МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ВІДСТЕЖЕННЯ ДАНИХ ПОГОДИ З ВИКОРИСТАННЯМ VUE.JS ТА LARAVEL»**

Виконав: студент 4 курсу, групи 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Я.В. Лісаченко
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, PhD, Столярова А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(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

Лісаченку Ярославу Володимировичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебзастосунку для відстеження даних погоди з використанням Vue.js та Laravel

керівник роботи Столярова Анастасія Валеріївна, PhD
(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. Аналіз вимог та розробка вебзастосунку погоди.

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.	22.01.2024	
2.	Збір вихідних даних.	19.02.2024	
3.	Обробка методичних та теоретичних	18.03.2024	
	джерел.		
4.	Розробка першого та другого розділу.	22.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	17.06.2024	

Студент _____ Я.В. Лісаченко
(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 вебзастосунку для відстеження даних погоди із застосуванням фреймворків Vue.js та Laravel»: 60 с., 23 рис., 1 табл., 11 джерел.

БАЗА ДАНИХ, ВЕБЗАСТОСУНОК, КОНТРОЛЕР, КОМПОНЕНТ, МОДЕЛЬ, ПОГОДА, ФРЕЙМВОРК.

Об’єкт дослідження – методи розробки та інтеграції компонентів вебзастосунків для відстеження даних погоди.

Предмет дослідження – процес розробки вебзастосунку в контексті його етапів та виявлення потенційних труднощів, що можуть виникнути під час цього процесу.

Мета роботи – розробка вебзастосунку для відстеження даних погоди за допомогою фреймворків Vue та Laravel.

Метод дослідження – порівняння, аналіз, описовий, структурний.

Результати та їх новизна: досліджено етапи розробки вебзастосунку для відстеження даних погоди, виявлено ключові проблеми та застосовано існуючі методи для збору та аналізу вимог до програмного забезпечення.

Взаємозв’язок з іншими роботами: ця робота базується на дослідженні процесу розробки вебзастосунку та виявлених труднощів, що виникають під час цього процесу.

У кваліфікаційній роботі розглянуто вебзастосунки, які використовуються для відстеження даних погоди. Зростання попиту на точні та швидкі прогнози погоди робить важливим розв’язок задач ефективного створення, розгортання та підтримки таких вебзастосунків. Тому розробка вебзастосунку для відстеження даних погоди є актуальною задачею.

SUMMARY

Bachelor's qualifying paper "Development of a Web-application for Tracking Weather Data Using Vue.js and Laravel": 60 pages, 23 figures, 1 table, 11 references.

DATABASE, WEB APPLICATION, CONTROLLER, COMPONENT, MODEL, WEATHER, FRAMEWORK.

The object of study is the methods of developing and integrating components of web applications for weather data tracking.

The subject of study is the process of developing a web application in the context of its stages and identifying potential difficulties that may arise during this process.

The purpose of the work is to develop a web application for weather data tracking using the Vue and Laravel frameworks.

The research method includes comparison, analysis, descriptive, and structural approaches.

Results and their novelty: the stages of developing a web application for weather data tracking were studied, key problems were identified, and existing methods for collecting and analyzing software requirements were applied.

Relationship with other works: this work is based on the study of the web application development process and the difficulties identified during this process.

The qualification thesis examines web applications used for weather data tracking. The growing demand for accurate and fast weather forecasts makes the solution to the problems of effective creation, deployment, and maintenance of such web applications important. Therefore, the development of a web application for weather data tracking is a relevant task.

ЗМІСТ

ВСТУП

Сучасний світ характеризується стрімким розвитком інформаційних технологій, що безпосередньо впливає на наше повсякденне життя. В умовах змінного клімату і зростання потреби в точних даних про погодні умови, розробка вебзастосунків для відстеження погоди стає все більш актуальною. Зручність і доступність отримання погодної інформації в режимі реального часу значно покращують планування діяльності як на індивідуальному, так і на бізнес-рівні. Це обумовлює актуальність даного проєкту.

Мета роботи – розробка вебзастосунку для відстеження даних погоди за допомогою фреймворків Vue та Laravel.

Досягнення поставленої мети передбачає реалізацію наступних завдань:

– проаналізувати сучасні технології розробки вебзастосунків для відстеження погоди;
– дослідити особливості використання фреймворків Vue та Laravel для створення інтерактивних вебзастосунків;
– розробити архітектуру вебзастосунку для відстеження даних погоди;
– спроєктувати та реалізувати основні компоненти вебзастосунку;
– протестувати функціональність вебзастосунку та його продуктивність.

Об’єкт дослідження – методи розробки та інтеграції компонентів вебзастосунків для відстеження даних погоди.

Предмет дослідження – процес розробки вебзастосунку в контексті його етапів та виявлення потенційних труднощів, що можуть виникнути під час цього процесу.

Методи дослідження – порівняння, аналіз, описовий, структурний.

Для досягнення результату було обрано фреймворки Vue та Laravel. JavaScript та PHP були використані для реалізації клієнтської та серверної частин відповідно, що дозволило забезпечити ефективну взаємодію між ними.

Для зберігання даних було обрано реляційну базу даних MySQL, що дозволяє зберігати структуровану інформацію та забезпечує високу швидкість доступу до даних. Використання Laravel спростило процес управління базою даних, а Vue дозволив створити сучасний інтерфейс користувача.

Основні положення і результати кваліфікаційної роботи доповідались, обговорювалися та знайшли схвалення на П’ятнадцятій Всеукраїнській, Двадцять другій регіональній науковій конференції молодих дослідників «Актуальні проблеми математики та інформатики» (Запоріжжя, 25-26 квітня 2024 р.), за результатами якої опубліковано тези.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, та переліку посилань. Перший розділ присвячений аналізу сучасних технологій та інструментів для розробки вебзастосунків, зокрема для відстеження погоди. У другому розділі наведено детальний опис архітектури та компонентів розробленого вебзастосунку, проілюстровано процес розробки та інтеграції основних елементів. У третьому розділі описано реалізацію функціональності вебзастосунку, результати тестування та оцінку продуктивності.

1 АНАЛІЗ ВИМОГ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз вимог замовника

Клієнт виражає бажання створення вебдодатку з метою відстеження погодних умов, використовуючи технології Vue та Laravel. Основною метою проєкту є розробка функціонального інструменту, що задовольнить потреби користувачів у моніторингу погоди. У рамках проєкту передбачено створення наступних основних сторінок:

- **сторінка авторизації та реєстрації:** можливість входу та реєстрації користувачів для доступу до персонального облікового запису;
- **головна сторінка:** перегляд погодних умов, реалізація пошуку та фільтрації за різними параметрами, зміна представлення погодних даних;
- **сторінка деталей погоди**: опис погоди, характеристики погодних умов, історія зміни погодних параметрів, можливість коментування погоди користувачами;
- **профіль користувача**: можливість коригування особистих даних користувача.

Основні переваги обраного інструментарію, які були визначені замовником, включають:

- **Vue:** простота використання та гнучкість для швидкого розгортання, активна спільнота розробників;
- **Laravel:** надійність та ефективність у роботі з великими обсягами даних, широкий функціонал та зручний інтерфейс для розробки;
- **MySQL:** висока продуктивність та надійність у роботі з базою даних, широкі можливості для оптимізації та адміністрування;
- **Eloquent:** зручний інструмент для роботи з базою даних, підтримка

відносин між таблицями;

- **Vuex:** ефективне управління станом додатку, покращення продуктивності та швидкодії;
- **ESLint:** створення єдиної стилістики коду та виявлення потенційних помилок для підвищення якості коду.

Отже, після ретельного аналізу обраного інструментарію можна визначити, що продукт спрямований на покращення обробки запитів для електронного комерційного ресурсу, зокрема, у сфері мережевих операцій, таких як взаємодія з базою даних або обмін повідомленнями через сокети. Головний акцент робиться на швидкій та ефективній роботі з мережею, сприяючи оптимізації введення та виведення даних. Важливо зазначити, що обрана інструментальна платформа сприяє зручності для розробників, допомагаючи виявляти та виправляти помилки ще на етапі розробки, що позитивно впливає на якість та ефективність процесу створення продукту.

Замовник також встановлює конкретні вимоги до деяких компонентів системи:

- **валідація пароля при реєстрації:** забезпечення визначеної довжини, використання лише латинських символів, збереження пароля у базі даних в зашифрованому вигляді з виключенням збереження у сирому форматі;
- **аутентифікація:** використання JWT-токенів для ефективного та безпечного механізму аутентифікації;
- **Role-Based Access Control (RBAC):** надання різних рівнів доступу, таких як адміністратор та звичайний користувач;
- **головна сторінка:** підтримка рекомендацій при введенні даних у поле пошуку, можливість натискати на Breadcrumbs для переходу до відповідного розділу, можливість безкінечної прокрутки або розділення на сторінки [9].

1.2 Визначення функціональних та нефункціональних вимог

Всі вимоги до програмних продуктів можна розділити на дві групи: функціональні та нефункціональні вимоги (НФВ). Перші описують «що» потрібно зробити, а другі – «як» повинна працювати система. Нефункціональні вимоги визначають умови, за яких продукт повинен функціонувати, і якості, якими він має володіти (наприклад, продуктивність, надійність, масштабованість). Вони мають велике значення, хоча безпосередньо і не описують основні функції системи. Від них залежить користувацький досвід.

Функціональні вимоги, часто скорочені як FR, представляють основні функціональні можливості або особливості, якими повинна володіти система програмного забезпечення, щоб відповідати призначеній меті. Простіше кажучи, ці вимоги визначають, що повинна робити система. Вони описують взаємодію між програмним забезпеченням і його користувачами, а також поведінку програмного забезпечення за різних умов.

Функціональні вимоги, зазвичай, мають такі характеристики:

- специфіка: вони детальні та конкретні, залишаючи мало місця для двозначності, окреслюють точні функції, входи та виходи системи;
- можливість перевірки: функціональні вимоги можна перевірити та підтвердити, щоб переконатися, що програмне забезпечення працює належним чином;
- орієнтація на користувача: вони тісно узгоджені з потребами та очікуваннями користувача, гарантуючи, що програмне забезпечення виконує заплановану мету;
- змінність: функціональні вимоги можуть змінюватися протягом проєкту, оскільки відгуки користувачів і потреби бізнесу розвиваються.

Щоб краще зрозуміти функціональні вимоги, розглянемо кілька прикладів для нашого вебзастосунку:

– реєстрація та аутентифікація користувачів;
– головна сторінка;
– детальна сторінка погоди;
– персональний кабінет користувача;
– панель адміністратора.

Функціональні вимоги формують основу проєкту програмного забезпечення та керують командою розробників у створенні бажаних функцій.

Нефункціональні вимоги, часто скорочені як NFR, доповнюють функціональні вимоги, вказуючи, як програмна система повинна виконувати певні функції. Вони визначають якості, характеристики та обмеження системи, а не її специфічні особливості. По суті, нефункціональні вимоги встановлюють стандарти продуктивності, безпеки та зручності використання системи.

Нефункціональні вимоги мають такі характеристики:

– якісні: на відміну від функціональних вимог, які зазвичай є кількісними, нефункціональні вимоги зосереджені на якісних аспектах, таких як продуктивність, надійність і безпека;
– глобально: нефункціональні вимоги застосовуються до всієї системи та впливають на її загальну поведінку;
– стабільність: вони, як правило, більш стабільні протягом життєвого циклу проєкту, зміни відбуваються рідше порівняно з функціональними вимогами;
– вимірність: хоча нефункціональні вимоги може бути важко точно визначити кількісно, їх все одно можна виміряти та протестувати.

Щоб краще зрозуміти нефункціональні вимоги, давайте розглянемо кілька прикладів для нашого вебзастосунку:

– продуктивність: система має завантажувати вебсторінку менш ніж за 3 секунди навіть за 100 одночасних користувачів;
– безпека: система має відповідати галузевим стандартам безпеки та протоколам шифрування;

– масштабованість: програма повинна мати змогу обробляти 50% збільшення трафіку користувачів протягом шести місяців без погіршення продуктивності.

Нефункціональні вимоги гарантують, що програмне забезпечення працює ефективно та відповідає очікуванням користувачів щодо продуктивності, безпеки та інших критичних аспектів.

