МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

С.Д. КУЗНІЧЕНКО, І.В. БУЧИНСЬКА

Картографічні WEB-сервіси

Конспект лекцій

Одеса
Одеський державний екологічний університет
2022

УДК 681.3
К89

Кузніченко С.Д., Бучинська І.В.
К89 Картографічні WEB-Сервіси: конспект лекцій. Одеса: Одеський державний екологічний університет, *2022*. 107
ISBN

Конспект лекцій з навчальної дисципліни "Картографічні WEB-Сервіси" для студентів 4 року навчання денної та заочної форм спеціальності 122 «Комп’ютерні науки».

В конспекті лекцій викладається інформація про нові засоби представлення та поширення просторової інформації, про програмні засоби та сучасні програми для створення WEB-картографічних застосунків. Ці знання є фундаментом для створення сучасних геоінформаційних систем різного призначення, без яких в наш час важко уявити будь-яке дослідження навколишнього середовища.

УДК 681.3

Рекомендовано методичною радою Одеського державного екологічного університету Міністерства освіти і науки України як конспект лекцій (протокол №__ від ____.____. ____ р.)

ISBN

ЗМІСТ

ВСТУП

Метою викладання дисципліни є формування у студентів загального уявлення про картографічні WEB-сервіси, як способу отримання і поширення просторової інформації, забезпечення засвоєння відомостей про теоретичні і практичні основи WEB-картографії; забезпечення формування умінь і навичок використання та створення картографічних WEB-сервісів для візуалізації картографічних даних.

У результаті вивчення дисципліни студенти повинні ***знати***:

– теоретичні та практичні основи створення картографічних WEB-сервісів; програмні засоби та сучасні програми для створення WEB-картографічних застосунків; особливості оформлення картографічних матеріалів для розміщення в мережі Інтернет.

вміти:

– використовувати нові засоби представлення та поширення просторової інформації; застосовувати програмні засоби та сучасні програми для створення WEB-картографічних застосунків; оформляти картографічні матеріали та розміщувати картографічні дані в мережі Інтернет.

Дисципліна «Картографічні WEB-сервіси» є варіативної дисципліною для освітньо-професійної програми «Комп'ютерні науки». Логічно і змістовно-методично дисципліна базується на компетенціях, сформованих при вивчені дисциплін: алгоритмізація та програмування, організація баз даних та знань, геоінформатика та ГІС.

1 ОСНОВИ WEB-КАРТОГРАФІЇ

1.1 Історія розвитку web-картографії та картографічних web-сервісів

Бурхливий розвиток Internet-технологій дозволяє використовувати нові засоби доставки просторової інформації. Впровадження їх в геоінформаційні технології сприяє розвитку web-картографії – комп'ютерної технології, призначеної для доставки просторових даних кінцевому користувачеві.

Основними завданнями web-картографії є:

1) візуалізація існуючої інформації – просторове представлення інформації;
2) полегшення роботи з просторовою інформацією в мережі Internet, пошук, прокладка маршрутів і інші послуги, засновані на місцезнаходження об'єктів (LBS – location based services).

Картографічні web-сервіси є результатом інтелектуальної діяльності великого числа фахівців в галузі картографії, програмуванні, проектуванні інформаційних систем, геоматики і постачальників даних дистанційного зондування Землі. Історія картографічних web-сервісів нерозривно пов'язана з розвитком програмно-апаратних засобів та комерційного програмного забезпечення, які можна застосувати для вирішення цілої низки завдань в різних галузях науки. Істотний вплив на розвиток картографічних web-сервісів надав розвиток web-технологій, в більш широкому сенсі найбільш перспективними і багатофункціональними сервісами на ринку цифрових технологій ставали загальнодоступні картографічні web-сервіси, які безпосередньо використовують у своїй роботі мережеві технології, для позначення таких сервісів в англомовній термінології широко вживається термін «Web mapping service».

Перша корпорація, що випустила програмне забезпечення в напрямку мережевої картографії, стала компанія Xerox, яка в 1993 році запустила web-картографічний сервіс Xerox PARC Map Viewer спільно з дослідницьким центром Пало-Альто, штат Каліфорнія. При роботі сервісу вперше була задіяна технологія відправлення запиту на сервер і отримання зображення в форматі GIF. Другим істотним ривком в сфері web-картографії стала поява з 1996 року інтерактивних картографічних web-систем. Інтерактивність картографічних сервісів передбачала спосіб взаємодії з кінцевим користувачем через призначений для користувача

інтерфейс і зв'язок графічних елементів на карті з базою даних, внаслідок чого на карті відображалися об'єкти з атрибутивною інформацією. На ринок інформаційних технологій з 1996 року по 1999 рік вийшли інтерактивні картографічні сервіси Mapquest, UMN MapServer, Geomedia WebMap 1.0, Terraserver USA. Вперше в 1996 році в картографічному сервісі Geomedia WebMap 1.0 була використана векторна графіка за допомогою компонента ActiveCGM (Computer Graphic Metafiles).

На ранніх етапах становлення web-картографії відмітною особливістю більшості сервісів була їх локальність і вузька тематична спрямованість, що серйозно обмежувало коло потенційних користувачів таких сервісів. Один з перших рішучих кроків по популяризації web-ГІС був зроблений в 1998 році у Великобританії, коли був запущений сайт (який успішно працює і до сих пір) www.streetmap.co.uk. Цей сервіс, на відміну від своїх попередників, ні орієнтований на візуалізацію локального ділянки земної поверхні і насичення її вузько тематичною інформацією. Навпаки, творці сервісу пішли іншим шляхом – вони надали найпростішу топографічну інформацію, але покрили всю територію Великобританії. Саме цей підхід і визначив шалену популярність сервісу: тисячі людей могли без особливих зусиль визначити місце розташування торгового центру, будинку і будь-якого іншого об'єкта, знаючи всього лише його поштовий індекс, а потім послати готову схему проїзду на друк.

Серед переліку програм, які зайняли в цей же час ринок інформаційних технологій, особливе положення і підтримку отримав картографічний web-сервіс UMN MapServer, розроблений Університетом Міннесоти. Національне космічне агентство США (NASA) профінансувало проект MapServer в 1994 році з метою підтримки web-доставки даних для лісового господарства. Пізніше в 1997 році проект виріс з необхідності надання даних дистанційного зондування по мережі тільки для фахівців лісового господарства в широко спектральний картографічний web-сервіс MapServer 1.0. На відміну від інших картографічних програм в 1999 році даний сервіс став першим проектом з відкритим вихідним кодом.

Паралельно в цей час в 1997 році розвивався інший картографічний проект, ініціатором якого стала Національна Геологічна Служба США (USGS). Рік по тому USGS запустила перший національний он-лайн атлас Сполучених Штатів Америки, в даний час, даний сервіс відомий як TerraServer USA. Істотний вплив на розробку даного сервісу і підтримку надали компанії Microsoft і Hewlett Packard (HP). Відмінною особливістю

такого спільного проекту стало використання технології протоколу WMS "Web Map Service", який був розроблений міжнародною некомерційною організацій Open GIS Consortium, що регламентує стандарти для всієї web-картографії.

Активна фаза в розвитку web-картографії наступила в 1999-2000 роках, на ринок індустрії інформаційних технологій вийшли програмні продукти від компанії ESRI і NASA. З цього часу починається епоха розподілених картографічних платформ. Так компанія ESRI випускає кілька видів програм для різних цілей, які включають в себе сервіси ESRI ArcIMS, ESRI Geography Network. Глобальні задачі ставить перед собою проект ESRI Geography Network (ESRI географія мережі), основним завданням якого є обмін і використання цифрової географічної інформації в Інтернеті, в якості додаткових параметрів передбачається створення на web-сервісі простих персональних карт.

На початку 2000 років визначною подією для web-картографії стає розробка одного з перших повністю тривимірного інтерактивного он-лайн віртуального глобуса. Національне космічне агентство США (NASA) розробляє відкритий віртуальний глобус, при написанні програмного коду була задіяна об'єктно-орієнтована мова програмування Java. Розробленому віртуальному глобусі була дана назва Nasa World Wind, і він працює і по теперішній час, став популярним у багатьох країнах. Сервіс Nasa World Wind володіє величезною колекцією супутникових знімків з високою роздільною здатністю, які зберігаються на сервері Nasa. Завдяки відкритому вихідному коду, програмний продукт можна модифікувати і розширювати. При роботі сервіс використовує специфікацію відкритої графічної бібліотеки (OpenGL – Open Graphics Library), яка містить більше 300 функцій для відображення складних тривимірних сцен.

Розвиток проектів з відкритим вихідним кодом, де кожен зареєстрований користувач може покращувати або вносити зміни в інтерфейс програми або карту, зумовило створення в 2004 році некомерційного веб-картографічного проекту Open Street Maps докладної, вільної і безкоштовної географічної карти світу. Засновник проекту – Стівен Кост, інженер з Великобританії.

Найбільш значущою датою в області web-картографії став 2005 рік. Компанія Google створює відразу два web-картографічних сервіси, які принципово відрізняються один від одного. Вихід у світ картографічного сервісу Google.Earth і Nasa World Wind знаменує собою початок розвитку

нової категорії в області web-картографії – он-лайн картографічних віртуальних глобусів. Програма Google.Earth володіє інтуїтивним інтерфейсом, великим набором різних функцій і поєднує в собі різні бази не тільки супутникових зображень, але і прив'язку фотографій місцевості яку вказує сам користувач. Програма представляє собою віртуальний глобус – модель всієї Земної кулі, дистрибутив програми необхідно встановлювати на комп'ютер користувача, все супутникові зображення завантажуються через глобальну мережу. Внаслідок розвитку Google.Earth з'являється можливість перегляду супутникових зображень, використовуючи часову динаміку завантаження знімків, що дає можливість аналізу територій за обраний певний період. Крім того, був використаний принципово новий підхід в організації самого сервісу: замість класичного підходу, в якому користувач надсилає запит на сервер, чекає обробки і отримує назад згенеровану «на льоту» картинку, всі дані були підготовлені і оброблені заздалегідь, що в поєднанні з технологіями AJAX, дозволило добитися незвичайно швидкою роботи з картами і «безшовності» даних при навігації.

Другий проект Google.Maps є картографічним сервісом, працюючим через браузер користувача, основне призначення якого є пошук і перегляд заданих географічних об'єктів. Паралельно, в цей же час вийшов перший реліз картографічного web-сервісу від компанії Microsoft під назвою Virtual Earth, надалі який був інтегрований в портал Live Search Maps, нині відомий як Bing.Maps.

У наступні роки було запущено кілька веб-картографічних проектів, які успішно позиціонували себе на ринку інформаційних послуг. Так в 2006 році був запущений проект Wikimapia – відкритий картографічний web-сервіс. Особливість сервісу – можливість перегляду супутникових зображень різних виробників, сервіс об'єднує в собі перегляд супутникових зображень різних компаній (Google, Яндекс, OSM). Завдяки принципу відкритого редагування, в сервісі Wikimapia, де зареєстровані користувачі самі додають і редагують інформацію, зокрема (контури доріг, межі територіальних об'єктів), на картографічний сервіс додано більше 25 мільйонів географічних об'єктів на 2015 рік. У 2007 році почав працювати картографічний сервіс Yahoo.Maps, який пропрацював до червня 2015 року. Також можна відмітити російський web-картографічний сервіс Яндекс.Карти, який володіє великим набором супутникових зображень з високою роздільною здатністю, постачальниками даних виступає компанія

NavteQ, сервіс взаємопов'язаний з пошуковою системою. Основне призначення сервісу – це пошук необхідної інформації на карті, використання для прокладки маршрутів і навігації, він був запущений в 2004 році.

Розвиток картографічних web-сервісів представлено в табл.1.1, де відображені в хронологічному порядку найважливіші технології, що вплинули на подальший розвиток web-сервісів і всієї web-картографії, рік початку введення в експлуатацію картографічного сервісу, і головний розробник сервісу.

Таблиця 1.1 – Розвиток картографічних web-сервісів

Назва	Рік запуску	Технологія	Розробник
Xerox PARC Map Viewer	1993	відправка запиту на сервер та отримання зображення у форматі GIF	корпорація Xerox
Geomedia WebMap 1.0	1996	векторна графіка за допомогою компонента ActiveCGM	фірма Intergraph
TerraServer USA	1997	використання протоколу WMS “Web Map Service”	Геологічна Служба США (USGS), Hewlett Packard, Microsoft
UMN MapServer	1999	використання відкритого вихідного коду	університет Міннесоти, Національне Космічне агентство США (NASA).
ESRI ArcIMS, ESRI Geography Network	2000	розробка розподілених картографічних сервісів за цільовим призначенням	компанія ESRI
Nasa World Wind	2003	технологія розроблення віртуального глобуса	Національне аероко-смічне агентство США(NASA)
Яндекс.Карты	2004	інтеграція інформаційно-довідкової платформи з картою	компанія Yandex
Google.Earth, Google.Maps	2005	глобальний картографічний сервер, використання AJAX	корпорація Google
Wikimapia	2006	відкритий інтерфейс API, для редагування важко-доступних територій	Олександр Корякін, Євген Савельєв

Історія картографічних web-сервісів налічує невеликий період тривалістю в 20 років. За даний період з 1993 року були розроблені незалежно один від одного більш 20 глобальних картографічних web-сервісів, при розробці яких використовувалися технології програмування, картографування, мережеві технології, дані дистанційного зондування Землі.

Працюючі на сьогоднішній день картографічні web-сервіси активно використовуються в таких областях як освіта, наука, природоохоронні відомства, прикладні дослідження, інформаційні послуги. Найбільш важливими тенденціями, які спостерігаються в останні півтора-два роки, є поява великої кількості безкоштовних проектів, що реалізують концепцію надання попередньо оброблених даних; збільшення можливостей персоніфікації сервісів; можливості по інтеграції власних даних з існуючими сервісами; глобальність сервісів; також спостерігається все більша інтеграція таких служб в повсякденне життя.

1.2 Класифікація web-картографічних сервісів і програм

Всі існуючі картографічні web-сервіси відносяться до розділу web-картографії. Класифікація картографічних геоінформаційних програм, які використовують технології web-картографії (рис. 1.1):

1) Картографічні web-сервіси.
2) Віртуальні глобуси.
3) Геоінформаційні програми.
4) Web-ГІС сервери і геопортали.

Картографічні web-сервіси є програмними засобами, заснованими на геоінформаційних технологіях. Фахівці в області геоінформаційних систем визначають геоінформаційні технології як сукупність програмно-технологічних засобів отримання нових видів інформації про навколишній світ. Геоінформаційні технології призначені для підвищення ефективності процесів управління, зберігання і подання інформації, обробки і підтримки прийняття рішень.

Сучасні картографічні web-сервіси класифікуються за функціональними можливостями, технологічним особливостям і інтерактивною взаємодією з кінцевим користувачем. Представлені в мережі сервіси за способом візуалізації даних діляться на віртуальні глобуси і картографічні сервіси. За технологічними особливостям на

кешувальні і динамічні. Існує також ряд спеціалізованих програмних продуктів, які використовують технології web-картографії, але відрізняються тим, що вимагають установки на комп'ютер користувача виконуваних файлів, мають ліцензію на використання і є комерційними продуктами з закритим вихідним кодом. У цю категорію потрапляють такі програми як ArcGIS, Map info, QGIS, gvSIG. Використання перерахованих програмних продуктів здійснюється для підготовки і редагування картографічних матеріалів і супутникових зображень перед опублікуванням їх в мережі Інтернет. Існує ще одна велика категорія програмних продуктів, що поєднує в собі web-технології і геоінформаційні технології, це картографічні Web-ГІС сервери: MapServer, GeoServer, OpenLayers, вони є інструментами для швидкої публікації географічних даних в мережі.

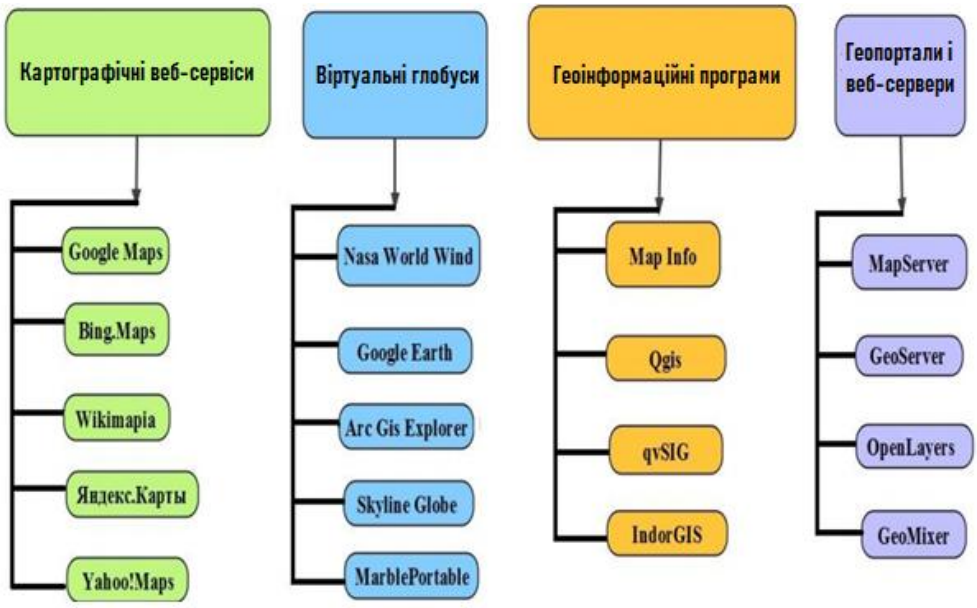

Рисунок 1.1 – Класифікація web-картографічних сервісов и програм

Технологічною особливістю категорії Web-серверів, є можливість створення інтерфейсу потрібної складності, в залежності від предметної галузі, наявність можливості інтеграції сервісу з базою даних, що підтримує класи просторових даних (PostgreSQL, SQL Server, MySQL, ArcSDE). Головна відмінність подібних систем від Google Maps – повний контроль над програмним забезпеченням і самими даними, однак за це доводиться розплачуватися більшою складністю установки і настройки,

часто вимагає хоча б початкового знання мов програмування (JavaScript, PHP) і основ адміністрування.

Принцип роботи картографічних web-сервісів заснований на використанні браузера, через який відображається географічна інформація. При такому способі взаємодії існує ряд технологічних особливостей, які впливають на відображення геопросторової інформації, зокрема швидкість відображення деталізованих знімків супутникової зйомки залежить від швидкості завантаження зображення розташованих на сервері зберігання даних, тому користувач повинен володіти високошвидкісним доступом в інтернет і хорошою пропускною спроможністю каналу зв'язку. У зв'язку з прямою залежністю відображення даних від типу і швидкості підключення до мережі інтернет в режимі он-лайн, виробники віртуальних глобусів і ряду інших web-гіс програм впровадили можливість кешування даних, в внаслідок чого завантаження даних потрібно тільки для отримання нових даних, перш завантажені зображення зберігаються в кеші програми. Даною особливістю володіє віртуальний глобус Google Earth і програма SAS.Planet.

1.2.1 Функціональні особливості віртуальних глобусів

Віртуальні глобуси – спеціальні програми, які встановлюються на комп'ютер користувача, є корисним прикладним інструментом для відображення просторової інформації (рис.1.2).

У віртуальних глобусах відомих виробників (Google, Nasa, Marble) присутні візуалізація і відображення тривимірних моделей наповнених атрибутивною інформацією. Незважаючи на те, що в мережі достатня кількість web-картографічних сервісів з супутниковими зображеннями, існують території, які в силу ряду певних обставин не відображаються з високим ступенем деталізації, також території можуть бути перекриті атмосферними явищами в момент зйомки, що ускладнює аналіз місцевості за різними критеріями і параметрам. Різнорідність картографічних матеріалів, якими володіють віртуальні глобуси, можуть доповнювати відсутню інформацію web-картографічних сервісів.

Рисунок 1.2 – Інтерфейс і візуалізація програми Nasa World Wind 1.4

Віртуальні глобуси, як правило, включають доступ за замовчуванням до якогось тайлового шару «підкладці» – базі даних, що є одночасно їх великим плюсом і не меншим мінусом, так як змінити цю підкладку в більшості випадків не можна. Також, як правило, цим інструментам властиві проблеми при роботі з великими обсягами даних користувача, налаштування, елементарним аналізом (обрізка, перетин шарів даних).

1.2.2 Функціональні особливості геоінформаційних програм і web-гіс серверів

Геоінформаційні програми, що використовують технології web-картографії є інструментами для обробки супутникових зображень, створення тематичних карт, звітів, щоденної роботи з оперативними даними і базами даних.

Особливістю використання геоінформаційних програм є підтримка поширених форматів даних, реляційних і просторових, форматів графічних даних. При обробці файлів зображень можна використовувати супутникові знімки, аерофотознімки, скановані карти і гібридні карти web-картографічних сервісів.

Структура географічних інформаційних програм являє собою користувальницький застосунок, розроблений на інформаціонно-логічних та об'єктно-орієнтованих мовах програмування, на відміну від web-картографічних сервісів вони володіють інструментами детальної обробки

зображень і здатністю витягувати і пов'язувати дані з системами управління базами даних (СКБД), як це показано на рис.1.3.

Географічні дані містять чотири інтегрованих компонента:

1) місце розташування;

2) властивості і характеристики;

3) просторові відносини;

4) час.

Геоінформаційні програми мають інструменти обробки географічних даних, якими не володіють веб-картографічні сервіси. Тому використання призначених для користувача геоінформаційних систем, таких як ArcMap, QGIS і MapInfo є необхідною дією для обробки і використання отриманих даних з веб-картографічних сервісів. Поділ на системи обробки геопросторової інформації і різнорідних картографічних матеріалів в мережі, зумовило створення інтерактивних web-геоінформаційних серверів або веб-ГІС серверів.

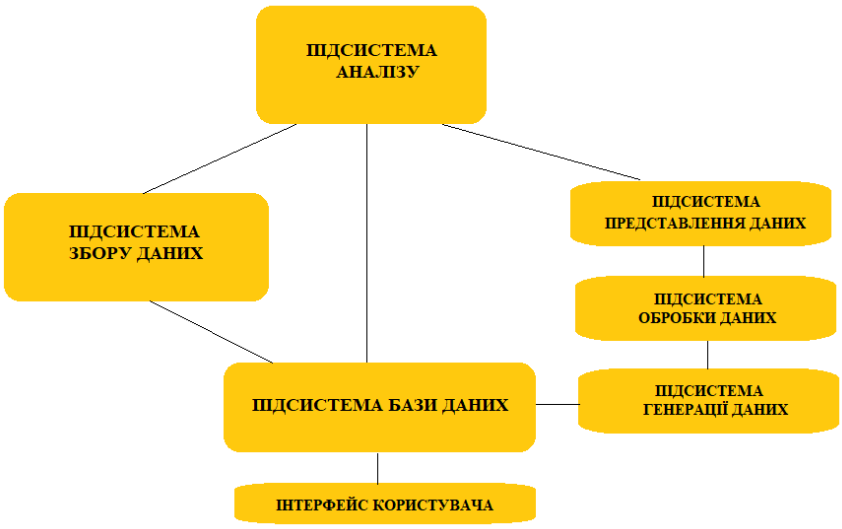

Рисунок 1.3 – Структура геоінформаційних програм

ГІС-сервером називається комплекс програмного забезпечення, призначеного для публікації геопросторових даних в локальних або глобальних мережах. ГІС-сервери – це складні інформаційні системи, вони є середовищем відображення геопросторових даних. На відміну від веб-картографічних сервісів, в яких відсутні механізми обробки супутникових зображень, ГІС-сервери надають способи для обробки, аналізу та персоналізації геопросторових даних. Мають спеціальний інтерфейс,

заснований на технології інтерфейсу прикладного програмування, відомого як API – (application programming interface). Одною з істотних специфікацій, якими володіють ГІС-сервери, це використання системних і апаратних ресурсів, розташованих не на комп'ютері користувача, а на сервері виробника, дана технологія спрощує взаємодію і розширює можливості тих категорій користувачів ГІС-серверів, у яких відсутня достатня продуктивність апаратних засобів комп'ютера для обробки великих обсягів геопросторової інформації. Взаємодія заснована на клієнт-серверних технологіях, користувач, використовуючи клієнтський додаток, надсилає запит на ГІС-сервер, в результаті обробки запиту ГІС-сервер відповідає користувачеві, при цьому повертаючи вже оброблені дані.

Алгоритм роботи з веб-ГІС сервером виглядає так, як показано на рис. 1.4.

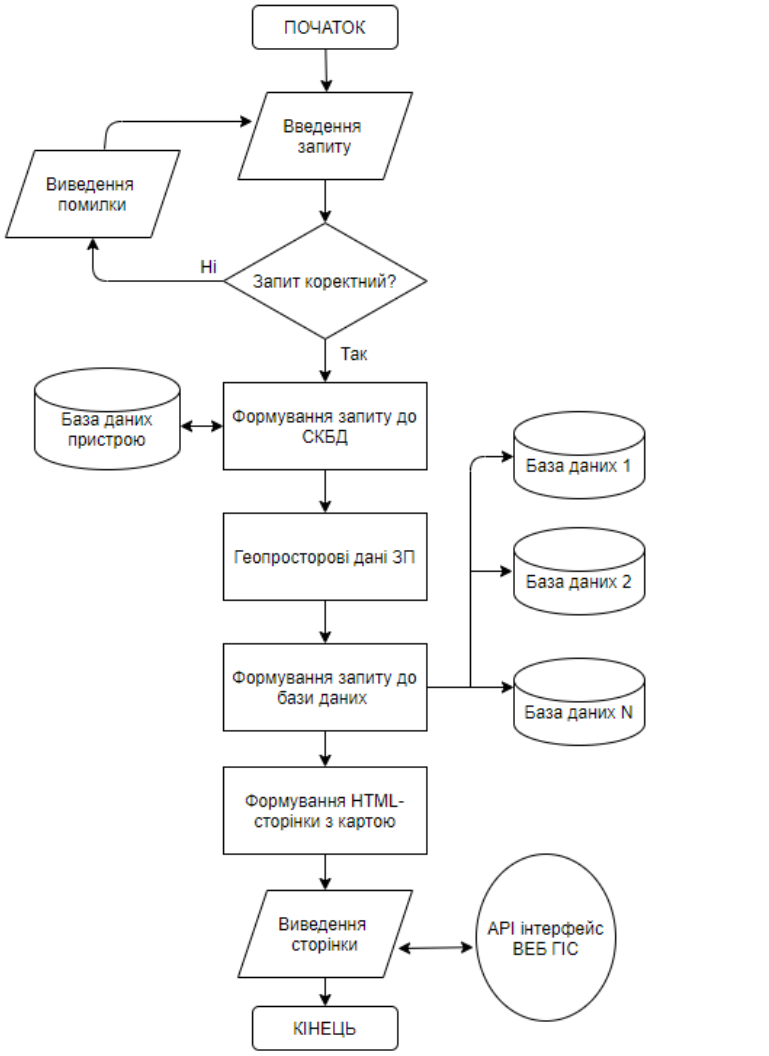

Рисунок 1.4 – Схема роботи веб-ГІС сервера при введенні запиту

1.2.3 Картографічні сервіси та геопортали

Картографічний сервіс (рис. 1.5) надає користувачеві можливість лише переглянути геодан через мережу і, в деяких випадках, провести нескладні маніпуляції з ними. Прикладами картографічних сервісів є Google Maps, Yandex-карти, сервіси, створювані ArcGIS for Server, Mapserver, і іншими варіантами серверних продуктів для роботи з ГІС.

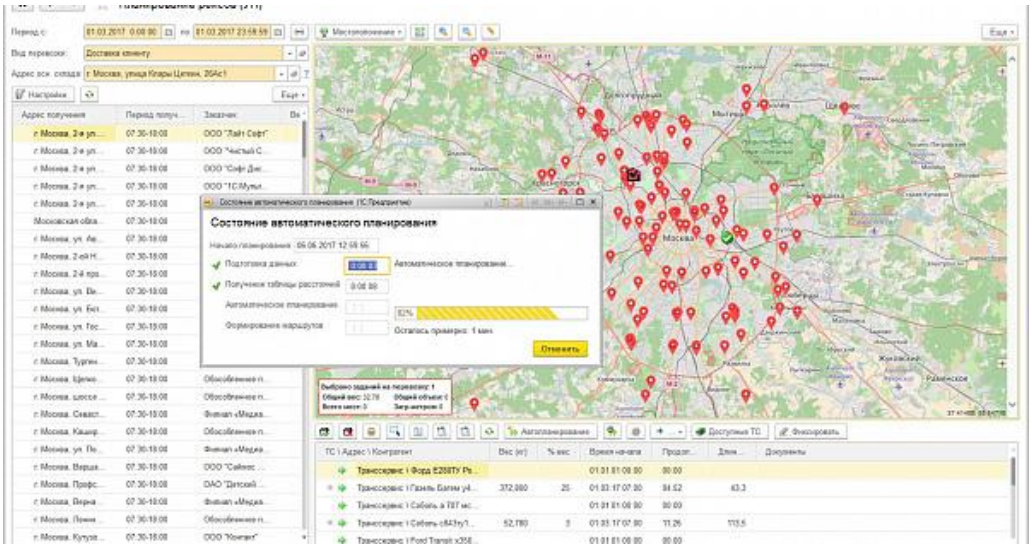

Рисунок 1.5 – Приклад картографічного сервісу

Функції геопортала значно ширше, ніж у картографічного сервісу. Геопорталом вважається web-портал, призначений для пошуку і доступу до геопросторової інформації та пов'язані з цим послуги (візуалізації, редагування, аналізу і так далі). Також геопорталом називається web-сайт або його еквівалент, перелік функцій якого, реалізованих у вигляді геосервісів (web-сервісів) включає в себе пошук наборів даних, їх візуалізацію, завантаження і трансформування, а також виклик інших сервісів. Іншими словами, геопортал є своєрідною надбудовою над картографічними сервісами, значно розширює їх можливості. Тип геопортала визначається набором функцій, які він здатний підтримати.

Картографічні сервіси хоча і мають в наявності функції ГІС, що забезпечують пошук об'єктів і маршрутів, але все ж ведеться пошук не інформаційних об'єктів з баз геоданих, а об'єктів реального світу за їх цифровими моделям.

Картографічні сервіси в більшості своїй є планами міст, картами-схемами територій (в тому числі і всієї земної кулі). Вони засновані на

даних дистанційного зондування Землі. У багатьох з них є створення, імпорт і експорт векторних просторових даних. В більшості картографічних сервісів підтримуються API (application programming interface - набір готових класів, процедур, функцій, що надаються сервісом, бібліотекою або додатком для використання його у зовнішніх програмних продуктах).

Геопорталів діляться на національні, регіональні та локальні. Найчастіше перші два типи геопорталів в останні роки більше нагадують національні і регіональні атласи, причому спектр їхніх можливостей значно ширше, ніж у їх паперових аналогів. Щодо до сфер застосування геопорталів, то вони найчастіше використовують в якості системи забезпечення прийняття управлінських рішень, кадастру і моніторингу просторових об'єктів, ГІС надзвичайних ситуацій, та ін. Розглянемо приклади геопорталів.

Геопортал адміністративно-територіального устрою України – інформаційний портал, на якому розміщені відомості про адміністративно-територіальний устрій України (рис.1.6). Портал містить: адміністративно-територіальні одиниці, топографію, населені пункти, автошляхи та залізниці, законодавчі дані про зміни адміністративно-територіального устрою, перспективні плани формування територій громад, бюджетні паспорти громад, дані про об’єднані територіальні громади, кількісні показники по навчальних закладах чи інших соціальних об’єктах, бюджету, розміру дотацій або субвенції тощо.

