МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ГЕОІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ПОГОДИ»**

Виконав: студент 4 курсу, групи 6.1210-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
В.А. Рудакевич
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Мильцев О.М.
(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

Рудакевичу Владиславу Артемовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка геоінформаційної системи моніторингу погоди

керівник роботи Мильцев Олександр Михайлович, к.ф.-м.н.
(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.
3. Вихідні дані до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Проєктування та розробка застосунку моніторингу 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.	09.01.2024	
2.	Збір вихідних даних.	02.02.2024	
3.	Обробка методичних та теоретичних	01.03.2024	
	джерел.		
4.	Розробка першого та другого розділу.	26.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	21.06.2024	

Студент _____ В.А. Рудакевич
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.М. Мильцев
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка геоінформаційної системи моніторингу погоди»: 57 с., 17 рис., 11 джерел.

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

Об’єкт дослідження – процес розробки геоінформаційної системи моніторингу погоди.

Мета роботи: розробка геоінформаційної системи моніторингу погоди.

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

Предмет дослідження – розробка геоінформаційної системи моніторингу погоди із застосуванням фреймворків Laravel та Vue.js.

У роботі наведено етапи розробки геоінформаційної системи моніторингу погоди із застосуванням фреймворків Laravel та Vue.js, виявлено ключові проблеми та застосовано нові методи для збору та аналізу вимог до програмного забезпечення. Описано предметну область. Розроблено геоінформаційну систему моніторингу погоди, яка використовує фреймворки Laravel та Vue.js для реалізації функціонала моніторингу погоди та комунікації з користувачами в реальному часі. Зростання попиту на інтерактивні онлайн сервіси робить важливим розв’язання таких задач, як розгортання та підтримка систем моніторингу погоди. Тому, розробка геоінформаційної системи моніторингу погоди із застосуванням фреймворків Laravel та Vue.js є актуальною задачею.

SUMMARY

Bachelor's qualifying paper "Development of a Geoinformation System for the Weather Monitoring": 57 pages, 17 figures, 11 references.

COMPONENT, CONTROLLER, DATABASE, FRAMEWORK, GEOINFORMATION SYSTEM, LARAVEL, MODEL, VUE.JS, WEB APPLICATION.

Object of study: the process of developing a weather monitoring geoinformation system.

Purpose of the work: development of a weather monitoring geoinformation system.

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

Subject of study: development of a weather monitoring geoinformation system using Laravel and Vue.js frameworks.

The work outlines the stages of developing a weather monitoring geoinformation system using the Laravel and Vue.js frameworks, identifies key issues, and applies new methods for collecting and analyzing software requirements. The subject area is described. A weather monitoring geoinformation system has been developed using the Laravel and Vue.js frameworks, which are used to implement weather monitoring functionalities and real-time communication with users. The growing demand for interactive online services highlights the importance of solving such tasks, deploying, and maintaining weather monitoring systems. Therefore, the development of a weather monitoring geoinformation system using the Laravel and Vue.js frameworks is a relevant task.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API	Application Programming Interface
CLI	Command Line Interface
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
ORM	Object-Relational Mapping
REST	Representational State Transfer
SQL	Structured Query Language
URL	Uniform Resource Locator

ВСТУП

У сучасному цифровому ландшафті розробка програмного забезпечення є однією з найбільш динамічних та інноваційних галузей. Щороку зростає вимоги до функціональності та ефективності програмних продуктів, від користувачів очікується вищий рівень зручності та доступності. У цьому контексті особливе значення набуває розробка геоінформаційних систем, що потребує високого рівня технічної експертизи, а також глибокого розуміння потреб ринку та вимог споживачів.

Мета цієї кваліфікаційної роботи полягає в аналізі етапів розробки геоінформаційної системи моніторингу погоди, з фокусом на використанні фреймворків Vue.js для клієнтської частини та Laravel для серверної. Відповідно до обраної теми, в роботі буде проаналізовано наступні ключові етапи:

- аналіз вимог користувачів та технічне завдання;
- проєктування архітектури з урахуванням особливостей збору та відображення метеоданих;
- розробка функціоналу для забезпечення точності та швидкості відображення даних погоди;
- тестування для перевірки стабільності та відповідності вимогам;
- впровадження системи та підготовка до релізу;
- підтримка та оновлення додатка для забезпечення актуальності та безпеки.

Кожен з цих етапів містить детальне обґрунтування вибору технологічного стека, а також аналіз потенційних складнощів і способів їх вирішення. Висвітлюються ключові аспекти та приклади реалізації, що демонструють практичне застосування кожного етапу у процесі створення геоінформаційної системи моніторингу погодних умов.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ

1.1 Поняття та принципи геоінформаційних систем

Геоінформаційна система (ГІС) – це інтегрована система, що складається з апаратних і програмних засобів, даних та персоналу, яка дозволяє збирати, зберігати, аналізувати, управляти та відображати географічні дані. ГІС дозволяють обробляти просторову інформацію, що включає дані про положення об’єктів на Землі, їхні характеристики та взаємозв’язки.

Основні компоненти ГІС:

– апаратне забезпечення: комп’ютери, сервери, мережеве обладнання та інші пристрої, необхідні для функціонування ГІС;
– програмне забезпечення: спеціалізовані програми для обробки геопросторових даних, такі як ArcGIS, QGIS та інші;
– дані: просторові та атрибутивні дані, що використовуються в ГІС – просторові дані включають координати об’єктів, а атрибутивні – їхні властивості;
– методи та технології: техніки та алгоритми, що використовуються для обробки, аналізу та візуалізації даних;
– люди: користувачі, аналітики, розробники та інші спеціалісти, які працюють з ГІС.

Принципи функціонування ГІС:

– збір даних: процес отримання просторових даних з різних джерел, таких як супутникові знімки, аерофотознімання, GPS, карти та інші;
– зберігання даних: організація даних у зручному для використання форматі, що забезпечує швидкий доступ до необхідної інформації;
– аналіз даних: застосування різних методів та інструментів для обробки та інтерпретації даних, виявлення просторових закономірностей та тенденцій;

– управління даними: забезпечення цілісності, точності та актуальності даних, а також їх захист від несанкціонованого доступу;
– відображення даних: використання карт, графіків, діаграм та інших візуальних засобів для представлення результатів аналізу у зручному для користувача вигляді.

ГІС знаходять широке застосування в різних галузях, включаючи:

– картографія: створення детальних та точних карт;
– управління ресурсами: моніторинг і управління природними ресурсами, такими як ліси, водні ресурси, землі;
– містобудування: планування та управління міською інфраструктурою;
– екологія: моніторинг стану навколишнього середовища та прогнозування природних явищ;
– транспорт: оптимізація транспортних маршрутів і управління логістикою;
– оборона та безпека: підтримка військових операцій та забезпечення громадської безпеки.

ГІС є потужним інструментом для прийняття обґрунтованих рішень у різних сферах, забезпечуючи точну та актуальну інформацію про просторові об’єкти та їхні характеристики. Наведемо приклади Геоінформаційних Систем (ГІС).

ArcGIS від компанії Esri є однією з найпоширеніших та найпотужніших платформ для роботи з геопросторовими даними. Вона містить різноманітні інструменти для створення, управління, аналізу та візуалізації географічної інформації.

ArcGIS використовується в різних галузях, таких як екологія, містобудування, транспорт, управління ресурсами та інші.

Розглянемо основні компоненти ArcGIS.

ArcMap:

– інструмент для створення, редагування та відображення карт;
– дозволяє працювати з векторними та растровими даними;
– підтримує аналіз просторових даних за допомогою різноманітних геообробних інструментів.

ArcCatalog:

– програма для управління геопросторовими даними;
– дозволяє переглядати, організовувати та редагувати метадані;
– забезпечує зручний доступ до баз даних та файлових систем.

ArcGIS Online:

– хмарний сервіс для створення та спільного використання карт і геопросторових даних;
– дозволяє публікувати вебкарти, вебдодатки та геообробні сервіси;
– підтримує спільну роботу користувачів та організацій.

ArcGIS Pro:

– сучасний додаток для роботи з 2D та 3D даними;
– підтримує аналіз великих обсягів даних і інтеграцію з іншими системами;
– дозволяє створювати складні картографічні продукти та анімації.

ArcGIS Enterprise:

– повний пакет програмного забезпечення для розгортання ГІС в інфраструктурі організації;
– включає сервери для обробки даних, вебдодатки та бази даних;
– забезпечує масштабованість та безпеку геопросторових даних.

Розглянемо основні можливості ArcGIS.

Збір та інтеграція даних: ArcGIS підтримує роботу з широким спектром форматів даних, включаючи векторні, растрові, табличні та 3D дані. Це дозволяє інтегрувати інформацію з різних джерел, таких як супутникові знімки, аерофотознімання, GPS та інші.

Аналіз просторових даних: вбудовані інструменти для геообробки

дозволяють проводити складний аналіз даних, виявляти просторові закономірності, моделювати процеси та робити прогнози.

Картографія та візуалізація: ArcGIS пропонує потужні інструменти для створення детальних карт, графіків та інших візуальних представлень даних. Це допомагає візуалізувати результати аналізу та приймати обґрунтовані рішення.

Спільна робота та публікація: за допомогою ArcGIS Online та ArcGIS Enterprise користувачі можуть спільно працювати над проєктами, публікувати карти та додатки, а також обмінюватися даними з іншими користувачами та організаціями.

Розширюваність та інтеграція: ArcGIS підтримує розробку користувацьких додатків та розширень за допомогою API та SDK (від англ. software Development Kit). Це дозволяє інтегрувати ГІС з іншими системами та налаштовувати її під конкретні потреби користувачів.

QGIS (Quantum GIS) – це відкрита та безкоштовна геоінформаційна система (ГІС), яка забезпечує широкий набір функцій для аналізу та візуалізації геопросторових даних. QGIS активно використовується у наукових дослідженнях, управлінні природними ресурсами, освіті та інших сферах. Її гнучкість, велика спільнота користувачів та безкоштовний доступ роблять її популярним вибором серед ГІС-фахівців та організацій по всьому світу.

Розглянемо основні можливості QGIS.

Підтримка численних форматів даних: QGIS підтримує широкий спектр форматів геопросторових даних, включаючи векторні та растрові формати, такі як Shapefile, GeoJSON, KML, GeoTIFF та багато інших. Це дозволяє інтегрувати інформацію з різних джерел і використовувати їх в одному проєкті.

Картографія та візуалізація: QGIS надає потужні інструменти для створення картографічних продуктів. Користувачі можуть налаштовувати стиль та символогію шарів, створювати тематичні карти та діаграми,

використовувати різноманітні картографічні проекції.

Аналіз просторових даних: інструменти для геообробки у QGIS дозволяють проводити складний просторовий аналіз. Це включає буферний аналіз, перехресний аналіз шарів, створення ізоліній, просторову статистику та багато іншого.

Підтримка плагінів: одна з ключових переваг QGIS – це велика бібліотека плагінів, які розширюють його функціональність. Користувачі можуть встановлювати плагіни для роботи з GPS, проведення специфічного аналізу, інтеграції з іншими ГІС-системами тощо.

Робота з базами даних: QGIS підтримує інтеграцію з різними базами даних, такими як PostGIS, SpatiaLite, MySQL та інші. Це дозволяє ефективно працювати з великими обсягами геопросторових даних та забезпечує зручний доступ до даних.

Відкритий код та активна спільнота: QGIS є проєктом з відкритим кодом, що означає, що він безкоштовний для використання та модифікації. Активна спільнота розробників і користувачів постійно вдосконалює програму, додає нові функції та виправляє помилки.

