МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ «ТУРИСТИЧНЕ АГЕНТСТВО» З ВИКОРИСТАННЯМ YII2»**

Виконав: студент 2 курсу, групи 8.1211-1іпз-дн
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
О.В. Шульга
(ініціали та прізвище)

Керівник завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н., Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

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

_____ Лісняк А.О.

(підпис)

« ____ » _____ 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Шульзі Олександру Владиславовичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проєкту) Розробка інформаційної системи «туристичне агентство» з використанням Yii2

керівник роботи (проєкту) Лісняк Андрій Олександрович, к.ф.-м.н., доцент

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

затверджені наказом ЗНУ від « ____ » _____ 2022 року № _____

2. Строк подання студентом роботи _____

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Приклади використання інформаційних систем
3. Переваги користування інформаційною системою.

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
1			
2			
3			
Додатки			

7. Дата видачі завдання _____

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану виконання кваліфікаційної роботи магістра	08.05.2022	
2.	Збір вихідних даних та аналіз предметної області.	12.05.2022	
3.	Обробка методичних та теоретичних джерел.	20.05.2022	
4.	Специфікація вимог до системи. Робота над першим розділом.	13.06.2022	
5.	Проєктування системи. Робота над другим розділом.	15.07.2022	
6.	Реалізація та тестування системи. Робота над третім розділом.	11.08.2022	
7.	Розробка керівництва користувача. Робота над додатками.	30.09.2022	
8.	Оформлення та нормоконтроль кваліфікаційної роботи бакалавра.	10.01.2023	
9.	Захист кваліфікаційної роботи бакалавра.	15.02.2023	

Студент _____ О.В. Шульга
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.О. Лісняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка інформаційної системи «Туристичне агентство» з використанням Yii2»: 58 с., 31 рис., 15 джерел.

АВТОМАТИЗОВАНА ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, КОНТРОЛЕР, МОДЕЛЬ, МОДЕЛЬ ДАНИХ, ФРЕЙМВОРК, ШАБЛОН.

Об’єкт дослідження – методи планування та моніторинг розташування туристичних рекреаційних зон.

Мета роботи – розробка автоматизованої інформаційної системи за допомогою фреймворка Yii.

Метод дослідження – порівняння, аналіз, описовий, структурний.

У кваліфікаційній роботі розглянуто автоматизовані інформаційні системи, які використовуються для планування подорожей. Зростання високого попиту населення на планування подорожей з лікувальною, пізнавальною або оздоровчою метою робить важливим розв'язок задач ефективного планування та моніторингу рекреаційних ресурсів України. Тому розробка автоматизованої інформаційної системи рекреаційно-туристичних об’єктів є актуальною задачею.

SUMMARY

Master's Qualification Thesis "Development of the Travel Agency Information System using Yii2": 58 pages, 31 figures, 15 references.

AUTOMATED INFORMATION SYSTEM, DATABASE, CONTROLLER, MODEL, DATA MODEL, FRAMEWORK, TEMPLATE.

Object of research is the methods of planning and monitoring the location of tourist recreational areas.

Aim of the study is to develop an automated information system on base of PHP framework Yii.

Methods of research are analytical, comparative, analysis and searching of information.

In the qualification work the automated information systems used for travel planning. An increase in the high demand of the population for planning travel with a therapeutic, cognitive or recreational purpose makes an important solution to the tasks of efficient planning and monitoring of recreational resources of Ukraine. Therefore, the development of an automated information system for recreation – tourist objects is a main task.

ЗМІСТ

ВСТУП

PHP-фреймворки мають величезні можливості і унікальні екосистеми, які підійдуть під величезну кількість завдань. Вони здатні створювати краще сформовані, безпечні і зрозумілі додатки або вебсайти за більш короткі проміжки часу.

Розробка програми або веб-сайту з нуля вимагає багато роботи. У багатьох випадках потрібно буде відтворити функції, які вже були виконані тисячі разів, що приблизно так само ефективно, як заново винаходити колесо. Програмні фреймворки можуть допомогти обійти цю проблему, надаючи основу, на яку можна спиратися.

У світі розробки програмного забезпечення термін «framework» відноситься до бібліотек файлів, які включають кілька основних функцій. Мета фреймворка – надати основу, яку можна використовувати для більш ефективної розробки проєктів. Щоб зробити це, він буде включати в себе безліч функцій, які потрібно буде кодувати з нуля, якщо необхідно почати з початку.

Фреймворки функціонують однаково, позбавляючи від необхідності постійно винаходити щось нове. Якщо необхідно розробити новий PHP-додаток, найкраще почати з пошуку фреймворка, який включає в себе всі функції, які знадобляться. Зараз існує багато варіантів на вибір. Всі вони унікальні в тому, як вони підходять до розвитку, і у всіх є свої плюси і мінуси.

1 ОГЛЯД ТА ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Загальні поняття інформаційних систем

Інформаційна система – система, яка організовує пам’ять і маніпулювання інформацією щодо проблемної сфери.

Характерною рисою інформаційної системи є те, що людина виступає активним учасником інформаційного процесу. Це виявляється в умовах функціонування автоматизованого робочого місця, коли людина (кінцевий користувач) здійснює інформації в систему підтримує її в актуальному стані, обробляє інформацію і використовує здобуті результаті в управлінні. Інформація служить способом опису взаємодії між джерелом й одержувачем. Те саме повідомлення одному одержувачі може давати багато інформації, а іншому – мало або нічого.

Потенційні можливості інформаційної системи реалізуються через їх функції, до яких належать:

– обчислювальна – вчасно і якісно виконує оброблення інформації в усіх аспектах, що цікавлять систему управління;

– відстежуюча – відстежує і формує всю необхідну для управління зовнішню та внутрішню інформацію;

– запам’ятовувальна – забезпечує безупинне накопичення, систематизації, збереження та відновлення всієї необхідної інформації;

– регулювальна – здійснює інформаційно-керуючий вплив на об’єкт управління та його ланки при відхилені їхніх параметрів функціонування від заданих значень;

– оптимізаційна – забезпечує оптимальні розрахунки в міру зміни цілей, критеріїв та умов функціонування об’єкта управління;

– прогнозна – визначає основні тенденції, закономірності та показники розвитку об’єкта управління;

– аналізаторна – визначає основні показники техніко-економічного рівня виробництва і господарської діяльності;

– документувальна – забезпечує формування всіх обліково-звітних, планово-розпорядницьких, конструкторсько-технологічних та інших форм документів.

Одним із методів проєктування інформаційних систем є розчленовування її на окремі частини, з яких у міру необхідності комплектують конкретну автоматизовану систему управління. Такий метод називають декомпозицією.

Будь-яка система по-своєму складна. Це означає, що сукупність інформації, яка характеризує систему, і сукупність зв’язків між її елементами неможливо сприйняти в цілому та повністю. Цим система відрізняється від будь-якої задачі. Тому, згідно з методом декомпозиції, для оперативного впровадження інформаційної системи необхідно забезпечити оптимальну її структурованість.

Оптимально структурована система є багаторівневою, багатоцільовою організованою сукупністю елементів (модулів) і задовольняє такі вимоги:

– кожен рівень ієрархії має повністю проглядатися і бути зрозумілим без детального знання нижчих рівнів;

– зв’язки між елементами на одному рівні ієрархії мають бути мінімальними;

– не повинно бути зв’язків між елементами через один рівень ієрархії;

– елемент вищого рівня має викликати елемент наступного рівня і, передаючи йому необхідну вхідну інформацію, має утворювати з ним єдине ціле;

– елемент вищого рівня має викликати елемент наступного рівня і, передаючи йому необхідну вхідну інформацію, має утворювати з ним єдине ціле;

– елемент наступного рівня після закінчення своєї роботи повертає управління елементу, що його викликав, передаючи йому результати своєї

роботи.

Жорсткими є обмеження щодо структури системи, оскільки неможливо розробити таку ідеальну систему, щоб потім не вносити в неї зміни. До того ж при експлуатації інформаційної системи основним режимом її роботи є режим саме внесення змін.

Внесення змін у добре структуровану систему стосується не багатьох елементів, які добре локалізуються. В іншому випадку внесення навіть дрібних змін призводить до перепроєктування, перепрограмування великих частин системи.

Збільшена декомпозиція функціональної частини автоматизованої інформаційної системи передбачає встановлення структури елементів (модулів) різних рівнів:

а) комплекс першого рівня: він охоплює автоматизовані підсистеми, комплекси задач;

б) комплекс другого рівня: до нього належать автоматизовані функції (задачі) управління;

в) комплекс наступних рівнів: це машинні процедури, що реалізуються управлінським персоналом.

Функціональну декомпозицію інформаційної системи доцільно здійснювати за об’єктивним принципом на основі такої схеми: виробниче об'єднання – промислове підприємство – виробництво – цех –технологічний процес (дільниця) – робоче місце (вертикальна декомпозиція) з виділенням функцій управління для кожного об’єкта за схемою планування: облік – контроль – аналіз – регулювання (горизонтальна декомпозиція).

Крім функціональної, використовують декомпозицію організаційного (компоненти інструктивно-методичного і документального забезпечення), інформаційного (компоненти поза машинної та внутрімашинної інформаційної бази), технічного (компоненти засобів введення, зберігання, оброблення, виведення, передавання даних), програмного (компоненти операційної системи ЕОМ, системи управління базами даних, функціональні,

організаційні компоненти системи оброблення даних), ергономічного й іншого забезпечень.

Метод декомпозиції використовують на кожній стадії проєктування інформаційних систем для зниження ступеня невизначеності та виділення багатьох проєктивних задач, послідовного структурування процесу їх розв'язування, а також для опису одержуваних структур у вигляді ієрархічно пов’язаних інформаційних сукупностей.

Декомпозиція передбачає існування кількох способів розчленування проєктованої системи. Завершенням її ж такий стан об’єкта, коли у процесі розчленування утворюються елементи (частини системи), що сприймаються як неподільні об’єкти.

За системою підходу до проєктування інформаційних систем будь-який об’єкт розглядається як певна системи, яку можна поділити на підсистеми, кожна з яких може бути поділена на підсистеми нижчого порядку. Підсистемами найнижчого порядку є елементи (задачі), внутрішня структура яких не важлива для розв’язання інших задач цього рівня. Однак слід мати на увазі, що властивості окремої підсистеми впливають на інші підсистеми та систему загалом.

1.2 Структура і склад інформаційної системи

Практично всі різновиди інформаційних систем незалежно від сфери їхнього застосування включають один і той набір компонентів:

а) функціональні компоненти: функціональні підсистеми (модулі, бізнеси-додатки), функціональні задачі, моделі й алгоритми;

б) компоненти системи опрацювання даних: інформаційне забезпечення, програмне забезпечення, технічне забезпечення, правове забезпечення, лінгвістичне забезпечення;

в) організаційні компоненти (персонал): нова організаційна структура

фірми, персонал (штати, посадові інструкції).

При цьому під функцією управління розуміється спеціальний постійний обов'язок одного або декількох осіб, виконання якого призводить до досягнення визначеного ділового результату.

Під функціональними компонентами розуміється система функцій управління – повний набір (комплекс) взаєморв'язаних у часу і просторі робіт із управління, необхідних для досягнення поставлених перед підприємством цілей.

Дійсно, будь-яка складна управлінська функція розчленовується на ряд більш дрібних задач і зрештою доводиться до безпосереднього виконавця. Саме від того, як буде виконане те або інше завдання окремим робітником, залежить успіх у рішенні кінцевих задач фірми в цілому. Таким чином, уся сукупність складних управлінських впливів повинна мати своїм кінцевим результатом доведення загальних задач, що стоять перед підприємством, до кожного конкретного виконавця незалежно від його службового положення.

Природно, приведені положення підкреслюють не тільки індивідуальний, але і груповий характер функцій управління, а діловий (практичний) результат утворюється не епізодично, а постійно.

Весь процес управління фірмою зводиться або до лінійного (наприклад, адміністративного) керівництва підприємством або його структурним підрозділом, або до функціонального керівництва (наприклад, матеріально-технічне забезпечення, бухгалтерський облік і т.п.).

