2. ЗАСОБИ ПУБЛІКАЦІЇ ТА ПРЕДСТАВЛЕННЯ ГЕОПРОСТОРОВИХ ДАНИХ В ІНТЕРНЕТ

Андрій Колотій, старший викладач[1], старший науковий співробітник[2]
[1] Кафедра математичного моделювання та аналізу даних
Навчально-науковий Фізико-технічний інститут
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
[2] Відділ космічних інформаційних технологій і систем
Інститут космічних досліджень НАН України та ДКА України

andrew.k.911@gmail.com

ВСТУП

Геопросторові дані відіграють ключову роль у багатьох сучасних сферах діяльності, від управління природними ресурсами до міського планування та екологічного моніторингу. Їх належна обробка, зберігання та представлення стали важливими завданнями для багатьох організацій та наукових установ. Зокрема, значне місце в цих процесах займають геопортали — веб-системи, які надають користувачам зручний доступ до різноманітних картографічних ресурсів і можливість інтерактивної взаємодії з ними.

Розвиток технологій, поява відкритих даних та вдосконалення мережевих інструментів значно спростили процеси публікації та візуалізації геопросторових даних в Інтернеті. Це призвело до зростання популярності геопорталів, які дозволяють користувачам без прив'язки до конкретного місця працювати з великими обсягами геоінформації. Інструменти геопорталів знаходять застосування в таких сферах, як моніторинг навколишнього середовища, розробка міської інфраструктури, управління ресурсами та створення систем «розумних міст».

В даному розділі наведено огляд сучасних підходів до публікації геопросторових даних через Інтернет, аналіз ключових компонентів та технологій для створення ефективних геопорталів, а також визначення основних факторів, що впливають на вибір відповідних технологічних рішень для їх реалізації.

2.1. ОГЛЯД ПОРТАЛІВ

Геопортали стали невід'ємною частиною сучасних інформаційних систем [1], які надають можливість зберігати, візуалізувати та аналізувати геопросторові дані без прив’язки до конкретного робочого місця. Вони є інструментами, що дозволяють об'єднати різноманітні джерела даних у зручному Веб-інтерфейсі, надаючи користувачам доступ до картографічної інформації, супутникових знімків, даних про землекористування тощо [2]. Сучасні геопортали використовуються в різних галузях, включаючи управління природними ресурсами, планування міської інфраструктури, екологічний моніторинг та застосунки розумного міста [3]. Завдяки розвитку технологій і доступності відкритих даних, створення геопорталів стало доступнішим, що сприяє їх широкому застосуванню. У цьому розділі розглядаються ключові компоненти та технології, які використовуються для створення геопорталів, а також фактори, що впливають на вибір технологічної платформи для їх реалізації.

Геопортали є потужними інструментами для зберігання, візуалізації та аналізу геопросторових даних. Вони об'єднують дані з різних джерел, надаючи користувачам можливість переглядати, аналізувати та взаємодіяти з цими даними. Розглянемо основні компоненти, які впливають на вибір тієї чи іншої технологічної платформи.

Першим кроком у створенні геопорталу *є визначення його цілей та вимог.* Необхідно чітко зрозуміти, які задачі має вирішувати геопортал. Це можуть бути такі завдання, як навігація, екологічний моніторинг, управління ресурсами, або вирішення питань, пов'язаних з розумними містами. Важливо також визначити основну *аудиторію користувачів* геопорталу. Це можуть бути громадські організації, державні установи, комерційні компанії або звичайні громадяни. Від цього залежить набір функцій, які повинні бути реалізовані, такі як пошук, фільтрація, аналіз даних та інтеграція з іншими сервісами.

Наступним етапом є збір та підготовка даних. Слід визначити, які саме дані потрібні для досягнення поставлених цілей, а також знайти відповідні джерела для їх отримання. Це можуть бути відкриті дані, комерційні дані, дані від державних установ або результати власних досліджень. Необхідно також врахувати типи даних, які будуть використовуватися, наприклад, векторні дані (точки, лінії, полігони) та растрові дані (супутникові знімки, карти

висот, аерофотозйомка). Дані можуть бути представлені у різних *форматах*, таких як Shapefile, GeoJSON, KML, GeoTIFF.