Розглянемо основні компоненти QGIS.

QGIS Desktop – основний додаток для роботи з геопросторовими даними. Він дозволяє створювати, редагувати та аналізувати карти, виконувати просторовий аналіз і працювати з базами даних.

QGIS Server – компонент для публікації карт та геопросторових даних у вигляді вебсервісів. QGIS Server підтримує стандарти OGC, такі як WMS, WFS і WCS, що дозволяє інтегрувати його з іншими веб-ГІС додатками.

QGIS Browser – легкий додаток для перегляду та управління геопросторовими даними. Він дозволяє швидко переглядати вміст файлів та баз даних, а також додавати їх у QGIS Desktop.

Google Earth Engine – це потужна платформа для аналізу геопросторових даних, яка використовує хмарні обчислення для обробки великих обсягів даних дистанційного зондування. Вона широко

використовується для моніторингу змін довкілля, управління земельними ресурсами, вивчення кліматичних змін та інших задач, що потребують обробки великої кількості геопросторових даних. Google Earth Engine забезпечує доступ до великої бібліотеки супутникових знімків та інструментів для їх аналізу.

OpenStreetMap (OSM) – це відкрита карта світу, яка створюється спільнотою користувачів. Вона забезпечує доступ до детальних картографічних даних, які можна використовувати для різних цілей, включно навігацію, планування маршрутів, аналіз просторових даних тощо. OSM використовується у численних проєктах, де потрібні точні та актуальні карти.

NASA World Wind – це віртуальний глобус, який надає доступ до тривимірних моделей Землі, створених на основі супутникових знімків та інших геопросторових даних. Ця платформа дозволяє користувачам візуалізувати різні типи даних, включаючи топографічні карти, дані про клімат, вегетацію та багато інших. NASA World Wind використовується для навчальних, наукових та інженерних цілей.

MapInfo – це комерційна геоінформаційна система, яка пропонує інструменти для створення карт, управління геопросторовими даними та їх аналізу. Вона використовується в різних галузях, таких як маркетинг, містобудування, транспорт та екологія. MapInfo дозволяє користувачам проводити просторовий аналіз, створювати тематичні карти та звіти.

Кожна з цих ГІС має свої унікальні характеристики та переваги, які роблять їх корисними для вирішення різноманітних задач у різних сферах діяльності.

1.2 Фреймворк Laravel: основні можливості та архітектура

Laravel – це популярний PHP-фреймворк, розроблений для створення вебдодатків з використанням структурованого та виразного синтаксису [2].

Він спрощує процес розробки шляхом використання широкого набору інструментів і бібліотек, що дозволяють швидко та ефективно створювати надійні та масштабовані вебзастосунки. Laravel дотримується принципів MVC (Model-View-Controller), що сприяє розділенню логіки додатка та інтерфейсу користувача, що полегшує підтримку та розвиток програмного забезпечення.

Розглянемо **основні можливості Laravel**.

Структурований синтаксис та інтуїтивний інтерфейс: Laravel забезпечує чистий та структурований синтаксис, який спрощує написання коду. Завдяки чітким та інтуїтивно зрозумілим інструментам, розробники можуть швидко опанувати фреймворк та почати працювати з ним.

ORM Eloquent: Eloquent ORM дозволяє працювати з базами даних, використовуючи об’єктно-орієнтований підхід. Це полегшує роботу з базами даних та дозволяє писати запити SQL на рівні моделей.

Маршрутизація: система маршрутизації Laravel дозволяє легко визначати маршрути для додатка, підтримуючи чисті та зручні URL. Це робить додатки більш організованими та легкими у підтримці.

Система аутентифікації та авторизації: Laravel надає вбудовані засоби завдяки пакету Sanctum для реалізації аутентифікації та авторизації користувачів. Це включає реєстрацію, входи та керування правами доступу до ресурсів додатка.

Система шаблонів Blade: Blade – це потужний шаблонізатор, який дозволяє використовувати зручні та елегантні конструкції для створення динамічних вебсторінок. Blade підтримує наслідування шаблонів, секції та багато іншого.

Міграції та сидування бази даних: міграції дозволяють легко керувати змінами у схемі бази даних, роблячи процес розробки більш контрольованим та організованим. Сидування забезпечує початкове заповнення бази даних тестовими даними.

Механізми кешування: Laravel підтримує різноманітні системи

кешування, такі як Memcached та Redis, що дозволяє значно підвищити продуктивність вебдодатків.

Робота з чергами: Laravel надає потужні інструменти для роботи з чергами, що дозволяє виконувати тривалі процеси у фоновому режимі, забезпечуючи кращу продуктивність та відгук додатка.

Розширення та пакети: Laravel має велику екосистему пакетів та розширень, які дозволяють легко додавати нові функції додатка. Спільнота Laravel активно розробляє та підтримує ці пакети.

Архітектура Laravel. Laravel дотримується архітектурного патерну MVC, що розділяє додаток на три основні компоненти: Модель (Model), Представлення (View) та Контролер (Controller). Це дозволяє розділити логіку додатка, представлення даних та управління ними, що робить код більш організованим та легким у підтримці.

Моделі (Models). Моделі представляють дані додатка та забезпечують взаємодію з базою даних. Вони використовують ORM Eloquent для зручної роботи з базами даних, дозволяючи розробникам оперувати даними за допомогою об’єктів.

Представлення (Views). Представлення відповідають за відображення даних користувачеві. У Laravel для цього використовується шаблонізатор Blade, що дозволяє легко створювати динамічні вебсторінки з використанням зручних та виразних конструкцій.

Контролери (Controllers). Контролери забезпечують взаємодію між моделями та представленнями. Вони обробляють запити користувачів, викликають відповідні моделі та повертають дані у вигляді представлень.

Маршрутизація. Система маршрутизації визначає, які дії слід виконати у відповідь на конкретні URL-запити. Це забезпечує чітке розмежування між URL-адресами та логікою додатка.

Сервісні провайдери (Service Providers). Сервісні провайдери відповідають за завантаження різних компонентів та конфігурацію додатка. Вони є основним місцем для ініціалізації різних сервісів у додатку.

Контейнери залежностей (Dependency Injection Container). Laravel використовує контейнер залежностей для управління залежностями та інверсією управління (IoC). Це дозволяє легко керувати класами та їх залежностями, забезпечуючи більш гнучку та тестовану архітектуру.

Laravel є потужним та гнучким фреймворком для розробки вебдодатків, який надає широкий набір інструментів та бібліотек для спрощення процесу розробки. Використання MVC-архітектури, потужних інструментів для роботи з базами даних, аутентифікації, кешування та іншими аспектами робить його ідеальним вибором для створення сучасних вебзастосунків.

1.3 Фреймворк Vue.js: основні можливості та архітектура

Vue.js – це прогресивний фреймворк для побудови користувацьких інтерфейсів, який став одним із найпопулярніших інструментів для веброзробки завдяки своїй простоті, гнучкості та потужності [1]. Він створений для поступового впровадження, що дозволяє інтегрувати його в наявні проєкти або використовувати як повноцінний фреймворк для розробки односторінкових додатків (SPA). Vue.js пропонує багатий набір можливостей, таких як реактивність, компонентний підхід, зручні директиви та інструменти для розробників, які значно полегшують процес створення вебдодатків.

Розглянемо **основні можливості фреймворку Vue.js.**

Реактивні дані. Vue.js використовує реактивну систему для автоматичного оновлення DOM, коли змінюються дані. Реактивність досягається за допомогою об’єктів, що спостерігаються, і системи сповіщення про зміни. Це означає, що зміни в даних миттєво відображаються у вигляді, без необхідності ручного оновлення інтерфейсу. Використання реактивних даних спрощує управління станом додатка і робить код більш

чистим і зручним для підтримки.

Компонентний підхід. Компоненти є основними будівельними блоками в Vue.js. Вони дозволяють розділити інтерфейс на незалежні, повторно використовувані частини. Кожен компонент має власну логіку, шаблон і стилі, що робить їх легко підтримуваними і тестованими. Компоненти можуть бути вкладені один в одного, обмінюватися даними через властивості і реагувати на події, що дозволяє створювати складні і гнучкі користувацькі інтерфейси.

Одностороннє та двостороннє прив'язування даних. Vue.js пропонує одностороннє прив'язування даних (one-way data binding), що означає, що зміни в моделі автоматично відображаються у вигляді. Двостороннє прив'язування (two-way data binding) корисне для форм, дозволяючи синхронізувати значення форми з даними в моделі в режимі реального часу. Це досягається за допомогою директиви v-model, яка забезпечує простий спосіб управління станом формових елементів.

Вбудовані директиви. Vue.js має набір вбудованих директив, які полегшують маніпулювання DOM. Директиви, такі як v-if, v-for, v-bind і v-on, дозволяють умовно відображати елементи, перебирати масиви, прив'язувати атрибути і обробляти події відповідно. Використання директив робить код шаблону більш читабельним і логічним, забезпечуючи інтуїтивне управління DOM.

Розширюваність та інтеграція. Vue.js можна легко інтегрувати з іншими бібліотеками та проєктами. Він може бути використаний як бібліотека для додавання реактивності в наявні проєкти або як повноцінний фреймворк для розробки нових додатків. Можливість створення плагінів і використання сторонніх модулів робить Vue.js дуже гнучким і зручним для розширення функціональності додатків.

Архітектура Vue.js.

Реактивна система. Vue.js використовує реактивні об'єкти JavaScript, які відслідковують зміни і автоматично оновлюють DOM. Це забезпечує

двосторонню прив’язку даних, що робить розробку більш зручною та ефективною. Важливою особливістю є те, що реактивність працює на рівні об’єкта, дозволяючи відстежувати зміни у вкладених об’єктах та масивах.

Компонентна архітектура. Компоненти є основою Vue.js. Вони інкапсулюють логіку, шаблони і стилі в окремі модулі, що робить код більш структурованим і легким для підтримки. Компоненти можуть взаємодіяти один з одним через властивості (props) і події (events), що дозволяє створювати гнучкі і масштабовані додатки. Ця модульна структура сприяє повторному використанню коду і полегшує його тестування.

Шаблони та компіляція. Vue.js використовує HTML-подібні шаблони для опису користувацького інтерфейсу. Шаблони можуть містити директиви і вирази, що дозволяє легко маніпулювати DOM. Ці шаблони компілюються в функції рендерингу під час виконання або на етапі збірки, що забезпечує високу продуктивність. Компіляція шаблонів в функції рендерингу дозволяє використовувати повну потужність JavaScript для динамічного створення інтерфейсу.

Інструменти розробника. Vue.js пропонує потужні інструменти для розробників, такі як Vue CLI і Vue DevTools. Vue CLI надає зручний інтерфейс для створення, налаштування і управління проєктами, включаючи підтримку сучасних фронтенд-інструментів і можливість легко додавати плагіни. Vue DevTools дозволяє налагоджувати додатки в браузері, надаючи можливості для аналізу стану компонентів, відстеження подій і перегляду історії змін.

1.4 Протоколи та методи передачі даних у вебзастосунків

Вебзастосунки стали невіднятною частиною сучасного Інтернету, забезпечуючи взаємодію користувачів з вебсайтами і сервісами через браузери та мобільні додатки. Одним з ключових аспектів ефективного

функціонування вебзастосунків є передача даних між клієнтом і сервером. У цій главі розглянемо основні протоколи та методи передачі даних, які використовуються у вебзастосунками.

HTTP/HTTPS. HTTP (Hypertext Transfer Protocol) – це основний протокол для передачі гіпертексту (тексту, зображень, відео та інших мультимедійних даних) через Інтернет. HTTP забезпечує клієнт-серверну модель, де браузер (клієнт) відправляє запити до вебсервера та отримує відповіді.