Тому декомпозиція інформаційної системи по функціональній ознаці містить у собі виділення її окремих частин, називаних функціональними підсистемами (функціональними модулями, бізнеса-додатками), що реалізують систему функцій управління. Функціональна ознака визначає призначення підсистеми, тобто те, для якої області діяльності вона призначена, і які основні цілі, задачі і функції вона виконує. Функціональні підсистеми в істотному ступені залежать від предметної області (сфери застосування) інформаційних систем.

Специфічні особливості кожної функціональної підсистеми містяться в так називаних «функціональних задачах» підсистеми. Звичайно управлінський персонал або зв'язує це поняття з досягненням визначених цілей функції управління, або визначає його як роботу, що повинна бути виконана визначеним засобом у визначений період. Проте з появою нових інформаційних технологій поняття «задача» розглядається ширше – як закінчений комплекс опрацювання інформації, що забезпечує або видачу прямих керуючих впливів на хід виробничого процесу, або видачу необхідної інформації для прийняття рішень управлінським персоналом. Таким чином, задача повинна розглядатися як елемент системи управління, а не як елемент системи опрацювання даних.

Вибір складу функціональних задач функціональних підсистем управління здійснюється звичайно з урахуванням основних фаз управління: планування; урахування, контролю й аналізу; регулювання (виконання).

Планування – це управлінська функція, що забезпечує формування планів, відповідно до яких буде організоване функціонування об'єкта управління. Звичайно виділяють перспективне (5-10 років), річне (1 рік) і оперативне (доба, тиждень, декада, місяць) планування.

Урахування, контроль і аналіз – це функції, що забезпечують одержання даних про стан керованої системи за визначений проміжок часу; визначення факту і причини відхилень фактичного стану об'єкта управління від його планованого стану, а також перебування розмірів цього відхилення. Облік ведеться по показниках плану в обраному діапазоні (обрії) планування (оперативний, середньостроковий і т.д.).

Регулювання (виконання) – це функція, що забезпечує порівняння планованих і фактичних показників функціонування об’єкта управління і реалізацію необхідних керуючих впливів при наявності відхилень від запланованих у заданому діапазоні (відрізку). Відповідно до виділених функціональних підсистем і з урахуванням фаз управління і визначається склад задач функціональних підсистем.

1.3 Класифікація інформаційних систем

Головні ідеї сучасних інформаційних технологій базуються на концепції баз даних. Відповідно до цієї концепції, основою інформаційних технологій є дані, які повинні бути організовані в бази даних у цілях адекватного відображення мінливого реального світу і задоволення інформаційних потреб користувачів. Одним з найважливіших понять в теорії баз даних є поняття інформації. Під інформацією розуміються будь-які відомості про які-небудь події, процеси, об’єкти.

Система управління базами даних (СУБД) – сукупність мовних і програмних засобів, призначених для створення, ведення і спільного використання БД багатьма користувачами [1].

Автоматизована інформаційна система – це система, що реалізує автоматизований збір, обробку, маніпулювання даними, що функціонує на основі ЕОМ та інших технічних засобів і включає відповідне програмне забезпечення та персонал. Надалі в цій якості буде використовуватися термін інформаційна система (ІС), який має на увазі поняття автоматизована.

Кожна ІС в залежності від її призначення має справу з тією або іншою частиною реального світу, яку прийнято називати предметною областю системи. Виявлення предметної області системи – це необхідний початковий етап розробки будь-якої ІС. Саме на цьому етапі визначаються інформаційні потреби всієї сукупності користувачів майбутньої системи, які, у свою чергу, зумовлюють зміст її бази даних.

Банк даних є різновидом ІС. Банк даних – це система спеціальним чином організованих даних: баз даних, програмних, технічних, мовних, організаційно-методичних засобів, призначених для забезпечення централізованого накопичення та колективного багатоцільового використання даних. Під задачами обробки даних зазвичай розуміється спеціальний клас розв'язуваних на ЕОМ задач, пов'язаних з видом, зберіганням, сортуванням, відбором по заданій умові і угрупуванням записів однорідної структури.

Окремі програми або комплекс програм, що реалізують автоматизацію рішення прикладних задач обробки даних, називаються додатками. Програми, створені засобами СУБД, відносять до додатків СУБД. Програми, створені поза середовищем СУБД з допомогою систем програмування, які використовують засоби доступу до БД, на приклад, Delphi або Visual Studio, називають зовнішніми програмами.

Інформаційні системи, створені засобами технології баз даних, інколи прийнято називати банками даних. Банк даних включає в себе [1]:

– технічні засоби;
– одну або декілька БД;
– СУБД;
– словник або каталог даних;
– адміністратора;
– обчислювальну систему;
– обслуговуючий персонал.

Словник або каталог даних служить для централізованого накопичення та опису ресурсу даних. Він містить опис предметної області, відомості про структуру БД, про зв'язки між елементами БД. Словник даних можна розглядати як частину самої бази даних. Адміністратор БД – людина або група осіб, які приймають рішення. Основні функції адміністраторів БД: участь у розробці БД, контроль правильності функціювання БД.

Обчислювальна система – включає програмні і апаратні засоби. Обслуговуючий персонал – це особи, прямими обов'язками яких є створення і підтримка коректного функціювання банку даних. Вони відповідальні за роботу банку даних і прикладного програмного забезпечення. До обслуговуючого персоналу відносяться: розробники й адміністратори бази даних, аналітики, програмісти.

1.4 PHP фреймворки для веброзробки

У світі розробки програмного забезпечення термін «framework» відноситься до бібліотек файлів, які включають кілька основних функцій. Мета фреймворка – надати основу, яку можна використовувати для більш ефективної розробки проєктів. Щоб зробити це, він буде включати в себе безліч функцій, які потрібно буде кодувати з нуля, якщо необхідно почати з початку.

Фреймворки функціонують однаково. Якщо ви хочете розробити новий PHP-додаток, найкраще почати з пошуку фреймворка, який включає в себе всі функції, які вам знадобляться. Зараз є багато варіантів на вибір. Всі вони унікальні в тому, як вони підходять до розвитку, і у всіх є свої плюси і мінуси.

Види фреймворків.

Zend Framework. Цей програмний продукт використовують багато професійних PHP-програмістів. Він робить життя web-розробника легше, перш за все, тому, що містить безліч корисних бібліотек. До них відносяться можливість інтеграції проєкту з YouTube і інших сервісів, спрощення роботи з базами даних, користувачами, кешуванням та ін.

Bootstrap. Популярний фреймворк, що допомагає швидко і якісно верстати макети сайтів. Включає в себе шаблони для створення шарів, кнопок, форм, блоків навігації та інших елементів web-сторінок.

Yii. Об'єктно-орієнтована фреймворк для створення масштабних web-додатків: інтернет-магазинів, CRM-систем та ін. Головними перевагами Yii є висока продуктивність і безпека.

Corona SDK. Найбагатший інструментарій для розробки ігор і додатків для Android. Його відмінною рисою є те, що він працює на власній мові програмування, який носить назву Lua.

1.5 Огляд фреймворку Laravel

Найбільш поширеним є PHP framework Laravel. Ця специфічна структура відома своїм елегантним синтаксисом, який легко зрозуміти і з яким приємно працювати (рис. 1.1).

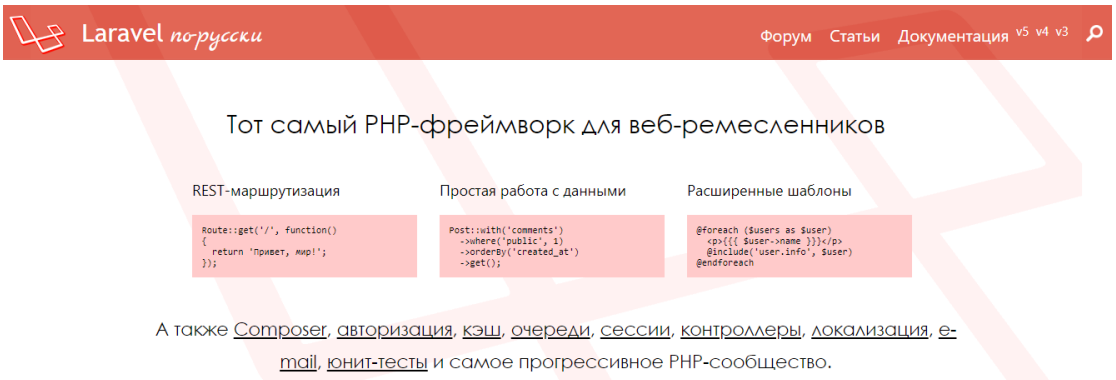

Рисунок 1.1 – Фреймворк Laravel

З Lavarel ви можете швидко приступити до роботи над своїми проєктами. Ви також зможете пропустити багато основ, оскільки ви отримуєте доступ до таких функцій, як автентифікація користувачів, управління сеансами і кешування. В цілому, Laravel володіє всіма функціональними можливостями, які вам знадобляться для створення сучасного PHP-додатка.

Ядро Laravel є надійним з точки зору продуктивності, і ви можете розширити платформу, використовуючи безліч доповнень.

Laravel також прекрасно інтегрується з іншими сторонніми бібліотеками і платформами, такими як Amazon Web Services (AWS), що дозволяє створювати високомасштабіруемие додатки. Для довгострокових завдань ви можете поставити їх в чергу для асинхронного виконання у фоновому режимі, що ще більше підвищує продуктивність.

Ключові особливості:

– використовуйте структуру, яка пишається елегантним синтаксисом;

– розширте базову функціональність Laravel, використовуючи доповнення;

– використовуйте вбудовані функції для управління маршрутизацією, управлінням користувачами, кешуванням і багатьом іншим;

– інтегруйте Laravel зі сторонніми бібліотеками і платформами, такими як AWS;

– запускайте завдання асинхронно у фоновому режимі для підвищення продуктивності.

Нарешті, Laravel може похвалитися високоактивним співтовариством, що означає, що пошук допомоги або навчальних посібників ніколи не буде проблемою. Якщо ви використовуєте фреймворк вперше, це робить Laravel ще кращим варіантом.

1.6 Фреймворк Angular

AngularJS – це каркас для розробки веб-додатків від Google. Добре підходить для динамічних веб-додатків, з використанням HTML для статичних веб-сторінок. Незамінний фреймворк не тільки для розробників ПЗ, але і для дизайнерів. AngularJS, Angular 2 і Angular 4 міцно влаштувалися серед затребуваних фреймворків.

Переваги:

– відкритий вихідний код;

– збереження фрагментів коду для подальшого використання;

– розробники стикаються з меншою кількістю помилок, оскільки прив'язка даних будується на базі Angular-елементів;

– підтримуються різні елементи MVC;

– добре працює в середовищі Agile;

– маса інструментів для тестів.

Недоліки:

– складний для новачків;

– Vue простіше в плані архітектури;

– API Angular величезна, і потрібно розібратися з багатьма концепціями.

1.7 Огляд фреймворку Node.js

Він дозволяє створювати легкі і швидкі програми. Сучасний, простий, швидкий і неблокіруючій. GoDaddy і Paypal – лише деякі з відомих компаній, які використовують Node.js. Ідеально підходить для додатків, пов'язаних з I/O і додатків для потокової передачі даних (рис. 1.2).

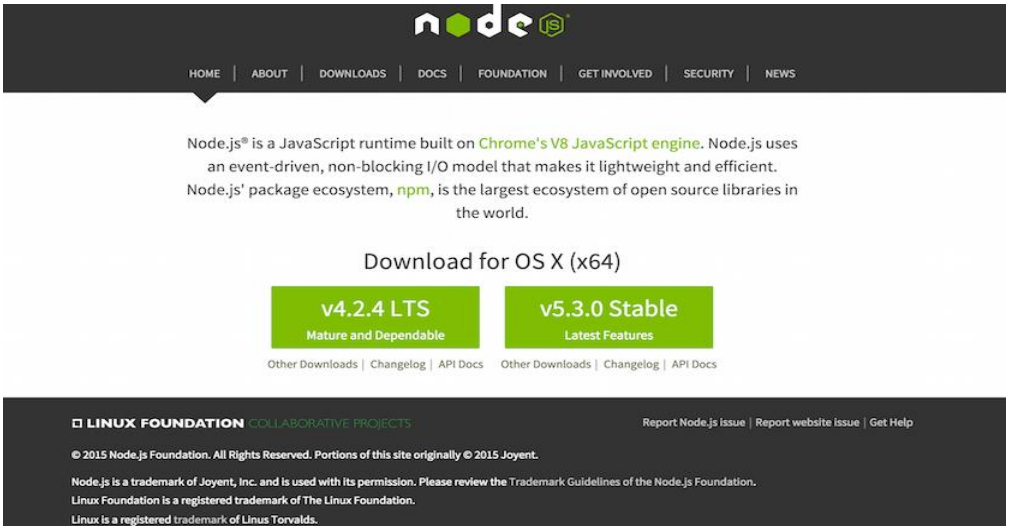

Рисунок 1.2 – Фреймворк Node.js

Переваги фреймворку:

– простий і швидкий;
– такі ПО можуть працювати на декількох хостах;
– включення швидких серверів.

