МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ЗБЕРІГАННЯ ЗОБРАЖЕНЬ МІСЦЕВОСТІ З ВИКОРИСТАННЯМ YII 2»**

Виконав: студент 2 курсу, групи 8.1218

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

Д.О.Мартинов
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.т.н. Чопоров С.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент зав. кафедри загальної математики, доцент, к.ф.-м.н. Зіновєєв І.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

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

Лісняк А.О.

(підпис)

« » 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Мартинову Дмитру Олександровичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка інформаційної системи для зберігання зображень місцевості з використанням Yii 2

керівник роботи (проекту) Чопоров Сергій Вікторович, доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » травня 2019 року № 811-с

2. Строк подання студентом роботи

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація інформаційної системи для зберігання зображень місцевості

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

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

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.09.2019	Виконано
2.	Збір вихідних даних.	30.09.2019	Виконано
3.	Обробка методичних та теоретичних	10.10.2019	Виконано
	джерел.		
4.	Розробка першого розділу.	25.10.2019	Виконано
5.	Розробка другого та третього розділу.	14.11.2019	Виконано
6.	Оформлення та нормоконтроль	27.12.2019	Виконано
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	15.01.2020	Виконано

Студент _____ _____
(підпис) (ініціали та прізвище)

Керівник роботи _____ _____
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка інформаційної системи для зберігання зображень місцевості з використанням Yii 2»: 62 с., 35 рис., 12 джерел, 1 додаток.

БАЗИ ДАНИХ, ЗОБРАЖЕННЯ, ЗОБРАЖЕННЯ МІСЦЕВОСТІ, ІНФОРМАЦІЙНА СИСТЕМА, FRAMEWORK, GII, PHP, YII 2.

Об'єктом дослідження кваліфікаційної роботи є особливості використання мови програмування PHP та фреймворку Yii 2 для розробки програмного забезпечення.

Предметом дослідження кваліфікаційної роботи є можливості та інструменти для розробки інформаційної системи для зберігання зображень місцевості.

Мета кваліфікаційної роботи – розробити інформаційну систему для зберігання зображень місцевості, з використанням фреймворку Yii 2.

У кваліфікаційній роботі розглядається розробка інформаційної системи для зберігання зображень місцевості. На основі цього матеріалу були розроблені діаграми та схеми програмного продукту та написаний код. Результати роботи можуть бути використані при розробці більш масштабної інформаційної системи для зберігання зображень місцевості.

SUMMARY

Master’s Qualifying Thesis «Development of an Information System for Terrain Images Storing Using Yii 2»: 62 pages, 35 figures, 12 references, 1 supplement.

DATABASES, FRAMEWORK, GII, IMAGE, INFORMATION SYSTEM, MAP, PHP, YII 2.

The object of the study is the features of using the PHP programming language and Yii 2 framework for software development.

The subject of the study is the opportunities and tools for developing an information system for terrain images storing.

Aim of the study: development of an information system for storing terrain images using framework Yii 2.

The qualification thesis deals with the development of an information system for storing terrain images. Based on this material, diagrams and diagrams of the software were drawn and code was written. The results of the work can be used in the development of a larger scale information system.

ЗМІСТ

ВСТУП

Мета кваліфікаційної роботи – розробка інформаційної системи для зберігання зображень місцевості з використанням фреймворку Yii 2.

Об'єктом дослідження кваліфікаційної роботи є особливості використання мови програмування PHP та фреймворку Yii 2 для розробки програмного забезпечення.

Предметом дослідження кваліфікаційної роботи є можливості та інструменти для розробки інформаційної системи.

Кваліфікаційна робота магістра складається зі вступу, основної частини, висновків, переліку посилань (12 найменувань). Основна частина містить три розділи (25 підрозділів):

а) 1 розділ – обирається мова програмування та середовище розробки, розглядається фреймворк Yii 2;

б) 2 розділ – описує проектування інформаційної системи. Він має у собі діаграми та схеми, що моделюють інформаційну систему;

в) 3 розділ – присвячений розробці коду, інтерфейсу та експериментальній перевірці працездатності системи.

1 ДОСЛІДЖЕННЯ І АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис поставленого завдання

У випускний кваліфікаційній роботі потрібно реалізувати інформаційну систему, що зберігає зображення місцевості з використанням фреймворку Yii 2. Вхідними даними будуть виступати координати місцевості та масштаб мапи. Вихідними даними буде запис у базі даних та один доступний для завантаження графічний файл з зображенням місцевості.

1.2 Обґрунтування вибору мови програмування і фреймворку

Для реалізації інформаційної системи буде використовуватися мова програмування PHP, але для підвищення рівня захищеності системи, якості коду та продуктивності роботи системи при розробці буде використовуватися фреймворк цієї мови програмування.

PHP-фреймворки за останній час набрали популярність і стали базовою платформою для розробки веб-застосунків. Використання цих систем, дозволяє економити велику кількість часу, зменшити навантаження на процес розробки, позбавляючи від проблеми повторюваного коду, і швидко створювати якісні програмні продукти. Між тим, використання PHP фреймворків робить процес створення програми значно більш легким і функціональним.

На сьогоднішній день існує величезна кількість PHP-фреймворків, кожний з яких заточений під використанні у розробці сайтів певного типу, до найпопулярніших відносяться:

а) ZendFramework;

б) CakePHP;

в) Symfony;

г) Yii.

Zend framework – це PHP-фреймворк, що створений і підтримується компанією Zend, співробітники якої є безпосередніми авторами мови PHP. Тому він є послідовником традицій і цінностей PHP – базується на простоті, об'єктно-орієнтованих принципах, дружній ліцензії і ретельно тестованому коді із застосуванням Agile методів [7, с. 105].

Можливості:

а) всі компоненти орієнтовані на PHP 5 та E_STRICT – сумісні;

б) вбудований генератор коду;

в) архітектура «використовуй тільки те, що необхідно» з мінімальними залежностями компонентів;

г) використовує розширюваний шаблон проектування MVC, підтримує макети і PHP-скрипти подання за замовчуванням;

д) підтримує безліч різних баз даних, включаючи MySQL, Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL, SQLite, and Informix Dynamic Server;

е) спеціальні класи для створення, відправлення, отримання email з допомогою mbox, Maildir, POP3 та IMAP4;

ж) гнучка система кешування з підтримкою безлічі сховищ.

Недоліки:

а) повільний без кешування;

б) повільна швидкість розвитку;

в) відсутній ORM.

CakePHP є фреймворком для PHP, який надає розширену архітектуру для розробки, обслуговування і розгортання web-додатків. Він використовує відомий шаблон проектування MVC, як і в об'єктно-реляційних фреймворках. Основною парадигмою CakePHP є збільшення продуктивності розробки, і, як наслідок, допомога програмісту у вигляді зменшенні обсягу написання коду.

Можливості:

а) диспетчер URL із застосуванням регулярних виразів;

б) генерація всього коду за схемою бази даних;

в) перевірка форм;

г) компоненти для авторизації, обмеження доступу (ACL), управління сесіями, cookies, подання деревоподібної інформації (у вигляді Nested Sets);

д) хелпери (компоненти) для генерації та заповнення форм, поділу на сторінки (paginate), управління кешем, JavaScript (в тому числі і AJAX);

е) генерація SQL-запитів, в тому числі для таблиць з відносинами один до багатьох і багато до багатьох, ORM;

ж) консольна інтеграція, клас Shell і завдання Task;

з) плагіни (як окремі програми), компоненти і поведінки;

Недоліки:

а) нестійкість до CSRF-атак;

б) строгі угоди з іменування;

в) низька швидкість розвитку.

Symfony – PHP фреймворк, який має велику бібліотеку класів, написаний на PHP 5. Архітектура має корисні компоненти та інструменти, призначені для створення складних веб-додатків. Symfony – це вільний каркас, написаний на PHP 5, який використовує патерн Model-View-Controller (MVC). Symfony пропонує швидку розробку і керування веб-додатками, що дозволяє легко вирішувати рутинні завдання веб-програміста [8, с. 106].

Перваги:

а) підтримує безліч баз даних (MySQL, PostgreSQL, SQLite, або будь-яка інша PDO-сумісна СУБД);

б) вбудовані класи для роботи з email;

в) гнучка система шаблонів у поданні;

г) вбудований кодогенератор;

д) висока продуктивність.

Недоліки:

а) підходить тільки для великих проектів;

б) відсутність вбудованої ORM.

Yii – це високопродуктивний компонентний PHP-фреймворк, призначений для швидкої розробки сучасних веб-додатків. Завдяки його компонентній структурі і відмінній підтримці кешування, фреймворк особливо підходить для розробки таких великих проектів як портали, форуми, системи керування вмістом (CMS), інтернет-магазини або RESTful-додатки.

На даний момент існує дві версії Yii: 1.1 та 2.0. Версія 1.1 є попереднім поколінням і знаходиться у стані підтримки. Версія 2.0 – це поточне покоління фреймворку, що використовує останні технології і протоколи, такі як Composer, PSR, простори імен, трейти і багато іншого. На цій версії будуть зосереджені основні зусилля кваліфікаційної роботи. Для роботи Yii 2.0 необхідно мати PHP версії 5.4.0 та вище. Після встановлення Yii ми отримуємо як фреймворк, так і шаблон проекту. Шаблон проекту – це робочий проект Yii, в якому реалізовано деякий базовий функціонал, такий як система входу/виходу користувачів, форма зворотнього зв’язку [9, с. 64]. Його код організовано в рекомендований спосіб.

Основні можливості і переваги Yii:

а) інтерфейси DAO і ActiveRecord для роботи з базами даних (PDO);

б) підтримка інтернаціоналізації;

в) перехоплення і обробка помилок;

г) україномовна документація;

д) парадигма модель-вид-контролер;

е) генерація базового PHP-коду для CRUD-операцій (скаффолдинг);

ж) підтримка тем оформлення для їх легкої зміни;

з) використання AJAX і інтеграція з jQuery;

и) низький поріг входження;

к) автоматичне тестування;

На основі проведеного аналізу фреймворк Yii має більше можливостей і переваг, а також не має істотних недоліків і саме тому підходить для реалізації інформаційної системи для зберігання зображень місцевості в порівнянні з іншими популярними фреймворками мови програмування PHP.

1.3 Аналіз середовища розробки

Перед розробкою інформаційної системи було проаналізовано декілька найбільш поширених середовищ розробки програмного забезпечення, таких як:

а) NetBeans;

б) JetBrains PhpStorm;

в) Qt Creator.

NetBeans IDE – вільне інтегроване середовище розробки (IDE) для мов програмування Java, JavaFX, C/C++, PHP, JavaScript, HTML5, CSS3, Python, Groovy, C#. Середовище може бути встановлене і для підтримки окремих мов, і у повній конфігурації. Також за допомогою різних розширень можна збільшувати кількість підтримуваних мов програмування та можливостей їх форматування. Середовище розробки NetBeans по замовчуванню підтримує розробку для платформ J2SE і J2EE.