HTTPS (HTTP Secure) – це розширення HTTP, яке використовує протокол SSL/TLS для забезпечення шифрування переданих даних. Це гарантує конфіденційність і цілісність інформації, захищаючи її від несанкціонованого доступу і модифікації під час передачі.

Основні методи HTTP:

– GET: використовується для отримання даних з сервера, наприклад, завантаження вебсторінки;
– POST: використовується для відправлення даних на сервер, застосовується для відправлення форм та завантаження файлів;
– PUT: використовується для оновлення ресурсу на сервері;
– DELETE: видаляє ресурс на сервері.

WebSocket. WebSocket – це протокол зв’язку, який забезпечує двосторонню комунікацію між клієнтом і сервером у реальному часі. На відміну від HTTP, який працює за принципом запит-відповідь, WebSocket дозволяє встановити постійне з’єднання, через яке дані можуть передаватися в обох напрямках безперервно.

WebSocket є ефективним для додатків, що потребують миттєвого обміну даними, таких як чат-додатки, онлайн-ігри та фінансові торгові платформи. Він зменшує накладні витрати на передачу даних і підвищує швидкість комунікації.

AJAX. AJAX (Asynchronous JavaScript and XML) – це метод, що дозволяє вебсторінкам оновлюватися асинхронно, обмінюючись даними з

вебсервером без перезавантаження сторінки. Це забезпечує динамічний користувацький досвід і покращує взаємодію з вебзастосунками.

REST. REST (Representational State Transfer) – це архітектурний стиль для побудови мережевих додатків, який використовує HTTP методи для реалізації CRUD операцій (Create, Read, Update, Delete) над ресурсами, представленими у вигляді URI.

Розглянемо основні принципи REST.

Статусність (Statelessness). Усі запити від клієнта до сервера мають бути незалежними один від одного. Це означає, що кожен запит повинен містити всю інформацію, необхідну для його обробки. Сервер не зберігає жодного контексту між запитами, що спрощує масштабування і підвищує надійність системи.

Уніформальні інтерфейси (Uniform Interfaces). REST використовує стандартні HTTP методи для виконання операцій над ресурсами. Основні методи включають:

– GET: отримання ресурсу;
– POST: створення нового ресурсу;
– PUT: оновлення наявного ресурсу;
– DELETE: видалення ресурсу;
– PATCH: часткове оновлення ресурсу.

Розділення клієнта і сервера (Client-Server Separation). Клієнт і сервер працюють незалежно один від одного. Сервер відповідає за обробку запитів і управління ресурсами, а клієнт – за відображення даних і взаємодію з користувачем. Це дозволяє розвивати клієнтську і серверну частини незалежно, без впливу одна на одну.

Кешування (Caching). RESTful сервіси повинні підтримувати кешування для підвищення продуктивності і зменшення навантаження на сервер. Відповіді сервера можуть містити заголовки, що вказують на те, чи можна кешувати відповідь і на який термін.

Єдиний інтерфейс (Uniform Interface). Усі взаємодії з сервісом повинні

відбуватися через стандартизований інтерфейс. Це забезпечує уніфікований підхід до доступу до ресурсів і спрощує розуміння та використання API.

Шарованість (Layered System). Архітектура системи може бути шарованою, де кожен шар виконує свою роль (наприклад, балансування навантаження, кешування, безпека тощо). Клієнт зазвичай не знає про існування проміжних шарів, що робить систему більш гнучкою і легкою для модифікації.

Код на вимогу (Code on Demand) (необов'язковий). Сервер може передавати клієнту виконуваний код (наприклад, JavaScript), який клієнт може виконати для розширення своєї функціональності. Це дозволяє додатково розширити можливості клієнта без необхідності оновлювати весь додаток.

GraphQL. GraphQL – це мова запитів для API, яка дозволяє клієнтам запитувати тільки ті дані, які їм потрібні. На відміну від REST, де кожен запит повертає фіксовану структуру даних, GraphQL дозволяє клієнту точно вказати, які поля необхідні.

Переваги GraphQL:

– ефективність: зменшує кількість переданих даних і кількість запитів;
– гнучкість: дозволяє отримувати кілька ресурсів у межах одного запиту;
– типізація: використовує сувору типізацію для запитів і відповідей, що допомагає запобігти помилкам.

Узагальнюючи, вебзастосунки використовують різноманітні протоколи та методи для передачі даних, кожен з яких має свої переваги та недоліки. Вибір відповідного інструменту залежить від специфічних вимог проєкту та його архітектури.

2 ПРОЄКТУВАННЯ ТА АНАЛІЗ ВИМОГ

2.1 Аналіз вимог до геоінформаційної системи моніторингу погоди

Геоінформаційна система (ГІС) моніторингу погоди є важливим інструментом для збору, обробки, аналізу та візуалізації даних про погодні умови. У даному контексті додаток створений на основі Laravel та Vue.js, і отримує дані з відкритих API геомоніторингу, таких як OpenMeteo. Система оновлює погоду для всіх міст України кожну годину. Розглянемо детальніше вимоги до такої системи, створивши модель прецедентів системи, котра наведена у рисунку 2.1.

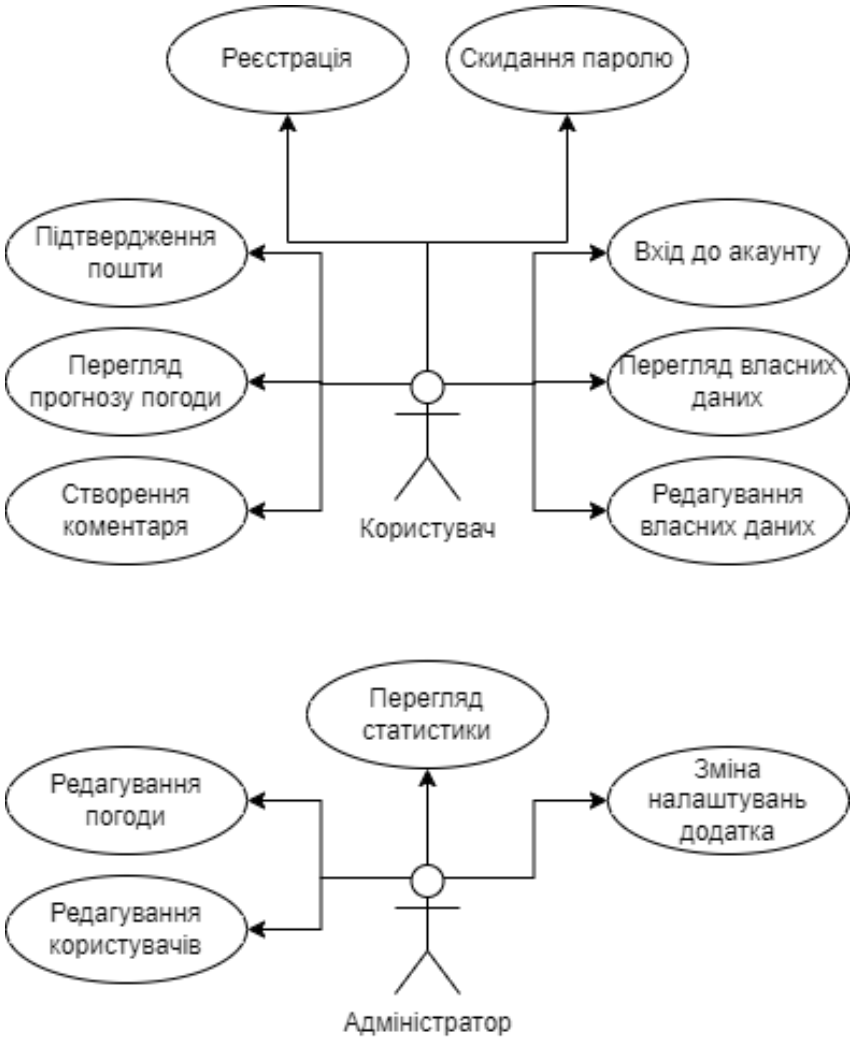

Рисунок 2.1 – Модель прецедентів системи

2.1.1 Функціональні вимоги

Авторизація:

– вхід у систему: можливість користувачам входити в систему за допомогою своїх облікових даних;
– реєстрація: реєстрація нових користувачів із підтвердженням електронної пошти;
– підтвердження електронної пошти: підтвердження електронної пошти для нових користувачів;
– відновлення паролю: відновлення пароля за допомогою електронної пошти.

Профіль:

– редагування персональної інформації: можливість користувачів редагувати свої особисті дані;
– зміна пароля: зміна пароля користувача;
– збереження улюблених місць: можливість користувачам зберігати улюблені місця для швидкого доступу;
– історія переглядів: відображення історії переглянутих місць користувачем.

Головна сторінка:

– карта України з погодою: відображення погодних умов на інтерактивній карті України;
– пошук за назвою місця або координатами: можливість пошуку місць за назвою або географічними координатами;
– відображення погоди вибраного місця: показ погодних умов для вибраного місця;
– підтримка кількох мов: можливість вибору мови інтерфейсу користувачем.

Детальна сторінка погоди:

– опис погоди: надання детального опису погодних умов;

– прогноз погоди: відображення прогнозу погоди на визначений період;
– відгуки про місце: можливість користувачам залишати відгуки про місце;
– графік статистики: відображення погодних даних у вигляді графіків та діаграм.

Адміністративна панель:

– керування користувачами: додавання, редагування, видалення користувачів та можливість їх блокування;
– керування даними про погоду: додавання, редагування, видалення даних про погоду;
– налаштування системи: налаштування частоти оновлення даних та інших параметрів системи.

Зовнішні сервіси:

– додавання API: інтеграція з різними API для отримання даних про погоду;
– інтеграція з соціальними сервісами: можливість інтеграції з соціальними мережами для спільного доступу до даних;
– експорт даних у CSV: можливість експорту даних у форматі CSV для подальшого аналізу.

Додаткова статистика:

– відображення недавно переглянутих місць: відображення списку нещодавно переглянутих місць користувачем;
– рекомендації місць: надання рекомендацій щодо цікавих місць на основі переглянутих даних;
– інформація про небезпечну погоду: відображення інформації про небезпечні погодні умови.

Аналітика:

– статистика активності користувачів: відображення статистики

активності користувачів у системі;
- інтерактивні звіти про погоду: створення інтерактивних звітів про погодні умови;
- генерація звітів про погоду за період: генерація звітів про погодні умови за обраний період.

2.1.2 Нефункціональні вимоги

Продуктивність:
- система повинна обробляти дані в режимі реального часу, забезпечуючи швидке оновлення та відображення інформації;
- оптимізація запитів до API для зменшення затримок.

Надійність:
- система повинна бути стійкою до збоїв і мати механізми для автоматичного відновлення після аварійних ситуацій;
- резервне копіювання даних для запобігання втраті інформації.

Масштабованість:
- система повинна легко масштабуватися для обробки збільшених обсягів даних і підтримки зростаючої кількості користувачів.

Безпека:
- захист даних від несанкціонованого доступу та забезпечення конфіденційності користувачів;
- використання шифрування для передачі чутливих даних.

Зручність використання:
- інтуїтивно зрозумілий інтерфейс користувача для полегшення доступу до даних і функцій системи;
- підтримка багатомовності для забезпечення зручності користувачів з різних країн.

2.1.3 Апаратні вимоги

Сервери:

– потужні сервери для обробки великих обсягів даних і забезпечення високої продуктивності;
– надійна система зберігання даних з резервним копіюванням для забезпечення доступності даних.

