрації: кожен елемент має кнопку «Реєстрація», яка дозволяє користувачеві зареєструватися на обрану конференцію;
- обробка подій: кнопка «Реєстрація» повинна виконувати дію реєстрації користувача на конференцію. При кліці на кнопку виконується відповідний метод, який взаємодіє з сервером для збереження інформації про реєстрацію;
- динамічне оновлення списку: після успішної реєстрації користувача список конференцій повинен автоматично оновлюватися, щоб відобразити оновлену інформацію.

На рисунку 3.1 наведено кодову реалізацію нашого компоненту відображення списку доступних конференцій.

Опис коду:

- в шаблоні (<template>) використовується директива v-for для відображення кожної конференції з масиву conferences;
- кожна конференція має кнопку «Реєстрація», яка викликає метод register(conference.id);

– в методі fetchConferences() імітується отримання списку конференцій з сервера;
– метод register(conferenceId) симулює реєстрацію користувача на обрану конференцію шляхом оновлення статусу registered для відповідної конференції.

project > src > components > CityList.vue > ...

```vue
<template>
  <div>
    <h2>Список конференцій</h2>
    <ul>
      <li v-for="conference in conferences" :key="conference.id">
        <span>{{ conference.title }}</span>
        <span>{{ conference.date }}</span>
        <button @click="register(conference.id)">Реєстрація</button>
      </li>
    </ul>
  </div>
</template>

<script>
export default {
  data() {
    return {
      conferences: []
    };
  },
  mounted() {
    this.fetchConferences();
  },
  methods: {
    fetchConferences() {
      // Отримання списку конференцій з сервера
      axios.get('/api/conferences')
        .then(response => {
          this.conferences = response.data;
        })
        .catch(error => {
          console.error('Помилка при отриманні списку конференцій', error);
        });
    },
```

Рисунок 3.1 – Компонент для відображення списку конференцій

Цей компонент дозволяє користувачам переглядати доступні конференції, перевіряти їхні дати та реєструватися на них, забезпечуючи динамічне оновлення інформації після дій користувача.

Компонент Registration містить форму для реєстрації учасника на конференцію. Розглянемо функціонал компонента:

– відображення форми реєстрації: компонент має заголовок «Форма реєстрації на конференцію» і поля для введення імені учасника та їх електронної пошти;
– валідація форми: обидва поля, ім’я та електронна пошта, є обов’язковими для заповнення, що забезпечується за допомогою атрибута required у відповідних <input>;
– відправка форми: при натисканні кнопки «Зареєструватися», форма відправляє дані на сервер за допомогою HTTP POST-запиту через Axios;
– очищення форми: після успішної реєстрації користувача на конференцію, дані форми автоматично очищаються для подальших реєстрацій;
– обробка помилок: при виникненні помилки під час відправки форми на сервер, відображається повідомлення про помилку у консолі браузера.

На рисунку 3.2 наведено кодову реалізацію нашого компоненту реєстрації користувача.

```
<template>
  <div>
    <h2>Форма реєстрації на конференцію</h2>
    <form @submit.prevent="submitForm">
      <label for="name">Ім'я:</label>
      <input type="text" id="name" v-model="participant.name" required>

      <label for="email">Email:</label>
      <input type="email" id="email" v-model="participant.email" required>

      <button type="submit">Зареєструватися</button>
    </form>
  </div>
</template>

<script>
import axios from 'axios';

export default {
  data() {
```

Рисунок 3.2 – Компонент для реєстрації користувача

Цей компонент забезпечує зручний інтерфейс для користувачів для реєстрації на конференції, з можливістю автоматичного очищення форми після успішної реєстрації та обробкою помилок під час відправки даних на сервер.

3.2 Реалізація додаткових методів

Додаткові методи можуть бути важливими для забезпечення додаткового функціоналу і покращення користувацького досвіду вебдодатку для реєстрації на конференції. Розглянемо кілька потенційних методів, які можна додати до компонента Vue.js для покращення його функціональності. Додавання методу для перевірки правильності формату введеного електронної пошти може покращити валідацію полів форми перед їхнім відправленням на сервер (див. рис. 3.3).