JetBrains PhpStorm – платний інтелектуальний редактор для PHP, HTML і JavaScript з можливостями аналізу коду на льоту, запобігання помилок у коді і автоматизованими засобами рефакторинга для PHP і JavaScript.

а) підтримка PHP 5.3, 5.4 та 5.5, включаючи генератори, співпрограми, простори імен, замикання, типажі, синтаксис коротких масивів, доступ до члена класу при інстанціюванні, розіменування масиву при виклику функції, бінарні літерали, вираження в статичних виклики тощо. PhpStorm може використовуватися як для сучасних, так і для традиційних проектів на PHP;

б) підтримка стандартів оформлення коду (PSR1/PSR2, Drupal, Symfony2, Zend);

в) детектор дубльованого коду;

г) PHP Code Sniffer (phpcs), котрий перевіряє код на льоту;

д) рефакторинги (перейменування, введення змінної/константи/поля, вбудовування змінної);

е) підтримка редагування шаблонів Smarty (підсвічування синтаксичних помилок, автодоповнення функцій і атрибутів Smarty, автоматична вставка парних дужок, лапок і закриваючих тегів тощо).

Qt Creator – середовище розробки, призначене для створення кросплатформових додатків з використанням бібліотеки Qt. Редактор підтримує розробку як на класичних мовах програмування C++, так і на мовах особливого призначення QML, JavaScript. Структура і візуальні параметри інтерфейсу задаються CSS-подібними блоками. Qt Creator може бути використаний для GCC або Microsoft VC++ в якості компілятора і GDB як зневаджувач. Для Windows-версій бібліотека комплектується заголовними і об'єктними файлами MinGW, компілятором, об'єктними та заголовними файлами. Також редактор володіє розширеним списком функцій для полегшення розробки систем та додатків для різних операційних систем.

Інтерфейс та кольорову схему IDE можна налаштовувати відповідно до будь-яких вподобань користувача, а також можуть бути завантажені із інших спеціалізованих ресурсів.

Виходячи з вище сказаного, для реалізації поставлених задач для нас найкраще підходить середовище розробки – «NetBeans», адже з поміж інших він найкраще заточений під розробку web-додатків, безкоштовний а також є зручним та швидким у використанні.

1.4 Вибір системи управління базами даних для реалізації

База даних – це сукупність взаємозв’язаних даних, що зберігаються разом та відповідають наступним властивостям:

а) дані логічно пов’язані між собою і несуть відповідну інформацію;

б) структура баз даних звичайно відповідає тому специфічному набору даних, які вона містить;

в) бази даних відображають тільки окремі аспекти реального світу, що дає змогу визначити їх як «мікросвіт».

Система управління базами даних (СУБД) поєднує відомості з різних джерел в одній реляційній базі даних. Створювані форми, запити і звіти дозволяють швидко й ефективно обновляти дані, отримувати відповіді на питання, здійснювати пошук потрібних даних, аналізувати дані, друкувати звіти, діаграми і поштові наклейки.

Розробка структури БД передбачає повний опис всіх атрибутів сутностей (полів таблиць). Кожна таблиця – масив з однорідних елементів, які прийнято називати записами. Запис – неподільна одиниця інформації в БД, але можна сформувати такий запит, щоб отримати якусь частину цієї інформації. Запис може містити в собі одну або кілька іменованих полів, що задаються при створенні таблиці. Кожне з полів має певний тип (наприклад, ціле число, текст та інші). У всі таблиці можна додавати записи, видаляти їх, робити пошук за таблицями і виводити потрібну інформацію. Всі ці дії виконуються за допомогою SQL (Structured Query Language) запитів SELECT, DELETE, CREATE, INSERT.

На сьогоднішній день існує досить багато систем управління базами даних. Було розглянуто деякі з них.

Oracle Database або Oracle RDBMS. Об'єктно-реляційна система управління базами даних компанії Oracle Corporation. Ця СУБД забезпечує ефективне, надійне і безпечне управління даними таких критично важливих для бізнесу додатків, як онлайнові середовища, виконує масштабну обробку транзакцій (OLTP), сховища даних з високою інтенсивністю потоку запитів, а також ресурсномісткі інтернет-додатки. Редакція Oracle Database Enterprise Edition надає інструментальні засоби і функції, що забезпечують відповідність вимогам сучасних корпоративних додатків в області

доступності та масштабованості. Ця редакція містить всі компоненти Oracle Database, а також допускає розширення за допомогою придбання додаткових модулів та програм.

Система Oracle Database дозволяє звертатися до даних з будь-якого додатку, розробленого із застосуванням технологій Microsoft. NET, Visual Studio та веб-додатків. Основною умовою є лише наявність справних бібліотек, що дають змогу підключатися до серверу бази даних

Oracle. Oracle Database є комерційною СУБД, але є її безкоштовна версія, яку можна без проблем скачати прямо з офіційного сайту компанії.

Microsoft SQL Server. Система Microsoft SQL Server відштовхується від концепції платформи даних Майкрософт: вона спрощує управління будь-якими даними в будь-якому місці і в будь-який момент часу. Вона дозволяє зберігати в базах даних інформацію, отриману з структурованих, напівструктурованих і неструктурованих джерел, таких як зображення та музика. У SQL Server 2008 є великий набір інтегрованих служб, які розширюють можливості використання даних: можна складати запити, виконувати пошук, проводити синхронізацію, робити звіти, аналізувати дані. Всі дані зберігаються на основних серверах, що входять до складу центру обробки даних. До них здійснюється доступ з настільних комп'ютерів і мобільних пристроїв. Таким чином, можна повністю контролювати дані незалежно від того, де вони збережені.

Система SQL Server дозволяє звертатися до даних з будь-якого додатку, розробленого із застосуванням технологій Microsoft. NET та Visual Studio, а також в межах сервісно-орієнтованої архітектури і бізнес-процесів – через Microsoft BizTalk Server. SQL Server дозволяє створити надійну, продуктивну, інтелектуальну платформу, що відповідає всім вимогам по роботі з даними. SQL Server є комерційною СУБД.

PostgreSQL. Безкоштовна об'єктно-реляційна система управління базами даних. Система PostgreSQL заснована на ядрі, створеному безліччю розробників. У подібних випадках розумно зосередитися на оснащенні

системи новими можливостями, але не займатися оптимальним їх втіленням, оскільки у випадку виникнення необхідності завжди можна буде повернутися до оптимізації відповідних ділянок коду.

MySQL. Безкоштовна система управління базами даних. MySQL є власністю компанії Oracle Corporation, що отримала її разом з поглиненою Sun Microsystems, що здійснює розробку і підтримку програми. Розповсюджується під GNU General Public License або під власною комерційною ліцензією. Крім цього, розробники створюють функціональність за замовленням ліцензійних користувачів.

MySQL є рішенням для малих і середніх додатків. Входить до складу серверів WAMP, LAMP і в портативні збірки серверів Denver, XAMPP. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми [12, с. 50].

Оскільки інтерфейс системи буде веб-додатком, потрібно обрати СУБД, з якою можуть співпрацювати мови програмування для написання веб-додатків. Вибір було зупинено на MySQL.

Висока продуктивність MySQL на вузькому колі задач – прямий наслідок її функціональної простоти.

1.5 Структура Yii

Yii використовує шаблон проектування модель-представлення-контролер (MVC, Model-View-Controller), який широко застосовується в веб-програмуванні [11, с. 43].

MVC призначений для поділу бізнес-логіки і користувача інтерфейсу, щоб розробники могли легко змінювати окремі частини програми, не зачіпаючи інші. В архітектурі MVC модель надає дані і правила бізнес-логіки, уявлення відповідає за інтерфейс користувача (наприклад, текст, поля

введення), а контролер забезпечує взаємодію між моделлю і представленням. Крім цього, Yii використовує фронт-контролер, який ще називають додатком (application), що інкапсулює контекст обробки запиту. Додаток збирає інформацію про запит і передає її для подальшої обробки відповідного контролера.

1.6 Життєвий цикл додатку Yii

Життєвий цикл додатку при обробці користувальницького запиту:

а) попередня ініціалізація додатку через CApplication::preinit();
б) ініціалізація обробника помилок;
в) реєстрація компонентів ядра;
г) завантаження конфігурації додатку;
д) ініціалізація додатку CApplication::init();
е) реєстрація поведінок додатку;
ж) завантаження статичних компонентів додатку;
з) виклик події onBeginRequest;
и) обробка запиту;
к) збір інформації про запит;
л) створення контролера;
м) запуск контролера;
н) виклик події onEndRequest.

1.7 Ключові компоненти додатку Yii

Yii визначає набір компонентів ядра, які надають можливості, необхідні
для більшості веб-додатків. Наприклад, компонент request використовується для збору інформації про запит користувача і надає різну інформацію, таку як

URL і cookies. Ставлячи властивості компонентів, можна змінювати стандартну поведінку Yii практично як завгодно.

Далі буде перераховано ключові компоненти, які використовує клас CWebApplication:

а) CAssetManager – керує публікацією файлів ресурсів (asset files);

б) CAuthManager – контролює доступ на основі ролей (RBAC);

в) CCache – надає можливості кешування даних;

г) CClientScript – керує клієнтськими скриптами (javascripts і CSS);

д) CPhpMessageSource – надає переклади системних повідомлень Yii-фреймворка;

е) CDbConnection – обслуговує з'єднання з базою даних;

ж) Cerrorhandler – оброблює не піймані помилки і виключення PHP;

з) CFormatter – форматує дані для їх подальшого відображення;

и) CPhpMessageSource – надає переклади повідомлень, використаних в Yii-додатку;

к) CHttpRequest – надає інформацію, що відноситься до запиту користувача;

л) CSecurityManager – забезпечує функціональність, пов'язану з безпекою (наприклад, хешування, шифрування);

м) CHttpSession – забезпечує функціональність, пов'язану із сесіями;

н) CStatePersister – надає метод для збереження глобального стану;

о) CUrlManager – надає функції парсинга і формування URL;

п) CWebUser – надає ідентифікаційну інформацію поточного користувача;

р) CThemeManager – управляє темами оформлення.

1.8 Реалізація MVC в Yii

Контролер (controller) – це екземпляр класу CController або успадкованого від нього класу. Він створюється об’єктом додатку тоді, коли користувач робить відповідний запит. Під час запуску контролер виконує відповідну дію, що зазвичай передбачає створення відповідних моделей і рендеринг необхідних представлень. У найпростішому випадку дія — це метод класу контролера, назва якого починається з action.

У контролера є дія за замовчуванням, яка виконується у випадку, коли користувач не вказує дію при запиті. За замовчуванням ця дія називається index. Змінити її можна шляхом встановлення значення CController::defaultAction.

Модель (model) – це екземпляр класу CModel або класу, успадкованого від нього. Модель використовується для зберігання даних або застосованих до них бізнес-правил.

Модель становить собою окремий об’єкт даних. Це може бути запис таблиці бази даних або HTML-форма з полями для вводу даних. Кожне поле об’єкту даних представляється атрибутом моделі, кожний атрибут має мітку та може бути перевірений на коректність за допомогою набору правил.

Yii надає два типи моделей: модель форми та Active Record. Обидва типи є розширенням базового класу CModel.