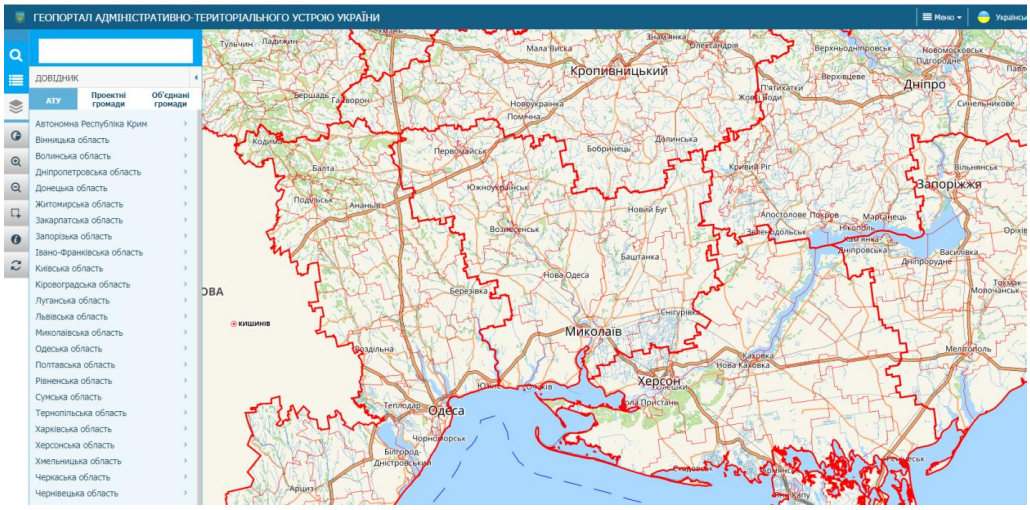

Рисунок 1.6 – Інтерфейс геопорталу адміністративно-територіального устрою України

Державна геодезична мережа України – інформаційний портал, на якому розміщені відомості про геодезичну інформацію та послуги по перетворенню та трансформуванню координат (рис.1.7).

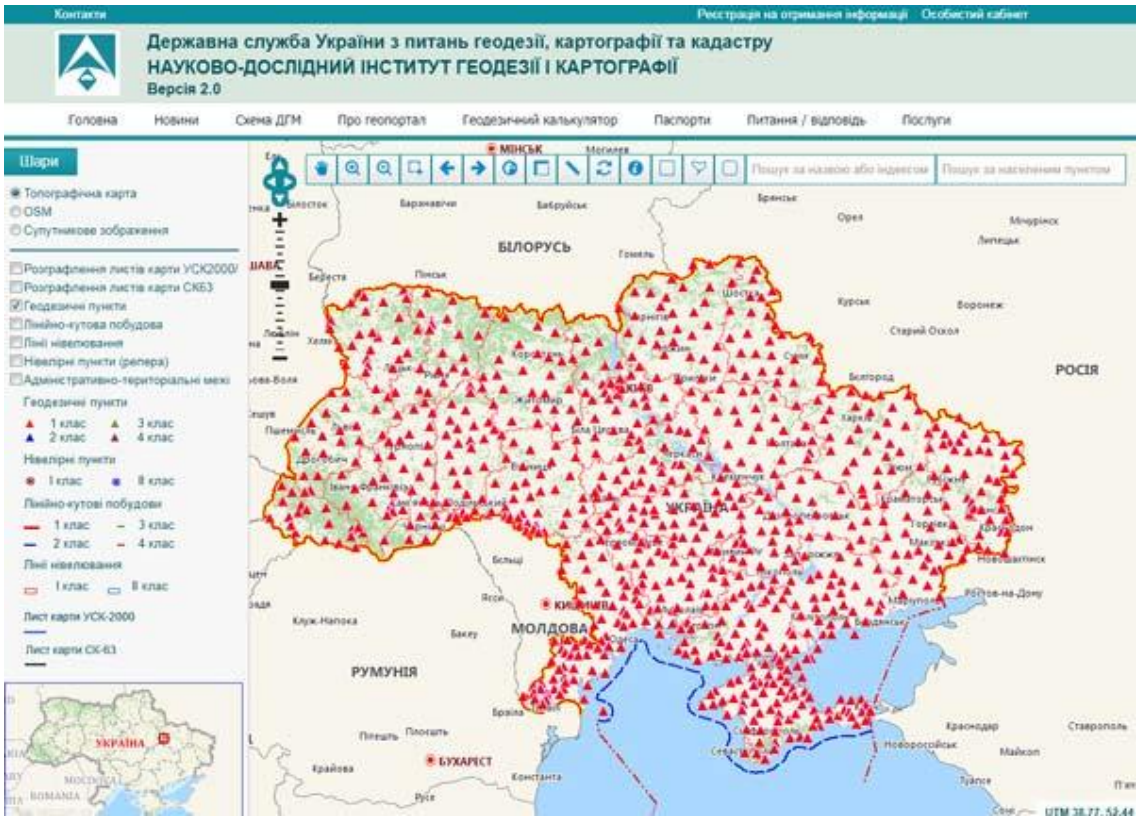

Рисунок 1.7 – Інтерфейс державної геодезичної мережі України

Серед геопорталів природних ресурсів України можна відмітити наступні національні геопортали:

Публічна кадастрова карта України (ПКК) **–** інформаційний портал, на якому розміщені відомості Державного земельного кадастру (рис.1.8).

Державна геологічна карта України – інформаційний портал, на якому розміщені відомості про результати геологічних, геофізичних, гідрогеологічних та інженерно-геологічних досліджень надр, моніторингу геологічного середовища і мінерально-сировинної бази.

Геопортал «Водні ресурси України» **–** інформаційний портал (рис.1.9), на якому розміщені відомості про дев’ять районів річкових

басейнів: річкові суббасейни, водогосподарські ділянки, екорегіони, кадастровий поділ тощо.

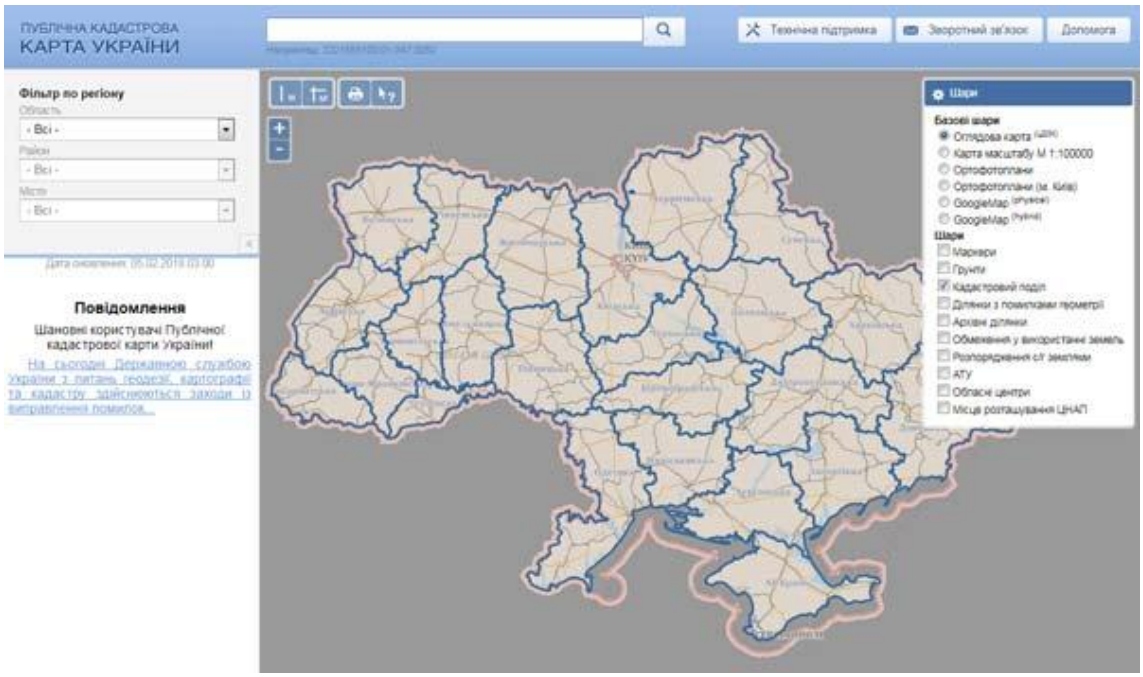

Рисунок 1.8 – Інтерфейс геопорталу Публічної кадастрової карти України

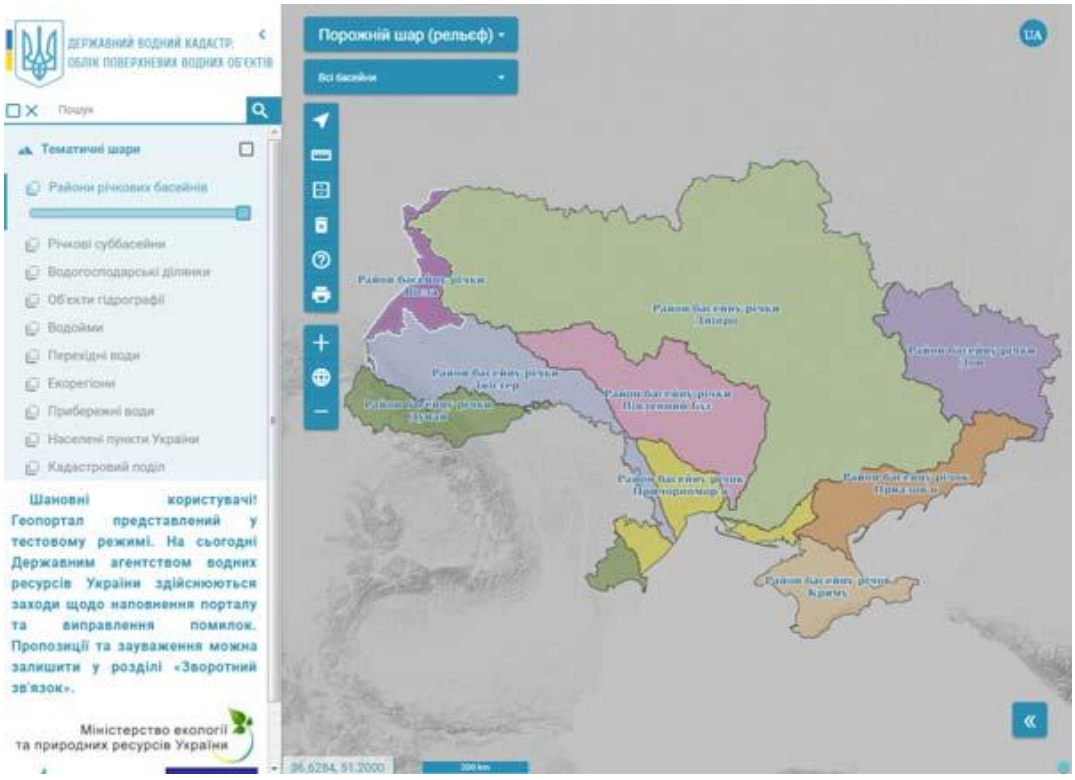

Рисунок 1.9 – Інтерфейс геопорталу «Водні ресурси України»

Геопортал «Лісовий фонд України» – інформаційний портал, на якому розміщені відомості з повидільною таксаційною інформацією для управління лісогосподарським виробництвом.

Також важливе місце серед національних геопорталів займають наступні геопортали екологічного моніторингу Землі:

Геопортал ЦПОСІ та КНП – інформаційний портал, на якому розміщені результати моніторингу території України за допомогою даних ДЗЗ (рис.1.10).

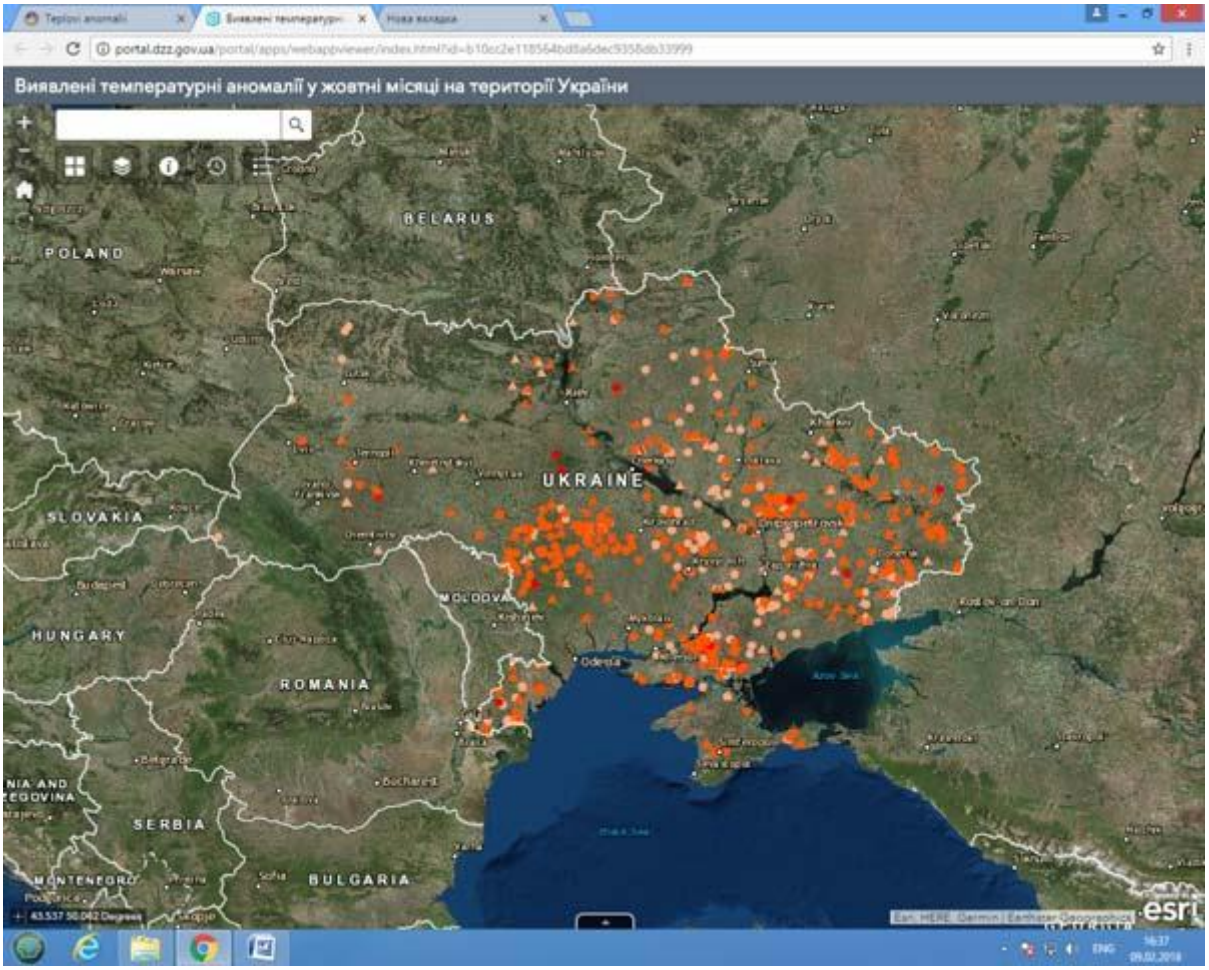

Рисунок 1.10 – Інтерфейс геопорталу дистанційного зондування Землі

Геопортал ДКАУ – інформаційний портал, на якому розміщені результати моніторингу території України за допомогою даних ДЗЗ (рис.1.11).

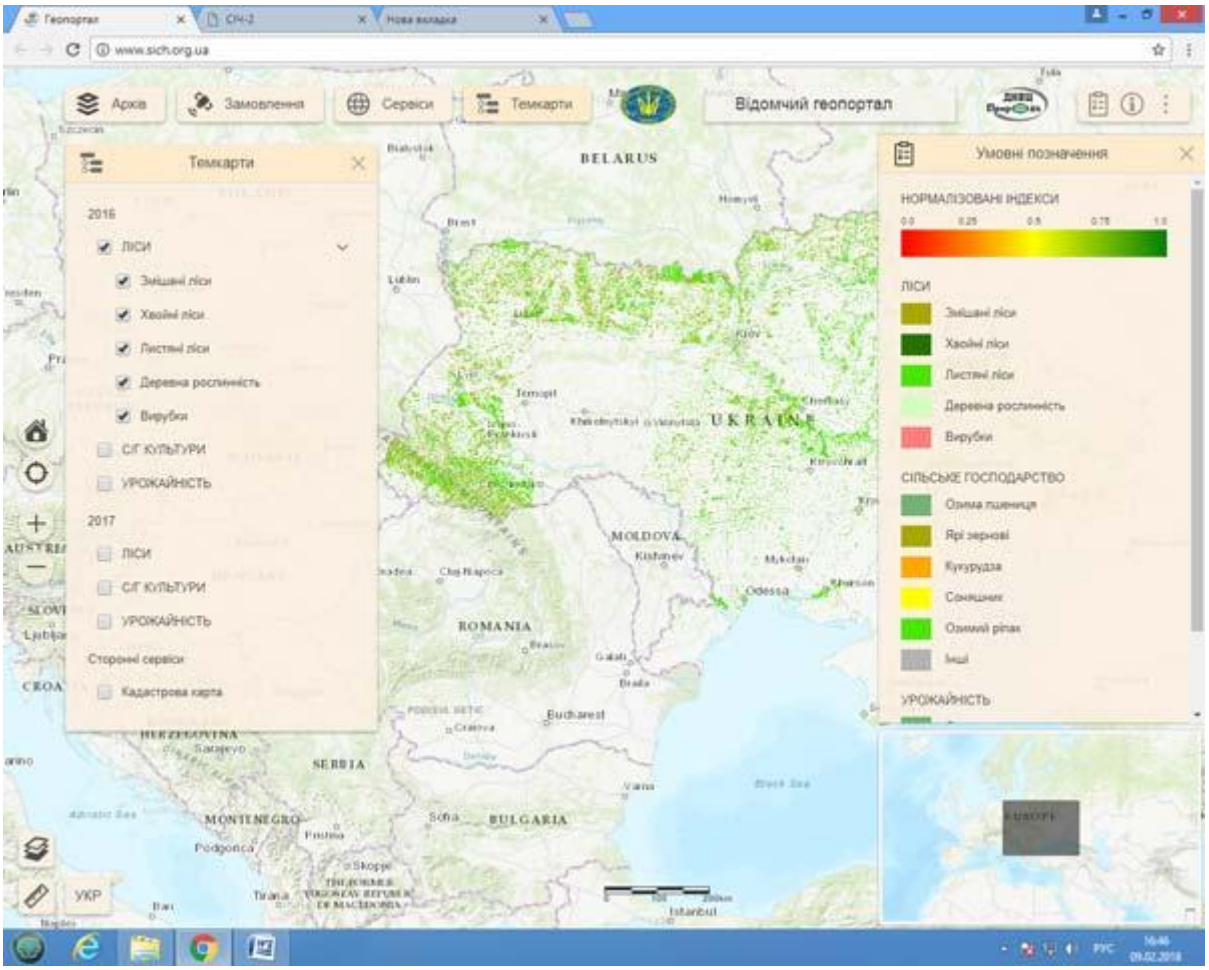

Рисунок 1.11 – Інтерфейс геопорталу ДКАУ

Наведемо також декілька прикладів геопорталів окремих областей, а саме:

Геопортал містобудівного кадастру Львівської обласної адміністрації – інформаційний портал, на якому розміщені містобудівні дані про райони (рис.1.12).

Геопортал містобудівного кадастру Одеської обласної адміністрації – інформаційний портал, на якому розміщені містобудівні дані про райони (рис.1.13).

Геопортал містобудівного кадастру Сумської області – інформаційний портал, на якому розміщені містобудівні дані про райони.

Геопортал містобудівного кадастру Чернівецької області – інформаційний портал, на якому розміщені містобудівні дані про райони (рис. 1.14).

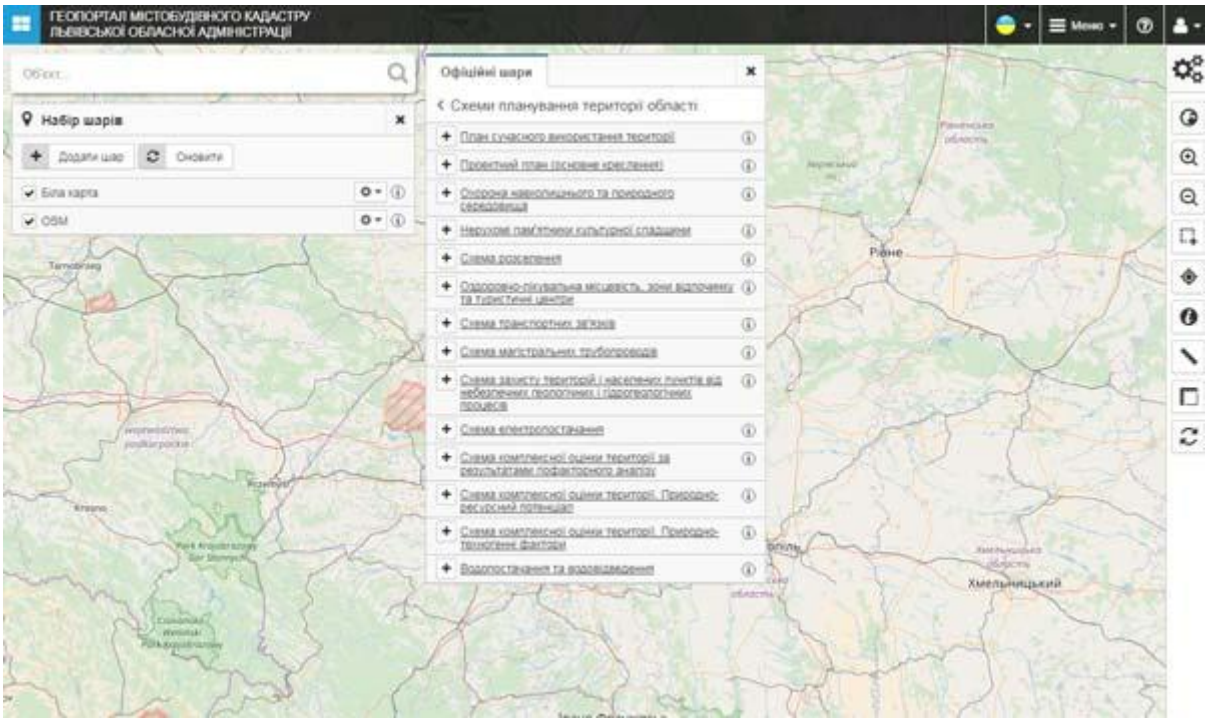

Рисунок 1.12 – Інтерфейс геопорталу містобудівного кадастру Львівської обласної адміністрації

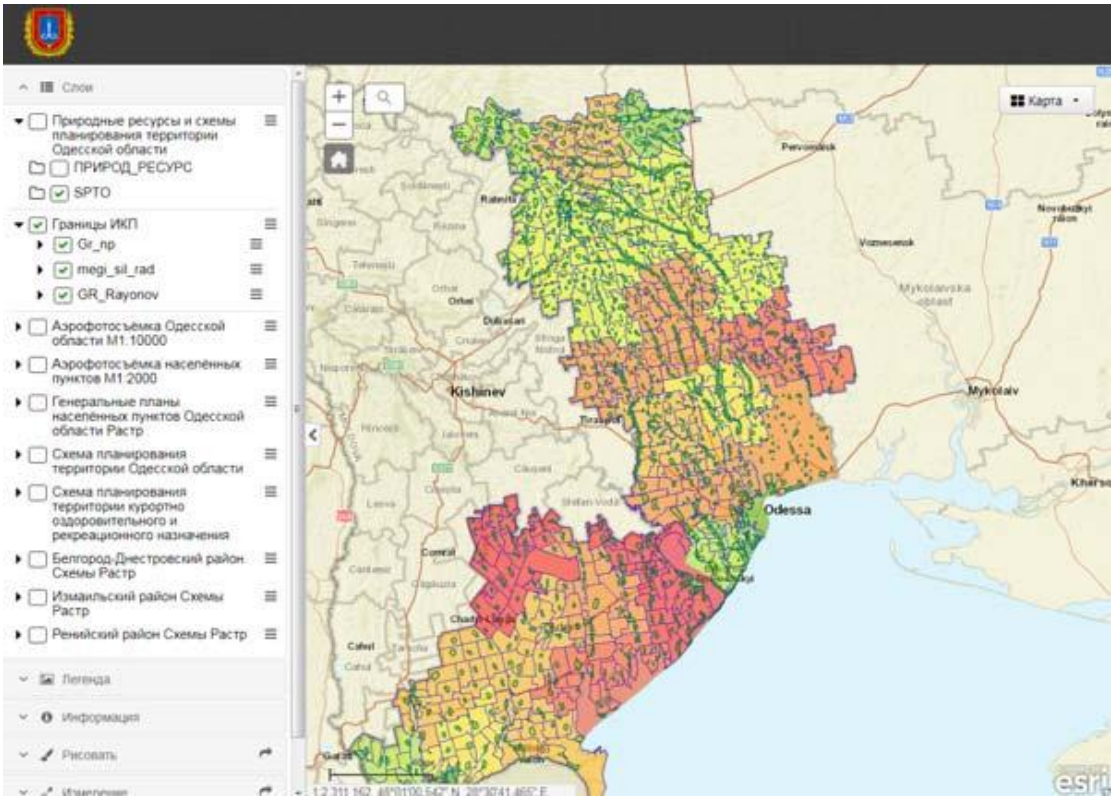

Рисунок 1.13 – Інтерфейс геопорталу містобудівного кадастру Одеської обласної адміністрації

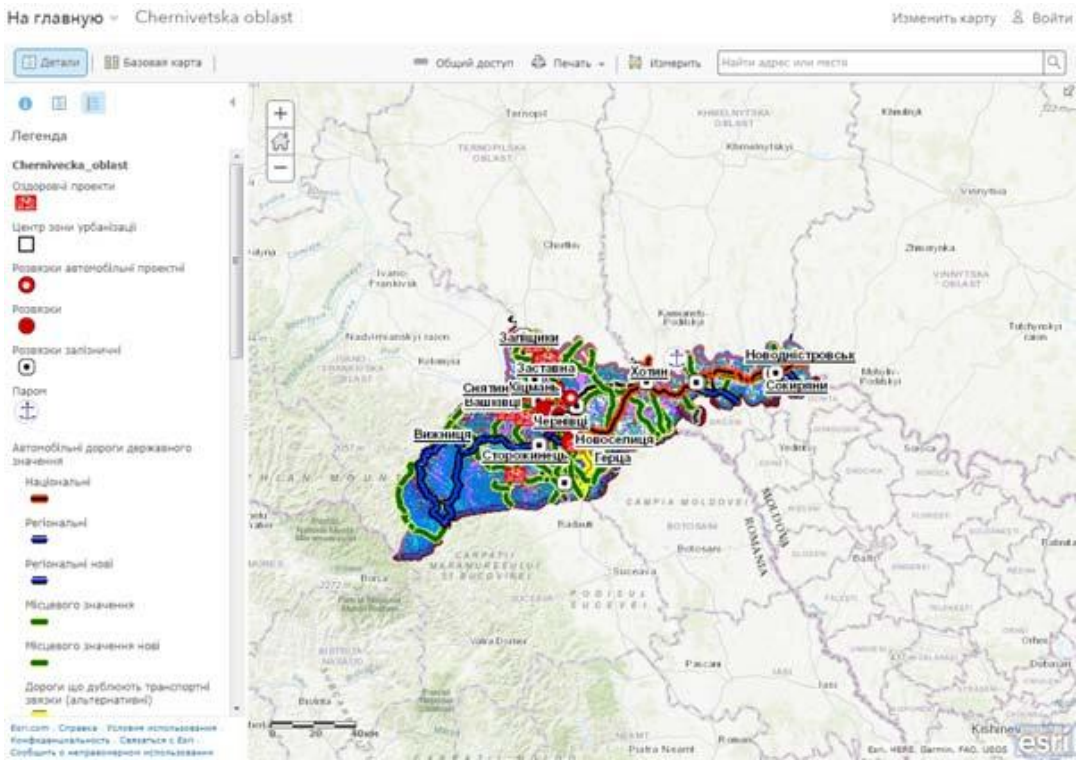

Рисунок 1.14 – Інтерфейс геопорталу містобудівного кадастру Чернівецької області

Серед геопорталів міст слід відмітити ***містобудівний кадастр*** – державну систему зберігання та використання геопросторових даних про територію, адміністративно-територіальні одиниці, екологічні, інженерно-геологічні умови, інформаційних ресурсів будівельних норм, державних стандартів і правил для задоволення інформаційних потреб у плануванні територій та будівництві, формування галузевої складової державних геоінформаційних ресурсів.

Система містобудівного кадастру включає:

1) організаційну структуру;
2) технічні та програмні засоби;
3) інформаційні ресурси;
4) каталоги та бази метаданих;
5) сервіси геопросторових даних;
6) будівельні норми, технічні регламенти та державні стандарти.

Містобудівний кадастр Києва – інформаційний портал, на якому розміщені відомості про містобудівну ситуацію (рис.1.15).

Геопортал Білоцерківської міської ради – інформаційний портал, на якому розміщені відомості про містобудівну ситуацію (рис.1.16).

