МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії програмного забезпечення, канд. техн. наук, доцент,

_____Є.О. Давиденко

«____»_____2022 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБЗАСТОСУНОК МОНІТОРИНГУ ПОГОДНИХ УМОВ ІЗ ФУНКЦІЄЮ ПОШУКУ ЛОКАЦІЇ ЗА КРИТЕРІЯМИ

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 − 408.21810825

Студент _____Д. П. Слінкін

підпис

«___»_____2022 р.

Керівник канд.техн.наук, доцент (б.в.з)

_____М. Л. Дворецький

підпис

«___»_____2022 р.

Консультант канд. техн. наук, доцент

_____А. О. Алексєєва

підпис

«___»_____2022 р.

Миколаїв – 2022

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – Програмне забезпечення

СКБД – Система керування базою даних

API – Application Programming Interface

HTTP – HyperText Transfer Protocol

REST – Representational State Transfer

MVC – Model-View-Controller

CSRF – cross-site request forgery

JSON – JavaScript Object Notation

AJAX – Asynchronous Javascript and XML

ВСТУП

У наш час ми не можемо уявити наше життя без інтернету, адже він розвивається настільки швидко і якісно у функціональності, що пару десятків років тому такий функціонал як відправка повідомлень була просто дивовижною річчю і багато хто забув що таке реальна пошта, а тепер цей функціонал став у рази якіснішим і для нас він уже став невід'ємною частиною, написати привіт з ранку або швидко попередити свого начальничка, що запізнишся і так далі. Інтернет не обмежується лише функціоналом надсиланням повідомлень, існує безліч вебзастосунків з дуже багатим функціоналом, наприклад: грати в ігри, проводити прямі трансляції, відеохостинг, моніторинг погодних умов та багато інших.

Вебзастосунки моніторингу погоди дуже поширені і затребувані в наш час, адже від погоди сьогодні чи завтра заздрості чи підеш ти гуляти, чи підеш на пляж. Так само дані надходять у застосунок у режимі, близькому до реального часу. Також існує можливість відображати ретроспективну інформацію (завантажену більш ранні терміни). На перший погляд, це звучить банально — однак раніше стежити за погодою на карті могли лише профільні відомства, а науковцям, метеорологам-аматорам і просто тим, кому цікава фактична погода на дачі або улюбленій лижній трасі, залишалося задовольнятися лише сухими цифрами з архівів. із сайтів погоди, інформаційних порталів або на термометрі за вікном. Тепер у створеному вебзастосунок ці цифри буквально "оживають" на карті.

Вебзастосунок – клієнт-серверний застосунок, у якому клієнт взаємодіє з вебсервером за допомогою браузера. Логіка таких застосунків зосереджена на сервері, а мета браузера у цій системі полягає лише у відображенні інформації. Браузер та сервер в свою чергу поєднані один з одним через запити за протоколом HTTP.

Перевагою використання вебзастосунків є кросплатформність, адже такі застосунки є доступними з будь-якого пристрою та будь-якої операційної

системи. Таких підхід можливо використовувати у інформаційних системах, що не потребують виконання ресурсомістких операцій на клієнті користувача, а значить і у системі, що розробляється.

Вебзастосунок, що розробляється представляє собою моніторинг погодних умов з функцією пошуку локації за критеріями. Основний функціонал застосунку має полягати в представленні інформації користувачам. Представлена інформація, це не тільки погодні умови певної локації, мається ще на увазі, новини про якісь погодні події та їх створювання будь яким користувачем.

В результаті швидкого аналізу існуючого програмного забезпечення в цій сфері можна зробити висновок, що подібні вебзастосунки, використовують дані про погодні умови через API [1] інших систем та застосунків які надають ці дані. Також подібні застосунки, використовують принцип SPA [2] застосунку, архітектура таких вебзастосонків створена так, що при першому завантаженні застосунку, завантажується весь контент, та далі під час роботи застосунок змінює контент(погодні умови) без перезавантаження вебзастосунку.

Вищезгадані API представляють застосункам величезний користь, тому що вся інформація редагується і створюється в одному застосунку і надає її всім іншим, тому інформація на подібних вебзастосунках буде скрізь однакова, як це і повинно бути. А також вищезгаданий SPA надає переваги користувачам такі як, можливість отримати моментальний доступ до функціоналу з будь-якого типу пристрою без проблем із сумісністю, обсягом пам'яті, економить час на повторне завантаження даних, та підвищує продуктивність роботи.

Забезпечення рішення для створення та подальшого управління вебзастосуноком моніторингу погодних умов.

Об’єкт роботи: Процес розробки вебзастосунку моніторингу погодних умов.

Предмет роботи: є підходи і технології розробки вебзастосунку моніторингу погодних умов.

Мета роботи: спрощення процесу пошуку локації за критеріями погодних умов шляхом розробки вебзастосунку моніторингу погоди та фільтрації за значеннями окремих характеристик.

Для того, щоб досягнути цієї мети необхідно виконати наступні завдання:

− проаналізувати існуючі системи та виокремити їх конструктивні особливості, переваги і недоліки;

− проаналізувати ринок наявних технологій і рішень для вирішення завдань;

− сформувати функціональні та нефункціональні вимоги до системи;

− спроектувати структуру бази даних та принцип роботи вебзастосунку;

− розробити та протестувати вебзастосунок із дотриманням поставлених вимог.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Система моніторингу, що розробляється представляє собою клієнт-серверний вебзастосунок, який має немало аналогів зі схожим функціоналом, проте не має зайвих функцій, що не передбачені у обраній предметній області. Це означає, що функціональні вимоги до системи мають бути визначені конкретним переліком функцій, що в свою чергу дозволить спроектувати та розробити систему, яка зможе правильно та логічно виконувати поставлене завдання. Для визначення цього переліку та структури інформаційної системи має бути проведений глибокий аналіз предметної області та існуючих методів і засобів для пошуку найкращого шляху вирішення поставленної задачі.

1.1 Структурні та функціональні особливості

За своєю побудовою та структурою об’єкт роботи є класичним представником трирівневих вебзастосунків – визначеної архітектурної моделі, що складається з трьох компонентів, а саме клієнтської частини застосунку, серверної частини застосунку і серверу баз даних.

Клієнтська частина застосунку це графічний компонент застосунку, що не має прямих зв’язків з базою даних і не навантажений бізнес логікою. Цей рівень містить у собі усі необхідні елементи для представлення інформації у системі, інтерфейси та клієнтської валідації. Клієнтська частина системи що розробляється буде містити у собі різноманітні види інтерфейсів для користувачів не в залежності від ролі останніх. Відвідувачі та гості системи матимуть доступ до моніторингу погодних умов, перегляду усієї передбаченої публічної інформації та матимуть доступ створювати новини певної теми.

Серверна частина застосунку містить у собі основну частину бізнес логіки системи і зв’язок безпосередньо з базою даних та клієнтською частиною застосунку. На цьому рівні як правило реалізується інтерфейс REST API і серверна валідація вхідної інформації.

Сервер бази даних забезпечує збереження даних і реалізується в основному за допомогою засобів СКБД. Логіка роботи з базою даних також може бути реалізована за допомогою засобів серверної частини застосунку, таких як міграції та сідери.

Як було зазначено раніше, функціональні вимоги до системи, що розробляється, представлені чітко визначеним списком функцій, а саме:

- перегляд новин погоди певної країни;
- перегляд новин погоди певного міста;
- перегляд обмеженої кількості новин погоди певної країни;
- перегляд обмеженої кількості новин погоди певного міста;
- перегляд певної новини;
- створення новини будь-яким користувачем;
- моніторинг погодних умов поточного розташування користувача, після підтвердження по надання даних;
- моніторинг погодних умов будь-якого населеного пункта за допомогою різних мов та систем числення: по фарінгейту та по цельсію;

1.2 Огляд існуючих аналогів

Виходячи з мети та завданнь роботи було проаналізовано аналогічні застосунки, а також існуючі рішення та вебтехнології.

Вебзастосунок моніторингу погодних умов Gismeteo [3] представляє собою обширний та багатофункціональний застосунок який створений за допомогою таких мов програмування як PHP та JavaScript.

Застосунок є інформаційним ресурсом погодних умов, а саме інформаційним ресурсом багатьох аспектів що до погоди, що має наступний основний функціонал:

- моніторинг погодних умов певного населеного пункта;
- перегляд погодних умов за минулий період;
- перегляд погодних умов за майбутній період, на місяць вперед;

– перегляд різних аспектів щодо погодних умов: вітер, пилок, якість дороги, тиск, вологість, геомагнітна активність;

– перегляд різних карт щодо погодних умов: карта опадків, карта температури, карта вітра, карта хмарності;

– можливість надати інформер погодних умов певного населеного пункта на будьякий сайт;

Завдяки використанню готових даних які беруться за допомогою системи API застосунок має такі переваги як:

– велика функціональність вебзастосунку (Рисунок 1.1, Рисунок 1.2);

– невелика вартість розробки;

– дуже великі показники швидкості роботи згідно з сервісом PageSpeed Insights від Google (Рисунок 1.3),

– використання продукту і як результат швидкий процес виробництва продукту.

Враховуючи особливості API, що використовується застосунок має певний перелік недоліків. До них належать:

– застарілий дизайн;

– відсутність можливості надати API для інших розробників;

– наявність реклами (Рисунок 1.4).