Модель форми – це екземпляр класу CFormModel. Вона використовується для зберігання даних, введених користувачем. Як правило, ми отримуємо ці дані, обробляємо, а потім позбавляємося від них. Наприклад, на сторінці авторизації модель такого типу може бути використана для представлення інформації про ім’я користувача та пароль. Детальний опис роботи з формами подано у розділі Робота з формами.

Active Record (AR) – це шаблон проектування, який використовується для абстрагування доступу до бази даних у об’єктно-орієнтованій формі. Кожен об’єкт AR являє собою екземпляр класу CActiveRecord або класу, успадкованого від нього, та представляє окремий рядок у таблиці бази даних. Поля цього рядку відповідають властивостям AR-об’єкту.

Представлення – це PHP-скрипт, який складається переважно з елементів користувальницького інтерфейсу. Він може охоплювати вирази PHP, проте рекомендується, щоб ці вирази не змінювали дані і залишалися відносно простими. Згідно з концепцією розділення логіки та представлення, більша частина коду логіки повинна бути розміщена у контролері або моделі, а не у скрипті представлення.

Представлення має ім’я, яке використовується з метою ідентифікації файлу скрипта представлення у процесі рендерингу. Ім’я представлення повинно співпадати із назвою файлу представлення. Наприклад, для представлення edit відповідний файл скрипта повинен називатися edit.php. Щоб відобразити представлення необхідно викликати метод CController::render(), вказавши ім’я представлення, при цьому метод спробує виявити відповідний файл у директорії protected/views/ControllerID.

Всередині скрипта представлення екземпляр контролера доступний через $this. Таким чином, ми можемо звернутися до властивостей контролера із коду представлення: $this->propertyName.

1.9 Типовий запит Yii

Розглянувши структуру додатків у Yii, перейдемо до специфіки взаємодії кожного компонента із системою для виконання базового запиту.

Наступна діаграма описує типову послідовність процесу обробки користувальницького запиту додатком. Детальний опис послідовності процесу обробки користувальницького запиту додатком:

а) користувач здійснює запит за допомогою URL і веб-сервер обробляє його, запускаючи скрипт ініціалізації index.php;

б) скрипт ініціалізації створює екземпляр додатку і запускає його на виконання;

в) додаток отримує детальну інформацію про запит користувача від компонента додатка request;

г) додаток визначає запитані контролер і дію за допомогою компонента urlManager. В даному прикладі контролером буде post, що відноситься до класу PostController, а дією – show, суть якого визначається контролером;

д) додаток створює екземпляр запитуваного контролера для подальшої обробки запиту користувача. Контролер визначає відповідність дії show методу actionShow в класі контролера. Далі створюються і застосовуються фільтри (наприклад, access control, benchmarking), пов'язані з даним процесом, і, якщо фільтри дозволяють, дія виконується;

е) дія зчитує з бази даних модель Post з ID рівним 1;

ж) дія підключає уявлення show, передаючи в нього модель Post;

з) подання отримує і відображає атрибути моделі Post;

и) подання підключає деякі віджети;

к) сформоване уявлення вставляється в макет сторінки;

л) дія завершує формування уявлення і виводить результат користувачеві.

1.10 Процес розробки додатку з використанням Yii

Розглянувши фундаментальні концепції Yii, доцільно описати загальний процес створення веб-додатків з використанням фреймворку. Основні етапи при розробці додатку виглядають так:

а) створення структури директорій. Утиліта Yii може бути використана для того, щоб прискорити цей процес;

б) конфігурація додатку шляхом модифікації файлу конфігурації програми. Цей етап також може зажадати написання деяких компонентів програми (наприклад, компонента управління користувачами);

в) створення класу моделі для кожного використовуваного типу даних. Для автоматичної генерації всіх необхідних моделей Active Record можна скористатися інструментом Gii;

г) створення класу контролера для кожного типу запиту. Класифікація призначених для користувача запитів залежить від поточних вимог. У загальному випадку, якщо клас моделі використовується користувачем, повинен існувати відповідний клас контролера. Утиліта Gii також може автоматизувати цей процес;

д) створення дій і уявлень. Саме тут і відбувається основна робота;

е) конфігурація необхідних фільтрів для дій в класах контролерів;

ж) створення тем оформлення при необхідності;

з) переклад повідомлень в разі, коли потрібна локалізація програмного продукту;

и) виявлення даних і уявлень, які можуть бути закешовані, і застосування відповідних технік кешування;

к) налаштування продуктивності і розгортання.

1.11 Висновки

У першому розділі кваліфікаційної роботи було проаналізовано основні засоби та методи створення інформаційної системи для зберігання зображень місцевості, в результаті чого для реалізації було обрано мови програмування PHP, фреймворк – Yii2, JavaScript, мову розмітки HTML5 та каскадну таблицю стилів CSS3. Також було обране, оптимальне для реалізації задач такого типу, середовище розробки – «NetBeans», адже з поміж інших він найкраще заточений під розробку web-додатків та є зручним та швидким у використанні.

Розглянуто фреймворк Yii, описані основні теоретичні відомості, поставлена задача і визначено перелік функцій, які необхідно реалізувати в

програмному продукті. Yii має багато можливостей які виділяють цей фреймворк на тлі інших і роблять розробку з його допомогою дуже швидкою і приємною:

а) відмінна підтримка. Для україномовних користувачів на офіційному форумі Yii створений власний розділ де завжди можна отримати відповіді на питання щодо фреймворку.

б) підтримка ООП. Фреймворк повністю заточений під п'яту версію php, що дозволяє підтримувати весь функціонал при об'єктно-орієнтованому програмуванні.

в) генератор коду. Yii надає відмінний вбудований генератор вихідного коду. Вказавши лише основні параметри – Yii генерує для вас загальну структуру програми яка буде містити всі необхідні моделі, контролери та відображення.

г) захист. Всі стандартні класи Yii заточені під високий рівень безпеки що при вмілому зверненні дозволяють повністю захистити сайт від Sql-Inj, XSS, CSRF і інших атак.

2 ПРОЕКТУВАННЯ СТРУКТУРИ ТА АРХІТЕКТУРИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Технічне завдання

Вивчивши поставлену задачу, дослідивши її актуальність і проаналізувавши чинні інформаційні системи, був складений список необхідних вимог для розроблюваного програмного продукту.

Інформаційна система для зберігання зображень місцевості розробляється на підставі наказу № 811-с «Про затвердження тем дипломних робіт студентів II курсу освітньо-кваліфікаційного рівня магістр денної форми навчання ЗНУ від 29.05.2019 року.

Інформаційна система для зберігання зображень місцевості отримує назву «MapSaver».

У програмному продукті необхідно реалізувати вибір певної місцевості на карті, зберігання координат та масштабу в базі даних, завантаження графічного файлу з зображенням місцевості, систему реєстрації та входу до системи.

Програмне забезпечення буде написано мовою програмування PHP з використанням фреймворку Yii 2 і вимагатиме для свого запуску лише веб-браузер: Google Chrome, Opera, Mozilla Firefox, Edge.

Вхідними даними виступають масштаб та координати карти. Вихідними – запис у базу даних та графічний файл зображення місцевості.

2.2 Алгоритм роботи веб-сервісу

На рисунку 2.1 зображується алгоритм роботи інформаційної системи для зберігання зображень місцевості

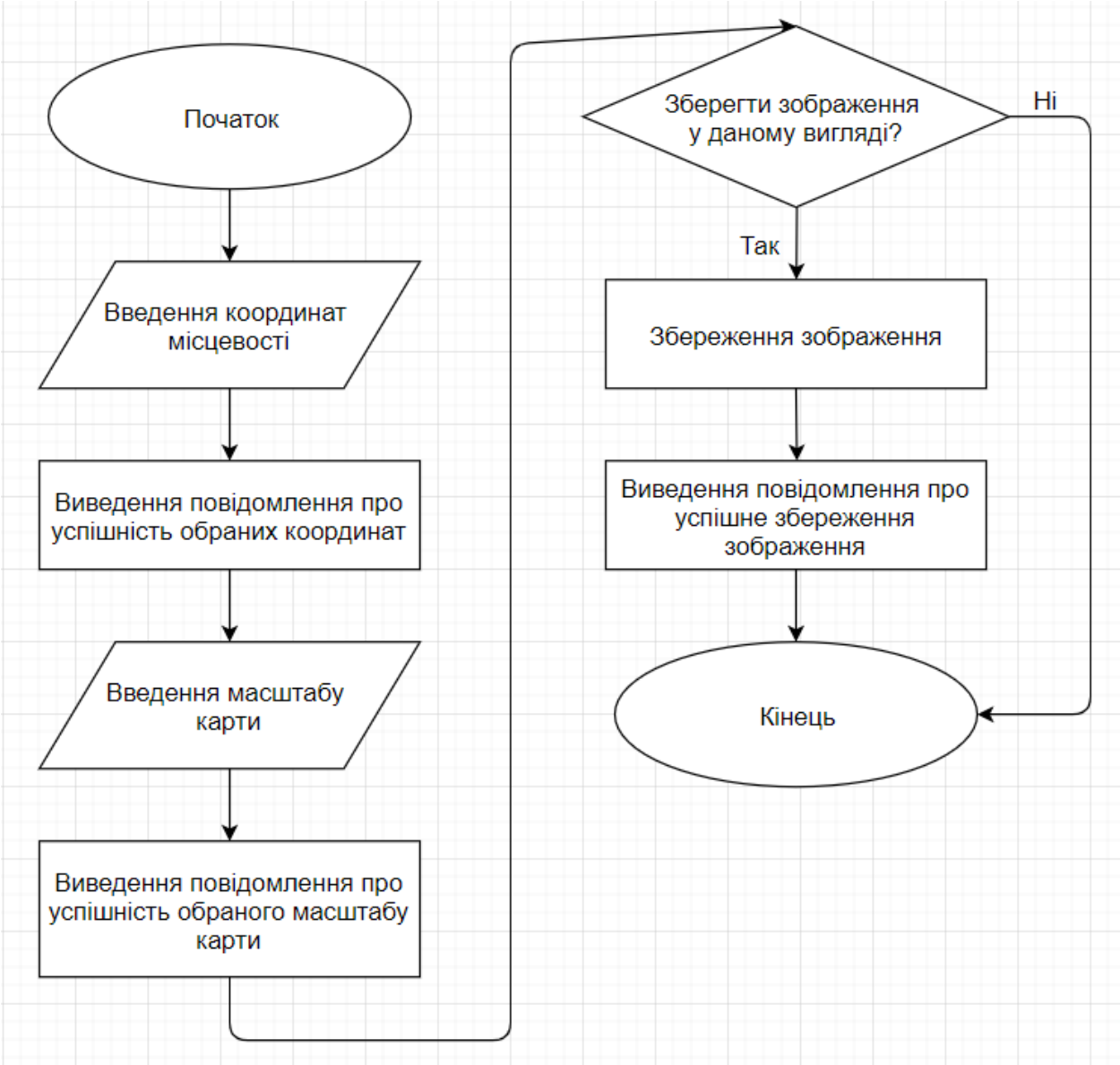

Рисунок 2.1 – Алгоритм роботи

На алгоритмі представлено зберігання місцевості у базу даних, починаючи з введення вхідних даних у вигляді координат та масштабу з супроводжуючими повідомленнями і закінчуючи додаванням даних в БД та збереженням зображення.