Архітектура геопорталу складається з кількох важливих компонентів. *Серверна частина* відповідає за обробку даних та управління запитами. Популярними рішеннями з відкритим вихідним кодом є GeoServer, MapServer та QGIS Server [4], які використовуються для публікації геопросторових даних. Для зберігання геопросторових даних часто використовують бази даних, зокрема PostGIS — розширення PostgreSQL. *Технології кешування*, такі як GeoWebCache або MapProxy, використовуються для підвищення швидкості завантаження карт та зменшення навантаження на обчислювальні ресурси. Стандартизовані API та сервіси, наприклад ті, що задовольняють OGC стандартам (WMS, WFS, WCS), забезпечують доступ до даних для інших систем і додатків. Сама можливість використання різноманітних картографічних сервісів стала знаковою віхою розвитку геопорталів [5].

Інтерфейс користувача є одним з ключових елементів геопорталу. В його основі зазвичай лежить Веб-додаток з інтуїтивно зрозумілим інтерфейсом. На сьогодні стандартом галузі є *JavaScript-бібліотеки*, такі як *Leaflet, OpenLayers або Mapbox GL JS*, які дозволяють створювати інтерактивні карти. Дизайн користувацького інтерфейсу (UX/UI дизайн) має забезпечувати зручність використання, доступність та привабливий зовнішній вигляд.

Інтеграція з іншими системами та розширюваність функціоналу є важливими аспектами для забезпечення гнучкості та масштабованості геопорталу. Геопортал має бути здатним, принаймні на рівні архітектури, інтегруватися з іншими інформаційними системами, такими як ERP, CRM, та забезпечувати можливість додавання нових функцій та модулів у майбутньому.

Останнім, але не менш важливим аспектом є *безпека та доступність*. Необхідно забезпечити захист даних та доступність лише для авторизованих користувачів, а також можливість *гнучкого керування доступом* до різних функцій порталу. Геопортал повинен бути доступним на різних пристроях і відповідати *вимогам доступності* для людей з особливими потребами.

Якщо підсумувати поточний *стан розвитку* популярних технологій [6] для створення геопорталів, то популярними технологіями та інструментами є наступні.

Сервери карт: GeoServer, MapServer, QGIS Server

Бази даних: PostGIS, MongoDB, CouchDB
Фронтенд-бібліотеки: Leaflet, OpenLayers, Mapbox GL JS тощо
Картографічні сервіси: Google Maps API, ArcGIS Online, Bing Maps

Нижче більш детально розглянуті основні серверні інструменти, клієнтські бібліотеки та «коробочні» (майже готові до використання) рішення на прикладі Mapbender.

2.2. СЕРВЕРНА ЧАСТИНА

2.2.1. GEOSERVER

GeoServer [7] — це популярний сервер з відкритим вихідним кодом для публікації та управління геопросторовими даними. Дистрибутиви програмного забезпечення доступні на сторінці проєкту Geoserver (рис. 1). Він дозволяє користувачам обмінюватись, обробляти і редагувати геопросторову інформацію відповідно до стандартів OGC (Open Geospatial Consortium), таких як WMS (Web Map Service), WFS (Web Feature Service), WCS (Web Coverage Service) тощо.