Функціональні вимоги, як випливає з назви, описують функції системи, яка буде розроблена. Це опис того, якою буде система і як вона функціонуватиме для задоволення потреб користувачів. Вони забезпечують чіткий опис того, як система має реагувати на конкретну команду, функції та те, що очікують користувачі.

Нефункціональні вимоги описуюсь обмеження системи, яка буде спроєктована. Ці вимоги не впливають на функціональність програми. Крім того, існує поширена практика підкласифікації нефункціональних вимог на різні категорії, наприклад:

– інтерфейс користувача;
– надійність;
– безпека;
– продуктивність;
– технічне обслуговування;
– стандарти.

Єдина відмінність між ними полягає в тому, що система не може функціонувати без задоволення всіх функціональних вимог. З іншого боку, система дасть вам бажаний результат, навіть якщо вона не задовольняє нефункціональним вимогам.

Синергія між функціональними та нефункціональними вимогами гарантує, що кінцевий продукт відповідає очікуванням користувачів, одночасно відповідаючи продуктивності, безпеці та іншим критичним критеріям.

Збираючи вимоги до проєкту, важливо враховувати обидва типи, щоб

створити вичерпний список, який слугуватиме основою для розробки. Visure Requirements ALM Platform – чудовий інструмент для керування та відстеження як функціональних, так і нефункціональних вимог протягом життєвого циклу розробки програмного забезпечення [8].

1.3 Фреймвокр Vue.js та його можливості

Vue.js – це прогресивний фреймворк для побудови користувацьких інтерфейсів. Розроблений колишнім співробітником Google Еваном Ю, Vue було презентовано 2014 року і відтоді він набув великої популярності завдяки своїй легкості, гнучкості та інтуїтивно зрозумілій структурі [1].

Vue.js, будучи прогресивним фреймворком, використовує архітектурний паттерн Model-View-ViewModel (MVVM), що сприяє розділенню інтерфейсу користувача та бізнес-логіки. Модель у MVVM являє собою об’єкти даних, які взаємодіють із сервером і бізнес-логікою. Вид – це DOM (Document Object Model), який відображає інформацію користувачеві. ViewModel, будучи сполучною ланкою, містить стани і методи, а також відповідає за реактивне зв’язування даних між моделлю і видом, завдяки чому при зміні даних у моделі автоматично оновлюється вид і навпаки.

Ця будова дає змогу розробникам Vue.js сфокусуватися на бізнес-логіці, тоді як фреймворк автоматично обробляє оновлення DOM. Система реактивності Vue, заснована на getter і setter для властивостей об’єктів, дає змогу відстежувати й керувати змінами даних, полегшуючи створення інтерактивних інтерфейсів.

Vue.js підходить для різноманітних вебпроєктів, включно зі створенням динамічних односторінкових застосунків (SPA), які забезпечують плавну і швидку взаємодію з користувачем, подібну до нативних застосунків. Фреймворк має гнучкість, яка дає йому змогу бути інтегрованим у різні сценарії використання, починаючи з невеликих проєктів і закінчуючи

масштабними підприємствами. Vue.js легко працює поряд з іншими бібліотеками і може бути вбудований в наявні сторінки, що дає змогу поступово застосовувати Vue у вже наявних проєктах. Така універсальність робить Vue.js популярним вибором серед розробників, які хочуть швидко й ефективно створювати інтерфейси користувацької взаємодії. Він підходить як для маленьких особистих проєктів, так і для великих комерційних додатків, завдяки своїй масштабованості, продуктивності та зручності підтримки.

Vue пропонує потужні директиви, як-от v-if, v-else-if, v-else, які дають змогу розробникам керувати умовним рендерингом блоків коду, роблячи інтерфейси користувацької взаємодії гнучкими й адаптивними. Ці директиви оцінюють логічні умови і залежно від їхньої істинності включають або виключають блоки з фінальної розмітки, що допомагає уникнути непотрібного відтворення і зберегти ресурси клієнтської сторони.Для роботи з масивами даних Vue використовує директиву v-for, яка дає змогу ітерувати списками і відображати інформацію у вигляді повторюваних елементів. Це спрощує створення динамічних списків, таблиць та інших структур, де кожен елемент списку може бути пов’язаний з унікальними даними, тим самим покращуючи взаємодію користувача з застосунком, оскільки забезпечує більш організоване і доступне представлення даних. Використання директиви v-for також сприяє спрощенню управління станом, оскільки зміни у вихідному масиві даних автоматично відображаються в інтерфейсі користувача без додаткового втручання розробника.

Vue.js значно полегшує роботу з формами завдяки директиві v-model. Ця директива створює двосторонню прив’язку даних між елементами форми та даними застосунку, що спрощує збір, валідацію та обробку користувацького введення. З v-model стан форми завжди синхронізований із даними застосунку, що полегшує створення складних форм без необхідності вручну оновлювати елементи DOM під час зміни даних.

Система компонентів у Vue.js дає змогу розробникам будувати масштабовані додатки, розбиваючи інтерфейс на дрібні, незалежні блоки з

власною логікою і шаблоном. Це сприяє кращій організації коду, його перевикористанню та тестуванню. Кожен компонент містить свої дані, методи та життєві цикли, що робить додаток структурованим і зручним для підтримки [5].

Розробники можуть також використовувати слоти для розширення компонентів і пропси для передавання даних між батьківськими та дочірніми компонентами, що додає гнучкості в управлінні компонентами та обмін даними всередині програми.

Наведемо основні особливості Vue.js.

Однією з помітних особливостей Vue.js є його реактивна система оновлення DOM. Фреймворк автоматично відстежує залежності між компонентами застосунку та їхніми даними, забезпечуючи в такий спосіб безшовне оновлення інтерфейсу в разі зміни стану. Ця система реактивності заснована на використанні геттерів і сеттерів для властивостей об’єктів, що дає змогу розробникам писати звичайний JavaScript-код, не піклуючись про ручне оновлення DOM. Такий підхід знижує ймовірність помилок і спрощує процес створення динамічних, чуйних інтерфейсів.

Vue.js набув популярності зокрема завдяки своїй доступності та легкості в освоєнні. Новачкам у веброзробці особливо легко увійти в тему за допомогою Vue, тому що документація фреймворка надзвичайно чітка, детальна і наповнена прикладами. Це знижує поріг входження і дає змогу розробникам на всіх рівнях швидко почати працювати з Vue, опановуючи функціональні можливості в міру зростання їхніх навичок і потреб проєкту.

Vue.js виділяється серед фреймворків своєю простотою інтеграції. Є можливість додати Vue у свій проєкт, просто під’єднавши скрипт, і почати використовувати його навіть на наявних сторінках без повного переписування коду. Така гнучкість робить його чудовим вибором для поступової міграції старих проєктів або для додавання інтерактивності на окремі сторінки. Це робить Vue особливо привабливим для розробників, які не хочуть або не можуть зануритися в повномасштабну фронтенд-розробку.

Vue описується як прогресивний фреймворк, що означає, що його можна використовувати як для невеликих завдань, так і для розробки великих застосунків. Він спроєктований так, щоб його можна було поступово застосовувати, і розробники можуть обирати, в яких частинах програми його використовувати. Це прогресивне впровадження робить його ідеальним для проєктів, які починаються з малого, але з часом розширюються та ускладнюються.

Vue.js прискорює процес розробки завдяки простоті використання та потужній інструментальній підтримці. Система компонентів полегшує масштабування застосунків і їхню підтримку, а інструменти, як-от Vue CLI, спрощують налаштування проєктів, гаряче перезавантаження й оптимізацію для продакшена. Такі функції, як декларативне відтворення і двостороння прив’язка даних, скорочують час, необхідний для написання шаблонного коду, що дає змогу розробникам зосередитися на створенні унікального користувацького досвіду.

Vue.js славиться своїм невеликим розміром. Мінімальна «вага» фреймворка означає, що він не накладає значного навантаження на час завантаження сторінок, що життєво важливо для користувацького досвіду, особливо на мобільних пристроях з обмеженим з’єднанням. Цей фактор робить Vue кращим вибором для додатків, де продуктивність є ключовим фактором. Можливість поступового завантаження тільки тих частин фреймворку, які дійсно потрібні, також допомагає в підтримці застосунків легковагими та швидкими.

1.4 Фреймвокр Laravel та його можливості

Laravel – це потужний PHP-фреймворк, який був створений канадським розробником Тейлором Отвеллом у 2011 році. Він був відповіддю на певні проблеми та недоліки PHP того часу, такі, як складність використання,

відсутність однорідності в структурі додатків та нестача зручних інструментів. Код може бути одночасно і витончено гарним, і потужним, що успішно доводить Laravel.

Сьогодні Laravel набув величезну популярність, і його в усьому світі цінують за синтаксис, продуктивні функції та велику екосистему. Безліч інструментів, доступних у Laravel, відкриває безмежні можливості для створення вебпроєктів різної складності, починаючи з невеликих сайтів і закінчуючи складними корпоративними рішеннями. Фінальний вигляд вебпродукту забезпечує стильний дизайн, але наскільки надійним буде результат і як саме все працюватиме в середині, визначає задумана архітектура.

Laravel має гнучку модульну архітектуру «модель-вид-контролер» (MVC). Архітектурна схема MVC – це спосіб організації коду, який дозволяє краще розуміти, як працює сайт і як він взаємодіє з базою даних та користувачами. Кожен з цих компонентів (модель, вид і контролер) виконує свою власну роботу: модель працює з даними, вид показує їх на екрані, а контролер керує тим, як вони взаємодіють один з одним.

На рисунку 1.1 зображено роботу архітектури MVC [11].

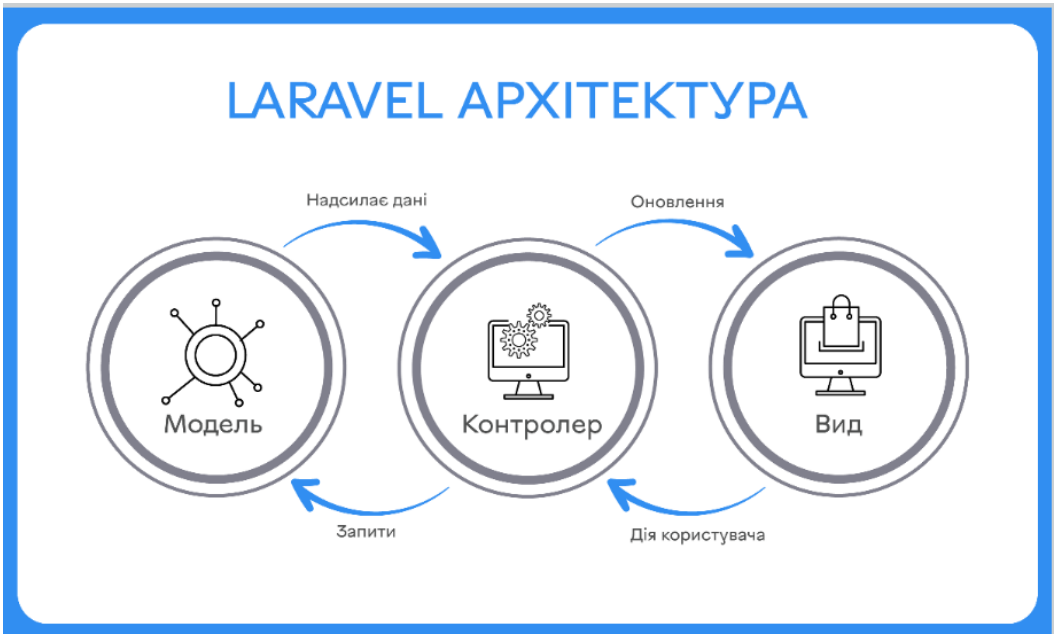

Рисунок 1.1 – Архітектурна схема MVC [11]

По суті, така архітектура MVC визначає, як зберігаються дані та як їх бачитиме користувач вашого вебпродукту: чи у вигляді відкритого онлайн-каталогу товарів, чи як закриті від сторонніх платні курси, чи як складну логістичну систему з різнорівневим авторизованим доступом. Така архітектура подобається розробникам своєю логікою та зручна для тестування.