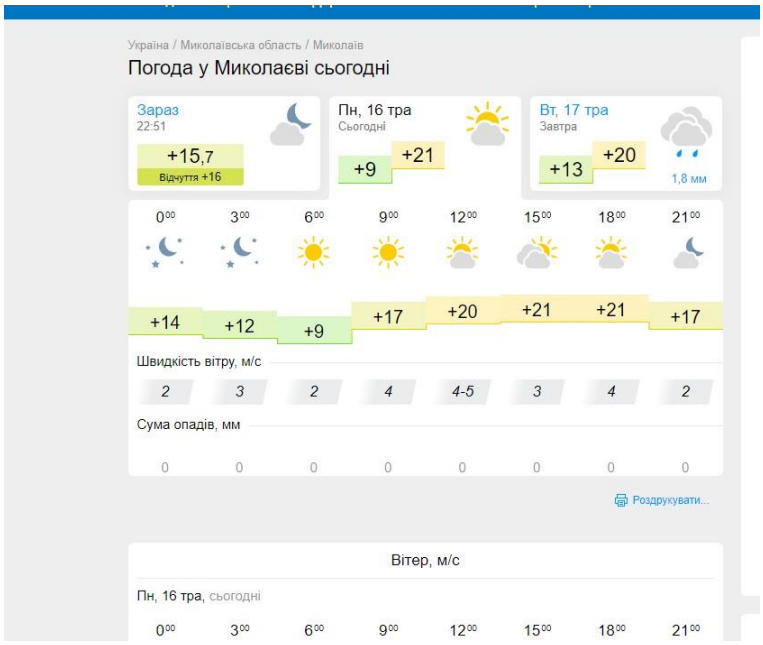

Рисунок 1.1 – Зображення головної сторінки застосунку

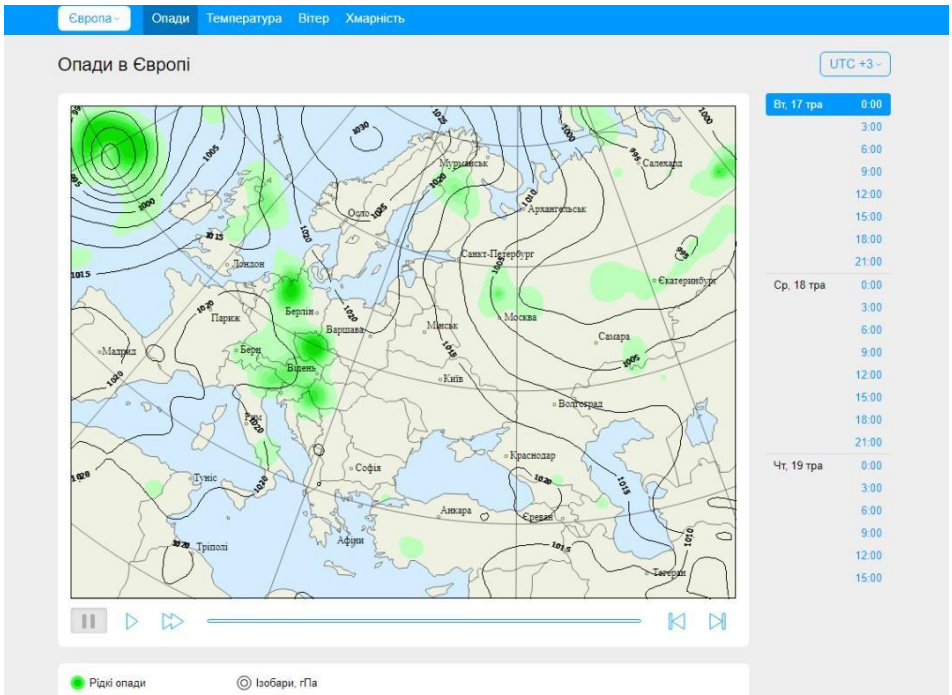

Рисунок 1.2 – Зображення сторінки з картами застосунку

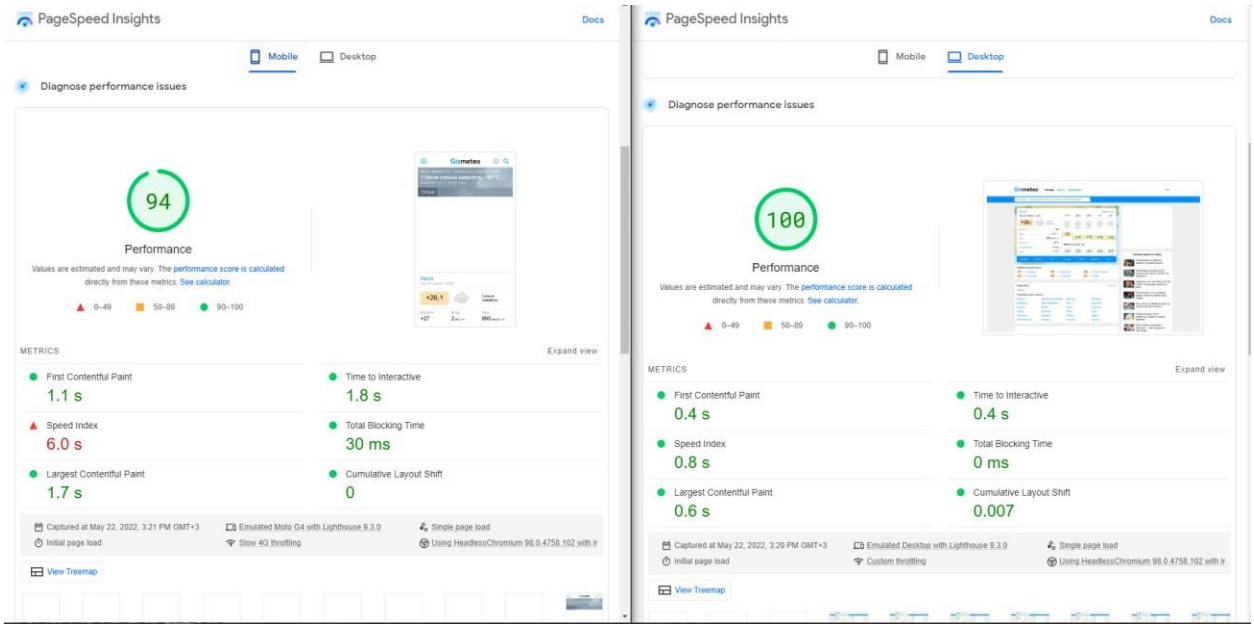

Рисунок 1.3 – Результат тестування швидкості роботи застосунку

Рисунок 1.4 – Зображення реклами на сторінках застосунку

Американський сайт моніторингу погодних умов та з новинами про погоду має подібну архітектуру, використовує віджет AccuWeather який надає усю інформацію чезе API, побудований він з використанням мов програмування PHP і JavaScript.

Основні функції головного інформаційного ресурсу університету полягають у:

– моніторинг погодних умов певного населеного пункта;

– перегляд погодних умов за минулий період;

– перегляд погодних умов за майбутній період, на місяць вперед;

– перегляд радару погоди(карта) з опадами;

– перегляд новин про погоду;

– перегляд відео щодо новин про погоду;

– перегляд різних аспектів щодо погодних умов: якість вітру, тиск, вологість тощо;

– перегляд ризику такі як: алергія, для здоров’я, шкідників (Рисунок 1.5);

– перегляд можливості заходу на свіжому повітрі, подорожі та поїздки на роботу (Рисунок 1.6);

Перевагами є:

– невелика вартість виробництва та використання продукту;

– швидкий процес виробництва продукту;

– зручний інтерфейс;

– велика функціональність вебзастосунку;

– наявність якісної адаптації під девайси з малим розміром екрану;

– відсутність реклами;

– непогана швидкість роботи на десктопі згідно з сервісом. PageSpeed Insights від Google (Рисунок 1.7),

Серед недоліків варто визначити наступні з них:

– застарілий дизайн вебзастосунку;

– повільна швидкість роботи для мобільних пристроїв (Рисунок 1.8).