Метеостанції та датчики:

– якісні та надійні метеостанції з датчиками для вимірювання різних погодних параметрів (якщо потрібні для додаткових даних).

2.1.4 Програмні вимоги

База даних:

– надійна система управління базами даних (СУБД) для зберігання та обробки великих обсягів метеорологічних даних;
– підтримка реляційних та нереляційних баз даних залежно від типу даних.

Алгоритми та моделі:

– використання перевірених математичних моделей для прогнозування погоди;
– алгоритми машинного навчання для покращення точності прогнозів.

Інтерфейс користувача:

– вебінтерфейс та мобільні додатки для доступу до системи;
– використання сучасних технологій фронтенд-розробки (Vue.js) для забезпечення зручності та швидкодії.

2.2 Моделювання бізнес-процесів та користувацьких сценаріїв

Моделювання бізнес-процесів та користувацьких сценаріїв є важливим етапом при розробці геоінформаційної системи моніторингу погоди. Це дозволяє зрозуміти, як система буде використовуватися в реальних умовах, які задачі вона буде вирішувати та які кроки необхідні для виконання цих задач.

2.2.1 Бізнес-процеси

Збір даних про погоду:

– запит до API OpenMeteo: система відправляє запити до API OpenMeteo для отримання актуальних даних про погоду;
– обробка отриманих даних: отримані дані перевіряються на коректність, проводиться їх валідація та агрегація;
– збереження даних: відфільтровані та перевірені дані зберігаються в базі даних.

Оновлення погодних даних:

– періодичний запит: система автоматично відправляє запити до API кожну годину для оновлення даних про погоду для всіх міст України;
– оновлення бази даних: оновлені дані замінюють попередні у базі даних.

Візуалізація погодних даних:

– побудова інтерактивної карти: на основі отриманих даних система будує інтерактивну карту України з відображенням актуальної погоди в різних містах;
– графіки: для кожного міста можуть бути побудовані графіки, що відображають зміни погодних умов.

Керування користувачами:

– реєстрація та вхід: користувачі реєструються в системі та входять у свої облікові записи;
– редагування профілю: користувачі можуть редагувати свої персональні дані, зберігати улюблені місця та переглядати історію відвідувань.

Адміністрування системи:

– керування користувачами: адміністратори можуть додавати, редагувати та видаляти користувачів, а також блокувати їх;
– налаштування системи: адміністратори можуть змінювати налаштування системи, наприклад, частоту оновлення даних.

2.2.2 Користувацькі сценарії

Сценарій 1. Перегляд актуальної погоди:

– користувач відкриває головну сторінку додатка;
– на інтерактивній карті України користувач обирає місто;
– система відображає актуальні погодні умови для вибраного міста.

Сценарій 2. Пошук місця за назвою або координатами:

– користувач вводить назву міста або координати у пошуковий рядок;
– система здійснює пошук і відображає результати;
– користувач обирає потрібне місце зі списку результатів;
– система відображає погодні умови для вибраного місця.

Сценарій 3. Реєстрація нового користувача:

– користувач переходить на сторінку реєстрації;
– заповнює форму реєстрації та відправляє її;
– система надсилає листа для підтвердження електронної пошти;
– користувач переходить за посиланням у листі для підтвердження;
– реєстрація успішно завершена, користувач може входити в систему.

Сценарій 4. Збереження улюбленого місця:

– користувач входить у свій обліковий запис;
– переглядає погодні умови для певного міста;
– натискає кнопку для збереження міста як улюбленого;
– система зберігає місто у списку улюблених місць користувача.

Сценарій 5. Адміністрування користувачів:

– адміністратор входить у систему;
– переходить до адміністративної панелі;
– вибирає розділ «Користувачі»;
– додає, редагує або видаляє користувача, а також може заблокувати користувача.

2.3 Проєктування архітектури системи

Проєктування архітектури системи є ключовим етапом у розробці геоінформаційної системи (ГІС) моніторингу погоди. Це дозволяє визначити структуру програмного забезпечення, розподіл функцій та взаємозв’язки між їх компонентами. Розглянемо, що включають основні аспекти архітектури такої системи.

Модульна структура. Система буде розділена на модулі з чіткою відповідальністю за певні функції. Основні модулі включають:

– модуль збору даних: відповідає за взаємодію з API OpenMeteo для отримання погодних даних;
– модуль обробки даних: включає логіку для валідації, обробки та агрегації отриманих даних перед їх збереженням у базі даних;
– модуль збереження даних: відповідає за взаємодію з базою даних для зберігання оновлених погодних даних;
– модуль візуалізації: забезпечує побудову інтерактивної карти та відображення погоди для користувачів.

Компоненти системи. Компоненти системи включають:

– вебсервер: надає інтерфейс для взаємодії з користувачами через вебдодаток;
– база даних: використовується для зберігання погодних даних, інформації про користувачів та іншої необхідної інформації;
– API сервіс: для взаємодії з зовнішніми сервісами, наприклад, OpenMeteo API;
– адміністративна панель: інтерфейс для адміністраторів для керування користувачами, даними погоди та налаштуваннями системи.

Технологічні вибори:

– фронтенд: Vue.js для створення інтерактивного інтерфейсу користувача;
– бекенд: Laravel для створення API, обробки бізнес-логіки та взаємодії з базою даних;
– база даних: MySQL для зберігання структурованої інформації про погоду, користувачів та інші дані;

Забезпечення безпеки. Важливою частиною архітектури є забезпечення безпеки системи:

– аутентифікація та авторизація: використання механізмів аутентифікації та авторизації для захисту доступу до персональних даних користувачів та адміністративних функцій;
– шифрування даних: захист конфіденційної інформації, такої як паролі користувачів і ключі API, шляхом використання шифрування;
– масштабованість та продуктивність: горизонтальне масштабування: можливість розширення системи шляхом додавання нових серверів або хмарних ресурсів для забезпечення продуктивності при зростанні обсягів даних та користувацького трафіку.

2.4 Проєктування бази даних

При проєктуванні бази даних для геоінформаційної системи моніторингу погоди, необхідно уважно розглянути всі аспекти зберігання, організації та доступу до даних. Ця система призначена для збору, аналізу та візуалізації інформації про погоду у різних місцях України. Важливо враховувати потреби користувачів у точній та швидкій інформації про погоду, а також забезпечення надійного зберігання та обробки даних.

Основні положення при проєктуванні бази даних

Нормалізація даних: база даних повинна бути нормалізованою для уникнення дублювання даних та забезпечення їх цілісності. Це означає, що кожен стовпець в таблиці має містити лише одне значення, а кожен рядок має бути унікальним.

Зв'язки між таблицями: використовуючи зовнішні ключі, створюються зв'язки між таблицями. Наприклад, зв'язок між таблицями users і favourite_locations дозволяє зв'язувати улюблені місця користувачів з їх профілями.

Індексація: для швидкого доступу до даних важливо створювати відповідні індекси на часто використовувані стовпці. Наприклад, індекс на стовпці email в таблиці users дозволяє швидко здійснювати пошук користувачів за email.

Оптимізація запитів: під час проєктування важливо враховувати типи запитів, які будуть виконуватися над базою даних, та оптимізувати структуру даних для швидкості та ефективності виконання.

Реляції між таблицями:

– один до багатьох (One-to-Many): наприклад, в таблиці locations можуть бути відносини з таблицею weather, де кожен запис погоди пов'язаний з одним місцем, але одне місце може мати багато записів погоди;

– багато до багатьох (Many-to-Many): наприклад, таблиця users може мати відношення до таблиці roles через таблицю model_has_roles, де кожен користувач може мати багато ролей, а кожна роль може бути пов’язана з багатьма користувачами;
– один до одного (One-to-One): наприклад, таблиця sessions може мати відношення до таблиці users, де кожна сесія пов’язана з одним користувачем, а кожен користувач може мати одну сесію.

Ця структура бази даних забезпечує гнучкість, ефективність та надійність для зберігання та обробки даних про погоду і відносини між користувачами, місцями та іншими сутностями додатка моніторингу погоди.

Повна структура бази даних наведена у рисунку 2.2 у вигляді ER-діаграми.

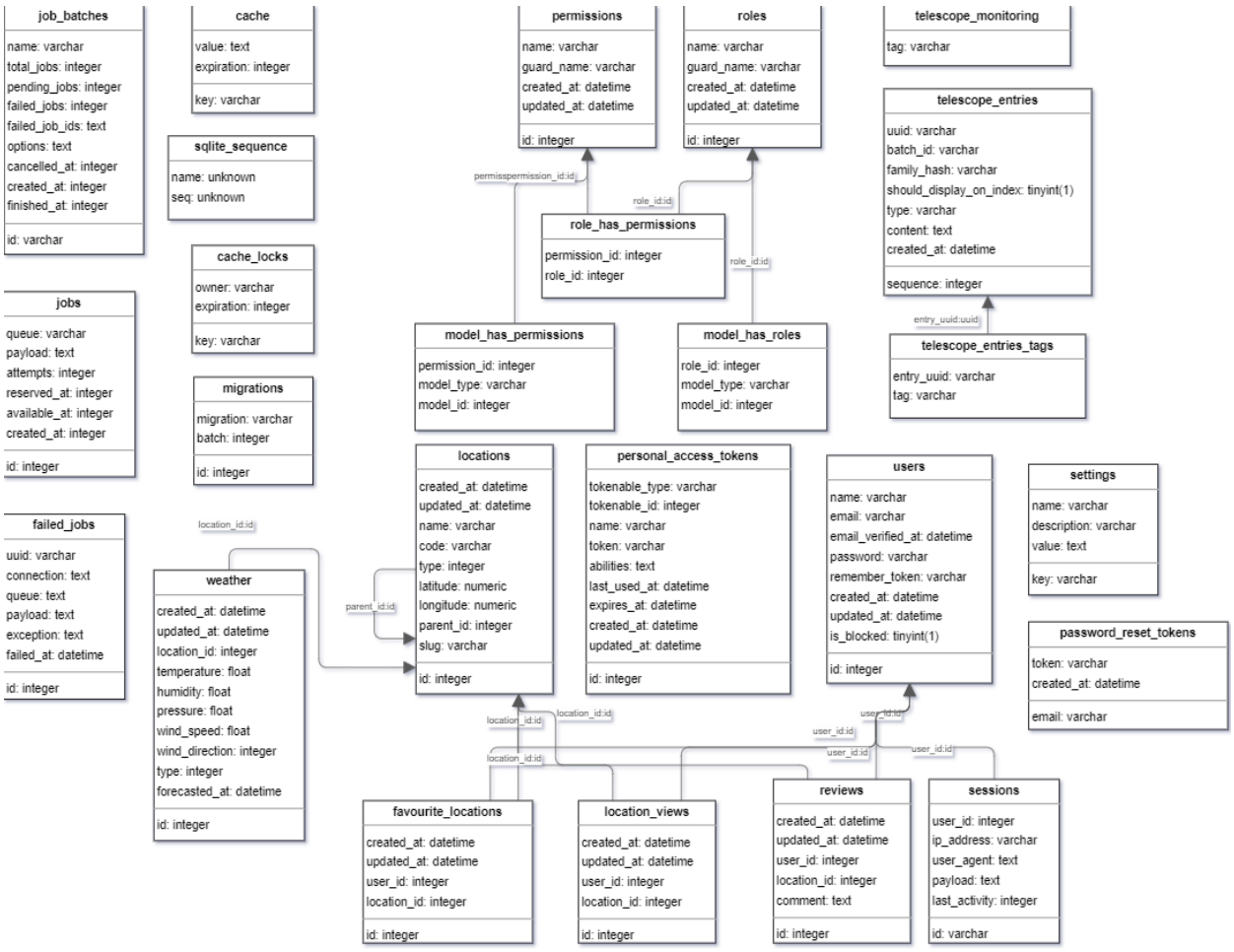

Рисунок 2.2 – ER діаграма бази даних

2.5 Вибір інструментів та технологій для розробки

При розробці геоінформаційної системи моніторингу погоди було обрано широкий спектр інструментів і технологій, що забезпечують ефективність, масштабованість і зручність розробки [6].