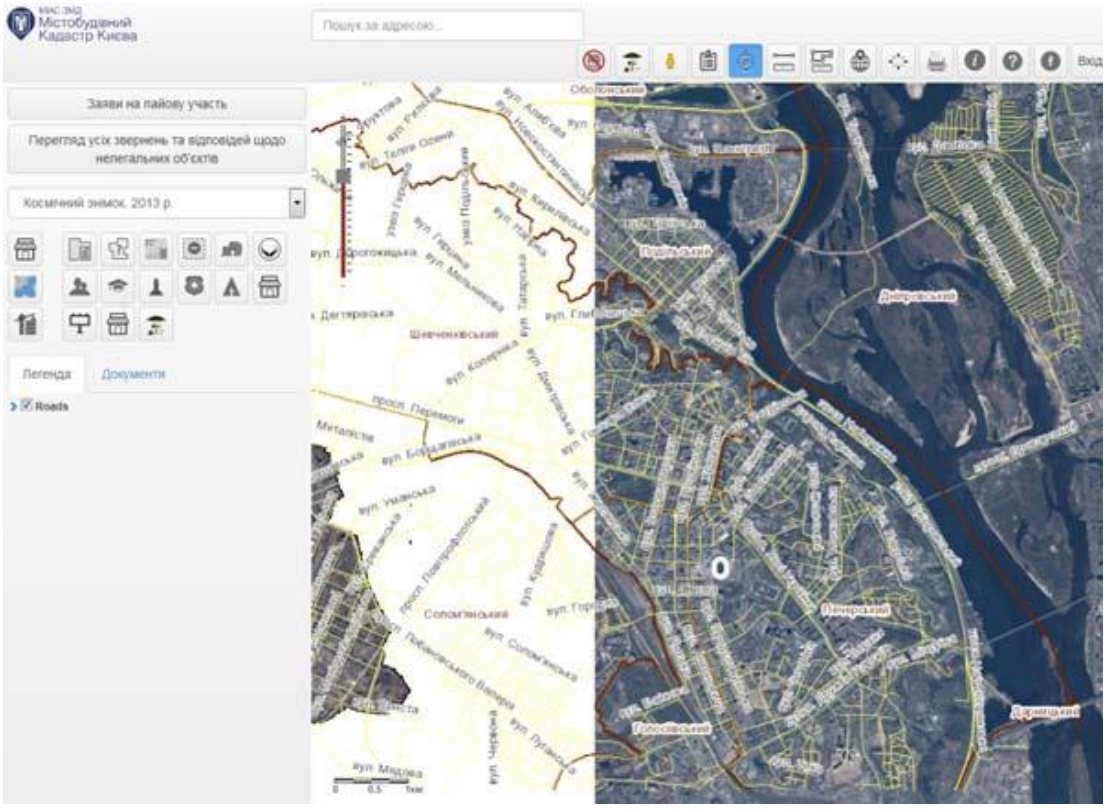

Рисунок 1.15 – Інтерфейс геопорталу містобудівного кадастру Києва

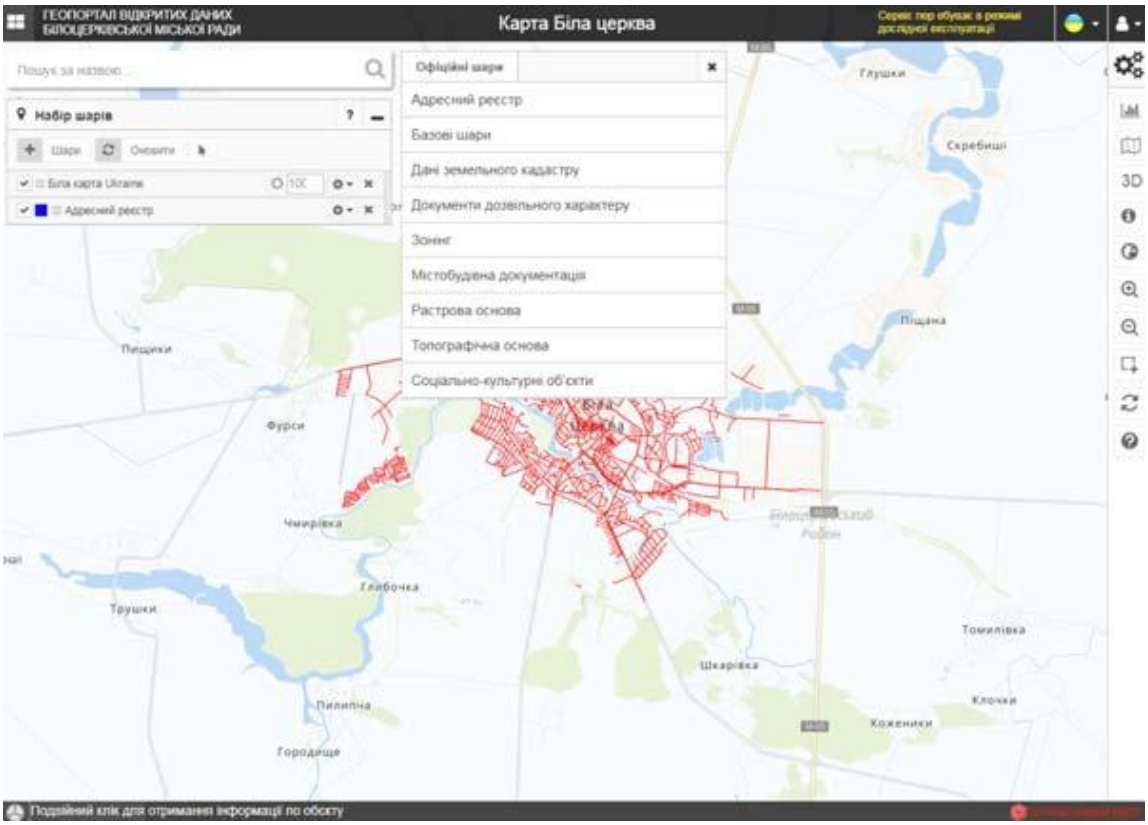

Рисунок 1.16 – Інтерфейс геопорталу Білоцерківської міської ради

Містобудівний кадастр м. Суми – інформаційний портал, на якому розміщені відомості про містобудівну ситуацію (рис.1.17).

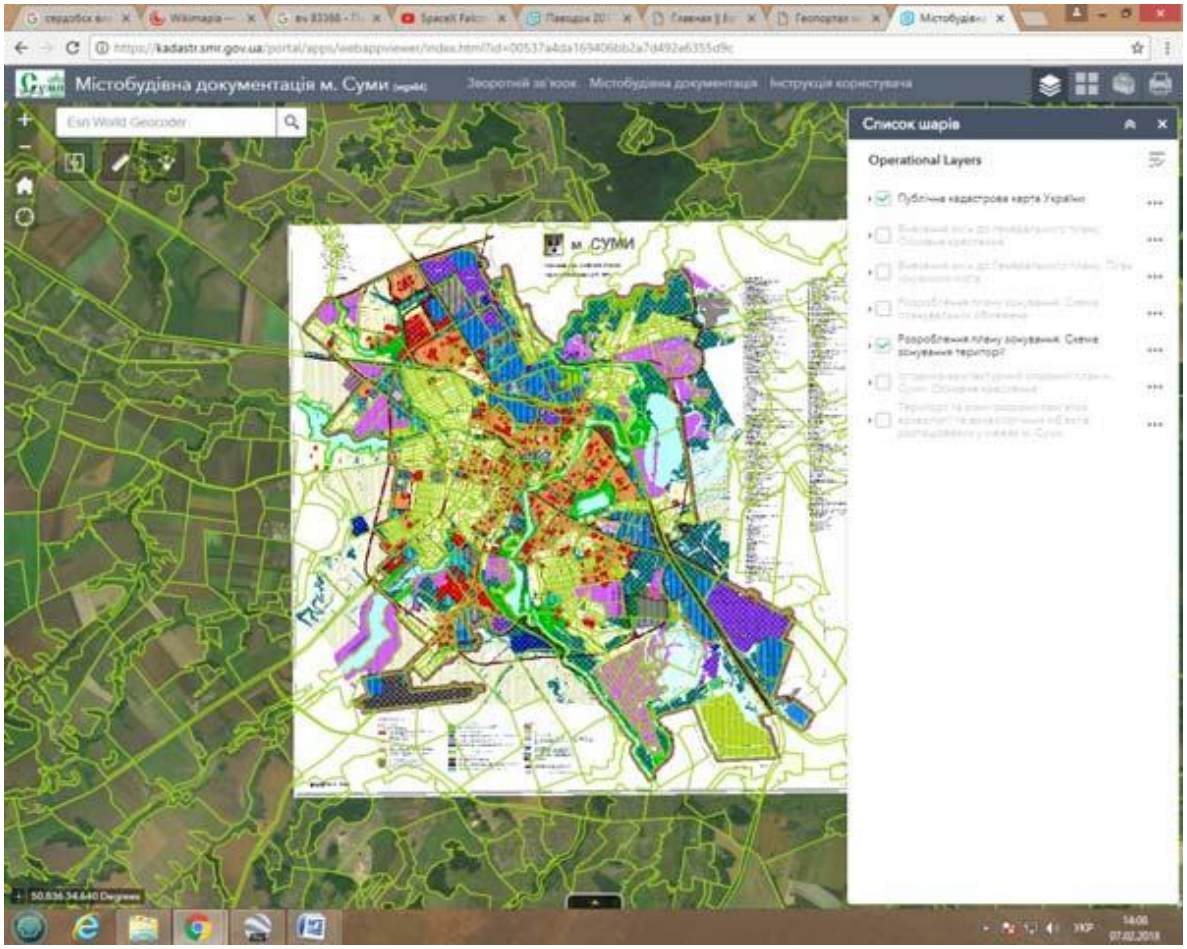

Рисунок 1.17 – Інтерфейс геопорталу містобудівного кадастру м. Суми

1.2.4 Стандарти в веб-картографії. Відкритий геопросторовий консорціум OGS

В геоінформаційному співтоваристві існує цілий ряд ключових організацій, які різними методами регулюють діяльність розробників. Найбільш зручним способом такого «контролю» в сучасному суспільстві стає впровадження і просування певних стандартів, протоколів і RFC розробки. На ринку web-ГІС присутні різні види організацій, найбільш активних представників яких можна класифікувати наступним чином:

1) ***Асоціації та регулюючі організації:*** OGC (Open Geospatial Consortium) – некомерційна організація, що займається підтримкою та просуванням стандартів і архітектур, пов'язаних з просторовими даними.

Членами консорціуму є найбільш значні компанії, чия діяльність пов'язана з просторовими даними. В стратегічну категорію членів організації входять USGS, NASA, NGA, головними членами є ESRI, Google, Microsoft та інші.

2) ***Opensource групи:*** OSGeo – також некомерційна організація, створена спеціально для підтримки проектів з відкритим кодом, як правило, підтримуються відкритими спільнотами фахівців. Проекти, що проходять інкубацію в OSGeo, отримують і місце в раді організації. Президентом організації є Frank Warmerdam, творець і один з основних авторів GDAL \ OGR.

3) ***Професійні ГІС:*** ESRI – корпорація, що спеціалізується на ГІС, і до недавніх пір не мала особливих конкурентів. Останнім часом ESRI активно намагається посилити свої позиції, що похитнулися на ринку web-картографії, розвиваючи ArcGIS Server. Незважаючи на наявність на ринку та інших гравців (Mapinfo, Autodesk), фактично є стандартом де-факто.

4) ***Інтернет-гіганти:*** Google і її компаньйони – група компаній (що включає також Microsoft, Yahoo і Yandex), яка розглядає web-картографічні проекти як один із способів розміщення реклами та активно розвивають онлайн присутність. В основному популярність досягається за рахунок надання широкого кола користувачів доступу до раніше недоступних баз даних космічної зйомки високого дозволу і супутніх технологій маршрутизації і пошуку.

5) ***Генератори даних:*** постачальники просторових даних, як правило, комерційних, наприклад цифрової картографічної інформації (Navteq / Teleatlas), супутникових даних (GeoEye, DigitalGlobe). В останній час в цьому секторі з'являються і некомерційні учасники (OpenStreetMap).

В даний час загальні принципи і стандарти в області розробки програмного забезпечення, що надає картографічні веб-сервіси, розробляються і декларуються міжнародною некомерційною організацією Open GIS consortium (OGC). OGC була заснована 25 вересня 1994 року і на момент створення включала тільки 8 членів. З 1992 по 2004 роки їх число зросло з 8 до 250, і на сьогоднішній день в OGS представлені найбільші комерційні, академічні та державні організації, що займаються розробкою або дослідженнями в галузі розвитку та розробки геоінформаційного або IT ПЗ (в тому числі такі найбільші корпорації як Boeing, Oracle, ESRI, MapInfo, Intergraph, Google (членство з весни 2006 року) і багато інших).

Багато в чому діяльність OGC в області геоінформаційних систем можна порівняти з діяльністю W3C по стандартизації процесів і технологій у всесвітній мережі. Так, однією з перших розробок OGC були стандарти по створенню GML – Geography Markup Language – мови групи XML, призначеної для опису географічно прив'язаних об'єктів. Варто відзначити, що GML може бути використаний і як мова моделювання, і як мову передачі просторової інформації в мережі.

В рамках даної теми доцільно розглянути поняття «інтероперабельність», яке відображає суть та завдання розробки стандартів веб-картографії.

Інтероперабельність (англ. Interoperability – здатність до взаємодії) означає можливість створення систем з довільних неоднорідних, розподілених компонентів на базі уніфікованих інтерфейсів або протоколів. Простіше кажучи, інтероперабельність – це здатність продуктів, систем або бізнес процедур працювати разом для виконання спільного завдання. Термін може бути визначений технічним способом чи більш широко з врахуванням соціальних, політичних та організаційних факторів. Зосередимося лише на технічній стороні взаємодії. Тоді інтероперабельність можна визначити як можливість обміну даними, запуску програм або передачі даних між різними функціональними блоками таким чином, щоб користувач мало або зовсім не знав унікальних характеристик цих блоків.

OGC визначає такі стратегічні цілі:

1) Надавати ринку безкоштовні і відкрито доступні стандарти, відчутну цінність для членів і вимірні вигоди для користувачів.

2) Лідирувати у всьому світі у створенні і встановленні стандартів, які дозволяють легко інтегрувати геопросторовий контент і послуги в бізнес і суспільні процеси, просторову мережу і корпоративні обчислення.

3) Сприяти впровадженню відкритих еталонних архітектур з просторовою підтримкою в корпоративних середовищах по всьому світу.

4) Просувати стандарти на підтримку формування нових та інноваційних ринків і додатків для геопросторових технологій.

5) Прискорювати освоєння ринком досліджень інтероперабельності за допомогою спільних процесів консорціуму.

Найважливішою діяльністю OGC є розробка специфікацій, які можна розділити на дві групи:

• **Абстрактні специфікації** (Abstract Specifications, AS) забезпечують концептуальну основу для більшості заходів з розробки специфікацій OGC. Відкриті інтерфейси та протоколи будуються та посилаються на абстрактну специфікацію, що забезпечується взаємодію між різними торговими марками та різними видами просторових систем обробки. Абстрактна специфікація надає еталонну модель для розробки OpenGIS специфікацій реалізації (OGC).

Абстрактна специфікація AS розбита на кілька тем (Topic), що покривають різні загальні питання, концепції і принципи геоінформатики. Повний їх список і короткі описи можна подивитися на сайті консорціуму OGC http://www.opengeospatial.org.

• **Специфікації реалізації** (Implementation Specifications, IS) розроблені для більш технічної аудиторії та детально визначають структуру інтерфейсу між програмними компонентами.

OGC має тісний взаємозв'язок з ISO / TC211[1)]. Абстрактна специфікація OGC поступово замінюється томами із серії ISO 19100, що розробляється цим комітетом. Наприклад, Служба OGC стандарт Web Map Service, GML та ін. вже є стандартами ISO.

Специфікації OGC пропонують наступні типи картографічних web-сервісов.

Web Map Service

a) визначає параметри запиту та надання картографічної (просторової) інформації у вигляді графічного зображення або набору об'єктів;
b) описує умови отримання та надання інформації про вміст карти (наприклад, властивості об'єкта в певному місці на карті);
c) характеризує умови отримання та надання інформації про можливості сервера за поданням різних типів картографічної інформації.

Web Feature Service

a) визначає умови отримання та оновлення просторово прив'язаної інформації клієнтської частиною програми з використанням Geography Markup Language (GML);
b) описує стандартний інтерфейс доступу і маніпуляції з географічними об'єктами за допомогою HTTP-протоколу.

[1)] ISO / TC 211 - це стандартний технічний комітет, сформований в рамках Міжнародної організації зі стандартизації (ISO), завданням якого є охоплення областей цифрової географічної інформації та геомантики. Він відповідає за підготовку серії міжнародних стандартів та технічних специфікацій нумерується в діапазоні, починаючи з 19101.

Web Coverage Service

a) розширює можливості WMS для надання растрової географічної інформації;
b) на відміну від WMS, coverage service розробляється для подання властивостей і значень в кожній конкретній точці географічного простору (а не для створення готових зображень), а також дозволяє проводити інтерпретацію даних не на сервері, а на клієнтської частини програми.

Однак зростання популярності картографічних web-сервісів породжує все більше число різних модифікацій існуючих мов і стандартів передачі просторових даних. Мабуть, уже в найближчий майбутньому OGC доведеться включити в сферу своїх інтересів розгляд спеціалізованих мов програмування, форматів передачі даних і стандартів, що їх описують.

Web Processing Service

Ця специфікація описує правила для вхідних і вихідних даних (запитів і відповідей на них) для сервісів геопроцесінга (геообробки), таких, як перетин полігонів та ін. WPS визначає веб-інтерфейси доступу до сервісу геопроцесінга. Геопроцесінг може включати будь-який алгоритм, розрахунок або модель, які оперують векторними або растровими даними з просторовою прив'язкою.

Web Map Tile Service

Ця специфікація розширює можливості WMS для створення кешованих картографічних сервісів.

Крім перерахованих відкритих картографічних стандартів, існують ще й інші, рідко використовувані. Наприклад, Grid Coverage Service (розширення WCS на сіткові дані), Sensor Planning Service, Web Coverage Processing Service (сервіс геообработки растрових даних), Sensor Observation Service (сервіс даних з датчиків – наприклад, даних про температуру) та інші.

1.2.5 Картографічний веб-сервіс Web Map Service (WMS)

Картографічний веб-сервіс (WMS) – це стандарт для візуалізації геопросторових даних через Інтернет. WMS дозволяє отримувати інформацію про карту через Інтернет або публікувати шари карти з персональної ГІС або системи обробки зображень в Інтернеті. У цьому

розділі розглянемо лише споживання картографічної інформації через Інтернет.

Операції WMS можна викликати за допомогою стандартного веб-браузера, відправляючи запити у формі URL-адреси. Зміст таких URL-адрес залежить від запитуваної операції. Тому при запиті карти URL-адреса вказує, яка інформація повинна відображатися на карті, яка частина Землі повинна бути нанесена на карту, бажана система координат, ширина і висота вихідного зображення, формат зображення та ін.

Веб-картографічний сервіс (WMS) забезпечує уніфікований доступ веб-клієнтів до карт, що відображаються картографічними серверами в Інтернеті. Таким чином, WMS:

– забезпечує динамічну побудову карти як зображення, так і серії графічних елементів як упакований набір даних географічних об'єктів;

– дає відповіді на основні питання про зміст карти;

– може інформувати інші програми про карти, які вони можуть створювати, і про те, які з них можуть бути запитані додатково.

WMS підтримує створення і відображення зареєстрованих і накладених карт, які надходять одночасно з декількох джерел як віддалених, так і різнорідних. WMS повертає растрове зображення, що є готовим для відображення.

Коли клієнтське та серверне програмне забезпечення реалізує WMS, будь-який клієнт може отримати доступ до карт з будь-якого сервера. Будь-який клієнт може комбінувати карти з одного або декількох серверів. Будь-який клієнт може запитувати інформацію з карти, що надається будь-яким сервером. Хоча програмістам для написання специфікацій потрібно писати код, кінцеві користувачі можуть скористатися перевагами продуктів, що містять їх, для публікації та доступу до геопросторової інформації. Користувачі програмного забезпечення можуть вибрати найкраще рішення для своїх потреб і не турбуватися про те, чи буде воно працювати з іншими рішеннями; якщо всі вони впроваджують один і той же стандарт WMS (рис. 1.18).

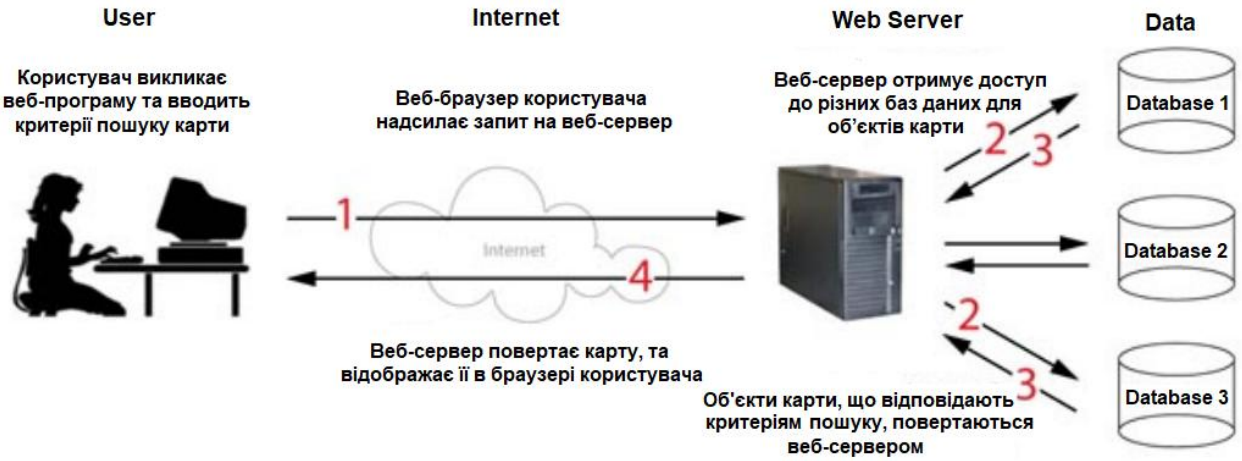

Рисунок 1.18 – Як працює Web Map Service

Зокрема, WMS визначає такі операції:

1) отримання та надання інформації про те, які типи карт може надавати сервер (GetCapabilities);

2) формат запиту на надання карти в растровому чи векторному форматах (GetMap);

3) отримання та надання інформації про контент карти, наприклад, про значення об'єкта в певному місці (GetFeatureInfo).

Файл GetCapabilities лежить на сервері, а тому запит GetCapabilities закінчується на веб-сервері (веб-сервер надсилає файл користувачеві). З іншого боку, запит GetMap повертає дані, які зберігаються в базах даних, тому веб-сервер повинен зв'язатися з базами даних і витягти запитувані дані. На рис.1.19 показано охоплення запитів GetCapabilities та GetMap.

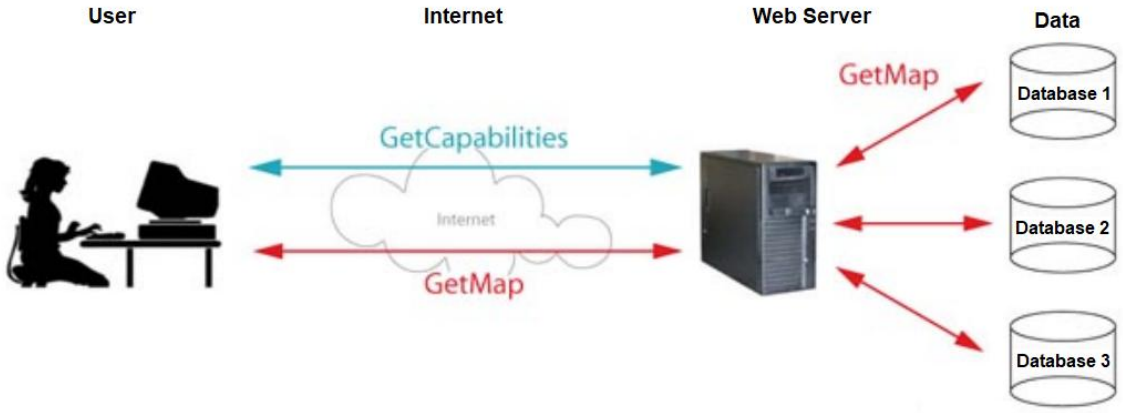

Рисунок 1.19 – GetCapabilities та GetMap запити

Операція GetMap повертає карту. Після отримання запиту GeetMap WMS повинен або задовольнити запит, або видати службовий виняток. У наступному рядку показано запит GetMap (URL-адреса – це один рядок, розбитий тут для читабельності):

http://sampleserver1.arcgisonline.com/ArcGIS/services/Specialty/ESRI_StatesCitiesRivers_USA/MapServer/WMSServer?version=1.3.0&request=GetMap&CRS=CRS:84&bbox=-178.217598,18.924782,-66.969271,71.406235&width=760&height=360&layers=0&styles=default&format=image/png

Результатом запиту GetMap є растрове зображення наведене на рис. 1.20.

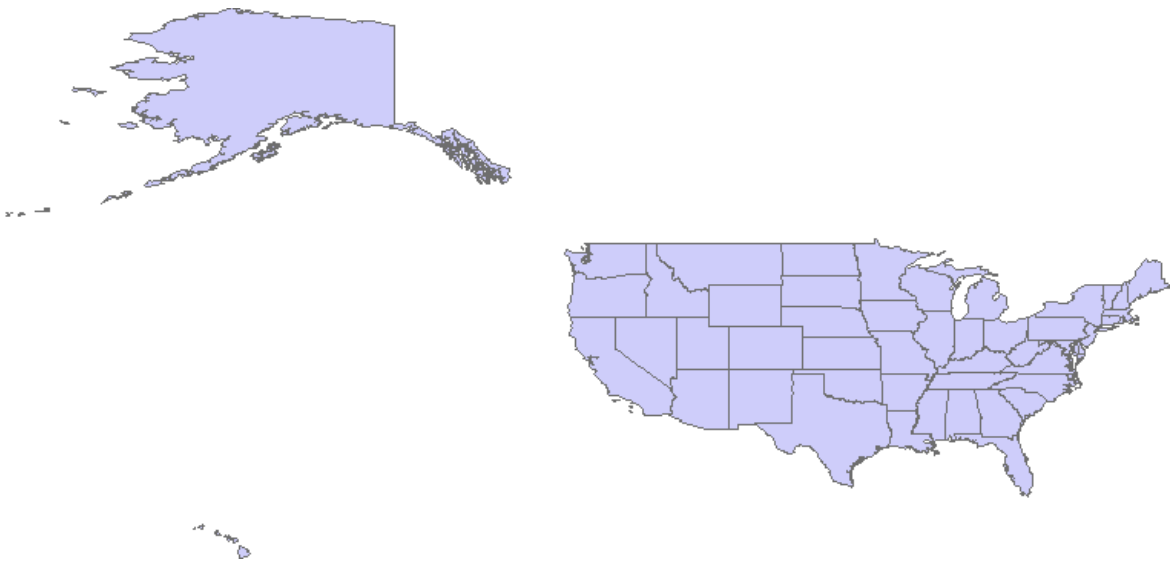

Рисунок 1.20 – Результат GetMap запиту

Параметри "http://sampleserver1.arcgisonline.com/ArcGIS/services/Specialty/ESRI_StatesCitiesRivers_USA/MapServer/WMSServer?" вказують місце розташування сервера карт. Параметри запиту GetMap слідують після визначення місця розташування сервера. Запит GetMap складається з наступних обов’язкових параметрів

Метою обов'язкової операції GetCapabilities є отримання метаданих служби, які є машиночитаними (і читаються людиною) опис інформаційного вмісту сервера та допустимі значення параметрів запиту. У наступному рядку показано запит GetCapabilities.

http://sampleserver1.arcgisonline.com/ArcGIS/services/Specialty/ESRI_StatesCitiesRivers_USA/MapServer/WMSServer?version=1.3.0&request=GetCapabilities&service=WMS

Параметри запиту	Опис
VERSION=version	Версія запиту
REQUEST=GetMap	Ім’я запиту
LAYERS=layer_list	Розділений комами список одного або декількох шарів карти
STYLES=style_list	Розділений комами список одного стилю візуалізації для кожного запитуваного шару
CRS=namespace:identifier	Система відліку координат
BBOX=minx,miny,maxx,maxy	Обмежувальні кути блоку (внизу ліворуч, вгорі праворуч) у одиницях виміру CRS (системи координат)
WIDTH=output_width	Ширина в пикселях зображення карти
HEIGHT=output_height	Висота в пикселях зображення карти
FORMAT=output_format	Вихідний формат карти

Параметри "`http://sampleserver1.arcgisonline.com/ArcGIS/services/Specialty/ESRI_StatesCitiesRivers_USA/MapServer/WMSServer?`" знову вказують місце розташування сервера карт.

Запит GetCapabilities складається з наступних двох обов’язкових параметрів запиту

Параметри запиту	Опис
SERVICE=wms	Тип сервісу
REQUEST=GetCapabilities	Ім’я запиту

Результатом GetCapabilities є XML-файл, що містить згадані метадані служби.

```
<WMS_Capabilities xmlns="http://www.opengis.net/wms" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-
instance" xmlns:esri_wms="http://www.esri.com/wms" version="1.3.0" xsi:sche
maLocation="http://www.opengis.net/wms
http://schemas.opengis.net/wms/1.3.0/capabilities_1_3_0.xsd
http://www.esri.com/wms
http://sampleserver1b.arcgisonline.com/ArcGIS/services/Specialty/ESRI_State
sCitiesRivers_USA/MapServer/WMSServer?version=1.3.0&service=WMS&request=Get
SchemaExtension">
...
<GetMap>
<Format>image/bmp</Format>
```

```
<Format>image/jpeg</Format>
<Format>image/tiff</Format>
<Format>image/png</Format>
<Format>image/png8</Format>
<Format>image/png24</Format>
<Format>image/png32</Format>
<Format>image/gif</Format>
<Format>image/svg+xml</Format>
<DCPType>
<HTTP>
<Get>
<OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simp
le" xlink:href="http://sampleserver1b.arcgisonline.com/ArcGIS/services/Spec
ialty/ESRI_StatesCitiesRivers_USA/MapServer/WMSServer"/>
</Get>
</HTTP>
</DCPType>
</GetMap>
...
<Layer>
<Title>
<![CDATA[ Layers ]]>
</Title>
<CRS>CRS:84</CRS>
<CRS>EPSG:4326</CRS>
<EX_GeographicBoundingBox>
<westBoundLongitude>-125.192865</westBoundLongitude>
<eastBoundLongitude>-66.105824</eastBoundLongitude>
<southBoundLatitude>19.416377</southBoundLatitude>
<northBoundLatitude>54.318281</northBoundLatitude>
</EX_GeographicBoundingBox>
<BoundingBox CRS="CRS:84" minx="-125.192865" miny="19.416377" maxx="-
66.105824" maxy="54.318281"/>
<BoundingBox CRS="EPSG:4326" minx="19.416377" miny="-
125.192865" maxx="54.318281" maxy="-66.105824"/>
<Layer queryable="1">
<Name>0</Name>
<Title>
<![CDATA[ States ]]>
</Title>
<Abstract>
<![CDATA[ States ]]>
</Abstract>
<CRS>CRS:84</CRS>
<CRS>EPSG:4326</CRS>
<EX_GeographicBoundingBox>
<westBoundLongitude>-178.217598</westBoundLongitude>
<eastBoundLongitude>-66.969271</eastBoundLongitude>
<southBoundLatitude>18.924782</southBoundLatitude>
<northBoundLatitude>71.406235</northBoundLatitude>
</EX_GeographicBoundingBox>
<BoundingBox CRS="CRS:84" minx="-178.217598" miny="18.924782" maxx="-
66.969271" maxy="71.406235"/>
```

```
<BoundingBox CRS="EPSG:4326" minx="18.924782" miny="-
178.217598" maxx="71.406235" maxy="-66.969271"/>
<Style>
<Name>default</Name>
<Title>0</Title>
<LegendURL width="76" height="16">
<Format>image/png</Format>
<OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="http
://sampleserver1b.arcgisonline.com/ArcGIS/services/Specialty/ESRI_StatesCit
iesRivers_USA/MapServer/WMSServer?request=GetLegendGraphic%26version=1.3.0%
26format=image/png%26layer=0" xlink:type="simple"/>
</LegendURL>
</Style>
</Layer>
```

В метаданих у xml-файлі можна знайти інформацію про версію параметрів GetMap, шари, стилі, crs (або srs), bbox та формат у вмісті тегів <Layer> та <GetMap>.

Змінивши параметри запиту GetMap (bbox, формат, шари) використовуючи інформацію отриману у файлі GetCapabilities, можна завантажити різні шари карти, як це показано на рис. 1.21.

Докладніше існуючи сучасні ГІС-сервери будуть розглянуті у главі 2.

Рисунок 1.21 – Результат GetMap запиту для шарів rivers і cities

1.2.6 Сервіс Web Feature Service (WFS)

Web Feature Service (WFS) – це інтерфейс, що дозволяє здійснювати запити до географічних об’єктів через Інтернет за допомогою незалежних від платформи викликів у вигляді URL-адрес. Географічні об’єкти можна розглядати як "вихідний код" за картою, де в якості інтерфейсу WMS повертається лише зображення, яке не можна редагувати або просторово аналізувати.

Специфікація WFS визначає інтерфейси для опису операцій з обробки даних географічних об'єктів. Операції з обробки даних включають можливість:

– отримати або запитати об’єкти на основі просторових та непросторових обмежень;

– створіть новий екземпляр об’єкту;

– видалити екземпляр об’єкту;

– оновити екземпляр об’єкту.

Дані передаються між WFS та клієнтом на мові розмітки GML.

Web Feature Service (WFS) визначає операції з маніпулювання інформацією про географічні об'єкти (точки, лінії та полігони). Ці операції дозволяють виконувати транзакції (query, create, update або delete) над просторовими даними через Інтернет. Геометричні описи об’єктів у специфікації WFS кодуються на GML.

Запит WFS складається з опису запитів та операцій перетворення даних, які слід застосувати. Запит генерується на клієнті та розміщується на сервері WFS. Сервер WFS зчитує та виконує запит, повертаючи результат у як набір об’єктів, закодованих в GML (рис.1.22). Потім клієнт із підтримкою GML може використовувати набір об’єктів.