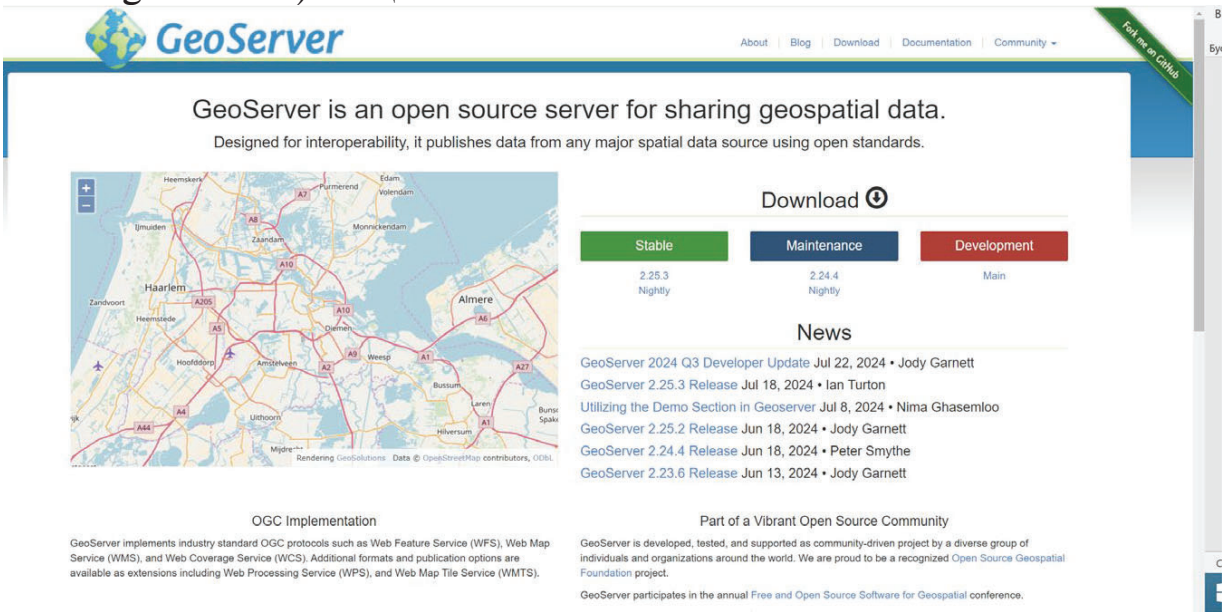


Рис. 1. Стартова сторінка проєкту GeoServer

GeoServer складається з кількох ключових компонентів, які забезпечують його функціональність та масштабованість.

Архітектура та ядро базуються на Java-технологіях, що дозволяє запускати GeoServer в будь-якому середовищі, яке підтримує JVM (Java Virtual Machine). Для управління конфігурацією та забезпечення гнучкості обробки Веб-запитів

GeoServer використовує Spring Framework [8]. Для роботи з геопросторовими даними використовується бібліотека GeoTools [9], яка підтримує багато форматів даних та операцій з геоданими.

Підтримка стандартів OGC [10] є важливим аспектом GeoServer. Сервіс WMS дозволяє динамічно генерувати карти з геопросторових даних, підтримуючи різні формати виводу (PNG, JPEG, GIF, SVG, PDF). Сервіс WFS забезпечує доступ до векторних даних і маніпуляцію ними у таких форматах, як GML та GeoJSON. Сервіс WCS надає доступ до растрових даних, таких як супутникові знімки, а *Web Processing Service* (WPS) дозволяє виконувати просторовий аналіз та обробку даних на сервері.

Управління даними в GeoServer включає можливість підключення до різноманітних баз даних, таких як PostGIS, Oracle Spatial, MySQL та MS SQL Server. GeoServer також підтримує роботу з основними форматами геопросторових даних, включаючи Shapefile, GeoTIFF, KML, CSV та GeoJSON. Управління джерелами даних, стилями та сервісами здійснюється через Веб-інтерфейс.

Конфігурація та налаштування здійснюються за допомогою інтуїтивно зрозумілої адміністративної панелі, яка дозволяє налаштовувати джерела даних, стилі, користувачів та політики доступу. Для налаштування сервера та його сервісів також можна використовувати XML-файли. GeoServer надає RESTful API для автоматизації завдань управління та конфігурації.

Безпека забезпечується завдяки підтримці різних механізмів аутентифікації, таких як LDAP та JDBC, а також можливості обмеження доступу до даних на рівні користувача або групи. Для захисту даних під час передачі підтримуються безпечні з'єднання за допомогою SSL/TLS.

Масштабованість і продуктивність GeoServer підвищуються завдяки використанню GeoWebCache для кешування растрових фрагментів і прискорення рендерингу карт. Крім того, GeoServer підтримує кластерну роботу, що дозволяє розподіляти обробку даних і підвищувати надійність (відмовостійкість) системи.

Інтеграція з іншими системами забезпечується завдяки підтримці плагінів та розширень, які дозволяють інтегрувати GeoServer з іншими сервісами та розширювати його функціональність. Наприклад, інтеграція з GDAL дозволяє підтримувати додаткові формати даних. GeoServer також легко взаємодіє з іншими GIS-системами, такими як QGIS та ArcGIS.

GeoServer є досить потужним, гнучким та універсальним рішенням для створення геопорталів, що дозволяє інтегрувати та

візуалізувати геопросторові дані з різних джерел. Його архітектура забезпечує високу продуктивність, масштабованість та безпеку, що робить GeoServer відмінним вибором для різноманітних застосувань у галузі геоінформаційних систем.