Backend (серверна частина).

Laravel:

– основна функціональність: Laravel обрано для створення монолітного додатка, оскільки цей фреймворк забезпечує зручність у роботі з маршрутизацією, ORM для роботи з базою даних, системою міграцій та шаблонізацією;
– пакети Laravel: використано пакети, такі як laravel/sanctum для автентифікації, laravel/telescope для відладки та профілювання, spatie/laravel-permission для керування дозволами користувачів.

База даних:

– в якості бази даних використовується MySQL, оскільки вона надійна, масштабована та підтримується Laravel з використанням Eloquent ORM [9];
– структура бази даних включає таблиці для користувачів, місць, погодних умов, сесій та інших важливих сутностей.

Frontend (клієнтська частина).

Vue.js:

– фреймворк: Vue.js використовується для побудови інтерактивного та динамічного інтерфейсу користувача [3];
– екосистема Vue: для створення компонентів та управління станом використовуються бібліотеки, такі як pinia для глобального стану та vue-router для маршрутизації.

Фронтенд бібліотеки та інструменти:

– Tailwind CSS: використовується для швидкого створення стильового оформлення без використання традиційних CSS-класів;

– Axios: для здійснення HTTP-запитів з фронтенду до сервера;
– Vue-i18n: для локалізації інтерфейсу користувача.

Інші інструменти та технології.

IDE і відладка: розробка виконується у PHPStorm, який забезпечує потужність та зручність для роботи з PHP та JavaScript. Відладка виконується з використанням Laravel Telescope та XDebug, ці інструменти значно прискорюють розробку.

Управління залежностями: використано Composer для PHP-залежностей та npm для JavaScript-залежностей.

Цей набір інструментів і технологій дозволяє забезпечити високу якість та ефективність розробки геоінформаційної системи моніторингу погоди з урахуванням потреб користувачів та бізнесових вимог проєкту.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ГЕОІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ ПОГОДИ

3.1 Реалізація серверної частини застосунку за допомогою Laravel

Розробка серверної частини геоінформаційної системи погоди на базі Laravel містить кілька ключових аспектів, які спрямовані на забезпечення ефективного збору, зберігання та представлення даних.

На рисунку 3.1 наведені основні класи додатка у вигляді моделі класів.

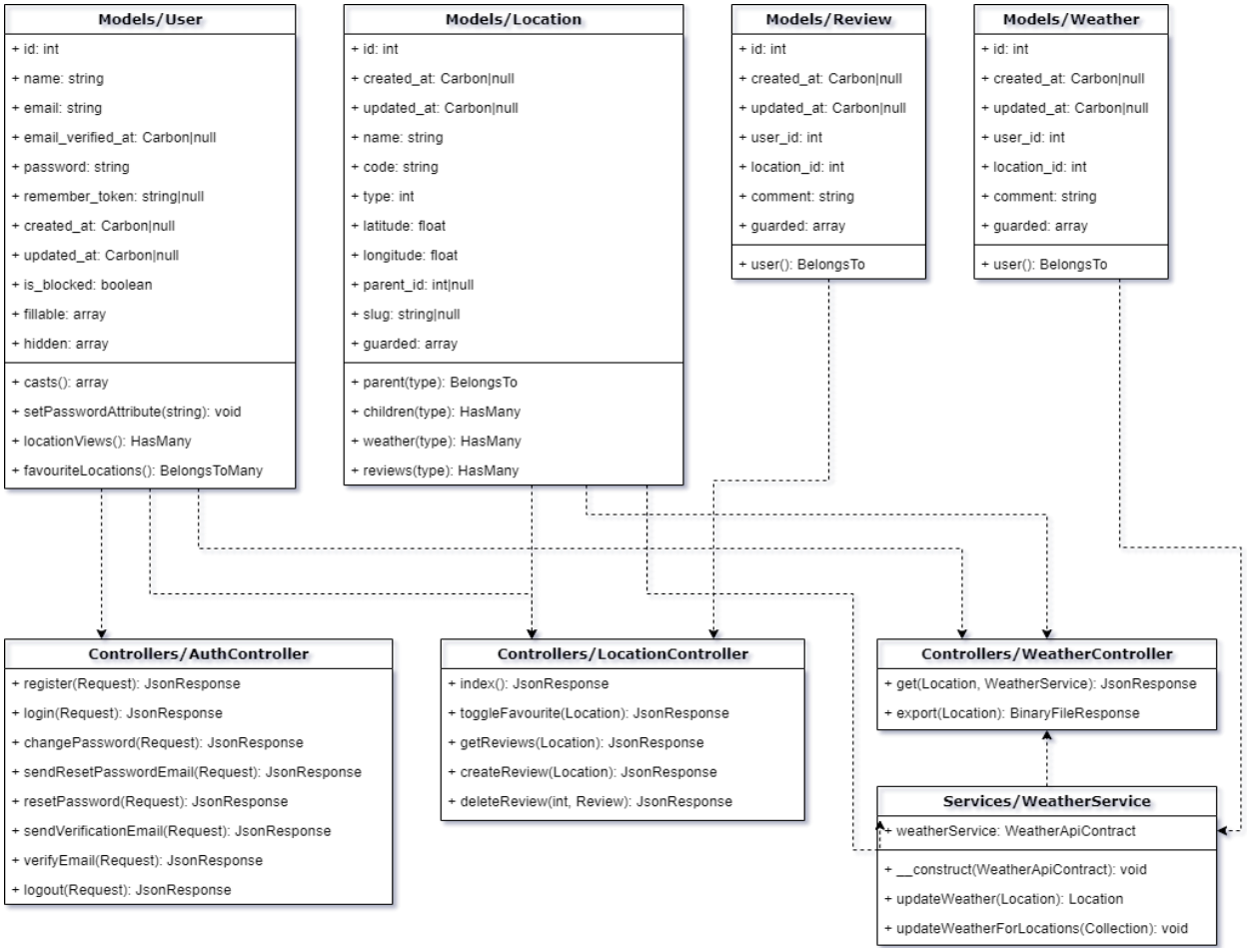

Рисунок 3.1 – Модель класів у додатку

Збір та обробка даних про погоду. Одна з основних частин додатка це збір інформації стану погоди в усіх областях та містах України, для цього було створено WeatherService з наступними методами:

– updateWeather – використовується для оновлення погоди в одній локації, тобто в області чи місті.
– updateWeatherForLocations – оновлює погоди у декількох локаціях, робить запит одразу з масивом координат.

Кожен з методів робить запит до сервісу OpenMeteo через WeatherApi котрий був інжектований у додаток через DI (Dependency Injection), після цього WeatherService повертає DTO котрі використовуються замість масивів, тому що масив може містити невизначені ключі та дані. Далі ці дані зберігаються до бази даних, або оновлюються якщо дати вже існують.

Отримання даних. Для отримання даних у фреймворку Laravel використовуються контролери. Щоб отримати прогноз погоди з бази – використовується контролер WeatherController, а саме метод get, у рисунку 3.2 наведено код даного методу.

```
public function get(Location $location, WeatherService $weatherService): JsonResponse
{
	$weather = $location
		->weather()
		->whereDate('forecasted_at', '>=', now()->startOfDay())
		->where('forecasted_at', '<=', now()->addDays(7)->endOfDay())
		->get();

		/** @var User|null $user */
		$user = \Auth::guard('sanctum')->user();

		if ($user && $user->locationViews()->orderBy('created_at', 'desc')->first()?->location_id !==
$location->id) {
			$user->locationViews()->create(['location_id' => $location->id]);

			$user->locationViews()
				->where('created_at', '<', now()->subDays(7))
				->delete();
		}

if ($weather->count() < 168) {
$weatherService->updateWeather($location);

$weather = $location
->weather()
->whereDate('forecasted_at', '>=', now()->startOfDay())
->where('forecasted_at', '<=', now()->addDays(7)->endOfDay())
				->orderBy('forecasted_at')
->get();
}

return response()->json(['weather' => $weather]);
}
```

Рисунок 3.2 – Код контролеру WeatherController

Основні частини метода – це отримання даних, запис перегляду користувача, перевірка актуальності даних та повертання даних у JSON форматі.

Аутентифікація. Інший не менш важливий контролер це AuthController, саме він контролює аутентифікацію користувачів. Він має наступні методи:

- register – відповідає за реєстрацію користувачів, тобто перевіряє введені дані та додає нового користувача у базу (в додаток до цього відправляє повідомлення на пошту користувача з приводу того що реєстрація була вдала);
- login – відповідає за вхід користувачів до додатка, а саме за їх аутентифікацію (також перевіряє дані перед їх обробкою);
- changePassword – відповідає за зміну пароля користувача, для цього перевіряє наявний пароль завдяки хешу (при успіху – робить запит до бази даних та змінює його);
- sendResetPasswordEmail – відправляє лист для скидання пароля користувачеві при його наявності у базі (лист містить тимчасове посилання);
- resetPassword – перевіряє токен що був отриманий у якості запит параметра, якщо він збігся – користувач отримує можливість змінити пароль;
- sendVerificationEmail – відправляє лист для перевірки пошти з тимчасовим посиланням з токеном;
- verifyEmail – перевіряє отриманий токен, якщо він коректний – оновлює користувача у базі даних та позначає його пошту як верифіковану;
- logout – видаляє поточну сесію користувача.

3.2 Реалізація клієнтської частини застосунку за допомогою Vue.js

Для створення клієнтської частини застосунку був використаний фреймворк Vue.js третьої версії. Основні відмінності від другої версії наступні:

- нова форма написання коду логіки – на заміну Options API було додано нову Composition API, яка дозволила більш логічно групувати код та зручно структурувати його;
- більш оптимізовані засоби для контролю реактивності – у новій версії використовуються нативні Proxy об’єкти з JavaScript що значно прискорило роботу фреймворку, щобільше – була виправлена проблема відсутності реактивності при багато-рівневих об’єктах [8];
- різноманітні додатки та оптимізація – загалом фреймворк значно прискорився. Також були додані корисні компоненти, наприклад компонент Teleport.

TypeScript. Для того щоб зробити додаток більш надійним – був використаний TypeScript. Через те що JavaScript це динамічно типізована мова програмування можуть виникати ситуації коли невідомо які властивості тримає в собі об’єкт – це одна з проблем яку вирішує TypeScript. При цьому для користувача нічого не зміниться бо це мова яка буде скомпільована у звичайний JavaScript.

Додаткові бібліотеки. Для прискорення розробки використовуються різноманітні готові плагіни чи бібліотеки котрі розв’язують поширені проблеми. В реалізації додатка було використано деякі поширені бібліотеки. Розглянемо їх детальніше.

Axios – плагін для виконання запитів до серверної частини додатка. Він забезпечує налаштування заголовків, використання аутентифікації, скасування запитів та багато іншого. Axios спрощує роботу з HTTP-запитами, дозволяючи легко виконувати GET, POST, PUT, DELETE та інші

запити до API. Крім того, він підтримує асинхронні операції, що допомагає зберігати інтерактивність користувацького інтерфейсу.