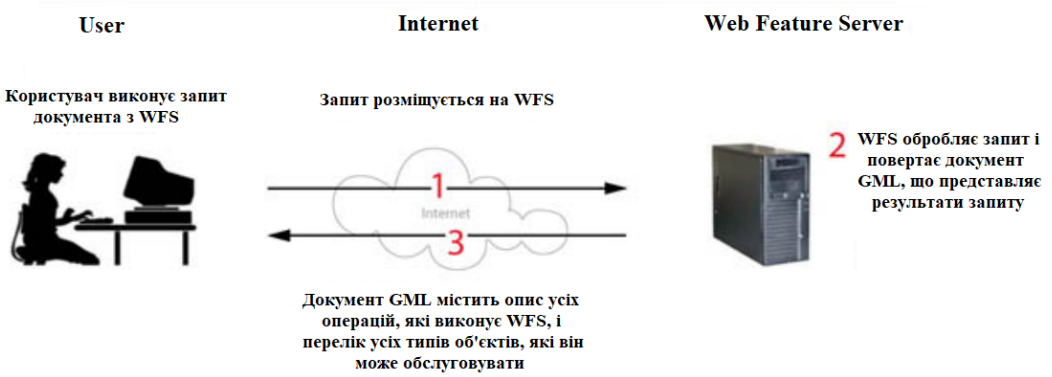

Рисунок 1.22 – Схема роботи WFS

Специфікація WFS визначає такі операції:

Операція	Опис
GetCapabilities (обов’язково)	WFS повинен вміти описувати свої можливості. Зокрема, він повинен вказати, які типи об’єктів він може обслуговувати та які операції підтримуються для кожного типу об’єкта.
DescribeFeatureType (обов’язково)	WFS повинен мати можливість, на запит, описувати структуру будь-якого об’єктного типу, яку він може

	обслуговувати.
GetGmlObject (необов'язково)	WFS може бути в змозі обслуговувати запит на отримання екземплярів елементів шляхом обходу XLinks, що посилаються на їхні XML-ідентифікатори. Крім того, клієнт повинен мати можливість вказати, чи слід також отримувати вкладені XLinks, вбудовані у дані повернутих елементів.
GetFeature (обов'язково)	WFS повинен мати можливість обслуговувати запит на отримання екземплярів об'єктів. В додаток, клієнт повинен мати можливість вказати, які властивості об'єкта потрібно отримати, і повинен мати можливість обмежувати запит просторово та непросторово.
Transaction (необов'язково)	WFS може обслуговувати запити на транзакції. Запит на транзакцію складається з операцій, які змінюють функції; тобто створювати, оновлювати та видаляти операції з географічними об'єктами.
LockFeature (необов'язково)	WFS може мати можливість обробляти запит на блокування в одному або декількох екземплярах типу об'єкта на час транзакції. Це гарантує підтримку серіалізованих транзакцій.

Відмінність між WMS та WFS демонструє рис. 1.23. Зазвичай ці і інші послуги поєднуються, а дані завантажуються з довільної кількості різних веб серверів. Наприклад: багато програм завантажують растрові дані з одного веб-сервера, що підтримує WMS, а об'єкти завантажуються з іншого веб- серверу, що підтримує WFS. На рис.1.24 показано поєднання різних послуг.

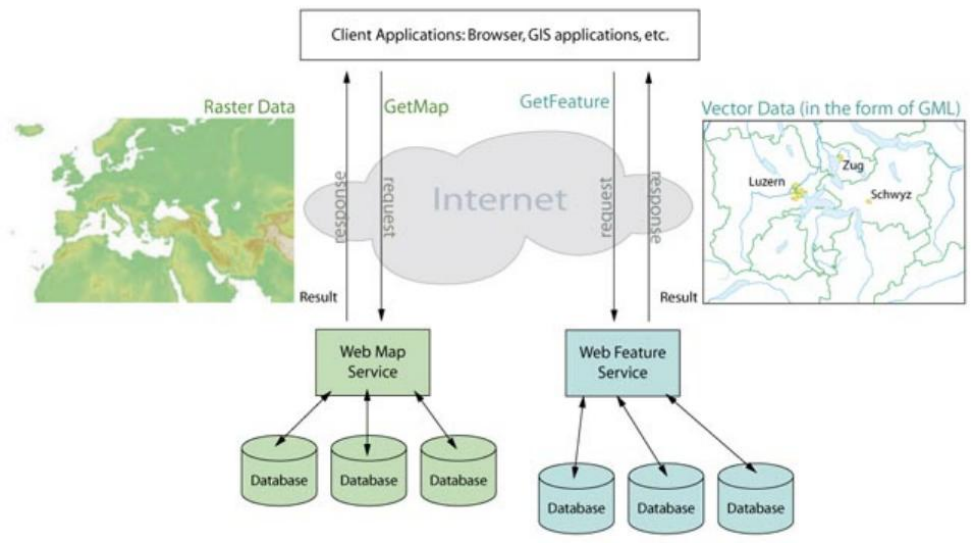

Рисунок 1.23 – Відмінності в роботі WMS і WFS

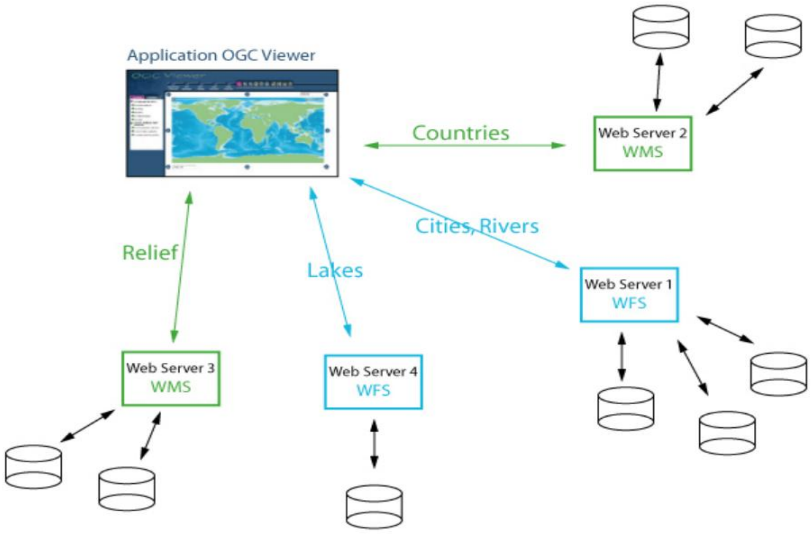

Рисунок 1.24 – Комбінування роботи WMS і WFS

1.3 Картографічний сервіс Google Maps API

Google Maps – безкоштовний картографічний сервіс від компанії Google, а також набір застосунків, побудованих на основі цього сервісу й інших технологій Google.

Сервіс являє собою карту та супутникові знімки всього світу і надає користувачам можливості панорамного перегляду вулиць (Google Street View), аналізу трафіку у реальному часі (Google Traffic), прокладання маршруту (автомобілем, пішки, велосипедом або громадським транспортом). З сервісом інтегрований бізнес-довідник і карта автомобільних доріг, з пошуком маршрутів.

Перегляд супутникового зображення може здійснюватися в режимі як «зверху-вниз» так і в «режимі польоту». Більшість аерознімків високої роздільної здатності зроблені з дронів, які пролітають над землею на висоті 240–460 м, інші зроблені з супутників.

Google Maps була розроблена як десктопна програма мовою C++ Ларсом та Дженсом Ейлстрап Расмуссенами за допомогою Where 2 Technologies. У жовтні 2004-го компанія Google перетворила її на веб-застосунок. Після додаткового придбання компанією візуалізації геопросторових даних та аналізатора трафіку в реальному часі, у лютому 2005-го програма стала загальнодоступною.

Використовуючи API Карт Google, з'являється можливість вставляти дані Google Maps на сторонній веб-сайт. Оскільки Google Maps кодується

практично повністю за допомогою JavaScript та XML, деякі кінцеві користувачі переробляють цей інструмент і створюють скрипти на стороні клієнта та вузли на стороні сервера, що дозволяє користувачеві або веб-сайту представляти розширені або індивідуальні функції в інтерфейсі Карт Google.

На сайті, з міркувань продуктивності, для передачі даних замість XML використовується JSON.

Maps JavaScript API – додає інтерактивну карту до веб-сайту користувача з опціями налаштування карт за допомогою власного вмісту та зображень.

Для створення мобільного або веб-застосунку, що працює з картами, розробнику спочатку необхідно отримати спеціальний ключ (API Key). Завдяки такому ключу вендор може контролювати використання свого сервісу, і при необхідності, зв'язуватися з розробником. Існують певні обмеження на кількість з'єднань з сервісом. Так, компанія Google встановила ліміт в 25000 підключень до Google Maps в день. Для отримання API Key необхідно створити власний проект на сайті Google https://cloud.google.com/maps-platform/, ключ буде видано саме для нього. Карти можна вставляти на будь-які сайти, крім сайтів із забороненою тематикою.

Розглянемо основні кроки, які треба виконати для створення простої карти за допомогою Google Maps API.

1) Перше, що треба зробити – це вставити ключ API Key Google Maps в URL-адресу, як показано в наступному коді. Ключ API JavaScript Карт Google з консолі API Google (http://code.google.com/apis/ console) краще замінити на YOUR_API_KEY. Якщо не змінити цю частину коду, карта не відобразиться через правила Google API.

```
<! DOCTYPE html>
<html>
<head>
<! - Include Google Maps JS API ->
<script type = "text / javascript"
src = "https://maps.googleapis.com/maps/api/js?
key = INSERT_YOUR_MAP_API_KEY_HERE & sensor = false ">
</script>
```

2) У <head> слід додати код стилю HTML, так щоб створити карту шириною 800 пікселів та висотою 500 пікселів в елемент <style> треба додати наступний код:

```
<style type = "text / css">
 #mapDiv {width: 800px; height: 500px; }
</style>
```

3) Наступні рядки JavaScript потрібні для запуску карти за допомогою Google Maps API. Об'єкт карти необхідно визначити поза функцією, щоб отримати доступ до нього з кожної частини коду.

```
<!-- Map creation is here -->
 <script type="text/javascript">
      //Defining map as a global variable to access from
      //other functions
 var map;
 function initMap() {
     //Enabling new cartography and themes
     google.maps.visualRefresh = true;

     //Setting starting options of map
     var mapOptions = {
         center: new google.maps.LatLng (39.9078, 32.8252),
         zoom: 10,
         mapTypeId: google.maps.MapTypeId.ROADMAP
     };

     //Getting map DOM element
      var mapElement = document.getElementById('mapDiv');
     //Creating a map with DOM element which is just
     //obtained
     map = new google.maps.Map(mapElement, mapOptions);
 }
```

Для карти треба задати три важливі параметри:

`center`: центр карти у форматі (широта та довгота).

`zoom`: ціле число, яке визначає рівень, на якому відображається карта. Карти Google мають рівні масштабування від 0 (світовий рівень) до 21+ (рівень вулиці). При збільшені рівня масштабування користувачі бачать більше деталей, але меншу площу території.

`mapTypeId`: параметр визначає типи базових карт, що відображаються на карті.

4) Наступний рядок коду прослуховує завантаження документа. Ця подія запускає функцію initMap, коли сторінка повністю завантажена. Це запобігає непередбачуваній поведінці, яка могла б виникнути через DOM та пов'язані з нею елементи. Елемент з id = "mapDiv" – це місце, куди буде додана карта. Цей елемент отримує свій стиль із визначених раніше тегів CSS, який має ширину 800 пікселів та висоту 500 пікселів.

```
          google.maps.event.addDomListener(window, 'load', initMap);
      </script>
</head>
  <body>
     <b>My First Map </b>
     <div id="mapDiv"></div>
  </body>
```

```
</html>
```

5) Далі можна ввести в браузер URL-адресу локального сервера, де зберігається файл map.html, та подивіться на результат, а саме карту з елементами управління навігацією в верхньому лівому кутку і елементом управління базовою картою в правому верхньому куті (рис.1.25).

Щоб створити просту повноекранну карту, треба задати наступний стиль:

```
<style type="text/css">
          html { height: 100% }
          body { height: 100%; margin: 0; }
          #mapDiv { width: 100%; height: 100%; }
</style>
```

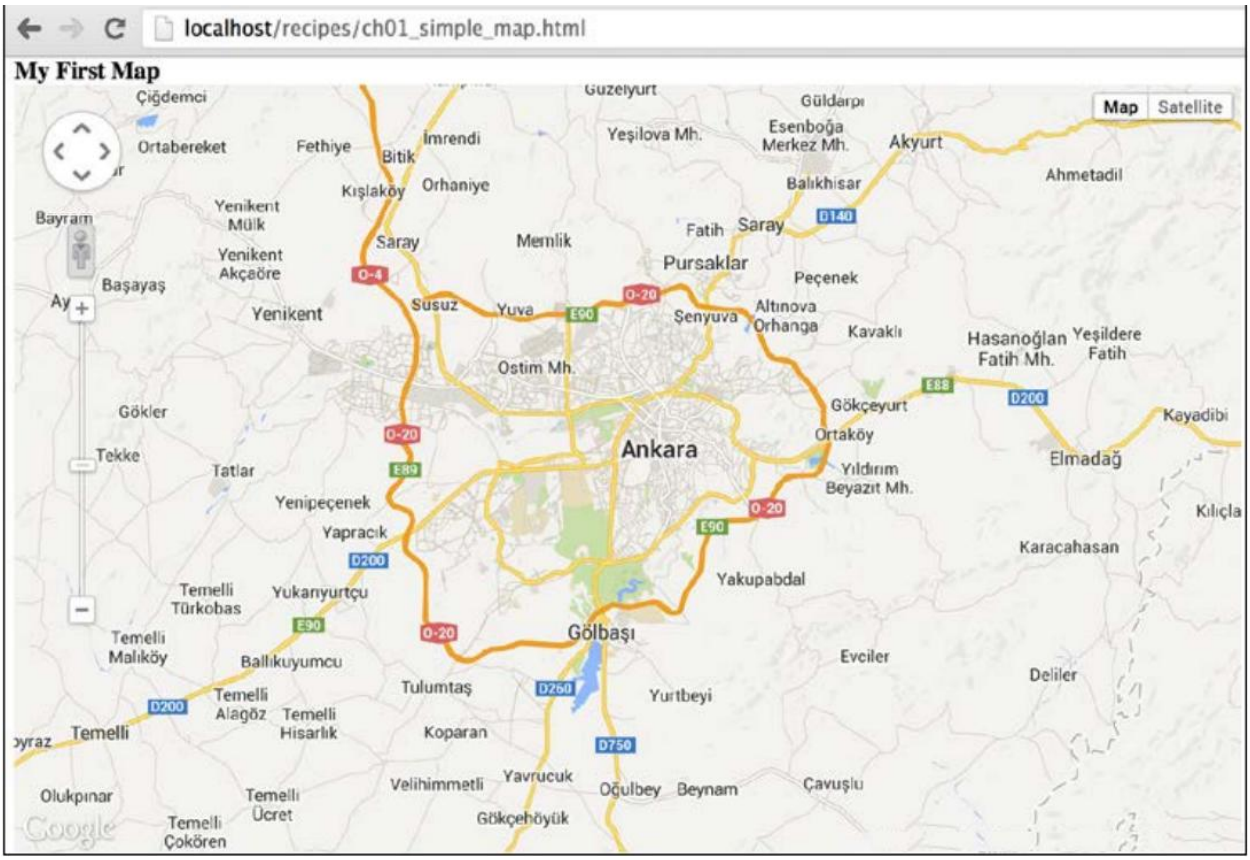

Рисунок 1.25 – Карта Google Maps

Розглянемо, як підготувати картографічний застосунок, який буде працювати в мобільних браузерах в повноекранному режимі, і наближатися до місця розташування пристрою за допомогою W3C Geolocation API. Для цього можна відредагувати попередній код наступним чином.

1) Вставити наступний рядок коду перед блоком <script>. Він повідомляє мобільні браузери, що поточний веб-застосунок розроблено для мобільних пристроїв:

```
<meta name = "viewport" content = "initial-scale = 1.0,
              user-scalable = no "/>
```

Теги <meta> використовуються браузерами і пошуковими системами, і їх не видно користувачам. В даному випадку тег <meta> використовується, щоб повідомити браузерам, що поточний веб-сайт оптимізований для мобільних браузерів. Цей тег <meta> вирішує проблеми масштабування, коли користувач збільшує або зменшує масштаб, тому що збільшення або зменшення масштабу має збільшувати або зменшувати масштаб карти, а не самого документа.

2) Додайти наступні стилі CSS, щоб зробити карту повноекранної.

```
<style type="text/css">
         html { height: 100% }
         body { height: 100%; margin: 0; }
         #mapDiv { width: 100%; height: 100%; }
 </style>
```

3) Наступний блок коду перевірить, чи підтримує браузер Geolocation API, і встановить центр карти відповідно до координат пристрою.

```
if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
        function(position) {
            var lat = position.coords.latitude;
            var lng = position.coords.longitude;
            //Creating LatLng object with latitude and
            //longitude.
            var devCenter = new google.maps.LatLng(lat, lng);
            map.setCenter(devCenter);
            map.setZoom(15);
       });
}
```

Для визначення місцезнаходження пристрою використовується W3C Geolocation API, реалізований браузерами. У стандарті HTML5 є простір імен навігатора, і браузер спочатку перевіряє підпростір імен геолокації, якщо підтримує Geolocation API. Якщо navigator.geolocation повертає об'єкт, то можна отримати координати за допомогою функції getCurrentPosition. Функція зворотного виклику отримує широту і довготу пристрою і створює об'єкт google.maps.LatLng. Потім за допомогою об'єкта devCenter запускається метод об'єкта карти setCenter, який був створений раніше. Це змінить центр карти в залежності від місця розташування пристрою. Останній рядок функції зворотного виклику використовується

для установки рівня масштабування карти. Його можна змінити відповідно до потреб.

Після завантаження файлу на відповідний сайт хостингу і перевірки URL на мобільному пристрої або стимуляторі, можна побачити карту свого розташування (рис.1.26).

Користувачі можуть змінювати властивості карти програмно: збільшувати/зменшувати масштаб, перетягувати карту, змінювати призначений для користувача інтерфейс або вмикати/вимикати інтерактивність миші. Для цього потрібно отримати доступ до карти і змінити потрібні властивості. Програмне зміна властивостей карти – одна з важливих частин Google Maps JavaScript API. У більшості картографічних застосунків користувач шукає певне місце, і застосунок повинен зосередитися на цій точці на карті. Це можливо за допомогою функцій об'єкту карти. Функцій карти багато, розглянемо тільки ті, що найбільш часто використовуються.

Рисунок 1.26 – Карта Google Maps на мобільному пристрої

На першому етапі необхідно створити об'єкт карти, щоб з ним можна було взаємодіяти. Якщо об'єкт карти не визначений, то з'явиться

повідомлення про помилку. Подібні проблеми виникають в більшості випадків через асинхронного поведінки JavaScript.

Змінити властивості карти досить просто, якщо виконати наведені нижче дії, а саме:

1) Додати наступні функції після функції initmap(). Ці функції викликаються кнопками, визначеними в HTML, які використовуються для взаємодії з картою.

```
        function zoomToIstanbul () {
         var istanbul = new google.maps.LatLng(41.0579,29.0340);
         map.setCenter(istanbul);
        }
        function zoomToStreet () {
         map.setZoom(18);
        }
        function disableDrag () {
         map.setOptions ({ draggable: false });
        }
        function setMaxZoom () {
         map.setOptions ({ maxZoom: 12 });
        }
        function setMinZoom () {
         map.setOptions ({ minZoom: 5 });
        }
        function changeUI () {
         map.setOptions ({ disableDefaultUI: true });
        }
        function disableScroll () {
         map.setOptions ({ scrollwheel: false });
}
```

2) Додати наступну функцію, щоб слухати події натискання кнопок, визначених у HTML-коді на кроці 4.

```
        function startButtonEvents () {
         document.getElementById('btnZoomToIst'
         ).addEventListener('click', function(){
         zoomToIstanbul();
         });
         document.getElementById('btnZoomToStr'
         ).addEventListener('click', function(){
         zoomToStreet();
         });
         document.getElementById('btnDisableDrag'
         ).addEventListener('click', function(){
         disableDrag();
         });
         document.getElementById('btnMaxZoom'
         ).addEventListener('click', function(){
         setMaxZoom();
         });
```

```
 document.getElementById('btnMinZoom'
 ).addEventListener('click', function(){
 setMinZoom();
 });
 document.getElementById('btnChangeUI'
 ).addEventListener('click', function(){
 changeUI();
 });
 document.getElementById('btnDisableScroll'
 ).addEventListener('click', function(){
 disableScroll();
 });
}
```

3) При ініціалізації карти потрібно викликати функцію startButtonEvents:

```
startButtonEvents ();
```

4) Додати наступні рядки коду HTML всередину тегу <body>. Це теги <button>, які відображатимуться на веб-сторінці. Кожен елемент кнопки прослуховує подію клацання, щоб запустити відповідну функцію.

```
<input id="btnZoomToIst" type="button" value="Zoom To
Istanbul">
<input id="btnZoomToStr" type="button" value="Zoom To
Street Level">
<input id="btnDisableDrag" type="button" value="Disable
 Drag">
<input id="btnMaxZoom" type="button" value="Set Max Zoom to
 12">
<input id="btnMinZoom" type="button" value="Set Min Zoom to
 5">
<input id="btnChangeUI" type="button" value="Change UI">
<input id="btnDisableScroll" type="button" value="Disable
 Scroll Zoom">
```

5) Ввести URL-адресу локального сервера, де зберігається файл *.html, у браузері та подивіться на результат. Завантажиться карта з кнопками вгорі. Кожна кнопка запускає різну функцію для взаємодії з картою (рис.1.27).

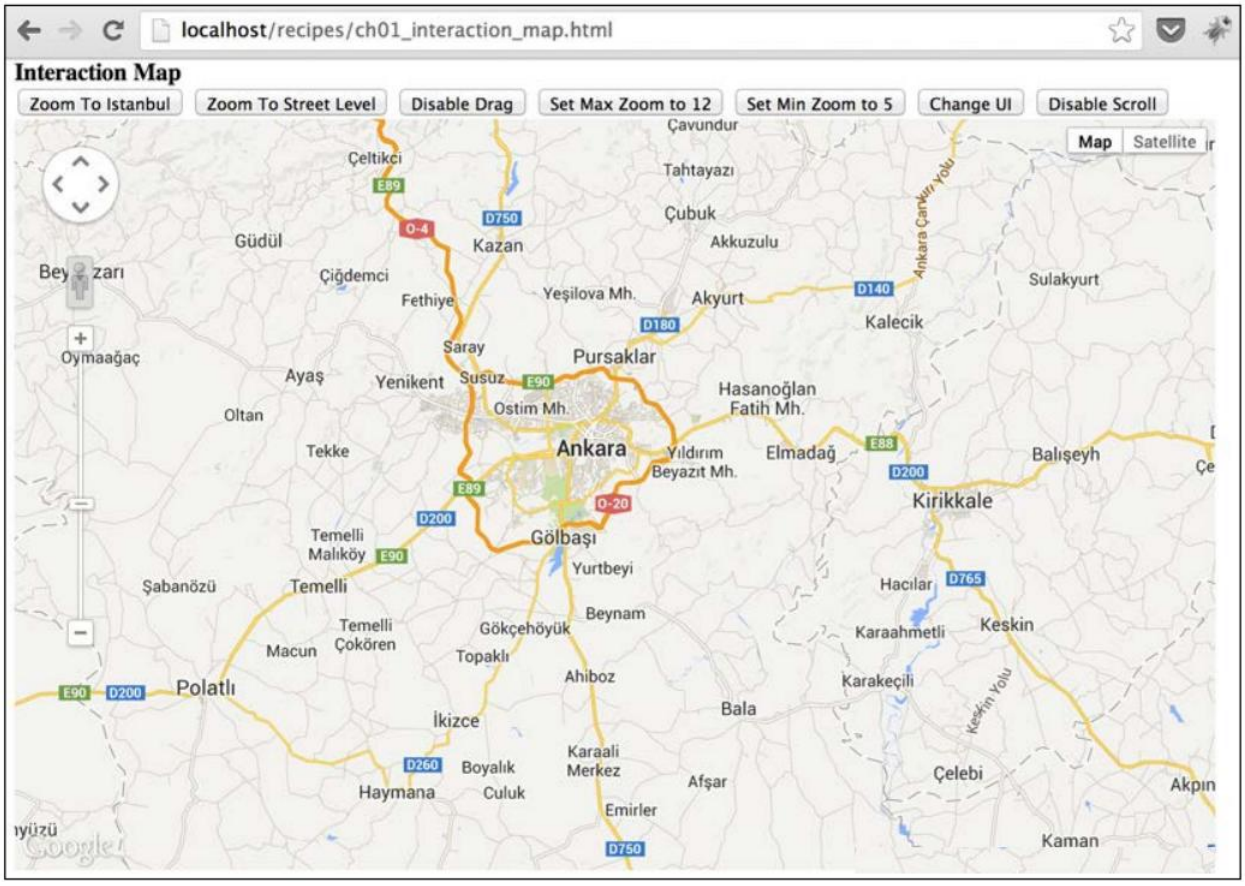

Рисунок 1.27 – Карта Google Maps з програмно зміненими властивостями

Google Maps JavaScript API має чотири різних базових карти, такі як ROADMAP, SATELLITE, HYBRID і TERRAIN. Базові карти показують дороги, супутникові зображення або рельєф місцевості, які можна використовувати в різних ситуаціях. Всі ці базові карти показані на рис.1.28.

Розглянемо як можна змінювати базові карти Google Maps програмно, для цього треба виконати наступні кроки:

1) Додати наступну функцію після функції initMap(). Вона прослуховує події натискання кнопок, доданих до HTML-коду на кроці 3 і встановлює базову карту відповідно до ідентифікаторів кнопок.

```
function startButtonEvents () {
 document.getElementById('btnRoad'
 ).addEventListener('click', function(){
 map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
 });
 document.getElementById('btnSat'
 ).addEventListener('click', function(){
 map.setMapTypeId(google.maps.MapTypeId.SATELLITE);
 });
```

```
        document.getElementById('btnHyb'
        ).addEventListener('click', function(){
        map.setMapTypeId(google.maps.MapTypeId.HYBRID);
        });
        document.getElementById('btnTer'
        ).addEventListener('click', function(){
        map.setMapTypeId(google.maps.MapTypeId.TERRAIN);
        });
}
```

Рисунок 1.28 – Базові карти Google Maps JavaScript API

2) При ініціалізації карти потрібно викликати функцію startButtonEvents:

```
startButtonEvents ();
```

3) Додати наступні рядки коду HTML перед елементом div карти. Це кнопки HTML для зміни базової карти:

```
<input id = "btnRoad" type = "button" value = "RoadMap">
<input id = "btnSat" type = "button" value = "Satellite">
<input id = "btnHyb" type = "button" value = "Hybrid">
<input id = "btnTer" type = "button" value = "Terrain">
```

4) Ввести URL-адресу локального сервера, де зберігається файл *.html, у браузері та подивіться на результат. Завантажиться карта з

кнопками вгорі. Кожна кнопка змінює базові карти відповідно до їх назв (рис.1.29).

Вказані типи карт визначені заздалегідь, але є можливість додати власні базові карти чи стилізовані карти до API та перейти до них.

Також можна визначити початкову базову карту в об'єкті mapOptions таким чином:

```
var mapOptions = {
        center: new google.maps.LatLng(39.9078, 32.8252),
        zoom: 10,
        mapTypeId: google.maps.MapTypeId.TERRAIN
 };
```

Після зміни параметрів карти вона буде відкрита з базовим типом TERRAIN.

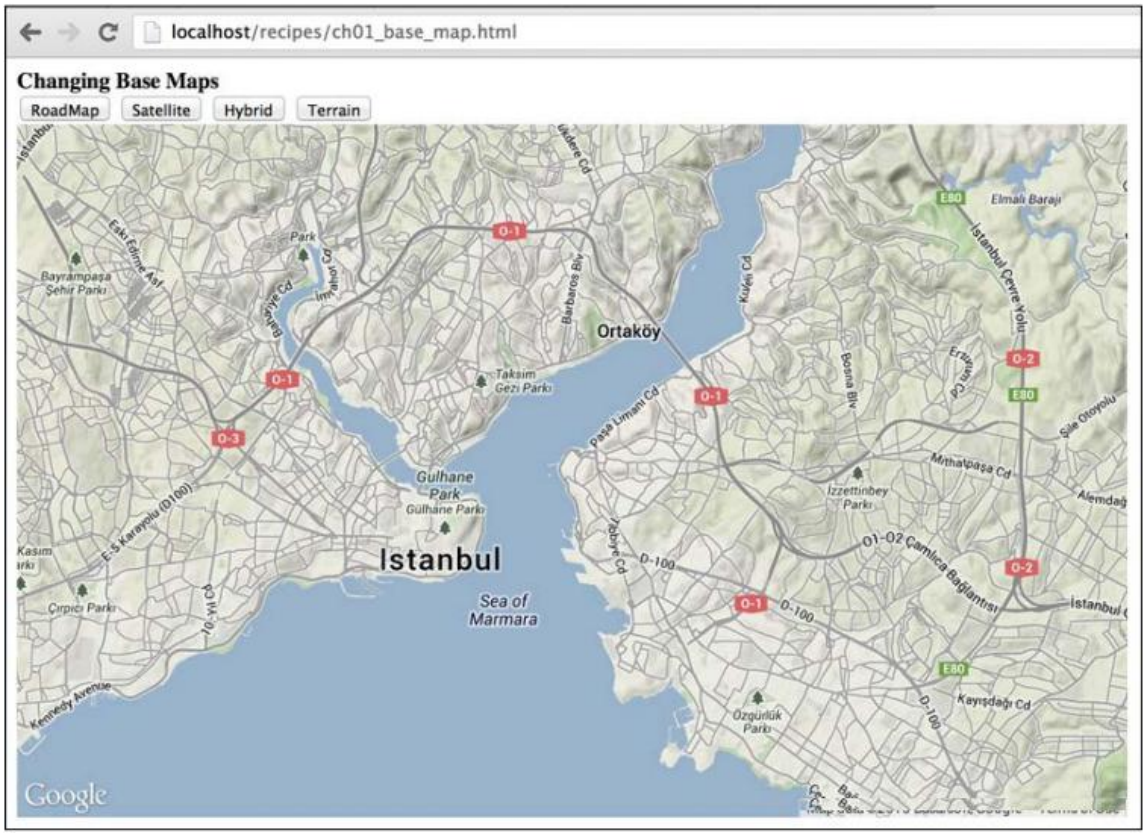

Рисунок 1.29 – Карта з кнопками для зміни базових карти Google Maps

1.4 Картографічний сервіс OpenStreetMap

OpenStreetMap – некомерційний веб-картографічний проект, який створює і надає вільні географічні дані і можливість створювати карти всього світу кому завгодно, хто це хоче. Проект надає карту всього світу,

яку може редагувати кожен, яка створюється практично з чистого аркуша по GPS-треках і поширюється під вільною ліцензією.

Ліцензія OpenStreetMap дозволяє вільно (або майже вільно) отримувати доступ до всіх растрових карт і всіх просторових даних, і цей проект спрямований на заохочення нового і цікавого використання цих даних.

Автор проекту Стів Кост був натхненний успіхом Вікіпедії і вирішив, що принцип вікі, закладений в основу Вікіпедії, може застосовуватися і для веб-картографії. Для реалізації своїх ідей в Великобританії в липні 2004 року він створив проект OpenStreetMap. Це сталося ще до появи Google Maps, і мета OpenStreetMap – отримати безкоштовну карту світу, спираючись на добровольців з GPS-пристроями, здавалася важкоздійсненним. Єдиним зразком для наслідування в подібному масовому добровільному зборі даних була Вікіпедія, яка на той час вже була серйозним конкурентом комерційних енциклопедій.