2.2.2. MAPSERVER

MapServer [11] — ще один з популярних серверів з відкритим вихідним кодом для публікації геопросторових даних у Веб-середовищі. Дистрибутиви програмного забезпечення доступні на сторінці проєкту MapServer (рис. 2). Він надає можливості для побудови карт і публікації географічних даних, а також підтримує різноманітні Інтернет-протоколи та формати. MapServer відомий своєю високою продуктивністю та гнучкістю в налаштуванні.

Серверна частина MapServer складається з кількох основних компонентів, кожен з яких забезпечує його функціональність та інтеграційні можливості.

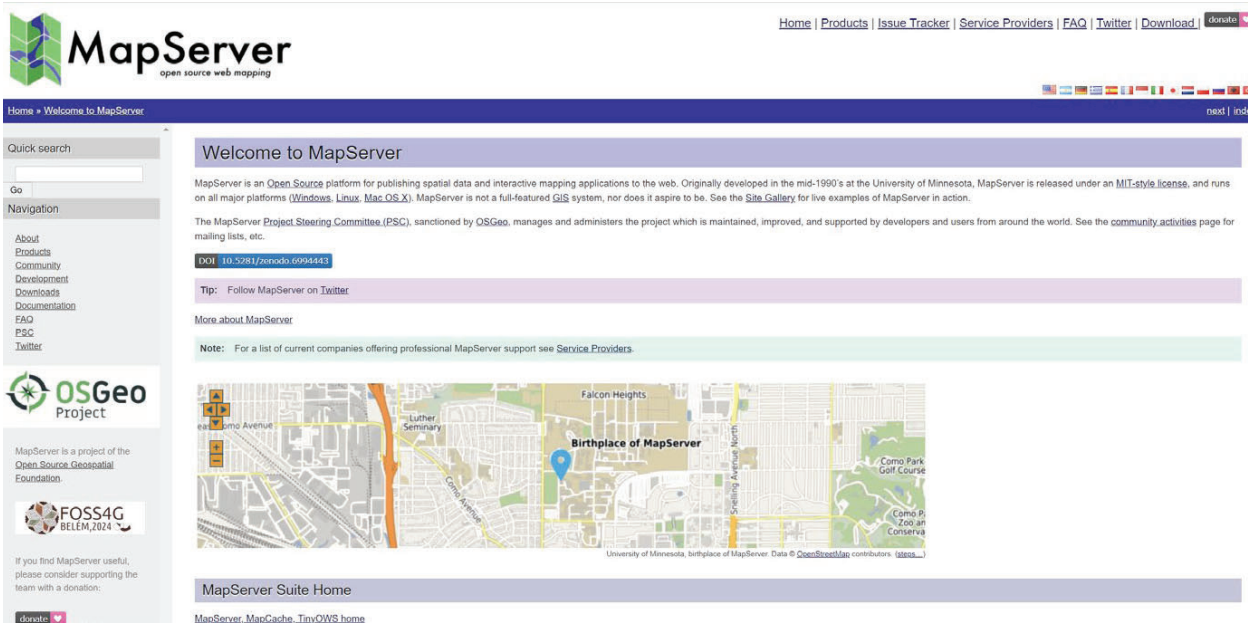


Рис. 2. Стартова сторінка проєкту MapServer

Архітектура та ядро MapServer побудовані на мовах програмування C і C++, що дозволяє досягати високої продуктивності та ефективного використання ресурсів. MapServer може функціонувати як CGI-додаток або через MapScript, що забезпечує його сумісність із різними середовищами та інтеграцію з іншими мовами програмування, такими як Python, PHP, Perl, Ruby, Java та C#.

Підтримка стандартів OGC є важливою частиною MapServer. Він забезпечує динамічне створення картографічних зображень через WMS сервіс, підтримуючи різні формати виводу, такі як PNG, JPEG, GIF, SVG та PDF. WFS дозволяє доступ до векторних геоданих у форматах GML та GeoJSON, а WCS надає доступ до растрових даних, таких як супутникові знімки та моделі висот. Крім того, MapServer підтримує WPS для виконання геопросторових обчислень і аналізів на сервері, а також Tile Map Service (TMS) і Web Map Tile Service (WMTS) для створення тайлів (фрагментів растрових зображень), що підвищує швидкість передачі та відображення карт.

Управління даними в MapServer здійснюється за допомогою підтримки різноманітних файлових форматів, включаючи Shapefile, GeoTIFF, та інші формати, сумісні з OGR [12] (частина бібліотеки GDAL (Geospatial Data Abstraction Library) для роботи з векторними даними), такі як CSV, KML та GML. MapServer підтримує підключення до баз даних, таких як PostGIS, Oracle Spatial і MySQL, що дозволяє обробляти векторні та растрові дані, а також накладати їх один на одного.