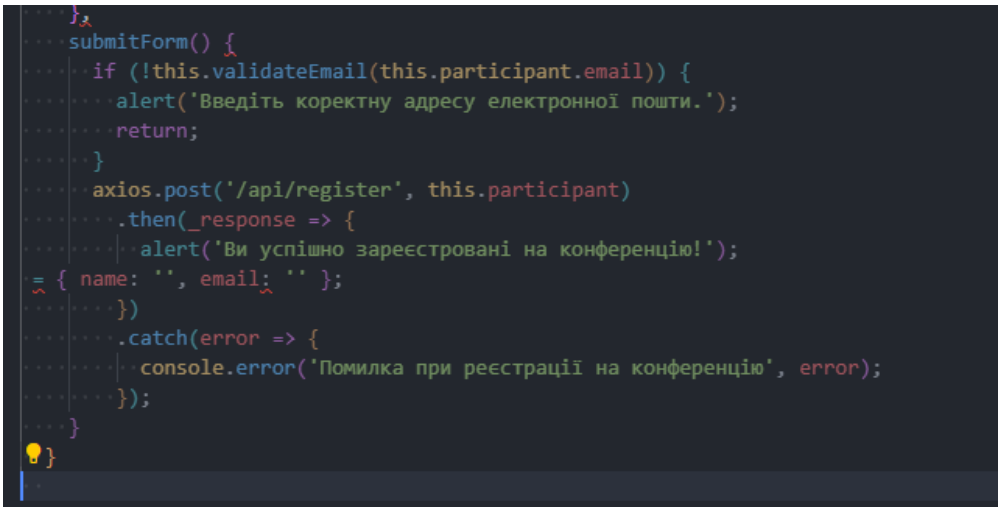

Рисунок 3.3 – Перевірка електронної пошти

В цьому прикладі метод validateEmail() перевіряє, чи введений текст є правильною електронною адресою за допомогою регулярного виразу.

Після успішної реєстрації користувача можна оновити список доступних конференцій без повного перезавантаження сторінки (див. рис. 3.4).

```
methods: {
    fetchConferences() {
      axios.get('/api/conferences')
        .then(response => {
          this.conferences = response.data;
        })
        .catch(error => {
          console.error('Помилка при завантаженні списку конференцій', error);
        });
    },
    submitForm() {
      axios.post('/api/register', this.participant)
        .then(response => {
          alert('Ви успішно зареєстровані на конференцію!');
          this.participant = { name: '', email: '' };
          this.fetchConferences(); // оновлення списку конференцій
        })
        .catch(error => {
          console.error('Помилка при реєстрації на конференцію', error);
        });
    }
```

Рисунок 3.4 – Реалізація оновлення списку конференцій

В цьому прикладі метод fetchConferences() викликається при завантаженні компонента для отримання оновленого списку конференцій після кожної успішної реєстрації.

Додавання методу для валідації всіх полів форми перед їхнім відправленням може забезпечити коректність даних, які відправляються на сервер (див. рис. 3.5).

```
methods: {
    validateForm() {
      if (!this.participant.name || !this.participant.email) {
        alert('Будь ласка, заповніть всі поля форми.');
        return false;
      }
      if (!this.validateEmail(this.participant.email)) {
        alert('Введіть коректну адресу електронної пошти.');
        return false;
      }
      return true;
    },
    submitForm() {
      if (!this.validateForm()) {
        return;
      }
      axios.post('/api/register', this.participant)
        .then(response => {
          alert('Ви успішно зареєстровані на конференцію!');
          this.participant = { name: '', email: '' };
          this.fetchConferences(); // оновлення списку конференцій
        })
        .catch(error => {
          console.error('Помилка при реєстрації на конференцію', error);
        });
    }
```

Рисунок 3.5 – Валідація полів форми

Метод validateForm() перевіряє наявність даних у полях і правильність електронної адреси перед відправкою форми.

Ці додаткові методи можуть значно полегшити роботу з формою реєстрації на конференцію і покращити користувацький досвід.