У січні 2007 року Кембридж став першим повністю намальовані містом, і до липня 2007 року, коли відбулася перша міжнародна конференція OSM під назвою The State of the Map, в проекті на той час було 9000 зареєстрованих учасників. Спонсорами заходу були в тому числі Google, Yahoo і Multimap.

У січні 2008 була реалізована можливість завантаження картографічних даних в GPS-пристрій для використання велосипедистами. У березні двоє засновників анонсували, що вони отримали венчурний капітал в 2.4 млн євро для CloudMade, комерційної компанії, яка буде використовувати дані OpenStreetMap.

У січні, коли сталося катастрофічне землетрус на Гаїті, тисячі учасників проекту взяли участь в складанні і актуалізації карти Гаїті. Це підняло популярність OpenStreetMap: був введений в широке використання термін «crisis mapping», багато ЗМІ написали про проект. Для використання в OSM свої супутникові знімки надали NOAA, GeoEye, DigitalGlobe, ErosB, CNES / Spot Image, JAXA / ALOS, Google, WorldBank, в навігаторах дані з картою Гаїті стали використовувати американські рятувальники.

Після того як відбулися руйнівні землетрус і цунамі в Японії і величезна кількість будинків просто змило, учасники OSM поза Японії за отриманими свіжим супутниковим знімкам (DigitalGlobe, JAXA/ALOS, MapQuest Open Aerial, Bing, Cnes/SpotImage, Aerial orthophotos from

Japanese mapping authority GSI) стали відзначати наслідки катастрофи, а самі японці на місцях – джерела води, магазини, телефони та інше.

1.4.1 Можливості платформи

Проект OpenStreetMap володіє великими функціональними можливостями. Від вільних картографічних даних, до розробки програмного
забезпечення на основі OSM. Варто відзначити велику кількість систем, що було розроблено на основі або з використанням OpenStreetMap. Серед них: картографічні системи, навігаційні системи, системи редагування картографічних даних і багато інших.

Виділимо основні особливості платформи:

1) Проект охоплює всю поверхню земної кулі.

2) Головною метою проекту є побудова не власне карти, а бази даних, що містить відомості про точки на земній поверхні. Таким чином, на основі зібраних в рамках проекту даних можна створювати карти різного виду і інші сервіси.

3) Карти OpenStreetMap двовимірні, без відображення висот над рівнем моря, ізоліній. Однак існують проекти, які відображають рельєфні карти, використовуючи дані про висоти зі сторонніх вільних джерел.

4) Можливий експорт карт в формати PNG, JPEG, SVG, PDF, PostScript.

1.4.2 Формат даних

Значний обсяг даних, що завантажуються в OSM, вивантажується з переносних пристроїв супутникової навігації або моніторингу. для конвертації координат з «сирого» (NMEA) або пропрієтарних форматів в формат GPX (заснований на XML) може використовуватися програма GPSBabel. Дані, зібрані в форматі WGS84 у вигляді широти/довготи, зазвичай показуються в проекції Меркатора. В цілому, вся структура даних схематично наведена на рис.1.30.

Всі дані можна умовно розбити на три основні групи:

1) типи даних, що описують у вигляді ієрархічної зв'язку сам об'єкт, як якусь просторову сутність, що має свій кінцевий результат – відомі координати всіх частин об'єкта;

2) інформаційна частина – це описова характеристика об'єкта, що не має до просторової географічній структурі об'єкта прямої відносини (його назва, фізичні, логічні та інші властивості);

3) службові атрибути об'єкта, необхідні для організації процесу зберігання і обробки інформації у вигляді набору даних, такі як унікальний ідентифікатор, стан об'єкта в базі, час останньої правки об'єкта в базі і т.д.

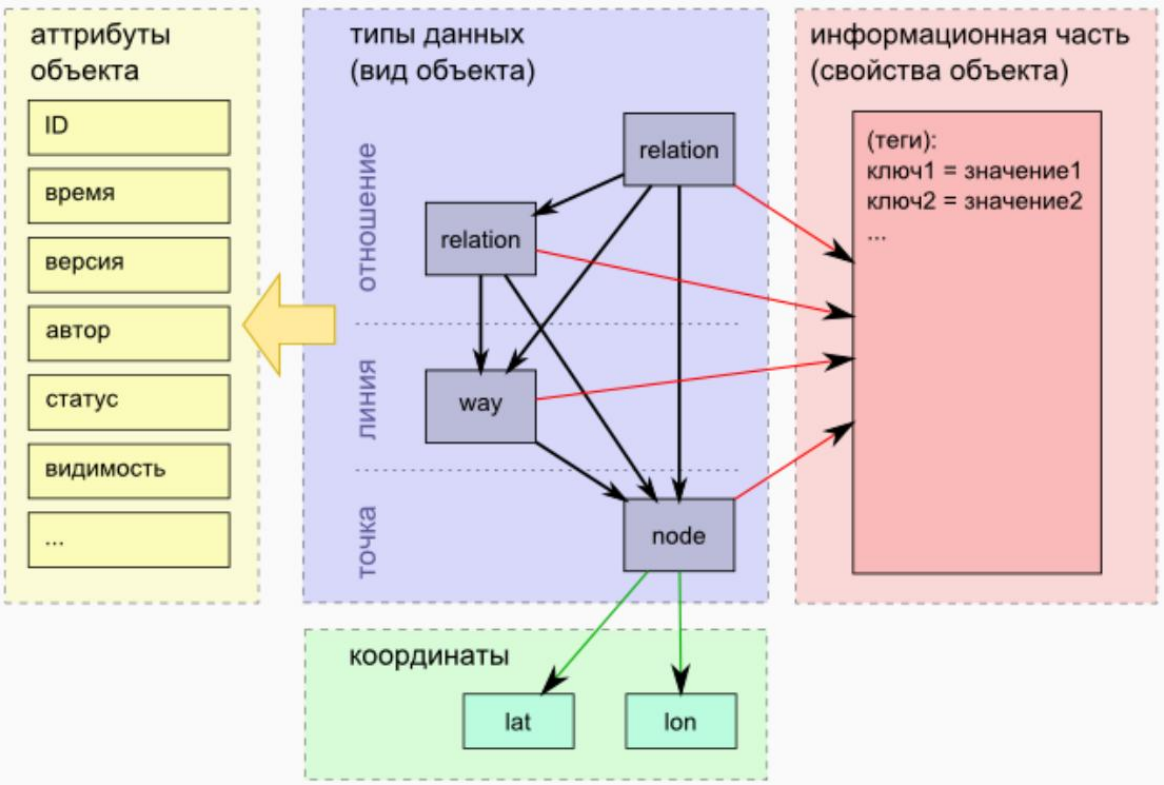

Рисунок 1.30 – Структура даних OpenStreetMap

1.4.3 Базові типи географічних даних в OpenStreetMap

Базових типів по суті всього три: точка (*node*), лінія (*way*) і відношення
(*relation*). Самі типи *node*, *way* і *relation* називаються так, тому що саме так вони були придумані спочатку. Всі без винятку об'єкти в OSM описуються цими трьома типами даних, після чого інформаційно наповнюються комбінаціями тегів. Модель даних в OSM будується на ієрархічній посилальній структурі, з чого випливає, що будь-який наступний тип даних не містить інформацію, що міститься в попередніх типах а утворює нову сутність, посилаючись на якусь множину об'єктів попереднього типу.

Так само слід згадати, що будь-який об'єкт має в структурі даних OSM свій ідентифікатор (*ID*), унікальний в межах даного типу об'єктів. Саме по цьому ідентифікатору і відбувається посилання на сам об'єкт. Розглянемо структуру базових типів по порядку.

Перший тип: точка (node) – це мінімальний набір даних, який містить в собі інформацію про парі координат: широта, довгота (lat, lon) і є базовим в ієрархічній моделі. Це єдиний тип даних, який зберігає саму географічну інформацію – координати, у вигляді широти і довготи. Модель даних OSM оперує виключно двомірними даними в межах проекції WGS84. Надалі ми будемо вважати, що координати – це не інформаційна складова об'єкта точки, а невід'ємна частина його структури. У XML нотації, об'єкт даного типу буде виглядати так:

```
    <node id = '19' lat = '58.888047127548994' lon = '49 .747870758186764'
/>
```

Одна точка з унікальним *id* рівним 19 і парою координат. Координати в OSM використовуються в десяткового запису, оскільки це набагато простіше обробляти ніж формати координат з хвилинами і секундами. Сама по собі точка може бути самостійним об'єктом, що описує якийсь точковий об'єкт (геометричний примітив) або не мати зовсім власної інформаційної складової, а бути частиною іншого об'єкта (лінії або відносини). При цьому, забігаючи трохи наперед, відзначимо, що точка одночасно може бути і самостійним об'єктом, що несе унікальну інформацію і бути частиною іншого об'єкта.

Другий тип даних: лінія (way) – це сукупність покажчиків на об'єкти типу точка (*node*). Як мінімум, лінія складається з однієї точки, тобто повинна містити як мінімум одне посилання на вже існуючий об'єкт типу точка. Лінія з однієї крапки не суперечить структурі даних OSM, але суперечить поняттям елементарної геометрії, тому правильна лінія завжди містить як мінімум посилання на два існуючих об'єкта типу точка.

Правильна XML нотація об'єкта типу лінія буде полягати в описі всіх необхідних точок, після чого йде сам запис про лінію, в якій перераховуються всі її точки. У найпростішому варіанті це буде виглядати так:

```
    <node id = '23' lat = '58.875047918145675' lon = '49
.785240674006126'/>
    <node id = '22 'lat = '58 .86687448573524' lon = '49 .737090974777324
'/>
    <way id = '24' >
    <nd ref = '22' />
    <nd ref = '23' />
```

```
</way>
```

Порядок перерахування точок в лінії важливий, він характеризує послідовність точок в лінії і напрям самої лінії, тобто у лінії завжди є початок і кінець, навіть якщо вона замкнута (в цьому випадку вони просто збігаються). В даному прикладі ми спочатку описали дві точки, задавши їх координати, а потім описали лінію, пославшись на *id* цих точок. Одна точка може входити в будь-яку кількість об'єктів ліній, при цьому повинна бути описана тільки один раз, тобто точка може бути загальною для двох ліній, в цьому випадку посилання на неї міститься в обох лініях. Таким чином будується цілісний граф об'єктів (найчастіше дорожній граф для розрахунку роутінга), який представляє з себе сукупність об'єктів (ліній), що мають зв'язок через їх загальні члени (точки).

Якщо треба створити ще одну лінію з уже існуючих точок 19 і 23, то можна описати її так:

```
<way id = '48 '>
        <nd ref = '19 '/>
        <nd ref = '23 '/>
</way>
```

В цьому випадку точки 19 і 23 вже описані вище, а точка 23 увійшла в склад двох ліній 24 і 48 і стала загальною для них.

Наші лінії 24 і 48 можна графічно представити в проекції меркартора як показано на рис.1.31. Підписи на малюнку – *id* об'єктів: червоні у точок, чорні у ліній; стрілкою вказано напрямок лінії, тобто обидві лінії закінчуються на точці 23.

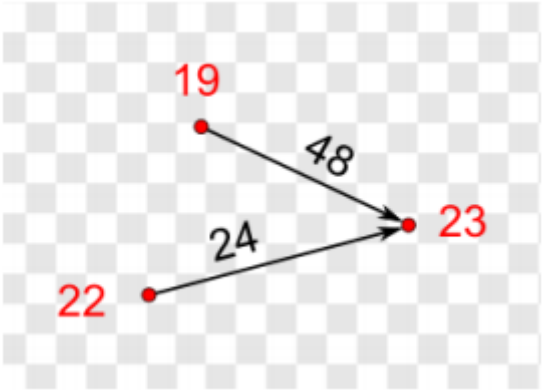

Рисунок 1.31 – Дві лінії

Наступний тип даних: відношення (relation). По суті всі об'єкти крім точки – вже відношення, проте лінії виділені в окремий тип даних як найбільш поширені, що описують основні геометричні примітиви: лінії,

полілінії та полігони. Для всіх більш складних геометричних об'єктів, а так само для об'єктів, що є не чисто геометричними, а логічними (колекції, списки, ієрархії взаємозв'язків) призначений універсальний тип даних – відношення.

В цілому, опис відношення відрізняється від лінії тим, що лінія – це завжди сукупність точок, а відношення – це сукупність будь-яких об'єктів, як точок і ліній, так і інших відношень. Отже у відношеннях вказується не тільки *id* об'єкта, але і його тип. У самому мінімальному варіанті відношення може містити посилання тільки на один об'єкт. Оперуючи об'єктами описаними в прикладах вище, можна написати відношення:

```
<relation id = '31'>
       <member type = 'way' ref = '24' />
       <member type = 'node' ref = '19' />
</relation>
```

Це фіктивне абстрактне відношення, яке описує, що в нього входить два об'єкти (члени відношення) – точка 24 і лінія 19 і більше, воно не несе ніякої іншої інформації. У реальному випадку у відношенні повинен бути зазначений тип, як тег (інформаційна складова) самого об'єкта відношення, а у членів відношення повинні бути вказані ролі в посиланнях на об'єкти.

Нижче наведено приклад найпоширенішого відношення типу «мультіполігон», яке описує один замкнутий зовнішній полігон з трьох точок з вирізаним з нього замкнутим полігоном теж з трьох точок меншого розміру. Про геометричні примітиви (замкнуті і не замкнуті полігони) і теги об'єктів мова піде далі, а поки слід звернути увагу на параметри ролей (*role*) у об'єктів відношення і наявність тега, що описує тип.

```
<node id = '1218' lat = '58 .8709411227295 'lon = '49 .75802101972955'
/>
<node id = '1216' lat = '58 .8704000725183 'lon = '49 .74703196841415'
/>
<node id = '1215' lat = '58 .8790558607720 'lon = '49 .74964840920353'
/>
<node id = '1209' lat = '58 .864718534520 'lon = '49 .780522410518245'
/>
<node id = '1207' lat = '58 .8633656498947 'lon = '49 .72453057762546'
/>
<node id = '1206' lat = '58 .892035483174 'lon = '49 .74755525657201'
/>
<way id = '1217'>
       <nd ref = '1215' />
       <nd ref = '1216' />
       <nd ref = '1218' />
       <nd ref = '1215' />
</ way>
<way id = '1208'>
       <nd ref = '1206' />
```

```
        <nd ref = '1207' />
        <nd ref = '1209' />
        <nd ref = '1206' />
</ way>
<relation id = '1 221'>
        <member type = 'way' ref = '1208' role = 'outer' />
        <member type = 'way' ref = '1217' role = 'inner' />
        <tag k = 'type' v = 'multipolygon' />
</ relation>
```

Роль outer говорить про те, що даний об'єкт буде зовнішнім контуром графічного об'єкта, а роль inner повідомляє про те, що простір всередині цього об'єкта необхідно виключити з площі результуючого об'єкта. Графічно отриманий мультіполігон показано на рис.1.32.

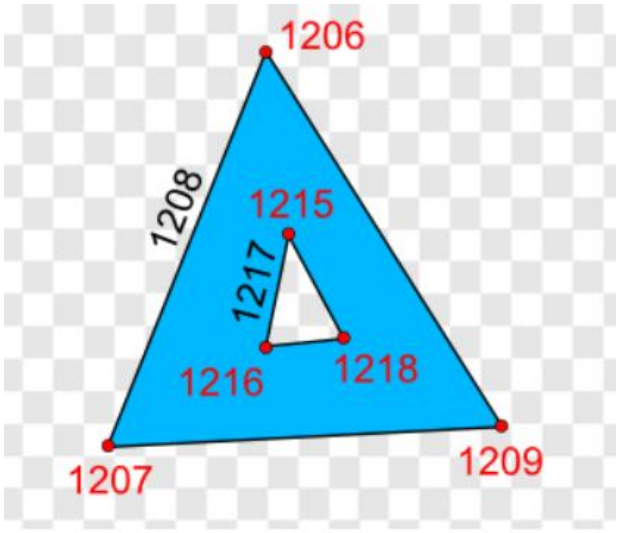

Рисунок 1.32 – Мультіполігон

Так само як і у лінії, у відношення порядок перерахування членів грає роль і враховується при використанні цього відношення. Наприклад, відношенням може бути не геометрична фігура, а маршрут громадського транспорту (логічна схема), тоді в нього входять послідовно ділянки доріг, за якими буде рухатися автобус і список точок – зупинки на яких він зупиняється, отже порядок включення доріг в відношення показує послідовність проходження маршруту, а порядок зупинок – послідовність їх відвідування.

Об'єкт *relation* може бути членом іншого *relation*, при цьому рівень вкладеності та ієрархія вгору нічим не обмежується. Структурне обмеження полягає в тому, що об'єкт *relation* не може бути членом самого себе, тобто містити посилань на самого себе. Рекурсія в структурі типів

даних в OSM неприпустима, хоча звичайно ніщо не заважає створити такий об'єкт і навіть цілком успішно увіткнути його в базу даних.

Визначивши три базових типу об'єктів треба ввести поняття початкового і кінцевого об'єкта. Початковим об'єктом є будь-який об'єкт, який входить до складу будь-якого іншого об'єкта, тобто є дочірнім по відношенню як мінімум до одного об'єкту, але сам при цьому ні включає жодного іншого об'єкта. У разі OSM це завжди точка. Або інше його визначення – початковий об'єкт несе в своїй структурі (не в інформаційній частині) тільки географічні координати і не містить посилань на інші об'єкти.

Кінцевим об'єктом є максимальний по ієрархії батьківський об'єкт, який не є дочірнім по відношенню ні до жодного іншого об'єкту, тобто не входить до складу жодного іншого об'єкта. Це може бути будь-який з трьох перерахованих типів: точки, лінії, відносини. Окремий точковий об'єкт, що нікуди не входить та складається з одного об'єкта типу точка, не є початковим об'єктом, оскільки не є початком ієрархії об'єктів, але є кінцевим об'єктом, оскільки на ньому закінчується просторовий опис об'єкта.

1.4.3.1 Інформаційна схема об’єктів

Тип об'єкта описує географічні (просторові) властивості об'єкта, але нічого не говорить про властивості самого об'єкта, його характеристиках, призначення та інше. Для це існує інформаційна частина структури даних OSM, заснована на принципах тегування об'єктів, тобто призначення їм певних міток і зазначенням властивостей цих міток. Теги задаються в вигляді пари *ключ = значення*, що в нотації XML для заданої лінії 24 виглядає так:

```
<way id = '24 '>
     <nd ref = '22 '/>
     <nd ref = '23 '/>
     <tag k = 'highway' v = 'primary' />
</way>
```

В даному випадку додана властивість лінії, а саме вказано тег *highway* зі значенням *primary*, що в прийнятій схемі тегування позначає, що лінія є основною дорогою (дорога класу нижче магістралі, але вище другорядної). Тегів у будь-якого об'єкта може бути скільки завгодно, щоб задати всі його основні властивості і описати всі другорядні параметри, а

так само в довільній формі доповнити об'єкт будь-якою інформацією. Сама схема тегування в OSM є одночасно найголовнішою її архітектурною перевагою, оскільки дозволяє описати практично будь-які властивості об'єкта, і одночасно самим слабким її місцем, оскільки будь-яка свобода у виборі способів позначення завжди породжує війни різних груп користувачів, які так і не зійшлися на думці, як позначати той чи інший спірний об'єкт.

Якщо трохи розширити інформаційний опис заданих двох ліній 24 і 48 з першої частини, то можемо отримати щось на зразок:

```
    <node id = '23 'lat = '58 .87753645355202' lon = '49 .79290110146539
'>
           <tag k = 'highway' v = 'traffic_signals' />
    </node>
    <node id = '22 'lat = '58 .87456113991739' lon = '49 .73690926857261
'/>
    <node id = '19 'lat = '58 .89362576054878' lon = '49 .7492065402827
'/>
    <way id = '48 '>
           <nd ref = '19 '/>
           <nd ref = '23 '/>
           <tag k = 'embankment' v = 'yes' />
           <tag k = 'highway' v = 'secondary' />
           <tag k = 'incline' v = 'up' />
           <tag k = 'lanes' v = '2' />
           <tag k = 'maxspeed' v = '60 '/>
           <tag k = 'name' v = 'вулиця Пожарського' />
    </way>
    <way id = '24 '>
           <nd ref = '22 '/>
           <nd ref = '23 '/>
           <tag k = 'highway' v = 'primary' />
           <tag k = 'lanes' v = '6' />
           <tag k = 'lit' v = 'yes' />
           <tag k = 'name' v = 'проспект Мініна' />
           <tag k = 'oneway' v = 'yes' />
           <tag k = 'ref' v = 'M84' />
    </way>
```

Лінія 48 стала «вулицею Пожарського», з обмеженням швидкості в 60 км/год, кількістю смуг рівним двом, що має позитивний градієнт ухилу в бік від точки 19 до точки 23, яка є другорядною дорогою і піднятою щодо рівня землі на насип. А лінія 24 – основна дорога (класом вище ніж secondary) з 6-тю смугами руху, що має стаціонарне освітлення та односторонній рух дозволений в напрямку від точки 22 в сторону точки 23, носить назву «проспект Мініна» і є частиною федеральної траси М84. Обидві дороги мають спільну точку 23, яка є перехрестям зі світлофором.

1.4.3.2 Геометричні примітиви

Одна з основних завдань для будь-яких географічних просторових даних – отримання графічного представлення об'єктів, описаних цими даними. Простіше кажучи рендеринг самих карт, схем, планів. Самі алгоритми, правила, стилі і методи рендеринга карт – це завдання прикладного софту, але все зводиться до візуалізації основних геометричних примітивів, які виходять з об'єктів трьох типів даних, перерахованих вище.

Отже, з чого формуються основні геометричні примітиви.

Точка (point) – це один об'єкт типу *node*. Положенню її в заданій проекції на карті відповідає її просторове положення в географічних координатах. Одна пара координат *lat* / *lon* транслюється в координати *x* / *y* карти з урахуванням проекції.

Лінія (line) – це найкоротша відстань між двома точками, відповідає об'єкту типу *way*, який містить два об'єкти *node*, оскільки ми оперуємо плоским простором, то відстань між будь-якими двома точками завжди буде прямий лінією.

Полілінія (polyline) – це пов'язана послідовність сегментів, де кожен сегмент являє собою одну лінію, пов'язану своїм кінцем з початком наступного сегмента. Вся послідовність є єдиним цілісним об'єктом. Відповідає об'єкту *way*, що містить три і більше *node* (рис. 1.33). Полілінія може бути об'єктом *relation*, що містить послідовно включені об'єкти *way*, де кожен наступний об'єкт *way* починається з об'єкта *node*, яким закінчився попередній *way*. Полілінія може бути або об'єктом *way*, що містить *node*, або *relation*, що містить *way*, тобто не може бути *relation*, що містить і *way* і *node* одночасно, проте може бути об'єктом *relation*, що містить одночасно *way* і інші *relation*, що містять тільки об'єкти *way* (рис. 1.34, 1.35).

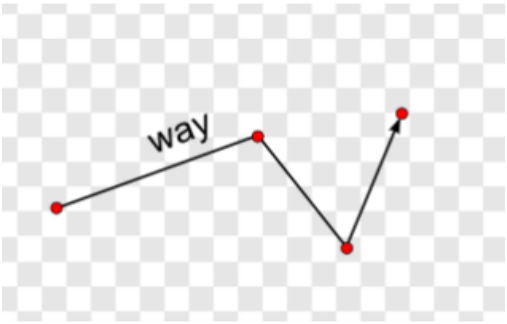

Рисунок 1.33 – polyline = way (node1, node2, node3, node4)

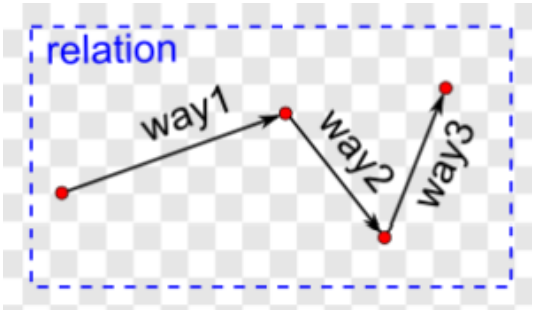

Рисунок 1.34– polyline = relation (way1 (node1, node2), way2 (node2, node3), way3 (node3, node4))

Види поліліній:

Полігон (polygon) – це замкнута полілінія, у якій остання точка збігається з першою. У типах даних OSM відповідає об'єкту *way* з декількома (три і більше) об'єктами *node*, при цьому кількість членів об'єкта *way* завжди більше на одиницю, тому що перший об'єкт *node* повторюється двічі: на початку і в кінці списку. Так само полігон може бути зібраний у вигляді *relation*, в який послідовно включаються *way*, що утворюють разом замкнутий контур, тобто початку кожного об'єкта *way* відповідає кінець іншого об'єкта *way*. На відміну від полігону в *way*, полігон в *relation* чи не дублює останній об'єкт в перерахуванні з першого, оскільки для *way* це необхідно у зв'язку з тим, що він посилається на об'єкти *node* і тільки за фактом дублювання *node* як першого і останнього елемента списку членів можна судити про те, що це замкнений полігон, а не лінійний об'єкт полілінія. У полігоні зібраному у вигляді *relation* останній об'єкт *node* останнього включеного об'єкта *way* або *relation*, що містить *way* відповідає першому об'єкту *node* першого включеного *way*.

У разі полігону описуваного у вигляді *relation*, обов'язково вказується тег *type = multipolygon* на самому об'єкті *relation*. Таким чином визначається, що мова йде про площадковий геометричний, а не про лінійний об'єкт.

Види полігонів:

Складові об'єкти – це об'єкти які неможливо описати одним примітивом *way*, завжди будуються на базі об'єктів *relation* у якого *type = multipolygon*. Полігони і полілінії описані у вигляді *relation* завжди можна спростити до одного об'єкта *way*, в той час як складений об'єкт у самому спрощеному випадку дає мінімум два об'єкти *way*. Наприклад, це

площадна фігура (полігон) з якої математично вирахувано інша фігура (полігон меншого розміру). Приклад мультіполігона, що є складовим об'єктом, з ілюстрацією був приведений на рис. 1.35, в описі об'єктів *relation*.

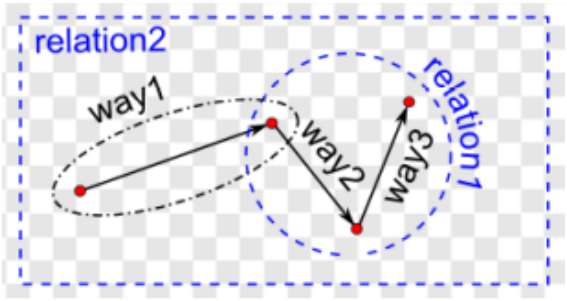

Рисунок 1.35 – polyline = relation2 (way1 (node1, node2), relation1 (way2 (node2, node3), way3 (node3, node4)))

Мультіполігони для складених об'єктів, так само як і полігони і полілінії можуть збиратися з простих *way* або з інших *relation*, що складаються з будь-якої кількості *way* (рис. 1.36). Для такого мультіполігона обов'язково треба зазначити ролі для кожного члена, що входить, будь то *way* або *relation*. Як мінімум повинен бути один член з роллю *outer*. Саме об'єкт або об'єкти з цією роллю задають головний (зовнішній) геометричний контур результуючого об'єкта. Об'єктів з роллю *inner* може і не бути в окремому випадку, але тоді такий мультіполігон – це звичайний полігон, просто описаний надлишково (рис. 1.37, 1.38).

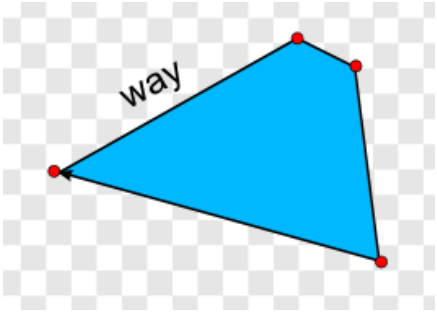

Рисунок 1.36 – polygon = way (node1, node2, node3, node4, node1)

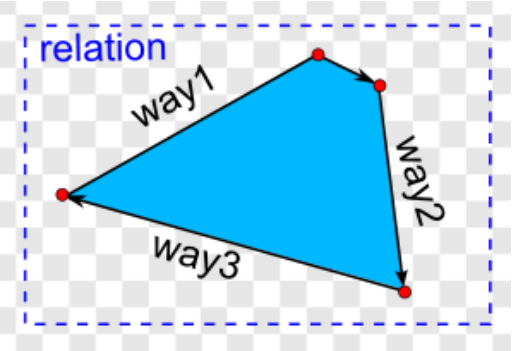

Рисунок 1.37 – polygon = relation (way1 (node1, node2, node3), way2 (node3, node4), way3 (node4, node1))

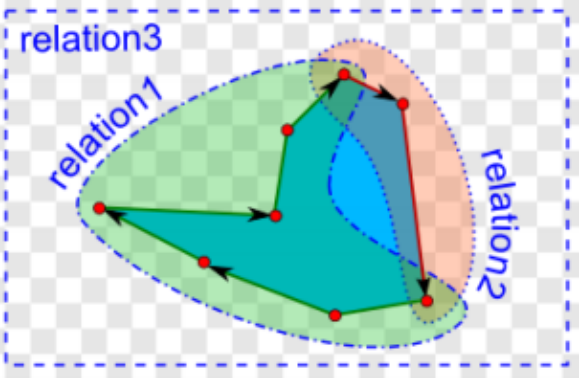

Рисунок 1.38 – polygon = relation3 (relation1 (way1 (node1, node2, node3), way2 (node3, node4), way3 (node4, node5), way4 (node5, node6, node7)), relation2 (way5 (node7, node8), way6 (node8, node1)))

Об'єкти з роллю *inner* вказують які ділянки, що знаходяться всередині зовнішнього контуру, не є частиною результуючої фігури. Наприклад це галявина в лісі або внутрішній двір будинку, обмежений з усіх боків стінами цього будинку.

Всі об'єкти з однією роллю в одному мультіполігоні повинні зібратися в один або кілька замкнутих, які не перетинаються по межах контурів.

2 ОСНОВИ РОЗРОБКИ КАРТОГРАФІЧНИХ ВЕБ-СЕРВІСІВ

2.1 Архітектура та компоненті Веб-ГІС

Веб-сервіси для просторових даних дозволяють використовувати з настільним програмним забезпеченням ГІС дані безпосередньо з серверів, без завантаження їх у вигляді файлів на власний комп'ютер. Таким чином, можна легко використовувати завжди актуальні дані. Всі найбільш поширені комерційні програмні продукти ГІС з відкритим вихідним кодом для настільних ПК підтримують веб-стандарти (MapInfo, ArcGIS, QGIS і GRASS). Веб-сервіси використовуються таким чином, що користувач підключається до сервісу за допомогою спеціального меню. Для підключення користувачеві необхідно знати URL-адресу сервера. Після підключення до сервера користувач отримує список доступних шарів карти. Веб-сервіси також легко використовувати в картографічних додатках в Інтернеті. Дані можуть бути запитані також безпосередньо за допомогою HTTP-запиту GET або POST. Як уже було розглянуто в п. 1.3 найбільш поширеними стандартами веб-сервісів Open Geospatial Consortium (OGC) є: сервіси WMS, WMTS, WFS та WCS.

Веб-ГІС розширюють можливості звичайного веб-застосування, наділяючи його функціями ГІС. Базова архітектура веб-ГІС відповідає архітектурі веб-застосувань, але з додаванням компонентів ГІС (рис.2.1).