Недоліки фреймворку:

– чи не для «некрізних» операцій;
– без тестів в Node.js робити нічого.

1.8 Yii – фреймворк на PHP

Yii – це популярний фреймворк для php-розробки, заснований на парадигмі MVC. Основна перевага – дуже висока швидкість роботи і, як наслідок, продуктивність.

Фреймворк активно розвивається спільнотою. Yii набагато простіший в порівнянні з фреймворками Symfony і Zend Framework, кодова база яких вельми об'ємна. Фреймворк Yii досить простий в освоєнні і у використанні, що сприяє швидкій розробці на ньому проєктів. Однак при виборі в якості платформи для створення веб-проєкту цього фреймворка варто все одно враховувати той факт, що швидкість розробки на ньому типових рішень все одно буде нижче, ніж розробка на CMS. Як і всі фреймворки, Yii «заточений» під розробку технічно складних веб-проєктів: бізнес-додатків, веб-сервісів, а також сайтів зі складною бізнес-логікою і вимогливих до швидкодії.

Основні переваги і можливості фреймворка Yii:

– забезпечує високу продуктивність щодо інших php-фреймворків;

– заснований на парадигмі MVC (Модель-Представлення-Контролер);

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

– підтримує інтернаціоналізацію;

– дозволяє кешувати як сторінки цілком, так і окремі фрагменти;

– здійснює перехоплення і обробка помилок;

– має функціонал роботи з формами, забезпечує їх побудова та валідацію;

– реалізовано аутентифікація і авторизація;

– зручний для реалізації AJAX-інтерфейсів, інтегрується з jQuery;

– у фреймворк вбудовані генератори базового PHP-коду для CRUD-операцій (скаффолдинг);

– підтримує теми оформлення;

– має можливість підключення сторонніх бібліотек;

– працює з міграціями баз даних (генерація, застосування і відкат);

– дозволяє здійснювати автоматичне тестування і вести розробку в стилі TDD;

– підтримує стиль REST.

2 ПРОЄКТУВАННЯ МОДЕЛІ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вимоги до створення сайту

Для створення сайту необхідно виконати певні вимоги щодо його реалізації, створення розділів, створення авторизації користувачів та інші. Вимоги до реалізації сайту такі:

– мова програмування PHP;

– використання бази даних MySQL для зберігання інформації;

– сервер Apache;

– сайт має мати високу бистродію;

– архітектура MVC.

Сайт повинен містити такі розділи:

– питання – відповіді(основний розділ сайту, в якому користувач ставить своє питання, а адміністратор дає відповідь);

– FAQ (часті запитання);

– інформація (інформація про структуру, цінову політику, апаратне та програмне забезпечення).

Сайт повинен підтримувати написання питання українською, російською та англійською мовами.

На сайті повинна бути реалізована авторизація.

Сайт повинен мати такі ролі користування:

– адміністратор (зареєстрований користувач, якому присвоєна роль адміністратора), має мати такі права доступу:

– редагування та видалення питань;

– відповідь на питання;

– перенесення питання до розділу частих запитань;

– заповнення інформаційних розділів інформацією.

Гість має мати такі права доступу:

– перегляд сайту;

– можливість написання запитання в розділі «Питання –відповіді».

Вимоги до функціоналу сайту:

– написання повідомлень користувачами;

– можливість адміністратора відповідати на повідомлення;

– захист від спаму(для реалізації буде використана капча);

– можливість адміністратора перенести питання з розділу “Питання – відповіді” до розділу “FAQ”(перенесення з редагуванням і без редагування);

– можливість адміністратора редагувати повідомлення користувача;

– редагування адміністратором контенту розділів “Інформація” та “FAQ”;

– відповідь на питання посиланням на відповідь, що була раніше (при цьому відповідь, на яку вказує посилання, повинна підсвічуватись);

– видалення адміністратором запитань користувачів;

– валідація форм.

Вимоги до дизайну сайту:

– сайт повинен бути адаптивним та кроссбраузерним;

– чіткий розподіл пари питання – відповідь;

– підтримка роботи на мобільних пристроях.

2.2 Процес розробки додатку з використанням Yii

Розглянувши фундаментальні концепції Yii, доцільно описати загальний процес створення веб-додатків з використанням фреймворку. Процес ґрунтується на тому, що аналіз вимог уже проведений, як і аналіз структури додатку.

Основні етапи при розробці додатку виглядають так:

Створення структури директорій. Утиліта yii може бути використана для того, щоб прискорити цей процес.

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

Створення класу моделі для кожного використовуваного типу даних. Для автоматичної генерації всіх необхідних моделей Active Record можна скористатися інструментом Gii, описаним в розділах «створення першого додатку» та «автоматична генерація коду».

Створення класу контролера для кожного типу користувальницького 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.

Створення дій і уявлень. Саме тут і відбувається основна робота.

Конфігурація необхідних фільтрів для дій в класах контролерів.

Створення тем оформлення при необхідності.

Переклад повідомлень в разі, коли потрібна локалізація додатку.

Виявлення даних і уявлень, які можуть бути закешовані, і застосування відповідних технік кешування.

Налаштування продуктивності і розгортання.

2.3 Концептуальна та логічна модель даних

Концептуальне проєктування – збір, аналіз і редагування вимог до даних. Для цього здійснюються наступні заходи:

а) обстеження предметної області, вивчення її інформаційної структури;

б) виявлення всіх фрагментів, кожний з яких характеризується призначеним для користувача представленням, інформаційними об'єктами і зв'язками між ними, процесами над інформаційними об'єктами;

в) моделювання і інтеграція всіх представлень.

Логічне проєктування – перетворення вимог до даних в структури даних.

На виході отримуємо СУБД-орієнтовану структуру бази даних і специфікації прикладних програм. На цьому етапі часто моделюють бази даних стосовно різних СУБД і проводять порівняльний аналіз моделей.

Для досягнення наочності в представлені концептуальної і зовнішніх схем бази даних були розроблені і зараз активно використовуються графічні моделі. Графічні семантичні моделі надають можливість формального і разом з тим наочного опису предметної області. Частини предметної області, що відповідають об’єктам, властивостям і зв’язкам зображуються у вигляді діаграм.

Фізичне проєктування – визначення особливостей зберігання даних, методів доступу, фізичне представлення бази даних в комп'ютері і т.д. На цьому рівні описується, як інформація зберігається в базі даних.

2.4 Проєктування структури сайту

Розроблена структура Web-сайту впливає на його навігаційну схему, від цього залежить як будь-який користувач зможе по ньому пересуватися і отримувати доступ до запропонованої інформації. Одним з найважливіших факторів є простота і зручність навігації. Від них залежить відвідуваність Web-сайту. Відвідувачі сайту повинні мати можливість швидко і просто перейти на будь-яку сторінку Web-сайту, в тому числі на головну. Загальне уявлення про сайт, його структуру та принцип роботи починають формуватися вже на цьому етапі. Відразу ж потрібно чітко продумати назви майбутніх розділів сайту, заголовки до сторінок, продумати та визначити переходи між ними, щоб розміщення інформації на сайті було точним і логічним. Погано структуровані сайти можуть погано впливати на відвідуваність сайту, а користувачі можуть не знайти потрібну інформацію під час їх перегляду.

Деякі особливості проєктування структури Web-сайту:

– необхідно дотримуватись одноманітності елементів: при розробці структури слід визначити ієрархію об'єктів (наприклад, кожен розділ включає в себе певні підрозділи і т.д.);

– слід уникати створення подібних сторінок: якщо однотипна інформація може бути розміщена на одній сторінці, не слід розробляти для неї окрему;

– не потрібно створювати розділи, які дублюють один одного;

– потрібно передбачити простий і швидкий доступ до всіх найбільш важливих розділів з усіх сторінок сайту;

– проєктування структури сайту має включати складання карти основних сторінок: обов'язково треба продумати які розділи найбільш необхідні для користувача і виключити ті, які будуть абсолютно марними;

– потрібно придумати свій ідеальний варіант структури, повністю відповідний до тематики ресурсу.

Для того, щоб розробка структури веб-сайту була максимально комфортною, її можна відобразити за допомогою комп'ютерних програм, таких як MS Visio, Power Point або звичайний Paint. Таким чином, можна в графічному вигляді скласти набір необхідних сторінок, а також продумати їх зв'язки один з одним. Також можна скористатися звичайною ручкою і папером, щоб намалювати структуру ресурсу. Варто зазначити, що для подібних цілей існує спеціалізоване ПО, але його використання не доцільно при створенні невеликих сайтів.

2.5 Послідовність роботи фреймворка Yii. Модель-представлення-контролер (MVC)

Yii використовує шаблон проєктування Модель-Представлення-Контролер (MVC, Model-View-Controller), який широко застосовується в веб-

програмуванні.

MVC призначений для поділу бізнес-логіки і призначеного для 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 користувальницький інтерфейс (наприклад, текст, поля введення), а контролер забезпечує взаємодію між моделлю і представленням.

Крім цього, Yii використовує фронт-контролер, званий додатком (application), який інкапсулює контекст обробки запиту. Додаток збирає інформацію про запит і передає її для подальшої обробки відповідного контролера[10].

Цей шаблон передбачає поділ системи на три взаємопов'язані частини:

а) модель даних – надає дані і реагує на команди контролера, змінюючи свій стан;

б) вигляд (інтерфейс користувача) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі;

в) модуль керування інтерпретує дії користувача, сповіщаючи модель про необхідність змін.

Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача, схематично зображено на рисунку 2.1.

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

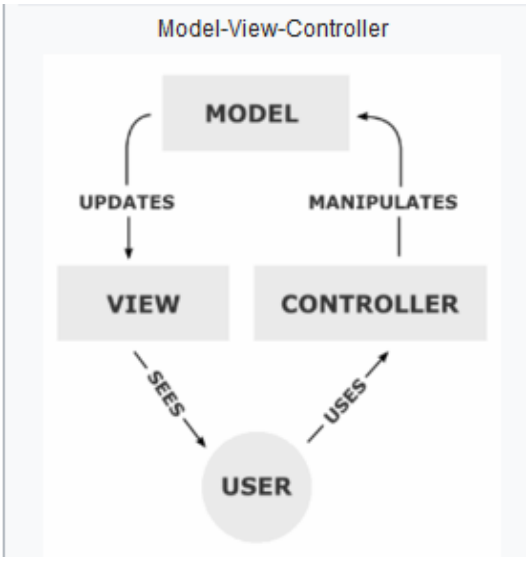

Рисунок 2.1 – Схематичне зображення MVC

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (уявлення, виду). За рахунок такого поділу підвищується можливість повторного використання коду. Найбільш корисне застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах або з різних точок зору. Зокрема, виконуються наступні завдання:

– до однієї моделі можна приєднати кілька видів, при цьому не зачіпаючи реалізацію моделі (наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і кругової діаграми);

– не торкаючись реалізацію видів, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних) – для цього досить використовувати інший контролер;

– ряд розробників спеціалізується тільки в одній з областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку, тому можливо добитися того, що програмісти, які займаються розробкою бізнес-логіки (моделі), взагалі не будуть обізнані про те, яке представлення буде використовуватися.

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом

функцій між компонентами. Модель (Model) відповідає за зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель зберігає ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проєкті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер вносить зміни до моделі даних, то інформація,

подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

2.6 Додатки фреймворка Yii2

Yii додатки організовані згідно шаблону проєктування модель-уявлення-контролер (MVC). Моделі представляють собою дані, бізнес логіку і бізнес правила; уявлення відповідають за відображення інформації, в тому числі і на основі даних, отриманих з моделей; контролери приймають вхідні дані від користувача і перетворюють їх в зрозумілий для моделей формат і команди, а також відповідають за відображення потрібного уявлення [9].

