ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка веб-системи «Природно-рекреаційні ресурси Одеської області»»

(тема кваліфікаційної роботи українською мовою)

«Development of the web system «Natural and recreational resources of Odesa region»»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач заочної форми навчання

спеціальності 122 Комп'ютерні науки

(код, назва спеціальності)

Освітня програма Комп'ютерні науки

(назва)

Хліненков Андрій Анатолійович

(прізвище, ім'я, по-батькові здобувача)

Керівник ст. викладач Вохменцева Т.Б. _____

(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент Перелигін Б.В.

(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .
№ 1 від 09 червня 2024 р.

Захищено на засіданні ЕК № 13 .
протокол № 9 від 19 червня 2024 р.

Оцінка добре / С / 75 .
(за національною шкалою/шкалою ECTS/ бали)

Завідувачка кафедри

_____ КАЗАКОВА Надія .
(підпис) (прізвище,ім'я)

Голова ЕК

_____ КОПИЧЕНКО Іван .
(підпис) (прізвище,ім'я)

Одеса 2024

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних;

ІС – інформаційна система;

КПК – кишеньковий персональний комп’ютер;

НФ – нормальна форма;

ООП – об’єктно-орієнтоване програмування;

ПЗ – програмне забезпечення;

СКБД – система керування базами даних;

ТЗ – технічне завдання;

API – Application Programming Interface (інтерфейс прикладного програмування)

CMS – Content management system (Система управління контентом)

CSS – Cascading Style Sheets (Каскадні таблиці стилів)

DTD – Document Type Declaration (Декларація типу документу)

ERD – Entity-Relationship Diagrams (Діаграми сутність-зв'язок)

HTML – HyperText Markup Language (Мова розмітки гіпертекстових документів)

MVC – Model-View-Controller (Модель-вид-контролер)

SQL – Structured Query Language (Мова структурованих запитів)

WYSIWYG – What You See Is What You Get (Що бачиш, то і отримаєш)

UCS – Universal Character Set (Універсальний Набір Символів)

UML – Unified Modeling Language (Універсальна мова моделювання)

URL – Uniform Resource Locator (Єдиний вказівник ресурсу)

ВСТУП

Сучасні інформаційні технології є потужним та функціональним засобом для створення безлічі веб застосунків, що дозволяють користувачам отримувати доступ до великих даних, обчислювальних потужностей веб сервісів та серверів.

В цьому контексті організація взаємодії між клієнтом та сервером, яка надає клієнту зручний інтерфейс для інформаційно-пошукових запитів є актуальним завданням. Такий інтерфейс може мати вигляд Web-сайту, що складається з сукупності сторінок, об'єднаних за змістом і розташованих на одному Web-сервері. Такий Web-сайт, організований за принципом безперебійного обслуговування різних інформаційних запитів в on-line режимі, може бути затребуваним в різних організаціях, підприємствах та установах.

Метою даного дипломної роботи є розроблення інформаційно-пошукової системи у вигляді вбудованого компонента Web-сайту для надання доступу до бази даних водних об’єктів Одеської області.

Для досягнення поставленої мети потрібно вирішити наступні завдання:

- аналіз предметної області;
- вибір та обґрунтування засобів розробки інформаційно-пошукової системи;
- обґрунтування вибору системи керування базами даних;
- обґрунтування вибору мов програмування;
- розробка дизайну сайта і верстка макету;
- розробка структури бази даних додатку;
- установка та налаштування компонент інформаційно-пошукової системи.

Структура кваліфікаційної роботи складається з вступу, чотирьох розділів, висновків, переліку посилань на 22 найменування, додатків. Повний обсяг проекту становить 57 сторінки, містить 15 рисунків і 7 таблиць.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Для того, щоб проектована інформаційно-пошукова система була ефективно реалізована, необхідно з самого початку розуміти що необхідно отримати в результаті. Тому спочатку проаналізуємо існуючі можливі рішення і сформулюємо основні вимоги до системі, потім приступимо до проектування.

Проект повинен являти собою простий механізм управління контентом, і містити як частину для користувача, так і адміністративну частину.

Частина системи для користувача – це те, що видно всім користувачам сайту. Являє собою форму пошуку за різними критеріями, видачу результатів пошуку і окремі інформаційні сторінки, що містять інформацію про водні об’єкти Одеської області за окремими районами.

На формі пошуку повинні бути присутніми елементи для введення і вибору за різними фільтрами:

- назва об’єкту;
- форма користування;
- місцезнаходження;
- тип об’єкту;
- басейн;
- площа.

Форма пошуку повинна мати списки значень, чекбокси і різні фільтри. При цьому необхідно щоб завдання критеріїв було інтуїтивно простим і дозволяло здійснення гнучкого пошуку. Форма не повинна бути перевантажена зайвою інформацією і дизайн форми гармонійно поєднувався з загальним дизайном сайту.

Результати пошуку необхідно представляти користувачеві з відповідною додатковою інформацією, такою як характеристики водного об’єкту, фотознімки, карта його розташування.

Адміністративна частина інформаційно-пошукової системи надає доступ для управління всією інформацією в зручному для користувача вигляді. Вона повинна надавати зручні механізми управління контентом і містити розділи для управління:

- списком районів;
- списком водних об’єктів за районами;
- форм додавання характеристик водних об’єктів та їх координат розташування на карті.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Проектування – це дуже важливий етап при розробці проекту. Від того, як добре буде спроектована база даних і архітектура інформаційної системи залежить її якісна робота. Не вірно прийняте рішення при проектуванні надалі може спричинити за собою серйозні проблеми при внесенні змін в систему. Тому етапу проектування необхідно приділити належну увагу.

Інформаційна-пошукова система, що розроблюється, повинна містити вбудований плагін, який являє собою форму пошуку за різними критеріями, видачу результатів пошуку і окремі інформаційні сторінки, що містять інформацію про водні об’єкти Одеської області за окремими районами.

2.1 Система управління контентом WordPress

При розробці сайтів можливі три варіанти:

1) Створення сайту з використанням «конструктора сайтів». Використання такого підходу дозволяє створити сайт без особливої підготовки і знання програмування. Необхідно підібрати підходящу платформу, пройти реєстрацію і можна приступити до наповнення контентом. Такі «конструктори», як правило, WYSIWYG містять редактор, який дозволяє писати і офор-

мляти інформацію як в звичному текстовому редакторі. Такі сайти мають стандартні шаблони і обмежену функціональність.

2) Створення сайту «з нуля». Даний шлях дозволяє мати повний контроль над розробкою та її результатами. Розробник має повну свободу вибору в прийнятті рішень. Грамотне використання мінімально необхідної кількості зовнішніх бібліотек може дати хороший приріст продуктивності. Водночас, може істотно збільшитися час розробки, тому що необхідно реалізувати функціонал, який може існувати в готових системах.

3) Створення сайту за допомогою систем управління контентом (вмістом). Використання систем управління контентом (CMS) дозволяє використовувати розробнику існуючий функціонал інформаційної системи, яка забезпечує спільний процес створення, редагування і управління контентом. Системи управління контентом дозволяють з легкістю розгорнути сайт з базовим функціоналом, що дозволяє не витрачати час на реалізацію «стандартних» функцій. Багато з систем управління контентом мають свій API, за допомогою якого можна розширити функції системи. Недолік даного підходу полягає в тому, що розробники можуть використовувати продиктовані самою системою стандарти і в разі виявлення вразливостей є ризик злому сайту, але, багато систем мають широке співтовариство користувачів, які швидко виправляють помічені уразливості.

Так як розробка оптимальної системи управління контентом займає дуже багато часу, було прийнято рішення використовувати CMS WordPress. Дана система дозволяє з легкістю розширювати свої стандартні можливості за допомогою плагінів і відмінно підходить для вирішення поставленого завдання.

Плагін WordPress – це програма або набір функцій, написаних на PHP, що додають певний набір можливостей або сервісів до блогу на WordPress, які легко об'єднуються з системою управління та WordPress функціоналом за допомогою плагінів Інтерфейсу прикладної програми (API) [1] Основна ідея WordPress – залишити ядро цілісним і незмінним, і в той же час дати розроб-

никам можливість змінювати його поведінку за допомогою спеціальних скріптов-плагінів.

2.2 Проектування бази даних

База даних є найважливішим елементом будь-якої інформаційної системи. Обрана система управління контентом дозволяє плагінам виробляти всілякі операції з базою даних. Для цього в WordPress є свій клас wpdb. Звертатися до методів даного класу можна через глобальну змінну $ wpdb. Сервером бази даних в CMS обрана система керування базами даних – MySQL [2].

Почнемо з етапу розробки концептуальної бази даних інформаційно-пошукової системи по водним об’єктам Одеської області. Цей етап включає в себе аналіз об'єктів реального світу, які необхідно змоделювати в базі даних і складається з етапів:

– Ідентифікація функціональної діяльності предметної області. В даному випадку мова йде про діяльність організації і в якості функціональної діяльності можна розглядати ведення реєстру водних об’єктів;

– Ідентифікацію об'єктів, які здійснюють цю функціональну діяльність і ідентифікувати всі сутності і взаємозв'язку між ними. Процес "ведення реєстру" ідентифікує такі сутності: ВОДНИЙ ОБ’ЄКТ, РАЙОН, БАСЕЙН, ТИП, ВЛАСНІСТЬ, ЗОБРАЖЕННЯ;

– Ідентифікацію характеристик цих сутностей;

– Ідентифікацію взаємозв'язків між сутностями.

Перерахуємо сутності:

– Сутність РАЙОН може включати такі характеристики як ідентифікатор, назва району;

– Сутність ТИП може включати такі характеристики як ідентифікатор, назва типу;

– Сутність БАСЕЙН може включати такі характеристики як ідентифікатор, назва басейну;

– Сутність ВЛАСНІСТЬ може включати такі характеристики як ідентифікатор, назва типу власності;

– Сутність ЗОБРАЖЕННЯ може включати такі характеристики як ідентифікатор, посилання на зображення об’єкту, заголовок для зображення;

– Сутність ВОДНИЙ ОБ’ЄКТ може включати такі характеристики як ідентифікатор, площу, код району, код басейну, код типу, код типу власності, код зображення, місцезнаходження, ім'я файлу, що містить графічне зображення.

Наступний етап проектування БД полягає у встановленні відповідності між сутностями і характеристиками предметної області і відносинами і атрибутами в нотації обраної СКБД. Оскільки кожна сутність реального світу володіє якимись характеристиками, в сукупності утворюють повну картину її прояви, можна поставити їм у відповідність набір відносин (таблиць) і їх атрибутів (полів).

Перерахувавши всі відносини і їх атрибути, вже на цьому етапі можна почати усувати зайві позиції. Кожен атрибут повинен з'являтися тільки один раз і треба вирішити, яке відношення буде власником якого набору атрибутів.

На наступному етапі визначаються атрибути, які унікальним чином ідентифікують кожен об'єкт. Це необхідно для того, щоб система могла отримати будь-який одиничний рядок таблиці.

Наступний етап передбачає вироблення правил, які будуть встановлювати і підтримувати цілісність даних. Будучи певними, такі правила в клієнт-серверних СКБД підтримуються автоматично – сервером баз даних; в локальних же СКБД їх підтримку доводиться покладати на користувальницький додаток.

Далі встановлюються зв'язки між об'єктами (таблицями і стовпцями) і виробляється дуже важлива операція для виключення надмірності даних – нормалізація таблиць.

Кожен з різних типів зв'язків повинен бути змодельований в базі даних. Існує кілька типів зв'язків: – зв'язок "один-до-одного", зв'язок "один-до-

багатьох", зв'язок "багато-до-багатьох". Зв'язок "один-до-одного" представляє собою найпростіший вид зв'язку даних, коли первинний ключ таблиці є в той же час зовнішнім ключем, що посилаються на первинний ключ іншої таблиці. Зв'язок "один- до-багатьох" в більшості випадків відображає реальну взаємозв'язок сутностей в предметної області. Вона реалізується вже описаною парою "зовнішній ключ-первинний ключ", тобто коли визначено зовнішній ключ, що посилається на первинний ключ іншої таблиці. Зв'язок "багато-до-багатьох" в явному вигляді в реляційних базах даних не підтримується. Однак є ряд способів непрямої реалізації такого зв'язку, які з успіхом відшкодовують її відсутність. Один з найбільш поширених способів полягає у введенні додаткової таблиці, рядки якої складаються із зовнішніх ключів, що посилаються на первинні ключі двох таблиць.