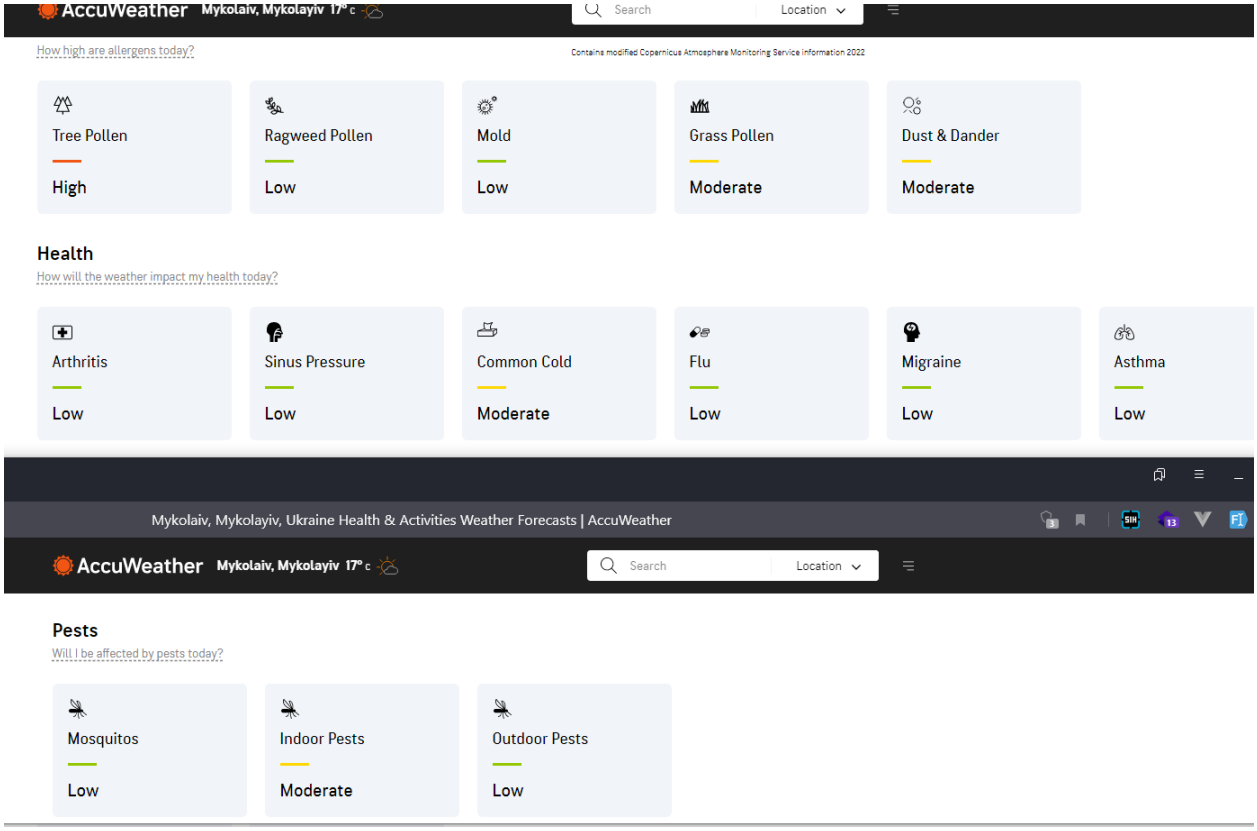

Рисунок 1.5 – Зображення ризиків до здоров’я

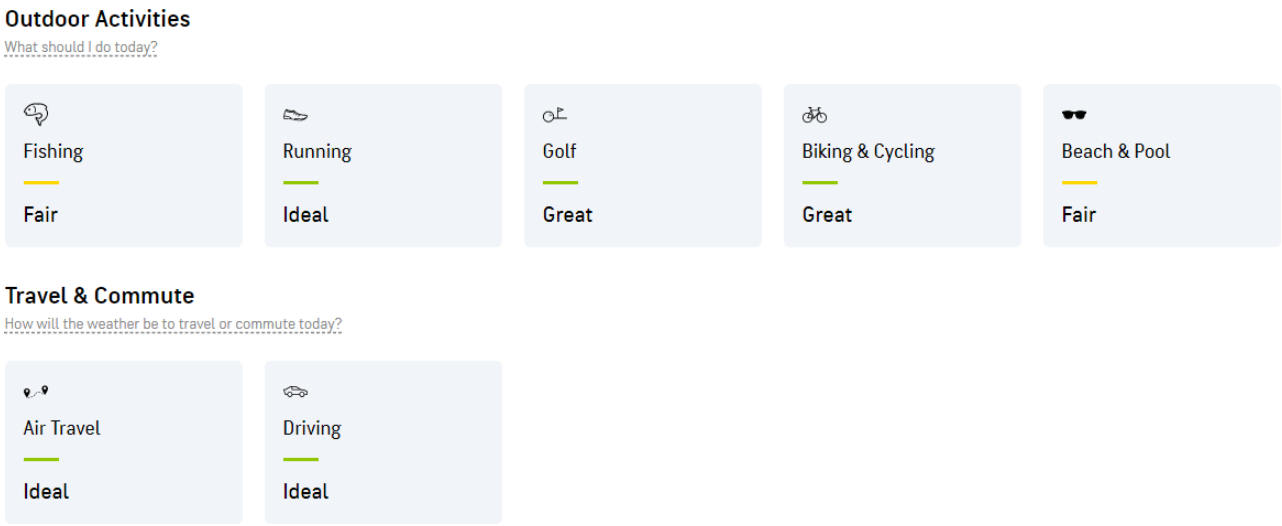

Рисунок 1.6 – Зображення можливості розваг та занять

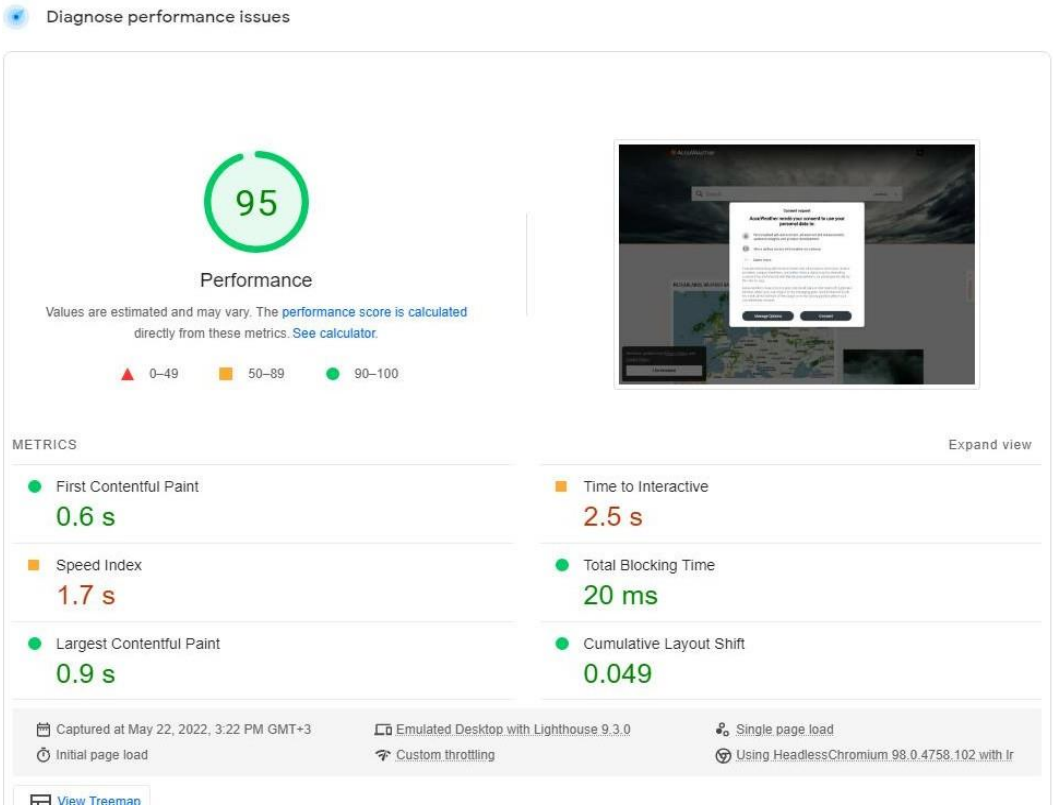

Рисунок 1.7 – Результат тестування швидкості роботи застосунку на десктопі

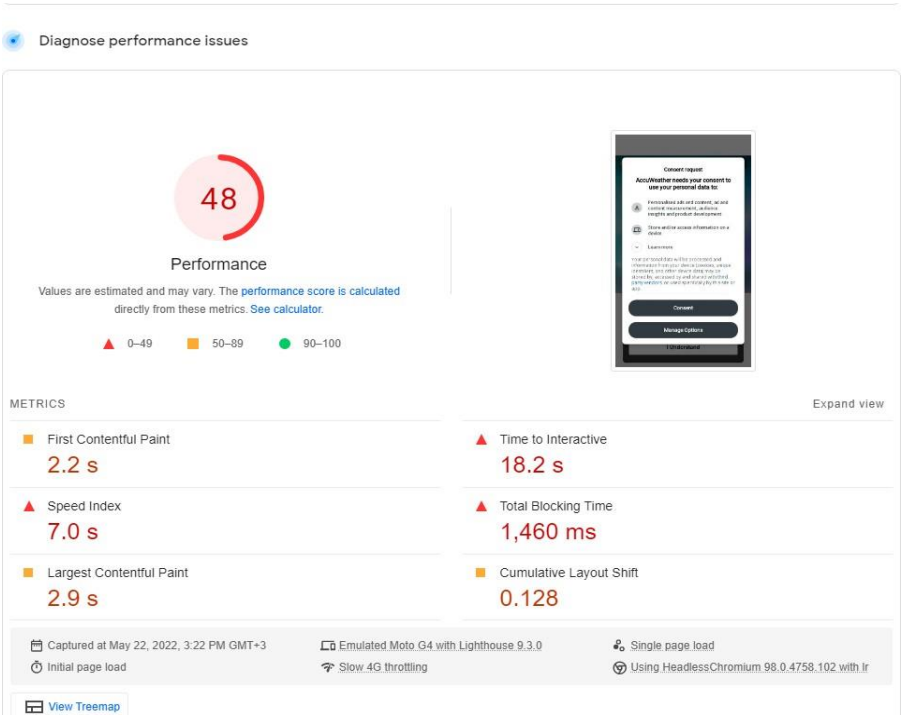

Рисунок 1.8 – Результат тестування швидкості роботи застосунку на смартфоні

1.3 Аналіз існуючих методів та засобів вирішення поставлених завдань

Для того щоб у повній мірі виконати поставлені завдання та досягти мети роботи не існує інших методів та шляхів, крім розробки сучасного, адаптивного та кросбраузерного трирівневого вебзастосунку.

Їх основною перевагою є кросплатформність, такі застосунки доступні з будь-якого пристрою або девайсу. Крім того, за попереднім аналізом вимог до системи, вона не потребує виконання ресурсомістких операцій і як результат такої характеристики – веб-застосунки – єдиний шлях до вирішення поточної проблеми.

Для того, аби створити клієнтську частину будь-якого вебзастосунку використовуються такі засоби та технології як HTML, CSS та JavaScript, Vue [4], VueX [5].

Що стосується серверної частини веб застосунку, найчастіше розробники схиляють свій вибір у бік PHP а точніше фреймворк Laravel [6], як найбільш популярної мови для створення вебзастосунків. Згідно з PHP використовується приблизно у 79% усіх вебзастосунків світу. Він є популярнішим у вісім разів за ASP.NET – його найближчого конкурента серед подібних вебтехнологій. Окрім цього, рівень використання PHP є незмінним впродовж довгого часу, що свідчить про стабільність у використанні цього інструменту.

Для роботи вебзастосунку, що створений за допомоги мови програмування найчастіше використовується СКБД MySQL. Це зумовлено наявністю стабільних розширень для роботи із зазначеною СКБД.

Написання усієї системи з нуля не має практичної мети, через велику втрату часу на розробку базових речей. Для зменшення кількості ресурсів на розробку системи використовують фреймворки. Вони містять величезну кількість заздалегідь написаного коду, що може бути використаний повторно,

а їх структура – це найкращі архітектурні рішення, що довели свої переваги впродовж років. До найбільш популярних фреймворків, що створені за допомогою мови програмування PHP належать такі як Laravel, Symphony, Yii2, тощо. Фреймворки, що створені за допомогою мови програмування JavaScript належать такі як Vue, React, Angular, тощо.

1.4 Специфікація вимог до ПЗ

Веб-застосунок, що розробляється представляє собою моніторинг погодних умов та новин про погоду. Система призначається для усіх користувачів які можуть скористатися вебзастосунком.

Сферою застосування продукту є інформаційний простір щодо погодних умов у різних населених пунктах.

Основні ролі користувачів:

– гість – користувач який має можливість дізнаватися погодні умови різних населених пунктів, дивитися новини стосовно погоди, а також має можливість написати свою новину про погоду.

За своєю структурою система є класичним представником трирівневих вебзастосунків і складається з трьох компонентів, а саме клієнтської частини застосунку, серверної частнини застосунку і серверу баз даних.

Функціонал системи представлений наступним переліком функцій:

Перегляд погодних умов первного населеного пункта на вебсайті.

Вхідна інформація: запит до REST API на отримання даних.

Вихідна інформація: відповідний результат запиту.

При введенні неправильних даних запит не виконується.

Перегляд погодних умов населеного пункта користувача.

Вхідна інформація: запит до REST API з координатами кристувача на отримання даних.

Вихідна інформація: відповідний результат запиту.

При відмінні наданні прав на місце находження запит не виконується.

Перегляд усіх новин.

Вхідна інформація: запит за певною URL адресою потім до REST API на на отримання даних.

Вихідна інформація: відповідний результат запиту.

При помилковому URL відповідь на запит негативна.

Перегляд новин за категрією.

Вхідна інфорація: запит до REST API на редагування інформації.

Вихідна інформація: відповідний результат запиту.

При введенні неправильних даних запит не виконується.

Перегляд певної новини.

Вхідна інфорація: запит за певною URL адресою.

Вихідна інформація: відповідний результат запиту.

При помилковому URL відповідь на запит негативна.

Перегляд певної новини.

Вхідна інфорація: запит до REST API на створення новини.

Вихідна інформація: відповідний результат запиту.

При помилці валідації запит не виконується.

Застосунок складається з трьох компонентів: клієнтська частина застосунку, серверна частина застосунку, сервер баз даних. Програмним забезпеченням для ведення інформаційної бази є СКБД MySQL.

Мовою для розробки ПЗ є PHP з використанням фреймворку Laravel та JavaScript з використанням фреймворку VueJS.

До вимог надійності можна віднести:

– високу відмовостійкість;
– стабільність роботи при нестандартних умовах.

До вимог супроводжуваності належать:

– застосунок має на меті максимально зручне використання функціональності системи.

– застосунок не повинен викликати зайвих проблем у користувача при використанні.

Вимоги практичності полягають у:

– максимально зручному використання функціональності системи;
– відсутності зайвих проблем у користувача при використанні.

Вимоги безпеки полягають у тому, що особисті дані користувачів не повинні бути використані у неправомірних цілях.

Висновки до розділу 1

В ході роботи описаної в поточному розділі було проведено детальний аналіз предметної області, а саме існуючих застосунків. Було виявлено їх переваги та недоліки, виявлено основні технологічні особливості та обмеження.

Виходячи із проведеного аналізу застосунків та ринку вебтехнологій було проаналізовано основні засоби для вирішення поставлених задач програмним шляхом і досягнення основної мети роботи. Тема роботи вебзастосунок моніторингу погодних умов із функцією пошуку локації за критеріями є актуальною.

Згідно з створено специфікацію вимог до ПЗ дотримуючись зазначеної структури. У специфікації вимог зазначено короткий опис ПЗ, що розробляється, діаграма з варіантами використання вебзастосунку, таблиці з сценаріями використання, функціональні та нефункціональні вимоги до ПЗ.

2 МОДЕЛЮВАННЯ СТРУКТУРИ БАЗИ ДАНИХ ПЗ ТА АРХІТЕКТУРА, ПРОЕКТУВАННЯ ПЗ

У процесі роботи виникла потреба у розробці певних проектних рішень для забезпечення повного та успішного виконання функціональних та нефункціональних вимог до ПЗ, що були встановлені в попередньому розділі. Для створення вищезгаданих проектних рішень використовуються різноманітні методики та підходи.

Одним з найбільш важливих елементів інформаційної системи є база даних, адже вона має повністю задовольнити зазначений у попередньому розділі функціонал. В даному вебзастосунку використовується метод API, основна інформація, а саме дані про погодні умови запрошуються з бази даних іншого вебзастосунку OpenWeather [7], який безкоштовно та за гроші надає інформацію про погодні умови. Але крім інформації що надається з API, в вебзастосунку використовується власна база даних. Для цього необхідно пройти етап проектування бази даних, який в тому числі полягає у створенні логічної та фізичної моделі даних.

Також в ході повного циклу розробки системи є вибір необхідних технічних рішень, що дозволять у повній мірі виконати поставлені завдання та досягти мети роботи, представлення ключових функціональних моментів та сценаріїв роботи застосунку за допомогою діаграм, схем та сценаріїв використання, а також розробка макетів інтерфейсів застосунку.

2.1 Проектування логічної моделі даних

Логічна модель даних – модель даних певної області, яка подає абстрактну структуру області інформації. Вона відображає логічні зв’язки між елементами даних.

Проектування логічної моделі буде виконуватись ER-методом (entity-relation, сутність-зв'язок), тобто логічна модель повинна складатися з сутностей, які мають бути пов’язані одна з одною.

Сутність – це абстракція реально існуючого об’єкту або явища. Проаналізувавши предметну область, можна виділити наступні типи сутностей:

- користувачі;
- новини.

Сутність «користувач» містить у собі інформацію про користувачів системи, задля реалізації функціональності застосунку у сфері написання новин про погодні умови.

Сутності «новини» містять інформацію щодо погодних умов: країна, місто, заголовок, детальніша інформація та ім’я автора.

Таблиця 2.1 – Атрибути таблиць у базі даних

Таблиця	**Атрибут**	**Опис атрибуту**
users	id	Ключ
	name	Назва користувача
	created_at	Дата створення користувача
	updated_at	Дата редагування користувача
news	id	Ключ
	title	Заголовок новини
	description	Опис новини
	city	Назва міста
	country	Назва країни
	user_id	Зовнішній ключ, id(users)
	created_at	Дата створення новини
	updated_at	Дата редагування новини

Згідно з [4] в результаті проєктування логічної моделі ER-методом результатом самого проєктування має бути ER-діаграма – спеціальна модель даних, що описує визначені сутності та зв'язки між ними.

Остаточна схема даних на основі таблиці атрибутів представлена на рисунку 2.1.

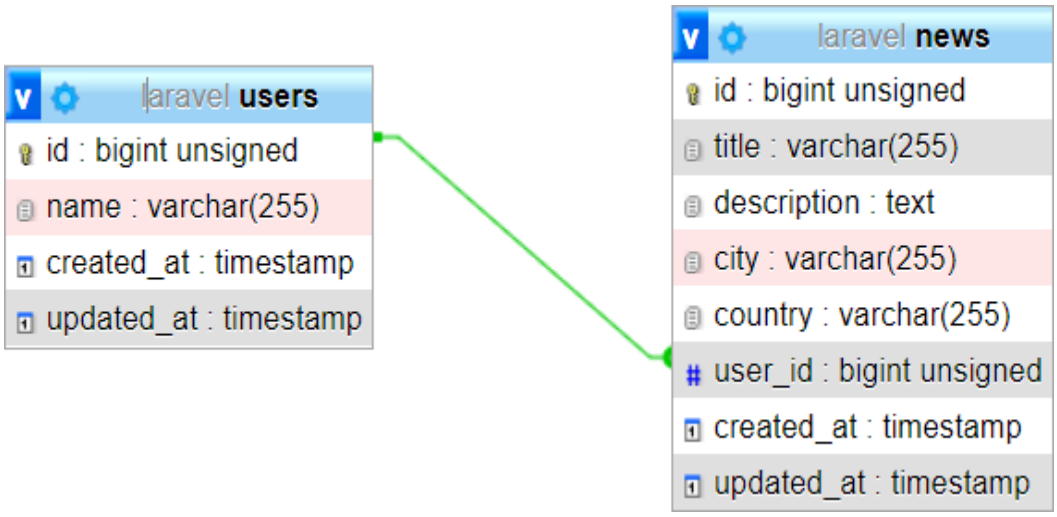

Рисунок 2.1 – Схема даних у базі

2.2 Проектування фізичної моделі даних

Фізична модель бази даних – це модель, що реалізована в СКБД. Вона є фактичним описом бази даних.

Для створення бази даних немає жодної потреби створювати таблиці за допомогою графічного інтерфейсу спеціальних застосунків або SQL-запитами вручну. Такі готові рішення як фреймворк Laravel реалізують технологію міграцій. Вона допомагає у роботі з базами даних, виконуючи SQL-запити самостійно, без втручання розробника, якому необхідно лише використати міграції.

Детальний опис механізму роботи міграцій наведений у третьому розділі.

2.3 Вибір технологій та мов програмування

Для розробки будь-якого вебзастосунку використовуються такі технології як HTML, CSS та JavaScript, а використання найбільш ефективних підходів до роздробки за допомоги вказаних мов дозволять побудувати сучасний і адаптивний вебзастосунок.