Код Laravel побудовано на принципах простоти, елегантності та читабельності. Виразний та інтуїтивно зрозумілий синтаксис Laravel дозволяє зробити розробку швидкою та ефективною, що заощаджує час та ресурси.

Чистий код ідеально підходить для проєктів зі складною логікою, оскільки в ньому все настільки логічно і зрозуміло, що неможливо заплутатися і нові члени команди розробки та навіть сторонні підрядники легко приєднуються до роботи. Це розвіює нічні жахи власників бізнесу залишитися сам на сам зі своїм гарним сайтом, але заплутаним кодом, в якому ніхто не розбирається.

Фреймворк Laravel має вбудовані функції для автентифікації, безпеки та управління базами даних, що робить його ідеальним для розробки бізнес-платформ в Інтернеті. До того ж його код регулярно оновлюється, а, значить, постійно покращує всі свої компоненти й захищає їх від найновіших загроз.

Функціонал Laravel дає дозволяє створити такого роду вебпроєкти:

– інтернет-магазини з усією системою упорядкування каталогів, замовлень, доставлянь, платежів та аналітики;
– бізнес-сайти компаній для представлення компаній та їхніх послуг або товарів;
– електронні каталоги з можливістю швидкого пошуку та фільтрації різних товарів або послуг;
– B2B сайти для обслуговування бізнес-клієнтів (Business to Business);
– CRM системи для керування всіма операційними процесами бізнесу та ведення клієнтської бази;
– ERP системи автоматизації бізнес-процесів підприємства;

- WMS для управління складськими процесами та ведення обліку запасів;
- SCM, HRMS, TMS, RPA, PMS, DMS та інші системи управління для автоматизації та оптимізації різних аспектів управління бізнесом або проєктами;
- біткоїн-біржі для торгівлі криптовалютами та обміну криптовалютних активів;
- аналітичні платформи для збору, аналізу та візуалізації даних;
- маркетплейси для обміну та продажу товарів або послуг між користувачами;
- проєкти типу “booking” для замовлення та бронювання послуг, готелів, турів тощо;
- освітні онлайн-платформи для навчання та отримання знань через Інтернет;
- системи звітності для збору та обробки даних для подальшого аналізу та звітування;
- сайти закладів харчування з можливістю доставлення їжі за адресою;
- інші вебпродукти.

Існує чимало фреймворків, що дозволяють створити такі типи вебпроєктів, але Laravel вирізняється своєю особливою гнучкістю та багатими інструментами для розробки.

Наведемо візуальну схему застосування Laravel на рисунку 1.2 та поглянемо, як його можна застосовувати для проєктування вебзастосунків та інших додатків [11].

Laravel надає цілі пакети інструментів та готових компонентів для розробки різних видів вебдодатків. Окремо його цінують за сет вдалих рішень для електронної комерції, як «кошик», обробка платежів, управління замовленнями, система автентифікації та облік користувачів. Також з Laravel легко реалізувати багатомовність та багатовалютність, що дозволяє спростити керування різними версіями контенту і надає інструменти локалізації для взаємодії з користувачами з різних країн.

Рисунок 1.2 – Сфера застосування Laravel [11]

Однією з головних причин, чому Laravel так важливий для власників онлайн-бізнесу, є його орієнтованість на безпеку. Окрім регулярного оновлення коду, він використовує захищені кешовані паролі, які ніколи не будуть збережені у базі даних у вигляді тексту, вбудовані функції безпеки, такі як захист від підробки міжсайтових запитів, запобігання SQL-ін'єкціям та захист від міжсайтових скриптів. Все це допомагає розробникам налаштувати різні рівні безпеки через фільтрацію HTTP-запитів до вебдодатка.

Оскільки кібератаки стають все більш поширеними та хитромудрими, вкрай важливо, щоб вебпродукт був захищений від потенційних загроз і втрати даних.

Laravel відомий швидкістю та ефективністю створення вебпродуктів завдяки вбудованій системі кешування та підтримці асинхронних завдань. Це вкрай важливо для роботи з проєктами, які мають одночасно опрацьовувати великий об'єм даних і багато транзакції, як фінансові додатки чи електронна комерція. Оптимізована архітектура Laravel забезпечує високу продуктивність вебдодатків навіть при великих навантаженнях трафіку. Тому Laravel ідеально підходить для онлайн-бізнесу, де важлива безперебійна та чутлива взаємодія з користувачем.

Laravel – це фреймворк з відкритим вихідним кодом, який доступний для всіх. Завдяки цьому він має велику та активну спільноту розробників, які постійно створюють нові плагіни та розширення.

Laravel має гнучку систему подій, яка може бути корисно для співпраці з командою програмістів. А також реагувати на них, не втручаючись у конкретні деталі технічної реалізації. Для розробників такі патерни зручні для визначення власних, запланованих завдань в самому фреймворку.

Laravel надає вбудовану підтримку тестування з використанням бібліотеки PHPUnit. Це дозволяє розробникам писати тести для свого додатка, переконатися в його правильній роботі, і запобігти появі помилок у майбутньому. Таке тестування допомагає виявити та виправити помилки на ранніх етапах розробки та бути впевненим, що нові зміни чи код не порушують створену раніше функціональність.

Інтерфейс командного рядка Artisan надає доступ до функцій та інструментів Laravel, таких бази даних, модульне тестування та автентифікації та авторизації користувачів. Також цей інтерфейс бере на себе більшість тих повторюваних і громіздких завдань з кодом, які багато розробників уникають виконувати вручну [2].

Інструмент об’єктно-реляційного відображення (ORM) дозволяє працювати з базами даних у більш інтуїтивно зрозумілий та об’єктноорієнтований спосіб, полегшуючи керування даними та маніпулювання ними. Laravel також постачається з вбудованою системою міграції, через яку розробники легко оновлюють та змінюють схему бази даних без складних SQL-запитів, що економить час.

1.5 Аналіз інших існуючих рішень на ринку

Заощадити час при розробці додатків, дозволяє використання open source ПЗ. Оскільки вони пройшли через численні вдосконалення і були

використані у багатьох проєктах, вони, як правило, перевершують розроблені компоненти. Дуже важливо враховувати доступність попередньо створених елементів, тем та інших інструментів, які можуть спростити створення додатка.

Глобальне управління станом часто використовується у зовнішніх додатках для зберігання таких даних, як інформація про користувача, токени тощо. Redux – найпопулярніший проєкт управління глобальним станом на JavaScript. Більшість React-розробників використовують офіційний React-біндинг для Redux, підтримуваний командою проєкту.

Оскільки React дуже популярний, за допомогою простого пошуку в Google або на GitHub дуже легко знаходити готові компоненти та пакети практично на будь-який смак та потребу.

Екосистема React також включає React Native, що дозволяє створювати нативні додатки для iOS та Android, написані на React. Таким чином, React може стати чудовим вибором для створення мобільних додатків з використанням вебтехнологій.

React є частиною стека MERN, до якого входять MongoDB, ExpressJS, React і NodeJS. Перевагою цього стека є єдина мова програмування – JavaScript.

Для розробки мобільних додатків існує перспективний проєкт під назвою Weex, розроблений компанією Alibaba. Однак Weex далеко не такий зрілий і потужний, як React Native. Більше того, оскільки проєкт розробляється і використовується більше в Китаї, складніше знайти документацію і вирішення проблем англійською мовою.

Vue часто використовується з Laravel через їх хорошу інтеграцію. Laravel пропонує повну JavaScript і CSS підтримку, дозволяючи використовувати Vue у Laravel проєктах.

Проєкт JS Framework Benchmark є гарним способом подивитися порівняння продуктивності різних фреймворків, шляхом виконання базових операцій над таблицею з 1000 рандомізованих записів. Нижче в таблиці 1.1

наводяться результати порівняння React і Vue за часом у мілісекундах, витраченим на кожну з них [7].

Таблиця 1.1 – Порівняння продуктивності фреймворків

Час на.../Фреймворк	**react-v17.0.2**	**vue-v3.2.37**
створення 1000 рядків	48.9 ± 0.3	43.2 ± 0.3
заміна всіх 1000 рядків	48.2 ± 0.3	43.5 ± 0.2
частковий апдейт	122.3 ± 1.7	107.8 ± 0.6
вибір рядка	38.9 ± 1.2	19.8 ± 0.8
обмін положенням 2 рядків	158.6 ± 0.5	28.1 ± 0.6
видалення рядка	12.3 ± 1.3	12.3 ± 1.1
створення 10000 рядків	665.3 ± 2.0	471.4 ± 1.5
додавання 1000 рядків до існуючих 1000	111.7 ± 0.2	93.9 ± 0.5
видалення всіх рядків з 1000	33.2 ± 0.8	29.6 ± 0.9
середнє геометричне всіх факторів	1.52	1.40

Як видно з таблиці, React справляється значно гірше, ніж Vue зі свапом рядків, і, по суті, це єдина суттєва різниця в бенчмарках рендерингу таблиці – в більшості випадків вони не дадуть сильно помітних результатів. Єдине, можна сказати, що операція вибору рядків у таблиці є досить поширеною, що ставить React в ледве помітне програшне становище.

Також React і Vue демонструють високі показники з використання пам’яті та швидкості ініціалізації. Базовий скрипт запускається за 150-200 мілісекунд [7].

У відкритому доступі є дані з використання проєкту Perf Track від Google Chrome Labs щодо перевірки продуктивність вебсайтів, написаних на наших фреймворках. Проаналізуємо дані, представлені на рисунках 1.3 та 1.4 [8]:

– червоний: незадовільний час;
– жовтий: середній час;
– зелений: задовільний час виконання.

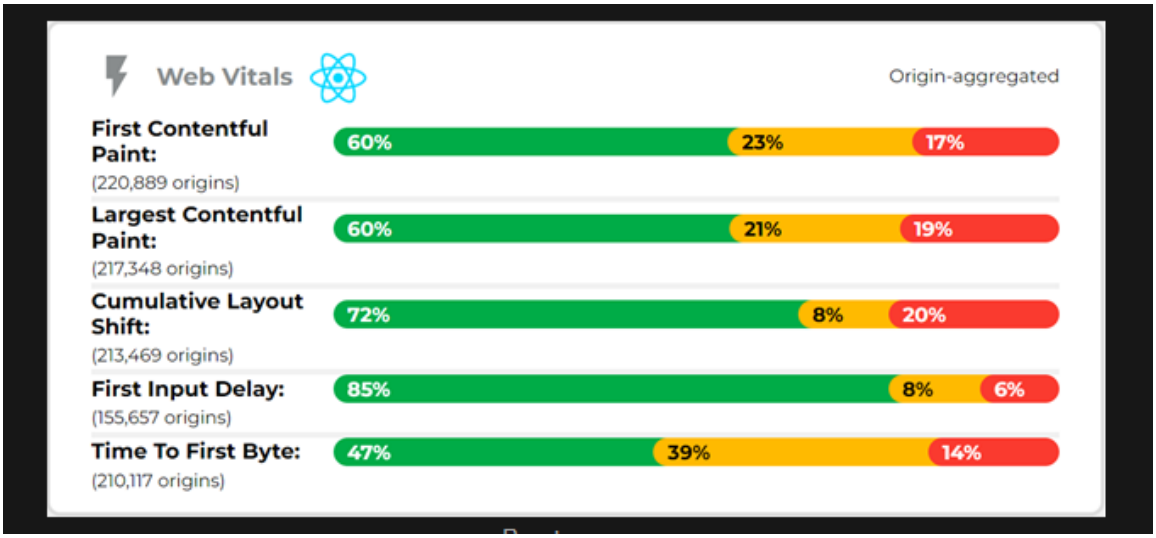

Рисунок 1.3 – Метрика React [8]