2.3 Діаграма станів

На рисунку 2.2 зображена діаграма станів, яка дозволяє зобразити всю

роботу інформаційної системи для зберігання зображень місцевості на кожному її етапі (в кожному її стані).

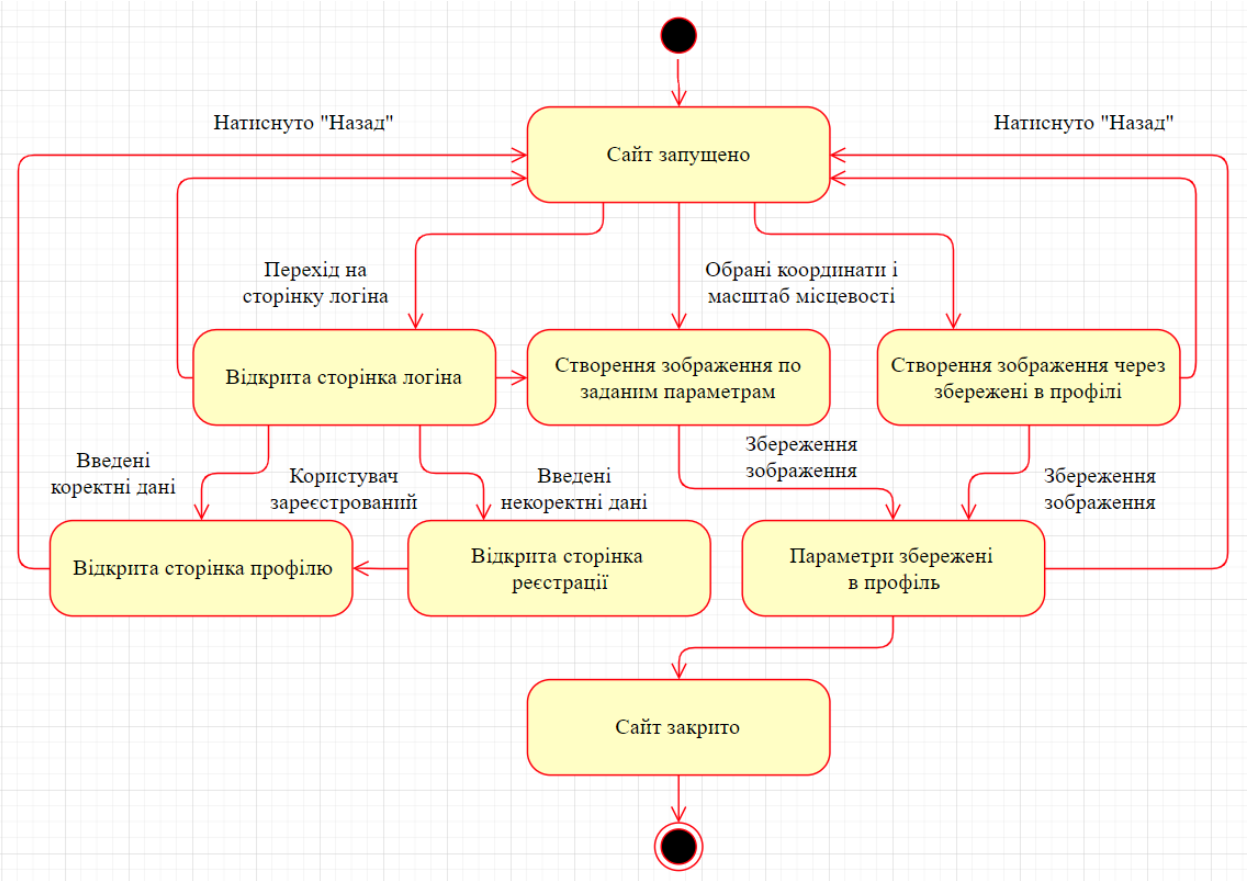

Рисунок 2.2 – Діаграма станів

На діаграмі зображені вісім станів. Після запуску сайту рекомендується перейти на сторінку логіну та авторизуватись або зареєструватись, після чого можна почати роботу з системою, а саме обрати місцевість для збереження самому або через історію пошуку.

2.4 Функціональна схема

На рисунку 2.3 зображено функціональну схему проекту, що містить у собі систему зберігання зображення місцевості та три підсистеми для роботи з картами, графічним файлом та авторизацією.

Рисунок 2.3 – Функціональна схема

Підсистеми авторизації і зберігання зображень пов’язані з базою даних, оскільки зберігають зображення та розпізнають користувача. Підсистема повідомлень повідомляє про все, що відбувається на сайті.

Вхідними даними виступають координати та масштаб місцевості, безпосередньо пов’язані з сервісом Google Maps.

На виході окрім запису в базі даних, при необхідності отримуємо графічний файл.

2.5 Діаграма прецедентів

На рисунку 2.4 зображено діаграму прецедентів, на якій користувач взаємодіє з інформаційною системою для зберігання зображень місцевості.

Користувач інформаційної системи може обирати координати та масштаб місцевості, входити та виходити із інформаційної системи, вносити та завантажувати дані в базу даних, продивитись правильність виконаних дій та завантажувати зображення у вигляді графічного файлу на свій пристрій

У системи немає адміністратора або якогось іншого актора, окрім користувача, бо систему вже зібрано і вносити зміни може лише програміст.

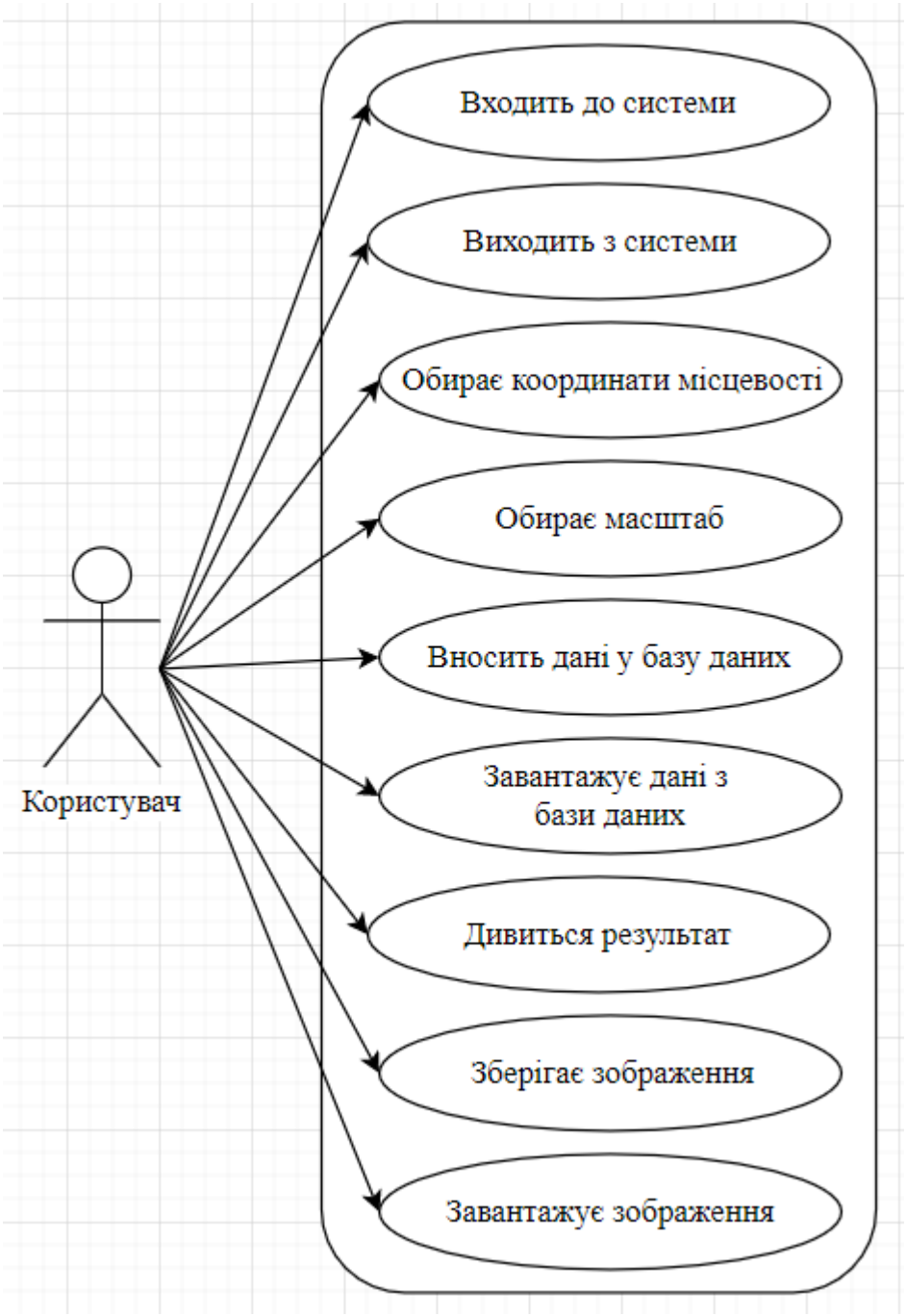

Рисунок 2.4 – Діаграма прецедентів

2.6 Діаграма бази даних

На рисунку 2.5 зображена діаграма бази даних, на якій показано відношення таблиць users і locations.

У базі даних використовується відношення один до багатьох, оскільки

у одного користувача може бути багато збережених локацій.

В свою чергу кожна локація збережена лише у одного авторизованого користувача.

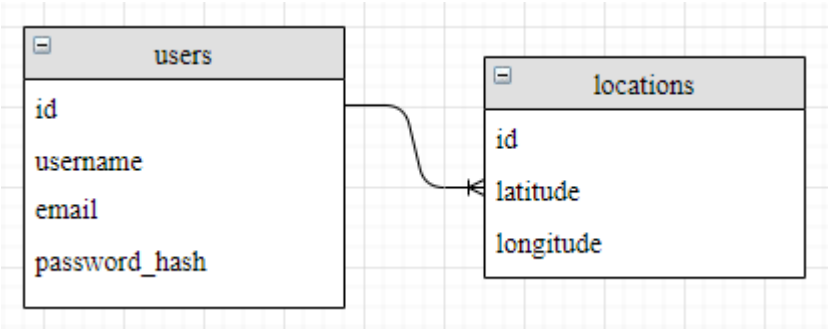

Рисунок 2.5 – Діаграма бази даних

2.7 Діаграма класів

На рисунку 2.6 зображено діаграму класів, що містить у собі класи User та Location.