Tailwindcss – бібліотека для спрощеного використання CSS стилів. Вона дозволяє швидко прототипувати та змінювати дизайн вебдодатку, що значно прискорює розробку. TailwindCSS використовує утилітарний підхід до CSS, що означає створення інтерфейсу за допомогою невеликих класів, які безпосередньо застосовуються до HTML елементів. Це зменшує кількість написаного коду та спрощує підтримку стилів.

Pinia – плагін для створення сховищ даних, які можуть бути доступні між сторінками та компонентами. Pinia надає реактивне сховище, що дозволяє легко керувати станом додатка, роблячи його доступним у будь-якій частині додатка. Це полегшує розробку складних додатків, де необхідно спільне використання стану між різними компонентами.

Primevue – велика бібліотека готових компонентів, пропонує широкий вибір компонентів для відображення або введення інформації, має вбудовані стилі, але дозволяє інтеграцію з tailwindcss, завдяки чому можна додати власні стилі кожному із компонентів.

Vue-router – одна з основних бібліотек, що контролює, яка сторінка відображається при певному URL. Vue-router забезпечує навігацію між сторінками додатка [5], дозволяючи створювати маршрути та компоненти, які відповідають за їх відображення. Це дозволяє легко організувати структуру додатка та забезпечує плавний перехід між різними розділами.

Vue-i18n – бібліотека для інтерналізації додатка. Вона дозволяє перекладати додаток на будь-яку кількість мов, що робить його доступним для користувачів з різних країн. Vue-i18n забезпечує простий інтерфейс для керування перекладами, дозволяючи розробникам легко додавати нові мови та змінювати існуючі переклади без необхідності змінювати код додатка.

Echars – потужна бібліотека для створення інтерактивних графіків та діаграм. Вона підтримує різні типи візуалізацій, такі як лінійні, стовпчикові, кругові, гістограми, карти, теплові карти та багато інших. ECharts забезпечує високу інтерактивність графіків, дозволяючи користувачам взаємодіяти з

графіками шляхом наведення курсора, масштабування, панорамування або фільтрації даних [4]. Бібліотека пропонує безліч опцій для налаштування зовнішнього вигляду та поведінки графіків, що дозволяє змінювати кольори, шрифти, розміри, а також налаштовувати анімації та ефекти. ECharts розроблена для обробки великих обсягів даних без втрати продуктивності завдяки використанню WebGL для рендерингу графіків, що дозволяє використовувати потужність графічного процесора. Графіки, створені за допомогою ECharts, підтримують мобільні пристрої, забезпечуючи гарну адаптивність та зручність використання.

3.3 Інтеграція компонентів та налаштування системи

Основна частина інтеграції компонентів та налаштування системи у Vue проєкті виконується в файлі app.ts. Цей файл відповідає за створення додатка, підключення різних плагінів та налаштувань, а також за ініціалізацію головних компонентів. Нижче наведений список компонентів та плагінів, використаних у проєкті.

Проєкт організовано за наступною структурою:

а) components/admin: містить компоненти, які використовуються в адміністративній частині додатка:
   1) UsersChart.vue: компонент для відображення діаграми користувачів;
   2) DailyForecastBlock.vue: компонент для відображення щоденного прогнозу погоди;
   3) HourlyForecastBlock.vue: компонент для відображення погодинного прогнозу погоди;
   4) HourlyForecastChart.vue: компонент для відображення діаграми погодинного прогнозу;
   5) LoadingSpinner.vue: компонент для відображення спінера завантаження;

6) ReviewsBlock.vue: компонент для відображення відгуків;
7) TemperatureBar.vue: компонент для відображення температурного графіка;
8) UkraineMap.vue: компонент для відображення мапи України з погодою;
9) WeatherIcon.vue: компонент для відображення іконки погоди;
10) WeatherName.vue: компонент для відображення назви погоди;

б) components/pages: містить основні сторінки додатка:
1) IndexPage.vue: головна сторінка з картою України та погодою;
2) admin: містить сторінки для адміністрування:
– IndexPage.vue: сторінка зі статистикою додатка;
– SettingsPage.vue: сторінка налаштувань;
– UsersPage.vue: сторінка з користувачами;
– WeatherPage.vue: сторінка з погодою;
3) auth: містить сторінки для аутентифікації:
– ForgotPasswordPage.vue: сторінка для відновлення пароля;
– LoginPage.vue: сторінка входу;
– LogoutPage.vue: сторінка виходу;
– RegisterPage.vue: сторінка реєстрації;
– ResetPasswordPage.vue: сторінка для скидання пароля;
– VerifyEmailPage.vue: сторінка для верифікації email;
4) ForecastPage.vue: сторінка прогнозу погоди;
5) ProfilePage.vue: сторінка профілю користувача;

в) plugins: містить плагіни, які використовуються у проєкті:
1) primevue: налаштування та інтеграція бібліотеки PrimeVue;
2) api.ts: налаштування API;
3) router.ts: налаштування маршрутизації;
4) echarts.ts: налаштування ECharts;

г) stores: містить сховища даних (store) для керування станом додатка:

1) locationsStore.ts: сховище даних для локацій;
2) userStore.ts: сховище даних для користувачів.

Для ініціалізації проєкту створено файл app.ts, де прописані всі необхідні плагіни, стилі та сховища, код цього файлу надано у рисунку 3.3.

```
import { createApp } from 'vue';
import App from '@/App.vue';

import { createPinia } from 'pinia';
import piniaPluginPersistedState from 'pinia-plugin-persistedstate';

import primeVue from 'primevue/config';
import primeVueTheme from '@/plugins/primevue';
import ToastService from 'primevue/toastservice';
import Tooltip from 'primevue/tooltip';

import '@/plugins/echarts';

import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome';
import '@/plugins/icons';

import router from '@/plugins/router';

import '@/assets/css/app.css';

import { createI18n } from 'vue-i18n';

import * as en from '@/assets/lang/en.json';
import * as uk from '@/assets/lang/uk.json';
import * as ja from '@/assets/lang/ja.json';

const app = createApp(App);

const pinia = createPinia();

pinia.use(piniaPluginPersistedState);

const i18n = createI18n({
	legacy: false, // you must set `false`, to use Composition API
	locale: 'uk',
	fallbackLocale: 'en',
	messages: {
		en,
		uk,
		ja,
	},
});



app.use(pinia)
.use(primeVue, { unstyled: true, pt: primeVueTheme })
.use(ToastService)
.use(router)
	.use(i18n)
	.directive('tooltip', Tooltip)
.component('font-awesome-icon', FontAwesomeIcon)
.mount('#app');
```

Рисунок 3.3 – Код файлу app.ts

3.4 Тестування функціональності та продуктивності

Тестування є важливою частиною процесу розробки, оскільки воно допомагає забезпечити якість та стабільність додатка. В цьому проєкті було сфокусовано на тестуванні функціональності та продуктивності, щоб переконатися, що всі компоненти працюють правильно та ефективно під навантаженням. Нижче описані підходи та інструменти, які використовуються для тестування Vue додатка [7].

Тестування функціональності. Тестування функціональності полягає у перевірці коректності роботи додатка відповідно до вимог та специфікацій. Для цього використовуються різні види тестування:

- юніт-тестування: перевіряє окремі компоненти або функції – для цього використовуються бібліотеки, такі як Vitest та Vue Test Utils;
- інтеграційне тестування: перевіряє взаємодію між кількома компонентами – це дозволяє виявити проблеми, які можуть виникнути при спільній роботі різних частин додатка;
- End-to-End тестування (E2E): перевіряє додаток в цілому, імітуючі дії користувачів – для цього використовуються інструменти, такі як Cypress або Selenium.

Тестування продуктивності. Тестування продуктивності полягає у перевірці, як додаток поводиться під навантаженням. Це включає вимірювання часу завантаження сторінок, швидкості рендерингу компонентів, а також аналіз використання ресурсів (CPU, пам’ять).

Лоад-тестування: перевіряє, як додаток працює під високим навантаженням. Інструменти, такі як Apache JMeter або k6, допомагають симулювати велику кількість користувачів та виміряти продуктивність [10].

Профілювання: аналізує використання ресурсів додатком для виявлення «вузьких місць». Браузери, такі як Chrome та Firefox, мають вбудовані інструменти для профілювання.

Моніторинг: безперервно відстежує продуктивність додатка у

реальному часі. Інструменти, такі як New Relic або Grafana, допомагають відстежувати метрики продуктивності та виявляти потенційні проблеми.

Приклад лоад-тесту за допомогою k6 наведено у рисунку 3.4.

```
import http from 'k6/http';
import { check, sleep } from 'k6';

export const options = {
vus: 100,
duration: '30s',
};

export default function () {
const res = http.get('https://weather.hexdev.me');
check(res, {
'status was 200': (r) => r.status === 200,
});
sleep(1);
}
```

Рисунок 3.4 – Код для тестування додатку

Інтеграція тестів у CI/CD. Для забезпечення стабільності розробки тести інтегруються у процес безперервної інтеграції та доставлення (CI/CD). Це дозволяє автоматично запускати тести при кожному коміті або перед розгортанням на продакшн:

- GitHub Actions: використовується для автоматичного запуску тестів при кожному пуші у репозиторій;
- Jenkins: встановлює тести у пайплайні розгортання додатка;
- GitLab CI: забезпечує автоматизацію процесу тестування та розгортання.

3.5 Виправлення помилок та оптимізація коду

В процесі розробки вебдодатків важливо не лише створити функціональний продукт, але й забезпечити його стабільність, ефективність та високу продуктивність [11]. Виправлення помилок та оптимізація коду є невіддільною частиною цього процесу, допомагаючи підтримувати якість

додатка та покращувати досвід користувачів. Нижче наведено підходи та методики, що використовуються для виправлення помилок та оптимізації коду у проєкті.

Виправлення помилок. Виправлення помилок передбачає виявлення та усунення дефектів, які можуть впливати на функціональність або продуктивність додатка. Основні кроки включають:

– логування: запис логів допомагає відстежувати події в додатку та виявляти помилки – у Laravel для цього використовуються вбудовані можливості логування;
– налагодження (debugging): використання інструментів налагодження, таких як Laravel Debugbar, дозволяє відстежувати виконання коду та виявляти проблеми;
– автоматичне тестування: регулярне виконання юніт-тестів та інтеграційних тестів допомагає виявляти помилки на ранніх стадіях розробки;
– рецензування коду: перегляд коду іншими розробниками (code review) дозволяє виявити потенційні проблеми та покращити якість коду.

Приклад логування помилок у Laravel відображено у рисунку 3.5.

```
try {
// Код, що може викликати помилку
} catch (Exception $e) {
Log::error('Помилка: ' . $e->getMessage());
}
```

Рисунок 3.5 – Код для логування помилки

Оптимізація коду. Оптимізація коду передбачає покращення його ефективності та продуктивності. Основні методи включають:

– оптимізація запитів до бази даних: використання ефективних запитів, мінімізація кількості запитів та кешування результатів,

також для найбільш ефективного користування базою даних слід використовувати Eager Loading, що дозволяє отримувати об’єкт разом з його реляцією – це розв’язує проблему N + 1 запитів;

– мінімізація завантаження сторінок: використання таких інструментів, як Vite, для мінімізації CSS та JavaScript файлів, оптимізації зображень;
– видалення зайвого коду: регулярне очищення проєкту від неактуального або невикористаного коду;
– профілювання продуктивності: використання інструментів, таких як Xdebug, для виявлення вузьких місць у продуктивності коду;
– використання кешування: застосування кешування для зменшення навантаження на сервер та прискорення відповіді додатка.