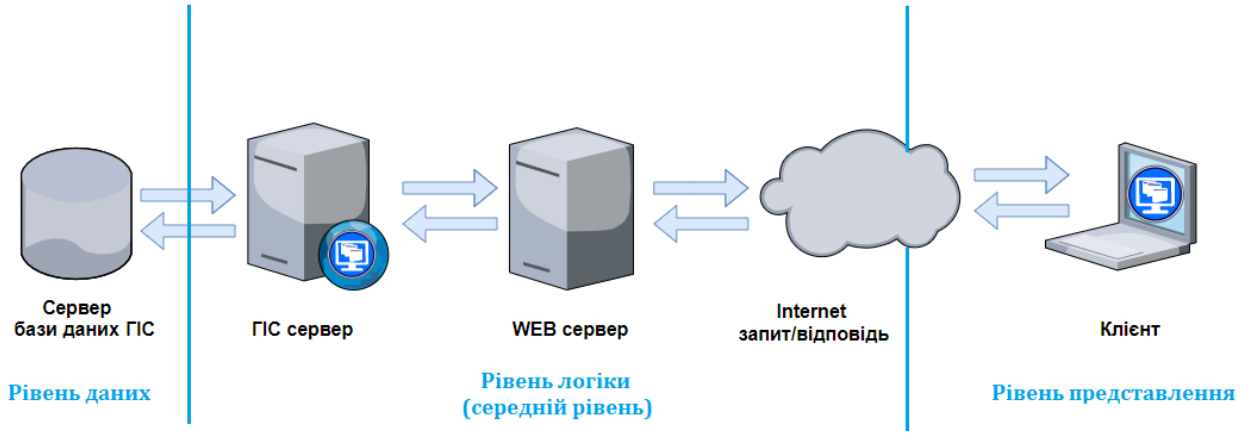

Рисунок 2.1 – Компоненти WebGIS-системи

Найпростіший веб-застосунок має, по суті, архітектуру клієнт-сервер, в якій клієнтом є веб-клієнт, а сервером– веб-сервер.

Веб-застосунки зазвичай мають трирівневу архітектуру, що складається з рівня даних, логічного рівня і рівня представлення (рис. 2.1).

Основний робочий процес веб-застосування виглядає наступним чином:

– за допомогою веб-клієнта, як правила браузера, користувач ініціює запит до веб-сервера, друкуючи URL в адресному рядку або клацаючи мишею по посиланню з URL на веб-сторінці;

– веб-сервер отримує запит, розбирає URL, знаходить відповідний докумет або скрипт, і повертає клієнтові документ або ж виконує скрипт і повертає в якості відповіді результат роботи скрипта. Відповідь зазвичай має формат HTML;

– веб-клієнт (браузер) отримує відповідь, відмальовує її і представляє користувачеві.

На рис.2.2 представлені деякі технології, які можуть використовуватися для розробки веб-застосувань.

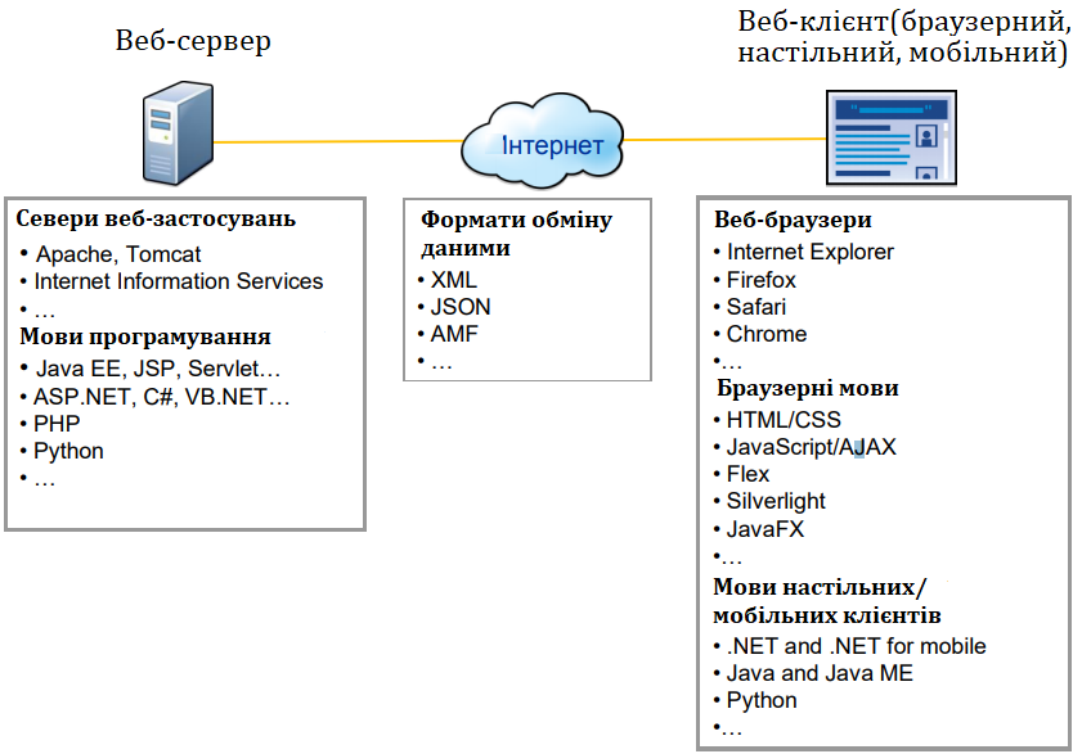

Рисунок 2.2 – Технології розробки веб-застосувань

2.1.1 Сервер веб-ГІС

Сервер веб-ГІС – це найбільш важливий компонент веб-ГІС. Його функціональність, можливість налаштування під користувача, масштабованість та продуктивність мають вирішальне значення для успішної роботи застосування веб-ГІС. Можливості та якість застосування веб-ГІС багато в чому визначаються сервером, що використовується.

Технологія ГІС-серверів розвивається вже два десятиріччя. Якщо взяти в якості прикладу продукти ESRI, в 1996 році були випущені ArcView IMS і MapObjects IMS – перше покоління комерційних серверних геоінформаційних продуктів. Вони стали важливим доповненням до програмного забезпечення ГІС і ознаменували початок руху програмного забезпечення ГІС в напрямку веб-платформи. Другим поколінням комерційних ГІС-серверів Esri став ArcIMS, випущений в 1998 році. Він дозволив підвищити продуктивність і розширити функціональні можливості, проте поступається за функціональністю

Сьогодні багато ГІС серверів доступні або як комерційні, або як відкриті/ безкоштовні продукти для публікації карт в Інтернет. Важливі з них, наведені в табл. 2.1.

Таблиця 2.1 – Опис ГІС серверів

№	Програмне забезпечення ГІС серверу	Сильні сторони	Статус
1	GeoServer	Продуктивність, безпека, підтримка векторів та веб-служби OGC	Open source (безкоштовна)
2	UMN MapServer	Продуктивність, підтримка растрових та векторних моделей даних	Open source (безкоштовна)
3	ArcGIS server	Технічна підтримка та ГІС обробка	Комерційна
4	QGIS mapserver	Зручна символізація карт за допомогою плагіна QGIS Desktop та PublishtoWeb	Open source (безкоштовна)
5	Skyline globe	3D візуалізація	Комерційна
6	Map guide	Підтримка та швидке налаштування	Open source (безкоштовна)
7	Degree	OGC Web сервіси	Open source

			(безкоштовна)
8	ERDAS APOLLO server	Підтримка растрів	Комерційна
9	Intergraph geoWeb server	Інженерно-орієнтоване застосування	Комерційна

Часто частиною ГІС серверу є сервери застосунків.

Сервер застосунків у середовищі ГІС, особливо у програмах Веб-ГІС – це програмне забезпечення, яке надає спеціальні програмні застосунки з такими сервісами, як система запитів, ГІС аналіз та обробка, генерація звітів, безпека даних та авторизації. Загалом сервери застосунків налаштовуються за допомогою інтерфейсу прикладного програмування (API, як, наприклад, Open Layers), об'єктів GIS та просторових бібліотек (наприклад, GDAL, OGR, Geotool). Багато серверів застосунків, таких як JBOSS (сервер додатків Java), постачаються в комплекті з ГІС-серверами, такими як сервер ERDAS APOLLO, і доступні як один пакет. Розвиток серверів застосунків із використанням відкритого API, такого як Open Layers API, Google API, Yahoo API стає дуже популярним у спільноті користувачів завдяки легкому розвитку та інтерактивній підтримці. Розробка сервера застосунків може здійснюватися за допомогою Java SDK, .net framework, PHP, JavaScript тощо (табл. 2.2).

Таблиця 2.2 – Важливі середовища розробки для Веб-ГІС застосунків

№	Середовища розробки	Сильні сторони	Статус
1	Open Layers	Бібліотека Java Script для доступу до шарів географічних даних усіх видів	Open source (безкоштовна)
2	Leaflet	Бібліотека Java Script . Підтримує більшість мобільних і стаціонарних платформ	Open source (безкоштовна)
	GDAL/OGR	Сумісність з будь-яким середовищем розробки	Open source (безкоштовна)
4	Geomajas	Агрегація та трансформація джерел даних ГІС	Open source (безкоштовна)
5	GeoTools	Створення, редагування та обробка даних ГІС з використанням фреймворку Java	Open source (безкоштовна)
6	GeoBase	Геокодування, навігація та оптимізація маршруту	Open source (безкоштовна)

7	GEOEXT	Розширений графічний інтерфейс із використанням API Open Layer	Open source (безкоштовна)

2.1.2 Веб-сервер

Веб-сервер – це комп'ютерна програма, яка використовує модель клієнт/сервер та протокол передачі гіпертексту (HTTP) та обслуговує файли, що формують веб-сторінки, веб-користувачам (на комп'ютерах яких є клієнти HTTP, що пересилають їх запити). Основною функцією веб-сервера є доставка веб-сторінок на запит клієнтам. Це означає доставку документів HTML та будь-якого додаткового вмісту, який може бути включений у документ, наприклад, зображень, таблиць стилів та сценаріїв. Агент користувача, зазвичай веб-браузер або веб-сканер, ініціює спілкування, подаючи запит на певний ресурс за допомогою HTTP, і сервер відповідає вмістом цього ресурсу або повідомленням про помилку, якщо не може цього зробити. Перелік популярних веб-серверів наведено в табл. 2.3.

Таблиця 2.3 – Популярні веб-сервери (2019 рік)

№	Продукт	Постачальник	Захват ринку (%)
1	Apache	Apache	44.3%
2	nginx	NGINX, Inc.	41.0%
3	IIS	Microsoft	8.9%
4	LiteSpeed Web Server	LiteSpeed Technologies	3.9%
5	GWS	Google	0.9%

Прикладами веб-серверів є:

– Apache Web Server і Tomcat від Apache Software Foundation. Apache - веб-сервер з відкритим кодом, що відрізняється простотою, швидкодією і надійністю. Він може працювати майже на всіх платформах – UNIX, Windows і Linux. Це програмне забезпечення для веб-сервера, що найбільш широко використовується, підтримує набір мов програмування Java, а також іншими мовами, якщо встановлені відповідні модулі розширення.

– IIS (Internet Information Services, раніше lnternet Information Server) від Microsoft. IIS – є розробкою web-сервера для роботи в середовищі

Microsoft Windows. Це теж популярний в світі веб-сервер, що підтримує набір мов програмування .NET, а також інші мови, якщо встановлені відповідні модулі розширення.

Іншими прикладами веб-серверів є Nginx, LiteSpeed Web Server та GWS від Google.

2.1.3 База даних ГІС

База даних ГІС – це середовище зберігання і управління даними веб-ГІС. У ній можуть знаходитися набори географічних даних різних типів включаючи прості векторні (точки, лінії, полігони) і растрові дані (наприклад, супутникові та аерознімки). Деякі бази даних ГІС підтримують і інші типи даних – дані САПР, 3D-дані, дані інженерних і транспортних систем, GPS-координати і геодезичні виміри. Бази даних ГІС варіюють від невеликих розрахованих на одного користувача до великих корпоративних, які можуть редагуватися і використовуватися одночасно багатьма користувачами. Бази даних одних ГІС зберігають тільки набори окремих просторових об'єктів, інші ж використовують складні моделі даних, що включають просторові відносини і поведінку, які критично важливі для цілого ряду завдань і аналітичних операцій в ГІС.

В основі веб-ГІС застосувань лежить база просторових даних. Відповіді, що даються веб-ГІС за стосунком будуть хороші настільки, наскільки хороша інформація, що міститься в базі даних ГІС. Якщо неформальні застосування можуть спиратися на неформальні джерела даних, то для професійних застосувань, як правило, потрібна високоякісна, своєчасно оновлювана географічна інформація з авторитетного джерела.

Якісна база даних ГІС зазвичай:

– зберігає багату колекцію просторових даних централізованим або розподіленим способом;

– застосовує складні правила і відносини в організації даних;

– підтримує геопросторові реляційні моделі (топології, дорожні мережі), забезпечує цілісність просторових даних;

– працює в середовищі доступу і редагування багатьох користувачів і підтримує версії даних;

– підтримує визначені користувачем типи просторових об'єктів і поведінку;

– має засоби для надійного захисту даних, резервного копіювання та відновлення;

– зберігає високу продуктивність при збільшенні обсягу даних і кількості одночасно підключених користувачів.

Сервер баз даних може мати файлову систему або реляційну систему керування базами даних (РСКБД), або комбінацію файлів та РСКБД. У типовому застосунку веб-ГІС просторові дані впорядковані в середовищі РСКБД, що забезпечує кращу продуктивність, безпеку даних, узгодженість даних та багато інших переваг для наборів даних ГІС. В табл. 2.4 наведені відомі програмні рішення для серверів баз даних на базі РСКБД, доступні для наборів геопросторових даних.

Таблиця 2.4 – Відомі сервери баз даних ГІС з підтримкою реляційної СКБД

№	Програмне забезпечення сервера БД	Сильні сторони	Статус
1	PostgreSQL + POSTGIS	Ефективність та вдосконалений аналіз	Open source (безкоштовна)
2	ArcSDE	Технічна підтримка	Комерційна
3	Oracle Spatial	Підтримка Java та зберігання загальних типів просторових даних у власному середовищі Oracle	Комерційна
4	MySQL	Сумісність з PHP та іншими відкритими вихідними кодами	Open source (безкоштовна)
5	TerraLib	Аналіз часових рядів і підтримується багатьма СКБД	Open source (безкоштовна)
6	SpatiaLite	Просторові розширення для бази даних SQLite з відкритим кодом	Open source (безкоштовна)
7	IBM DB2	Сильна обробка помилок	Комерційна

2.1.4 Клієнти веб-ГІС

Клієнтська програма веб-ГІС виконує дві функції. По-перше, вона є інтерфейсом кінцевого користувача для всієї системи. Вона взаємодіє з користувачем, збирає вхідні дані від користувача, посилає запити на сервер і представляє результати користувачеві. Більшість кінцевих користувачів не знають, і їм не потрібно знати, що з себе представляє обслуговуючий їх сервер (сервери). Все, що вони знають про систему, це наскільки швидкий, стабільний і дружній користувачеві клієнт.

По-друге, клієнт, особливо "товстий" клієнт, може виконувати деякі завдання геопросторової обробки, такі як динамічна класифікація для тематичного відображення на карті, кластерний аналіз або побудова "теплових" карт. Клієнтами веб-ГІС зазвичай є веб-браузери, але ними можуть бути також настільні, мобільні або навіть серверні застосування (коли сервер виступає в якості клієнта іншого сервера).

Браузерний клієнт. Веб-браузери є основним видом клієнтів веб-ГІС. Перші браузерні клієнти були, за нинішніми мірками, простуваті і повільні. Згодом веб-браузери вийшли за рамки статичного HTML і простого JavaScript. За допомогою технологій AJAX, Flex, Silverlight і JavaFX тепер можна створювати насичений, динамічний і дружній користувачеві інтерфейс, який отримав назву RIA. При використанні додаткових API, таких як інтерфейси ArcGIS для JavaScript, Flex або Silverlight, веб-браузери можуть легко виконувати багато функцій традиційних ГІС. Як приклад браузерного клієнта можна назвати ArcGIS Explorer Online.

Настільний клієнт. В якості клієнтів веб-ГІС можуть виступати настільні застосунки, розроблені, наприклад, за допомогою Java, .NET або інших мов програмування. Ці програми виконуються поза веб-браузера, і тому не обмежені його "пісочницею" (тобто жорстко контрольованим і обмеженим набором ресурсів для виконання гостьових програм). Вони мають доступ до локальних ресурсів – файлів, баз даних і периферійних пристроїв. Це повноцінні програми, які особливо підходять для ресурсномістких застосувань.

Прикладами настільних клієнтів є ArcGIS Explorer Desktop, Google Earth і ArcGIS Desktop. Роль багатьох традиційних настільних продуктів ГІС змінюється додатково до автономної роботи або роботи в якості клієнта локального сервера, вони можуть також працювати як клієнтські

програми веб-ГІС. Вони можуть бути хорошим рішенням тоді, коли браузерні клієнти не володіють достатньою потужністю, особливо якщо мова йде про професійних користувачів. Наприклад, ArcGIS Desktop може використовувати картографічні та інші типи сервісів, що надаються в Інтернеті або з хмари, і виконувати багато складних операції, недоступні в браузерних клієнтів.

Мобільний клієнт. Мобільні пристрої, які дають можливість використовувати ГІС в дорозі, стають все більш популярним типом клієнтів веб-ГІС. Їх можна розділити на дві головні категорії: клієнти на основі браузера і клієнти на основі нативного застосування. Можливості мобільних браузерів сильно розрізняються. Якщо одні мобільні браузерні клієнти здатні надавати лише найпростіші ГІС-функції з примітивним призначеним для користувача інтерфейсом, то інші наближаються до цілком "дорослих" веб-браузерів настільних комп'ютерів і здатні надавати багато функцій ГІС зі складним призначеним для користувача інтерфейсом. Нативні мобільні застосування, розроблені на .NET Mobile, Java Mobile Edition (Java MO) та інших мовах, мають перед браузерними перевагу, що полягає в можливості доступу до локальних периферійних пристроїв, таких як приймач GPS. Останній дозволяє отримувати точну інформацію про місцезнаходження користувача, що необхідно для виконання ряду завдань, наприклад, в геодезії, рятувальних операціях, для показу дорожньої обстановки в режимі реального часу, навігації та визначення місцезнаходження друзів або підприємств обслуговування, що цікавлять користувача.

Прикладами нативних мобільних клієнтських застосунків є ArcGIS Mobile, ArcGIS for iPhone, Microsoft Bing Maps Mobile, Google Earth Mobile і MapQuest Mobile.

Хоча більшість клієнтів веб-ГІС створені для конкретних застосувань, деякі з них можуть працювати з різними типами веб-вмісту. Такі клієнти називаються геобраузерамі. Геобраузерамі називають візуалізатори карт, які дозволяють переглядати стандартні картографічні веб-служби та дані, доступні в Інтернеті. Ось кілька прикладів геобраузеров:

– Carbon Project Gaia – це 2D-геобраузер. Він надає доступ до декількох видів геопросторових ресурсів – OGC WMS, WMTS, WCS, WFS, KML, GML, Microsoft Bing Maps, Yahoo Maps і OpenStreetMap, а також до файлових форматів, наприклад, до шейп-файлів Esri.

– ArcGIS Explorer Desktop можна використовувати як 2D- або 3D-геобраузер. Він може відображати вміст і послуги таких стандартів як OGC WMS, KML/KMZ, GeoRSS, служби ArcGIS Server і ArcIMS.

– ArcGIS Desktop теж можна вважати геобраузером, оскільки він дозволяє переглядати і запитувати служби ArcGIS Server, а також OGC WMS, WFS і WCS.

Віртуальний глобус – це ще одна популярна різновид клієнтів веб-ГІС. Віртуальний глобус – це тривимірна програмна модель (представлення) Землі. Він дозволяє переміщатися в віртуальному середовищі, змінюючи точку і напрямок зору. Багато з них беруть дані з Інтернету і тому називаються онлайновими віртуальними глобусами.

Онлайнові віртуальні глобуси, як правило, можна використовувати безкоштовно. Популярними прикладами є NASA World Wind (рис. 2.3), Google Earth, ArcGIS Explorer Desktop, Microsoft Bing Maps і SkylineGlobe. Більшість інтерактивних віртуальних глобусів є також геобраузерамі, оскільки на поверхню, що відображається в них, можна накладати і відображати різноманітні стандартизовані джерела геопросторових даних.

Існує ряд вузьких місць і інших чинників, які можуть негативно вплинути на якість програми веб-ГІС. До них належать такі:

– навантаження на сервер ГІС зверненням безлічі користувачів;

– навантаження на базу даних ГІС частими операціями читання/ запису даних;

– навантаження на інтернет-з'єднання для передачі великих обсягів даних;

– обмежена ГІС-функціональність клієнта, особливо браузерних;

– відсутність досвіду та професіоналізму у кінцевих користувачів.

Існують різні шляхи вирішення цих проблем, в тому числі оптимізація веб-служб, наприклад, за рахунок попереднього кешування, налаштування алгоритмів і системи, резервування, балансування навантаження і ефективного використання смуги пропускання з'єднання з Інтернетом, правильного розподілу робочого навантаження між клієнтом і сервером, проектування ефективного призначеного для користувача середовища та ін.

Рисунок 2.3 – Віртуальний NASA World Wind

2.1.5 Тонкі і товсті клієнти

Основне міркування при проектуванні будь-якої клієнт-серверної системи – розподіл робочого навантаження між клієнтом і сервером. Залежно від цього розподілу, рішення веб-ГІС можна віднести до архітектури або тонкого, або товстого клієнта.

В архітектурі з тонким клієнтом велика частина роботи покладається на сервер, клієнту залишається менша частина. По суті, клієнт просто відсилає запити користувача до сервера. Сервер ж виконує всю обробку, наприклад, створює карту і здійснює аналіз (рис.2.4). Результат зазвичай формується в форматі HTML з вбудованими зображеннями у форматі GIF, PNG або JPEG і повертається клієнтові для відображення.

Прикладом тонкого клієнта є PARC Map Viewer – перший в світі застосунок веб-ГІС. Переваги такого підходу полягають в наступному:

– Крім веб-браузера користувачеві не потрібно встановлювати ніякого програмного забезпечення, навіть плагіну до браузеру.

– Оскільки "важкі" операції виконуються сервером, клієнту не потрібно особливих обчислювальних потужностей, застосування цього типу можуть добре працювати навіть на малопродуктивних комп'ютерах.

Рисунок 2.4 – Архітектура веб-ГІС з тонким клієнтом

З недоліків можна відмітити:

– Повільний час відгуку: користувачі використовують браузер, і завантаження нового HTML-фрейма займає багато часу.

– Менша інтерактивність: на стороні клієнта можливості програми та браузера обмежені.

– Векторні дані не відображаються на стороні клієнта: браузери без додаткових плагінів не можуть читати векторні файли.

Як правило, веб-браузер може обробляти документи HTML та вбудовані растрові зображення у стандартних форматах. Щоб мати справу з іншими форматами даних, такими як векторні дані, відеокліпи чи музичні файли, функціональність браузера має бути розширена. Використовуючи абсолютно однакову комунікацію клієнт-сервер в архітектурі тонкого клієнта, формат векторних файлів використовувати не можна. Щоб подолати цю проблему, більшість браузерних програм пропонують механізм, який дозволяє програмам третього рівня працювати разом з браузером як плагін.

Функціональність інтерфейсу користувача перейшла від простого отримання документів до більш інтерактивних програм. В даний час можливості користувальницького інтерфейсу поєднуються із віддаленими викликами. Основними перевагами моделі з товстим клієнтом є можливість використовувати векторні дані та сучасний інтерфейс. Недоліками ГІС на стороні клієнта є необхідність додаткового програмного забезпечення.

2.1.6 Формати обміну даними

Сучасна веб-методологія вважає за краще відокремлювати дані від форматування і обмін даними від їх подання. Від формату даних, що використовується в обміні повідомленнями між сервером і клієнтом, може сильно залежати навантаження на канал зв'язку і, відповідно, ефективність програми. Для обміну даними в Інтернеті існують три основні формати:

XML (Extensible Markup Language) – розширювана мова розмітки, яка дозволяє визначати власні теги і атрибути. У XML багато переваг: вона незалежна від платформи (файли XML – простий текст); вона самоописова (теги і атрибути у файлі – слова звичайної мови); вона може бути оброблена автоматично, тому що добре структурована; файли XML можуть бути звірені з їх схемою для перевірки. Завдяки цим перевагам XML є одним з часто використовуваних форматів обміну даними в Інтернеті. Однак у нього є і деякі недоліки.

Він громіздкий, оскільки використовує теги зі звичайних слів для розмітки даних. Крім того, аналіз файлів XML (тобто, витяг значень даних з певних вузлів XML-документа) не відрізняється ефективністю, особливо в JavaScript.

Приклад документа на XML:

```
<address>
     <name>
         <title>Mrs.</title>
         <first-name> Mary   </first-name>
         <last-name>  McGoon </last-name>
    </name>
    <street>  1401 Main Street </street>
    <city>Anytown</city>
    <state>NC</state>
     <postal-code>   34829 </postal-code>
 </address>
```

JSON (JavaScript Object Notation) – це полегшений формат обміну даними між комп'ютерами. Як сказано у визначенні Стандарту скриптового мови програмування ECMA (Європейської асоціації виробників комп'ютерів), він є похідною від літералів JavaScript. JSON більш компактний, ніж XML, його конструкції легше розбираються в JavaScript, для якого JSON є внутрішньо використовуваним типом даних. Основне застосування JSON – програмування веб-застосувань, де він служить альтернативою XML. У прикладі нижче показано використання XML і JSON для опису однакової інформації, що стосується імен та статі трьох персон.

XML

```
<persons>
  <person>
    <name>Ford Prefect</name>
    <gender>male</gender>
  </person>
  <person>
    <name>Arthur Dent</name>
    <gender>male</gender>
  </person>
  <person>
    <name>Tricia McMillan</name>
    <gender>female</gender>
  </person>
</persons>
```

JSON

```
{
  "persons": [
    {
      "name": "Ford Prefect",
      "gender": "male"
    },
    {
      "name": "Arthur Dent",
      "gender": "male"
    },
    {
      "name": "Tricia
McMillan",
      "gender": "female"
    }
  ]
}
```

AMF (Action Message Format – це двійковий формат, розроблений Adobe і використовуваний, головним чином, для обміну даними між клієнтськими застосунками на Adobe Flex/Flash і веб-серверами. AMF є власним типом даних Flex, що робить там його обробку більш ефективною в порівнянні з JSON. У порівнянні з XML, який також є власним типом даних у Flex, AMF набагато компактніші, завдяки чому його передача та обробка більш ефективні. Хоча AMF найбільше використовується у Flex-застосунках, існують також реалізації AMF в PHP, Java і .NET.

2.1.6.1 Формат геоданих GeoJSON

Розглянемо формати обміну геоданими.

GeoJSON є форматом геоданих орієнтованим на роботу в мережі, головно на асинхронне оновлення за допомогою AJAX. Він являє собою спосіб запису геоданих, використовуючи синтаксис JSON. Поточний стандарт GeoJSON прийнято в серпні 2016 року, і з ним можна ознайомитися за адресою tools.ietf.org/html/rfc7946. Розширення GeoJSON файлів – .json або специфічне .geojson.

Складний об'єкт GeoJSON може бути представлений геометрією (geometry), об'єктом (feature) або колекцією об'єктів (feature collection). GeoJSON підтримує наступні геометричні типи: Point (точка), LineString (ламана), Polygon (полігон), MultiPoint (мультиточка), MultiLineString (мультіломаная), MultiPolygon (мультіполігон) і GeometryCollection

(колекція геометрій). Об'єкт (feature) в GeoJSON складається з геометрії і додаткових властивостей, колекція об'єктів (feature collection) – з набору об'єктів (feature).

Приклад код GeoJSON:

```
{
  "type": "Feature",
  "geometry": {
     "type": "Point",
     "coordinates": [125.6, 10.1]
  },
  "properties": {
     "name": "Dinagat Islands"
  }
}
```

2.1.6.2 Формат геоданих GML

Окремо варто відзначити згадану вище географічну мову розмітки (GML), яка розроблена на основі XML для транспортування та зберігання географічної інформації, включаючи як геометрію, так і властивості географічних об’єктів. Це відкритий стандарт обміну даними, який добре підходить для передача малого та середнього обсягу інформації. GML можна використовувати з усіма стандартними інструментами XML.

Основні характеристики GML:

• достатньо розширюваний, щоб підтримувати широкий спектр просторових завдань;

• забезпечує ефективне кодування геопросторової геометрія (точок, ліній, багатокутників);

• забезпечує кодування просторової інформації та просторових відносини, які читаються як машинами, так і людьми;

• відокремлює контент від вигляду (з одних і тих же даних можна створювати різні карти);

• дозволяє легку інтеграцію просторових та непросторових XML даних;

• забезпечує взаємодію незалежно розроблених додатків.

Відповідно до (Geoconnections.org), існує п’ять основних причин, чому слід використовувати GML:

1. GML – це стандарт OGC та ISO для кодування географічних об'єктів. Таким чином, багато постачальників вже створили продукти для виробництва або використання GML або, швидше за все, зроблять це.

Оскільки GML піддався ретельному аналізу, то він з більшою ймовірністю буде повним і безпомилковим, ніж інші методи кодування географічних об'єктів.

2. Оскільки GML заснований на XML, він сумісний з Інтернетом і успадковує від XML всі інструменти та методи, які працюють на XML схемах та екземплярах.

3. GML об'єктно-орієнтований, тому всі, що, наприклад, потрібно знати про географічне шосе (або про його ділянку), може бути закодовано в одному об'єкті без необхідності шукати іншу інформацію.

4. GML є модульним, тому можна кодувати один об'єкт або великий набір функцій, відповідних всьому листу карти з безліччю шарів (з різними рівнями складності між ними).

5. GML є обов'язковим і вбудований в інші специфікації OGC, а саме в сервіси веб-функцій і кодування фільтрів.

Нижче показано фрагмент файлу GML.

```
<gml:Polygon>
    <gml:outerBoundaryIs>
            <gml:LinearRing>
                    <gml:coordinates> 0,0 100,0 100,100 0,100 0,0
                    </gml:coordinates>
            </gml:LinearRing>
   </gml:outerBoundaryIs>
</gml:Polygon>
<gml:Point>
   <gm l:coordinates>100,200</gml:coordinates>
</gml:Point>
<gml:LineString>
   <gml:coordinates>100,200 150,300</gml:coordinates>
</gml:LineString>
```

Також треба згадати GeoSPARQL – мову запитів, яка розроблена OGC для сервісів, створених за допомогою відкритих картографічних стандартів. Іншими словами, вона служить для створення запитів до ГІС-серверів точно так же, як мова SQL служить для написання запитів до серверів баз даних.

2.1.6.3 Формат геоданих KML

Формат геоданих Keyhole (KML) – це нотація XML для вираження географічних анотацій і візуалізації в двомірних картах і тривимірних браузерах Землі. KML був розроблений для використання з Google Планета Земля. KML, який став популярним завдяки Google, доповнює

GML. У той час як GML – це мова для кодування географічного вмісту для будь-якої програми, що описує спектр об'єктів програми та їх властивостей (наприклад, мости, дороги, транспортні засоби та ін.), KML – це мова для візуалізації географічної інформації, адаптована для Google Планета Земля. KML можна використовувати для візуалізації вмісту GML, а вміст GML можна «стилізувати» за допомогою KML для цілей подання. KML – це перш за все транспорт для тривимірного зображення, а не транспорт для обміну даними.

Цікавою властивістю KML є можливість записувати не лише графічні примітиви у тривимірному (широта, довгота, висота) просторі, а й задавати точку зору і властивості віртуальної камери, що робить його дуже зручним для побудови тривимірних «реалістичних» картосхем. KML підтримує створення тривимірних об’єктів як витяганням плоских примітивів за Z-координатою, так і підставляти на місце об’єктів тривимірні моделі в широко підтриманому форматі COLLADA. Ще одною важливою особливістю KML є можливість посилатися з одного KML-файла на інші, в тому числі віддалені.