У проектованої БД встановлено зв'язок один-до-багатьох між сутностями РАЙОН, БАСЕЙН, ТИП, ВЛАСНІСТЬ, ЗОБРАЖЕННЯ і сутністю ВОДНИЙ ОБ’ЄКТ.

Після визначення таблиць, полів, індексів і зв'язків між таблицями слід нормалізувати проектовану базу даних. Важливість нормалізації полягає в тому, що вона дозволяє розбити великі відносини, як правило, містять велику надмірність інформації, на більш дрібні логічні одиниці, що групують тільки дані, об'єднані "по природі". Таким чином, ідея нормалізації полягає в наступному. Кожна таблиця в реляційній базі даних задовольняє умові, відповідно до якого в позиції на перетині кожного рядка і стовпчика таблиці завжди знаходиться єдине значення, і ніколи не може бути безлічі таких значень.

Процес нормалізації включає: усунення повторюваних груп (приведення до 1НФ); видалення частково залежних атрибутів (приведення до 2НФ); видалення транзитивно залежних атрибутів (приведення до 3НФ).

ER- діаграма розробленої бази даних наведена на рис.2.1. Вигляд проектованої бази даних після нормалізації демонструють табл.2.1 – табл.2.6.

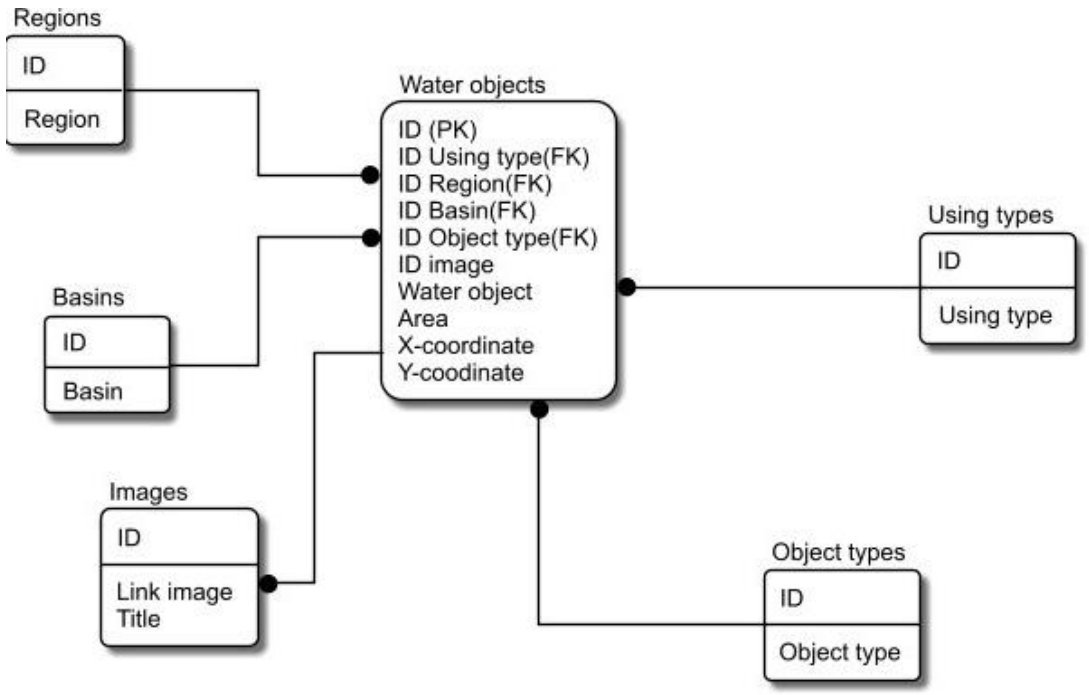

Рисунок 2.1 – ER- діаграма бази даних плагіну

Таблиця 2.1 – Структура таблиці Regions

Ім’я поля	Тип
ID	INTEGER
REGION	CHAR(30)

Таблиця 2.2 – Структура таблиці Basins

Ім’я поля	Тип
ID	INTEGER
BASIN	CHAR(30)

Таблиця 2.3 – Структура таблиці Object_types

Ім’я поля	Тип
ID	INTEGER
OBJECT_TYPE	CHAR(30)

Таблиця 2.4 – Структура таблиці Using_types

Ім’я поля	Тип
ID	INTEGER
USING_TYPE	CHAR(30)

Таблиця 2.5 – Структура таблиці Images

Ім’я поля	Тип
ID	INTEGER
IMAGE	CHAR(255)
TITLE	CHAR(30)

Таблиця 2.6 – Структура таблиці Water_objects

Ім’я поля	Тип	Опис
ID	INTEGER	унікальний індекс
WATER_OBJECT	CHAR(30)	назва водного об’єкту
ID REGION	INTEGER	район
ID BASIN	INTEGER	басейн
ID OBJECT_TYPE	INTEGER	тип водного об’єкту
ID USING_TYPE	INTEGER	тип користування водним об’єктом
ID IMAGE	INTEGER	зображення водного об’єкта
AREA	FLOAT	площа водного об’єкту
Coordinate X	FLOAT	координата х місця розташування
Coordinate Y	FLOAT	координата у місця розташування

Таблиці Regions і Basins містять назву всіх районів і басейнів річок Одеської області відповідно.

Таблиця Object_types містить назву типів водних об’єктів (річка, ставок, джерело і т.п.). Таблиця Using_types містить назву типів використання водних об’єктів (у оренді, у власності, колективне використання тощо). Таб-

лиця Images містить посилання на зображення водних об’єктів. Таблиця Water_objects містить всю необхідну інформацію про водний об’єкт.

Всі поля ID є первинними ключами для своїх таблиць. Поле ID REGION таблиці Water_objects є зовнішнім ключем до таблиці Regions (додаток Б). Поле ID BASIN таблиці Water_objects є зовнішнім ключем до таблиці Basins. Поле ID OBJECT_TYPE таблиці Water_objects є зовнішнім ключем до таблиці Object_types. Поле ID USING_TYPE таблиці Water_objects є зовнішнім ключем до таблиці Using_types.

2.3 Проектування архітектури плагіна

Система управління контентом WordPress не накладає на розробку плагінів ніяких обмежень. В якості основи плагіна був обраний шаблон проектування Model-View-Controller (MVC) [3] (рис. 2.2).

Ідеї MVC сформулював Трюгве Реєнскауг (Trygve Reenskaug) під час роботи у Xerox PARC наприкінці 70-х років. Завдання, яке Реенскауг вирішував разом із групою дуже сильних розробників, полягала у тому, щоб спростити взаємодію пересічного користувача з комп'ютером. Необхідно було створити засоби, які з одного боку були б гранично простими та зрозумілими, а з іншого – давали б можливість керувати комп'ютером та складними програмами. Реєнскауг працював у команді, яка займалася розробкою портативного комп'ютера "для дітей різного віку" – Dynabook, а також мови SmallTalk під керівництвом Алана Кея. Саме тоді й там закладалися поняття доброзичливого інтерфейсу. Робота Реєнськауга разом із командою багато в чому вплинула на розвиток сфери IT.

Проект, над яким працював Реннскауг, вівся протягом 10 років. А перша публікація про MVC від його творців побачила світ ще через 10 років.

Оскільки інформації про MVC з першоджерела довго не було, а також з інших причин, з'явилася велика кількість різних трактувань цього поняття.

В результаті багато хто вважає MVC схемою або патерном проектування. Рідше MVC називають складеним патерном або комбінацією кількох патернів, що працюють спільно для реалізації складних програм. Але насправді, як було сказано раніше, MVC — це насамперед набір архітектурних ідей/принципів/підходів, які можна реалізувати різними способами з використанням різних шаблонів.

Model-View-Controller (MVC, «Модель-вид-контролер») – схема використання декількох шаблонів проектування, за допомогою яких модель додатку, призначений для користувача інтерфейс і взаємодія з користувачем розділені на три окремих компонента таким чином, щоб модифікація одного з компонентів чинила мінімальний вплив на інші.

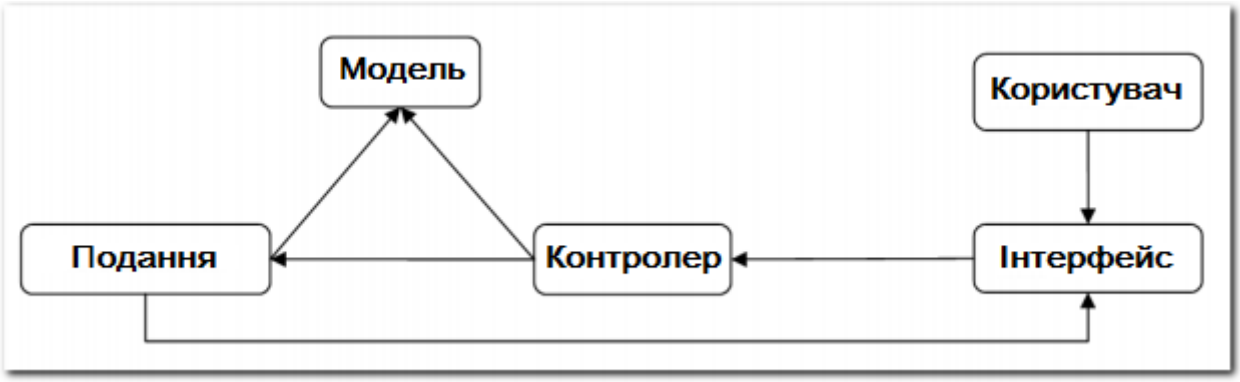

Рисунок 2.2 – Шаблон проектування MVC

Користувач, взаємодіючи з інтерфейсом, управляє контролером, який перехоплює дії користувача. Далі контролер повідомляє модель про дії користувача, таким чином змінюючи стан моделі. Контролер також повідомляє вид. Вид, використовуючи поточний стан моделі, будує призначений для користувача інтерфейс.

Основою патерну є відділення моделі даних програми, її логіки і представлення даних один від одного. Таким чином, дотримуючись правилом «розділяй і володарюй», вдається будувати стрункe програмне забезпечення,

в якому, по-перше, модель не залежить від уявлення і логіки, а по-друге, призначений для користувача інтерфейс надійно відділений від керуючої логіки. 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 призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

Основна мета дотримання принципів MVC - відокремити реалізацію бізнес-логіки програми (моделі) від її візуалізації (виду). Такий поділ підвищить можливість повторного використання коду. Користь застосування MVC найбільш наочна у випадках, коли користувачеві потрібно надавати ті самі дані в різних формах. Наприклад, як таблиці, графіка чи діаграми (використовуючи різні види). При цьому, не торкаючись реалізації видів, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних). Якщо дотримуватися принципів MVC, можна спростити написання програм, підвищити читання коду, полегшити розширення та підтримку системи в майбутньому.

3 ОБГРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ

3.1 Мова розмітки гіпертекстових документів HTML

HTML (HyperText Markup Language – мова розмітки гіпертекстових документів) – стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML) [4]. Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML є похідною мовою від SGML, успадкувавши від неї визначення типу документу та ідеологію структурної розмітки тексту. HTML разом із каскадними таблицями стилів та вбудованими скриптами – це три основні технології побудови веб-сторінок. HTML впроваджує засоби для:

- створення структурованого документу шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації із Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео та інших об'єктів до тексту.

Документ HTML. Для поліпшення взаємодії, SGML вимагає аби кожна похідна мова (HTML у тому числі) визначала свою кодову таблицю для кожного документа, яка складається з репертуара (перелік різноманітних символів) та позиції символа (перелік цифрових посилань на символи з репертуара). Кожен документ HTML – це послідовність символів з репертуара.

HTML використовує найповнішу кодову таблицю UCS (Universal Character Set – Універсальний Набір Символів).