Приклади не оптимізованого та оптимізованого коду наведено на рисунку 3.6.

```
// Неоптимізований запит
$users = User::all();
foreach ($users as $user) {
$posts = $user->posts;
}

// Оптимізований запит з використанням жадібного завантаження
$users = User::with('posts')->get();
```

Рисунок 3.6 – Приклади повільного та оптимізованих запитів

Виправлення помилок та оптимізація коду є критично важливими етапами розробки, що дозволяють забезпечити високу якість, стабільність та продуктивність вебдодатку.

Використання сучасних інструментів та підходів допомагає виявляти та усувати проблеми на ранніх стадіях, забезпечуючи безперебійний та ефективний процес розробки.

3.6 Готовий проєкт

Після вдалого деплою проєкту на сервер отримуємо повністю робочий додаток для аналізу погоди.

Головна сторінка. При вході до додатку користувач потрапляє на головну сторінку. На рисунку 3.7 зображено головну сторінку додатка.

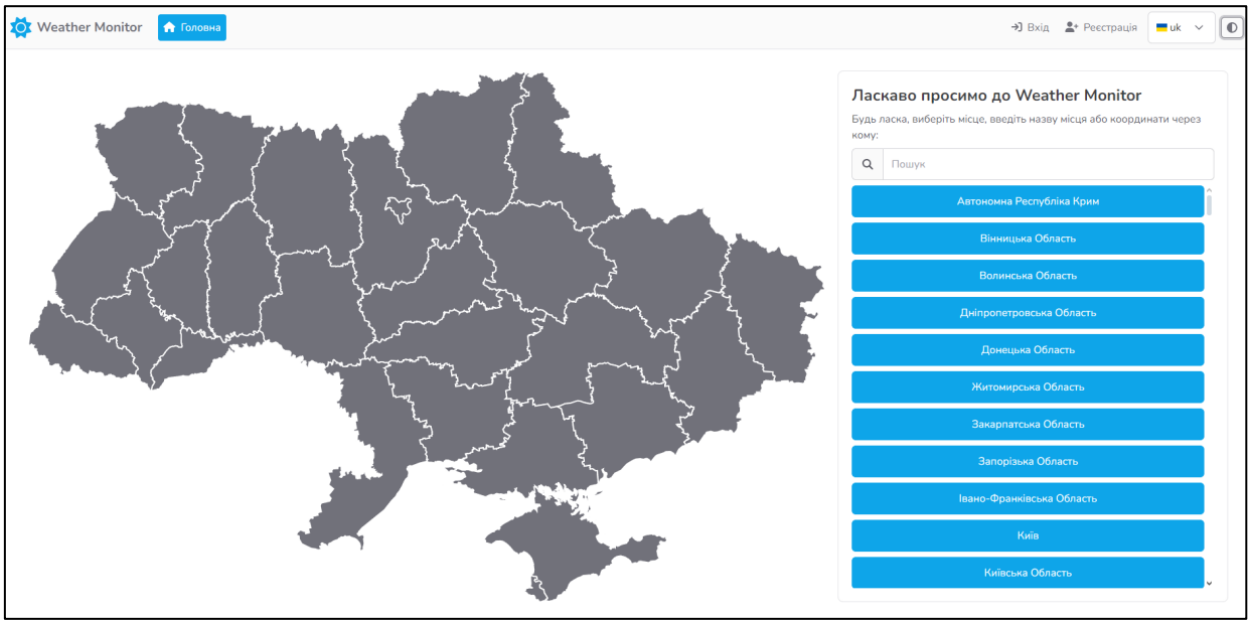

Рисунок 3.7 – Головна сторінка додатка

На головній сторінці користувач бачить інтерактивну мапу України, на кожну область користувач може натиснути щоб перейти на сторінку з погодою.

Для спрощення пошуку міст також є стовпець зі списком областей та міст з пошуком – це дозволяє користувачеві ввести назву бажаного міста чи координати.

Сторінка прогнозу погоди. Після того як користувач обрав бажане місце – з’являється сторінка з прогнозом погоди на 7 днів. Сторінка наведена у рисунку 3.8. Також доступні такі параметри як:

– швидкість повітря;

– напрямок повітря;

– тиск;
– вологість;
– мінімальна температура за день;
– максимальна температура за день.

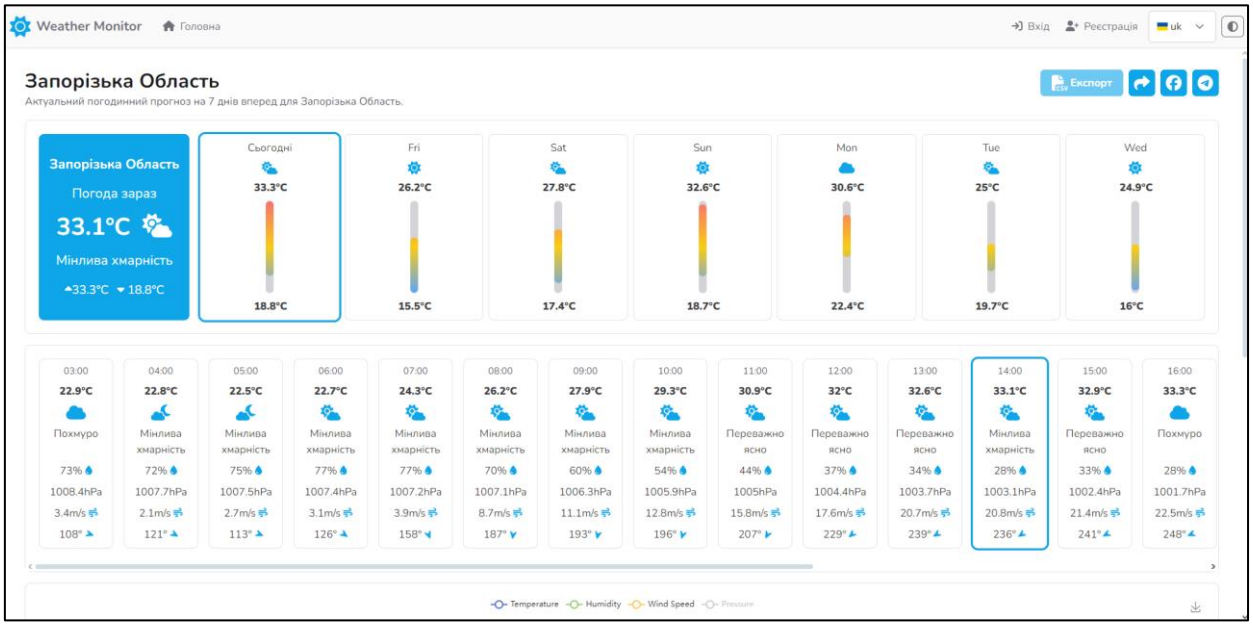

Рисунок 3.8 – Верхня частина сторінки прогнозу погоди

Також користувачеві доступний графік змін усіх даних за весь тиждень. Під графіком розташований блок відгуків міста де зареєстровані користувачі можуть створювати свої коментарі.

Графік та блок відгуків наведені у рисунку 3.9.

Рисунок 3.9 – Нижня частина сторінки прогнозу погоди

На всіх сторінках зверху завжди доступна панель керування поточним положенням у додатку. Якщо кнопка котра відповідає за сторінку котра зараз відчинена має такий ж URL що і поточний URL у браузері – ця кнопка буде підсвічена іншим кольором.

Також на цій панелі для неаутентифікованого користувача знаходяться кнопки для переходу на сторінки логіну та реєстрації. У випадку, якщо користувач вже попередньо виконав логін у додаток – для нього буде доступна сторінка профіля.

Профіль користувача. На сторінці профілю користувач може побачити власні дані котрі він вказав для реєстрації, ці дані можливо змінити. Також доступна зміна паролю.

Правіше від цих даних розташовано два блоки – перший відповідає за недавно переглянуті місця які додаються автоматично при перегляді та другий котрий зберігає улюблені місця. При їх наявності – місця можна відкрити для перегляду прогнозу погоди.

Сторінка наведена у рисунку 3.10.

Рисунок 3.10 – Сторінка профіля користувача

Адміністрування. Якщо користувач має роль адміністратора – на верхній панелі стануть доступні сторінки для керування додатком. Верхня панель наведена у рисунку 3.11.

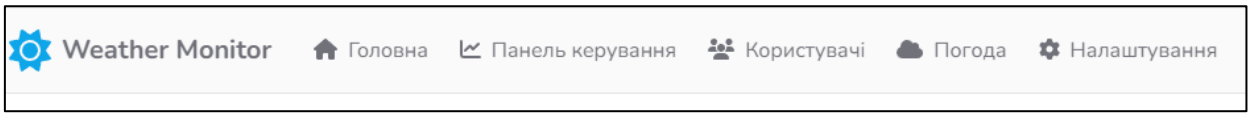

Рисунок 3.11 – Верхня панель навігації для адміністраторів

Адміністрування – головна сторінка. На головній сторінці адміністратор може побачити статистику додатку а саме: кількість поточних користувачів, кількість місць у системі, кількість коментарів та кількість погодних записів. Сторінка наведена у рисунку 3.12.

Рисунок 3.12 – Головна сторінка адміністратора

Адміністрування – керування користувачами. На сторінці керування користувачами адміністратор може переглянути список наявних користувачів у системі, їх дані та є можливість створювати, редагувати, блокувати та видаляти користувачів, тобто CRUD дії. Сторінка наведена у рисунку 3.13.

Адміністратор – керування погодою. Ця сторінка також як і сторінка користувачів дозволяє виконувати CRUD дії, але вже над погодою. Система автоматично додає погоду до системи, але якщо є необхідність – записи можна корегувати. Сторінка наведена у рисунку 3.14.

Адміністратор – налаштування додатка. Ця сторінка дозволяє контролювати налаштування додатка, наприклад частоту оновлення

прогнозів, URL для API погоди та ключ для сервісу обробки місць. Сторінка наведена у рисунку 3.15.