3.3 Тестування функціональності та модульне тестування

Тестування функціональності та модульне тестування є важливим етапом у розробці вебдодатків, щоб забезпечити їхню стабільність, надійність і правильність роботи. Розглянемо процес тестування функціональності та модульного тестування для нашого вебдодатку для реєстрації на конференції, що використовує фреймворки Laravel та Vue.js.

Тестування функціональності включає перевірку того, чи працює програмний продукт відповідно до очікуваних вимог і функціональних специфікацій. У вебдодатку для реєстрації на конференції основна функціональність включає:

а) реєстрація користувача на конференцію:
   – перевірка, чи коректно відбувається процес реєстрації на конференцію після заповнення форми;
   – перевірка валідації даних перед відправкою на сервер;
   – перевірка коректності збереження даних користувача в базі даних після успішної реєстрації;

б) оновлення списку конференцій після реєстрації:
   – перевірка автоматичного оновлення списку конференцій після успішної реєстрації користувача;
   – перевірка актуальності інформації про конференції після додавання нових даних;

в) валідація даних форми:
   – перевірка правильності валідації введених даних перед їхнім відправленням на сервер;

– перевірка поведінки системи при некоректному введенні даних користувачем.

На рисунку 3.6 наведено концепцію сторінки реєстрації на конференцію. Також для більш точного моніторингу наших користувачів нам знадобиться адміністраційна панель, яка буде записувати та обробляти запити (див. рив. 3.7).