Проте, однієї кодової таблиці недостатньо для того, щоб браузери могли правильно відтворювати документи HTML. Для цього браузерам потрібно «знати» специфічну кодову таблицю документа, яку автор має зазначати зав-

жди в елементі meta із параметром charset. За замовчуванням використовується кодова таблиця ISO-8859-1, відома також як Latin-1.

Розмітка в HTML складається з чотирьох основних компонентів: елементів (та їхніх атрибутів), базових типів даних, символьних мнемонік та декларації типу документа.

Документ HTML 4.01 складається з трьох частин [5]:

1) Декларація типу документу (Document type declaration, Doctype), на початку документа, в якій визначається тип документа (DTD).

2) Шапка документу (знаходиться в межах елементу head), в якій записані загальні технічні відомості або додаткова інформація про документ, яка не відтворюється безпосередньо в браузері.

3) Тіло документу (може знаходитися в елементах body або frameset), в якому міститься основна інформація документа.

Елементи являють собою базові компоненти розмітки HTML. Кожен елемент має дві основні властивості: атрибути та зміст (контент). Існують певні настанови щодо кожного атрибута та контента елемента, які треба виконувати задля того, щоб HTML-документ був визнаний валідним.

У елемента є початковий тег, який має вигляд <element-name>, та кінцевий тег, який має вигляд </element-name>. Атрибути елемента записуються в початковому тегу одразу після назви елемента, контент елемента записується між його двома тегами. Наприклад: <element-name element-attribute=”attribute-value”>контент елемента</element-name>.

Деякі елементи, наприклад br, не містять контенту, тож і не мають кінцевого тега. Елемент може не мати початкового та кінцевого тега (наприклад, елемент head), проте він завжди буде представлений в документі.

Більшість з атрибутів елементу являє собою пару «назва-значення», розділених між собою знаком рівняння, та записаних у початковому тегу одразу після назви елемента. Значення атрибуту може бути окреслено лапками (подвійними або одиничними), також, якщо значення атрибуту складається з певних символів, його можна не виділяти лапками зліва. Проте, не виділення

значення атрибутів в лапки вважається небезпечним кодом. На відміну від атрибутів виду «назва-значення», є певні атрибути, що впливають на елемент, назва яких лише з'явилась в початковому тегу (наприклад, атрибут ismap елементу img).

Оскільки HTML є похідною мовою від SGML, усі типи даних HTML ґрунтуються на базових типах даних SGML (наприклад, PCDATA, CDATA, NAME, ID, NUMBER).

Кожен елемент має дві властивості – атрибути і вміст, які мають певні значення. Всі можливі значення цих двох властивостей прописуються відповідно до визначених у DTD типів даних [6].

Існують такі випадки, коли в документі потрібно використати якийсь символ, якого немає в обраній для документу кодовій таблиці. Для таких випадків можливо замінити символ на еквівалентне йому SGML-посилання на символ (мнемоніку).

Для перегляду HTML-розмітки документа можна використовувати будь-який текстовий редактор. Для перегляду документу, відтвореного за правилами HTML-розмітки, використовується браузер.

3.2 Формальна мова опису зовнішнього вигляду документу CSS

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг – можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS) [7].

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздруковано-

му вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.)

Один і той самий HTML або XML документ може бути відображений по-різному залежно від використаного CSS.

Стандарт CSS визначає порядок та діапазон застосування стилів, те, в якій послідовності і для яких елементів застосовуються стилі. Таким чином, використовується принцип каскадності, коли для елементів вказується лише та інформація про стилі, що змінилася або не визначена загальнішими стилями. Переваги [8]:

- інформація про стиль для усього сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок;
- різна інформація про стилі для різних типів користувачів: наприклад, великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв;
- сторінки зменшуються в об'ємі та стають більш структурованими, оскільки інформація про стилі відділена від тексту та має певні правила застосування і сторінка побудована з урахуванням їх;
- прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі. Досягається за рахунок того, що сучасні браузери здатні кешувати (запам'ятовувати) інформацію про стилі і використовувати для всіх сторінок, а не завантажувати для кожної.

CSS має порівняно простий синтаксис і використовує небагато англійських слів для найменування різних складових стилю.

Стилі складаються зі списку правил. Кожне правило має один або більше селектор (Selector) та блок визначення (Declaration block). Блок визначення складається із оточеного фігурними дужками списку властивостей [9].

3.3 Прототипно-орієнтована сценарна мова програмування JavaScript

JavaScript (JS) – динамічна, об’єктно-орієнтована мова програмування [10]. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Мова JavaScript також використовується для програмування на стороні серверу (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF-документів тощо.

JavaScript класифікують як прототипну (підмножина об’єктно-орієнтованої), мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам’яттю, наслідування, функції як об’єкти першого класу.

JavaScript є однією з найпопулярніших мов програмування в інтернеті. Але спочатку багато професіональних програмістів скептично ставилися до мови, цільова аудиторія якої складалася з програмістів-любителів. Поява AJAX змінила ситуацію та повернула увагу професійної спільноти до мови. В результаті, були розроблені та покращені багато практик використання JavaScript (зокрема, тестування та налагодження), створені бібліотеки та фреймворки, поширилося використання JavaScript поза браузером.

JavaScript має низку властивостей об’єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об’єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має ряд властивостей, притаманних функціональним мовам,– функції як об’єкти першого класу, об’єкти як спис-

ки, каррінг (currying), анонімні функції, замикання (closures) – що додає мові додаткову гнучкість.

JavaScript має С-подібний синтаксис, але в порівнянні з мовою Сі має такі корінні відмінності [11]:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів;
- функції як об'єкти першого классу;
- обробка винятків;
- автоматичне приведення типів;
- автоматичнее прибирання сміття;
- анонімні функції.

JavaScript містить декілька вбудованих об’єктів: Global, Object, Error, Function, Array, String, Boolean, Number, Math, Date, RegExp. Крім того, JavaScript містить набір вбудованих операцій, які, строго кажучи, не обов’язково є функціями або методами, а також набір вбудованих операторів, що управляють логікою виконання програм. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від С), але спрощений порівняно з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, приміром, декларація змінної не містить її типу, властивості також не мають типів, а декларація функції може стояти в тексті програми після неї [12].

При використанні в рамках технології DHTML JavaScript код включається в HTML-код сторінки і виконується інтерпретатором, вбудованим в браузер. Код JavaScript вставляється в теги <script></script> з обов’язковим по специфікації HTML 4.01 атрибутом type=»text/javascript», хоча в більшості браузерів мова сценаріїв за умовчанням саме JavaScript.

3.4 Бібліотека jQuery

jQuery – популярна JavaScript-бібліотека з відкритим сирцевим кодом [13]. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом (John Resig). Згідно з дослідженнями організації W3Techs, Jquery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

jQuery є вільним програмним забезпеченням під ліцензією MIT (до вересня 2012 було подвійне ліцензування під MIT та GNU General Public License другої версії) [14].

Основне завдання jQuery –це надавати розробнику легкий та гнучкий інструментарій кросбраузерної адресації DOM об’єктів за допомогою CSS та Xpath селекторів. Також даний фреймворк надає інтерфейси для Ajax-застосунків, обробників подій і простої анімації.

Принцип роботи jQuery полягає в використанні класу (функції), який при звертанні до нього повертає сам себе. Таким чином, це дозволяє будувати послідовний ланцюг методів.

3.5 Скриптова мова програмування PHP

PHP (Hypertext Preprocessor, гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools –мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby) [15]. PHP підтримується переважною більшістю хостинг-провайдерів. PHP– проект відкритого програмного забезпечення.

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпе-

ки, але погіршує інтерактивність сторінок. Але ніщо не забороняє використовувати PHP для генерування і JavaScript-кодів які виконуються вже на стороні клієнта.

PHP – мова, код якої можна вбудовувати безпосередньо в html-код сторінок, які, у свою чергу, будуть коректно оброблені PHP-інтерпретатором.

В PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase. Через стандарт відкритого інтерфейсу зв'язку з базами даних (Open Database Connectivity Standard – ODBC) можна підключатися до всіх баз даних, до яких існує драйвер [16].

PHP 5 володіє прекрасним потенціалом реалізації об'єктного програмування. Окрім цього, PHP збагатився рядом цінних розширень для роботи з XML, різними джерелами даних, генерації графіки і інше. Серед інших украй корисних доповнень в PHP 5 слід зазначити нову схему обробки виключень. Конструкція try/catch/throw дозволяє весь код обробки помилок локалізувати в одному місці сценарію.

У PHP 5 також включені два нових модулі для роботи з протоколами – SimpleXML і SOAP. SimpleXML дозволяє значно спростити роботу з XML-даними, представляючи вміст XML-документа у вигляді PHP-об'єкта. Розширення SOAP дозволяє будувати на PHP сценарії, що обмінюються інформацією з іншими застосуваннями за допомогою XML-повідомлень поверх існуючих веб-протоколів, наприклад, HTTP. Новий модуль PHP 5 MySQLi (MySQL Improved) призначений для роботи з MySQL-сервером версій 4.1.2 і вище, реалізовуючи не тільки процедурний, але і об'єктно-орієнтований інтерфейс до MySQL [17]–[18]. Додаткові можливості цього модуля включають – SSL,контроль транзакцій, підтримка реплікації та ін. Очевидно, що, на цьому історія PHP не закінчується. Слід очікувати наступних версій мови із розширеними можливостями

3.6 Мова структурованих запитів SQL

SQL (Structured query language – мова структурованих запитів) – декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних і її модифікації, системи контролю за доступом до бази даних. Сам по собі SQL не є ні системою керування базами даних, ні окремим програмним продуктом.

SQL – це діалогова мова програмування для здійснення запиту і внесення змін до бази даних, а також управління базами даних. Багато баз даних підтримує SQL з розширеннями до стандартної мови. Ядро SQL формує командна мова, яка дозволяє здійснювати пошук, вставку, оновлення, і вилучення даних, використовуючи систему управління і адміністративні функції. SQL також включає CLI (Call Level Interface) для доступу і управління базами даних дистанційно.

SQL складається з наступних елементів мови [19]:

- положення, які є складовими компонентами заяв і запитів;
- предикати для тризначної логіки (3VL) (так / ні / невідомо) або логічних значень, які використовуються для обмеження наслідків заяв та запитів або змінити хід виконання програми;
- запити для видачі скалярних величин або таблиць, що складаються із стовпців і рядків даних;
- запити вилучення даних на основі певних критеріїв;
- заяви впливу на схеми і даних, зв'язків;
- заяви управлінням транзакціями, ходом виконання програми, завдань та діагностики.

Розглянемо переваги SQL:

1) Незалежність від конкретної СКБД. Незважаючи на наявність діалектів і відмінностей в синтаксисі, в більшості своїй тексти SQL-запитів, що містять, DDL і DML, можуть бути досить легко перенесені з однієї СКБД в

іншу. Існують системи, розробники яких спочатку закладалися на застосування щонайменше кількох СКБД (наприклад: система електронного документообігу Documentum може працювати як з Oracle, так і з Microsoft SQL Server та IBM DB2). Природно, що при застосуванні деяких специфічних для реалізації можливостей такої переносимості добитися вже дуже важко.

2) Наявність стандартів. Наявність стандартів і набору тестів для виявлення сумісності і відповідності конкретній реалізації SQL загальноприйнятому стандарту тільки сприяє «стабілізації» мови. Правда, варто звернути увагу, що сам по собі стандарт місцями занадто формалізований і роздутий в розмірах, наприклад, Core-частину стандарту SQL:2003 включає понад 1300 сторінок тексту.

3) Декларативність. За допомогою SQL програміст описує тільки те, які дані потрібно витягнути або модифікувати. Те, яким чином це зробити, вирішує СКБД безпосередньо при обробці SQL-запиту. Проте не варто думати, що це повністю універсальний принцип – програміст описує набір даних для вибірки або модифікації, проте йому при цьому корисно уявляти, як СКБД розбиратиме текст його запиту. Особливо критичні такі моменти стають при роботі з великими базами даних і зі складними запитами – чим складніше сконструйований запит, тим більше він допускає варіантів написання, різних за швидкістю виконання, але тих самих за набором даних.