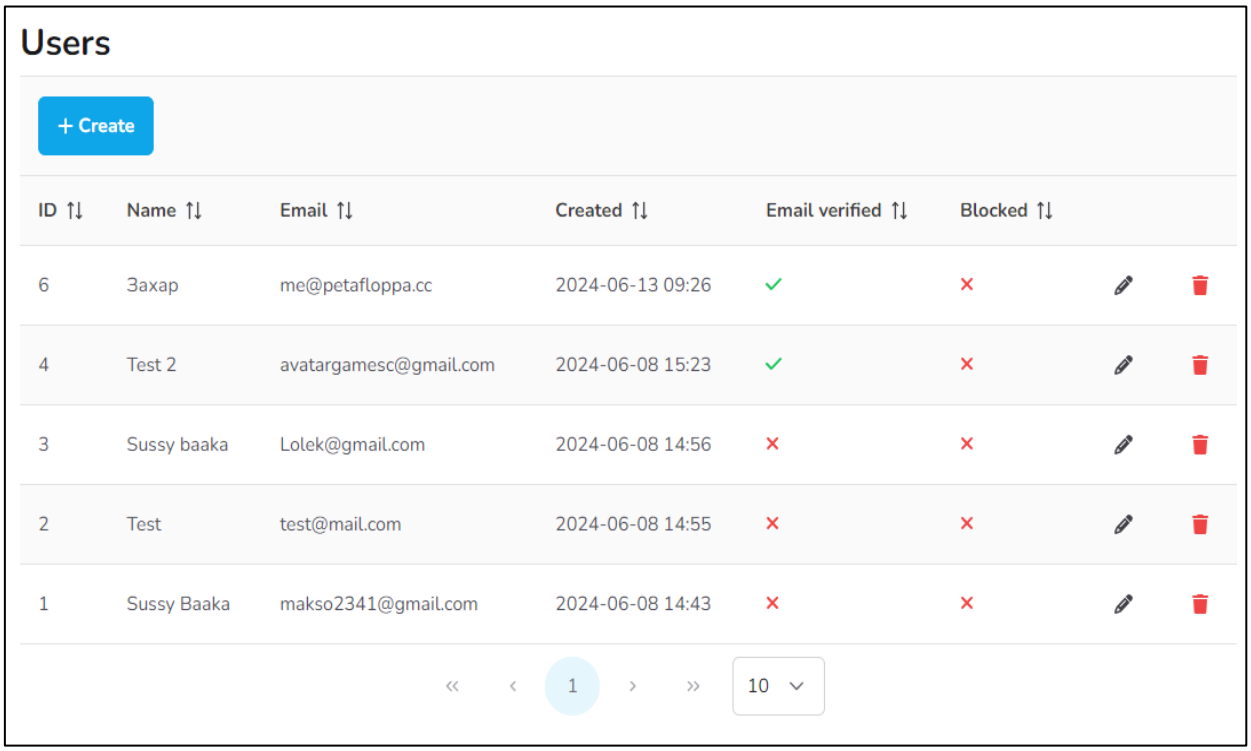

Рисунок 3.13 – Список користувачів у панелі адміністратора

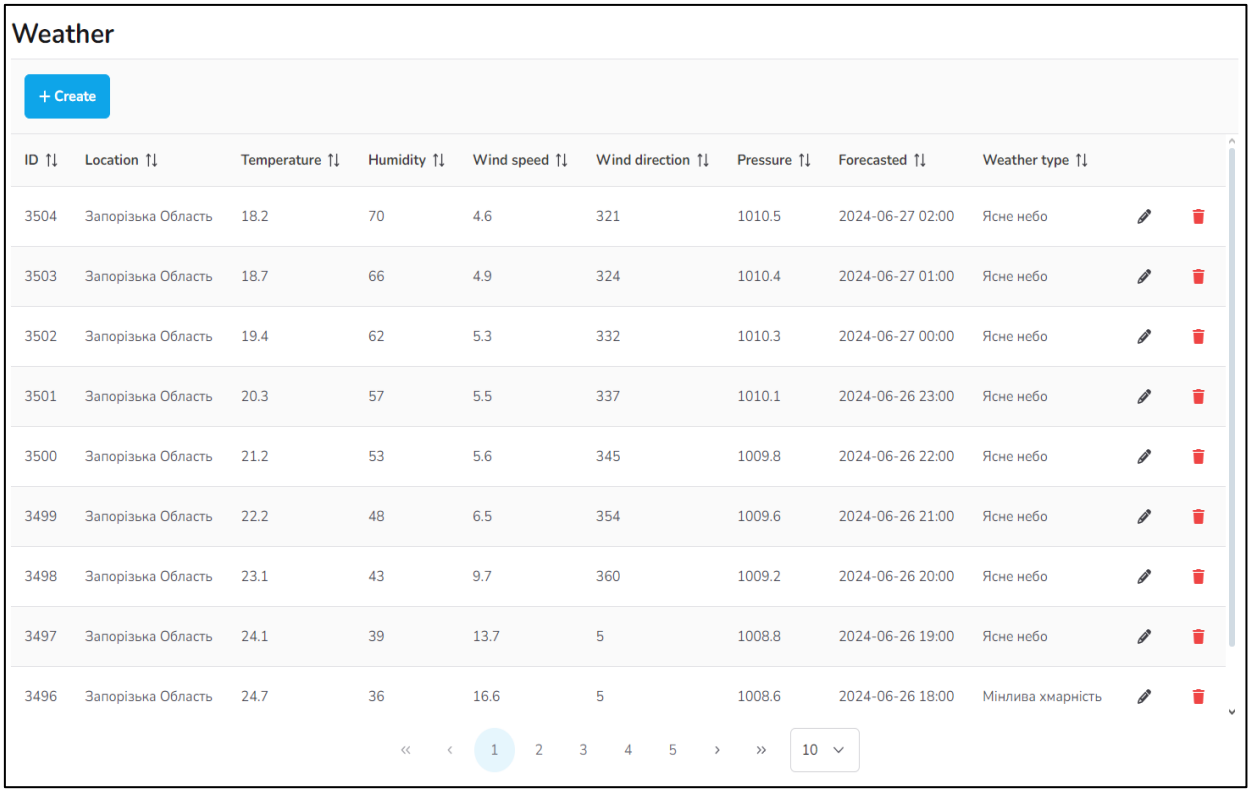

Рисунок 3.14 – Список записів погоди у панелі адміністратора

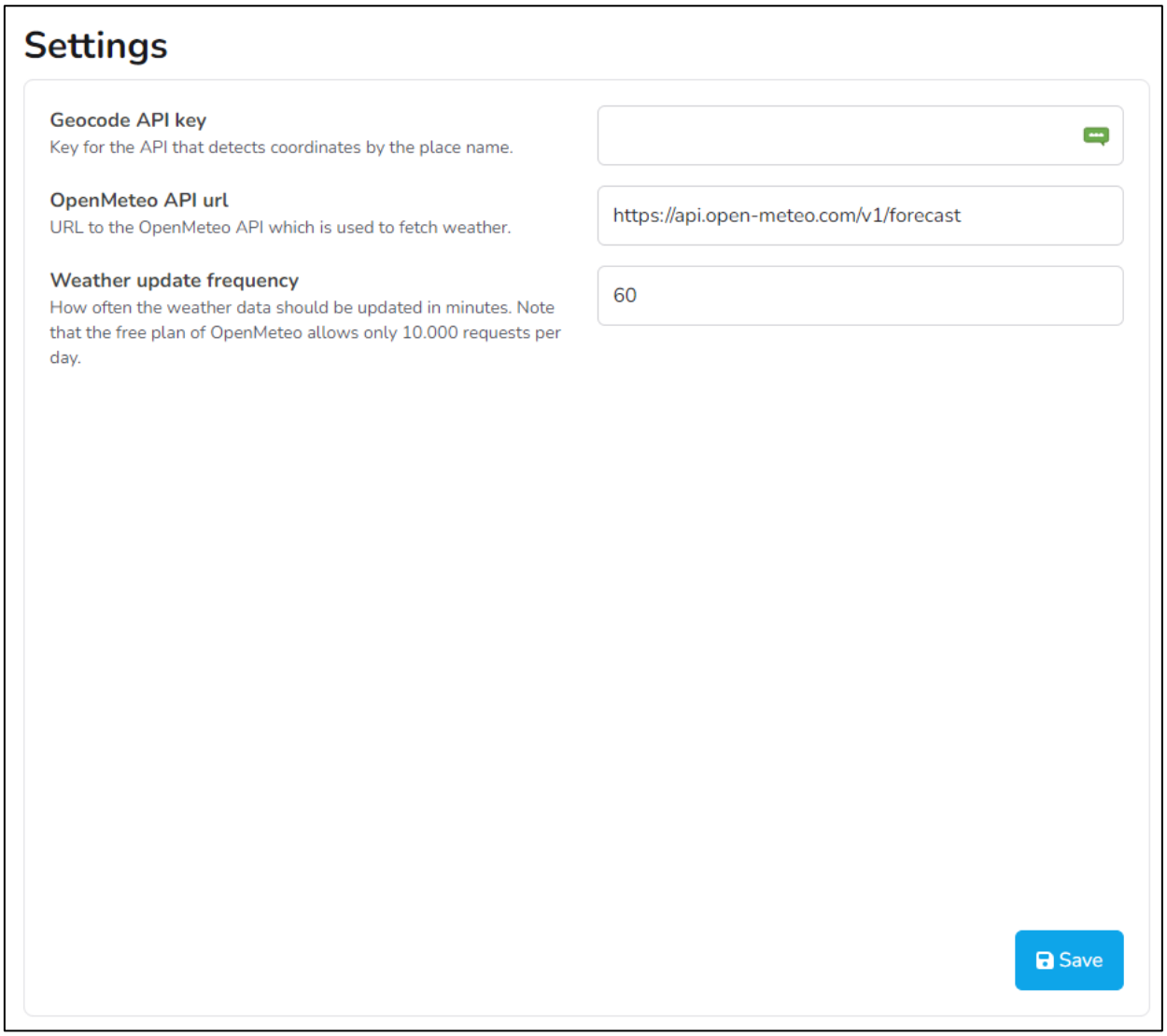

Рисунок 3.15 – Сторінка налаштувань додатка

Завдяки формі таблиці – ці налаштування легко розшири та додати інші при необхідності.

ВИСНОВКИ

В результаті проведених досліджень в процесі виконання кваліфікаційної роботи освітнього рівня «Бакалавр» можна сформулювати наступні висновки.

Розробка серверної частини:

– використання Laravel дозволило створити надійний і масштабований серверний додаток, що інтегрується з зовнішніми API для отримання актуальних даних про погоду;
– запроваджені механізми аутентифікації та авторизації забезпечили високий рівень безпеки користувацьких даних.

Розробка клієнтської частини:

– використання Vue.js забезпечило створення інтерактивного та динамічного інтерфейсу користувача;
– інтеграція з TailwindCSS та PrimeVue сприяла швидкому прототипуванню та покращенню візуальної привабливості додатка.

Вибір технологій та інструментів:

– PHPStorm як основне середовище розробки дозволило ефективно писати та налагоджувати код;
– використання різноманітних пакетів та плагінів значно спростило процес розробки та розширило функціональність додатка.

Інтеграція компонентів

– використання Pinia для управління станом додатка забезпечило зручне та ефективне збереження даних між сесіями;
– бібліотека echarts дозволила створити інтерактивні графіки для візуалізації погодних даних.

Тестування та оптимізація

– автоматичне тестування допомогло виявити та виправити помилки на ранніх етапах розробки, що забезпечило стабільність додатка;

– оптимізація запитів до бази даних та використання кешування підвищили продуктивність додатка.

Виправлення помилок та оптимізація коду:

– ефективне логування та налагодження допомогли швидко знаходити та усувати помилки;
– регулярне очищення проєкту від зайвого коду забезпечило високу якість та підтримуваність коду.

Ці висновки підкреслюють успішність виконання поставлених завдань і демонструють досягнуті результати у розробці функціонального та ефективного вебдодатку для збору та відображення погодних даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документація фреймворку Vue.js. URL: https://vuejs.org (дата звернення: 02.03.2024).
2. Документація фреймворку Laravel. URL: https://laravel.com (дата звернення: 24.03.2024).
3. Interactive Excellence: UI/UX Innovations Demystified. URL: https://medium.com (дата звернення: 04.04.2024).
4. Документація бібліотеки echarts. URL: https://echarts.apache.org/en/index.html (дата звернення: 20.04.2024).
5. Документація бібліотеки Vue-Router. URL: https://router.vuejs.org/ (дата звернення: 02.04.2024).
6. Nixon R. Learning PHP, MySQL & JavaScript. O'Reilly Media, Inc., 2021. 825 p.
7. Shavin M. Learning Vue: Core Concepts and Practical Patterns for Reusable, Composable, and Scalable User Interfaces 1st Edition. O'Reilly Media, 2024. 348 p.
8. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. Marijn Haverbeke, 2011. 435 p.
9. Matula T. Laravel Application Development Cookbook. Packt Publishing, 2013. 272 p.
10. Kleppmann M. Designing Data-Intensive Applications. O'reilly Media, 2017. 613 p.
11. ByteByteGo. System Design Interview. 2020. 270 p.