Для вирішення завдання з технічної точки зору необхідно було знайти рішення, що є найбільш популярним у використанні та таким, що задовольняє усім вимогам до системи. Більш того необхідно врахувати недоліки проаналізованих існуючих систем, щоб не допустити їх у системі, що розробляється, або працювати над тим, щоб зменшити їх вплив на процес використання застосунку. Виходячи з проведеного аналізу аналогічних систем і ринку наявних вебтехнологій було прийнято рішення про використання PHP та JavaScript як основної мови для розробки вебзастосунку.

Заради зменшення кількості ресурсів на розробку системи було прийнято рішення про використання фреймворку Laravel – безкоштовного опенсоурс фреймворку, що створений за допомогою мови програмування PHP. Він дозволяє використовувати його для розробки складних вебзастосунків, щоб зробити увесь процес простішим, швидшим та безпечнішим. Згідно з [5] він є найпопулярнішим фреймворком на сьогоднішній день.

Ключові особливості Laravel полягають у:

– елегантній структурі та зрозумілому синтаксисі;

– структурованій та детальній документації;

– вбудованих функціях для керування маршрутизацією, користувачами, кешуванням;

– власній консолі Artisan для роботи з різними інструментами;

– інструментах для роботи з базою даних, таких як міграції, сідери, ORM Eloquent та Fluent Query Builder;

– валідаторах – структурах перевірки даних за певними параметрами;

– Blade – шаблонізаторі для представлень;

– Jobs – інструменті для виконання асинхронних завдань;

– Mix – інструменті для компіляції стилів та скриптів;

– CSRF захисті за замовчуванням – механізмі токенізації для уникнення атак з інших ресурсів.

Laravel має найширшу та найкращу спільноту розробників, що використовують цей фреймворк, що є запорукою легкого вирішення проблем та довготривалої підтримки впродовж років.

Крім того, Laravel є ще й найшвидшим PHP фреймворком серед усіх інших вебтехнологій, що використовують PHP як основну мову розробки і роботи [6].

Що стосується клієнтської частини вебзастосунку використовуються такий безкоштовний JavaScript опенсоурс фреймворк як VueJS. Легко інтегрується у проекти з використанням інших JavaScript-бібліотек. Може функціонувати як веб-фреймворк для розробки SPA програм у реактивному стилі.

У SPA архітектура влаштована таким чином, що при завантаженні першої сторінки з сервера на комп'ютер передається весь необхідний для роботи код (HTML, CSS та JavaScript). Далі контент підвантажується та оновлюється без перезавантаження сторінки.

Після завантаження першої сторінки вся взаємодія із сервером відбувається через запити AJAX. Запити AJAX повертають дані у форматі JSON. У цьому формування верстки сторінки відбувається за клієнта.

Реактивність означає, що представлення моделі MVC змінюється в міру зміни моделі. У Vue розробники просто прив'язують уявлення до відповідної

моделі, і Vue автоматично спостерігає за змінами моделі і перемальовує уявлення. Ця функція робить управління станом Vue досить простим та інтуїтивно зрозумілим.

Ефекти переходу

− Автоматично застосовувати CSS класи при переходах та анімації;

− Ви можете працювати зі сторонніми бібліотеками анімації CSS, як-от Animate.css;

− Використовуйте JavaScript функції для перехоплення переходу, щоб керувати безпосередньо DOM;

− Може використовуватися у поєднанні зі сторонніми бібліотеками анімації JavaScript, такими як Velocity.js.

При проектуванні і розробці вебзастосунків з використанням баз даних використовують СКБД – система, що забезпечує визначення бази даних, її створення, контроль та використання даних у ній. Для роботи вебзастосунку знадобиться СКБД MySQL.

2.4 Розробка сценаріїв використання та UML-діаграм

Діаграм, схеми та спеціальні описи, що називаються сценаріями використання, допомагають краще зрозуміти вимоги до системи, виправити їх за потреби та не допустити непорозумінь під час розробки. В ході роботи пропонується створити як мінімум по одному прикладу з необхідних діаграм та описів, що наведені в підрозділах поточного розділу, а також здійснити.

Сценарії використання системи можуть бути представлені у графічному вигляді. Приклади примітивних сценаріїв для користувачів наведені нижче: сценарій моніторингу погодних умов певного населеного пункту (Рисунок 3.1), сценарій моніторингу погодних умов теперішнього місцезнаходження користувача (Рисунок 3.2), сценарій перегляду усіх новин (Рисунок 3.3), сценарій перегляду новин про погоду стосовно обраної категорії (Рисунок 3.4),

сценарій перегляду певної новини про погоду (Рисунок 3.5), сценарій створення новини про погоду користувачем (Рисунок 3.6).

Рисунок 3.1 – сценарій моніторингу погодних умов певного населеного пункту

Рисунок 3.2 – сценарій моніторингу погодних умов теперішнього місцезнаходження користувача

Рисунок 3.3 – сценарій перегляду усіх новин

Рисунок 3.4 – сценарій перегляду новин про погоду стосовно обраної категорії

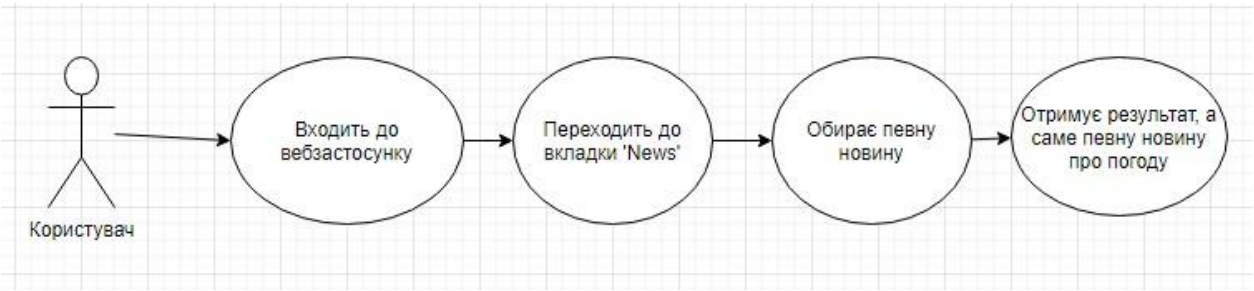

Рисунок 3.5 – сценарій перегляду певної новини про погоду

Рисунок 3.6 – сценарій створення новини про погоду користувачем

Сценарії використання можуть бути представленими в текстовому вигляді, крім того, розподіляються за формами – короткі, поверхневі і повні. В ході роботи було розроблено по одному прикладу для кожної форми.

Перший сценарій (коротка форма): перегляд веб-сайту користувачем.

Гість відвідує сторінку ‘News’, переходить до повного списку новин. Гість обирає новину, що його цікавить. Гість переходить на сторінку цієї новини. Гість читає новину.

Другий сценарій (поверхнева форма): створення новини користувачем.

Головний сценарій (успішний):

Користувач відвідує сторінку ‘Create News’. Заповнює назву новини, назву країни, назву міста, опис новини про погоду, ім’я автора. Зберігає інформацію про новину.

Альтернативні сценарії:

1. Користувач намагається створити новину без обов’язкових полів чи контенту. Валідація системи повідомляє користувача про помилки.
2. Користувач намагається створити новину з недостатньою кількістю знаків в одному з полей. Система повідомляє користувача про те, що він помилився.
3. Користувач намагається створити новину з порушенням ліміту по знакам в одному з полей. Система повідомляє користувача про те, що він помилився.

Третій сценарій (повна форма): створення новини про погоду користувачем.

У цьому сценарії буде описаний процес створення новини про погоду.

Level: сторінка створення новини ‘Create News’.

Primary Actor: Користувач.

Stakeholders and interests:

1. Користувач/автор: зацікавлений в тому щоб, написати новину про погоду певного населеного пункту.
2. Користувач: зацікавлені у тому, щоб дізнатися новини про погоду певного населеного пункту.

Main Success Scenario:

1. Користувач переходить до сторінки ‘Create News’.

2. Користувач заповнює усі необхідні поля які задовольняють валідацію, а саме: назва країни, назва міста, заголовок новини, опис новини, ім’я користувача.
3. Користувач зберігає новину.

Extentions:

1. В будь який час користувач виконує дію, що відхиляється від сценарію.
   a. Користувач намагається створити новину без обов’язкових полів чи контенту. Валідація системи повідомляє користувача про помилки.
   b. Користувач намагається створити новину з недостатньою кількістю знаків в одному з полей. Система повідомляє користувача про те, що він помилився.
   c. Користувач намагається створити новину з порушенням ліміту по знакам в одному з полей. Система повідомляє користувача про те, що він помилився.
2. Виникає технічна помилка.
   a. Не вдалося під’єднатися до БД.
   b. Виникла помилка на серверній частині застосунку, що зробила неможливим виконання змін.

Special Requirements:

1. Повнорозмірний пристрій (таблет, ноутбук, десктоп) для того аби бачити усю інформацію на екрані і не помилитися під час внесення інформації.
2. Стабільне підключення до мережі Інтернет для успішної відправки запитів на сервер та отримання очікуваного респонсу.

Technology and Data Variations List:

Користувач має бути впевненим, що його інформацію про погоду певного населеного пункту відповідає реальності.

Frequency of Occurrence:

Система має працювати безперервно і в будь-якому випадку надавати користувачу відповідний респонс.

Miscellaneous (Open Issues):

1. Система має бути протестована на прогавини за різних обставин додавання інформації в БД.
2. Провести аналіз використання різних елементів інтерфейсу, що є зайвим, що необхідно додати.

Для того, щоб зібрати та узагальнити вимоги до системи та її функціоналу, використовується діаграма варіантів використання. Її мета полягає в представлені загальної картини того, як повинен працювати застосунок. Саме така діаграмма була розроблена для системи, що розробляється (Рисунок 3.7).

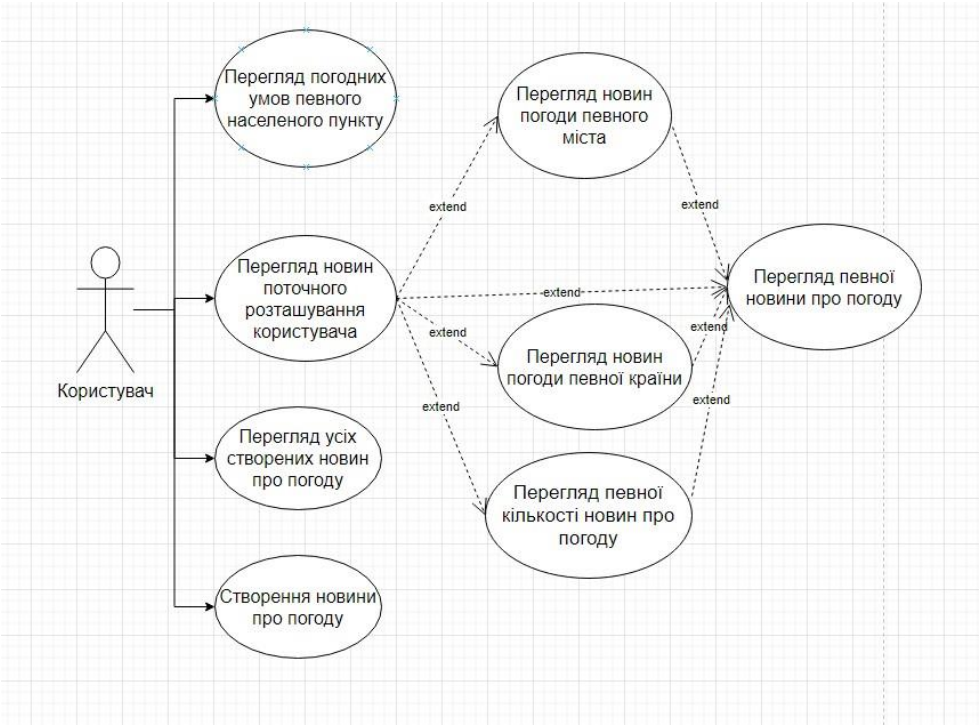

Рисунок 3.7 – Діаграма варіантів використання системи

Для того, щоб конкретизувати конкретну функцію або уривок діяльності, використовуються діаграми діяльності. Для прикладу в ході роботи було спроектовано саме такий тип діаграми, що описує процес створення новини про погоду користувачем (Рисунок 3.8).