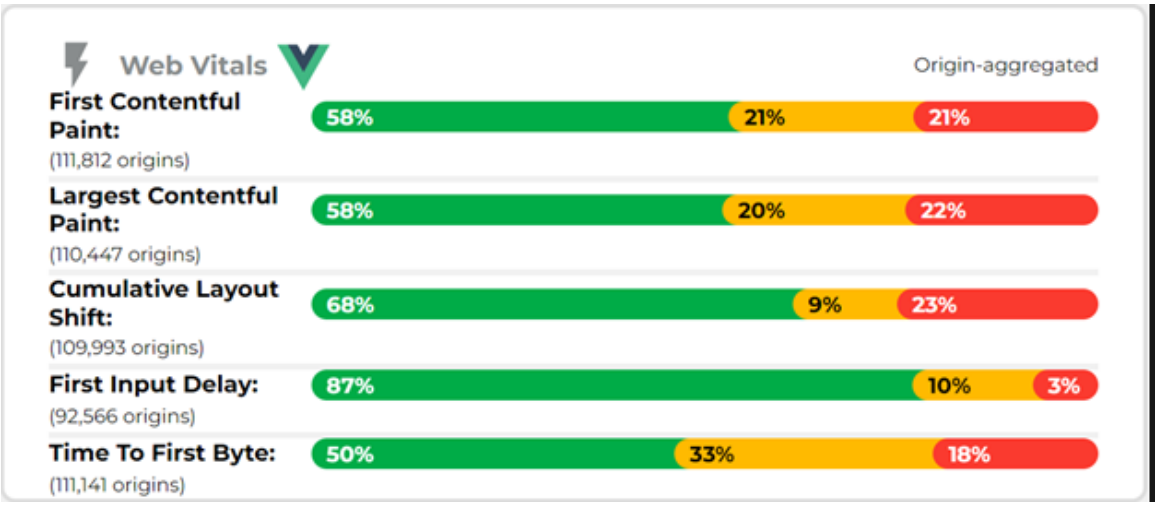

Рисунок 1.4 – Метрика Vue [8]

Варто зазначити, що наведені дані були зібрані станом на 1 листопада 2020 року для мобільних пристроїв і на них впливає не лише вибір фреймворка, а й безліч інших факторів. Для третього пункту, непередбачуваного зсуву лейауту сторінки під час завантаження елементів, вибір фреймворка взагалі мало впливає. Однак спробуємо зробити деякі висновки [8].

Перше відображення контенту та найбільше відображення контенту показує, що Vue і React краще справляються із завантаженням і рендерингом сторінки. Затримка першого введення (3 пункт) є тимчасовим проміжком між дією користувача (клік на кнопку, введення даних) і реакцією сторінки на неї, і обидва фреймворки показують винятково хороші результати за цим параметром. Також варто зазначити, що понад 70% додатків на Vue завантажують менше 1 MB JavaScript для своєї роботи, тоді як інші фреймворки зазвичай займають набагато більше пам’яті. Однак, потрібно розуміти, що це може означати, що більшість маленьких проєктів написані на Vue [8].

Основними двома техніками, що покращують роботу фронтенд додатків, є SSR (server-side rendering) і віртуалізація. Рендеринг на стороні сервера – це здатність додатка скомпілювати HTML-файли на сервері у повністю завантажену сторінку для користувача, а віртуалізація – це завантаження компонентів у міру їх потреби (наприклад, у міру скролінгу сторінки).

React має офіційний пакет ReactDOMServer для рендерингу на стороні сервера, а для віртуалізації багато хто використовує сторонні бібліотеки React-Virtualized і React-Window.

Vue також має офіційний SSR пакет Server-Renderer, проте з віртуалізацією потенційні розробники можуть зіткнутися з проблемами, оскільки навіть найпопулярніша бібліотека Vue Virtual Scroll List має певну кількість багів і не настільки стабільна, як її аналоги для інших фреймворків.

Важливою складовою при виборі фреймворка для девелоперів, які не мають попереднього досвіду, є концепції, які кожен фреймворк привносить, а також активність спільноти.

На перший погляд може здатися, що React – найпростіший у використанні фреймворк, просто імпортуємо бібліотеку, і можна писати JavaScript з використанням React API. Однак на самому початку ми вже розглянули, наскільки незручно виглядає простий “Hello world!” приклад,

написаний на чистому JavaScript, тому кожен початківець React девелопер повинен змиритися з фактом, що альтернативи навчанню JSX немає, оскільки його використання є аксіомою в спільноті для створення HTML-лейауту компонентам. Спочатку необхідність частково переходити на «мікс» з JavaScript і HTML може здатися дещо неінтуїтивною. З іншого боку, починаючи з версії 16.8 React вводить функціональні компоненти, що значно спрощує синтаксис і дозволяє швидко створювати компоненти у вигляді простих ES6 стрілкових функцій з типовим JSX лейаутом у зворотному значенні.

Також варто зазначити, що версія 17.0 додала можливість часткового оновлення додатка, що дозволяє зберегти функціонал, який спирається на застарілі концепти, і робить React хорошим вибором для довгострокової перспективи.

React є найчастіше завантажуваним фреймворком за статистикою npm, і це означає, що у користувачів не повинно виникати проблем з пошуком рішень на можливі проблеми при розробці, рівно як і активність спільноти дозволяє не тільки успішно шукати, але й задавати свої питання на популярних платформах типу StackOverflow.

Одним із концептів, з якими потрібно познайомитися початківцям Vue розробникам, є розширений HTML синтаксис з директивами. Більшість основних Vue директив інтуїтивно зрозумілі – v-if для рендерингу за умови, v-for для рендерингу в циклі, v-on для біндінгу функціоналу до івент лісенерів і так далі. Наявність лейауту, функціоналу і стилів в одному .vue файлі з інтуїтивним синтаксисом також робить розробку кожного компонента максимально простою і без необхідності перескакувати між файлами.

З релізом Vue 3 у 2020 році, творці фреймворка вирішили безліч проблем, які мала спільнота з використанням Vue для великих проєктів, ввівши можливості для більш широкого повторного використання функціоналу між компонентами. Також була покращена підтримка таких проєктів за допомогою рефакторингу вихідного коду Vue на TypeScript.

Оскільки Vue є наймолодшим фреймворком, розмір спільноти природно менший, ніж у React, і загальна популярність старої версії Vue 2 переважно серед китайських користувачів також створювала деякі проблеми в обміні знаннями та пошуку відповідей на питання. Однак за тією ж статистикою npm – Vue є лідером у прирості завантажень за останній рік. Значні покращення у Vue 3 дозволяють бути впевненими, що Vue скоро може обігнати за популярністю інші фреймворки.

Ми розглянули ряд основних аспектів, що відрізняють React і Vue. На завершення хочеться сказати, що обидва фреймворки знаходяться в активній розробці та регулярно отримують оновлення, тому в сучасних реаліях можна без зайвого занепокоєння використовувати будь-який із них. Неможливо передбачити довгострокову релевантність будь-якого з фреймворків, але важливо помітити, що Vue розвивається активніше інших, переймаючи корисні концепти у React.

При виборі фреймворка для розробки важливо визначити наявність доступних спеціалістів, а також часові ресурси, необхідні для навчання нових. Досвідченість команди може стати вирішальним фактором. І нарешті, сам проєкт, його складність, розмір і напрямок також можуть вплинути на ваше рішення.

Тепер ми можемо спокійно перейти до порівняння нашої серверної частини. Порівнювати ми будемо Laravel та WordPress.

Laravel, як PHP-фреймворк з виділеним сервером, забезпечує відмінну продуктивність завдяки своїй ефективній кодовій базі та функціям оптимізації. Це дозволяє розробникам писати чистий і організований код, що призводить до швидшого часу завантаження та підвищення загальної продуктивності. WordPress, хоча і відомий своєю простотою використання та гнучкістю, іноді може відставати в продуктивності, особливо для більших вебсайтів з обширними плагінами та темами. Проте, вебсайти WordPress все ще можуть досягати вражаючої швидкості за допомогою відповідних методів оптимізації.

Laravel надає різноманітні вбудовані функції та оптимізації для підвищення швидкості вебсайту, такі як кешування маршрутів, індексація бази даних та ефективна обробка запитів. Розробники також можуть використовувати такі інструменти, як Laravel Mix, для компіляції ресурсів та оптимізації. WordPress пропонує кілька плагінів та методів оптимізації для підвищення швидкості вебсайту, включаючи популярні плагіни кешування, такі як W3 Total Cache та WP Super Cache, плагіни для оптимізації зображень, такі як Smush, та відкладене завантаження зображень та скриптів. Крім того, використання легковажних тем та мінімізація використання непотрібних плагінів може ще більше підвищити продуктивність вебсайту WordPress.

Безпека має першочергове значення у веброзробці, і як Laravel, так і WordPress приділяють їй пріоритетну увагу. Давайте розглянемо запропоновані ними функції безпеки та найкращі практики захисту додатків.

Laravel може похвалитися надійним набором вбудованих функцій безпеки, включаючи захист від SQL-ін’єкцій, підробку міжсайтових запитів (CSRF) та атаки за допомогою міжсайтових скриптів (XSS). Він також надає такі функції, як захист маршрутів, шифрування та безпечні механізми аутентифікації «з коробки».

WordPress з захищеним від DDoS-атак VPS має надійні заходи безпеки. Крім того, основне програмне забезпечення WordPress регулярно отримує оновлення безпеки та патчі для усунення вразливостей. Крім того, WordPress пропонує функції безпеки, такі як ролі користувачів та дозволи, підтримка HTTPS та перевірка цілісності файлів.

Laravel надає розробникам високий рівень контролю та гнучкості над їх додатками завдяки своїй модульній архітектурі та обширним бібліотекам. За допомогою движка створення шаблонів Laravel Blade та інструменту командного рядка artisan розробники можуть легко налаштовувати кожний аспект своїх додатків, від внутрішньої логіки до дизайну інтерфейсу.

З іншого боку, WordPress пропонує користувачам зручну платформу з тисячами тем та плагінів для налаштування. WordPress відзначається тим, що

надає широкий спектр можливостей налаштування за допомогою тем та плагінів.

Laravel дозволяє розробникам впроваджувати найкращі практики SEO безпосередньо у свої додатки. Розробники можуть використовувати систему маршрутизації Laravel для створення чистих та оптимізованих для пошукових систем URL-адрес, оптимізуючи їх за ключовими словами. Крім того, движок створення шаблонів Laravel Blade дозволяє розробникам структурувати та маркувати контент таким чином, щоб його легко індексували пошукові системи.

WordPress, з іншого боку, відомий своїми можливостями SEO завдяки таким функціям, як налаштовані постійні посилання, автоматична генерація карт сайту та вбудована підтримка мета-тегів та описів. WordPress також пропонує багато SEO-плагінів, таких як Yoast SEO та All in One SEO Pack, які розширюють можливості SEO платформи, надаючи розширені функції, такі як аналіз контенту, генерація XML-карти сайту та розмітка схеми.

У кінцевому підсумку, обидва фреймворки мають свої переваги та недоліки, і вибір між Laravel та WordPress залежить від конкретних потреб проєкту, рівня експертизи розробника та інших факторів. Якщо вам потрібен великий контроль над додатком та гнучкість налаштування, Laravel може бути кращим вибором. З іншого боку, якщо ви шукаєте простоту в управлінні вебсайтом та доступ до широкого спектру тем та плагінів, то WordPress може бути більш підходящим варіантом.

Laravel надає розробникам інструменти і гнучкість для створення користувацьких рішень електронної комерції, адаптованих до конкретних вимог. Завдяки надійній архітектурі Laravel та обширним бібліотекам розробники можуть легко впроваджувати такі функції, як управління каталогом товарів, функціональність кошика покупок, безпечні платіжні шлюзи та обробка замовлень. Крім того, інтеграція Laravel з популярними платформами електронної комерції, такими як Stripe і PayPal, спрощує реалізацію функцій обробки платежів та оформлення замовлення.

WordPress, з його обширною екосистемою плагінів, пропонує декілька плагінів для електронної комерції, таких як WooCommerce, Easy Digital Downloads та Shopify, які надають готові рішення для створення вебсайтів електронної комерції. Ці плагіни пропонують такі функції, як управління продуктами, відстеження запасів, обробка замовлень та інтеграція платежів. Спрощуючи користувачам налаштування їх інтернет-магазинів та управління ними без великого досвіду розробки.