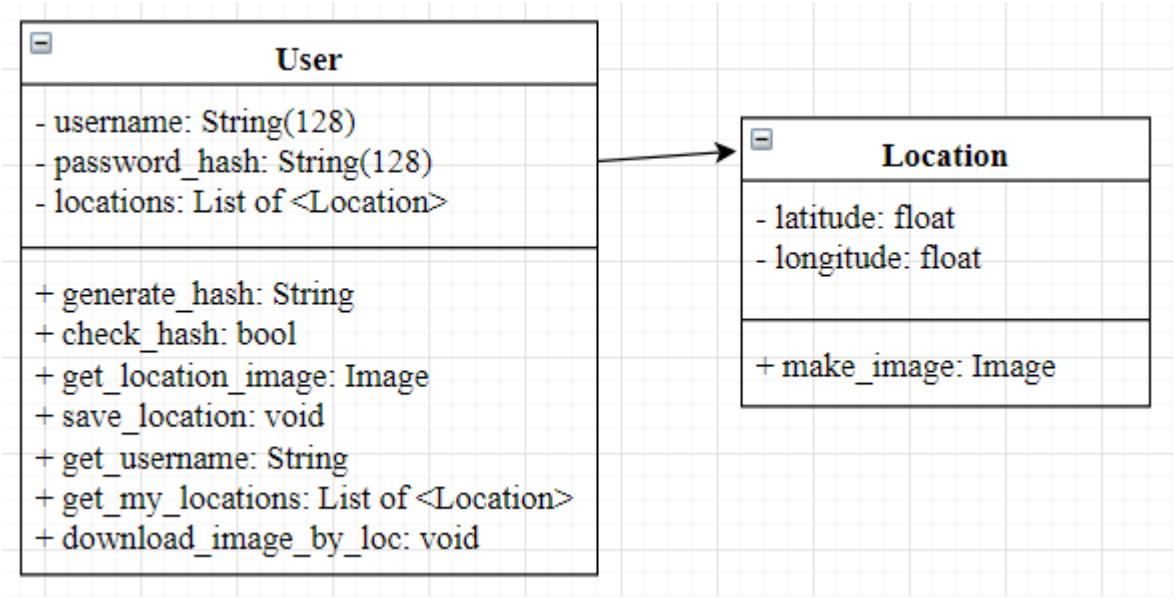

Рисунок 2.6 – Діаграма класів

Користувач за допомогою веб-форми передає через AJAX POST-запит

на сервер, де його підхоплює обробник /getimage, після чого дані форми передаються в метод класу Location і за допомогою обробника отримують зображення. Після того як зображення створене, обробляє on_success обробник AJAX-запиту, картинка промальовується і внутрішній вміст оновлюється без перезавантаження сторінки. Якщо користувач залогіненний, дані про запитані координати та масштаб зберігаються в базу даних. Якщо користувачеві потрібно буде це зображення ще раз, замість введення координат та масштабу він зможе через список своїх запитів знайти і провести потрібну операцію знову.

2.8 Макет шаблону інтерфейсу

На рисунку 2.7 зображено макет шаблону інтерфейсу системи для зберігання зображень місцевості.

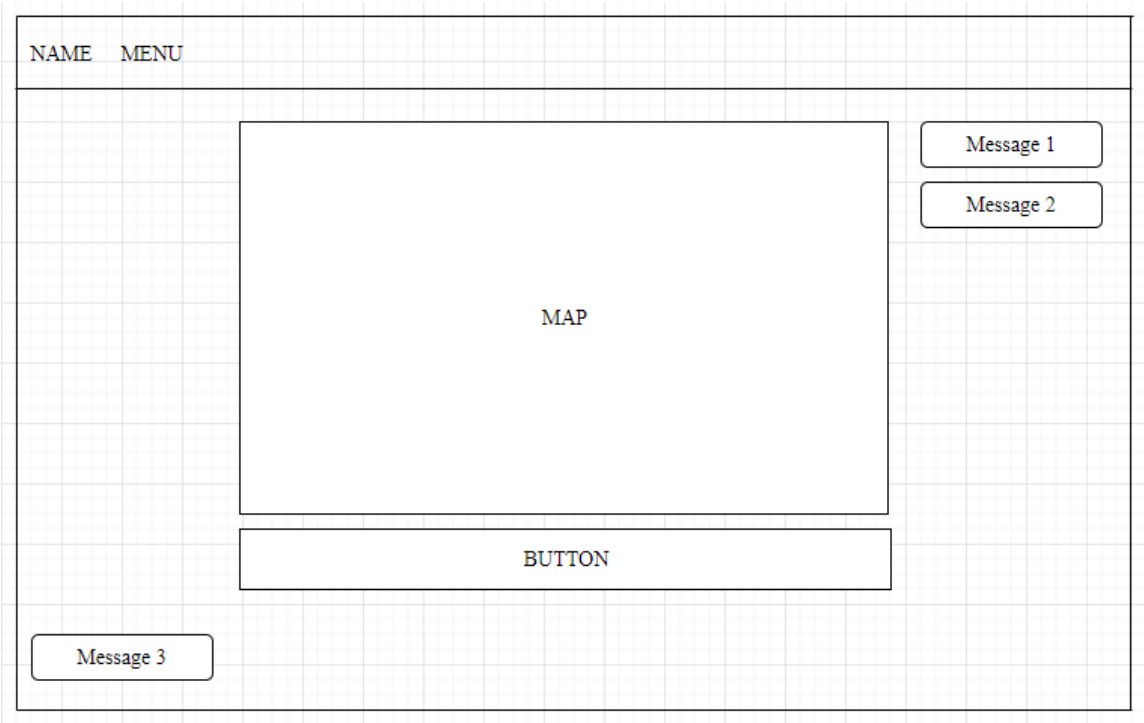

Рисунок 2.7 – Макет шаблону інтерфейсу

Макет шаблону інтерфейсу буде складатися з імені самої системи у хедері, декількох пунктів меню (перехід на головну сторінку, список вже збережених та нещодавно переглянутих зображень місцевості, профіль користувача), робочої області у вигляді карти, кнопки для пошуку певної місцевості та повідомлення про успішність додавання зображення місцевості в базу даних.

2.8 Висновки

У другому розділі кваліфікаційної роботи було створено технічне завдання, визначені вхідні та вихідні дані, намальовано макет шаблону інтерфейсу інформаційної системи, складені основні вимоги, сформований алгоритм роботи, а також розроблені діаграми та схеми програмного продукту.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Вибір програмного забезпечення

Для оптимізації розробки проекту та зручності були використані наступні програми:

а) пакет Open Server (серверна платформа);

б) графічний інтерфейс PhpMyAdmin для управління СУБД MySQL;

в) редактор програмного коду SublimeText.

Кожен з цих додатків має низку унікальних, незамінних властивостей, які полегшують розробку і тестування web-додатку.

3.2 Створення БД за допомогою PhpMyAdmin

PhpMyAdmin - веб-додаток з відкритим кодом, написаний на мові PHP і представляє собою веб-інтерфейс для адміністрування СУБД MySQL. PHPMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати команди SQL і переглядати вміст таблиць і баз даних. Додаток користується великою популярністю у веб-розробників, тому що дозволяє управляти СУБД MySQL без безпосереднього введення SQL-команд, надаючи дружній інтерфейс. Саме він буде використовуватися для роботи з базою даних в сайті, що створюється.

3.3 Встановлення Yii

Встановити Yii можна двома шляхами: використовуючи менеджер пакунків Composer або завантаживши файл архіву [2]. Перший варіант є

бажанішим, тому що дозволяє встановлювати нові розширення або оновлювати Yii простим виконанням однієї команди.

Після стандартного встановлення Yii ми отримуємо як фреймворк, так і шаблон проекту. Шаблон проекту – це робочий проект Yii, в якому реалізовано деякий базовий функціонал, такий як система входу/виходу користувачів, форма зворотнього зв’язку і т. д.

Також Yii надає інший шаблон із назвою Розширений шаблон проекту, який краще використовувати у середовищі розробки в команді для розробки складних додатків.

Користувачам Linux та Mac OS X потрібно виконати наступні команду:

```
curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer
```

Рисунок 3.1 – встановлення Composer

При роботі з Windows, необхідно завантажити та запустити Composer-Setup.exe.

Якщо ж Composer вже було встановлено раніше, необхідно переконатись, що використовується його остання версія. Це можна перевірити за допомогою команди composer self-update.

Після встановлення Composer, встановити Yii можна виконавши наступну команду з директорії, яка доступна через Web, як зображено на рисунку 3.2:

```
composer global require "fxp/composer-asset-plugin:^1.4.1"
composer create-project --prefer-dist yiisoft/yii2-app-basic basic
```

Рисунок 3.2 – Встановлення Yii за допомогою Composer

Перша команда встановить плагін ресурсів composer (composer-asset-plugin), що дозволить керувати залежностями пакунків Bower та NPM за

допомогою Composer. Цю команду потрібно виконати лише один раз. Друга команда встановить Yii у директорію під назвою basic. За бажанням, можна обрати інше ім’я для директорії.

Встановлення Yii з архіву складається з трьох кроків:

а) завантаження архіву з офіційного сайту фреймворку Yii;

б) розпакування архіву в директорію, доступну через Web.

в) редагування файлу конфігурації config/web.php, а саме ввести таємний ключ до пункту cookieValidationKey.

Після успішного встановлення необхідно налаштувати свій веб-сервер або використати вбудований веб-сервер PHP, виконавши консольну команду php yii serve із директорії web.

3.4 Встановлення Gii

Встановлювати розширення Gii для генерації коду рекомендується за допомогою Composer, для цього необхідно виконати команду, як зображено на рисунку 3.3:

```
php composer.phar require --dev --prefer-dist yiisoft/yii2-gii
//або додати
"yiisoft/yii2-gii": "~2.0.0"
//до секції require-dev файлу composer.json.
```

Рисунок 3.3 – Встановлення Gii за допомогою Composer

Коли розширення Gii встановлено, його необхідно підключити до файлу конфігурації інформаційної системи [1, с. 85]:

```
return [
    'bootstrap' => ['gii'],
```

```
    'modules' => [
        'gii' => 'yii\gii\Module',
        // ...
    ],
    // ...
];
```

Рисунок 3.4 – Підключення Gii

Отримати доступ до Gii можна буде за допомогою localhost.