Крім MVC, Yii додатки також мають наступні сутності:

– вхідні скрипти: це PHP скрипти, які доступні безпосередньо кінцевому користувачеві програми. Вони відповідальні за запуск і обробку вхідного запиту;

– додатки – це глобально доступні об'єкти, які здійснюють коректну роботу різних компонентів програми та їх координацію для обробки запиту;

– компоненти програми – це об'єкти, зареєстровані в додатку і надають різні можливості для обробки поточного запиту;

– модулі – це самодостатні пакети, які включають в себе повністю всі кошти для MVC. Додаток може бути організовано за допомогою декількох модулів;

– фільтри – це код, який повинен бути виконаний до і після обробки запиту контролерами;

– віджети – це об'єкти, які можуть включати в себе уявлення. Вони можуть містити різну логіку і бути використані в різних уявленнях.

Нижче на діаграмі представлена структурна схема програми (див. рис. 2.2).

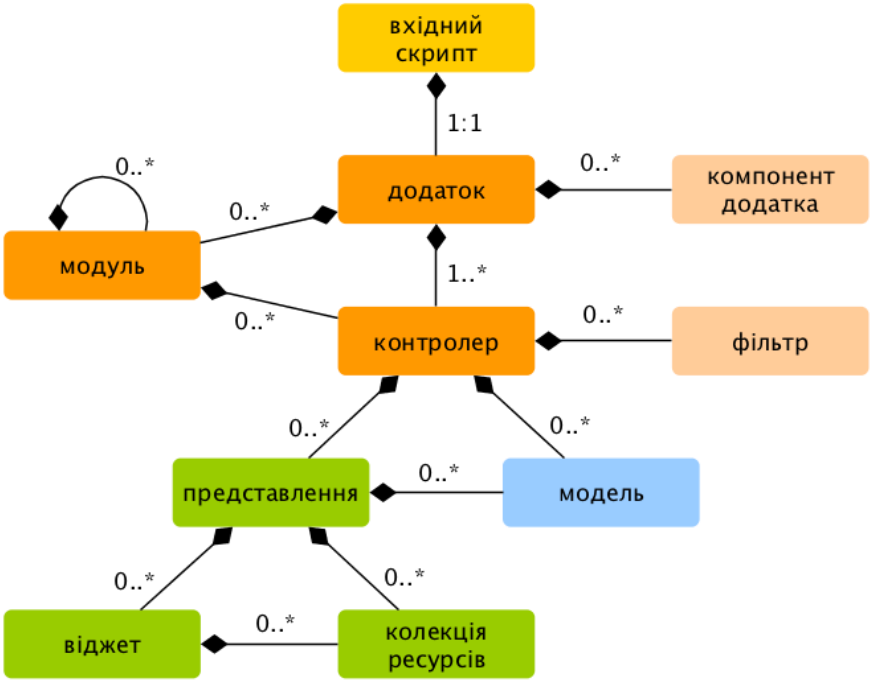

Рисунок 2.2 – Структурна схема MVC

Загальний процес створення веб-додатків з використанням фреймворку має такі етапи:

– створення структури директорій: утиліта yiic може бути використана для того, щоб прискорити цей процес;

– конфігурація додатка шляхом модифікації файлу конфігурації програми: цей етап також може запропонувати написання деяких компонентів програми (наприклад, компонента управління користувачами);

– створення класу моделі для кожного використовуваного типу даних: для автоматичної генерації всіх необхідних моделей Active Record можна скористатися інструментом Gii;

– створення класу контролера для кожного типу користувальницького запиту: класифікація призначених для користувача запитів залежить від поточних вимог і, у загальному випадку, якщо клас моделі використовується користувачем, повинен існувати відповідний клас контролера; утиліта Gii також може автоматизувати цей процес;

– створення дій і уявлень: саме тут і відбувається основна робота;

– конфігурація необхідних фільтрів для дій в класах контролерів;

– створення тем оформлення при необхідності;

– переклад повідомлень в разі, коли потрібна локалізація додатків;

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

– налаштування продуктивності і розгортання.

2.7 Концептуальна та логічна модель даних

Концептуальне проєктування – збір, аналіз і редагування вимог до даних. Для цього здійснюються наступні заходи:

1) обстеження предметної області, вивчення її інформаційної структури;

2) виявлення всіх фрагментів, кожний з яких характеризується призначеним для користувача представленням, інформаційними об'єктами і зв'язками між ними, процесами над інформаційними об’єктами;

3) моделювання і інтеграція всіх представлень.

Логічне проєктування – перетворення вимог до даних в структури даних.

На виході отримуємо СУБД-орієнтовану структуру бази даних і специфікації прикладних програм. На цьому етапі часто моделюють бази даних стосовно різних СУБД і проводять порівняльний аналіз моделей.

Для досягнення наочності в представлені концептуальної і зовнішніх схем бази даних були розроблені і зараз активно використовуються графічні моделі. Графічні семантичні моделі надають можливість формального і разом з тим наочного опису предметної області. Частини предметної області, що відповідають об’єктам, властивостям і зв’язкам зображуються у вигляді діаграм.

Відповідну до предметної області концептуальну модель розташування туристичних рекреаційних зон у відповідних населених пунктах можна

зобразити у вигляді діаграми, як показано на рисунку 2.3. Кожна область країни має певну кількість рекреаційних ресурсів, котрі розташовані у певних населених пунктах і мають своє призначення.

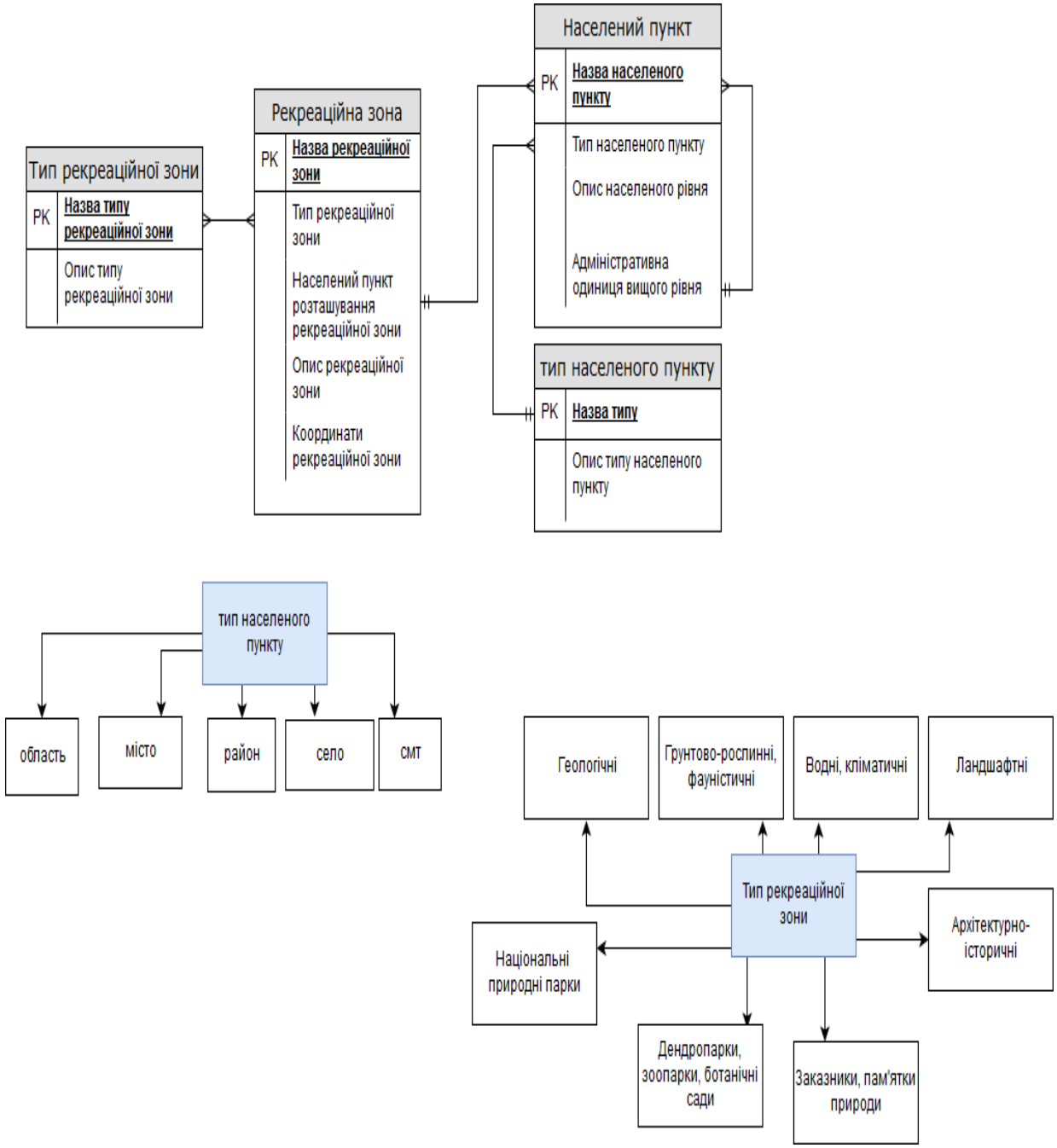

Рисунок 2.3 – Концептуальна модель даних

Фізичне проєктування – визначення особливостей зберігання даних, методів доступу, фізичне представлення бази даних в комп'ютері і т.д. На цьому рівні описується, як інформація зберігається в базі даних.

3 ЕТАПИ РОЗРОБКИ WEB – РЕСУРСУ НА БАЗІ PHP ФРЕЙМВОРКУ YII

3.1 Встановлення Yii на локальний сервер

Для початку роботи над створенням вебресурсу необхідно встановити на комп’ютер необхідні програми: Open Server Panel, PhpStorm, git.

Рисунок 3.1 – Успішне встановлення Yii 1.1

Також для роботи потрібно скачати Yii з сайту http://www.yiiframework.com/ та розпакувати його в папку сайту на локальному сервері(C:\WebSever\OpenServer\domains\GuestBook). Етапи встановлення Yii на локальний сервер можуть мати наступний вигляд:

– запускаємо командний рядок;

– вводимо команду: cd C:\;

– вводимо команду: WebSever\OpenServer\modules\php\PHP-5.4\php.exe C:\WebSever\OpenServer\domains\GuestBook\framework\yiic webapp C:\WebSever\OpenServer\domains\GuestBook\;

– далі відобразиться, повідомлення про те чи бажаємо створити додаток; вводимо: yes.

Отримано результат як зображено на рисунку 3.1.

Для перевірки того, чи дійсно створений додаток, потрібно ввести в браузері назву папки, в яку розпакований фреймворк : guestbook/ Отриманий результат зображено на рисунку 3.2.

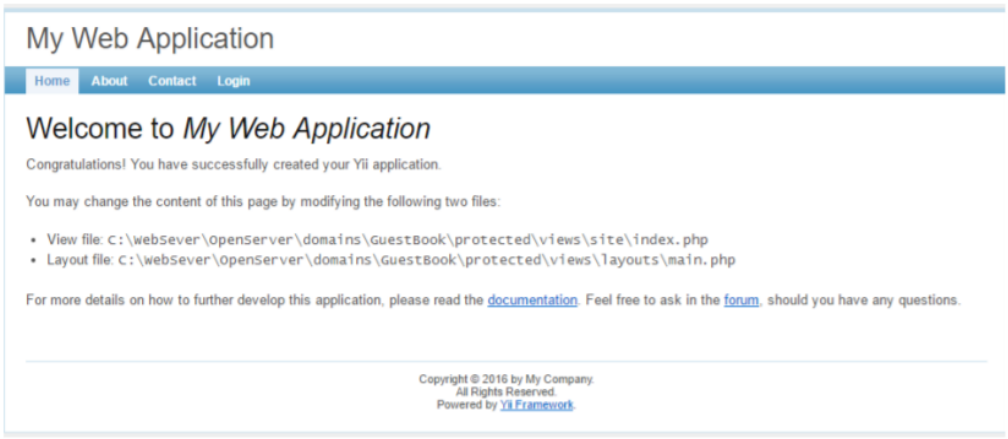

Рисунок 3.2 – Стандартно-згенерований додаток Yii

3.2 Робота фреймворка Yii2 з базою даних

Побудовані поверх PDO, Yii DAO (об'єкти доступу до даних) забезпечують об'єктно-орієнтована API для доступу до реляційних баз даних.