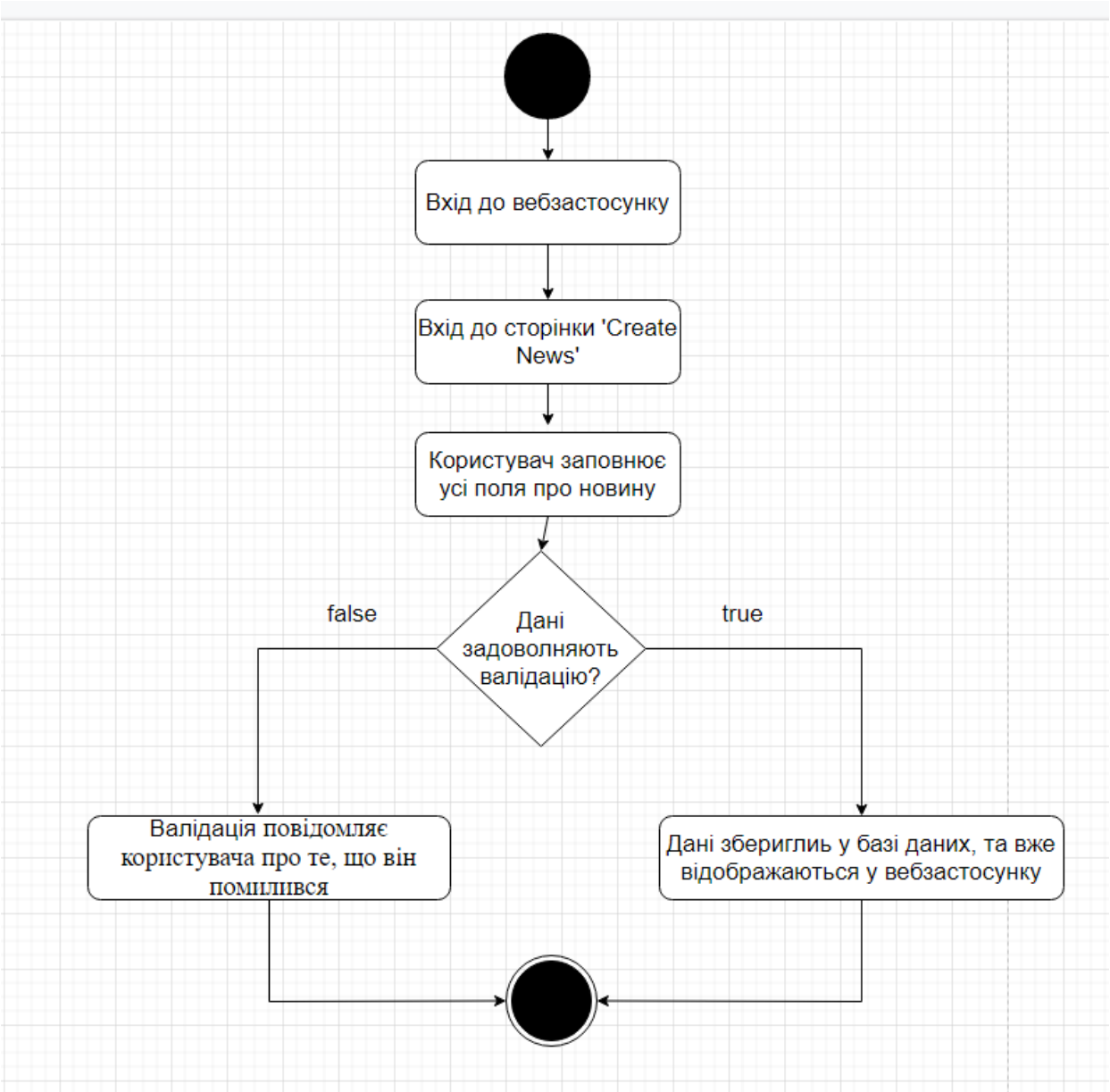

Рисунок 3.8 – Діаграма діяльності

Діаграма класів та класів-контролерів відображає статичні елементи системи, а саме класи, атрибути класів, їх тип даних, відношення та методи. В ході роботи було спроектовано діаграму класів (Рисунок 3.9) та класів

контролерів (Рисунок 3.10). системи, яка описує необхідну структуру класів інформаційної системи.

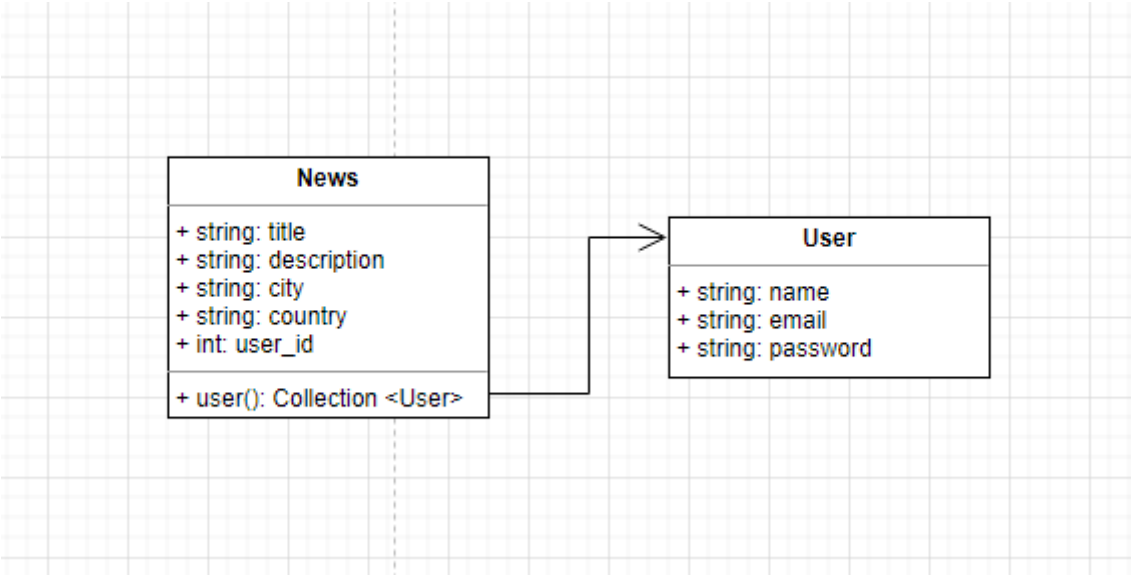

Рисунок 3.9 – Діаграма класів

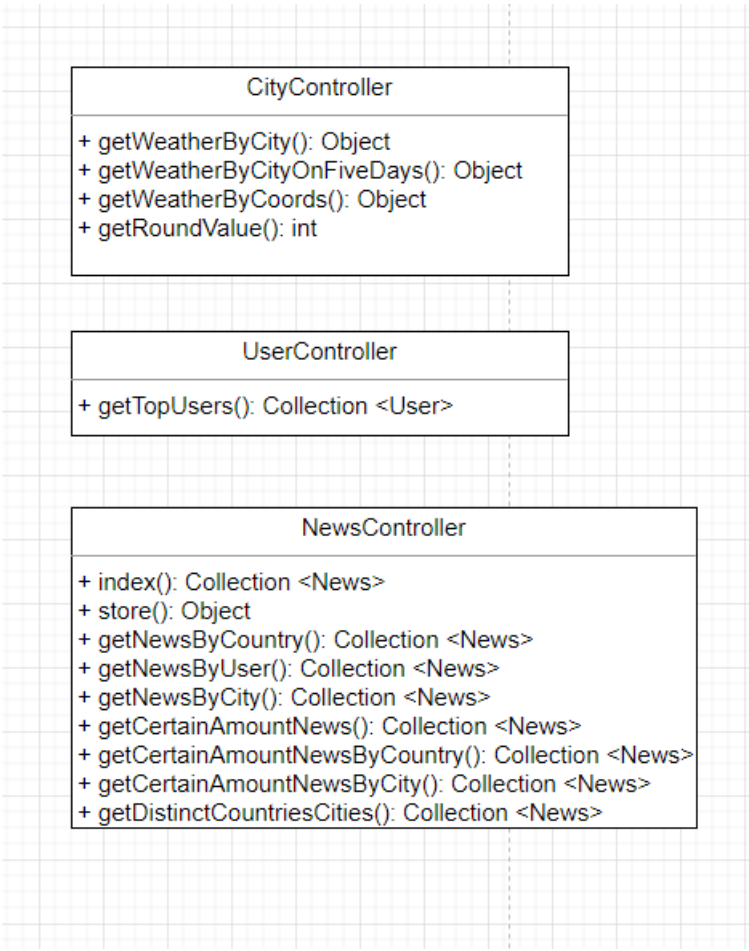

Рисунок 3.10 – Діаграма класів-контролерів

Слід зазначити, що діаграма класів є лише умовним та досить обмеженим описом того, як будуть виглядати класи системи, адже значна кількість системних класів або контролерів не містять у собі бізнес-логіки.

Цей тип діаграми може містити позначення деяких елементів поведінки за допомогою відображення методів класів. Наприклад, на наведеній вище діаграмі описуються спеціальні «магічні властивості» – методи, що створюються під час побудови системи за допомогою фреймворку Laravel. Вони відображають зв’язки між визначеними сутностями та слугують для динамічного зв’язку в ході виконання коду.

Діаграми послідовності представляють собою ще один із способів як описати певні сценарії використання. Основна її перевага в тому, що на ранньому етапі проектування сценаріїв використання є можливість точно дізнатися структуру взаємодіючих компонентів. Це дозволить трансформувати визначені компоненти у конкретні класи та методи цих класів.

В ході роботи було спроектовано діаграму послідовності, яка описує процеси створення новини про погоду у системі (Рисунок 3.11).

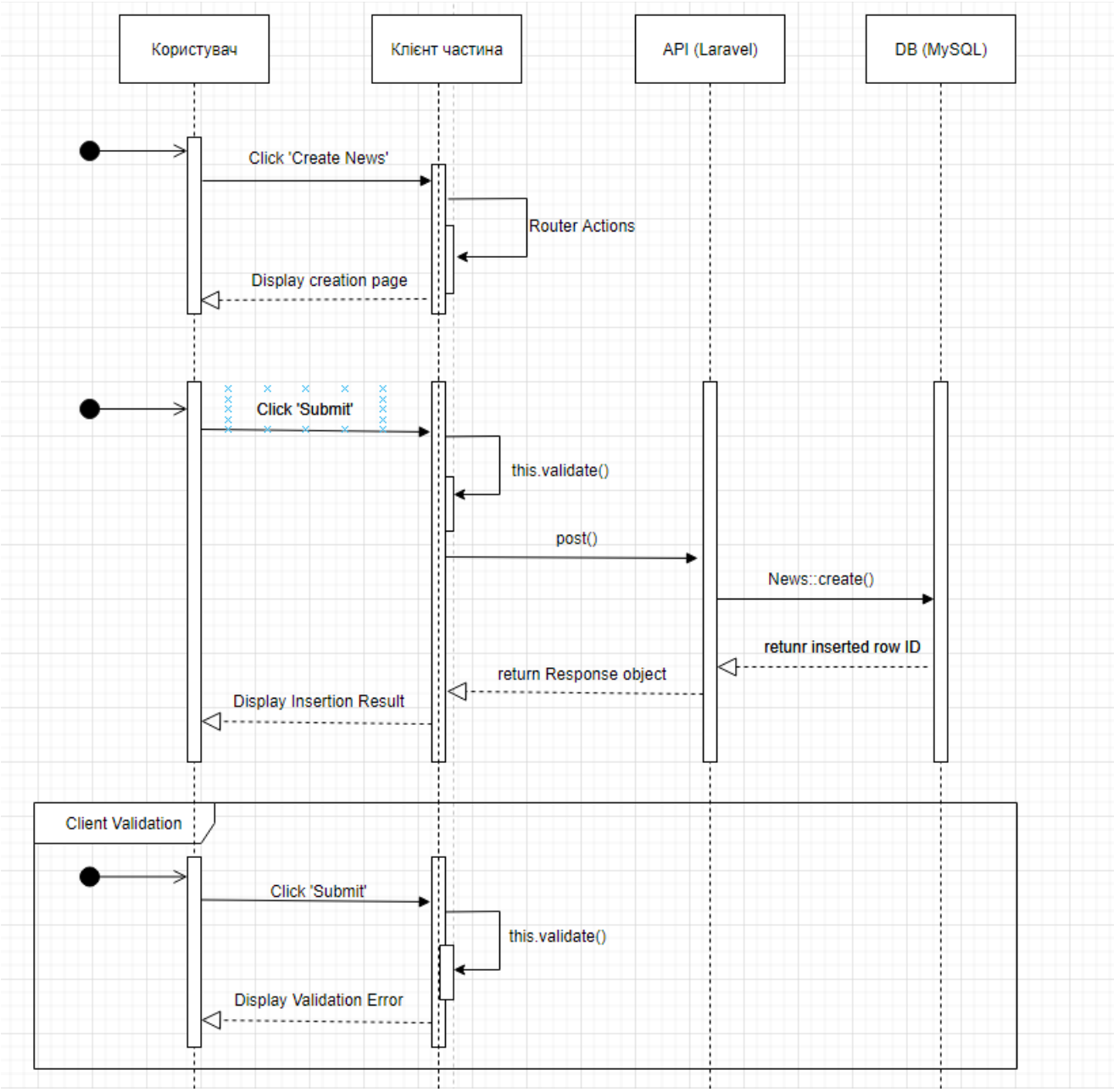

Рисунок 3.11 – Діаграма послідовності

2.5 Опис інтерфейсів ПЗ

Розробка моніторингу погодних умов ускладнюється без проектування та розробки макетів інтерфейсів, а у випадку вебзастосунків стає майже неможливою, адже вкрай необхідно уявляти зовнішній вигляд сторінок, їх склад та структуру.