При порівнянні Laravel та WordPress обидві платформи мають явні переваги. Laravel відрізняється гнучкістю та кастомізацією для розробників, що робить його ідеальним для створення індивідуальних рішень. З іншого боку, WordPress вражає своєю зручною платформою CMS, особливо підходить для вебсайтів, спрямованих на контент, і магазинів електронної комерції [6].

2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

2.1 Проєктування архітектури вебзастосунку

Проєктування архітектури вебзастосунку є критично важливим етапом, що забезпечує належну структуру, масштабованість та підтримуваність системи. У випадку вебзастосунку для відстеження даних погоди з використанням Vue.js та Laravel, архітектура представлена наступними основними компонентами:

– клієнтська частина (Frontend);
– серверна частина (Backend);
– база даних;
– зовнішні сервіси;
– система аутентифікації та авторизації.

Клієнтська частина вебзастосунку для відстеження даних погоди будується на основі Vue.js, популярного фреймворку для створення інтерактивних користувацьких інтерфейсів. Основні задачі фронтенду включають рендеринг UI, обробку користувацьких подій, управління станом та взаємодію з сервером через API.

Основні компоненти фронтенду:

– головний компонент (App.vue): основний шаблон застосунку, який включає загальні елементи інтерфейсу, такі як шапка, підвал, навігація;
– компоненти сторінок: відповідають за різні частини застосунку, такі як головна сторінка, сторінка прогнозу, налаштування користувача;
– дочірні компоненти: використовуються для створення окремих елементів інтерфейсу, таких як форма пошуку, віджети погоди, графіки;
– роутінг: визначає маршрути для різних сторінок застосунку,

дозволяє створювати односторінкові додатки (SPA), де переходи між сторінками відбуваються без перезавантаження;

– маршрути: кожен маршрут відповідає за відображення конкретного компонента при переході на відповідну URL-адресу;
– управління станом: централізоване управління станом застосунку, що дозволяє зберігати глобальний стан, такий як дані користувача, налаштування, результати API-запитів;
– мутації та дії: мутації змінюють стан у відповідь на дії, які можуть бути викликані компонентами для асинхронних операцій (наприклад, отримання даних з сервера);
– HTTP-запити: бібліотека для надсилання HTTP-запитів до сервера, отримання даних погоди з API, відправки форм та інших даних.
– CSS/SCSS: використовуються для створення стильових правил для компонентів. SCSS додає функціональності, такі як змінні, вкладеність, міксіни;
– UI-бібліотеки: Vuetify або BootstrapVue для швидкого створення привабливого та функціонального дизайну.

Наведемо візуальну структуру нашого фронтенду на рисунку 2.1:

Основні файли та їх функції:

– main.js: головний файл, який ініціалізує Vue-застосунок, підключає роутер та Vuex;
– App.vue: головний шаблон застосунку, який містить основну розмітку;
– router/index.js: конфігурація маршрутизації, яка визначає, які компоненти відображаються на різних URL;
– store/index.js: конфігурація Vuex, яка підключає всі модулі управління станом;
– store/modules/weather.js: модуль Vuex для управління даними погоди, включаючи стан, мутації, дії та геттери.

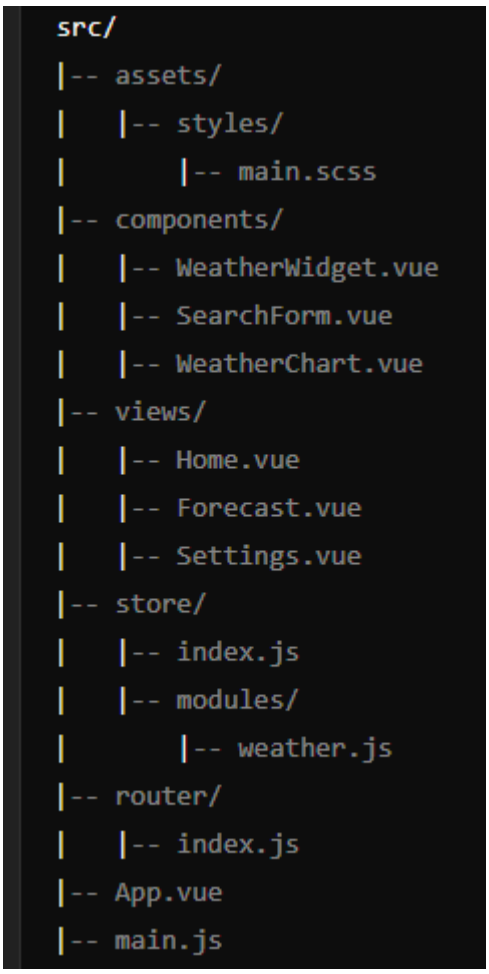

Рисунок 2.1 – Структура фронтенд-застосунку

Серверна частина вебзастосунку для відстеження даних погоди будується на основі Laravel, потужного PHP фреймворку, який надає зручні інструменти для розробки вебзастосунків. Laravel забезпечує просту і зрозумілу структуру, що дозволяє легко реалізувати обробку запитів, управління базою даних, аутентифікацію користувачів та взаємодію з зовнішніми API [7].

Основні компоненти бекенду:

– контролери;
– маршрути;
– моделі;
– міграції;
– сервіси;
– середовища.

У Laravel-застосунку контролери відповідають за обробку різних запитів та управління логікою додатку. Вони дозволяють організувати код інтуїтивно зрозумілим способом, розділяючи логіку між різними обробниками запитів.

У Laravel маршрути визначаються у файлі routes/web.php для вебзастосунків та routes/api.php для API-застосунків. Вони вказують, які HTTP-запити будуть відправлені до яких контролерів для обробки [2].

Наведемо візуальну структуру нашого бекенду на рисунку 2.2.

```
app/
|-- Http/
|   |-- Controllers/
|       |-- WeatherController.php
|       |-- UserController.php
|-- Models/
|   |-- Weather.php
|   |-- User.php
|-- Services/
|   |-- WeatherService.php
config/
database/
|-- migrations/
routes/
|-- web.php
|-- api.php
.env
```

Рисунок 2.2 – Структура бекенду

Для зберігання даних у вебзастосунку про погоду з використанням Laravel будемо використовувати базу даних MySQL. MySQL є однією з найпоширеніших реляційних баз даних у світі веброзробки, і вона має великий набір функцій, надійність та широку підтримку спільноти [3].

Візуалізуємо налаштування бази даних у Laravel на рисунку 2.3.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=your_database_name
DB_USERNAME=your_database_username
DB_PASSWORD=your_database_password
```

Рисунок 2.3 – Налаштування бази даних

Наведемо основні файли та методи бекенду, які відповідають за обробку запитів, взаємодію з базою даних та зовнішніми сервісами:

– WeatherController.php: контролер для обробки запитів, пов’язаних з погодою;
– UserController.php: контролер для обробки запитів, пов’язаних з користувачами;
– Weather.php: модель, що відображає структуру таблиці з даними погоди;
– User.php: модель, що відображає структуру таблиці з користувачами.
– WeatherService.php: сервіс для взаємодії з зовнішнім API для отримання даних погоди;
– web.php: файл маршрутів для вебзапитів;
– api.php: файл маршрутів для API-запитів;
– .env: файл конфігурації середовища.

Для отримання актуальних даних про погоду у вебзастосунку можна використовувати зовнішні сервіси, такі як OpenWeatherMap API. OpenWeatherMap надає широкі можливості для отримання інформації про погоду з усього світу за допомогою простого API.

Використання OpenWeatherMap API дозволить вебзастосунку отримувати оновлені дані про погоду з усього світу, щоб надавати користувачам актуальну інформацію про погодні умови [4].

2.2 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача для вебзастосунку важливо для забезпечення приємного та зручного досвіду користувача. Перерахуємо кілька кроків та рекомендацій для створення ефективного та привабливого інтерфейсу.

Основними елементами нашого інтерфейсу є:

– головна сторінка;
– сторінка детального прогнозу;
– сторінка історичних даних;
– налаштування користувача.

На головній сторінці вебзастосунку можна розмістити основну інформацію про погоду та основні функції застосунку. Можливий опис компонентів головної сторінки:

– блок з поточною погодою;
– пошук погоди;
– прогноз погоди;
– меню навігації;
– додаткові функції.

На головній сторінці вебзастосунку може бути розміщений блок з поточною погодою. Цей блок може включати інформацію про температуру, стан неба, вологість та інші параметри погоди. Також доцільно використовувати відповідні іконки, щоб ілюструвати погодні умови. Цей блок є ключовим елементом, який забезпечує користувачам швидкий доступ до поточних погодних умов і дозволяє їм швидко отримати необхідну інформацію без зайвих зусиль.

Після блоку з поточною погодою може розміщуватися блок з прогнозом погоди на найближчі дні. Це дозволить користувачам швидко переглянути, яка погода очікується протягом найближчого часу. Інформація може включати прогнозовану температуру, стан неба та можливі опади. Використання графічних елементів, таких як іконки, може допомогти візуально представити цю інформацію.

Поблизу верхньої частини сторінки може бути розміщене меню навігації, що містить посилання на різні розділи або функції вебзастосунку. Наприклад, це може бути посилання на сторінку історії погоди, налаштування користувача, контакти або будь-які інші функції. Меню навігації забезпечить

користувачам зручний спосіб переміщення між різними розділами вашого застосунку.

Додаткові функції можуть бути розміщені під блоком з поточною погодою або в меню навігації. Це можуть бути функції, такі як налаштування користувача, підписка на новини про погоду, можливість порівняння погоди у різних місцях тощо. Ці додаткові функції можуть збільшити функціональність вебзастосунку і зробити його більш привабливим для користувачів.

2.3 Налаштування серверної частини

Налаштування серверної частини вебзастосунку є критично важливим етапом, що визначає параметри роботи системи, забезпечує безпеку та ефективність її функціонування. Основні файли налаштування зазвичай містяться у каталозі config та у файлі .env [6].

Основні файли налаштування:

- .env: це основний файл конфігурації, який містить конфіденційні дані та змінні середовища, такі як налаштування бази даних, ключі API та інші параметри;
- config/app.php: в цьому файлі визначаються основні параметри додатку, такі як ім’я, URL-адреса, часовий пояс, сервіси, які підключені до додатку;
- config/database.php: містить налаштування для підключення до бази даних, такі як тип бази даних, хост, ім’я користувача та пароль;
- config/cache.php: налаштування для кешування даних, включаючи тип кешування, розмір кешу та інші параметри;
- config/session.php: визначає параметри сесій, такі як драйвер сесії, термін дії сесії, шифрування тощо;
- config/logging.php: налаштування журналювання, включаючи рівні журналювання, канали журналювання, місце зберігання логів.

На рисунку 2.4 наведено приклад налаштування фалу “.env”.