Це основа для інших, більш просунутих, методів доступу до баз даних, включаючи будівник запитів і active record.

При використанні Yii DAO в основному використовується чистий SQL і масиви PHP. Як результат, це найефективніший спосіб доступу до баз даних. Проте, так як синтаксис SQL може відрізнятися для різних баз даних, використовуючи Yii DAO потрібно докласти додаткових зусиль, щоб зробити додаток не залежних від конкретної бази даних.

Yii DAO з коробки підтримує наступні бази даних:

– MySQL;
– MariaDB;
– SQLite;
– PostgreSQL;
– CUBRID;
– Oracle;
– MSSQL.

Для створення проєкту Yii2 з використанням composer, а саме: composer create-project – prefer-dist yiisoft/yii2-app-basic relax.loc.

Для створення міграції для проєкту з існуючої бази даних необхідно використати наступне: php yii migrate / create init_tables. Перед цим необхідно налаштувати зв'язок з БД у файлі config / db.php (рис. 3.3).

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

Рисунок 3.3 – Налаштування зв’язку з БД

Можна налаштувати декілька компонентів підключення, якщо в додатку використовується кілька баз даних.

Під час налаштування підключення, обов'язково потрібно вказувати ім'я джерела даних через параметр dsn. Формат DSN відрізняється для різних баз даних.

3.3 Створення головної сторінки

Головна сторінка сайту (доменне ім'я, index.html) повинна бути оптимізована за словосполученням, які найбільшою мірою важливі для всього змісту сайту. Саме головна сторінка – перший кандидат на потрапляння в топ пошукових систем.

Для посторінкового відображення матеріалів на домашній сторінці сайту в класі-контролері веб-додатки SiteController потрібно змінити метод actionIndex (див. рис. 3.4).

```
        public function actionIndex()
        {
            $query = RelaxRegion::find()->orderBy('name');
            $pages = new Pagination(['totalCount' => $query->count(), 'pageSize' => 2]);
            $relax_regions          =$query->offset($pages->offset)->limit($pages->limit)-
>all();
            return $this->render('index', compact('relax_regions', 'pages'));
        }
```

Рисунок 3.4 – Фрагмент коду

В змінну $query витягуються все з моделі RelaxRegion (рекреаційні ресурси) з сортуванням по назві. Потім в змінній $pages створюється компонент посторінкового виведення із зазначенням кількості записів і кількості виведених на сторінку. Далі в змінній $relax_regions задається підмножина записів із зсувом і лімітом з компонента сторінок. Вкінці, при рендері сторінки index за допомогою методу compact в уявлення домашньої

сторінки передаються змінні relax_regions і pages.

У представленні домашньої сторінки (файл view / site / index.php) здійснюється виведення коротких відомостей про рекреаційні зони. Заголовки є посиланнями на детальне уявлення.

Нижче на рисунку 3.5 наведено програмний код для створення головної сторінки.

```
<?php
use yii\helpers\Html;
use yii\widgets\LinkPager;
$this->title = 'Інформаційна система рекреаційних ресурсів України';
?>
<div class="site-index">
    <div class="jumbotron">
        <h1>Інформаційна система рекреаційних ресурсів України</h1>
    </div>
    <div class="body-content">
        <div class="row">
            <div class="col-lg-12">
                <?php if (!empty($relax_regions)): ?>
                    <?php foreach ($relax_regions as $rr): ?>
                        <div class="panel panel-default">
                            <div class="panel-heading">
                                <h2class="panel-title"><a
href="<?=\yii\helpers\Url::to(['relax-region/view', 'id' => $rr->id]); ?>"><?=$rr-
>name; ?></a></h2><?=$rr->city->nameWithParent; ?>
                            </div>
                            <div class="panel-body"><?=$rr->description; ?></div>
                            <div class="panel-footer">
                                Типи ресурсів:
                                <?php
                                $types = $rr->types;
                                foreach ($types as $type): ?>
                                    <?=$type['name']; ?>;
                                <?php endforeach; ?></div>
                        </div>
                    <?php endforeach; ?>
                <?php endif; ?>
            </div>
        </div>
        <?=LinkPager::widget(['pagination' => $pages]) ?>
    </div>
</div>
```

Рисунок 3.5 – Програмний код для створення головної сторінки

В результаті отримаємо головну сторінку інформаційної системи, як зображено на рисунку 3.6.

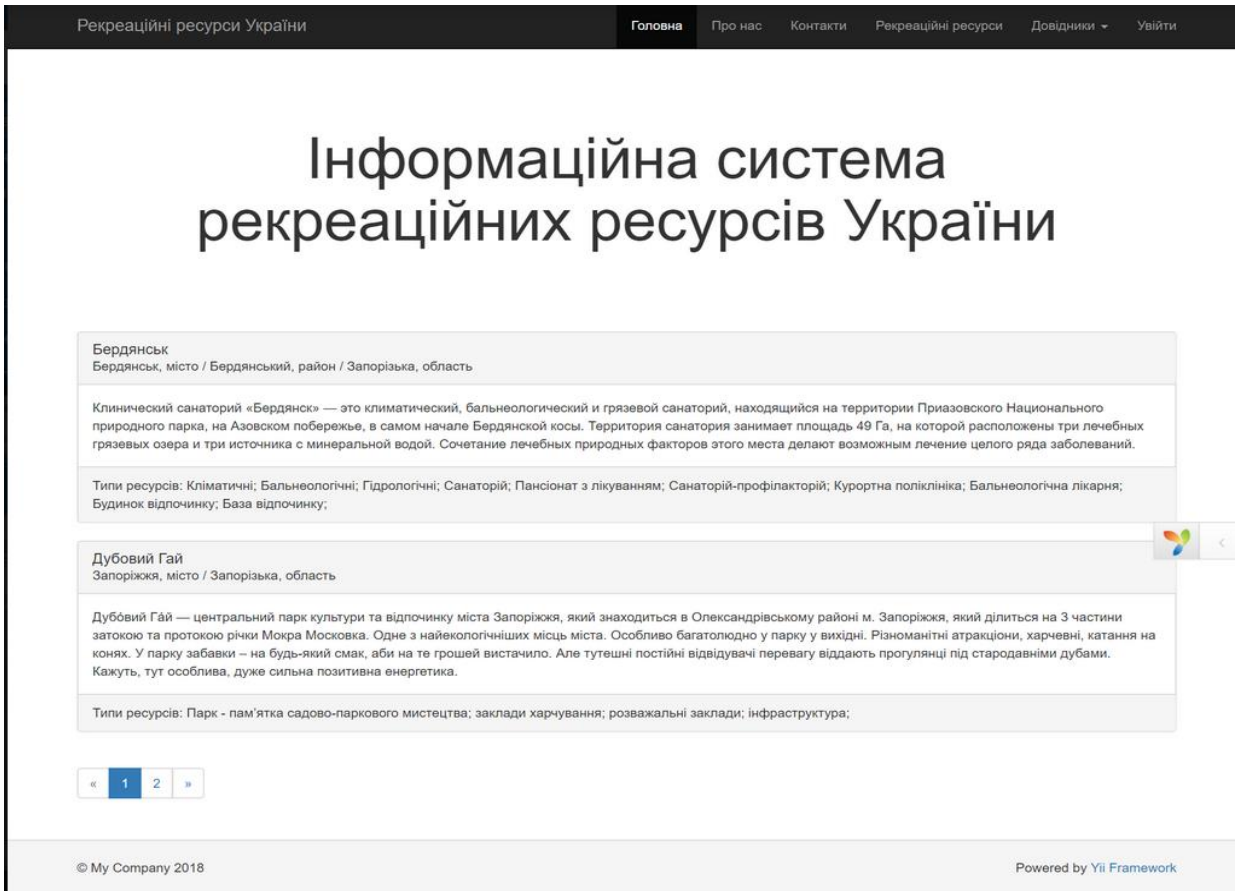

Рисунок 3.6 – Головна сторінка інформаційної системи

3.4 Створення підрозділів інформаційної системи

Для відображення підрозділу «Тип адміністративної одиниці» необхідно створити для нього модель та представлення. Для створення моделі використаємо програмний код, представлений на рисунку 3.7.

Далі необхідно створити представлення за допомогою такого коду, який наведено на рисунку 3.8.

```
class CityType extends \yii\db\ActiveRecord
{
    public static function tableName()
```

```
        {
            return 'city_type';
        }
        /**
         * {@inheritdoc}
         */
        public function rules()
        {
            return [
                [['name'], 'required'],
                [['description'], 'string'],
                [['name'], 'string', 'max' => 50],
            ];
        }
        public function attributeLabels()
        {
            return [
                'id' => 'Ідентифікатор',
                'name' => 'Назва',
                'description' => 'Опис',
            ];
        }
        public function getCities()
        {
            return $this->hasMany(City::className(), ['type_id' => 'id']);
        }
    }
```

Рисунок 3.7 – Створення моделі

```
    $this->title = 'Типи адміністративних одиниць';
    $this->params['breadcrumbs'][] = $this->title;
    ?>
    <div class="city-type-index">
        <h1><?= Html::encode($this->title) ?></h1>
        <?php Pjax::begin(); ?>
        <?php // echo $this->render('_search', ['model' => $searchModel]); ?>
        <p>
            <?= Html::a('Додати новий тип адміністративної одиниці', ['create'],
['class' => 'btn btn-success']) ?>
        </p>
        <?= GridView::widget([
            'dataProvider' => $dataProvider,
            'filterModel' => $searchModel,
            'columns' => [
                ['class' => 'yii\grid\SerialColumn'],
                'id',
                'name',
```

```
                'description:ntext',
                ['class' => 'yii\grid\ActionColumn'],
            ],
        ]); ?>
        <?php Pjax::end(); ?>
    </div>
```

Рисунок 3.8 – Створення представлення

В результаті отримаємо сторінку на нашому сайті, як зображено на рисунку 3.9, з різними типами адміністративних одиниць, наприклад, село, місто, селище міського типу. Кожен тип має свій ідентифікатор, у створеній базі даних, має свою назву та загальний опис. Також кожен тип можна як просто переглядати, так редагувати і видаляти.

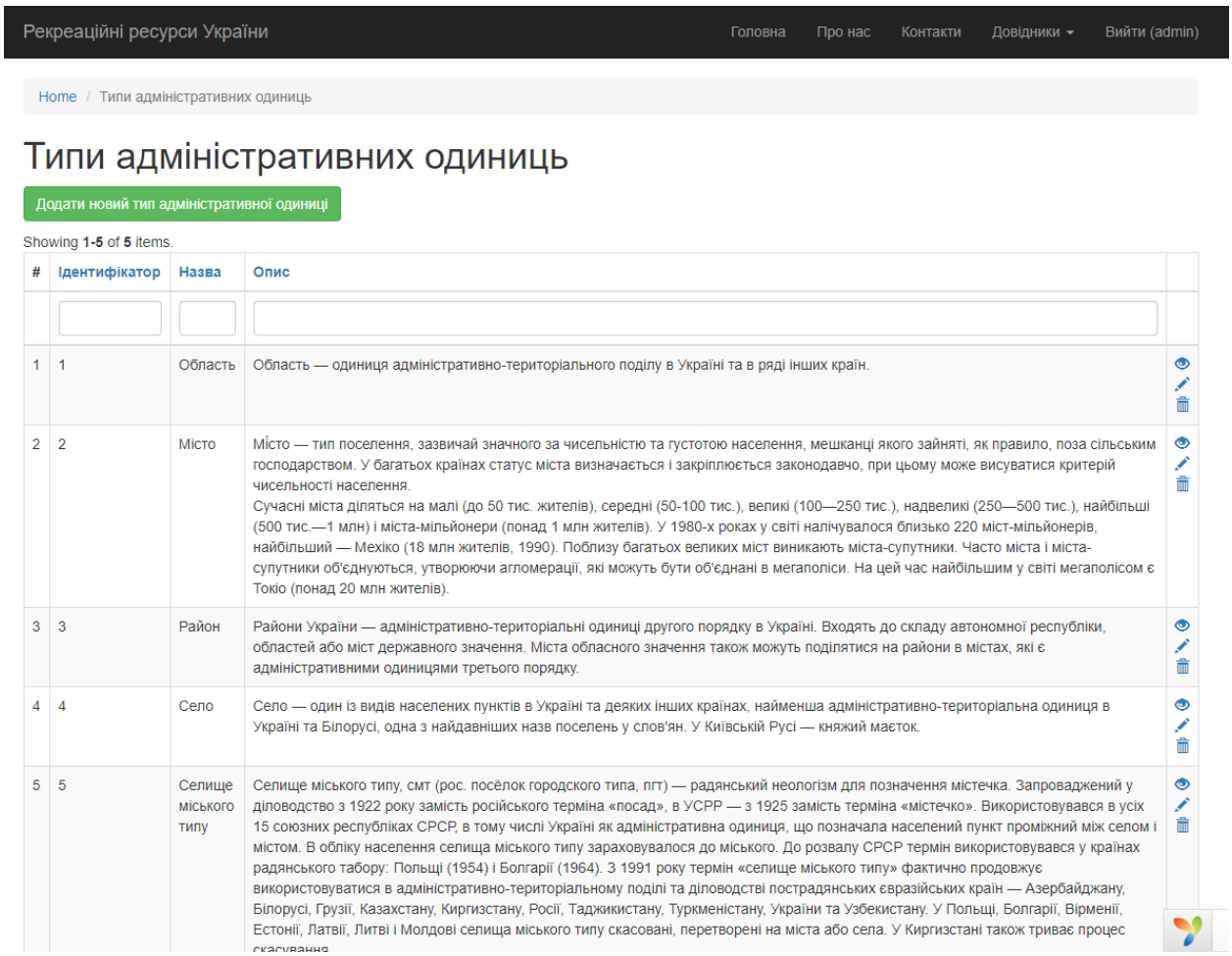

Рисунок 3.9 – Створення підрозділу «Типи адміністративних одиниць»