До недоліків SQL слід віднести наступне:

1) Невідповідність реляційної моделі даних. Творці реляційної моделі даних Едгар Кодд, Крістофер Дейт та їхні прихильники указують на те, що SQL не є істинно реляційною мовою. Зокрема вони указують на такі проблеми SQL:

- рядки, що повторюються;
- невизначені значення (null);
- явна вказівка порядку стовпчиків зліва направо;
- стовпці без імені та імена стовпчиків, що дублюються;
- відсутність підтримки властивості «=»;

- використання покажчиків;
- висока надлишковість.

3.7 Система керування базами даних MySQL

MySQL – вільна система керування реляційними базами даних. MySQL була розроблена компанією «TcX» для підвищення швидкодії обробки великих баз даних [20]. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних.

MySQL має подвійне ліцензування. MySQL може розповсюджуватися відповідно до умов ліцензії GPL. Але за умовами GPL, якщо якась програма використовує бібліотеки MySQL, то вона теж повинна розповсюджуватися за ліцензією GPL. Проте це може розходитися з планами розробників, які не бажають відкривати сирцеві тексти своїх програм. Для таких випадків передбачена комерційна ліцензія компанії Oracle, яка також забезпечує якісну сервісну підтримку. В разі використання та розповсюдження програмного забезпечення з іншими вільними ліцензіями, такими як BSD, Apache, MIT та інші, MySQL дозволяє використання бібліотек MySQL за ліцензією GPL [21].

Гілка MySQL 5.5 містить ряд значних поліпшень, пов'язаних з підвищенням масштабованості та швидкодії, серед яких:

- використання за замовчуванням рушія InnoDB.
- підтримка напівсинхронного (semi-synchronous) механізму реплікації, заснованого на патчах до InnoDB від компанії Google.
- поліпшення функцій з партіціювання даних. Розширений синтаксис для розбиття великих таблиць на кілька частин, розміщених в різних файлових системах (partitioning). Додані операції RANGE, LIST і метод оптимізації «partition pruning».

- новий механізм оптимізації вкладених запитів та операцій JOIN.
- перероблена система внутрішніх блокувань.
- інтегровані патчі Google з оптимізацією роботи InnoDB на CPU з великою кількістю ядер.

MySQL – компактний багатонитевий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання. MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатонитковості, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

3.8 Підхід до створення web-застосувань AJAX

AJAX (Asynchronous JavaScript And XML) – підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані [22]. AJAX – один з компонентів концепції DHTML.

Про AJAX заговорили після появи в лютому 2005-го року статті Джесі Джеймса Гарретта (Jesse James Garrett) «Новий підхід до веб-застосунків». AJAX – не самостійна технологія. Це ідея. На рис.2.1 наведена модель класичного додатку для мережі (зліва) в прямому порівнюванні із застосуванням Ajax (зправа).

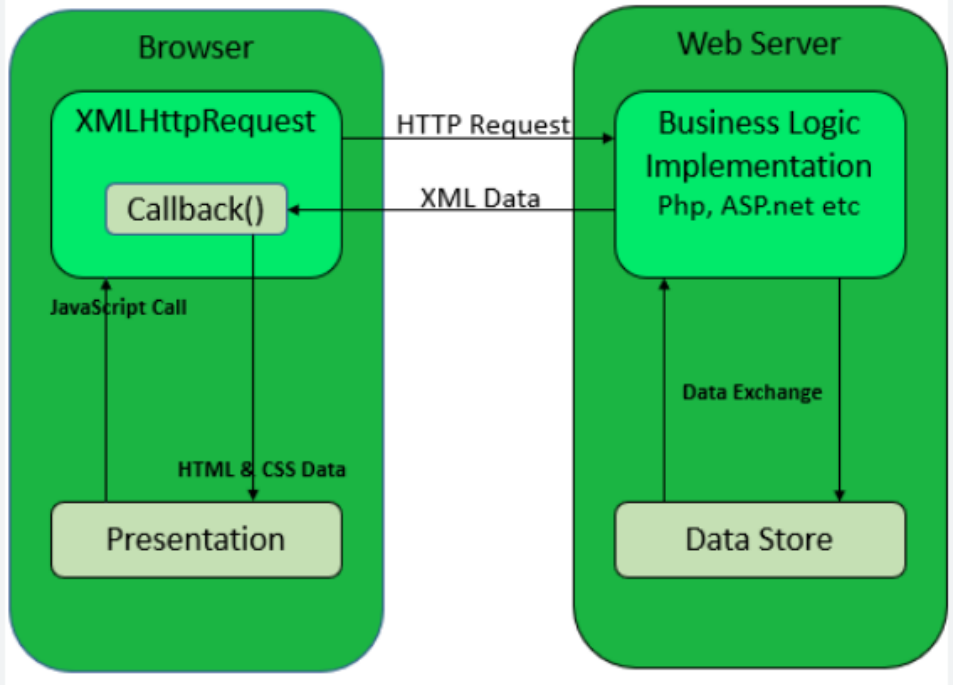

Рисунок 2.1 – Схема роботи AJAX

AJAX – це не самостійна технологія, а швидше концепція використання декількох суміжних технологій. AJAX підхід до розробки, який призначений для користувачів інтерфейсів, комбінує кілька основних методів і прийомів:

- використання DHTML для динамічної зміни змісту сторінки;
- використання XMLHttpRequest для звернення до сервера «на льоту», не перезавантажуючи всю сторінку повністю;
- альтернативний метод – динамічне підвантаження коду JavaScript в тег <SCRIPT> з використанням DOM, що здійснюється із використанням формату JSON);
- динамічне створення дочірніх фреймів.

AJAX – асинхронний, тому користувач може переглядати далі контент сайту, поки сервер все ще обробляє запит. Браузер не перезавантажує web-сторінку і дані посилаються на сервер без візуального підтвердження (крім випадків, коли ми самі захочемо показати процес з’єднання з сервером).

Порівняння класичного підходу та AJAX. Класична модель веб-застосування:

- користувач заходить на веб-сторінку і натискає на який-небудь її елемент;
- браузер надсилає запит серверу;
- у відповідь сервер генерує повністю нову веб-сторінку і відправляє її браузеру і т.д. З боку сервера можлива генерація не всієї сторінки наново, а тільки деяких її частин, з подальшою передачею користувачу.

Модель AJAX:

- користувач заходить на веб-сторінку і натискає на який-небудь її елемент;
- браузер відправляє відповідний запит на сервер;
- сервер віддає тільки ту частину документа, яка змінилася.

Переваги:

- економія трафіку;
- зменшення навантаження на сервер;
- прискорення реакції інтерфейсу;
- майже безмежні можливості для інтерактивної обробки. Наприклад, при введенні пошукового запиту в Google виводиться підказка з можливими варіантами запиту.

Недоліки:

- відсутність інтеграції зі стандартними інструментами браузера;
- інший недолік зміни вмісту сторінки при постійному URL полягає в неможливості збереження закладки на бажаний матеріал;
- динамічно завантажувати вміст недоступний пошуковикам (якщо не перевіряти запит, звичайний він або XMLHttpRequest). Пошукові машини не можуть виконувати JavaScript, тому розробники мають подбати про альтернативні способи доступу до вмісту сайту;

- старі методи обліку статистики сайтів стають неактуальними. Багато сервісів статистики ведуть облік переглядів нових сторінок сайту. Для сайтів, сторінки яких широко використовують AJAX, така статистика втрачає актуальність;
- ускладнення проекту. Перерозподіляється логіка обробки даних – відбувається виділення і часткове перенесення на сторону клієнта процесів первинного форматування даних. Це ускладнює контроль цілісності форматів і типів;
- потрібно включений JavaScript в браузері;
- низька швидкість при грубому програмуванні;
- ризик фабрикації запитів іншими сайтами. Для захисту використовують POST-запит. Але GET вважається ідемпотентність і тому кешируєтся, POST – ні, тому Google вставляє в початок відповіді нескінченний цикл: AJAX може робити з відповіддю що завгодно, у тому числі прибрати цикл, а тег <script> підключить скрипт як є і зациклиться.

4 РЕАЛІЗАЦІЯ ПЛАГІНУ «ВОДНІ РЕСУРСИ»

4.1 Особливості CMS WordPress

Система управління контентом WordPress складається з ядра (Core), тем оформлення і набору скриптів-плагінів, що підключаються і відключаються.

Щоб зрозуміти як розширюється функціонал WordPress належним чином, необхідно розібратися з функціонуванням ядра системи. В ядрі реалізований набір функцій (Core API) для роботи з усіма необхідними компонентами. Core API, згідно, необхідно використовувати тому що:

– Core API надає хукі, дії (actions), фільтри та інші допоміжні функції, які роблять всю «важку» роботу (доступ до БД, помилки під час введення, безпека, створення форм);

– Core API дозволяє повністю інтегрувати в адміністративну частину сторінки налаштування плагіна / теми;

– використання Core API забезпечує сумісність вашого коду в майбутньому.

В Core API входять наступні API:

– API консолі (Dashboard Widgets API) – дозволяє дуже просто додавати нові віджети в адміністративну консоль;

– API доступу до бази даних (Database API) – забезпечує методи доступу до даних, що зберігаються в шарі бази даних;

– API параметрів (Options API) – простий і стандартизований спосіб створення, отримання, відновлення й видалення параметрів в базі даних;

– Plugin API – дозволяє створювати фільтри і дії (actions) використовуючи функції «зачіпки» і методи. Забезпечує взаємодію плагінів з ядром WordPress;

– API інформаційного заголовка (File Header API) – в WordPress теми і плагіни можуть складатися з одного або декількох файлів, один з яких міс-

тить інформаційний заголовок, що містить мета-інформацію (назва, версія, автор і т.д.) щодо конкретного шаблону або плагіна;

– API файлової системи (Filesystem API) – спочатку був створений для власних автоматичних оновлень ядра WordPress, дозволяє здійснювати різні способи підключення до локальної файлової системі;

– HTTP API – стандартизує HTTP запити на WordPress;

– API метаданих (Metadata API) – для маніпулювання метаданими різних типів об'єктів;

– Quicktags API – дозволяє додавати додаткові кнопки в редактор тексту в текстовому режимі (HTML);

– Rewrite API – для управління правилами RewriteRule;

– Settings API – дозволяє сторінкам адміністративної панелі створювати і управляти настройками в напівавтоматичному режимі;

– Shortcode API – надає можливості для створення спеціальних видів контенту, які користувач може додати в пост або інші сторінки шляхом вставки відповідного коду. наприклад для вставки галереї досить в потрібному місці вставити код [gallery];

– API для модифікації тим оформлення (Theme Modification API);

– API налаштування тем (Theme Customization API) – дозволяє додавати призначені для користувача параметри, характерні для конкретної теми;

– Transients API – пропонує можливості для зберігання кеірованних даних;

– Widgets API – дозволяє створювати розробникам власні віджети для використання в темах;

– XML-RPC WordPress API – дозволяє іншим системам підключатися і взаємодіяти з WordPress.

Використовуючи перераховані вище API можна змінювати стандартну поведінку системи, при цьому не турбуватися про сумісність в разі поновлення ядра.

4.2 Розширення функціональності CMS WordPress

Для розширення функціональності системи за допомогою плагіна є дві можливості-створення тегів шаблонів або використання спеціальним чином іменованих функцій – зачіпок.

При додаванні функціональності за допомогою тегів шаблонів, тому, хто хоче використовувати ваш плагін, необхідно додати ці теги в свою тему, в панель, в секцію вмісту записи, або в інше відповідне місце. Щоб оголосити тег шаблону, досить написати функцію на PHP, і задокументувати її для користувачів плагіна на сторінці присвяченій плагіну і/або в головному файлі плагіна.

Інший шлях додавання функціональності за допомогою плагіна – використання функцій-зачіпок. Для того, щоб плагіни мали можливість впливати на роботу ядра Wordpress або на кінцевий результат його дій, була придумана система так званих зачіпок (часто їх без перекладу так і називають «хукамі» від англ. hook – гачок, зачіпка).