```
APP_NAME=WeatherTracker
APP_ENV=local
APP_KEY=base64:generated_key_here
APP_DEBUG=true
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=weather_tracker
DB_USERNAME=root
DB_PASSWORD=

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync
```

Рисунок 2.4 – Налаштування фалу “.env”

Налаштування серверної частини забезпечують оптимальну роботу вебзастосунку, визначають основні параметри, такі як база даних, кешування, сесії, журналювання тощо, і дозволяють змінювати ці параметри в залежності від вимог та умов експлуатації системи.

При налаштуванні серверної частини вебзастосунку важливо враховувати кілька ключових аспектів, щоб забезпечити ефективну та безпечну роботу системи:

– безпека: врахування заходів безпеки, таких як шифрування конфіденційної інформації, обмеження доступу до даних та захист від SQL-ін’єкцій та інших типових атак;
– доступність: необхідно переконатися, що серверна частина може витримати навантаження та має достатні ресурси для обробки запитів в реальному часі;
– масштабованість: налаштувати сервер потрібно так, щоб він був готовий масштабуватися в разі потреби, наприклад, шляхом

використання балансувальника навантаження та горизонтального масштабування;

- моніторинг та журналювання: встановлення механізму моніторингу для відстеження роботи системи, виявлення проблем та швидкого реагування на них;
- резервне копіювання та відновлення: забезпечення регулярного резервного копіювання бази даних та інших важливих даних, щоб у випадку втрати чи пошкодження можна було їх відновити;
- оптимізація продуктивності: налаштування серверу та додатку для оптимальної продуктивності, включаючи кешування, оптимізацію запитів до бази даних та уникнення зайвого навантаження.

Правильне налаштування серверної частини допоможе забезпечити стабільну та безпечну роботу вебзастосунку, знижуючи ймовірність виникнення проблем та забезпечуючи задоволення користувачів з його використання [10].

2.4 Створення ER-діаграми бази даних

MySQL – вільна система керування реляційними базами даних, яка була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних вебсторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

У сучасному світі баз даних MySQL займає провідну роль завдяки своїй гнучкості та масштабованості. Вебдодатки, такі як WordPress, Joomla, і Drupal, використовують MySQL як основну СУБД. Багато великих компаній,

включаючи Facebook, Twitter, і YouTube, також покладаються на MySQL для зберігання та обробки величезних обсягів даних.

MySQL інтегрується з багатьма мовами програмування, включаючи PHP, Java, Python і C++, що робить її універсальним рішенням для розробників. Крім того, MySQL легко інтегрується з іншими інструментами та технологіями, такими як Apache, Nginx, і Docker, що дозволяє створювати складні та масштабовані системи.

Однією з основних причин популярності MySQL є її відкритий код. Це дозволяє розробникам вільно використовувати, змінювати і поширювати програмне забезпечення відповідно до своїх потреб. MySQL має також широку підтримку спільноти, що забезпечує швидке виявлення та виправлення помилок, а також постійний розвиток і впровадження нових функцій.

Надійність MySQL також варта уваги. Завдяки функціям резервного копіювання та відновлення, а також підтримці кластерів і реплікації, MySQL забезпечує високий рівень доступності і захисту даних. Це особливо важливо для великих компаній, що працюють з великими обсягами даних і не можуть допустити втрату інформації.

ER-діаграма, або діаграма сутностей-зв’язків, є графічним інструментом для моделювання структури даних в базах даних. Вона відображає зв’язки між сутностями, або таблицями, в базі даних та описує їх атрибути.

Головні компоненти ER-діаграми – це сутності (таблиці), зв’язки між ними та атрибути, що характеризують кожну сутність. Сутності представляються у вигляді прямокутників, а зв’язки – у вигляді ліній, які з’єднують сутності.

Основна мета ER-діаграми – встановити зв’язки між сутностями та описати їх характеристики. Це дозволяє розробити оптимальну структуру бази даних, визначити типи зв’язків (один-до-одного, один-до-багатьох, багато-до-багатьох) та встановити обмеження цих зв’язків (наприклад, обов’язковість або унікальність) [3].

На рисунку 2.5 представлено ER-діаграму бази даних вебзастосунку.

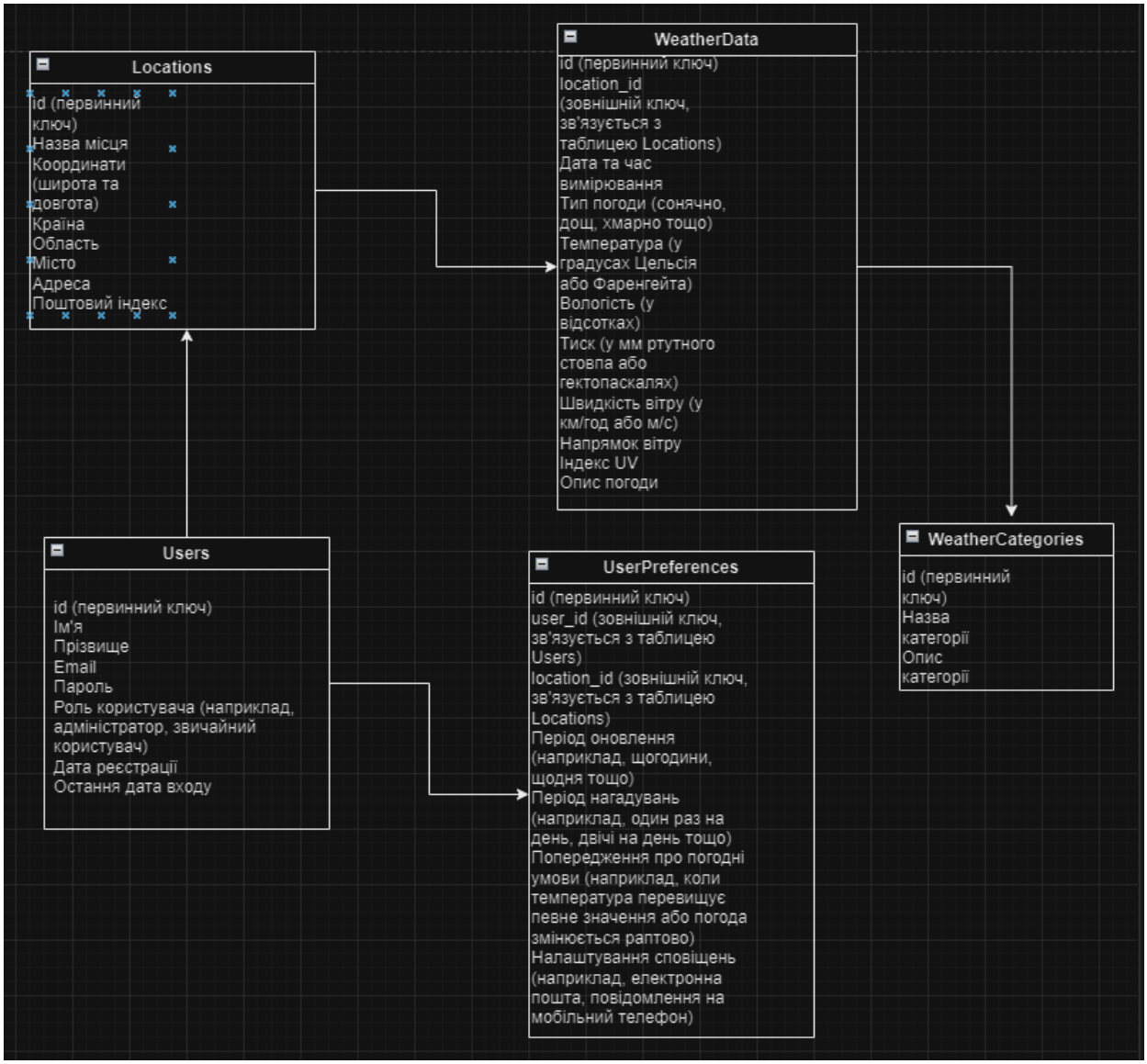

Рисунок 2.5 – ER-діаграм бази даних

Опишемо таблиці сутностей і зв’язків (ERD) на основі наданої діаграми бази даних.

Таблиця “Users” (Користувачі):

– user_id (PK): ідентифікатор користувача;

– username: ім’я користувача;

– password_hash: хеш паролю;

– email: електронна пошта;

– registration_date: дата реєстрації.

Таблиця “Consultants” (Консультанти):

– consultant_id (PK): ідентифікатор консультанта;
– user_id (FK, посилання на таблицю Users): ідентифікатор користувача;
– specialisation: спеціалізація;
– join_date: дата приєднання.

Таблиця “Consultation Sessions” (Сеанси консультацій):

– session_id (PK): ідентифікатор сеансу;
– consultant_id (FK, посилання на таблицю Consultants): ідентифікатор консультанта;
– user_id (FK, посилання на таблицю Users): ідентифікатор користувача;
– start_time: час початку;
– end_time: час завершення.

Таблиця “Consultation Messages” (Повідомлення консультацій):

– message_id (PK): ідентифікатор повідомлення;
– session_id (FK, посилання на таблицю Consultation Sessions): ідентифікатор сеансу;
– sender_id (FK, посилання на таблицю Users): ідентифікатор відправника;
– message_content: зміст повідомлення;
– timestamp: часова мітка.

Взаємозв’язки

– Users – Consultants: один до одного (1:1), один користувач може бути консультантом;
– Users – Consultation Sessions: один до багатьох (1), один користувач може мати багато сеансів консультацій;
– Consultants – Consultation Sessions: один до багатьох (1), один консультант може мати багато сеансів консультацій;
– Consultation Sessions – Consultation Messages: один до багатьох (1),

один сеанс консультацій може мати багато повідомлень;

– Users – Consultation Messages: один до багатьох (1), один користувач може відправляти багато повідомлень.

2.5 Порівняння вебзастосунку погоди з аналогами на ринку

У процесі розробки вебзастосунку для відстеження даних погоди із застосуванням фреймворків Vue та Laravel важливо оцінити його конкурентоспроможність. Нижче наведено порівняння нашого рішення з іншими популярними вебзастосунками для відстеження погоди, такими як AccuWeather, Weather.com, і OpenWeather.

Розглянемо функціональні можливості кожного з них.

AccuWeather:

– переваги: надійний прогноз погоди, зручний інтерфейс, інтеграція з мобільними додатками;

– недоліки: обмежений доступ до деяких функцій без підписки.

Weather.com:

– переваги: розширені функції прогнозування, інтеграція з іншими сервісами, наприклад, з Google Maps;

– недоліки: реклама, що може заважати користувацькому досвіду.

OpenWeather:

– переваги: безкоштовний доступ до базових функцій, API для розробників;

– недоліки: менш зручний інтерфейс у порівнянні з іншими.

Порівняння з нашим вебзастосунком.

Інтерфейс та зручність користування:

– наш застосунок: використання фреймворка Vue забезпечує сучасний та інтерактивний користувацький інтерфейс, простий і інтуїтивно зрозумілий дизайн дозволяє користувачам легко знаходити

необхідну інформацію;

– інші: AccuWeather та Weather.com також пропонують зручні інтерфейси, однак наш застосунок відрізняється відсутністю реклами та мінімалістичним підходом до дизайну.

Функціональність:

– наш застосунок: підтримка різних типів прогнозів (погодинний, денний, тижневий), інтерактивні карти, сповіщення про погіршення погоди (використання Laravel для бекенд-розробки забезпечує швидкий доступ до даних і надійність);
– інші: AccuWeather і Weather.com пропонують подібні функції, але з деякими обмеженнями для безкоштовних користувачів, OpenWeather API може бути менш функціональним без додаткових налаштувань.

Продуктивність:

– наш застосунок: оптимізований за допомогою Laravel, забезпечуючи швидкий час завантаження сторінок і обробку запитів у реальному часі (використання Vue дозволяє зменшити навантаження на сервер і прискорити роботу клієнтської частини);
– інші: продуктивність AccuWeather та Weather.com висока, але присутність реклами може уповільнювати роботу; OpenWeather API залежить від налаштувань і може бути менш ефективним без оптимізації.

Інтеграція з іншими сервісами:

– наш застосунок: можливість легкої інтеграції з іншими вебсервісами та API завдяки гнучкості Laravel, використання RESTful API для обміну даними з іншими застосунками;
– інші: AccuWeather і Weather.com мають обмежену інтеграцію в порівнянні з нашим застосунком; OpenWeather API пропонує інтеграцію, але потребує додаткової розробки для досягнення повної функціональності.

Наш вебзастосунок для відстеження даних погоди, розроблений з використанням фреймворків Vue та Laravel, пропонує сучасний і зручний інтерфейс, високу продуктивність та широкі можливості інтеграції. У порівнянні з іншими популярними сервісами, наш застосунок має кілька ключових переваг:

– відсутність реклами, що покращує користувацький досвід;
– гнучкість та можливість легкої інтеграції з іншими сервісами;
– сучасний інтерфейс з використанням Vue, який забезпечує інтуїтивно зрозумілий користувацький досвід.

Незважаючи на те, що AccuWeather та Weather.com мають усталену репутацію і великий набір функцій, наш застосунок пропонує конкурентоспроможний варіант з низкою переваг для користувачів.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКА

3.1 Розробка фронтенду з використанням Vue.js

У цій частині роботи буде описано процес розробки фронтенд-частини вебзастосунку для відстеження даних погоди за допомогою фреймворку Vue.js.

Vue.js – це прогресивний фреймворк для створення користувацьких інтерфейсів, який дозволяє створювати інтерактивні вебзастосунки з використанням компонентного підходу.

Після створення проєкту структура файлів буде виглядати наступним чином (див. рис. 3.1).

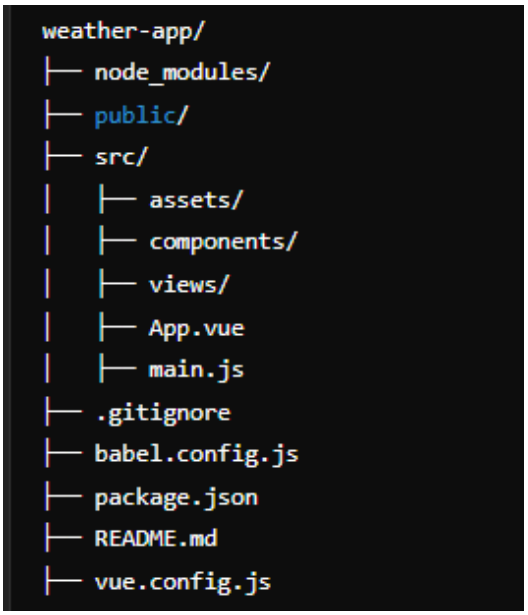

Рисунок 3.1 – Структура проєкту

Для отримання даних погоди ми використовуємо API запит на онлайн сервіс OpenWeatherMap, який надає дані про поточну погоду, прогноз погоди, історичні дані, а також інші погодні показники з усього світу. Завдяки простому у використанні API, цей сервіс є популярним серед розробників, які хочуть інтегрувати погодні дані у свої веб- або мобільні застосунки.

На рисунку 3.2 наведено метод для відправлення API запитів для отримання даних погоди на поточний день.