Для перегляду необхідно використати програмний код, представлений

на рисунку 3.10.

```
        $this->title = $model->name;
        $this->params['breadcrumbs'][] = ['label' => 'Типи адміністративних одиниць', 'url' => ['index']];
        $this->params['breadcrumbs'][] = $this->title;
        ?>
        <div class="city-type-view">
            <h1><?= Html::encode($this->title) ?></h1>
            <p>
                <?= Html::a('Редагувати', ['update', 'id' => $model->id], ['class' => 'btn btn-primary']) ?>
                <?= Html::a('Видалити', ['delete', 'id' => $model->id], [
                    'class' => 'btn btn-danger',
                    'data' => [
                        'confirm' => 'Ви підтверджуєте видалення запису?',
                        'method' => 'post',
                    ],
                ]) ?>
            </p>
            <?= DetailView::widget([
                'model' => $model,
                'attributes' => [
                    'id',
                    'name',
                    'description:ntext',
                ],
            ]) ?>
        </div>
```

Рисунок 3.10 – Програмний код для перегляду

В результаті отримаємо вікно для редагування та видалення записів (рис. 3.11).

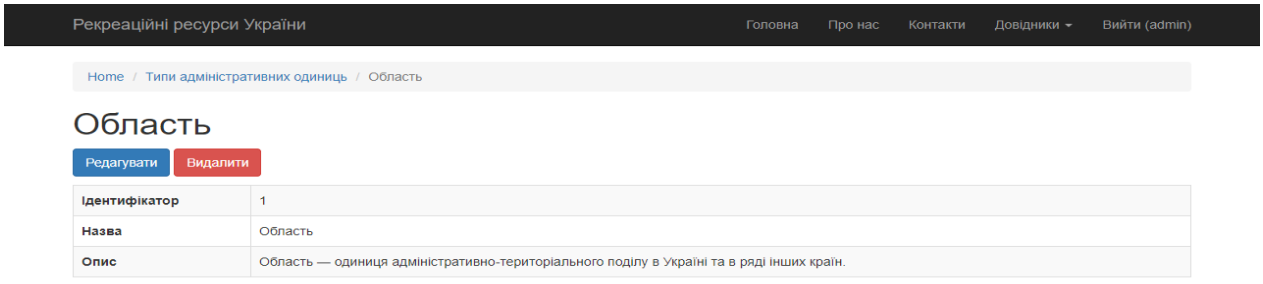

Рисунок 3.11 – Редагування та видалення записів

Для створення форми оновлення типів адміністративних одиниць використаємо код, який наведено на рисунку 3.12.

```
        $this->title = 'Форма оновлення типу адміністративної одиниці: ' . $model->name;
        $this->params['breadcrumbs'][] = ['label' => 'Типи адміністративних одиниць', 'url' => ['index']];
        $this->params['breadcrumbs'][] = ['label' => $model->name, 'url' => ['view', 'id' => $model->id]];
        $this->params['breadcrumbs'][] = 'Update';
        ?>
        <div class="city-type-update">
            <h1><?= Html::encode($this->title) ?></h1>
            <?= $this->render('_form', [
                'model' => $model,
            ]) ?>
        </div>
```

Рисунок 3.12 – Код для створення форми оновлення типів адміністративних одиниць

В результаті отримаємо форму, як зображено на рисунку 3.13.

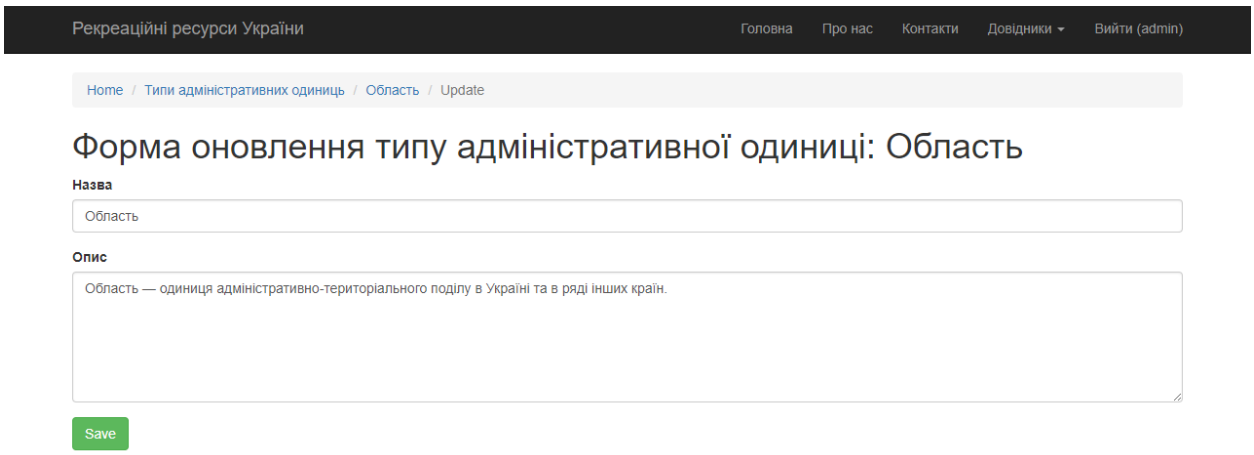

Рисунок 3.13 – Форма оновлення типів адміністративних одиниць

Для створення форми додавання нового типу адміністративної одиниці використаємо програмний код з рисунку 3.14.

В результаті отримаємо форму, як зображено на рисунку 3.15.

```
        $this->title = 'Форма додавання нового типу адміністративної одиниці';
        $this->params['breadcrumbs'][] = ['label' => 'Типи адміністративних одиниць', 'url' => ['index']];
        $this->params['breadcrumbs'][] = $this->title;
        ?>
        <div class="city-type-create">
            <h1><?= Html::encode($this->title) ?></h1>
            <?= $this->render('_form', [
                'model' => $model,
            ]) ?>
        </div>
```

Рисунок 3.14 – Код для створення форми додавання нового типу адміністративної одиниці

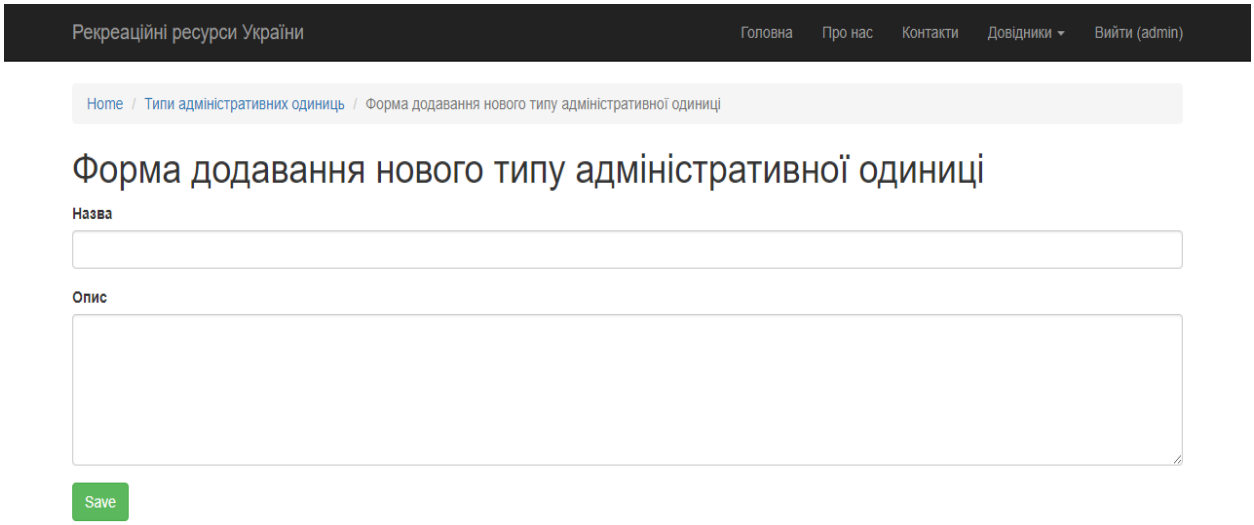

Рисунок 3.15 – Форма додавання нового типу адміністративних одиниць

3.5 Створення довідників адміністративних одиниць

Для створення довідників адміністративних одиниць спочатку необхідно створити модель (див. рис. 3.16).

```
    class City extends \yii\db\ActiveRecord
    {
        public static function tableName()
        {
            return 'city';
        }
```

```
        /**
        public function rules()
        {
            return [
                [['name', 'type_id'], 'required'],
                [['type_id', 'parent_id'], 'integer'],
                [['description'], 'string'],
                [['name'], 'string', 'max' => 50],
                [['type_id'], 'exist', 'skipOnError' => true, 'targetClass' => CityType::className(), 'targetAttribute' => ['type_id' => 'id']],
                [['parent_id'], 'exist', 'skipOnError' => true, 'targetClass' => City::className(), 'targetAttribute' => ['parent_id' => 'id']],
            ];
        }
        public function attributeLabels()
        {
            return [
                'id' => 'Ідентифікатор адміністративної одиниці',
                'name' => 'Назва адміністративної одиниці',
                'type_id' => 'Тип адміністративної одиниці',
                'description' => 'Опис адміністративної одиниці',
                'parent_id' => 'Адміністративна одиниця вищого рівня',
                'fullName' => 'Назва адміністративної одиниці',
            ];
        }
        public function getType()
        {
            return $this->hasOne(CityType::className(), ['id' => 'type_id']);
        }
        public function getParent()
        {
            return $this->hasOne(City::className(), ['id' => 'parent_id']);
        }
        public function getCities()
        {
            return $this->hasMany(City::className(), ['parent_id' => 'id']);
        }
        public function getRelaxRegions()
        {
            return $this->hasMany(RelaxRegion::className(), ['city_id' => 'id']);
        }
        public function getFullName()
        {
            return $this->name . ', ' . mb_strtolower($this->type->name, 'UTF-8');
        }
    }
```

Рисунок 3.16 – Код для створення моделі

Для того, щоб записи були розміщені у вигляді дерева необхідно використати код з рисунку 3.17.

```
        <?php
        use yii\helpers\Html;
        use yii\grid\GridView;
        use yii\widgets\Pjax;
        use leandrogehlen\treegrid\TreeGrid;
        use yii\helpers\ArrayHelper;
        $this->title = 'Адміністративні одиниці';
        $this->params['breadcrumbs'][] = $this->title;
        ?>
        <div class="city-index">
            <h1><?= Html::encode($this->title) ?></h1>
            <?php Pjax::begin(); ?>
            <?php // echo $this->render('_search', ['model' => $searchModel]); ?>
            <p>
                <?= Html::a('Додати новий запис', ['create'], ['class' => 'btn btn-success'])
?>
            </p>
            <?= TreeGrid::widget([
                'dataProvider' => $dataProvider,
                'keyColumnName' => 'id',
                'parentColumnName' => 'parent_id',
                'columns' => [
                    'fullName',
                    ['class' => 'yii\grid\ActionColumn']
                ]
            ]); ?>
    <?php Pjax::end(); ?>
```

Рисунок 3.17 – Код для розміщення записів у вигляді дерева

В результаті отримаємо сторінку інформаційної системи з повним переліком адміністративних одиниць, розподілених по областях у формі дерева, як зображено на рисунку 3.18.