Принцип її дії полягає в тому, що кожна більш-менш важлива елементарна функція в ядрі Wordpress перед тим як повернути якийсь результат своєї роботи або зробити якусь важливу дію «намагається» виконати додаткові інструкції (рядки коду) призначені саме для неї в файлах плагіна. Таку спробу вона робить за допомогою зачіпок, які прописані в тілі цієї функції.

Все зачіпки в Wordpress діляться на дві категорії – Фільтри та Дії. Фільтри (Filters) дійсно призначені для «фільтрування» (зміни) будь-яких даних перед тим як вони будуть виведені на сторінці або додані для зберігання в базу даних. Це фільтрація спаму, помилок або просто помилкового введення в формах, звідки власне і появилася назва.

Дії (Actions) призначені для заміни різних дій ядра вашими діями (наприклад, зміни рядка запиту до бази даних), в програмуванні таку зміну дій базового функціоналу ще називають перевантаженням.

4.3 Проектування плагіну

Блок «Інформаційно-пошукова система Одеської області» є реалізацію роботи плагіна, етапи розробки якого наводяться у наступному розділу дипломного проекту. Кожен з блоків володіє заголовком, для якого ми підбираємо фон, шрифт і позиціонування.

Інформаційна-пошукова система представлена картою Одеської області з вказаними кордонами районів (рис.4.1). Карта виконана у форматі SVG.

Рисунок 4.1 – Вигляд інформаційно-пошукової системи

SVG (Scalable Vector Graphics) – це мова розмітки масштабованої векторної графіки, що входить в підмножину розширюваної мови розмітки XML. Кожний район має свій class і id та є окремим об’єктом, який можна використовувати для організації інтерактивних переходів. Карта створена у графічному редакторі Xara. За допомогою CSS задані бажані стилі та кольори відображення карти при натисканні на об’єкти кнопкою миші. Далі SVG був доданий до HTML сторінки за допомогою наступного коду:

```
<object data="map.svg" type="image/svg+xml" id="imap" width="500"
height="420"></object>
```

4.4 Реалізація плагіна

Одна з головних причин популярності CMS WordPress це легкість, з якої можна розширювати її функціонал. Модулі дають практично безмежні можливості в розширенні функцій системи. Існує безліч готових плагінів, які можна скачати в репозиторії.

Першим кроком у створенні плагіна для WordPress є створення нового PHP файлу для коду плагіна. Назва плагіна (Plugin Name) має бути унікальним і відповідати функціоналу плагіна для більш легкої ідентифікації в каталозі плагінів. Унікальне назва необхідно через те, що всі плагіни розташовані в одному каталозі і є ризик запуску ьіншого плагіна, що має таку ж назву. Плагіном може бути як один файл, так і папка, що містить всі необхідні файли для роботи плагіна.

Початок файлу повинна містити стандартний інформаційний Заголовок. Цей заголовок дозволяє WordPress зрозуміти, що пагін існує, додати його в панель управління плагінами, де він може бути активований, завантажити його і запустити його функції. Без заголовка пагін ніколи не буде активований і запущений.

Формат заголовка має вигляд:

```
<? Php
```

```
/ *
Plugin Name: Назва плагіна
Plugin URI: http: // страніца_с_опісаніем_плагіна
_і_его_обновленій
Description: Короткий опис плагіна.
Version: Номер версії плагіна, наприклад: 1.0
Author: Ім'я автора плагіна
Author URI: http: // страніца_автора_плагіна
* /
?>
```

Мінімальна інформація, яка потрібна WordPress, щоб виявити плагін – його назва. Решта інформації використовується для створення таблиці плагінів на сторінці управління плагінами. Порядок рядків неважливий.

Для зручності подальшого викладу розділимо нашу задачу на наступні підзадачі:

- створення окремої сторінки управління плагіном;
- реалізація архітектури плагіна;
- реалізація функціонала управління районами в плагіні;
- реалізація функціонала управління басейнами річок в плагіні;
- реалізація функціонала управління типами водних об’єктів в плагіні;
- реалізація функціонала управління типа використання водних об’єктів;
- виведення інформації користувачеві за шаблоном.

Так як для вирішення поставленого завдання в реалізації плагіна необхідно організувати зберігання і обробку великих обсягів інформації, нестандартне для WordPress виведення контенту і здійснення маніпуляцій над ним, було прийнято рішення в реалізації власної архітектури плагіна. При розробці серверних веб додатків на мові PHP все частіше застосовується об'єктно-орієнтований підхід з використанням архітектурного шаблону MVC. Цей шаблон передбачає поділ системи на кілька компонентів, які називаються модель (Model), вид (View) і контролер (Controller), завдяки чому бізнес-логіка програми відділяється від призначеного для користувача інтерфейсу. Такий додаток простіше змінювати, масштабувати, тестувати і супроводжувати.

Багато плагінів мають опції і налаштування, які дозволяють налаштувати плагін. Для зручності користувача в системі WordPress є можливість

створити спеціальну сторінку в адміністративній частині, на якої можна керувати опціями плагіна. Для створення такої сторінки необхідно використовувати функцію-зачіпку create_water_res() . Програмний код PHP файлу плагіну наведено у додатку А. На рис. 4.2 представлений вигляд адміністративної панелі плагіну.

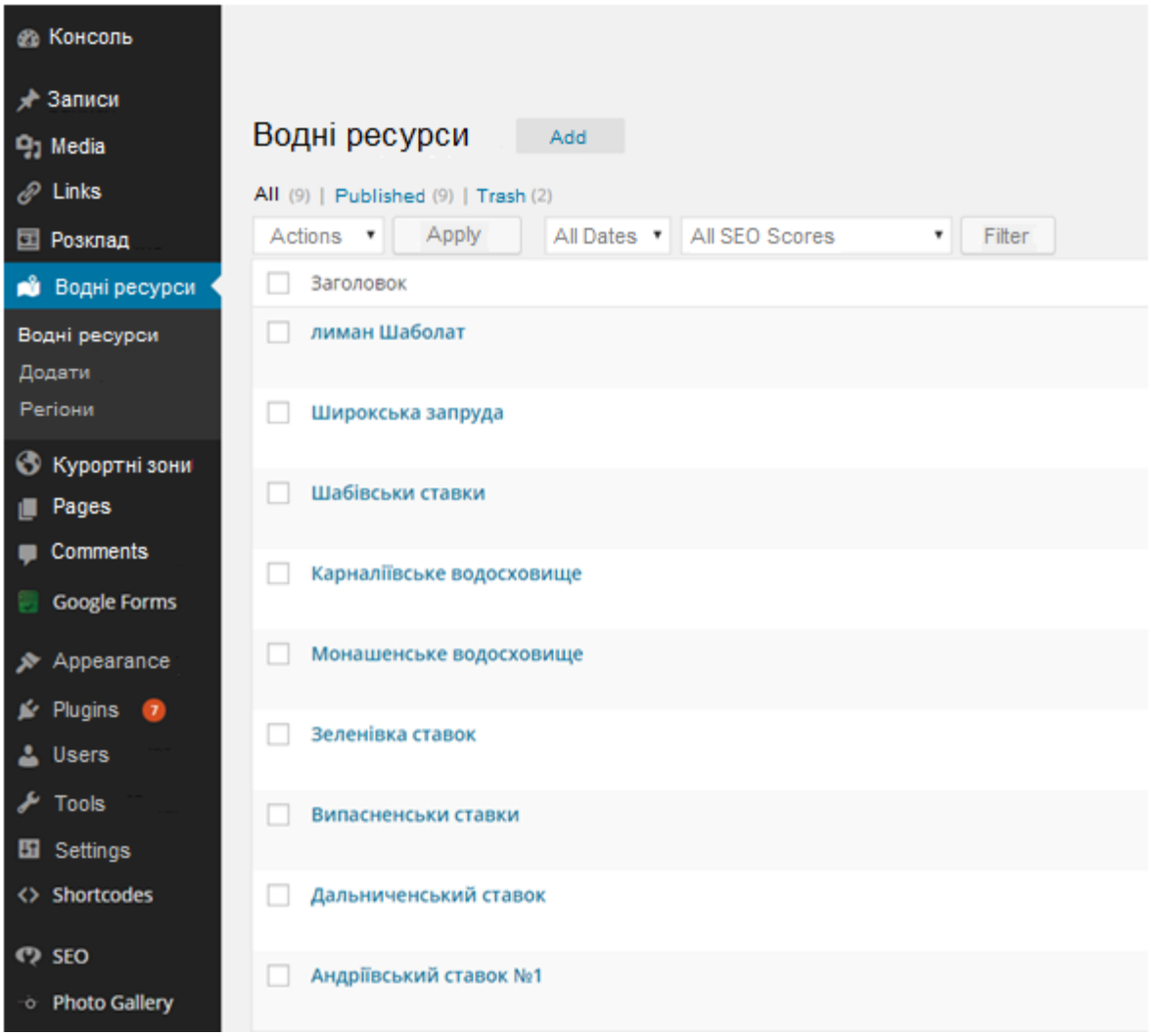

Рисунок 4.2 – Сторінка водних ресурсів

Частина плагіна, що реалізує функціонал управління районами, дозволяє користувачам створювати, видаляти і змінювати інформацію про райони в адміністративній панелі. Фрагмент сторінки створення, видалення і зміни

інформації про райони Одеської області в адміністративній частині плагіну «Водні ресурси» наведений на рис. 4.3.

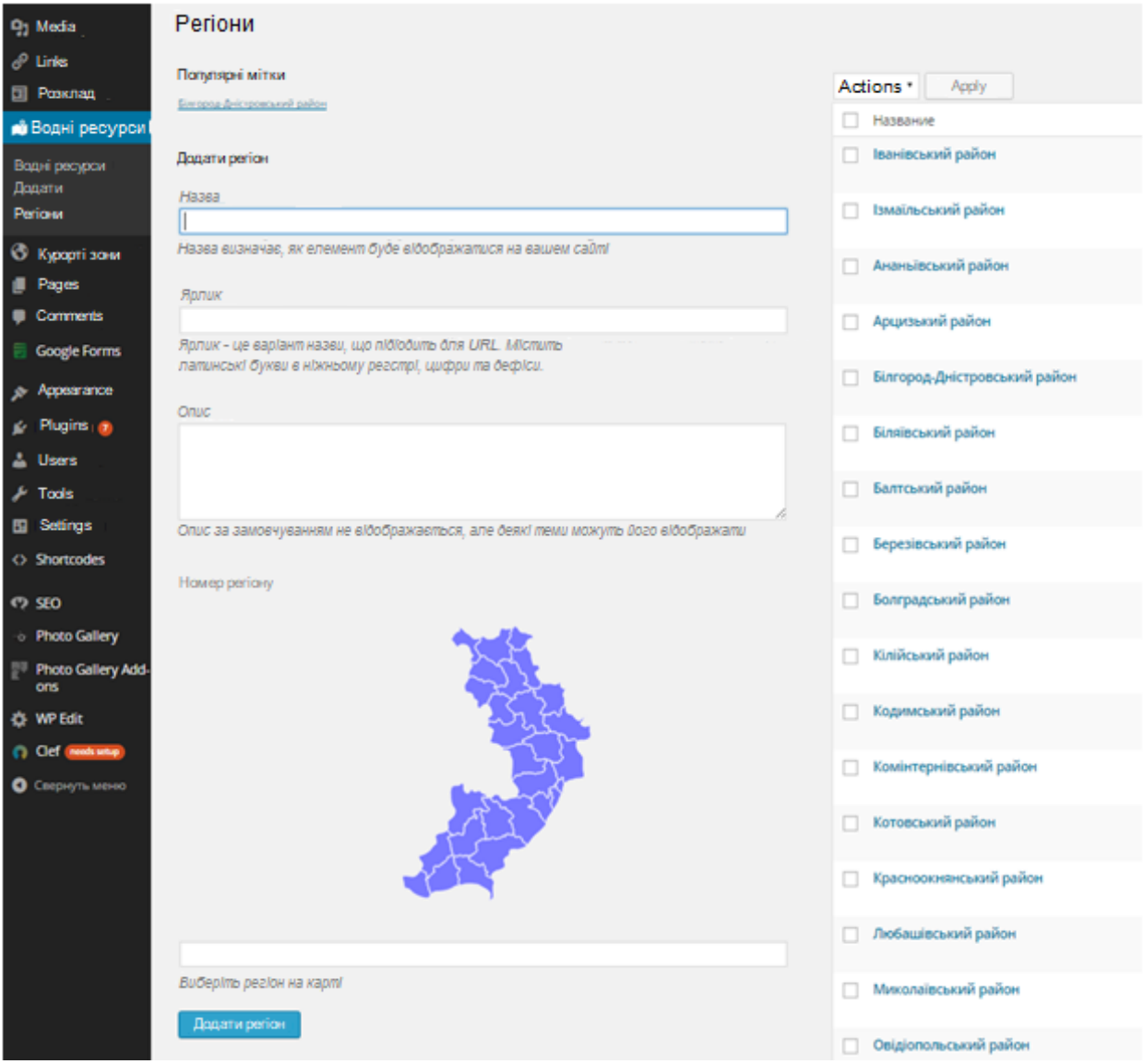

Рисунок 4.3 – Фрагмент сторінки створення, видалення і зміни інформації про райони

В адміністративну частину плагіну винесена карта Одеської області, кожному району якої привласнено унікальний номер water_res_region_N. Для додавання району адміністратору потрібно вибрати його на карті, при цьому його номер автоматично з’явиться під картою. Далі необхідно додати назву

району, ярлик – варіант назви, відповідний для URL. Зазвичай містить тільки латинські букви в нижньому регістрі, цифри та дефіси. За бажанням можна додати опис для району. Після всіх налаштувань адміністратор повинен натиснути кнопку «Додати регіон» і назва району з’явиться в правій частині сторінки (рис.4.3).