В базовому шаблоні проекту структура конфігурації трохи інакша, тому розширення Gii необхідно сконфігурувати у config/web.php, як це зображено на рисунку 3.5:

```
// ...
if (YII_ENV_DEV) {
    // налаштування конфігурації для середовища розробки
    $config['bootstrap'][] = 'debug';
    $config['modules']['debug'] = 'yii\debug\Module';
    $config['bootstrap'][] = 'gii';
    $config['modules']['gii'] = 'yii\gii\Module';
}
```

Рисунок 3.5 – Конфігурування Gii

Таким чином, щоб налаштувати IP-адреси, необхідно зробити як показано нижче:

```
if (YII_ENV_DEV) {
    $config['bootstrap'][] = 'debug';
    $config['modules']['debug'] = 'yii\debug\Module';
    $config['bootstrap'][] = 'gii';
```

```
$config['modules']['gii'] = [
    'class' => 'yii\gii\Module',
    'allowedIPs' => ['127.0.0.1', '::1', '192.168.0.*', '192.168.178.20'],
];}
```

Рисунок 3.6 – Налаштування IP-адрес для доступу

3.5 Створення і налаштування підключення бази даних

Для створення проекту потрібно налаштувати Yii для роботи з базою даних:

```
<?php
return array(
'connectionString'=>'mysql:host=localhost;dbname=mehodical,
'emulatePrepare' => true,
'username' => 'root',
'password' => '',
'charset' => 'utf8', ); ?>
```

Рисунок 3.7 – Налаштування з’єднання з базою даних

Фреймворк доступний разом із вбудованим генератором коду Gii. Gii дозволяє генерувати моделі на основі таблиць в базі даних, а так само CRUD-контролери для основних дій з управління записами, такими як додавання запису (Create), перегляд запису (Read), редагування запису (Update) і видалення (Delete) [4, с. 100]. Для активації Gii в файл конфігурації програми було додано опис підключення модуля gii, як зображено на рисунку 3.8:

```
'Gii' => array (
'Class' => 'system.gii.GiiModule',
```

```
'Password' => 'generate',
'IpFilters' => array ( '127.0.0.1', ':: 1'),
),
```

Рисунок 3.8 – Активація кодогенератора Gii

Для налаштувати з’єднання з базою даних, необхідно створити клас Active Record, визначити дію та створити представлення [3, с. 15]. Для цього необхідно створити базу даних, з якої будемо отримувати дані. Слід зазначити, що створити базу даних можна на SQLite, MySQL, PostgreSQL, MSSQL або Oracle, оскільки Yii має вбудовану підтримку для багатьох баз даних. Для створення інформаційної системи для зберігання зображень місцевості використовується система управління базами даних MySQL.

Після встановлення, треба відредагувати файл config/db.php і змініть параметри для відповідності БД:

```
<?php
return
[
 'class' => 'yii\db\Connection',
 'dsn' => 'mysql:host=localhost;dbname=supportmad_claim',
 'username' => 'root',
 'password' => '',
 'charset' => 'utf8'];
```

Рисунок 3.9 – Бібліотека PDO для роботи з базою даних

Файл конфігурації config/db.php є типовим інструментом налаштування на основі файлів. Даний файл конфігурації визначає параметри, які необхідні для створення та ініціалізації екземпляру yii\db\Connection, через який можна робити SQL-запити до зазначеної бази даних.

Файл конфігурації config/db.php буде включений до конфігурації

головного додатка config/web.php, який визначає як має бути проініціалізований екземпляр додатку.

Також для створення адаптивного дизайну потрібно завантажити та встановити доповнення Yiistrap2 [5, с. 117]:

```
'import'=>array(
'bootstrap.helpers.*',
'bootstrap.behaviors.*',
'bootstrap.components.*',
'bootstrap.form.*',
'bootstrap.widgets.*',
),
'components'=>array(
'bootstrap' => array(
'class' => 'bootstrap.components.TbApi',
),
```

Рисунок 3.10 – Встановлення доповнення Yiiastrap (main.php)

Надалі необхідно налаштувати систему доступу на основі ролей:

```
'components'=>array(
'user' => array(
class' => 'WebUser',
),
'authManager' => array(
'class' => 'PhpAuthManager',
'defaultRoles' => array('guest'),
),
),
```

Рисунок 3.11 – Налаштування системи доступа на основі ролей (main.php)

Після цього нам необхідно створити контролер, налаштувати правила URL для красивого відображення та увімкнути підтримку Json [6, с. 208].

```
namespace app\controllers;
use yii\rest\ActiveController;
class UserController extends ActiveController
{
    public $modelClass = 'app\models\User';
}
```

Рисунок 3.12 – Створення контролеру

Клас контролера успадковується від yii\rest\ActiveController. Ми задали modelClass як app\models\User, тим самим вказавши контролера, до якої моделі йому необхідно звертатися для редагування або вибірки даних.

```
'urlManager' => [
    'enablePrettyUrl' => true,
    'enableStrictParsing' => true,
    'showScriptName' => false,
    'rules' => [
        ['class' => 'yii\rest\UrlRule', 'controller' => 'user'],
    ],
]
```

Рисунок 3.13 – Налаштування правил URL

Налаштування вище додають правило для контролера user, яке надає доступ до даних користувача через красиві URL і логічні дієслова HTTP.

Для того щоб API міг приймати дані в форматі JSON, треба налаштувати властивість parsers у компонента request application component на використання yii\web\JsonParser JSON даних на вході:

```
'request' => [
    'parsers' => [
        'application/json' => 'yii\web\JsonParser',
    ]
]
```

Рисунок 3.14 – Включення JSON на прийом даних

3.6 Авторизація та аутентифікація

Компонент user керує статусом аутентифікації користувача і вимагає, вказаний identity class, який буде містити логіку аутентифікації. У конфігурації інформаційної системи, identity class для user заданий як app\models\User.

```
return [
    'components' => [
        'user' => [
            'identityClass' => 'app\models\User',
        ],
    ],];
```

Рисунок 3.15 – Використання класу user

Фільтри контролю доступу (ACF) є простим методом, який найкраще використовувати в додатках з простим контролем доступу. Як видно з їх назви, ACF – це фільтри, які можуть приєднуватися до контролера або модулю як поведінку [10, с. 116]. ACF перевіряє набір правил доступу, щоб переконатися, що користувач має доступ до запрошенням дії:

```
use yii\filters\AccessControl;
```

```
class SiteController extends Controller
{
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'only' => ['login', 'logout', 'signup'],
                'rules' => [
                    [
                        'allow' => true,
                        'actions' => ['login', 'signup'],
                        'roles' => ['?'],
                    ],
                    [
                        'allow' => true,
                        'actions' => ['logout'],
                        'roles' => ['@'],
                    ],
                ],
            ],
        ];}}
```

Рисунок 3.16 – Використання ACF-фільтру

В основному клас yii\web\User використовують як компонент додатка user.

Можна отримати identity поточного користувача, використовуючи вислів Yii::$ app->user->identity. Воно поверне екземпляр identity class, який представляє поточного аутентиффікованого користувача, або null, якщо

поточний користувач не аутентифікований (наприклад, гість). Наступний код показує, як отримати іншу пов'язану з аутентифікацією інформацію з yii\web\User:

```
$identity = Yii::$app->user->identity;
$id = Yii::$app->user->id;
$isGuest = Yii::$app->user->isGuest;
$identity = User::findOne(['username' => $username]);
Yii::$app->user->login($identity);
```

Рисунок 3.17 – Використання компоненту user

Метод yii\web\User::login() встановлює identity поточного користувача в yii\web\User. Якщо сесії включені, то identity буде зберігатися в сесії, так що стан статусу аутентифікації буде підтримуватися на всьому протязі сесії. Якщо включений вхід, заснований на cookie (так званий "запам'ятай мене" вхід), то identity також буде збережена в cookie так, щоб статус аутентифікації користувача міг бути відновлений протягом усього часу життя cookie.

3.7 Реєстрація JavaScript та CSS

Вбудовувані скрипти корисні для конфігурації динамічно генеруємого коду і невеликих повторно використовуваних фрагментів інтерфейсу, що містяться в віджети. Для їх додавання можна використовується метод registerJs():

```
$this->registerJs(
    "$('#myButton').on('click', function() { alert('Button clicked!'); });",
    View::POS_READY,
    'my-button-handler'
```

```
);
```

Рисунок 3.18 – Реєстрація вбудованого скрипту

Зовнішній скрипт може бути доданий в такий спосіб:

```
$this->registerJsFile(
    '@web/js/main.js',
    ['depends' => [\yii\web\JqueryAsset::className()]]
);
```

Рисунок 3.19 – Реєстрація зовнішнього скрипту

Зареєструвати CSS-файл можна в таким чином:

```
$this->registerCssFile("@web/css/themes/black-and-white.css", [
    'depends' => [\yii\bootstrap\BootstrapAsset::className()],
    'media' => 'print',
], 'css-print-theme');
```

Рисунок 3.20 – Реєстрація CSS-файлу

3.8 Обробка помилок

Оброблювач помилок реєструється в якості компонента додатка з ім'ям errorHandler. Використання обробника помилок здійснюється за допомогою команди, як показано на рисунку 3.21:

```
return [
    'components' => [
        'errorHandler' => [
            'maxSourceLines' => 20,    ],    ], ];
```

Рисунок 3.21 – Використання обробника помилок

3.9 Опис інтерфейсу інформаційної системи

На рисунку 3.22 зображена головна сторінка інформаційної системи для зберігання зображень місцевості. На ній показана карта сервісу Google Maps, на якій ми обираємо місцевість для додавання в базу. Координати та масштаб вибраної місцевості автоматично зберігаються в буфері перед додаванням в базу даних.

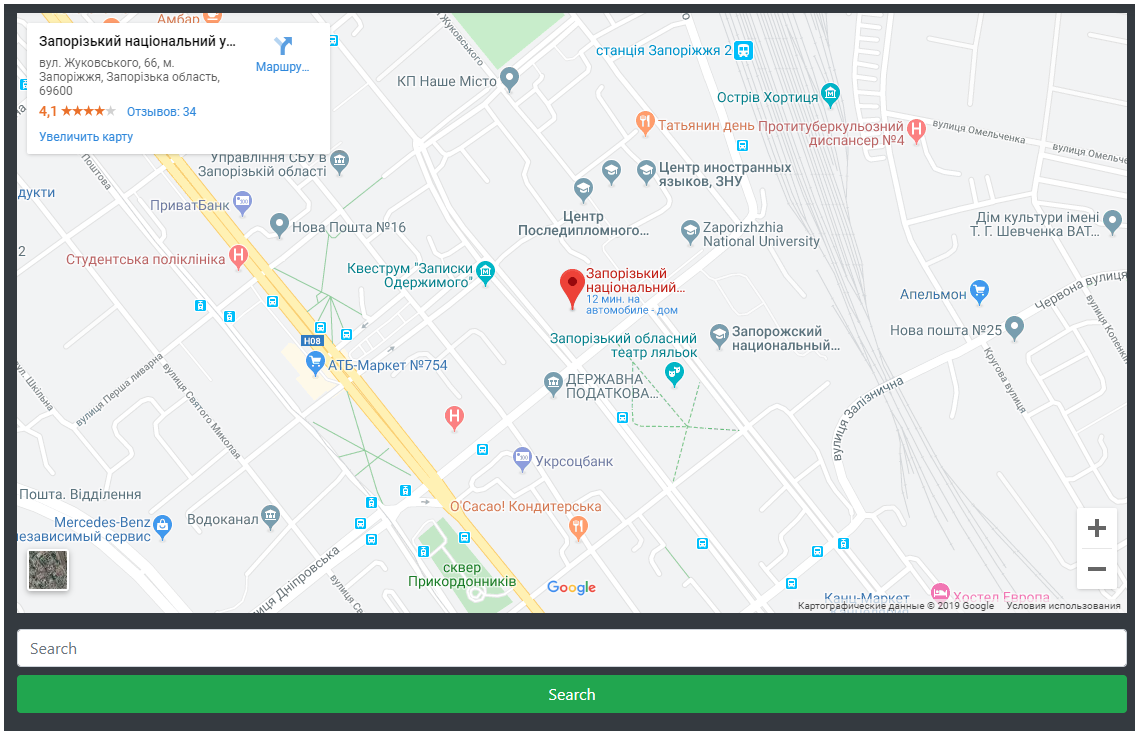

Рисунок 3.22 – Головна сторінка інформаційної системи

На рисунку 3.23 зображений пошук по сайту інформаційної системи.