Вебзастосунок як правило має певні незмінні елементи розмітки, такі як верхній навігаційний бар (Рисунок 3.12) та підвал (Рисунок 3.13). Буде змінюватися лише контент сторінки, який розташований посередині.

Рисунок 3.12 – Дизайн верхнього навігаційного бару

Рисунок 3.13 – Дизайн підвалу

(Рисунок 3.14) відображає графічне представлення домашньої сторінки вебзастосунку, складається зі статичних елементів, меню, інформацію про погодні умови, стрічка та карту.

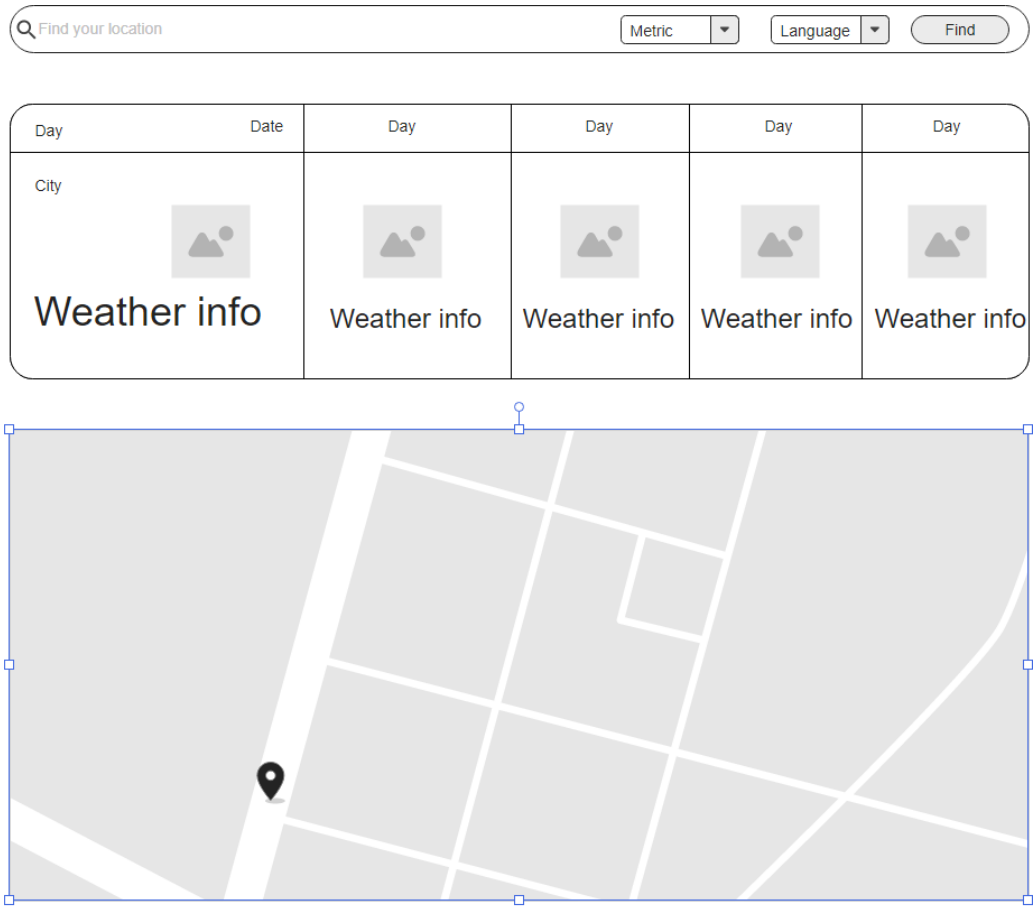

Рисунок 3.14 – Дизайн домашньої сторінки

Сторінка з усіма новинами про погоду (Рисунок 3.15) має містити усі новини, стрічку з можливістю обрати категорію пошуку новин: за країною, за містом, за кількістю новин, боковий сайтбар з користувачами, містами який можна розширити та країнами який теж можна розширити.

Рисунок 3.15 – Дизайн сторінки усіх новин

Сторінка певної новини (Рисунок 3.16) має містити інформацію про певну новину, заголовок та опис.

Рисунок 3.16 – Дизайн певної новини

Сторінка створення новини про погоду (Рисунок 3.17) має містити відповідні поля для створення, також відповідні поля про помилки коли валідація не пройшла успішно. Після успішного створення новини, користувач отримує відповідне повідомлення (Рисунок 3.18).

Рисунок 3.17 – Дизайн сторінки створення новини

Рисунок 3.18 – Дизайн сторінки успішно створеної новини

Висновки до розділу 2

В межах розділу було проаналізовано можливі методики розробки певних проектних рішень для забезпечення повного та успішного виконання функціональних та нефункціональних вимог до ПЗ. Спроектовано логічну модель даних, що представлена остаточною схемою даних у вигляді ER-діаграми.

Також були проаналізовані наявні на ринку технології і прийнято рішення про те, які саме з них необхідно використовувати для успішної та безпроблемної реалізації поставленої задачі.

На основі вимог до ПЗ сконструйовано діаграми та сценарії до системи, що розробляється, а саме сценарії використання у різних формах, діаграму варіантів використання, діаграму діяльності, діаграму класів та діаграму послідовності.

Виходячи з вимог до ПЗ та структури застосунку було спроектовано макети інтерфейсів публічної частини вебзастосунку, а саме домашньої сторінки, сторінки усіх новин, сторінки певної новини та сторінки створення новини про погоду.

3 КОДУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кожен вебзастосунок складається з двох частин. Перша – програмна реалізація невидимої для користувача частини застосунку, що імплементує бізнес-логіку усієї інформаційної системи (back-end). Друга – створення та реалізація візуальних представлень за заздалегідь спроектованими макетами інтерфейсів (front-end).

Фреймворк Laravel реалізує відокремлення представлення від моделі та контролерів за патерном MVC loh, а компоненти фреймворку дозволяють працювати з базою даних та з ресурсами проекту досить просто. Отже, реалізація моделей, контролеру та представлень, робота з базою даних та компіляція стилів і скриптів проходить окремо.

3.1 Робота з базою даних

Майже кожен сучасний веб-застосунок взаємодіє з базою даних. Laravel забезпечує власну підтримку чотирьох баз даних:

- MySQL 5.7 і вище;
- PostgreSQL 9.6 і вище;
- SQLite 3.8.8 і вище;
- SQL Server 2017 і вище.

Laravel робить взаємодію з базами даних надзвичайно простою в різних підтримуваних базах даних, використовуючи систему міграцій та сідерів, зручний Query Builder та Eloquent ORM [9].

Міграції у Laravel представляють собою певний контроль версій для вашої бази даних, що дозволяє вашій команді визначати та ділитися визначенням схеми бази даних програми. Laravel забезпечує підтримку повного циклу робоии з міграціями для створення та обробки таблиць у всіх підтримуваних системах баз даних Laravel [10].

Було розроблено міграції для усіх основних сутностей системи (зазначених у діаграмі класів у попередньому розділі), що розробляється у ході роботи (Додаток А).

Laravel включає можливість заповнення бази даних тестовими даними або даними за замовчуванням за допомогою класів-сідерів. Усі класи-сідери зберігаються в каталозі сідерів [11].

За замовчуванням для розробників визначено клас DatabaseSeeder. З цього класу є можливість використовувати метод call() для запуску інших класів сідерів, наприклад класу сідеру для налаштувань за замовчуванням (Додаток Б), що дозволяє контролювати порядок запуску сідерів.

Laravel також включає можливість заповнення бази даних фейковими даними за допомогою factory та методу faker який предоставляє різноманітні дані (Додаток В).

3.2 Програмна реалізація

Так як при створенні веб-застосунку використовується фреймворк Laravel, то відпадає необхідність ручного створення моделей та контролерів, описаних у базі даних сутностей. Більш того, їх створення відбувається одночасно зі створенням міграцій.

Фреймворк також реалізує зручну роботу з маршрутизацією, тобто, застосунок, отримуючи запит, за складовою цього запиту визначає який метод якого контролеру буде викликаний. Інформація про маршрути вебзастосунку зберігається у файлах маршрутизації в файлі api (Додаток Г).

Коли застосунок визначає, який саме метод виконується, запит передається до цього ж методу конкретного контролеру. У кожному з методів може зберігатися необхідний код обробки та підготовки інформації для подальшого використання.

Наприклад, при надходженні запиту на перегляд останніх новин для відображення їх на домашній сторінці вебзастосунку викликається метод index() контролеру NewsController (Додаток Ґ).

Варто відмітити, що розробник майже цілком відсторонений від реалізації моделей (Додаток Е), тобто, немає необхідності визначати поля, конструктори, геттери чи сеттери для класу моделі, адже це виконується засобами фреймворку. Користувачу необхідно лише реалізувати код з'язку з об'єктами інших класів за допомогою методів belongsTo() (Додаток Д).

3.3 Керівництво адміністратора

Першим кроком у використанні застосунку є процес його встановлення та першого запуску на вебсервері. Так як було зазначено, що запуск відбувається вперше, відбудеться автоматичний запуск необхідних міграцій (Рисунок 3.1) для створення таблиць у базі даних, та заповнювання данними за допомогою сідерів (Рисунок 3.2).

```
pavel@DESKTOP-2M4TBUJ c:\OpenServer\domains\lara\weather-laravel
# php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (17.45ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (35.51ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (28.21ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated:  2019_12_14_000001_create_personal_access_tokens_table (46.50ms)
Migrating: 2022_02_21_122532_create_news_table
Migrated:  2022_02_21_122532_create_news_table (67.32ms)
```

Рисунок 3.2 – Запус команди міграції

```
pavel@DESKTOP-2M4TBUJ c:\OpenServer\domains\lara\weather-laravel
# php artisan db:seed
Seeding: Database\Seeders\UserSeeder
Seeded:  Database\Seeders\UserSeeder (31.00ms)
Database seeding completed successfully.
```

Рисунок 3.3 – Запуск команди для сідерів

3.4 Керівництво користувача

Після успішного завершення процесу створення таблиць та даних, можна вже використовувати вебзастосунок в повному обсязі. При першому візиту вебзастосунку, показується головна сторінка (Рисунок 3.4), де користувач може ввести назву населеного пункту, та дізнатися погоду. Також він може дати дозвіл на надання даних про своє місцезнаходження, та автоматично знайде інформацію про погоду відносно його місцезнаходження.

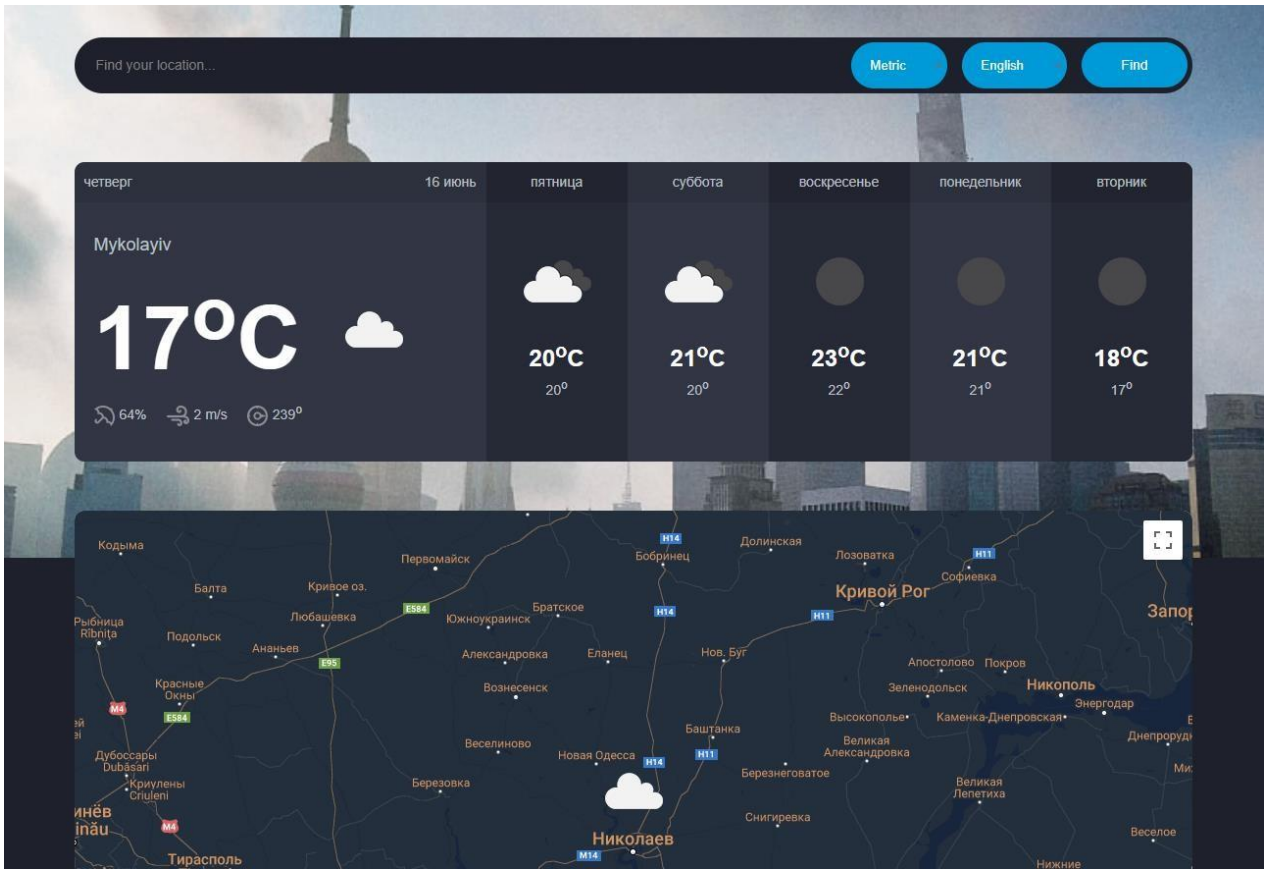

Рисунок 3.4 – Інтерфейс головної сторінки

Також у вебзастосунку мається сторінка з останніми новинами про погоду (Рисунок 3.5) у різних населених пунктах. Користувач має можливість скористатись фільтрами, та вивести новини певної країни, певного населеного пункту або певної кількості новин (Рисунок 3.6).