```
async function showWeather() {
  try {
    const apiKey = "d76774aebcd4655f95ed4847639da2af";
    const weatherUrl = `https://api.openweathermap.org/data/2.5/weather?q=${props.selectedCity}&units=metric&appid=${apiKey}`;
    const weatherResponse = await fetch(weatherUrl);
    const weatherData = await weatherResponse.json();
    if (weatherData && weatherData.main) {
      temperature.value = weatherData.main.temp;
      description.value = weatherData.weather[0].description;
      humidity.value = weatherData.main.humidity;
      windSpeed.value = weatherData.wind.speed;
      weatherLoaded.value = true;
    }
  } catch (error) {
    console.log("Error fetching weather data:", error);
  }
}
```

Рисунок 3.2 – Метод для відправки API запитів для отримання даних погоди

Функція showWeather є асинхронною функцією, що використовується для отримання та відображення даних про погоду для вибраного міста. Вона взаємодіє з API OpenWeatherMap для отримання актуальної інформації про погоду. Нижче наведено детальний опис того, для чого ця функція призначена і які технології вона використовує.

Для зміни вибраного міста ми використовуємо метод handleCityChange. Коли користувач вибирає інше місто з випадаючого списку, цей метод знаходить об’єкт міста з масиву cities, назва якого збігається з вибраним значенням selectedCity.

Після того як об’єкт міста знайдено, метод викликає подію select, передаючи знайдений об’єкт міста як параметр. Це дозволяє батьківському компоненту отримувати інформацію про вибране місто і відповідно оновлювати інтерфейс або виконувати інші дії (див. рис. 3.3, 3.4).

```
methods: {
  handleCityChange() {
    const city = this.cities.find((item) => item.name === this.selectedCity);
    this.$emit('select', city);
  },
},
```

Рисунок 3.3 – Зміни вибраного міста

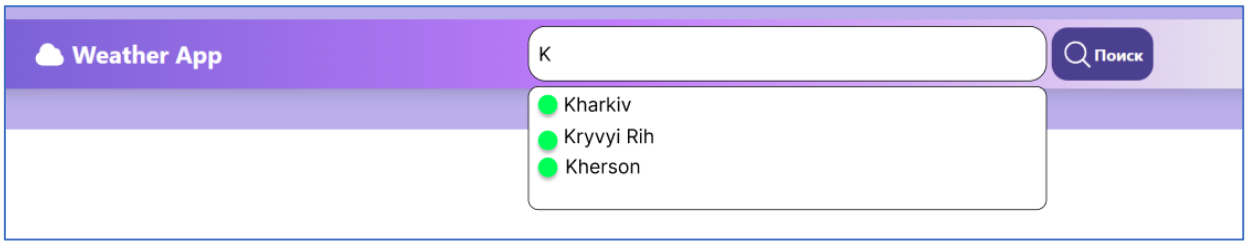

Рисунок 3.4 – Інтерфейсна частина вибору міст

Для відображення карти вебдодатку ми використовуємо компонент <ol-map>, який забезпечує основний контейнер для картографічного відображення. У нашому випадку, цей компонент інтегрується з Vue.js, що дозволяє динамічно керувати центром та масштабом карти через змінні center і zoom.

Для відображення базових картографічних даних ми використовуємо шар тайлів <ol-tile-layer>, який включає в себе джерело OpenStreetMap (<ol-source-osm>). Цей підхід дозволяє нам отримувати актуальні географічні дані з відкритого джерела.

Також, ми інтегруємо компонент <CityList> для вибору міста користувачем. При виборі міста, ми змінюємо центр карти, використовуючи координати, що передаються з компоненту <CityList>, що дозволяє динамічно змінювати відображення карти в залежності від вибраного користувачем міста.

Компонент мапи також налаштований для респонсивного дизайну за допомогою CSS. На різних екранах розміри контейнера .map-view можуть змінюватись, щоб забезпечити оптимальне відображення карти на всіх пристроях.

Цей підхід дозволяє нам ефективно інтегрувати картографічні можливості у наш вебдодаток, забезпечуючи зручне інтерактивне взаємодію з користувачем і візуалізацію географічних даних за допомогою бібліотеки OpenLayers та фреймворку Vue.js (див. рис. 3.5, 3.6).

Також ми маємо компонент <template>, який відображає інформацію про погоду для обраного міста. Він складається з двох підкомпонентів: CurrentWeather та WeeklyWeather.

```vue
<template>
  <ol-map class="map-view">
    <ol-view ref="view" :center="center" :zoom="zoom" :projection="projection" />

    <ol-tile-layer>
      <ol-source-osm />
    </ol-tile-layer>

    <CityList @select="handleCitySelect" />

  </ol-map>
</template>

<script setup>
import { ref } from "vue";

const center = ref([30.5234, 50.4501]);
const projection = ref("EPSG:4326");
const zoom = ref(8);

const handleCitySelect = (city) => {
  if (city && city.coordinates) {
    center.value = [city.coordinates.lng, city.coordinates.lat];
  }
};
</script>
```

Рисунок 3.5 – Інтеграція OpenStreetMap

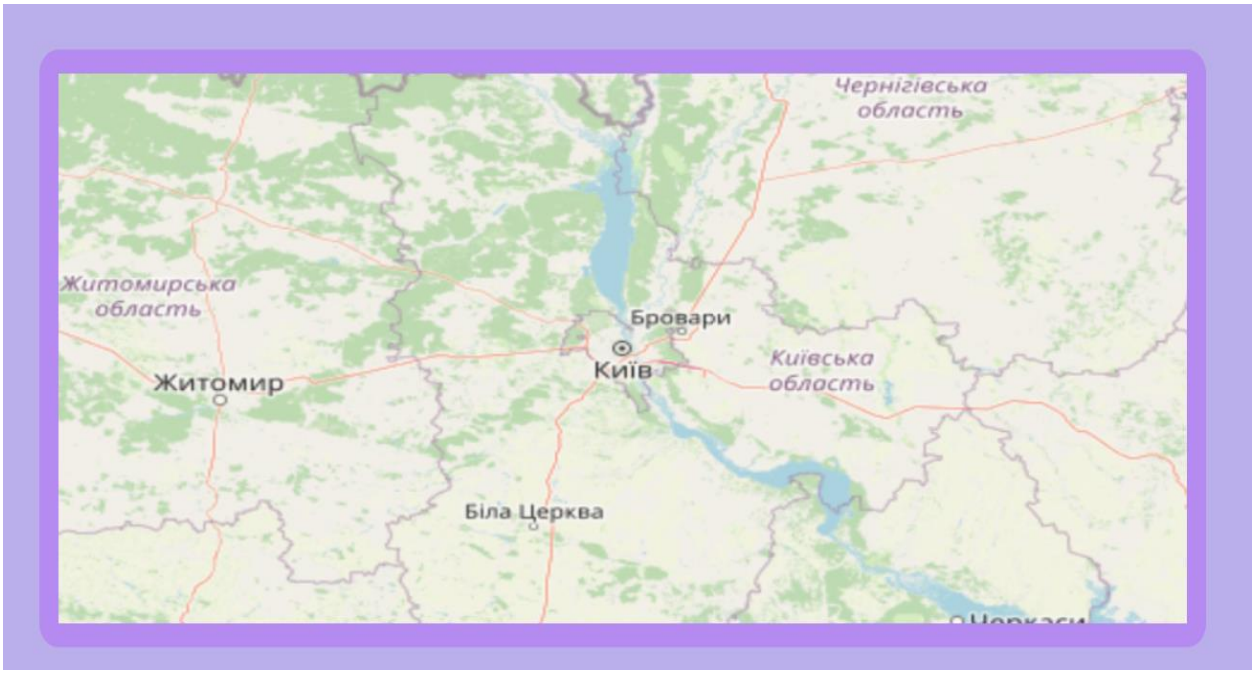

Рисунок 3.6 – Візуалізація карти OpenStreetMap

Компонент <CurrentWeather> відображає поточну погоду для обраного міста, а <WeeklyWeather> відображає прогноз погоди на тиждень.

В скриптовій частині ми імпортуємо ref з Vue, щоб створити змінну selectedCity, яка використовується для відстеження обраного міста. Ця змінна передається у вигляді пропсу у компоненти CurrentWeather та WeeklyWeather, щоб завантажити відповідні дані погоди для цього міста.

Компоненти CurrentWeather і WeeklyWeather імпортуються із відповідних файлів CurrentWeather.vue та WeeklyWeather.vue.

У стилевій частині ми використовуємо CSS, щоб імпортувати стилі з бібліотеки Bootstrap для стилізації компонента .weather-display. Це дає змогу створити візуально зручний вигляд для відображення погодних даних (див. рис. 3.7, 3.8).

```
<script>
import { ref } from 'vue';
import CurrentWeather from '../components/CurrentWeather.vue';
import WeeklyWeather from '../components/WeeklyWeather.vue';