Рисунок 3.23 – Вхідний дані

На рисунку 3.24 показані як будуть виглядати вже збережені в базу даних зображення місцевості.

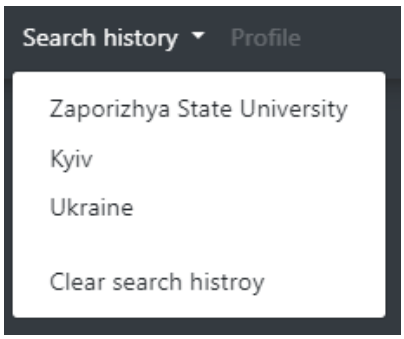

Рисунок 3.24 – Вихідний файл

На рисунку 3.25 показано меню сайту інформаційної системи. Воно складається з безпосередньо переходу на головну сторінку, профілю користувача та історії пошуку, яке в свою чергу складається з вже збережених зображень місцевості.

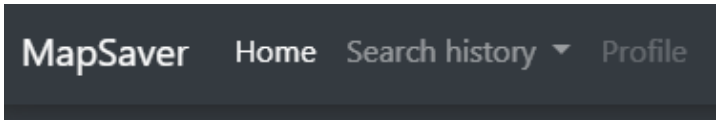

Рисунок 3.25 – Вихідний файл

Оскільки ми працюємо з базами даних у системі для зберігання зображень буде використовуватись реєстрація та авторизація. Авторизація та реєстрація користувача інформаційної системи для зберігання зображень місцевості зображена на рисунках 3.26 та 3.27.

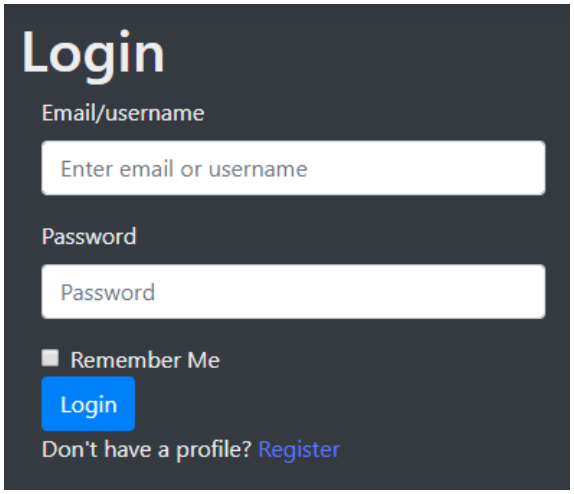

Рисунок 3.26 – Авторизація користувача у системі

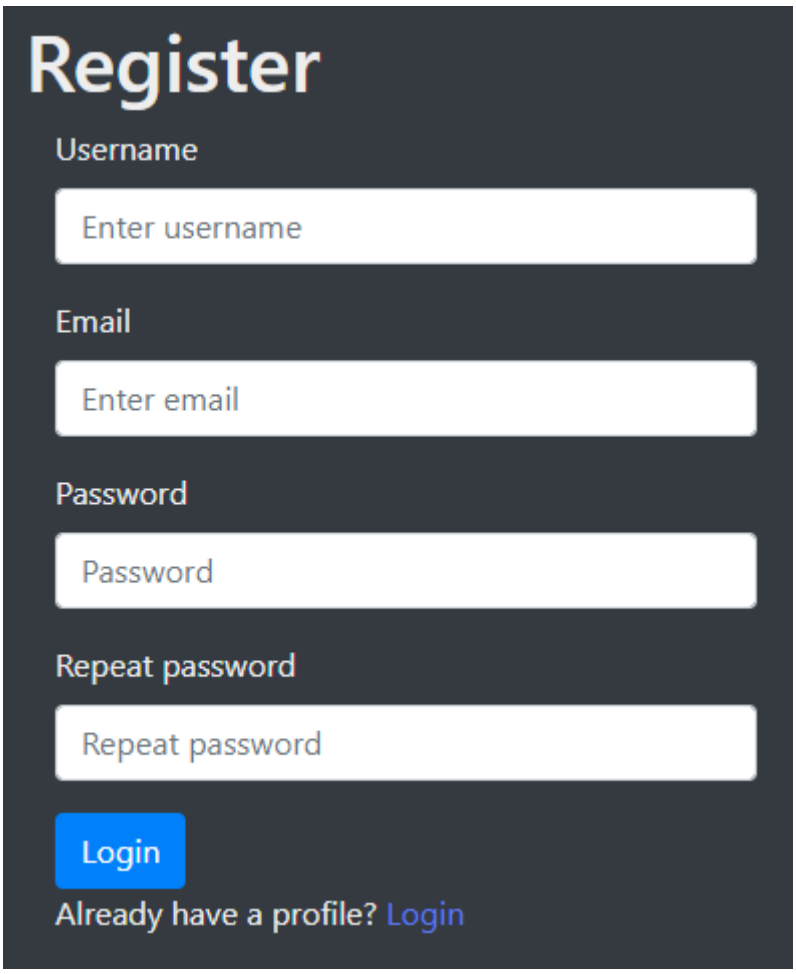

Рисунок 3.27 – Реєстрація користувача у системі

3.10 Проведення тестування

Під час створення інформаційної системи, на етапі розробки, одним із найважливіших етапів вважається перевірка працездатності.

Під час тестування програмний продукт перевіряють на відповідність вимогам технічного завдання, відповідність технічним характеристикам.

Сайт є крос-платформним, тому коректно відображається на будь-яких пристроях, із будь-якою роздільною здатністю екрана. Інформаційну систему було протестовано на планшеті, телефоні, ноутбуці та на широкоформатному екрані. У всіх випадках система працювала без помилок.

Розроблена система також є кросбраузерною тому коректно працює із всіма браузерами. Інформаційну систему було протестовано у таких

браузерах як: Googlre Chrome, Mozila, Opera, Safari, Internet Explorer 11 та Edge. Чітко виражених відмінностей виявлено не було, робота системи була на кожному з них коректною та плавною, а усі прогнозовані результати відповідали отриманим. Зависань чи трудностей у використанні системою виявлено не було, усі блоки не змінило своєї логічної структури.

При швидкості інтернету 100 МВ/с головна сторінка завантажилася за 1,23 с.; при швидкості 20 МВ/с сторінка завантажилася за 3,55с.; при швидкості інтернету 10 МВ/с – завантажився за 8.78 с.; при швидкості 1 МВ/с – завантажився за 11.76 с. Виходячи з отриманих результатів можна зробити висновок що інформаційна система працює швидко.

Серед гіперпосилань немає неактивних або неправильних. Переходи на сторінки сайту здійснюються швидко та коректно.

Статичні та динамічні зображення відображаються коректно. Фонове зображення адаптується під розміри дисплея.

Усі заголовки сторінок відображаються правильно. Іконка сайту відображається на всіх сторінках;

Зображення місцевості будуються плавно та без втрати кольору чи плавності руху.

Не було знайдено грубих синтаксичних та орфографічних помилок. Текст легко читати та сприймати.

Виходячи з вище перерахованих результатів, можна зробити висновок що інформаційна система повністю готова до використання, працює коректно, швидко та якісно.

3.11 Висновки

У третьому розділі кваліфікаційної роботи була розглянута реалізація інформаційної системи для зберігання зображень місцевості та її інтерфейс.

Був створений веб–сайт за допомогою фреймворку Yii 2, для якого

характерні усі основні особливості сайту, побудованого за допомогою обраного фреймворку. Він має трирівневу структуру Model–View–Controller, а всі дані, що містяться на сайті, зберігаються в базі даних. Для удосконалення інтерфейсу було використано доповнення YiiStrap, яке є аналогом фреймворку Bootstrap для Front-End розробки.

Проведено тестування системи на різних пристроях із різними операційними системами на наявність помилок, яке показало, що програмний продукт повністю придатний для користування.

Створено інструкцію для користувача, у якій детально описано які кроки потрібно зробити щоб використовувати інформаційну систему.

ВИСНОВКИ

У кваліфікаційній роботі була розроблена інформаційна система для зберігання зображень місцевості з використанням фреймворку Yii 2.

Для розробки було обрано мову програмування PHP та фреймворк Yii 2, JavaScript, мову розмітки HTML 5 та каскадну таблицю стилів CSS 3. Задачу кваліфікаційної роботи виконано в повному обсязі: розроблено та описано алгоритми основних блоків роботи програмного продукту, розроблено програмну реалізацію системи та проведено ряд тестів для підтвердження коректності роботи програмного продукту та подальшої підтримки системи.

Програмний продукт має хороші перспективи розвитку, а саме його функціональність можна легко розширити, додавши, наприклад, інші види карт, або збільшивши кількість вхідних та вихідних даних. Так само створену систему можна використовувати в іншому програмному забезпеченні, наприклад, в розробці інших програм для зберігання зображень місцевості.

Основні результати, отримані при виконанні кваліфікаційної роботи наступні:

а) виконано аналіз сучасного стану питання, на базі якого було визначено доцільність розробки нового продукту;

б) проведено аналіз існуючих реалізацій та досліджено їхні переваги та недоліки, в результаті якого була підтверджена актуальність розробки нової інформаційної системи;

в) розроблено графічну та структурну схему програмного продукту.

г) проведено тестування на різних пристроях і при різних розширеннях дисплею, в результаті чого було підтверджено, що програмний продукт являється адаптивним та кросплатформеним.

ПЕРЕЛІК ПОСИЛАНЬ

1. Keck B. Yii2 for beginers. M.: Web, 2015. 691 с.

2. Повний посібник з Yii 2. URL: https://www.yiiframework.com/doc/guide/2.0/uk/intro-yii (Дата звернення: 09.12.2019).

3. Качанов А., Ткаченко В., Головин А. Букварь по PHP и MySQL. StealthWeb, 2010. 31 с.

4. Веллинг Л., Томсон Л. Разработка веб-приложений с помощью PHP и MySQL. Москва: Вильямс, 2010. 848 с.

5. Локхарт Д. Современный PHP: новые возможности. СПб.: Символ-Плюс, 2016. 270 с.

6. Зандстра М. PHP. Объекты, шаблоны и методики программирования [пер. с англ.]. СПб.: Вильямс, 2015. – 576 с.

7. Кастаньето Д., Рават Х., Шуман С., Сколло К., Велиаф Д. Профессиональное PHP программирование. СПб.: Символ-Плюс, 2012. 224 с.

8. Сафронов М., Уайнсет Д. Разработка веб-приложений в Yii 2. 2015. 109 с.

9. Зандстра М. PHP: об’екти, шаблони і методики програмування [пер. с англ.]. СПб.: Диалектика-Вильямс, 2018. 576 с.