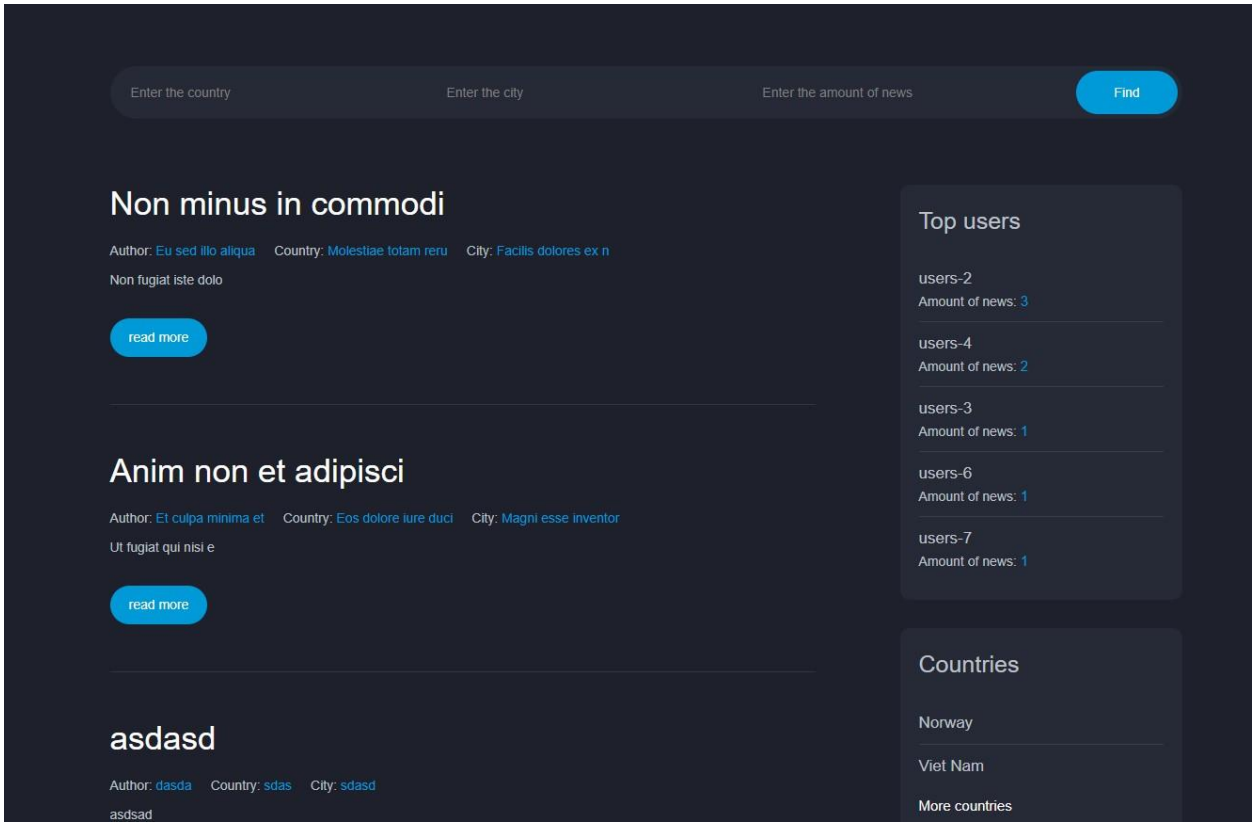

Рисунок 3.5 – Інтерфейс сторінки з новинами

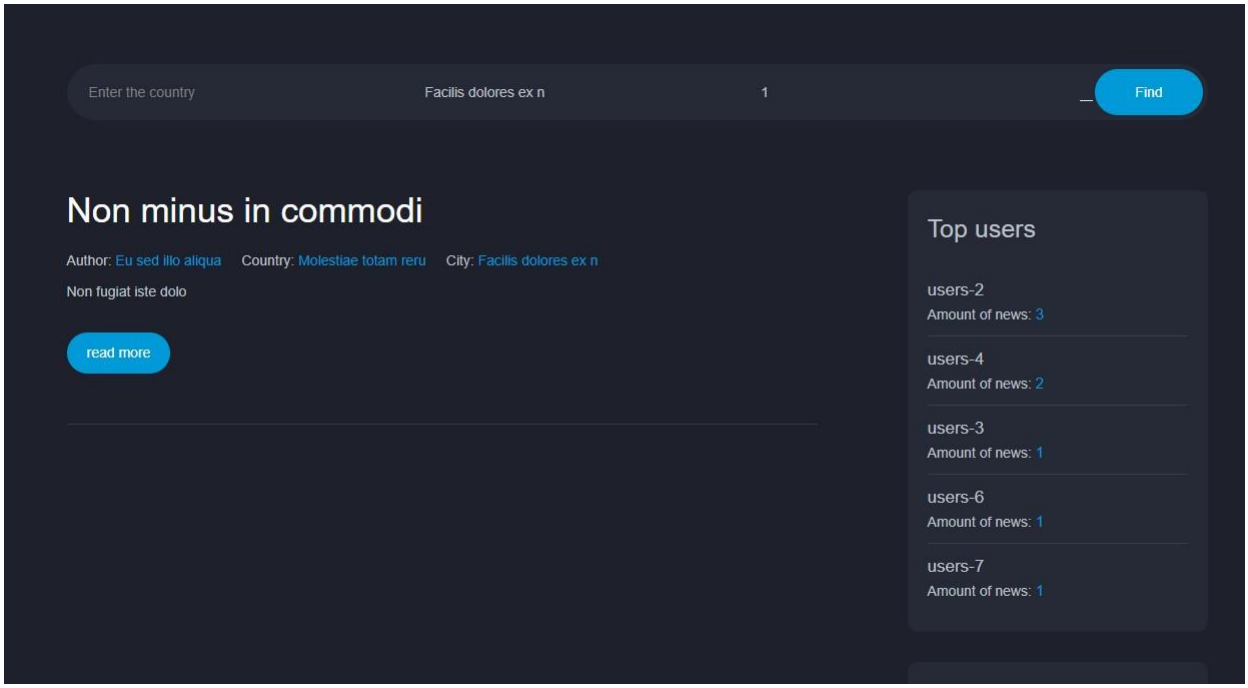

Рисунок 3.6 – Інтерфейс сторінки з новинами з використанням фільтрів

На сторінці ‘News’ також можна перйти до певної новини просто клацнути на заголовок її. Сторінка певної новини (Рисунок 3.7) складається з усіх необхідних полів для новини, а саме: загловок, автор, країні, населений пункт та опис.

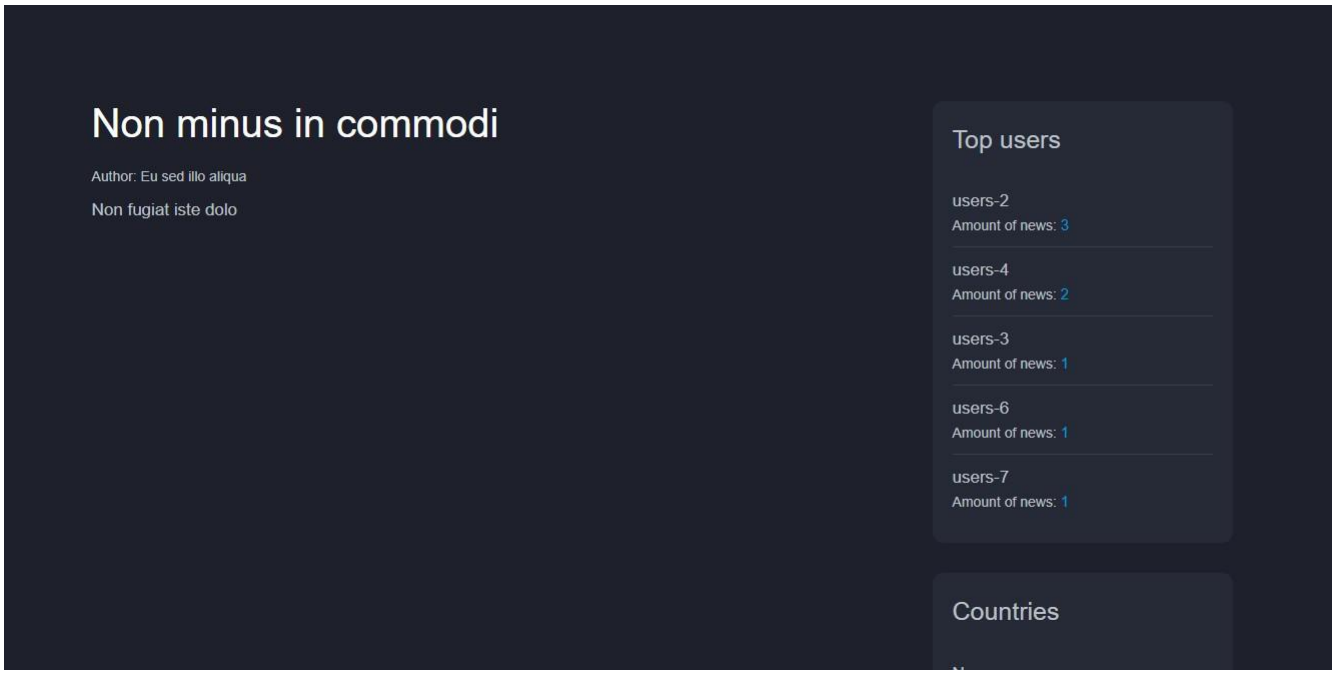

Рисунок 3.7 – Інтерфейс сторінки певної новини

Також користувачу надана можливість створювати новини, треба перейти до сторінки ‘Create News’ (Рисунок 3.8). На даній сторінці маються усі необхідні поля для створення, коли користувач вводить данні, але вони не задовольняють валідацію, з’являються відповідні помилки (Рисунок 3.9), якщо валідація пройшла буде відповідний текст на сторінці (Рисунок 3.10).