Подібні налаштування можна виконати для того, щоб вказати басейн річки, якому належить водний об’єкт, його тип і форму власності.

Щоб додати новий водний об’єкт необхідно вибрати пункт «Додати» і з’явиться сторінка для редагування відомостей про водний об’єкт (рис.4.4).

На цій сторінці слід ввести ім’я нового водного об’єкту, бажану інформацію, фотозображення. Після редагування даних в правій частині сторінки (рис.4.4) можна обрати район до якого відноситься водний об’єкт і додати координати для виведення карти з маркером розміщення даного об’єкту.

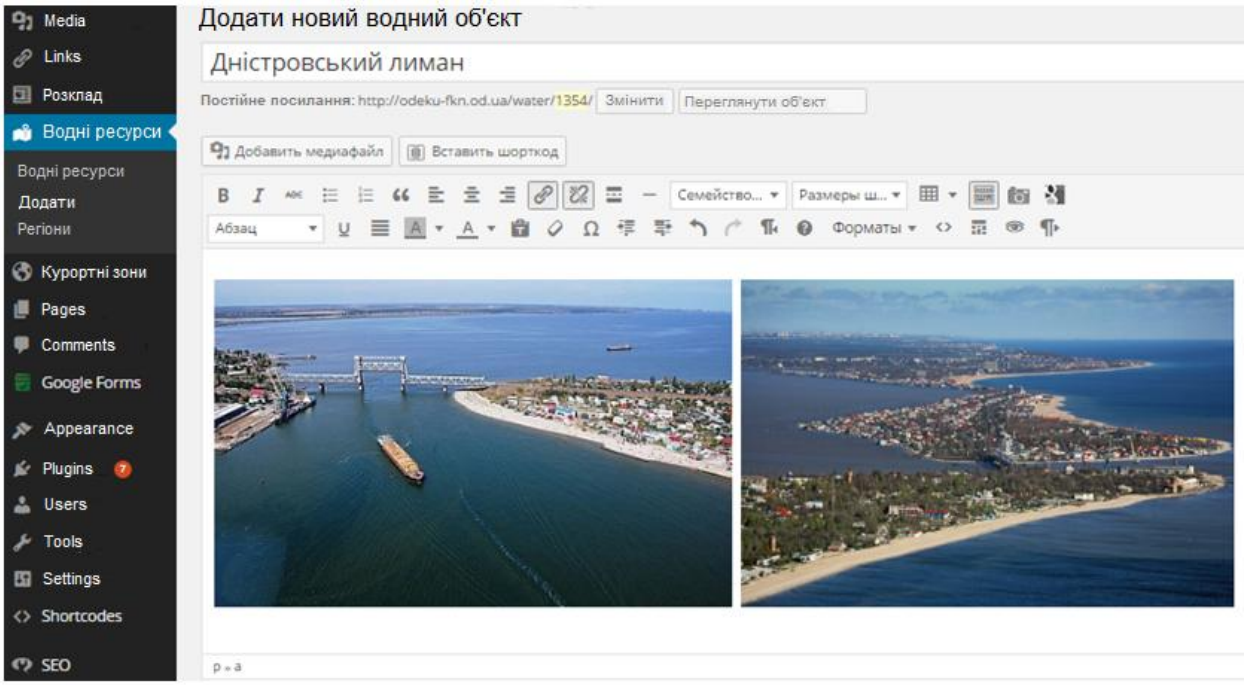

Рисунок 4.4 – Фрагмент сторінки додавання інформації
про водний об’єкт

Після усіх необхідних налаштувань необхідно нажати кнопку «Опублікувати» (рис.4.5) і новий водний об’єкт зявиться на сторінки водних ресурсів і буде доступним для показу користувачеві.

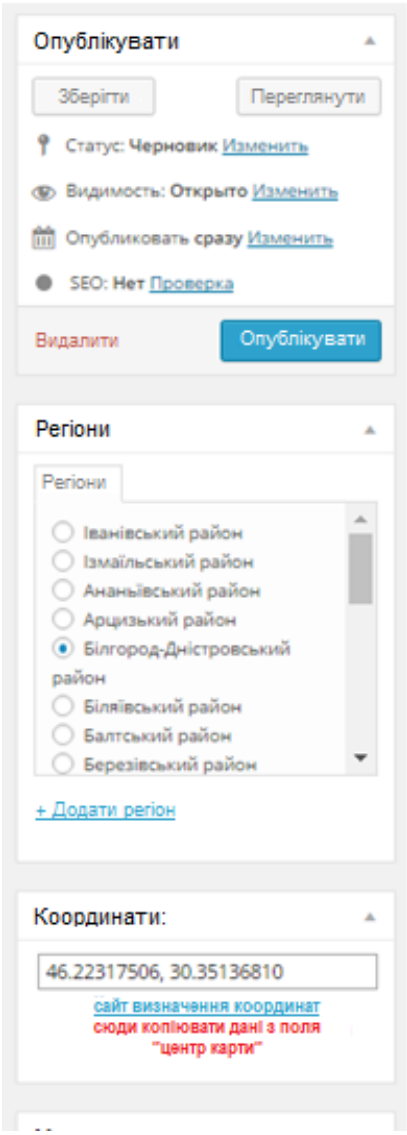

Рисунок 4.5 – Фрагмент сторінки додавання інформації про водний об’єкт (вібір району і місця розташування)

Окремо треба розглянути реалізацію механізму пошуку. Важко знати наперед, що може знадобиться шукати в базі даних. Чим більше критеріїв для пошуку задано в формі пошуку, тим більше фільтрів знадобиться в запиті.

Вирішити цю проблему можна додаванням додаткових методів до класів, які відповідають за формування запиту.

Це не дуже гнучкий варіант. Інший варіант використовувати патерн Identity Object, який інкапсулює умовну частину запиту до бази даних таким способом, що різні комбінації можуть об'єднуватися під час виконання.

Клас Identity Object зазвичай складається з набору методів, які можна викликати для побудови критерію запитів. У нашому класі Identity Object були реалізовані наступні методи:

– Table() – встановлює з якої таблиці бази даних буде здійснюватися вибірка;

– Field() – ім'я поля з обраної таблиці для порівняння;

– Eq(), lt(), gt() – операції рівності, «менше ніж», «більше ніж».

Оперуючи вищепереліченими методами можна задати різні критерії для вибірки даних. При цьому, відмовившись від жорстко закодованих методів, щоб не втратити контроль над безпекою та отримати помилку в разі звернення до відсутнім в таблиці полю – є масив, що містить набір обмежень. Визначивши стан об'єкта, він передається методу, який відповідає за формування SQL-запиту.

Перевагою використання даного патерну є легкість і гнучкість формування критеріїв. Наприклад, такі рядки коду:

```
$ idobj-> table ( 'regions')
-> Field ( 'region')
-> Eq ( 'Білгород-Дністровський район');
```

сформують умова для SQL-запиту подібно пропозицією: таблиця «Регіони», поле «назва регіону» дорівнює «Білгород-Дністровський район». Таким чином, знаючи імена таблиць і поля бази даних, використовуючи клас Identity Object можна задати необхідні умови для вибірки даних з бази даних.

Інтерпретація умови для SQL-запиту відбувається в «магічних» методах _toString () класу композиту (Identity Object) і його частинах.

При створені плагіну окрема увага приділялося його оптимізації. Швидкість завантаження веб-сайтів завжди була актуальною темою. Існують різні способи зменшення часу – оптимізація параметрів сервера, мінімізація розміру CSS, Javascript і інших файлів.

Також важливим фактором є швидкість, з якою веб-сторінки формуються на сервері. В рамках етапу оптимізації плагіна була проведена оптимізація для пошукових систем, оптимізація бази даних і організовано кешування сторінок. Оптимізація сайту під пошукові системи (SEO) дуже важлива, адже від того як оптимізований сайт залежить позиція в результатах видачі пошукової системи. Для всіх генеруються плагіном сторінок створюються заголовки, мета-теги description і keywords. Оптимізація бази даних полягала в оптимізації запитів і структури таблиць бази даних. Так після реалізації прототипу плагіна були виявлені «вузькі місця» і проведена денормализация бази даних. Денормализация – навмисне приведення структури бази даних в стан, що не відповідає критеріям нормалізації, зазвичай приводиться з метою прискорення операцій читання з бази за рахунок додавання надлишкових даних.

Також на цьому етапі були більш ретельно проаналізовані запити до бази даних і встановлені відповідні індекси. Для перегляду, в якому порядку зв'язуються таблиці і які індекси використовуються в запиті на вибірку використовується оператор EXPLAIN. Індекси застосовуються для швидкого пошуку рядків с вказаним значенням одного стовпчика. Без індексу читання таблиці здійснюється по всій таблиці починаючи з першого запису, поки не будуть знайдені відповідні рядки. Чим більше таблиця, тим більше накладні витрати. Так як дані в базі даних оновлюються не часто, має сенс задіяти кешування сторінок для зниження навантаження на базу даних. У нашому плагіні кешування сторінок здійснюється в файл, такий підхід зменшує час формування веб-сторінки на сервері при повторному її перегляді.

4.5 Виведення даних для користувача

Як і багато сучасних систем управління контентом, WordPress має вбудовану підтримку тим, що дозволяє з легкістю змінити зовнішній вигляд сайту. Існує безліч готових шаблонів, як платних, так і безкоштовних, які можна переглянути і завантажити на офіційному сайті та інших джерелах. Також їх можна розробити самостійно. При розробці програмісти дотримуються прийнятих в WordPress стандартів кодування – весь код, пов'язаний з виведенням даних на екран користувача, виносять в окремі функції. Такі функції зазвичай не реалізують ніякої логіки, а лише приймають набір даних і повертають HTML-код.

Для частини плагіну, яка виводить інформацію для користувача була розроблена форма пошуку приведена на рис. 4.6.