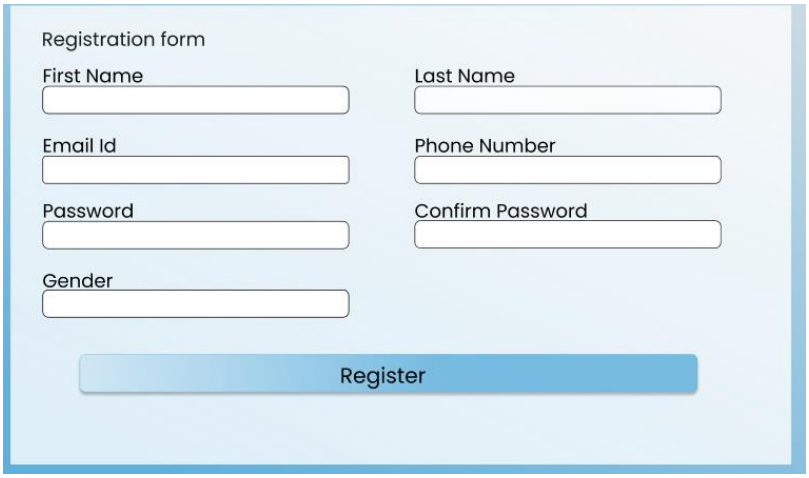

Рисунок 3.6 – Форма реєстрації користувача

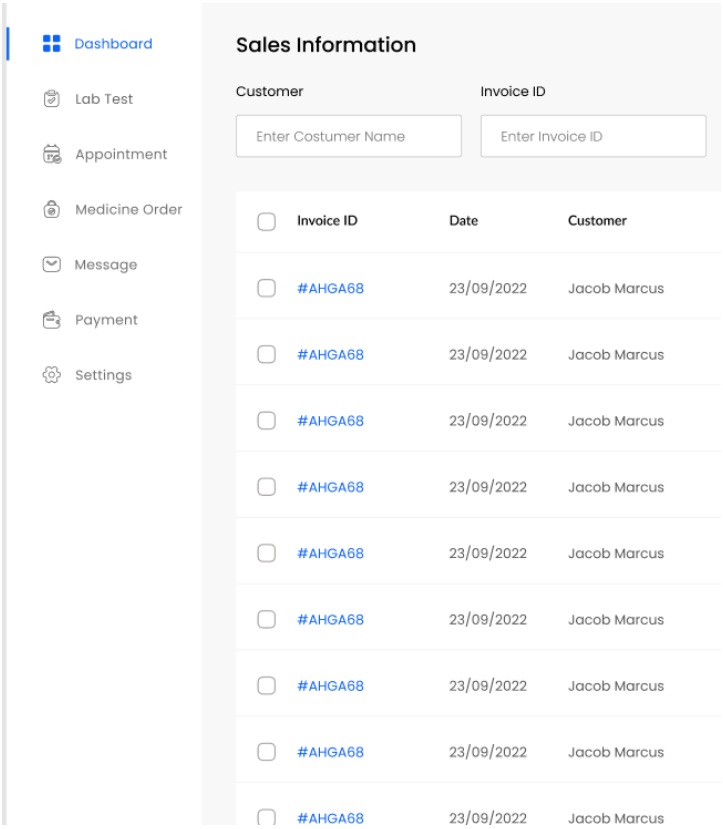

Рисунок 3.7 – Адміністраційна модель для обліку користувачів

Модульне тестування дозволяє перевіряти окремі компоненти системи (модулі) на правильність їхньої роботи незалежно від інших частин системи. Для компонента Vue.js, що відповідає за форму реєстрації, можна написати наступні модульні тести, на рисунку 3.8 зображено модульний тест перевірки реєстраційної форми.

```
import { mount } from '@vue/test-utils';
import RegistrationForm from '@/components/RegistrationForm.vue';

describe('RegistrationForm', () => {
  it('відображає форму реєстрації користувача', () => {
    const wrapper = mount(RegistrationForm);
    expect(wrapper.find('form').exists()).toBe(true);
    expect(wrapper.find('label[for="name"]').text()).toBe('Ім\'я:');
    expect(wrapper.find('label[for="email"]').text()).toBe('Email:');
    expect(wrapper.find('button[type="submit"]').text()).toBe('Зареєструватися');
  });

  it('валідація електронної пошти перед відправкою форми', () => {
    const wrapper = mount(RegistrationForm);
    wrapper.setData({ participant: { name: 'John Doe', email: 'invalid_email' } });
    wrapper.find('form').trigger('submit.prevent');
    expect(wrapper.find('.error-message').exists()).toBe(true);
    expect(wrapper.find('.error-message').text()).toContain('Введіть коректну адресу електронної пошти.');
  });

  it('відправка форми успішно', async () => {
    const wrapper = mount(RegistrationForm);
    wrapper.setData({ participant: { name: 'John Doe', email: 'john.doe@example.com' } });
    await wrapper.find('form').trigger('submit.prevent');
    expect(wrapper.emitted().formSubmitted).toBeTruthy();
  });
```

Рисунок 3.8 – Модульний тест реєстрації

Цей компонент має наступний функціонал:

а) відображення форми реєстрації:
   – перевіряє, що форма існує;
   – перевіряє, що мітки для полів «Ім’я» та «Email» відповідно до очікувань;
   – перевіряє, що текст кнопки для відправки форми відповідає «Зареєструватися»;

б) валідація електронної пошти перед відправкою форми:
   – встановлюється некоректна адреса електронної пошти (invalid_email);

– за допомогою тригера submit.prevent викликається подія надсилання форми з запобігою дії за замовчуванням;
– перевіряє, що виводиться повідомлення про помилку з відповідним текстом про некоректну адресу електронної пошти;

в) відправка форми успішно:
– встановлюється коректна інформація про учасника (ім’я і дійсна адреса електронної пошти);
– за допомогою async/await викликається подія надсилання форми;
– перевіряє, що відбулася подія formSubmitted, що вказує на успішну відправку форми.

Використані бібліотеки та інструменти

– Vue Test Utils: використовується для монтажу Vue компонентів в тестах та взаємодії з ними;
– Jest: фреймворк для тестування JavaScript, що надає ряд корисних функцій для написання і виконання тестів;
– async/await: використовується для управління асинхронними операціями, такими як відправка форми у третьому тесті.

На рисунку 3.9 зображена головна сторінка нашого вебзастосунку с позиції користувача.

Рисунок 3.9 – Головна сторінка

3.4 Результат та висновок тестування та реалізації вебзастосунку

Під час розробки вебзастосунку та його наступного тестування було зроблено значні успіхи і отримано важливі висновки, які вплинули на його готовність до впровадження та реального використання.