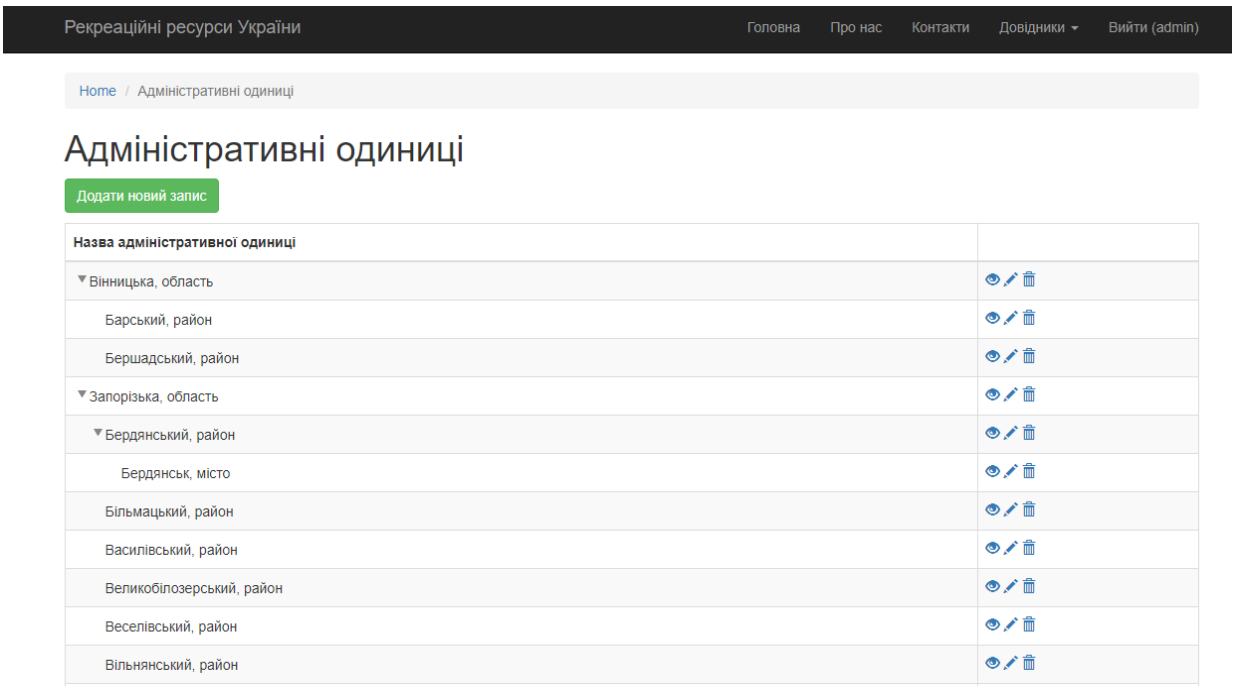

Рисунок 3.18 – Перегляд адміністративних одиниць

Для додавання нової адміністративної одиниці використаємо програмний код, наведений на рисунку 3.19.

```
$this->title = 'Форма додавання нової адміністративної одиниці';
$this->params['breadcrumbs'][] = ['label' => 'Адміністративні одиниці', 'url' => ['index']];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="city-create">
    <h1><?= Html::encode($this->title) ?></h1>
    <?= $this->render('_form', [
        'model' => $model,
    ]) ?>
</div>
```

Рисунок 3.19 – Код для створення форми додавання адміністративної одиниці

В результаті отримаємо форму додавання нової адміністративної одиниці, як зображено на рисунку 3.20.

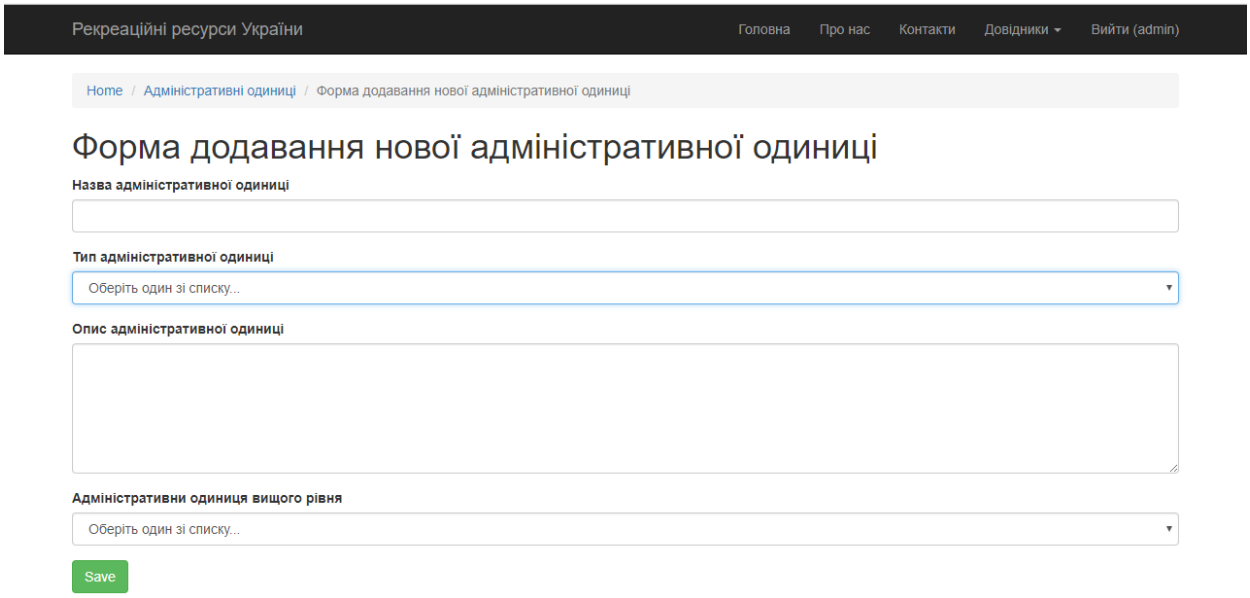

Рисунок 3.20 – Форма додавання адміністративної одиниці

Для перегляду однієї адміністративної одиниці використаємо код з рисунку 3.21.

```
        $this->title = $model->getFullName();
        $this->params['breadcrumbs'][] = ['label' => 'Адміністративні одиниці', 'url' =>
['index']];
        $this->params['breadcrumbs'][] = $this->title;
        ?>
        <div class="city-view">
            <h1><?= Html::encode($this->title) ?></h1>
            <p>
                <?= Html::a('Редагувати', ['update', 'id' => $model->id], ['class' => 'btn
btn-primary']) ?>
                <?= Html::a('Видалити', ['delete', 'id' => $model->id], [
                    'class' => 'btn btn-danger',
                    'data' => [
                        'confirm' => 'Ви підтверджуєте видалення запису?',
                        'method' => 'post',
                    ],
                ]) ?>
            </p>
            <?= DetailView::widget([
                'model' => $model,
                'attributes' => [
                    'id',
                    'name',
                    'type.name',
                    'description:ntext',
```

```
                [
                    'label' => 'Адміністративна одиниця вищого рівня',
                    'value' => ($model->parent) ? $model->parent->getFullName() : ''],
            ],
        ]) ?>
    </div>
```

Рисунок 3.21 – Код для перегляду однієї адміністративної одиниці

В результаті отримаємо сторінку, на якій можна редагувати, наприклад змінити назву адміністративної одиниці або її опис, або видалити адміністративні одиниці, приклад сторінки зображено на рисунку 3.22.

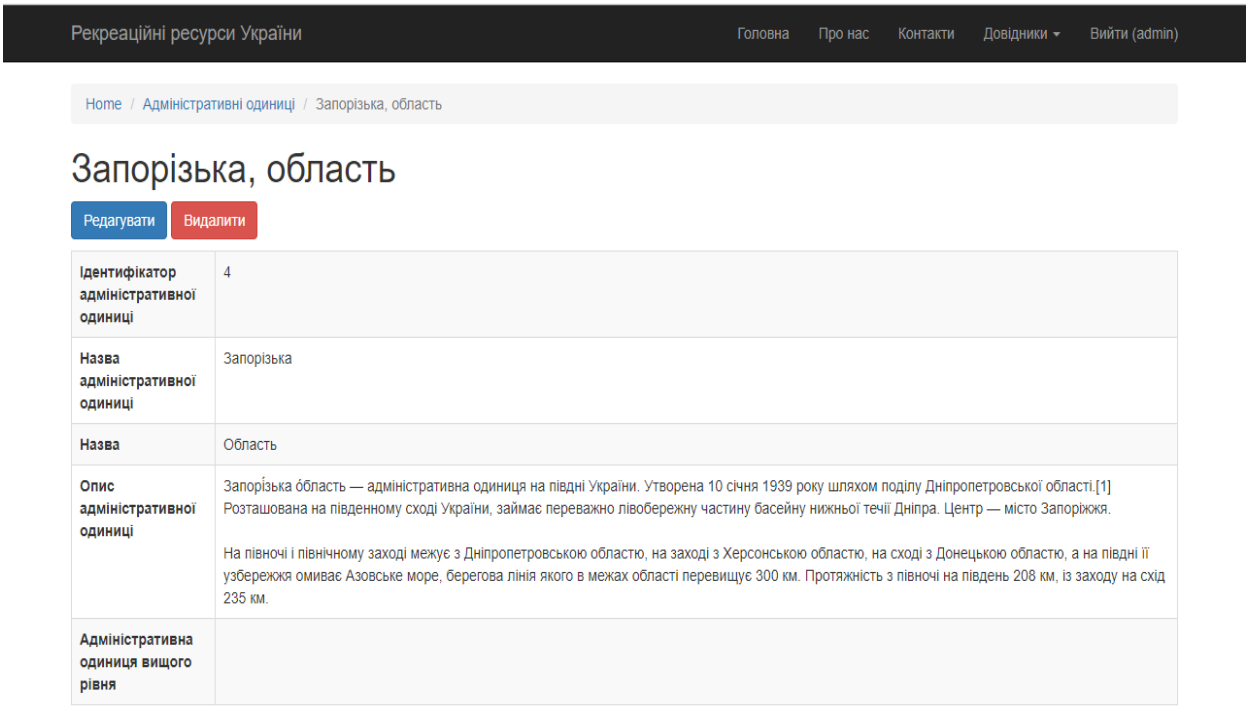

Рисунок 3.22 – Редагування адміністративної одиниці

3.6 Створення та перегляд сторінок сайту

Для перегляду всіх ресурсів використаємо програмний код з рисунку 3.23.

```
        $this->title = 'Рекреаційні ресурси';
        $this->params['breadcrumbs'][] = $this->title;
        ?>
        <div class="relax-region-index">
            <h1><?= Html::encode($this->title) ?></h1>
            <?php Pjax::begin(); ?>

            <?php // echo $this->render('_search', ['model' => $searchModel]); ?>
            <p>
                <?= Html::a('Створити новий рекреаційний ресурс', ['create'], ['class' =>
'btn btn-success']) ?>

            </p>
            <?= GridView::widget([
                'dataProvider' => $dataProvider,
                'filterModel' => $searchModel,
                'columns' => [
                    ['class' => 'yii\grid\SerialColumn'],
                    'id',
                    'name',
                    [
                        'attribute' => 'city',
                        'label' => 'Місце знаходження',
                        'value' => 'city.nameWithParent'
                    ],
                    'address',
                    ['class' => 'yii\grid\ActionColumn'],
                ],
            ]); ?>
            <?php Pjax::end(); ?>
        </div>
```

Рисунок 3.23 – Код для перегляду всіх ресурсів

В результаті отримаємо список всіх ресурсів, які мають свій ідентифікатор, назву, місце знаходження та точну адресу, як зображено на рисунку 3.24.