В результаті такого значної різниці в цілях кодування вмісту GML для зображення з використанням KML призводить до значної і безповоротної втрати структури і ідентичності в кінцевому KML. Більше 90% структур GML (таких як, крім іншого, метадані, системи координат, системи відліку по горизонталі і вертикалі, геометрична цілісність кіл, еліпсів, дуг та ін.) Не можуть бути перетворені в KML без втрат або нестандартного кодування. Аналогічним чином, через те, що KML використовується в якості транспорту зображення, кодування вмісту KML в GML призведе до значної втрати структур зображення KML, таких як області, правила рівня деталізації, інформація про перегляд і анімації, а також інформація про стилях і багатомасштабне уявлення. Можливість відображати мітки на декількох рівнях деталізації відрізняє KML від GML, оскільки відображення виходить за рамки GML.

2.2 Бібліотеки створення карт на стороні клієнта

2.2.1 Бібліотека Leaflet

Leaflet є сучасним відкритим проектом, написаною на JavaScript бібліотекою для відображення мобільних інтерактивних карт. Вона

розроблена Володимиром Агафонкіним та його командою. Хоча вона і займає лише 33 КБ, але має всі функції, які можуть знадобиться більшості розробників для відображення інтернет карт.

Leaflet була задумана як бібліотека, що однаково добре працює і на десктопних браузерах і на мобільних пристроях (iPhone/iPad, Android) – дуже швидка, легковажна, з простим API, гарним та зрозумілим ООП-кодом. На відміну від OpenLayers розробники не намагалися додати туди складний функціонал, створюючи складний та об'ємний код – лише основне, мінімальний набір, який задовольняє потребам 99% карт в онлайні (тайли, маркери, вектори, попапи), але реалізуючи їх найкращим чином.

Бібліотека Leaflet розроблена з розрахунком на простоту та продуктивність, а також на зручність використання де завгодно. Вона працює ефективно на всіх основних настільних та мобільних платформах, використовуючи HTML5 і CSS3 на сучасних браузерах. Функціональність бібліотеки може бути розширена за допомогою великої кількості плагінів, що мають добре документований та простий API, гарний та легко зрозумілий код.

До можливостей бібліотеки Leaflet відноситься наступні доступні шари: шар тайлів; маркери; вспливаючі елементи; векторні елементи: лінії, багатокутники (полігони), круги, круглі маркери, GeoJSON шари, шари WMS, групи шарів. Бібліотека Leaflet підтримує наступні браузери: Firefox 3.6+; Chrome; Safari 5+; Opera 11.11+; IE 7–9; IE 6; Safari for iOS 3/4/5+; WebKit for Android 2.2+, 3.1+, 4+.

Розглянемо як за допомогою бібліотеки Leaflet можна підготувати HTML-сторінку.

Спочатку у заголовку html-файла підключаємо бібліотеку Leaflet:

```
<script src=" http://cdn.leafletjs.com/leaflet-0.7/leaflet.js"></script>
```

Для нормального відображення карти та керуючих елементів необхідно підключити таблицю стилів:

```
<link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7/leaflet.css" />
```

В якості контейнера карти створимо в тілі html-сторінки пустий блок з ідентифікатором *map*:

```
<div id="map"></ div>
```

Основний код програми будемо створювати окремо в файлі *main.js*, підключемо його:

```
<script src="main.js"></script>
```

В результаті отримуємо html-сторінку:

```
<html>
   <head>
      <title>Example</title>
      <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-
0.7/leaflet.css"/>
      <script src="http://cdn.leafletjs.com/leaflet-
0.7/leaflet.js"></script>
   <style>
     body{margin:0;padding:0;}
     #map{position:absolute;left:0;top:0;bottom:0;width:100%;}
      </style>
   </head>
   <body>
      <div id="map"></div>
      <script src="main.js"></script>
   </body>
</html>
```

Для створення фігури на карті нам потрібно відмалювати карту у підготовленому блоці *map*.

```
var map = L.map('map');
```

де `'map'` – ідентифікатор нашого блоку.

Щоб завантажувалося якась конкретна область карти необхідно задати начальні координати і масштаб (метод setView([широта, довгота], масштаб)):

```
map.setView([48.7819,44.7777],14);
```

або

```
var map = L.map('map').setView([48.7819,44.7777],14);
```

После цього створимо шар , що містить зображення фрагментів карти:

```
L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png').addTo(map);
```

Метод `addTo(map)` додає шар на карту *map*. Результат на рис.2.5.

Рисунок 2.5 – Відображення карти

Розглянемо більш докладно як створювати маркери, круши та вспливаючі повідомлення.

Нехай, дано набір даних з 10 точок, який потрібно відобразити на карті:

```
data = [[48.79481 , 44.74777],
        [48.78028, 44.75485],
        [48.76463, 44.78363],
        [48.76719, 44.78686],
        [48.77923, 44.81202],
        [48.78289, 44.80787],
        [48.79229, 44.77988],
        [48.78893, 44.76422],
        [48.79407, 44.75757],
        [48.79667, 44.75634]];
```

Відобразити точку можна різними способами, розглянемо два основних: круг та маркер.

Звичайно маркер створюється командою `L.marker([широта, довгота]`. Створимо маркери для всіх точок:

```
for(i in data) {
     L.marker(data [ i ]).addTo(map);
}
```

Результат можна побачити на рис. 2.6. Для того, щоб при натисканні на маркер вспливало повідомлення з його координатами, виконаємо прив’язку к маркеру так званий *popup*:

```
for(i in data){
   marker = L.marker(data[i]).addTo(map);
   marker . bindPopup(data[i][0]+’,’+ data[i][1]);
}
```

або

```
      for(i in data) {
          L.marker(data[i]).addTo(map).bindPopup(data[i][0]+',
'+data[i][1]);
      }
```

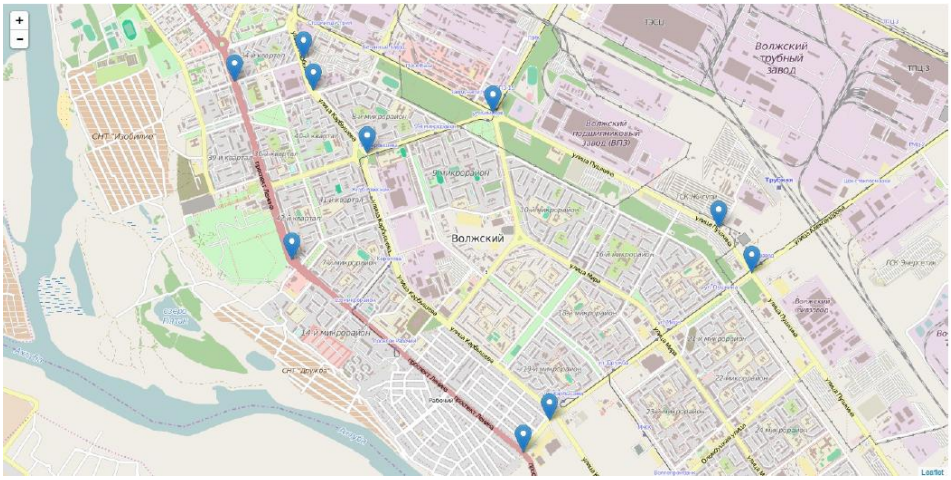

Рисунок 2.6 – Створення маркеров

Команда bindPopup(html_text) дозволяє при натисканні на маркер відображати текст, що оформлений за допомогою html-тегів, записаний в змінній html_text. Стандартно, на карті одночасно може відображатися не більше одного повідомлення, томи при виклику нового повідомлення старе автоматично зачиняється. Тому при натисканні маркера отримуємо карту, що наведена на рис. 2.7.

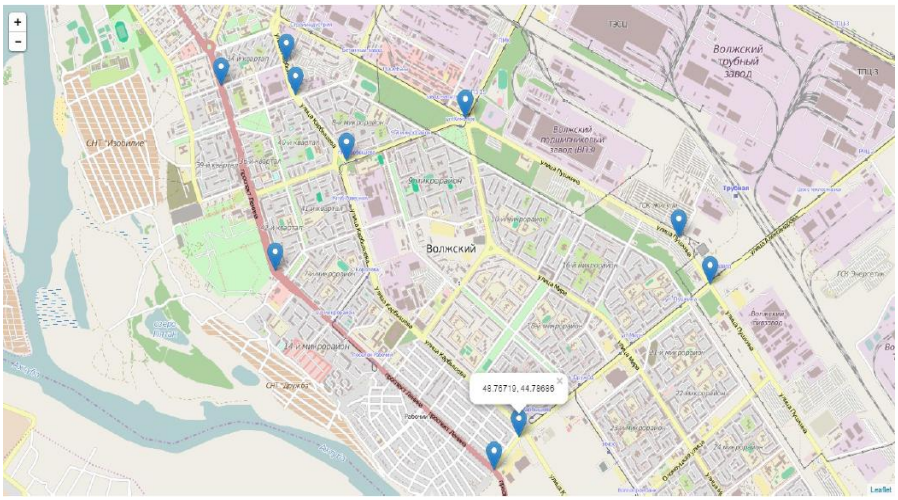

Рисунок 2.7 – Прив’язка повідомлень к маркерам

Далі замість маркеров відобразимо точки кругами. Для створення кругу використовується команда **L.circle([широта, долгота], радіус)**.Відобразимо круги для всіх точок:

```
for(i in data) {
   L.circle(data[i],50).addTo(map);
}
```

Використовує вже знайомий метод **bindPopup**, прив’яжемо повідомлення з координатами на натискання круга (рис.2.8):

```
for(i in data) {
  L.circle(data[i],50).addTo(map).bindPopup(data[i][0]+',
'+data[i][1]);
}
```

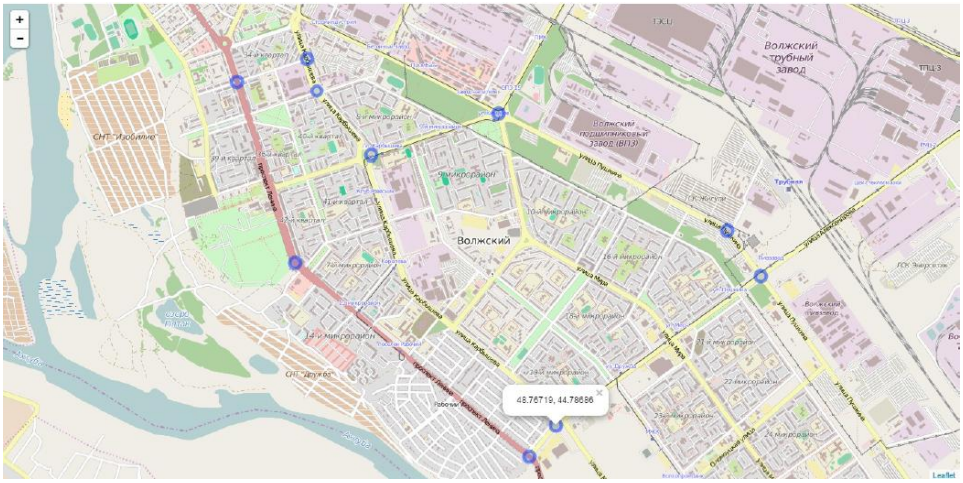

Рисунок 2.8 – Створення кругів

Прив’яжемо взпливаюче повідомлення про координати до натискання на карту в будь-якому місті. Для цього потрібно відловлювати події клика та викликати функцію відображення вспливаючого повідомлення (рис. 2.9).

Щоб при натисканні лівої кнопки миші на карту виконувалася функція onClick потрібно викликати on карти map:

```
map.on('click',onClick);
```

Тут ’click’ – це назва події. Зараз при клацанні на карту буде викликана функція onClick, якій передадуть так званий *event object* в якості аргумента. В нашому випадку це буде *MouseEvent*.

Рисунок 2.9 – Спливаюче повідомлення при натисканні на карту

Тепер напишемо функцію виклику спливаючого повідомлення. Для початку поза функцією визначимо об'єкт типу L.popup, який і буде нашим повідомленням. У функції onClick просто будемо змінювати координати виклику та текст повідомлення.

```
var popup = L.popup();
function onClick (e) {
  lat = e.latlng.lat.toFixed(5);
  lng = e.latlng.lng.toFixed(5);
  popup.setLatLng(e.latlng)
       .setContent('<b>' + lat + ' , ' + lng + ' </b> ')
       .openOn(map);
}
```

Розберемо тіло функції по рядкам. Першими двома рядками ми задаємо змінним lat та lng деякі значення. e.latlng – це об'єкт, що містить координати точки, у нашому випадку – точки кліка користувача на карті; e.latlng.lat – широта, e.latlng.lng – довгота. Таким чином, змінні lat і lng зберігають координати точки кліка на карті з 5 знаками після коми. У наступному рядку задаємо координати спливаючого повідомлення методом setLatLng. Потім ставимо контент – текст повідомлення. Виводимо координати кліка. Останнім рядком, як не важко здогадатися, показуємо повідомлення на карті.

При необхідності виділити область на карті або прокласти маршрут використовують об'єкти polygon та polyline (рис. 2.10).

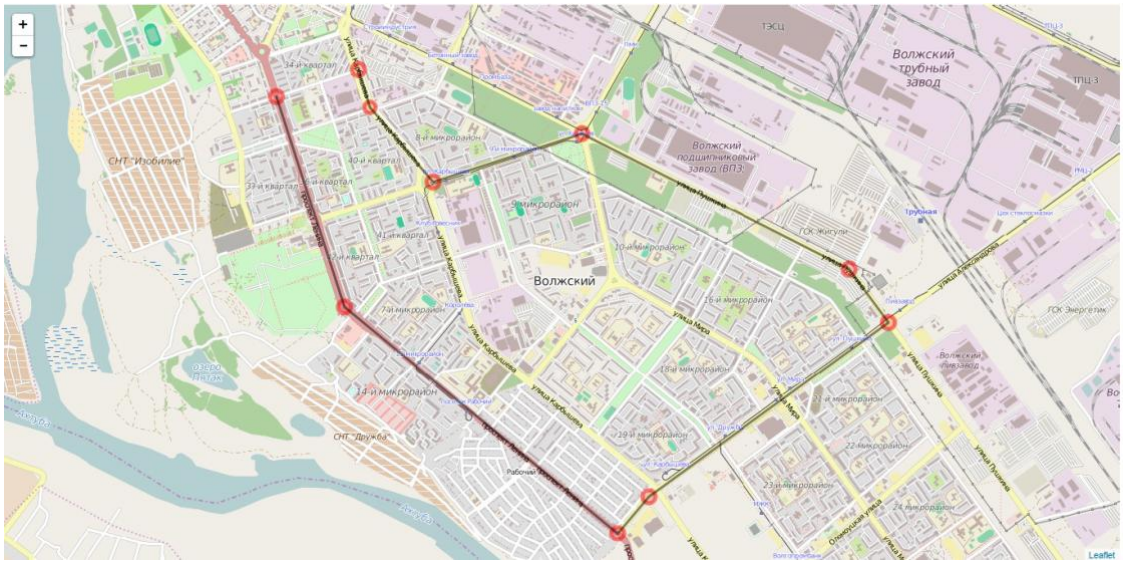

Рисунок 2.10 – Створення лінії

Для створення ламаної за заданими точками необхідно зробити наступне: `L.polyline(data).addTo(map);`

З'явиться ламану, яка з'єднує точки із заданого набору. За бажанням можна змінити її колір та товщину. Для цього після набору даних функції передати об'єкт властивостей ламаною; за колір відповідає властивість color, за товщину – weight:

```
L.polyline(data,{color:'black',weight:3}).addTo(map);
```

Таким же чином можна змінювати стиль кіл, що зображають точки.

```
L.circle(data[i],50,{color:’#f00’}).addTo(map);
```

Після цього кола будуть червоного кольору, а лінія – чорна та вузька (рис.2.10).

Для виділення області (полігону) необхідно створити об'єкт polygon (рис. 2.11):

```
L. polygon (data, {color: 'green'}). addTo (map);
```

Після цього область, що задається точками, забарвиться у зелений колір. Якщо замість масиву координат data передати масив масивів точок [data, data1, data2, . . .], то області, що перекриваються, будуть вирізатись з полігону, а неперекривні – додаватися.

Рисунок 2.11 – Створення полігону

Прив'яжемо до полігону якесь повідомлення:

```
L.polygon(data,{color:'green'}).addTo(map)
.bindPopup('<i>I\'msogreen</i>');
```

Повний текст програми (логічна змінна markers відповідає за відображення точок у вигляді маркерів замість кіл, логічна змінна polyline відповідає за малювання ламаної замість полігону):

```
var map = L.map('map').setView([48.7819,44.7777],14);
L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png').addTo(map);
map.on('click',onClick);

var popup = L.popup();
function on Click (e) {
  lat = e.latlng.lat.toFixed(5);
  lng = e.latlng.lng.toFixed(5);
  popup.setLatLng(e.latlng)
       .setContent('<b>' + lat + ',' + lng + '</b>')
       .openOn(map);
}
data = [[48.79481,44.74777],
       [48.78028,44.75485],
       [48.76463,44.78363],
       [48.76719,44.78686],
       [48.77923,44.81202],
       [48.78289,44.80787],
       [48.79229,44.77988],
       [48.78893,44.76422],
       [48.79407,44.75757],
       [48.79667,44.75634]];
markers = false;
for (i in data){
  if (markers) marker = L.marker(data[i]);
```

```
  else marker = L.circle(data[i],50,{color:'#f00'}).addTo(map);
  marker.addTo(map).bindPopup(data[i][0]+','+data[i][1]);
}

polyline = false;
if (polyline){
   L.polyline(data,{color:'black',weight:3}).addTo(map);
}  else {
   L.polygon(data,{color:'green'}).addTo(map)
    .bindPopup('<i>I\'msogreen</i>');
}
```

2.2.2 Бібліотека OpenLayers

OpenLayers 3 – це бібліотека з відкритим вихідним кодом, написана на JavaScript, призначена для створення карти на основі програмного інтерфейсу (API). Бібліотека доступна в усіх компонентах бібліотеки Rico JavaScript і Prototype JavaScript Framework.

OpenLayers дозволяє дуже швидко і легко створювати веб-інтерфейси для відображення картографічних матеріалів, представлених у різних форматах і розташованих на різних серверах. Завдяки розробнику OpenLayers є можливість створити, наприклад, власну карту, що включає слої, надані різними серверами, наприклад, Mapserver, ArcIMS або GeoServer.

Крім візуалізації WMS та WFS слоїв на єдиній веб-карті, а також редагування наданих даних (тільки для серверів WFS-T) OpenLayers має такі можливості:

– додавання на карту панелі навігації (за замовчуванням). На панелі знаходяться кнопки зсуву карти (північ-південь, захід-схід), збільшення та зменшення масштабу;

– зсув карти за допомогою миші;

– зміна масштабу карти при прокручуванні середнього колеса миші;

– отримання координат точки, над якою знаходяться покажчик миші;

– додавання панелі керування видимістю/невидимістю шарів картки;

– вибору довільного об'єкта та отримання атрибутивної інформації про нього;

– управління прозорістю шарів карти, що використовуються;

– додавання до карти елементів, що визначаються користувачем (точок, ліній, полігонів);

– багато інших можливостей OpenLayers, які разом з прикладами їх використання та документацій доступні на офіційному сайті.

Розглянемо, як відбувається налаштування бібліотеки. Оскільки OpenLayers є набором скриптів, написаних на JavaScript, ця бібліотека не вимагає встановлення у звичному сенсі. Тому для того, щоб почати працювати з OpenLayers, достатньо завантажити бібліотеку та розпакувати її у свій домашній каталог. Хоча з бібліотекою постачається безліч додаткових матеріалів (документація, приклади тощо), для початку роботи знадобиться далеко не все. Для нормальної роботи достатньо скопіювати в каталог, де зберігається ваш проект, файл build/OpenLayers.js і каталоги theme і img, на цьому установка завершена.

У OpenLayers API є два основних поняття: це об'єкти Map і Layer. Map зберігає інформацію про проекцію, географічне охоплення, одиниці виміру та інші параметри карти в цілому. Всередині картки (Map) дані задаються за допомогою одного або кількох об'єктів Layer. Layer містить інформацію про шари даних, які будуть розміщені на карті, і про те, як кожен із цих шарів повинен відображатися на карті.

Розглянемо кроки створення простої карти. Спочатку потрібно підготувати вихідний файл, в який буде вбудований об'єкт OpenLayers, який відповідає за відображення карт (OpenLayers підтримує вбудовування карт у будь-який блоковий елемент HTML-коду). Крім цього, в текст сторінки потрібно вставити тег із посиланням на скрипт із бібліотеки OpenLayers:

```
<html>
<head>
  <title>OpenLayers Example</title>
    <script src="OpenLayers.js"></script>
    </head>
    <body>
      <div style="width:100%; height:100%" id="map"></div>
</body>
</html>
```

Зрозуміло, що попереднього прикладу ще мало, щоб відобразити карту на сторінці. Потрібно створити ще об'єкт, який відповідатиме за відображення картки:

```
var map = new OpenLayers.Map('map')
```

У конструкторі об'єкта map використовується один аргумент – назва елемента HTML, пов'язаного з картою, або ID цього елемента.

Наступним кроком буде додавання шарів об'єкту map. OpenLayers підтримує безліч різних форматів, у цьому прикладі ми розглянемо шар WMS, що надається ресурсом MetaCarta:

```
var wms = New OpenLayers.Layer.WMS(
  "http://labs.metacarta.com/wms/vmap0",
  {'layers':'basic'});
map.addLayer(wms);
```

Тут перший параметр у конструкторі – адреса WMS сервера. Другий аргумент – об'єкт, що зберігає параметри додавання їх до рядка WMS-запроса.

Нарешті, щоб вивести карту, потрібно вказати її центр і масштаб (zoom). У нашому прикладі ми розмістимо карту на всій площі вікна (точніше, елемента "map"), для цього скористаємося функцією zoomToMaxExtent. У результаті має вийти наступний файл:

```
<html>
<head>
  <title>OpenLayers Example</title>
    <script
    src="http://openlayers.org/api/OpenLayers.js"></script>
    </head>
    <body>
      <div style="width:100%; height:100%" id="map"></div>
      <script defer="defer" type="text/javascript">
        var map = new OpenLayers.Map('map');
        var wms = new OpenLayers.Layer.WMS( "OpenLayers WMS",
            "http://labs.metacarta.com/wms/vmap0",           {layers:
'basic'});
        map.addLayer(wms);
        map.zoomToMaxExtent();
      </script>
</body>
</html>
```

А ось що з цього вийшло (рис. 2.12).

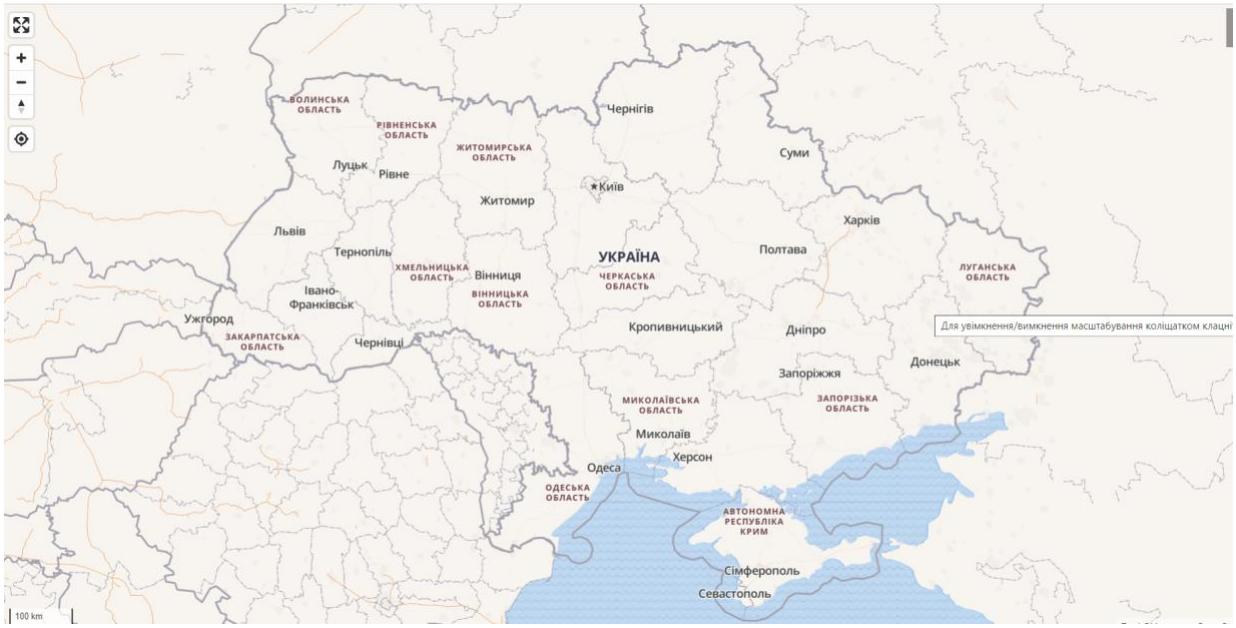

Рисунок 2.12 – Вигляд карти

2.3 Картографічні веб-сервери

2.3.1 ArcGIS Server

ArcGIS Server – продукту третього покоління. ArcGIS Server надає найбільш повний набір функціональних можливостей (рис. 2.13), в тому числі:

– Публікацію в Інтернеті або інтрамережі плоских карт, об'ємних глобусів, знімків, інструментів запиту, пошуку, редагування просторових об'єктів, вилучення даних, спостереження за об'єктами, знаходження адрес і населених пунктів, маршрутизації, обробки геометрії (наприклад, зміна координатної системи) і можливості роботи з метаданими.

– Підтримку новітніх функцій гео-обробки, що настроюються під користувача.

– Надання функціональності за допомогою веб-служб – програмних веб-компонент, які можна багаторазово використовувати і по-різному поєднувати для створення нових застосунків. Технологія веб-служб дозволяє знизити надмірність даних і застосунків, оптимізувати системну конфігурацію і консолідувати системи підприємства. Веб-служби в стилі REST (Representational State Transfer, зміна станів через передачу уявлень), що стали останнім часом популярними, продемонстрували безліч нових

переваг додатково до традиційних веб-служб на основі SOAP (Simple Object Access Protocol, простий протокол доступу до об'єктів).

– Відповідність галузевим стандартам, таким як WMS, WFS, WCS, CSW, GML і KML, розробленим OGC, що забезпечує сумісність на рівні веб-служб з продуктами різних виробників в стилі "plug and play".

– Способи створення високоякісних сервісів, включаючи кешування, оптимізацію і інші методи забезпечення високої продуктивності і масштабованості.

– Сервіси з використанням токенов і інші методи захисту геопросторових веб-служб.

– Широкі можливості налаштування під користувача. Підтримка різних сучасних інтерфейсів прикладних програм (API), за допомогою яких можна побудувати високо інтерактивні та привабливі клієнтські програми для веб-браузера, настільних комп'ютерів і мобільних пристроїв.

Рисунок 2.13 – Набір функцій веб-служб ArcGIS Server

Розглянемо декілька серверів карт, якими можна користуватися безкоштовно.

2.3.2 QGIS Server

Сервер QGIS – це реалізація WMS із відкритим кодом (1.3.0 та 1.1.1) з розширеними картографічними функціями. Спочатку він був розроблений в Інституті картографії в Цюріху.

Сервер QGIS має:

– покращений WMS (сервіс веб-карт) за допомогою HTTP GET. Підтримує GetCapabilities, GetMap, GetStyle, GetFeatureInfo та спеціальний

стиль за допомогою Styled Layer Descriptor (підтримувані стандарти: WMS 1.3.0, WMS 1.1.1 та SLD[1)] 1.0.0);

– SOAP[2)] через HTTP POST;

– власна конфігурація з SLD. Зручна символізація карт за допомогою плагіна QGIS Desktop та PublishtoWeb;

– картографічні розширення SLD (схеми, шаблони та власні символи із масштабованою векторною графікою);

– обмін картографічними правилами з операцією GetStyle.

QGIS Server є написаним на C++ застосуванням FastCGI/CGI (Common Gateway Interface), яке працює спільно з веб-сервером (наприклад, Apache або Lighttpd). Він використовує QGIS для відтворення карти і ГІС-логіки. Графічна підсистема реалізована за допомогою бібліотеки Qt, що дозволило отримати крос-платформність. На відміну від інших WMS-рішень, QGIS Server використовує картографічні правила в SLD/SE і як мову конфігурації сервера, і для опису призначених для користувача картографічних правил.

Крім того, проект QGIS Server надає розширення «Publish to Web» для QGIS, за допомогою якого можна експортувати поточні шари і символіку в проект для QGIS Server (включаючи правила відображення в форматі SLD).

Так як QGIS і QGIS Server використовують одні і ті ж бібліотеки візуалізації, карта, опублікована в Інтернет, виглядає точно так само, як і в настільній ГІС. Модуль «Publish to Web» підтримує базову символіку, більш складні правила картографічної візуалізації задаються вручну. В якості конфігураційних файлів використовується стандарт SLD і його розширення, таким чином, необхідно знати тільки одну стандартизовану мову, що значно зменшує складність створення карт для Інтернет.

2.3.3 UMN Mapserver

MapServer – це середовище розробки з відкритим вихідним кодом для створення просторових інтернет-програм. MapServer дозволяє здійснювати рендеринг просторових даних (карт, зображень та векторних

[1)] Styled Layer Descriptor (SLD) – стандарт кодування OpenGIS Web Map Service (WMS) визначає кодування, яке розширює стандарт WMS, дозволяє визначати символізацію та розкраску географічних об’єктів та даних покриття.

[2)] SOAP (англ. Simple Object Access Protocol) — протокол обміну структурованими повідомленнями в розподілених обчислювальних системах, базується на форматі XML.

даних) для Інтернету. MapServer спочатку розроблявся в Університеті Міннесоти (UMN) у співпраці з НАСА та Департаменту природних ресурсів Міннесоти.

UMN WebServer підтримує:

– сотні форматів растрових та векторних даних:
    • JPEG, PNG, GIF, TIFF/GeoTIFF та ін.;
    • ESRI shapefile, GML, PostGIS, Interlis та ін.;
– веб-специфікації OGC:
    • WMS, WFS, SLD, GML, WCS, WMC, SOS та ін.;
– можливість роботи на різних операційних системах:
    • Windows, Linux, Mac OS X тощо;
– підтримка популярних мов сценаріїв та середовищ розробки:
    • PHP, Python, Perl, Ruby, Java, .NET;
– перепроеціювання на льоту за допомогою бібліотеки Proj.4;
– високоякісну візуалізацію;
– повністю настроюваний користувачем результат виведення програми;
– безліч готових до використання середовищ додатків з відкритим вихідним кодом.

На рис.2.14 представлена базова архітектура застосувань з MapServer.

Простий застосунок MapServer складається з:

• Файл карти (Map File) – файл конфігурації структурованого тексту для програми MapServer. Він визначає область карти, повідомляє програмі MapServer, де знаходяться дані та куди виводити зображення. Він також визначає шари карти, включаючи джерело даних, проекції та символіку. Він повинен мати розширення .map, інакше MapServer його не зможе розпізнати.

• Географічні дані – MapServer може використовувати багато типів географічних даних. Типовим форматом є формат Shape ESRI.