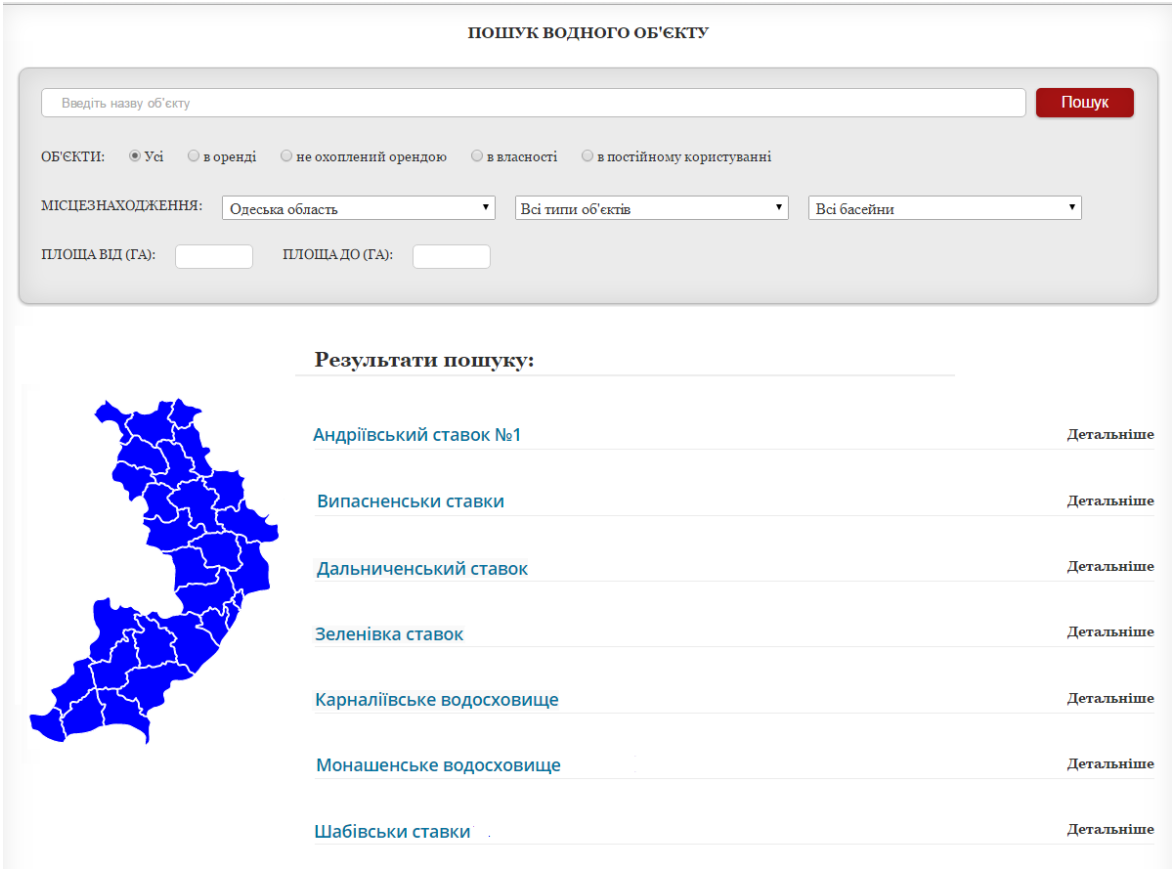

Рисунок 4.6 – Форма пошуку

При натисканні на район інтерактивної карти завантажується сторінка з формою пошуку, де користувач може обрати необхідні фільтри пошуку. За замовчуванням на формі виводяться списком водні ресурси для обраного району. Натиснувши Детальніше користувач має можливість переглянути інформацію по обраному водному об’єкту (рис.4.7).

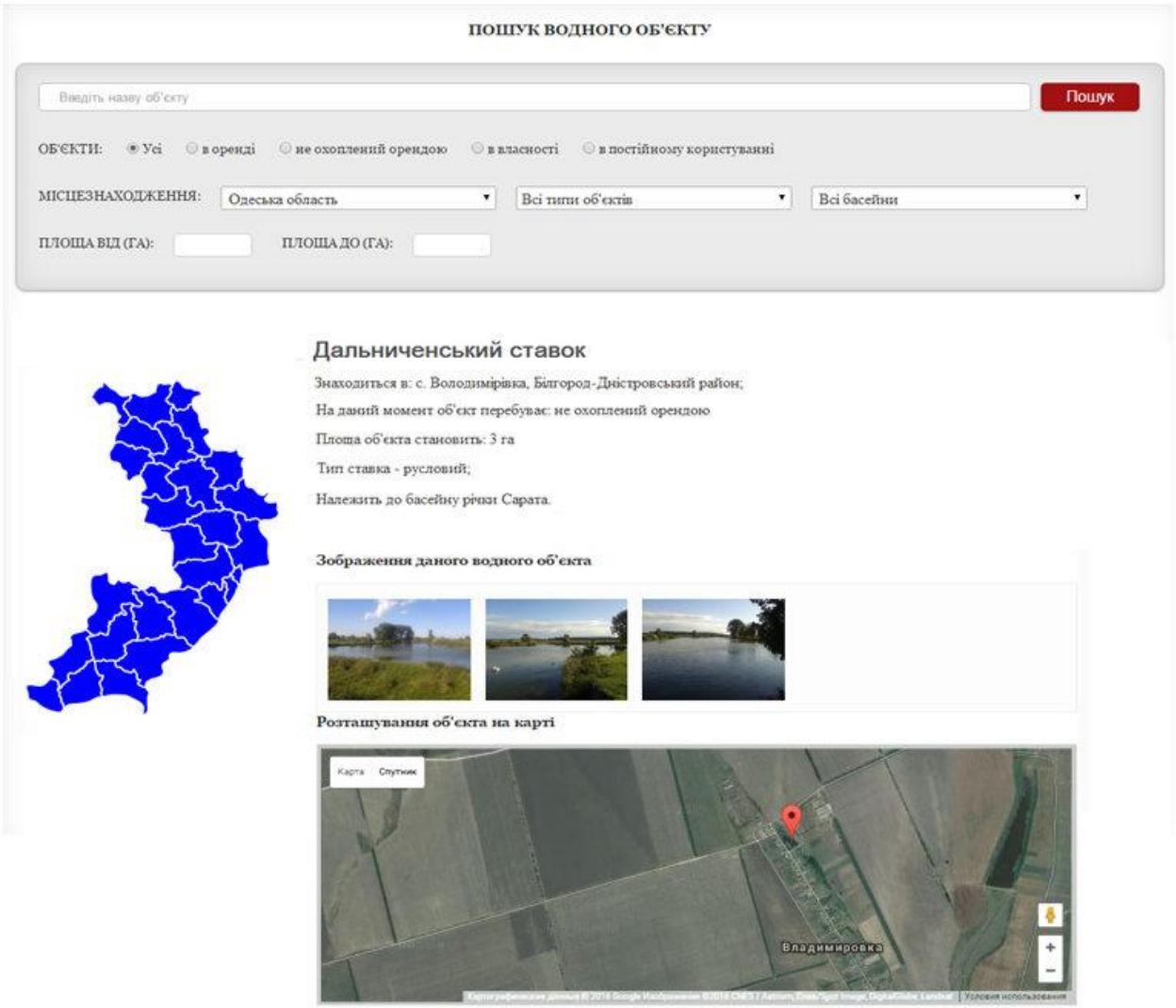

Рисунок 4.7 – Результат пошуку

Останнім часом все більше планшетних комп'ютерів, смартфонів і навіть телевізорів використовується для перегляду web-сторінок. Розробникам потрібно брати до уваги велике розмаїття розмірів екранів різних пристроїв. Адаптивний веб-дизайн дозволяє відображати вміст сайтів найкращим чином

на екранах пристроїв, використовуваних для перегляду. Використовуючи адаптивний веб-дизайн немає необхідності в створенні окремих версій веб-сайту для окремих видів пристроїв.

Twitter Bootstrap – популярний фреймворк для розробки адаптивних і мобільних web-додатків. Його використання дозволило з легкістю оформити дані, що подаються користувачу.

Bootstrap включає в себе безліч компонентів і утиліт. Одним з найважливіших компонентів є сіткова система, що дозволяє створити макет сторінки, розбивши його на стовпці і рядки, в які поміщається контент. Стандартно система налаштована на 12 стовпців (колонок в сітці), але цей і безліч інших параметрів можна змінити, скомпілювавши власну версію Bootstrap, яка буде містити лише необхідні компоненти, утиліти і CSS. Використовуючи сітковий систему можна налаштувати відображення сторінки для різних видів пристроїв.

Для зручності користувачів був підключений плагін Bootstrap Select, що дозволяє створювати меню, що випадає. Плагін має безліч опцій і методів для настройки списку. Є можливість пошуку серед елементів списку і ряд інших зручних механізмів для взаємодії зі списками.

ВИСНОВКИ

Метою даної дипломної роботи було розроблення інформаційно-пошукової системи у вигляді вбудованого компонента WEB сайту для надання доступу до даних по водним об’єктам Одеської області.

В ході виконання проекту була вивчена предметна область, сформульовані вимоги до інформаційної системи організації, проведено проектування бази даних, розроблена архітектура системи і призначеного для користувача інтерфейсу, спроектована архітектура плагіна «Водні ресурси» і здійснена програмна реалізація плагіну для системи управління контентом WordPress. Зроблено аналіз і оптимізація плагіну. Ці етапи докладно описані в пояснювальній записці, тексти програмних модулів наведені в додатках.

Таким чином, всі поставлені завдання виконані і мета роботи досягнута.

Перевагами розробленої інформаційної системи є:

– простий, інтуїтивно зрозумілий інтерфейс;

– інформування користувачів про місцезнаходження обраних водних ресурсів;

– надання користувачам доступу до БД водних об’єктів Одеської області за різними фільтрами пошуку і виведення результатів у вигляді текстової інформації, зображення об’єкту і карти його розташування.

Система реалізована з використанням сучасних програмних засобів (СКБД MS SQL, система управління контентом WordPress, мова програмування PHP і JavaScript, технологія Ajax).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Документація для розробника. URL: http://codex.wordpress.org/Developer_Documentation (дата звернення 13.05.2024)
2. Леон Аткинсон: «MySQL. Бібліотека професіонала», 2002. 31 с.
3. MVC: Model View Controller. URL: https://uk.wikipedia.org/wiki/Model-View-Controller (дата звернення 18.05.2024)
4. HTML: HyperText Markup Language. URL: http://uk.wikipedia.org/wiki/HTML (дата звернення 18.05.2024)
5. Шикула, О. М. Вступ до сучасного Web-дизайну: HTML5+CSS3. Навчальний посібник. Київ: ІПДО, 2019. 240 с.
6. Мулеса О.Ю. Основи HTML та CSS. Лабораторний практикум. Ужгород, 2019. 53 с.
7. CSS: Cascading Style Sheets. URL:http://uk.wikipedia.org/wiki/CSS (дата звернення 18.05.2024)
8. Нечухаєва Н.В., Расчубкін В.Г.: Вивчення мови HTML з використанням каскаду стилів CSS та елементів мови JavaScript. Навч. посібник. Дніпропетровськ: НМетАУ, 2012. 60с
9. Босько В.В., Константинова Л.В., Марченко К.М., Улічев О.С. Web-програмування. Частина 1 (frontend): навч. посіб. Кропивницький: ЦНТУ, 2022. 208 с.
10. JavaScript. URL: http://uk.wikipedia.org/wiki/JavaScript (дата звернення 18.05.2024)
11. Цеслів О.В. Основи програмування та веб-дизайн. Навч. посіб. К.,2020. 149 с.
12. JavaScript: Онлайн підручник. URL: http://www.wisdomweb.ru/JS/javascript-first.php (дата звернення 18.05.2024)
13. jQuery. URL:http://uk.wikipedia.org/wiki/JQuery (дата звернення 18.05.2024)

14. jQuery: Онлайн підручник URL: http://www.wisdomweb.ru/JQ/ jquery-first.php (дата звернення 18.05.2024)
15. PHP: Hypertext Preprocessor. URL: http://uk.wikipedia.org/wiki/PHP (дата звернення 18.05.2024)
16. Довідник з PHP. URL: http://www.php.su (дата звернення 18.05.2024)
17. Васильєв О.М. Програмування мовою PHP. Ліра-К, 2022. 368 с.
18. Молчанов В. П., Пандорін О. К. Технології розробки WEB-ресурсів. Навч. Посібник. Харків : ХНЕУ ім. С. Кузнеця, 2019. 130 с.
19. SQL: Structured Query Language. URL:http://uk.wikipedia.org/ wiki/SQL (дата звернення 18.05.2024)
20. Довідник з MySQL. URL:http://www.mysql.ru/ docs/man/ (дата звернення 18.05.2024)
21. Балик Н.Р., Мандзюк В.І. Бази даних MySQL: Навч. посіб. Тернопіль: Навчальна книга Богдан, 2010. 160 с.
22. AJAX: Asynchronous Javascript and XML. URL: http:// uk.wikipedia.org/wiki/AJAX (дата звернення 18.05.2024)