Конфігурація та налаштування здійснюються через основний конфігураційний файл MapFile, який описує джерела даних, шари, стилі та проєкції. MapFile використовується для налаштування відображення карт і визначення параметрів роботи сервера. Крім того, MapServer підтримує складні стилі та символи для візуалізації геопросторових даних, включаючи використання Styled Layer Descriptor (SLD).

Безпека в MapServer забезпечується через налаштування контролю доступу до даних і сервісів за допомогою аутентифікації та авторизації, а також підтримку шифрування з'єднань за допомогою HTTPS для захисту даних під час передачі.

Масштабованість і продуктивність досягаються завдяки використанню зовнішніх інструментів для кешування картографічних елементів (прямокутних фрагментів карти або тайлів), таких як MapCache, що підвищує швидкість відображення карт. Крім того, оптимізація конфігураційних файлів і налаштувань сервера дозволяє покращити загальну продуктивність системи.

Інтеграція з іншими системами здійснюється через MapScript API, який дозволяє інтегрувати MapServer з іншими програмами та мовами програмування, такими як Python, PHP, Perl, Ruby, Java, C#. Крім того, MapServer підтримує плагіни для розширення функціональності та інтеграції з іншими системами.

MapServer є потужним та гнучким інструментом для публікації геопросторових даних у Веб-середовищі, але вже на мові C/C++. Він забезпечує високу продуктивність, підтримку стандартів OGC та можливість інтеграції з різними системами і мовами програмування. Це робить MapServer ідеальним вибором для створення геопорталів і геоінформаційних систем різної складності, коли акцент робиться на швидкодію.

2.2.3. QGIS SERVER

QGIS Server [13] — це серверний компонент з відкритим вихідним кодом, який дозволяє публікувати геопросторові дані і сервіси на основі проєктів QGIS. Дистрибутиви програмного забезпечення доступні на сторінці проєкту MapServer (рис. 3). QGIS Server надає можливість створювати картографічні продкути для Веб та геоінформаційні сервіси, підтримуючи стандарти OGC, такі як WMS, WFS, WCS та ін. Завдяки QGIS Server можна легко розгортати комплексні ГІС-рішення, використовуючи переваги QGIS Desktop для попередньої підготовки даних і проєктів [14].