• HTML-сторінки – інтерфейс між користувачем та MapServer. Проста програма CGI MapServer може включати дві HTML-сторінки:

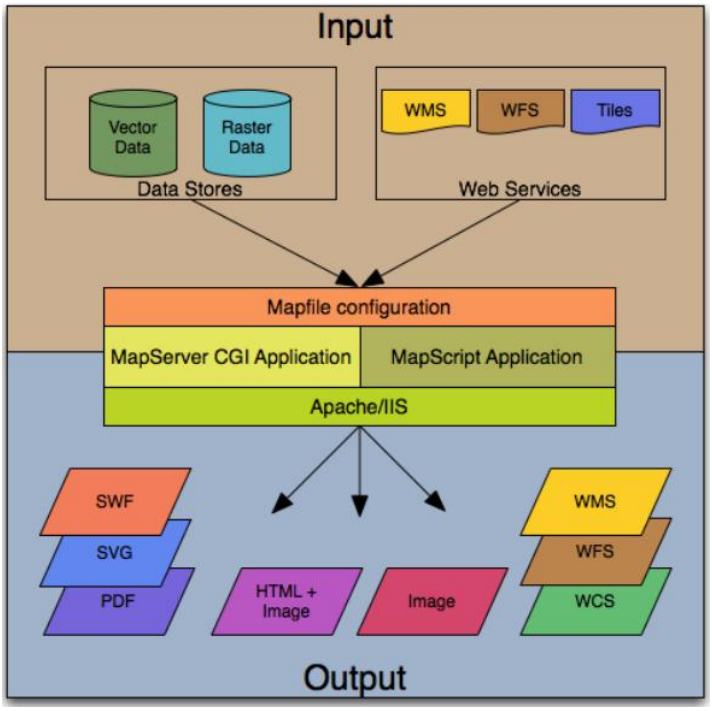

Рисунок 2.14 – Базова архітектура застосувань MapServer

– Файл ініціалізації (Initialization File) – використовує форму із прихованими змінними для надсилання початкового запиту на веб-сервер і MapServer. Цю форму можна розмістити на іншій сторінці або замінити, передавши інформацію про ініціалізацію як змінну в URL-адресу.

– Файл шаблону (Template File) – контролює, як карти та легенди, що виводяться MapServer, відображатимуться у браузері. Посилаючись на змінні CGI MapServer у шаблоні HTML, користувач дозволяєте MapServer заповнювати їх значеннями, пов’язаними з поточним станом програми (наприклад, ім’я зображення на карті, ім’я посилального зображення, розмір карти тощо), оскільки воно створює HTML-сторінку в браузері для читання. Шаблон також визначає, як користувач може взаємодіяти з програмою MapServer (перегляд, масштабування, панорамування, запит).

• MapServer CGI – двійковий або виконуваний файл, який приймає запити та повертає зображення, дані тощо. Він розміщений у каталозі cgi-bin або скриптах веб-сервера. За замовчуванням ця програма називається mapserv.exe.

• Web/HTTP Server – обслуговує HTML-сторінки при натисканні на них користувачем браузера. На комп'ютері, на якому встановлюється

MapServer, необхідна наявність працюючогоWeb (HTTP) -сервера, наприклад, Apache або Microsoft Internet Information Server.

Таким чином, основу MapServer становить CGI-програма mapserv.exe, яка приймає від користувача параметри (файл опису карти, шари карти, розміри поточної карти і т.п.), зазначені в адресному рядку браузера, після чого відображає необхідну карту користувачеві. Зазвичай це відбувається так: отримавши від користувача запит, MapServer генерує растровий файл і вбудовує його в html-документ, що відсилається користувачеві. Які шари прийматимуть участь при генерації файлу, як саме вони будуть відображатися, чи будуть підписані об'єкти на карті, а також багато іншого вказується в спеціальному файлі з розширенням .map, який передається програмі mapserv.exe в якості одного з параметрів, заданих користувачем в рядку адреси.

Велика частина роботи над створенням web-ресурсу як раз і складається з написання map- файлу. Мається на увазі, що MapServer вже встановлений і працює.

2.3.3.1 Приклади конфігурування MAP-файлу

Приклад map-файлу можна переглянути нижче.

```
MAP
  IMAGETYPE      GIF
  EXTENT         34.59 49.58 34.63 49.6
  SIZE           400 300
  SHAPEPATH      "/ms4w/apps/example/shp/"
  IMAGECOLOR     255 255 255

  LAYER # Визначаємо полігональний шар
    NAME         veget
    DATA         Vegetation_region
    STATUS       ON
    TYPE         POLYGON

    CLASS
      NAME       "Рослинність"

      STYLE
        COLOR        232 232 232
        OUTLINECOLOR 32 32 32
      END
```

```
    END
  END # Кінець визначення шару
END # Кінець визначення карти
```

Map-файли починаються з ключового слова MAP, що позначає початок "Map"-об'єкту. Закриває map-об'єкт ключове слово END в кінці файлу. Вся карта, яка буде відображатися користувачеві описується всередині.

Всередині MAP-об'єкта визначаються нові об'єкти – шари (LAYER). Обов'язково потрібно визначити принаймні один шар. Кількість шарів обмежена зверху (за замовчуванням – максимум 100 шарів). Якщо потрібна більша кількість шарів, доведеться перекомпілювати MapServer (файл map.h).

Всередині шару потрібно визначити як мінімум один клас. Класів може бути декілька, але не більше 10 (інакше знову доведеться перекомпілювати MapServer).

Всередині класу визначаються стилі: як саме даний клас відобразити на карті.

Щоб подивитися результат, треба набрати у вікні браузера:

```
http://localhost/cgi-bin/mapserv.exe?map=/ms4w/apps/example/
polt.map&layer=veget&mode=map
```

Якщо ви працюєте не на локальній машині, то замість localhost потрібно вказати ім'я сервера. Повинно з'явитися таке зображення (рис. 2.15):

Рисунок 2.15 – Результат відображення класу

Розглянемо докладніше, що написано в map-файлі:

IMAGETYPE GIF – тип растра, який згенерує MapServer для відправлення користувачеві.

EXTENT x1 y1 x2 y2 – координати нижнього лівого і верхнього правого кутів карти (географічне охоплення), що відображається на екрані. У нашому випадку шари карти представлені в географічних координатах (десяткові градуси, широта/довгота, WGS84) і всі вони укладаються в прямокутник з координатами кутів: (34.59 49.58, 34.63 49.6).

SIZE a b – розміри карти в пікселях.

SHAPEPATH "адреса" – місце розташування shp-файлів.

IMAGECOLOR r g b – колір фону карти в форматі RGB.

Параметри об'єкта "LAYER":

NAME "ім'я" – назва шару.

DATA "ім'я файлу" – назва shp-файлу, що відповідає даному шару.

STATUS ON|OFF|DEFAULT – перемикач, що означає чи потрібно відображати шар на мапі. Буде показуватися шар або не буде, залежить від параметрів в адресному рядку або шаблоні.

TYPE [point | line | polygon | circle | annotation | raster | query | chart] – тип шару. Для виразів point|line |polygon – шар буде точковий, лінійний або полігональний. Однак, якщо є shp-файл з полігонами, то можна присвоїти йому тип line. При цьому шар буде відображатися як лінійна тема.

Стосовно представленого раніше адресного рядка браузера, то він складається з кількох частин: перша частина (http: //localhost/cgi-bin/mapserv.exe) – виклик CGI програми mapserv.exe. Решта – це параметри, що передаються програмі mapserv.exe. Виклик програми відділяється від параметрів знаком "?". Параметри передаються парами "параметр = значення", і відокремлюються один від одного знаком "&" наприклад, параметр "map=/ms4w/apps/example/polt.map" вказує MapServer де знаходиться map-файл, на основі якого буде будуватися карта. Параметр "layer = veget" вказує MapServer, що в карту потрібно включити шар veget. І, нарешті, параметр "mode = map" означає, що MapServer повинен послати отриману карту користувачеві відразу, без створення тимчасового зображення на диску.

У випадку коли векторна карта складається з декількох типів об’єктів, то можна задати декілька класів, що відповідають одному шару. В кожному класі визначити різні умовні позначення для одного і того ж шару в залежності від значень атрибутів певного об'єкту. Наприклад, нехай з файлом Vegetation_region.shp пов'язана атрибутивная таблиця у вигляді dbf-файлу. Серед полів dbf-файлу є поле "CodeTopo" в якому зберігається

код об'єкта. Наприклад, код 71100000 відповідає деревної рослинності, код 71314000 – трав'яний, код 72310000 – болотистій місцевості.

```
MAP
  IMAGETYPE      GIF
  EXTENT         34.59 49.58 34.63 49.6
  SIZE           400 300
  SHAPEPATH      "/ms4w/apps/example/shp/"
  IMAGECOLOR     255 255 255
  LAYER
    NAME         veget
    DATA         Vegetation_region
    STATUS       ON
    TYPE         POLYGON
    CLASSITEM    "CodeTopo"
    CLASS
      NAME       "Дерева"
      EXPRESSION '71100000'
      STYLE
        COLOR        12 200 12
      END
    END
    CLASS
      NAME       "Трава"
      EXPRESSION '71314000'
      STYLE
        COLOR        12 255 12
      END
    END
    CLASS
      NAME       "Болота"
      EXPRESSION '72310000'
      STYLE
        COLOR        120 120 255
      END
    END
  END
END
```

Об'єкт LAYER містить параметр CLASSITEM, в якому вказується поле, яке використовується для класифікації об'єктів. В об'єкті CLASS з'являється додатковий параметр EXPRESSION. У найпростішому випадку після слова EXPRESSION вказується рядок, і, якщо атрибут shp-об'єкта

збігається із зазначеною рядком, то shp-об'єкт відносять до даного класу. Замість рядка можна використовувати різні логічні вирази.

Результат запуску MAP-файлу (рис. 2.16):

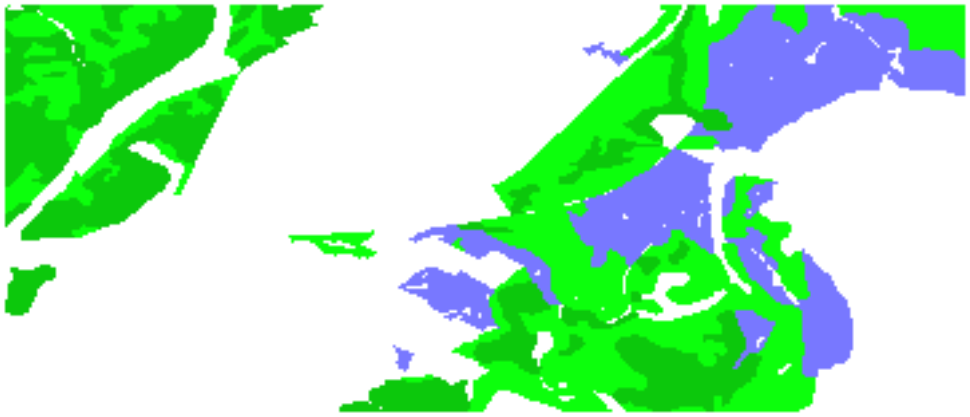

Рисунок 2.16 – Результат запуску MAP-файлу

Для кожного класу можна використовувати власний стиль умовних позначень, для цього в розділі STYLE потрібно вказати параметр SYMBOL. Крім вбудованого набору символів, що задаються за умовчанням, MapServer підтримує також підключення окремих файлів, в яких зібрані описи умовних знаків.

Опису створених умовних знаків зручно зберігати в окремому файлі і посилатися на нього коли це потрібно. Щоб підключити файл, в якому містяться описи умовних знаків, потрібно до об'єкта MAP додати параметр SYMBOLSET "шлях_до_файлу / ім’я_файлу". А в розділі CLASS потрібно зробити посилання на потрібний символ за допомогою SYMBOL "ім’я_сімволу".

```
MAP
  ...
  SYMBOLSET       "./symbols/symbols35.sym"

  LAYER
    ...
    CLASS
      NAME        "Болота"
      EXPRESSION '72310000'
      STYLE
```

```
        SYMBOL 'Line1'
        COLOR  120 120 255
      END
    END
  END
END
```

Таблиця 2.5 – Приклади створення власних умовних позначень

Умовна познака	Код	Пояснення
triangle 	SYMBOL NAME "triangle" TYPE vector POINTS 0 4 2 0 4 4 0 4 END END	В команді POINTS вказуються координати точок (x, y) трикутника в умовній системі координат. Перша точка – початок руху: з точки з координатами (0, 4) проводиться лінія в точку (2, 0), далі лінія піде в точку (4, 4) і остання точка збігається з першою.
cross 	SYMBOL NAME "cross" TYPE vector POINTS 0 0 1 1 -99 -99 0 1 1 0 END END	Хрест, на відміну від трикутника неможливо намалювати одним розчерком пера, не відриваючи його від паперу. Від'ємні координати позначають місце, в якому "перо піднімається".
diagonalfill 	SYMBOL NAME "diagonalfill" TYPE vector TRANSPARENT 0 POINTS	"Діагональна заливка" для полігональних об'єктів. Можна використовувати параметр TRANSPARENT (прозорість): 0 – непрозорий об'єкт, 100 – повністю прозорий.

	0 10 10 0 END END	
dashed1	SYMBOL NAME 'dashed1' TYPE ELLIPSE POINTS 1 1 END FILLED true STYLE 10 5 5 10 END END	"Пунктирна лінія". Вираз "STYLE 10 5 5 10 END" означає, що лінія тягнеться 10 пікселів, потім 5 пікселів розрив, потім лінія з'являється знову (5 пікселів) і знову розрив в 10 пікселів.

Для того, щоб вказати, як підписувати об'єкти, використовується параметр LABELITEM "ім'я_поля", в якому вказується назва поля, що зберігає підписи. У розділі LABEL вказується багато параметрів, але призначення найбільш важливих параметрів (COLOR, SHADOWCOLOR, SIZE, TYPE) зрозуміло.

Якщо виникає проблема з кодуванням, вирішити її можна додавши параметр ENCODING. Для shp-файлів значення кодування – CP1251 (кодування Windows-1251).

До map-об'єкту, до початку описів шарів, потрібно додати параметр FONTSET "./fonts/fonts.list". Файл fonts.list зберігає інформацію про шрифти, які доступні MapServer. Приклад вмісту файлу fonts.list:

```
# Це список шрифтів
arial            arial.ttf
arial-italic         ariali.ttf
```

Файли шрифтів arial.ttf і ariali.ttf можна взяти з стандартного каталогу Windows.

Вміст MAP-файлу з додаванням підписів до об’єктів, а саме назв вулиць, наведено нижче (рис. 2.17):

```
LAYER
    NAME          street
    DATA          Street_polyline
    STATUS        ON
    TYPE          LINE
    LABELITEM     "LatName"    # підписи зберігаються в
полі "LatName"
```

```
  CLASS
    NAME        "Вулиці"
    STYLE
      COLOR         12 12 12
    END
    LABEL
      COLOR 132 31 31
      SHADOWCOLOR 218 218 218
      SHADOWSIZE 2 2
      TYPE TRUETYPE
      FONT arial-italic
      SIZE 7
      ANTIALIAS TRUE
      POSITION CL
      PARTIALS FALSE
      MINDISTANCE 100
      BUFFER 3
      ENCODING CP1251
    END
  END
END
```

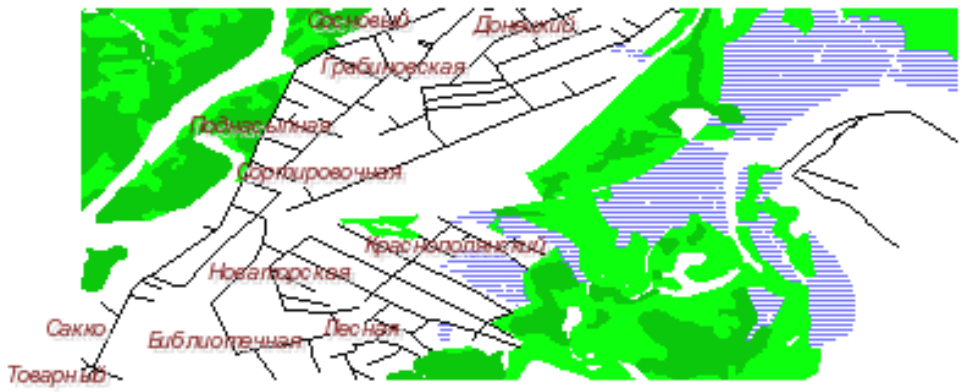

Рисунок 2.17 – MAP-файл з додаванням підписів до об’єктів

2.3.3.2 Створення інтерактивної карти

Додати інтерактивності створеній карті досить просто, для цього потрібно, по-перше, змінити один параметр в адресному рядку: замість

"mode = map" потрібно написати "mode = browse", і, по-друге, написати невеликий html-шаблон, який буде містити карту і елементи оформлення\навігації.

В папку templates потрібно додати файл template.html наступного змісту:

```
<html>
<head>
<title>Пример карты</title>
</head>
<body>
<!-- Начало формы MapServer -->
<form name="mapserv" method="GET" action="/cgi-bin/mapserv.exe">
  <!-- Скрытые CGI переменные -->
  <input type="hidden" name="map" value="[map]">
  <input type="hidden" name="imgext" value="[mapext]">
  <input type="hidden" name="imgxy" value="[center]">
  <input type="hidden" name="layer" value="veget">
  <input type="hidden" name="layer" value="street">
  <input type="hidden" name="layer" value="grid">
  <input type="hidden" name="mode" value="browse">
  <table align="center" width="410" border="1">
  <tr>
    <td colspan="3" align="center" valign="top">
      <input type="image" name="img" src="[img]" width="400" height="300" border="0">
    </td>
  </tr>
</table>
</form>
</body>
</html>
```

Після цього створити в map-файлі ще один розділ:

```
MAP
  ...
  WEB
    TEMPLATE  './templates/template.html'
```

```
    IMAGEPATH '/ms4w/tmp/ms_tmp/'
    IMAGEURL  '/ms_tmp/'
  END
  LAYER
  ...
  END
  ...
END
```

і набрати в рядку адреси:

http://localhost/cgi-bin/mapserv.exe?map=/ms4w/apps/example/polt4.map&layer=veget&layer=street&layer=grid&mode=browse

Рядок TEMPLATE './templates/template.html' вказує, в якому каталозі знаходиться шаблон і як він називається. Після того, як користувач ввів адресу (або клікнув по карті), MapServer обробляє цей шаблон: знаходить CGI змінні (укладені в квадратні дужки) і замінює їх на значення цих змінних. Наприклад, [map] буде замінено на "/ms4w/apps/example/polt4.map", а [mapext] на "615120 5493650 617760 5495760". Тег [img] буде замінений на ім'я створеного тимчасового файлу, в якому зберігається графічне зображення карти (наприклад, "\ms4w\tmp\ms_tmp\ MS11877753442492.gif"). Після цього знову отриманий html-файл відсилається користувачеві.

Таким чином, при натисканні мишкою на карті, MapServer буде зсувати зображення так, що точка в яку було зроблено клацання мишкою, виявлялася в центрі карти.

Розглянемо приклад шаблона, який дозволяє створити інструменти масштабування карти.

```
<html>
  <head>
  <title>Пример карты</title>
  </head>
  <body>
  <!-- Начало формы MapServer -->
  <form name="mapserv" method="GET" action="/cgi-
bin/mapserv.exe">
      <input type="hidden" name="map" value="[map]">
```

```
      <input type="hidden" name="imgext"
value="[mapext]">
      <input type="hidden" name="imgxy"
value="[center]">
      <input type="hidden" name="layer" value="veget">
      <input type="hidden" name="layer"
value="street">
      <input type="hidden" name="layer" value="grid">
      <input type="hidden" name="mode" value="browse">
      <table align="center" width="680" border="0">
      <tr>
        <td>
            <table width="400" border="1">
            <tr>
                <td>
                <!-- SPECIFY MAP MODE -->
                <div align="center">Map Mode:<br>
                    <select name="mode">
                    <option
value="browse">Browse</option>
                    <option value="map">Map</option>
                    </select>
                </div>
                </td>
                <td>
                <!-- FORM SUBMIT BUTTON -->
                <div align="center">
                    <input type="submit"
name="submit"

value="Refresh">
                    </div>
                </td>
                <td>
                <!-- ZOOM/PAN CONTROLS -->
                <div align="center">Map Control: <br>
                    <select name="zoom">
```

```
                        <option value="4"
[zoom_4_select]>

Zoom In 4x</option>
                        <option value="3"
[zoom_3_select]>

Zoom In 3x</option>
                        <option value="2"
[zoom_2_select]>

Zoom In 2x</option>
                        <option value="1"

[zoom_1_select]>Recenter</option>
                        <option value="-2"
                                [zoom_-
2_select]>Zoom Out 2x</option>
                        <option value="-3"
                                [zoom_-
3_select]>Zoom Out 3x</option>
                        <option value="-4"
                                [zoom_-
4_select]>Zoom Out 4x</option>
                    </select>
                </div>
                </td>
            </tr>
            <tr>
            <!-- DISPLAY THE MAPSERVER-CREATED MAP
IMAGE -->
                <td colspan="3" align="center"
valign="top">
                    <input type="image" name="img"
src="[img]" width="400" height="300" border="0">
                </td>
            </tr>
        </table>
```

```
      </td>
  </tr>
  </table>
  </form>
  </body>
  </html>
```

В розділі "ZOOM/PAN CONTROLS" наводиться приклад того, як використовується змінна "zoom". Якщо її значення дорівнює 1, то масштаб карти при натисканні кнопки "Refresh" не змінюється, якщо значення змінної zoom позитивне, то масштаб збільшується в zoom раз. Якщо zoom <0 то масштаб карти зменшується в zoom раз.

Для додавання масштабної лінійки треба створити відповідний розділ MAP-файлу:

```
MAP
...
  SCALEBAR
    IMAGECOLOR 255 255 255
    LABEL
      COLOR 0 0 0
      SIZE TINY
    END
    STYLE 1
    SIZE 100 2
    COLOR 0 0 0
    UNITS METERS
    INTERVALS 2
    TRANSPARENT FALSE
    STATUS ON
  END

  LAYER
  ...
  END
...
END
```

Після цього додати в html-шаблон ще одну комірку таблиці:

```
<td align="right"><img src="[scalebar]"></td>
```

2.3.4 GeoServer

Також, як і MapServer, GeoServer є картографічним сервером з відкритим вихідним кодом, який серед багатьох інших можливостей, реалізує наступні специфікації OGS: WMS, WFS, WCS. Однак, на відміну від MapServer, GeoServer реалізує специфікацію WFS-T (WFS-Transaction). Це означає, що використовуючи GeoServer, можна не тільки отримувати дані для побудови на їх основі власних карт, але також редагувати отримані дані з подальшим автоматичним оновленням вихідної інформації на сервері. Серед підтримуваних форматів можна виділити: JPEG, PNG, SVG, KML / KMZ, GML, PDF, ESRI Shapefile і інші.

Іншою цікавою особливістю, що відрізняє GeoServer від MapServer або FeatureServer (ще одного картографічного сервера з відкритим вихідним кодом), є візуальна система управління файлами налаштувань і опису даних для проектів GeoServer, що поставляється з GeoServer. Ця система реалізована у вигляді веб-інтерфейсу REST API і надає користувачеві можливість інтерактивного створення та зміни розроблювального картографічного ресурсу.

GeoServer – це Java-сервлет. Він працює всередині контейнера сервлетів, наприклад Apache Tomcat. Його також можна використовувати автономно за допомогою вбудованого контейнера сервлетів Jetty, що дуже зручно при локальному тестуванні.

Веб-інтерфейс GeoServer наведено на рис. 2.18.

Розділ About & Status містить загальну інформацію про runtime змінні та дозволяє перевіряти журнали, щоб досліджувати помилки та попередження GeoServer під час виконання запиту клієнта. Посилання "Server Status" відкриває форму, яка містить основні параметри конфігурації та інформацію про поточний стан GeoServer. Секція "GeoServer Logs" дозволяє читати повідомлення, попередження та помилки, що містяться у log-файлі. На панелі "Contact Information" можна додати контактну інформацію про організацію та осіб, які керують сервісом. За посиланням "About GeoServer" міститься інформації про збірку, шляхи пошуку документації GeoServer, відстежувача помилок та wiki.

Рисунок 2.18 – Веб-інтерфейс GeoServer

Розділ Data містить посилання на механізм конфігурації даних. В цій області можна налаштувати доступ до даних і спосіб, яким вони надаються клієнтам. В розділі відкривається форма, що включає перелік шарів, опублікованих на GeoServer (рис. 2.19). Крім того, якщо жодних даних не додано, все одно можна знайти кілька зразків шарів, що вже перелічені. Клацнувши на посилання OpenLayers, яке розміщено праворуч від імені шару, можна відкрити зразок веб-програми, щоб подивитися, як дані виглядають. Посилання KML дозволяють завантажувати дані в формат,

придатний для попереднього перегляду на Google Планета Земля. Є також кілька інших доступних форматів, перелічених у спадному списку.

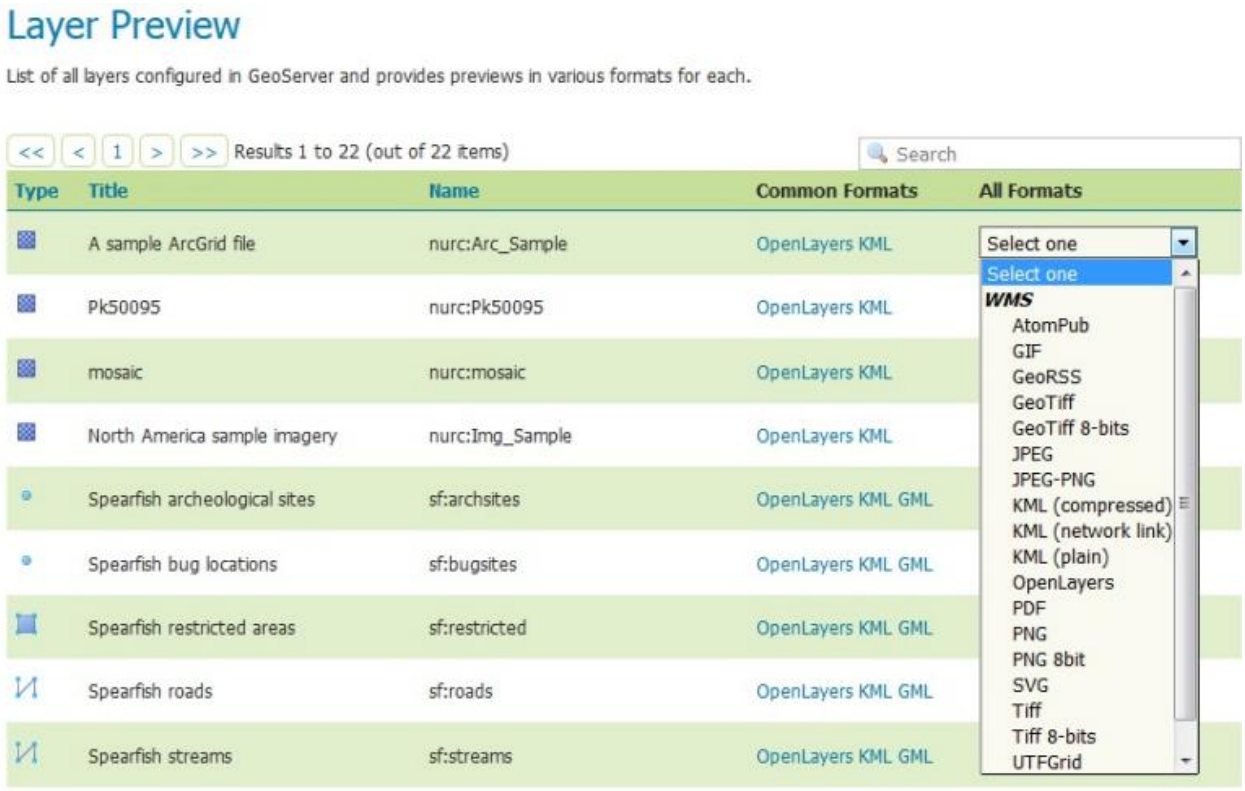

Рисунок 2.19 – Вигляд секції Layer Preview

ГЛОСАРІЙ

EPSG – рефікс простору імен, відноситься до геодезичного набору даних групи European Petroleum Survey Group (EPSG), який визначає числові ідентифікатори для багатьох загальних систем координат. ПРИКЛАД EPSG: 4326 відноситься до системи координат WGS 84.

Geography Markup Language (GML) – це XML-подібний код для транспортування та зберігання географічної інформації, включаючи як геометрію, так і властивості географічних об’єктів. Це відкритий стандарт обміну даними, який добре підходить для передачі малого та середнього обсягу інформації.

Інтерфейс – це програмне забезпечення, яке дозволяє незалежним системам взаємодіяти один з одним.

Інтероперабельність – це здатність продуктів, систем або бізнес-процесів працювати разом для досягнення спільного завдання. Термін може бути визначений технічним способом або більш широко, з врахуванням соціальних, політичних та організаційних факторів.

ISO/TC 211–стандартний технічний комітет, сформований в рамках Міжнародної організації зі стандартизації (ISO), якому доручено охоплювати сфери цифрової географічної інформації та геомантики. Він відповідає за підготовку серії міжнародних стандартів і технічних специфікацій, пронумерованих в діапазоні, починаючи з 19101.

Метадані – це "дані про дані", структуровані, закодовані дані, що описують характеристики сутності, що несе інформацію, для допомоги в ідентифікації, виявленні, оцінці та управлінні описаними сутностями.

Open Geospatial Consortium (OGC) – є міжнародним консорціумом з 333 компаній, урядових установ та університетів, які беруть участь у процесі консенсусу щодо розробки загальнодоступних специфікацій, які підтримують сумісні рішення, що "вмикають гео" Інтернет, послуги бездротового зв'язку та локації, а також загальноприйняті IT. Специфікації дозволяють розробникам технологій робити складну просторову інформацію та послуги доступними та корисними для всіх видів додатків.

OpenGIS – є зареєстрованою торговою маркою OGC і є торговою маркою, пов'язаною зі специфікаціями та документами, які вироблені OGC. Специфікації OpenGIS є основними "продуктами" OGC і були розроблені членом для вирішення конкретних проблем інтероперабельності.

Request for Comments (RFC)– відносно Інтернету та інженерії комп'ютерних мереж, документи Request for Comments (RFC) являють собою ряд меморандумів, що охоплюють нові дослідження, інновації та методології, які застосовуються до Інтернет-технологій.

Styled Layer Descriptor (SLD) – це кодування, яке розширює специфікацію сервісу WMS, щоб дозволити визначену користувачем символізацію даних об’єкта. Це дозволяє користувачам (або іншим системам) визначати, які елементи або шари відображаються за допомогою яких кольорів або символів.

Uniform Resource Locator (URL) – це компактний рядок символів, що використовується для ідентифікації ресурсу. Це дозволяє взаємодіяти з представленням ресурсу через мережу за допомогою конкретних протоколів.

Web Feature Service – це інтерфейс, що дозволяє запитувати географічні об’єкти в Інтернеті, використовуючи незалежні від платформи виклики (результат – GML-файл). Інтерфейс WMS повертає лише зображення, яке неможливо відредагувати або просторово проаналізувати.

Веб-картографування – це перегляд географічної інформації через Інтернет, включаючи презентацію карт загального призначення для відображення місцеположень та географічних фонів.

Web Map Service – це стандарт, який забезпечує стандартний інтерфейс для запитів та доступу до шарів карт з картографічного сервісу.

XML – розшифровується як eXtensible Markup Language, розширювана мова розмітки, що запропонована консорціумом W3C, стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML. XML-документ складається із текстових знаків, і придатний до читання людиною.

Навчальне електронне видання

КУЗНІЧЕНКО Світлана Дмитрівна
БУЧИНСЬКА Ірина Вікторівна

" КАРТОГРАФІЧНІ ВЕБ-СЕРВІСИ "

Конспект лекцій

Видавець і виготовлювач
Одеський державний екологічний університет
вул. Львівська, 15, м. Одеса, 65016
тел./факс: (0482) 32-67-35
E-mail: info@odeku.edu.ua
Свідоцтво суб'єкта видавничої справи
ДК № 5242 від 08.11.2016