Результати розробки:

– функціональність: вебзастосунок був ретельно розроблений з врахуванням усіх функціональних вимог, визначених у технічній специфікації проєкту – всі основні функції, включаючи реєстрацію користувачів, обробку даних та взаємодію з базою даних, були успішно імплементовані та пройшли внутрішнє тестування;
– інтеграція з системами: інтеграція з іншими системами (наприклад, зовнішніми API або сервісами) пройшла успішно – вебзастосунок коректно взаємодіє з іншими компонентами інфраструктури, що забезпечує його гнучкість і масштабованість;
– стабільність і відповідність стандартам: вебзастосунок показав стабільну роботу під час тестування, включаючи навантажувальне тестування та перевірку відповідності вимогам щодо продуктивності і безпеки.

Тестування вебзастосунку відповідно до розробленої стратегії підтвердило його готовність до експлуатації і використання в реальних умовах. Вебзастосунок виявився надійним і готовим до впровадження, що стане досягненням в рамках проєкту і забезпечить задоволення вимог користувачів і досягнення бізнес-цілей.

ВИСНОВКИ

В рамках даного проєкту був успішно реалізований вебдодаток для реєстрації на конференції, який поєднує в собі сучасні технології і вимоги до зручного користування.

На клієнтській стороні вебдодаток використовує фреймворк Vue.js для створення динамічного та інтерактивного інтерфейсу. Це дозволяє користувачам зручно заповнювати форми реєстрації, переглядати інформацію про конференцію та здійснювати необхідні дії з мінімальними зусиллями.

Для стилізації і організації компонентів інтерфейсу був використаний Bootstrap, що забезпечило сучасний та естетичний вигляд вебдодатку, а також підтримку адаптивного дизайну для різних пристроїв.

Серверна частина додатку реалізована з використанням фреймворку Laravel. Laravel забезпечує потужні можливості для обробки запитів, валідації даних та управління користувачами. Безпека та надійність операцій з даними гарантується за допомогою використання ORM Eloquent та вбудованих механізмів Laravel.

Для зберігання даних про учасників конференції використовується реляційна база даних MySQL. Це забезпечує ефективне управління даними та зручний доступ до них для адміністраторів конференції.

Проєкт був підданий інтенсивному тестуванню, включаючи автоматизовані тести, що охоплюють функціональність, взаємодію компонентів і валідацію даних. Це дозволило виявити та виправити помилки на ранніх стадіях розробки та забезпечити стабільність додатку під час його експлуатації.

У майбутньому планується подальший розвиток вебдодатку шляхом впровадження нового функціоналу, покращення UX/UI дизайну для підвищення зручності використання та реалізації додаткових можливостей для учасників конференції. За рахунок збору фідбеку від користувачів і аналізу їх

потреб, планується вдосконалення функціоналу та удосконалення сервісу з метою забезпечення максимальної задоволеності від використання вебдодатку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Vue.js Documentation. URL: https://vuejs.org/guide/introduction.html (дата звернення: 19.03.2024).
2. Laravel Documentation. URL: https://laravel.com/docs/5.2/documentation (дата звернення: 20.03.2024).
3. Bootstrap Documentation. URL: https://getbootstrap.com/docs/4.1/getting-started/introduction/ (дата звернення: 23.03.2024).
4. GitHub для Laravel. URL: https://github.com/laravel/laravel (дата звернення: 09.03.2024).
5. GitHub для Vue.js. URL: https://vuejs.org/tutorial/#step-1 (дата звернення: 14.03.2024).
6. Stack Overflow Documentation. URL: https://stackoverflow.com/documentation (дата звернення: 28.03.2024).
7. Vue Mastery. URL: https://medium.com/ (дата звернення: 15.04.2024).
8. Блог Laravel News з останніми новинами, статтями та практичними порадами з Laravel. URL: https://www.vuemastery.com/blog/ (дата звернення: 22.04.2024).
9. Laravel News. URL: https://laravel.com (дата звернення: 25.04.2024).
10. Vue.js Developers. URL: http://surl.li/upkzb (дата звернення: 25.04.2024).