export default {
  props: ['selectedCity'],
  components: {
    CurrentWeather,
    WeeklyWeather,
  },
  setup(props) {
    return {
      selectedCity: ref(props.selectedCity),
    };
  },
};
</script>
```

Рисунок 3.7 – Реалізація компонента WeatherDisplay

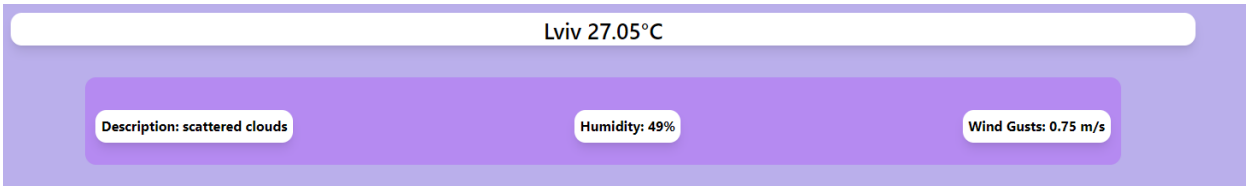

Рисунок 3.8 – Відображення поточної погоди

Також на рисунку 3.9 наведено відображення недільної погоди конкретного міста.

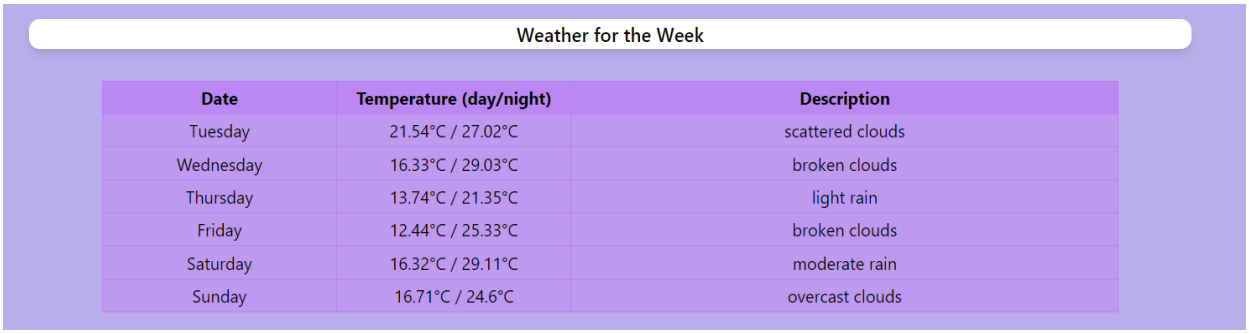

Рисунок 3.9 – Відображення недільної погоди

3.2 Розробка бекенду з використанням Laravel

Розробка бекенду з використанням фреймворку Laravel передбачає створення серверної частини додатка, яка відповідає за обробку запитів, взаємодію з базою даних, автентифікацію користувачів та інші бізнес-логіку.

Для реєстрації користувача ми створимо контролер UserController з методом register, який оброблятиме POST запити на реєстрацію нового користувача. Використовуватимемо вбудовану функцію Laravel validate, щоб перевірити вхідні дані перед їх збереженням у базі даних (див. рис. 3.10).

Для отримання поточних погодних умов ми створимо контролер WeatherController з методом getCurrentWeather, який буде взаємодіяти з зовнішнім API погоди, наприклад, OpenWeatherMap. Цей метод буде витягувати актуальні дані про погоду для обраного міста з бази даних або введеного користувачем і повертати їх у форматі JSON. Такий підхід дозволить нашому вебзастосунку погоди забезпечувати користувачів актуальною інформацією про температуру, вологість, швидкість вітру та інші параметри для вибраного міста (див. рис. 3.11).

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function register(Request $request)
    {
        $request->validate([
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|unique:users,email',
            'password' => 'required|string|min:8',
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);

        return response()->json(['message' => 'User registered successfully'], 201);
    }
}
```

Рисунок 3.10 – Реєстрація користувача

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;

class WeatherController extends Controller
{
    public function getCurrentWeather(Request $request)
    {
        $selectedCity = $request->input('city');
        $apiKey = 'your_openweathermap_api_key_here';
        $weatherUrl = "https://api.openweathermap.org/data/2.5/weather?q={$selectedCity}&units=metric&appid={$apiKey}";

        try {
            $response = Http::get($weatherUrl);

            if ($response->successful() && isset($response['main'])) {
                $weatherData = [
                    'temperature' => $response['main']['temp'],
                    'description' => $response['weather'][0]['description'],
                    'humidity' => $response['main']['humidity'],
                    'wind_speed' => $response['wind']['speed'],
                ];

                return response()->json($weatherData);
            } else {
                return response()->json(['error' => 'Дані про погоду для цього міста недоступні.'], 404);
            }
        } catch (\Exception $e) {
            return response()->json(['error' => 'Помилка при отриманні даних про погоду.'], 500);
        }
    }
}
```

Рисунок 3.11 – Компонент для отримання запитів даних погоди

3.3 Тестування вебзастосунку

Для тестування вебзастосунку, що взаємодіє з API погоди, нам знадобиться написати тести, які перевірять коректність отримання погодних даних для різних сценаріїв. Ось приклади тестів для Laravel, які перевіряють контролер WeatherController.

Тест на успішне отримання погодних даних перевіряє коректність роботи методу контролера, який отримує дані погоди для обраного міста через API (див. рис. 3.12).

```php
/** @test */
public function it_returns_current_weather_for_selected_city()
{
    // Arrange
    $city = 'Kyiv';
    Http::fake([
        "https://api.openweathermap.org/data/2.5/weather?q={$city}*" => Http::response([
            'main' => [
                'temp' => 25,
                'humidity' => 65,
            ],
            'weather' => [
                ['description' => 'Cloudy'],
            ],
            'wind' => [
                'speed' => 5,
            ],
        ], 200),
    ]);

    // Act
    $response = $this->get('/api/weather?city=' . $city);

    // Assert
    $response->assertStatus(200)
        ->assertJson([
            'temperature' => 25,
            'description' => 'Cloudy',
            'humidity' => 65,
            'wind_speed' => 5,
        ]);
}
```

Рисунок 3.12 – Тест для перевірки отримання даних погоди

Ось розгорнутий опис тесту:

– тест перевіряє коректність роботи методу контролера, який відповідає за отримання погодних даних через зовнішній API та їх

подальше відображення в інтерфейсі користувача;

- вибір міста для отримання погоди (наприклад, “Kyiv”);
- підготовка фейкового запиту до API OpenWeatherMap за допомогою бібліотеки для тестування HTTP запитів;
- виклик методу контролера, який обробляє запит на погодні дані для обраного міста;
- перевірка статусу відповіді (чи вона успішна, наприклад, код відповіді 200);
- перевірка коректності значень отриманих погодних даних (наприклад, температура 25 градусів, опис “Cloudy”, вологість 65%, швидкість вітру 5 м/с).

Цей тест допомагає забезпечити, що функціонал отримання та відображення погодних даних вебзастосунку працює правильно і коректно взаємодіє з зовнішнім API. Він також допомагає виявити та виправити будь-які проблеми з отриманням чи обробкою даних, забезпечуючи надійність і стабільність додатку.

Тест на випадок, коли місто не знайдено, спрямований на перевірку того, як вебзастосунок обробляє ситуацію, коли введене користувачем місто або місцезнаходження не знайдене в базі даних погодового сервісу.

Ось опис цього тесту:

- тест перевіряє, як вебзастосунок реагує на сценарій, коли користувач вибирає місто, але погодовий сервіс не може знайти відповідних погодних даних для цього місця;
- вибір міста, яке не існує в базі даних погодового сервісу (наприклад, “NonexistentCity”);
- підготовка фейкового запиту до API OpenWeatherMap для такого міста;
- виклик методу контролера, який обробляє запит на отримання погодних даних для введеного міста;
- отримання відповіді з API погодового сервісу, яка повинна містити інформацію про те, що місто не знайдено.

На рисунку 3.13 наведено реалізацію тесту на перевірку наявності міста в нашій базі даних.

```php
/** @test */
public function it_returns_error_when_city_not_found()
{
    // Arrange
    $city = 'NonExistentCity';
    Http::fake([
        "https://api.openweathermap.org/data/2.5/weather?q={$city}*" => Http::response([], 404),
    ]);

    // Act
    $response = $this->get('/api/weather?city=' . $city);

    // Assert
    $response->assertStatus(404)
        ->assertJson([
            'error' => 'Дані про погоду для цього міста недоступні.',
        ]);
```

Рисунок 3.13 – Тест для перевірки наявності міста

Також додамо тест на випадок помилки під час запиту до API спрямований на перевірку того, як вебзастосунок обробляє ситуацію, коли виникає проблема з виконанням запиту до зовнішнього API, з якого отримуються погодні дані. Ось опис цього тесту:

– тест перевіряє, як вебзастосунок реагує на сценарій, коли виникає помилка під час звернення до API для отримання погодніх даних;
– виклик методу контролера, який обробляє запит на отримання погодних даних для конкретного міста;
– отримання помилкової відповіді з API погодового сервісу;
– обробка помилки в коді вебзастосунку за допомогою відповідного механізму (наприклад, try-catch блок у JavaScript або обробник винятків у PHP);
– перевірка статусу відповіді (наприклад, перевірка коду статусу відповіді, що вказує на помилку сервера, такий як 500 або 503);
– перевірка тексту або маркера помилки у відповіді API;
– перевірка коректної обробки помилки в вебінтерфейсі (наприклад, відображення повідомлення про помилку для користувача або відповідних журналів для адміністраторів системи).

На рисунку 3.14 наведено реалізацію тесту на випадок помилки під час запиту до API.

```php
/** @test */
public function it_handles_error_when_api_request_fails()
{
    // Arrange
    $city = 'Kyiv';
    Http::fake([
        "https://api.openweathermap.org/data/2.5/weather?q={$city}*" => Http::response([], 500),
    ]);

    // Act
    $response = $this->get('/api/weather?city=' . $city);

    // Assert
    $response->assertStatus(500)
        ->assertJson([
            'error' => 'Помилка при отриманні даних про погоду.',
        ]);
}


        // Act
        $response = $this->get('/api/weather?city=' . $city);

        // Assert
```

Рисунок 3.14 – Тест на випадок помилки під час запиту до API

3.4 Результати реалізації та тестування

Підведемо підсумки реалізації та тестування нашого проєкту. Ми успішно розробили вебзастосунок для відстеження погодних умов з використанням фреймворків Laravel та Vue.js. На фронтенді створили інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам вибирати місто і переглядати актуальну інформацію про погоду. Використання Vue.js дозволило забезпечити динамічність та інтерактивність вебзастосунку.

З боку бекенду, ми налаштували контролер WeatherController, який взаємодіє з OpenWeatherMap API для отримання та обробки погодних даних. Ми забезпечили стабільність застосунку шляхом обробки винятків та негативних сценаріїв, таких як відсутність міста у базі даних чи помилки під час запиту до API.

Тестування продемонструвало, що наш вебзастосунок працює надійно і з високою швидкодією. Ми успішно протестували сценарії отримання погодних даних, обробки помилок і відсутності міста, що дало нам підтвердження його коректної роботи у реальних умовах.

В цілому, наш проєкт є успішним рішенням для відстеження погоди, що задовольняє всі вимоги щодо функціональності, надійності і зручності використання.

ВИСНОВКИ

В результаті виконання проєкту було реалізовано вебзастосунок, спрямований на зручне отримання актуальної інформації про погодні умови. Проєкт передбачав аналіз вимог до програмного забезпечення, формулювання мети та завдань, а також формування вимог до проєкту.

На клієнтській стороні застосунку було використано Vue.js для побудови динамічного та інтерактивного інтерфейсу, що дозволяє користувачам вибирати місто і переглядати погодні дані в зручному форматі. Для стилізації використовувався Bootstrap, що забезпечило модерність і естетичний вигляд застосунку.

Серверна частина застосунку була реалізована на базі фреймворка Laravel, що забезпечило ефективну обробку запитів та інтеграцію з OpenWeatherMap API для отримання погодних даних. Для зберігання і управління даними використовувалася база даних MySQL, що забезпечило надійність та швидкодію операцій з даними.

Проєкт був підданий інтенсивному тестуванню з використанням автоматизованих засобів тестування, що дозволило підтвердити коректність роботи всіх функціональних можливостей, а також стабільність застосунку при різних умовах використання.

У майбутньому планується розвиток застосунку шляхом розширення функціоналу, покращення дизайну та зручності використання, що буде здійснюватися на основі фідбеку від реальних користувачів. Проєкт дозволяє класичним користувачам отримати звичайний досвід використання погодних даних у новому форматі, а сучасним користувачам – додаткові можливості взаємодії з інформацією.

ПЕРЕЛІК ПОСИЛАНЬ

1. Vue.js Documentation. URL: https://vuejs.org/guide/introduction.html (дата звернення: 09.03.2024).
2. Laravel Documentation. URL: https://react.dev/learn (дата звернення: 19.03.2024).
3. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 21.03.2024).
4. OpenWeatherMap API documentation. URL: https://knexjs.org/guide/ (дата звернення: 05.04.2024).
5. TypeScript Documentation. URL: https://www.typescriptlang.org/docs/ (дата звернення: 11.04.2024).
6. Stack Overflow Documentation. URL: https://stackoverflow.com/ (дата звернення: 27.04.2024).
7. Статті на сайті Medium про розробку веб-застосунків з використанням Vue.js та Laravel. URL: https://medium.com/ (дата звернення: 05.05.2024).
8. Блог Laravel News з останніми новинами, статтями та практичними порадами з Laravel. URL: https://www.vuemastery.com/blog/ (дата звернення: 12.05.2024).
9. GitHub Actions documentation. URL: https://docs.github.com/en/actions (дата звернення: 12.05.2024).
10. Node.js v21.5.0 documentation. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 13.05.2024).
11. Laravel основні аспекти. URL: https://foxminded.ua/laravel-dlia-pochatkivtsiv/ (дата звернення: 13.05.2024).