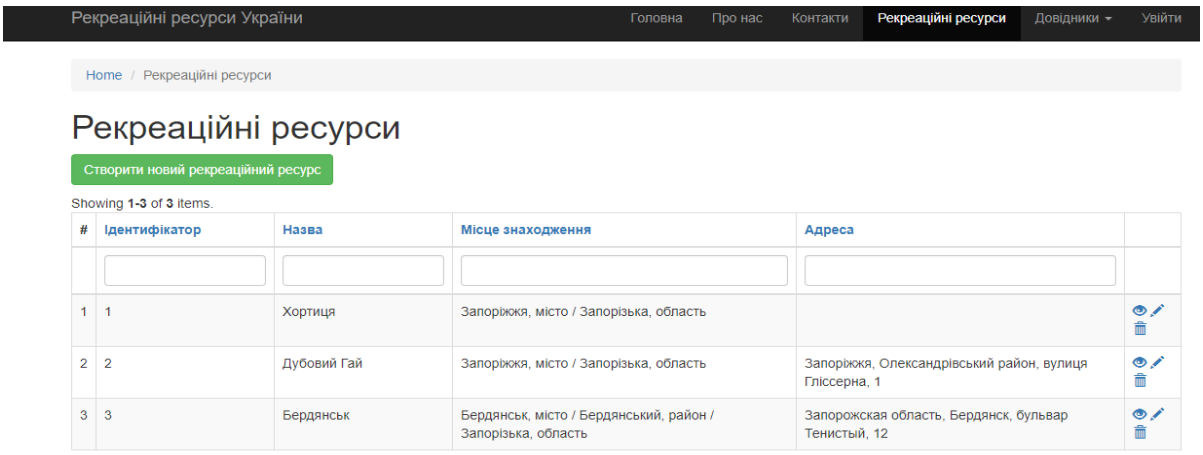

Рисунок 3.24 – Список рекреаційних ресурсів

Для перегляд одного ресурсу з точним виводом місцезнаходження на карту openstreetmap використаємо код, представлений на рисунку 3.25.

```
        $this->title = $model->name;
        $this->params['breadcrumbs'][] = ['label' => 'Рекреаційні ресурси', 'url' => ['index']];
        $this->params['breadcrumbs'][] = $this->title;
        ?>
        <div class="relax-region-view">
            <h1><?= Html::encode($this->title) ?></h1>
            <p>
                <?= Html::a('Редагувати', ['update', 'id' => $model->id], ['class' => 'btn btn-primary']) ?>
                <?= Html::a('Видалити', ['delete', 'id' => $model->id], [
                    'class' => 'btn btn-danger',
                    'data' => [
                        'confirm' => 'Ви підтверджуєте видалення запису?',
                        'method' => 'post',
                    ],
                ]) ?>
            </p>
            <?= DetailView::widget([
                'model' => $model,
                'attributes' => [
                    'id',
                    'name',
                    'city.nameWithParent',
                    'description:ntext',
                    'address',
                    'lat',
                    'lng',
```

```
                [
                    'label' => 'Типи рекреаційних ресурсів',
                    'value' => function ($model)
                    {
                        $res = '';
                        foreach($model->getTypes()->all() as $post)
                        {
                            $res .= $post->getAttribute('name') . '; ';
                        }
                        return $res;
                    }
                ]
            ],
        ]) ?>
        <div id="mapdiv" style="height: 400px"></div>
        <script src="http://www.openlayers.org/api/OpenLayers.js"></script>
        <script>
            map = new OpenLayers.Map("mapdiv");
            map.addLayer(new OpenLayers.Layer.OSM());
            var lonLat = new OpenLayers.LonLat(<?=$model->lng ?>, <?=$model->lat ?>)
                .transform(
                    new OpenLayers.Projection("EPSG:4326"), // transform from WGS 1984
                    map.getProjectionObject() // to Spherical Mercator Projection
                );
            var zoom=13;
            var markers = new OpenLayers.Layer.Markers( "Markers" );
            map.addLayer(markers);
            markers.addMarker(new OpenLayers.Marker(lonLat));
            map.setCenter (lonLat, zoom);
        </script>
    </div>
```

Рисунок 3.25 – Фрагмент коду

В результаті роботи отримаємо сторінку інформаційної системи з точним повним описом кожного туристичного рекреаційного ресурсу, місцем знаходження, адресою, географічними координатами, що відмічені на інтерактивній карті та відміткою, до якого типу даний ресурс відноситься. Приклад представлення ресурсу зображено на рисунку 3.26.

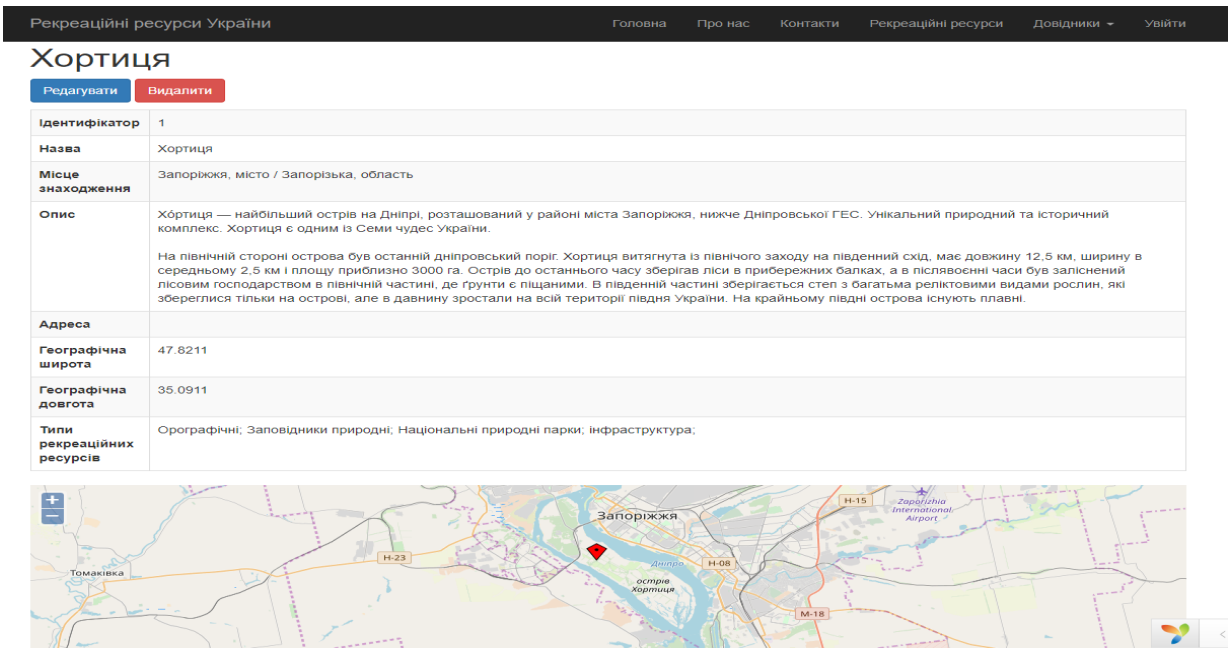

Рисунок 3.26 – Приклад представлення туристичного рекреаційного ресурсу

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є модель і автоматизована інформаційна система на базі PHP фреймворку Yii для ведення обліку та моніторингу туристичних рекреаційних зон.

Підсумком ретельного вивчення предметної області, аналізу вже існуючих інформаційних систем, виділення їх недоліків та переваг, стало концептуальне проєктування предметної області, побудова ER-діаграми з виділенням сутностей, їх типів та зв’язків та розробка єдиної уніфікованої моделі реляційної бази даних.

Створена база даних, до якої розроблено дружній інтерфейс користувача, котра є основою автоматизованої інформаційної системи.

Для розробки інформаційної системи був використаний фреймворк Yii2. Yii – це високопродуктивний компонентний PHP фреймворк, призначений для швидкої розробки сучасних веб-додатків. Завдяки його компонентній структурі і відмінною підтримкою кешування, фреймворк особливо підходить для розробки таких проєктів, як портали, форуми, CMS, магазини або RESTful-додатки. Для організації коду Yii використовує архітектурний патерн MVC (Model-View-Controller). MVC призначений для поділу бізнес-логіки і призначеного для 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 користувальницький інтерфейс (наприклад, текст, поля введення), а контролер забезпечує взаємодію між моделлю і представленням.

Розроблена інформаційна система може бути використана для туристичних підприємств, також можуть користуватися звичайні громадяни для ознайомлення з типами рекреаційних зон, їх характеристиками та розташування у межах нашої країни. Спеціалізацією розробленої моделі є надання розгорнутої інформації про різноманітні рекреаційні туристичні

об’єкти, їх конкретні адреси з географічними координатами для детального визначення на карті, характеристики та загальний огляд населених пуктів, які мають рекреаційні зони.

Інформаційна система є доступною для користувачів базового рівня підготовки та обізнаності у предметній області.

Отже, мета, поставлена на початку роботи, досягнута, всі завдання виконані.

Можливими напрямками розвитку може бути наповнення сайту більш детальною інформацією про туристичні рекреаційні зони не тільки по Україні, а й по всьому світу, вдосконалення інтерфейсу й функціональних можливостей.

ПЕРЕЛІК ПОСИЛАНЬ

1. Петров П.В. Введення в системи баз данних. Спб.: ИТМО, 2010. 128 с.

2. Райордан Р. Основи реляційних баз даних. Київ : Редакция, 2001. 384 с.

3. Конноллі Т. Бази даних: проектування, реалізація та супровід. Теорія та практика. Київ : Вильямс, 2000. 1120 с.

4. Дейт К.Дж. Введення в системи баз даних. Київ : Вільямс, 2006. 1328 с.

5. Проектування баз даниxx. Загальні рекомендіції. URL: http://www.mstu.edu.ru/education/materials/zelenkov/ch_5_1.html (дата звернення 16.10.2022).

6. Економіка. Управління. Інновації. Інноваційні технології в туризмі. URL: http://tourlib.net/statti_ukr/glebova2.htm (дата звернення 15.10.2022).

7. Розробка макету сайту. Принципи та рекомендації. URL: http://dist.org.ua/pluginfile.php/7033/ mod_resource/content/1/2.%20%D0%A0%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0%20%D0%BC%D0%B0%D0%BA%D0%B5%D1%82%D1%83%20%D1%81%D0%B0%D0%B9%D1%82%D1%83.pdf (дата звернення 03.11.2022).

8. Створюємо свій сайт. Навчально-методичний ресурс циклової комісії обліково-економічних дисциплін. URL:http://aek-oed.pl.ua/stvoryuyemo-sviy-sayt/ (дата звернення 13.11.2022).

9. Фреймворк Yii. Вікіпедія – вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Yii (дата звернення 03.12.2022).

10. Фреймворк Yii2 з Нуля до Профі. Yii2: Навчання URL: https://coursehunters.net/course/webformyself-yii2 (дата звернення 02.12.2022).

11. Створення Web-сторінок. Інструментарій для створення Web-

сторінок. URL: http://gymlit.in.ua/re_%D0%A1%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F_Web-%D1%81%D1%82%D0%BE%D1%80% D1%96% D0%BD%D0 %BE%D0%BA (дата звернення 16.10.2022).

12. Працюємо над створенням якісного web-сайту. Веб студія. URL: http://www.web-master.lviv.ua/blog/pratsyujemo-nad-stvorennyam-yakisnoho-web-sajtu.html (дата звернення 16.10.2022).

13. Що таке внутрішня структура сайту. Веб розробка. URL: http://korusno-znatu.in.ua/internet/web-rozrobka/vnytrinnya-stryktyra-sajty.php (дата звернення 19.10.2022).

14. Дизайн сайтів: інформаційні сайти. Веб студія. URL: http://webstudio2u.net/ua/design-site/707-dizain-informatsionnykh-saitov.html (дата звернення 21.11.2022).

15. Плескач В. Л. Інформаційні системи і технології на підприємствах. Засоби створення Web-сайтів. URL: http://westu-dents.com.ua/glavy/27290-zasobi-stvorennya-Web-saytv.html (дата звернення 29.11.2022).