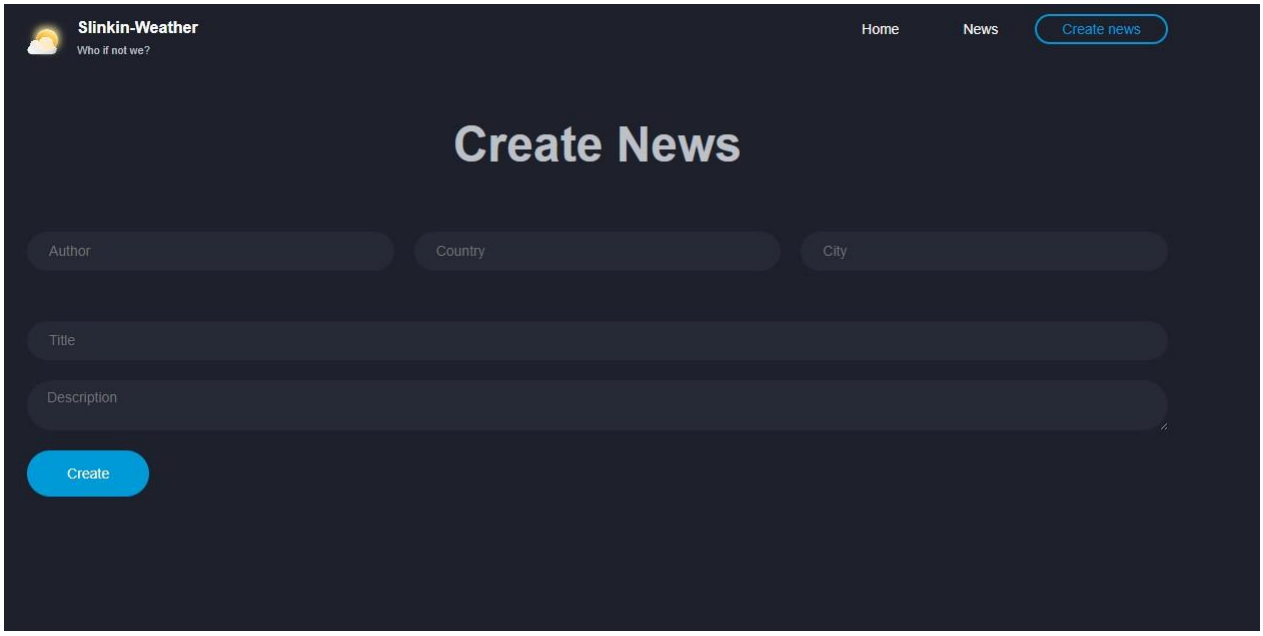

Рисунок 3.8 – Сторінка створювання новини

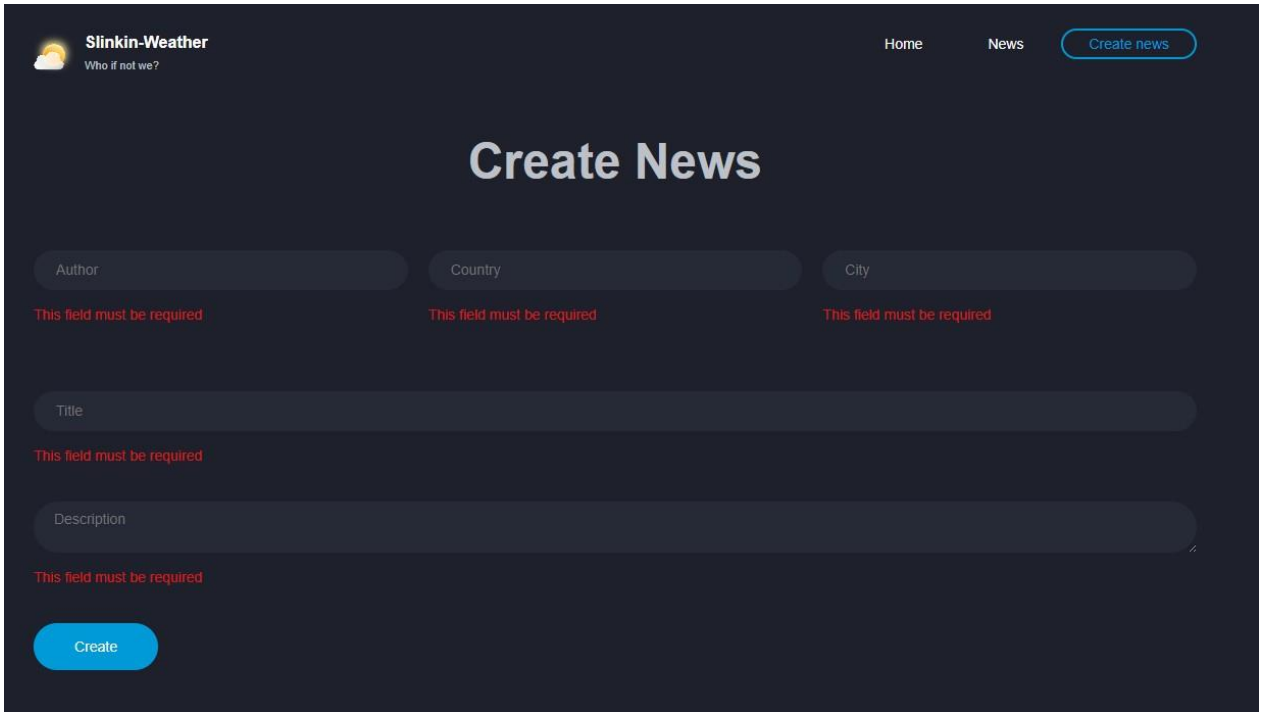

Рисунок 3.9 – Валідація спіймала та вивела помилки

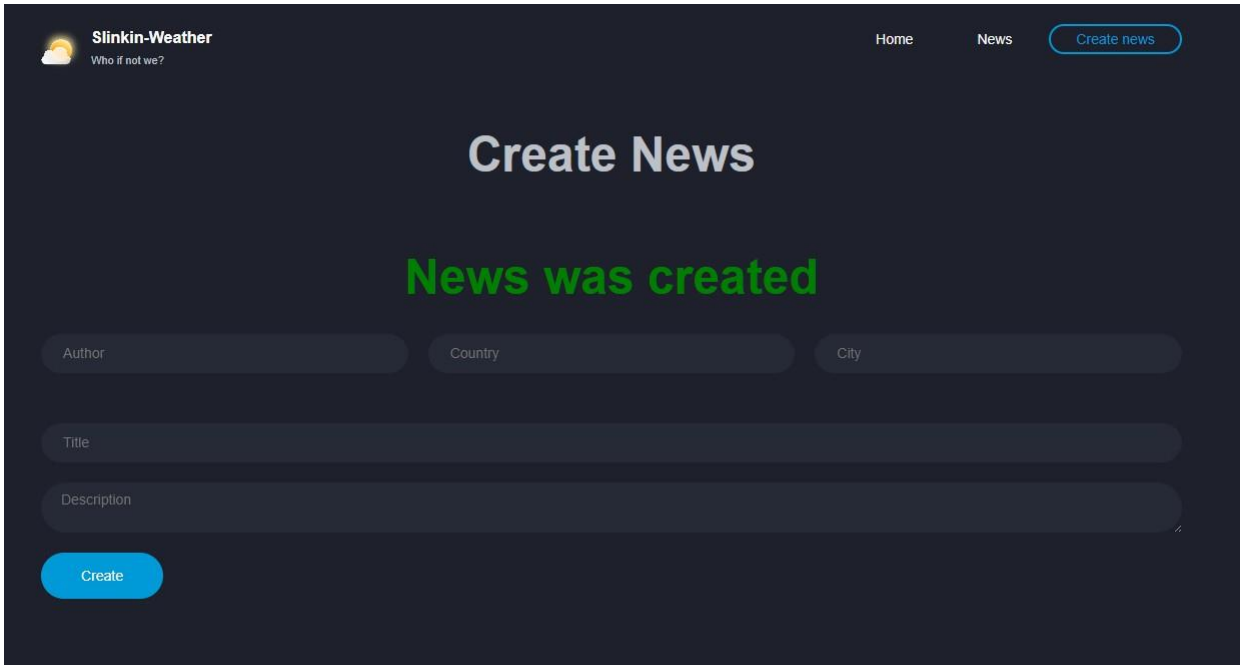

Рисунок 3.10 – Новина успішно створена

Висновки до розділу 3

В межах розділу з використанням обраних мов програмування та технологій було реалізовано вебзастосунок дотримуючись вимог до ПЗ, заздалегідь розроблених моделей, проєктних рішеннь та створених інтерфейсів ПЗ.

Використовуючи функціонал вебзастосунку, було проведено його тестування. За результатами тестування можна зробити висновок, що створена система повністю задовольняє функціональні та нефункціональні вимоги, які були поставлені на початку роботи, а вебзастосунок повністю готовий до використання.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи бакалавра було досягнуто її головної мети, що полягала у спрощення процесу пошуку локації за критеріями погодних умов шляхом розробки вебзастосунку моніторингу погоди та фільтрації за значеннями окремих характеристик. Вищезгаданої мети вдалося досягнути завдяки виконанню усіх поставлених завдань.

Проведено детальний аналіз предметної області та існуючих застосунків для реалізації мети роботи, було проаналізовано їх переваги та недоліки, виявлено основні технологічні особливості та обмеження, проаналізовано основні засоби для вирішення поставлених задач програмним шляхом і досягнення основної мети роботи.

На основі результату проведеного аналізу було проаналізовано наявні на ринку технології для успішної та безпроблемної реалізації поставленої задачі і детально спроектовано логічну модель даних системи, що представляє собою ER-діаграму, яка описує структуру зберігання даних у БД, успішне підключення до API, яка надає основні дані до вебзастосунку, а також сконструйовано діаграми та сценарії до системи, що розробляється. Дотримуючись вимог до структури та функціональності ПЗ було створено макети програмних інтерфейсів.

Результатом виконання роботи є розроблений вебзастосунок моніторингу погодних умов із функцією пошуку локації за критеріями. Кінцевий програмний продукт задоволняє усі поставлені перед ним функціональні та нефункціональні вимоги.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. How to Use an API: веб-сайт. URL: https://technologyadvice.com/blog/information-technology/how-to-use-an-api/ (дата звертання 02.11.2021);

2. Introduction to Vue.js with a Single Page Application (SPA): веб-сайт. URL: https://www.red-gate.com/simple-talk/development/javascript/introduction-to-vue-js-with-a-single-page-application-spa-in-visual-studio/ (дата звертання 10.12.2021);

3. GISMETEO: Погода в Україні: веб-сайт. URL: https://www.gismeteo.ua/ua/ (дата звертання 20.01.2022);

4. What is Vue?. URL: https://vuejs.org/guide/introduction.html (дата звертання 25.01.2022);

5. What is Vuex?: веб-сайт. URL: https://vuex.vuejs.org/ (дата звертання 29.01.2022);

6. The PHP Framework for Web Artisans: веб-сайт. URL: https://laravel.com/docs/9.x (дата звертання 02.02.2022);

7. OpenWeather Weather forecasts, nowcasts and history in fast and elegant way. URL: https://openweathermap.org/ (дата звертання 15.02.2022);

8. How Laravel implements MVC and how to use it effectively: веб-сайт. URL: https://blog.pusher.com/laravel-mvc-use/ (дата звертання 20.02.2022);

9. Eloquent ORM: веб-сайт. URL: https://laravel.com/docs/5.0/eloquent (дата звертання 25.02.2022);

10. Database: Migrations - Laravel - The PHP Framework For Web Artisans: веб-сайт. URL: https://laravel.com/docs/8.x/migrations/ (дата звертання 26.02.2022).

11. Database: Seeding - Laravel - The PHP Framework For Web Artisans: веб-сайт. URL: https://laravel.com/docs/8.x/seeding/ (дата звертання 26.02.2022).

Додаток А
Код створення міграцій

В данному коді виконуєтсья наступне, за допомогою ORM Eloquent, створюється таблиця користувачів у бд з полями: id, name, created_at, updated_at. У наступному класі міграції CreateNewsTabel створюэться таблиця з новинами, які мають наступні поля: title, description, city, country, user_id як зовнішній ключ.

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

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
//            $table->string('email')->unique();
//            $table->timestamp('email_verified_at')->nullable();
//            $table->string('password');
//            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}

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

class CreateNewsTable extends Migration
{
    /**
     * Run the migrations.
     **
     */@return void
```

```
    public function up()
    {
        Schema::create('news', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->text('description');
            $table->string('city');
            $table->string('country');
            $table->foreignId('user_id')
                ->constrained('users')
                ->onDelete('cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('news');
    }
}
```

Додаток Б

Створення сідерів

В класі DatabaseSeeder виконуєтсья наступне, викликаються вказані сідери у методі call, та за допомогою Factory створюється 10 рандомних новин. У класі UserSeeder створюється у таблиці 10 стрічок даних за допомогою ORM Eloquent.

```
namespace Database\Seeders;

use App\Models\News;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
    public function run()
    {
        $this->call([
            UserSeeder::class,
        ]);
        // \App\Models\User::factory(10)->create();
        News::factory(10)->create();
    }
}


namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
```

```
class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        for ($i = 1; $i <= 10; $i++) {
            DB::table('users')->insert([
                'name' => 'users-' . $i,
                'created_at' => now(),
                'updated_at' => now(),
            ]);
        }
    }
}
```

Додаток В

Створення та використання фабрики

В класі NewsFactory беруться та повертаються рандомні дані, за допомогою класу faker, який має різні методи які повертають рандомні дані: цифри, речення, слова, країни і т.д.

```
namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;
use App\Helpers\Helper;

class NewsFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        $country = Helper::getCountriesList();
        $keys = array_keys($country);
        $random = $country[array_rand($keys)];
        return [
            'title' => $this->faker->sentence(4),
            'description' => $this->faker->sentence(100),
            'city' => $this->faker->city(),
            'country' => $random,
            'user_id' => $this->faker->numberBetween($min = 1, $max = 10),
        ];
    }
}
```

Додаток Ґ

Контролер новин у застосунку

В класі NewsController створюється метод index, який використовує ORM, щоб отримати усі новини з БД які будуть відображані у вебзастосунку.

```
use App\Models\News;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;


class NewsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $news = News::orderBy('id', 'DESC')->get();

        foreach ($news as $singleNews) {
            $singleNews['author'] = $singleNews->user->name;
        }
        return $news;
    }
}
```

Додаток Д

Модель новин та її особливості

В класі моделі News вказуються поля з бази даних, також використовуэться метод belongsTo(зворотній один до багатьох) до класу моделі User.

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class News extends Model
{
    use HasFactory;

    protected $fillable = [
        'title',
        'description',
        'city',
        'country',
        'user_id',
    ];
```

```
    public function user() {
        return $this->belongsTo(User::class);

    protected $table = 'news';
    }
}
```

Додаток Е

Модель користувачів та її особливості

В класі User реалізується модель та вказуються поля з бази даних, та скриті поля також.

```
namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
```