10. Ратшиллер Т., Геркер Т. PHP: разработка web-приложений. Библиотека программиста. СПб.: Питер, 2011. 384 с.

11. Суэринг С., Парк Д., Конверс Т. PHP и MySQL. Библия программиста. Москва: Вильямс, 2010. 912 с.

12. Джей-Яргер Р., Риз Д., Кинг Т. MySQL и mSQL: Базы данных для небольших предприятий и Интернета. СПб.: Символ-Плюс, 2013. 560 с.

Додаток А

Код інформаційної системи

```
use google\maps\LatLng;
use google\maps\services\DirectionsWayPoint;
use google\maps\services\TravelMode;
use google\maps\overlays\PolylineOptions;
use google\maps\services\DirectionsRenderer;
use google\maps\services\DirectionsService;
use google\maps\overlays\InfoWindow;
use google\maps\overlays\Marker;
use google\maps\Map;
use google\maps\services\DirectionsRequest;
use google\maps\overlays\Polygon;
use google\maps\services\DirectionsClient;
$direction = new DirectionsClient([
   'params' => [
      'language' => Yii::$app->language,
   ]
]);
$data = $direction->lookup();
$map = new Map([
   'center' => $coord,
]);
$directionsRequest = new DirectionsRequest
$polylineOptions = new PolylineOptions([
   'strokeColor' => '#FFAA00',
   'draggable' => true
```

```
]);
$directionsRenderer = new DirectionsRenderer([
    'map' => $map->getName(),
    'polylineOptions' => $polylineOptions
]);
$directionsService = new DirectionsService([
    'directionsRenderer' => $directionsRenderer,
    'directionsRequest' => $directionsRequest
]);
$marker = new Marker([
    'position' => $coord,
]);
$marker->attachInfoWindow
$map->addOverlay($marker);
$polygon = new Polygon([
    'paths' => $coords
]);
$polygon->attachInfoWindow(new InfoWindow([
    ]));
$map->addOverlay($polygon);
echo $map->display();
<?php
namespace application\widgets;
class GoogleMap extends TbBaseInputWidget
{
  public $options;
  private $cs;
  public function init()
  {
    parent::init();
```

```
    $this->cs = \Yii::app()->getClientScript();
  }
  public function run()
  {
    $options = array();
    if (empty($this->model[$this->attribute])) {
      if (empty($this->options['mapCenter'])) {
      }
      if (empty($this->options['point'])) {
        $options['point'] = $options['mapCenter'];
      }
      if (empty($this->options['mapZoom'])) {
        $options['mapZoom'] = 10;
      }
      if (empty($this->options['mapType'])) {
        $options['mapType'] = 'roadmap';
      }
    } else {
      $options = $this->model[$this->attribute];
    }
    $id = \CHtml::activeId($this->model, $this->attribute);
    $mapId = "GMapsID-" . $this->getId();
    $searchControlId = 'pac-input-' . $this->getId();
function getMarkerPosition(marker) {
  return [marker.getPosition().lat(), marker.getPosition().lng()];
}
function initMap () {
  var map;
  var options = {
    mapCenter: [{$options['mapCenter'][0]}, {$options['mapCenter'][1]}],
```

```
  point: [{$options['point'][0]}, {$options['point'][1]}],
  mapZoom: {$options['mapZoom']},
  mapType: "{$options['mapType']}"
};
var latLng = new google.maps.LatLng(options.point[0], options.point[1]);
map = new google.maps.Map(document.getElementById('{$mapId}'), {
    center: latLng,
    zoom: options.mapZoom,
    mapTypeId: options.mapType
});
var marker = new google.maps.Marker({
  position: latLng,
  map: map,
  draggable: true
});
var input = document.getElementById('{$searchControlId}');
var searchBox = new google.maps.places.SearchBox(input);
map.controls[google.maps.ControlPosition.TOP_LEFT].push(input);
map.addListener('bounds_changed', function(){
  searchBox.setBounds(map.getBounds());
});
var markers = [];
markers.push(marker);
searchBox.addListener('places_changed', function(){
  var places = searchBox.getPlaces();
  if (places.length == 0) {
    return;
  }
  markers.forEach(function(marker){
    marker.setMap(null);
```

```
  });
  markers = [];
  var bounds = new google.maps.LatLngBounds();
  places.forEach(function(place){
   var icon = {
    url: place.icon,
    size: new google.maps.Size(71, 71),
    origin: new google.maps.Point(0, 0),
    anchor: new google.maps.Point(17, 34),
    scaledSize: new google.maps.Size(25, 25)
   };
   markers.push(new google.maps.Marker({
    map: map,
    title: place.name,
    position: place.geometry.location,
    draggable: true
   }));
   if (place.geometry.viewport){
    bounds.union(place.geometry.viewport);
   } else {
    bounds.extend(place.geometry.location);
   }
  });
  map.fitBounds(bounds);
  googleMapSave();
 });
google.maps.event.addListener(markers[0], 'drag', function(){
  googleMapSave();
 });
 map.addListener('zoom_changed', function(){
```

```
    googleMapSave();
  });
  map.addListener('maptypeid_changed', function(){
    googleMapSave();
  });
  map.addListener('mouseout', function(){
    googleMapSave();
  });
  function googleMapSave(){
    options = {
      mapCenter: getMarkerPosition(markers[0]),
      point: getMarkerPosition(markers[0]),
      mapZoom: map.getZoom(),
      mapType: map.getMapTypeId()
    };
  }
  googleMapSave();
}
    $css = <<<CSS
.controls {
  margin-top: 10px;
  border: 1px solid transparent;
  border-radius: 2px 0 0 2px;
  box-sizing: border-box;
  -moz-box-sizing: border-box;
  height: 32px;
  outline: none;
  box-shadow: 0 2px 6px rgba(0, 0, 0, 0.3);
}
#{$searchControlId} {
```

```
            background-color: #fff;
            font-family: Roboto;
            font-size: 15px;
            font-weight: 300;
            margin-left: 12px;
            padding: 0 11px 0 13px;
            text-overflow: ellipsis;
            width: 300px;
          }
          #{$searchControlId}:focus {
            border-color: #4d90fe;
          }
          CSS;
            $this->cs->registerCss($this->getId(), $css);
            echo \CHtml::openTag('div', array('class' => 'map-container'));
            echo \CHtml::openTag('div', array('style' => 'width: 100%;height:400px',
'id' => $mapId));
            echo \CHtml::closeTag('div');
            echo \CHtml::closeTag('div');
            echo \CHtml::activeHiddenField($this->model, $this->attribute);
          }
        }
            NavBar::begin([
                'brandUrl' => Yii::$app->homeUrl,
                'options' => [
                    'class' => 'navbar-inverse navbar-fixed-top',
                ],
            ]);
            $menuItems = [
                ['label' => 'Home', 'url' => ['/site/index']],
```

```
        ['label' => 'Places', 'url' => ['/place']],
        ['label' => 'About', 'url' => ['/site/about']],
        ['label' => 'Contact', 'url' => ['/site/contact']],
    ];
    if (Yii::$app->user->isGuest) {
        $menuItems[] = ['label' => 'Signup', 'url' => ['/site/signup']];
        $menuItems[] = ['label' => 'Login', 'url' => ['/site/login']];
    } else {
        $menuItems[] = [
            'label' => 'Logout (' . Yii::$app->user->identity->username . ')',
            'url' => ['/site/logout'],
            'linkOptions' => ['data-method' => 'post']
        ];
    }
    echo Nav::widget([
        'options' => ['class' => 'navbar-nav navbar-right'],
        'items' => $menuItems,
    ]);
    NavBar::end();
?>
public function actionCreate_geo()
{
    $model = new Place();
    if ($model->load(Yii::$app->request->post())) {
   } else {
        return $this->render('create_geo', [
            'model' => $model,
        ]);
    }
<div class="place-form">
```

```
<?php $form = ActiveForm::begin(); ?>
<div class="col-md-6">
<?= $form->field($model, 'name')->textInput(['maxlength' => 255]) ?>
<?= $form->field($model, 'website')->textInput(['maxlength' => 255]) ?>
<?= $form->field($model, 'place_type')
->dropDownList(
$model->getPlaceTypeOptions(),
)->label('Type of Place') ?>
<?= $form->field($model, 'notes')->textArea() ?>
<?= BaseHtml::activeHiddenInput($model, 'lat'); ?>
<?= BaseHtml::activeHiddenInput($model, 'lng'); ?>
<div class="form-group">
<?= Html::submitButton($model->isNewRecord ? 'Create' : 'Update',
['class' => $model->isNewRecord ? 'btn btn-success' : 'btn btn-primary']) ?>
</div>
</div> <!-- end col 1 --><div class="col-md-6">
<div id="preSearch" class="center">
</div>
<div id="searchArea" class="hidden">
<div id="autolocateAlert">
</div> <!-- end autolocateAlert -->
<p>Searching for your current location...<span id="status"></span></p>
<article>
</article>
<div class="form-actions hidden" id="actionBar">
</div> <!-- end action Bar-->
</div> <!-- end searchArea -->
</div> <!-- end col 2 -->
<?php ActiveForm::end(); ?>
</div>
```

```
        public function getLocation($place_id) {
            $sql = 'Select AsText(gps) as gps from {{%place_gps}} where place_id
= '.$place_id;
            $model = PlaceGPS::findBySql($sql)->one();
            $gps = new \stdClass;
            if (is_null($model)) {
              return false;
            } else {
              list($gps->lat, $gps->lng) = $this->string_to_lat_lon($model->gps);
            }
            return $gps;
          }
          public function actionCreate_place_google()
          {
            $model = new Place();
            if ($model->load(Yii::$app->request->post()))
        else {
                return $this->render('create_place_google', [
                    'model' => $model,
                ]);
            }
          }
        <div class="place-create">
            <h1><?= Html::encode($this->title) ?></h1>
            <?= $this->render('_formPlaceGoogle', [
                'model' => $model,
            ]) ?>
        </div>
        <?
                                'libraries' => 'places',
```

```
                ));
            var input = document.getElementById('place-searchbox');
            var options = $options;
            searchbox = new google.maps.places.Autocomplete(input, options);
            setupListeners();
        })();" , \yii\web\View::POS_END );
        ?>
        public function actionCreate_place_google()
        {
          $model = new Place();
          if ($model->load(Yii::$app->request->post())) {
             if (Yii::$app->user->getIsGuest()) {
              $model->created_by = 1;
             } else {
              $model->created_by= Yii::$app->user->getId();
             }
             $form = Yii::$app->request->post();
             $model->save();
             $model->addGeometry($model,$form['Place']['location']);
             return $this->redirect(['view', 'id' => $model->id]);
        class SearchLocation extends \yii\base\Model
        {
           public $address;
        }
        $model = new SearchLocation();
        $form = \yii\widgets\ActiveForm::begin();
        $form->field($model,                                    'address')-
>widget(\maplocation\SelectMapLocationWidget::className(),
        \yii\widgets\ActiveForm::end();
```

Рисунок 3.28 – Код інформаційної системи з використанням Yii 2