ДОДАТОК А

Програмний код плагіна «Водні ресурси»

```
<?php
/*
Plugin Name: Модуль водных ресурсов
Plugin URI: http://vk.com/profiler4111
Description: Позволяет выводить на сайте водные ресурсы.
Version: 1.0
Author: Кобзар В.
Author URI: http://vk.com/profiler4111
License: GPLv2
*/
function create_water_res() {
    register_post_type( 'water_res',
        array(
            'labels' => array(
                'name' => 'Водные ресурсы',
                'singular_name' => 'Водный ресурс',
                'add_new' => 'Добавить',
                'add_new_item' => 'Добавить новый водный ресурс',
                'edit' => 'Редактировать',
                'edit_item' => 'Редактировать водный ресурс',
                'new_item' => 'Новый водный ресурс',
                'view' => 'Просмотреть',
                'view_item' => 'Просмотреть водный ресурс',
                'search_items' => 'Найти водный ресурс',
                'not_found' => 'Ни одного водного ресурса не найдено',
                'not_found_in_trash' => 'Ни одного водного ресурса не найдено
в корзине',
                'parent' => 'Родительский водный ресурс'
            ),
                  'taxonomies' => array('water_res_regions'),
            'public' => true,
            'menu_position' => 15,
            'supports' => array('title','editor','thumbnail'),
            'menu_icon' => 'dashicons-location-alt',
//            'publicly_queriable' => true,
//            'show_ui' => true,
//            'exclude_from_search' => true,
//            'show_in_nav_menus' => false,
//            'has_archive' => false,
                'rewrite' => array(
                'slug' => 'water',
            ),
        )
    );
      register_taxonomy(
            'water_res_regions',
            'water_res',
            array(
                  'labels' => array(
                        'name' => 'Регионы',
                        'add_new_item' => 'Добавить регион',
                        'new_item_name' => "Регион"
                  ),
```

```
                  'show_ui' => true,
                  'show_tagcloud' => false,
                  'query_var' => true,
                  'hierarchical' => false,
                  'rewrite' => array(
                        'slug' => 'water-res'
                  )
            )
      );
}
function water_admin_init(){
      add_meta_box("water_data_gps", "Кооринаты:", "water_data_gps_callback",
"water_res", "side", "low");
}
if(!function_exists("the_post_thumbnail_url")) {
      function the_post_thumbnail_url($size = 'post-thumbnail') {
            $url = get_the_post_thumbnail_url(NULL, $size);
            if ($url) {
                  echo esc_url($url);
            }
      }}
if(!function_exists("get_the_post_thumbnail_url")) {
      function get_the_post_thumbnail_url( $post = null, $size = 'post-
thumbnail' ) {
            $post_thumbnail_id = get_post_thumbnail_id( $post );
            if ( ! $post_thumbnail_id ) {
                  return false;
            }
            return wp_get_attachment_image_url( $post_thumbnail_id, $size );
      }}
if(!function_exists("wp_get_attachment_image_url")) {
      function wp_get_attachment_image_url( $attachment_id, $size =
'thumbnail', $icon = false ) {
            $image = wp_get_attachment_image_src( $attachment_id, $size, $icon
);
            return isset( $image['0'] ) ? $image['0'] : false;
      }}
if(!function_exists("otf_get_attachment_image_src")) {
      /*
             * Check for thumb of $size and generate it if it doesn't exist.
             *
             * @param int $post_id Post ID for which you want to retrieve
thumbnail
             * @param string $size Name of thumbnail size to check for; same
as
             *     the slug used to add custom thumb sizes with
add_image_size().
             * @return array An array containing: array( 0 => url, 1 => width,
2 => height )
             *
             */
      function otf_get_attachment_image_src($post_id, $size = 'thumbnail',
$force = FALSE) {
            $attachment_id = get_post_thumbnail_id($post_id);
            $attachment_meta = wp_get_attachment_metadata($attachment_id);
            $sizes = array_keys($attachment_meta['sizes']);
            if (in_array($size, $sizes) && empty($force)) {
```

```
                return wp_get_attachment_image_src($attachment_id, $size);
            }
            else {
                include_once ABSPATH . '/wp-admin/includes/image.php';
                $generated = wp_generate_attachment_metadata(
                    $attachment_id, get_attached_file($attachment_id));
                $updated = wp_update_attachment_metadata($attachment_id,
$generated);
                return wp_get_attachment_image_src($attachment_id, $size);
            }
      }
}
add_image_size('water_res_thumb', 410, 200, true); // For Our Menu Carousel

function water_res_save_details() {
      global $post;
      if( !isset($post) ) return;
      if( isset($_POST["water_coordinats"]) )
            update_post_meta($post->ID, "water_coordinats",
@$_POST["water_coordinats"]);
}
function water_data_gps_callback () {
      global $post;
      $custom = get_post_custom($post->ID);
      $coordinats = @$custom["water_coordinats"][0];
      ?><table style="width:100%;">
      <tbody>
      <tr>
            <td><input name="water_coordinats" value="<?=$coordinats;?>"
style="width:100%;box-sizing:border-box;" /></td>
      </tr>
      <tr>
            <td style="text-align: center"><a
href="http://dimik.github.io/ymaps/examples/location-tool/"
target="_blank"><b>сайт определения координат</b></a><br><span style="color:
red;">сюда копировать данные из поля "центр карты"</span></td>
      </tr>
      </tbody>
      </table><?php
}
function water_data_location_callback () {
      global $post;
      $custom = get_post_custom($post->ID);
      $coordinats = @$custom["water_coordinats"][0];
      ?>
      <table style="width:100%;">
            <tbody>
            <tr>
                  <td colspan="2"><img id="water_res_map" src="<?=plugins_url(
'/img/water_res_map.svg', __FILE__ );?>" class="svg" /></td>
            </tr>
            <tr>
                  <td>Номер региона</td>
                  <td><input type="text" name="water_res_region"
id="water_res_region"></td>
            </tr>
            </tbody>
```

```
	</table>
	<style>
		#water_res_map {
			min-height: 350px;
		}
		#water_res_map g > path {
			stroke: white;
			stroke-width: 1px;
			fill: rgb(120, 120, 255);
		}
		#water_res_map g[activity="active"] > path, #water_res_map g >
path:hover {
			fill: rgb(80, 80, 180);
		}
	</style>
	<script>
		(function($){
			$('img.svg').each(function(){
				var $img = jQuery(this);
				var imgID = $img.attr('id');
				var imgClass = $img.attr('class');
				var imgURL = $img.attr('src');
				jQuery.get(imgURL, function(data) {
					// Get the SVG tag, ignore the rest
					var $svg = jQuery(data).find('svg');
					// Add replaced image's ID to the new SVG
					if(typeof imgID !== 'undefined') {
						$svg = $svg.attr('id', imgID);
					}
					// Add replaced image's classes to the new SVG
					if(typeof imgClass !== 'undefined') {
						$svg = $svg.attr('class', imgClass+'
replaced-svg');
					}
					// Remove any invalid XML tags as per
http://validator.w3.org
					$svg = $svg.removeAttr('xmlns:a');
					// Replace image with new SVG
					$img.replaceWith($svg);
					$svg.find('path').click(function(){

	$(this).parents('g#Spread').children().attr('activity','nonactive');

	$(this).parent().attr('activity','active');
						console.log($(this).attr('id'));

	$('#water_res_region').val($(this).attr('id'));
					});
				}, 'xml');
			});
		})(jQuery);
	</script><?php
}
function water_res_regions_add_map_field() {
	?>
	<div class="form-field">
```

```
            <label for="water_res_region_val"><?php _e( 'Номер региона',
'pippin' ); ?></label>
            <div style="text-align: center;"><img id="water_res_map"
src="<?=plugins_url( '/img/water_res_map.svg', __FILE__ );?>" class="svg"
/></div>
            <input type="text" name="term_meta[water_res_region]"
id="water_res_region_val" value="">
            <p class="description"><?php _e( 'Выберите регион на кар-
те','pippin' ); ?></p>
      </div>
      <style>
            #water_res_map {
                  min-height: 350px;
                  max-height: 350px;
            }
            #water_res_map g > path {
                  stroke: white;
                  stroke-width: 1px;
                  fill: rgb(120, 120, 255);
            }
            #water_res_map g[activity="active"] > path, #water_res_map g >
path:hover {
                  fill: rgb(80, 80, 180);
            }
      </style>
      <script>
            (function($){
                  $('img.svg').each(function(){
                        var $img = jQuery(this);
                        var imgID = $img.attr('id');
                        var imgClass = $img.attr('class');
                        var imgURL = $img.attr('src');
                        jQuery.get(imgURL, function(data) {
                              // Get the SVG tag, ignore the rest
                              var $svg = jQuery(data).find('svg');
                              // Add replaced image's ID to the new SVG
                              if(typeof imgID !== 'undefined') {
                                    $svg = $svg.attr('id', imgID);
                              }
                              // Add replaced image's classes to the new SVG
                              if(typeof imgClass !== 'undefined') {
                                    $svg = $svg.attr('class', imgClass+'
replaced-svg');
                              }
                              // Remove any invalid XML tags as per
http://validator.w3.org
                              $svg = $svg.removeAttr('xmlns:a');
                              // Replace image with new SVG
                              $img.replaceWith($svg);

                              $svg.find('path').click(function(){

      $(this).parents('g#Spread').children().attr('activity','nonactive');

      $(this).parent().attr('activity','active');
                                    console.log($(this).attr('id'));
```

```
      $('#water_res_region_val').val($(this).attr('id'));

                              });
                        }, 'xml');
                  });
            })(jQuery);
      </script><?php
}
function water_res_regions_edit_map_field($tag) {
      $t_id = $tag->term_id; // Get the ID of the term you're editing
      $term_meta = get_option( "taxonomy_$t_id" ); // Do the check
      ?>
      <table class="form-table">
            <tbody>
                  <tr class="form-field form-required">
                        <th scope="row">
                              <label for="term_meta[water_res_region]"><?php
_e( 'Тип недели по порядку', 'pippin' ); ?></label>
                        </th>
                        <td>
                              <input type="text"
name="term_meta[water_res_region]" id="term_meta[water_res_region]"
value="<?php echo $term_meta['water_res_region'] ?
$term_meta['water_res_region'] : ''; ?>">
                              <p class="description"><?php _e( 'Введите значе-
ние','pippin' ); ?></p>
                        </td>
                  </tr>
            </tbody>
      </table><?php
}
function weather_res_save_tax_metas( $term_id ) {
      if ( isset( $_POST['term_meta'] ) ) {
            $t_id = $term_id;
            $term_meta = get_option( "taxonomy_$t_id" );
            $cat_keys = array_keys( $_POST['term_meta'] );
            foreach ( $cat_keys as $key ) {
                  if ( isset ( $_POST['term_meta'][$key] ) ) {
                        $term_meta[$key] = $_POST['term_meta'][$key];
                  }
            }
            // Save the option array.
            update_option( "taxonomy_$t_id", $term_meta );
      }
}
add_action( 'init', 'create_water_res' );
add_action( 'admin_init', 'water_admin_init' );
add_action('save_post_water_res', 'water_res_save_details', 10, 3);
add_action( 'water_res_regions_add_form_fields',
'water_res_regions_add_map_field', 10, 2 );
add_action( 'water_res_regions_edit_form_fields',
'water_res_regions_edit_map_field', 10, 2 );
add_action( 'edited_water_res_regions', 'weather_res_save_tax_metas', 10, 2
);
add_action( 'create_water_res_regions', 'weather_res_save_tax_metas', 10, 2
);
```