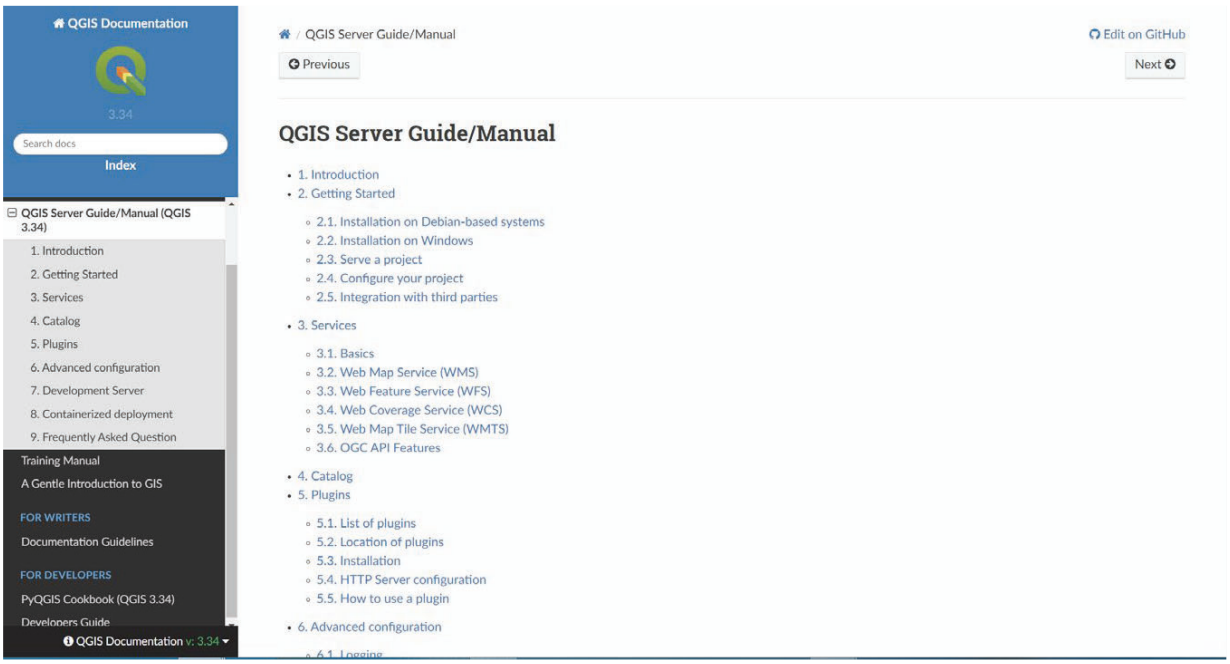


Рис. 3. Стартова сторінка проєкту QGIS Server

Серверна частина QGIS Server складається з кількох основних компонентів, які забезпечують його функціональність та гнучкість у використанні.

Архітектура та ядро QGIS Server побудовані на C++ і підтримують розширення через Python, що дозволяє досягати

високої продуктивності та гнучкості в налаштуваннях. QGIS Server використовує ядро QGIS, яке забезпечує доступ до всіх функцій і можливостей QGIS Desktop для обробки та візуалізації даних. Крім того, QGIS Server підтримує QGIS Processing Framework, що надає велику кількість інструментів для обробки даних, включаючи обчислення, аналіз та трансформацію даних.

Підтримка стандартів OGC в QGIS Server включає кілька важливих сервісів. WMS дозволяє створювати та публікувати картографічні зображення в різних форматах, таких як PNG, JPEG, SVG та PDF, із підтримкою стилізації шарів. WFS забезпечує доступ до векторних геоданих у форматах GML та GeoJSON, а WCS надає доступ до растрових даних, таких як супутникові знімки та моделі висот. QGIS Server також підтримує WPS для серверної обробки даних з використанням геоалгоритмів із QGIS Processing Toolbox, а також WMTS для публікації карт у вигляді набору растрових фрагментів, що підвищує швидкість рендерингу та відображення карт.

Управління даними в QGIS Server здійснюється за допомогою підтримки підключення до різних геопросторових баз даних, таких як PostGIS, Oracle Spatial, MySQL та SpatiaLite. Сервер також працює з широким спектром геопросторових форматів, включаючи Shapefile, GeoTIFF, KML, GML, GPX, GeoJSON та багато інших. Крім того, QGIS Server підтримує роботу з різними проєкціями і можливість трансформації між ними в реальному часі.

Конфігурація та налаштування QGIS Server базуються на використанні проєктів QGIS (.qgs/.qgz), які визначають шари, стилі, символогію та загальний вигляд карт. Це дозволяє використовувати всі можливості QGIS Desktop без необхідності додаткових налаштувань сервера. Підтримка SLD (Styled Layer Descriptor) і QML (QGIS Layer Style File) дозволяє детально налаштовувати відображення даних, а можливість налаштування серверних проєктів через графічний інтерфейс QGIS Desktop робить процес конфігурації простим і зручним.

Безпека QGIS Server забезпечується за допомогою різних механізмів аутентифікації та авторизації, таких як базова аутентифікація, OAuth2 і LDAP. Сервер також дозволяє налаштовувати права доступу до різних шарів і сервісів для різних груп користувачів, а підтримка SSL/TLS забезпечує шифрування з'єднань для захисту даних під час передачі.

Масштабованість і продуктивність QGIS Server досягається через оптимізацію рендерингу карт, використання кешування та

інтеграцію з зовнішніми системами кешування для зменшення навантаження на сервер. Крім того, сервер підтримує розподілені обчислення і кластеризацію, що забезпечує високу доступність та надійність сервісів.

Інтеграція з іншими системами в QGIS Server можлива завдяки підтримці плагінів Python, які дозволяють розширювати функціональність сервера та інтегрувати його з іншими системами. Використання PyQGIS надає програмний доступ до функціональності QGIS Server, що дозволяє автоматизувати процеси та інтегрувати різні системи. Сервер також легко інтегрується з іншими ГІС-системами, такими як QGIS Desktop, ArcGIS, GRASS GIS та інші інструменти.

QGIS Server є потужним і гнучким інструментом для публікації геопросторових даних та картографічних сервісів, але має додаткові вимоги до проєктів, які може підтримувати. Його тісна інтеграція з QGIS Desktop робить його ідеальним рішенням для організацій, які вже використовують QGIS у своїй роботі. Завдяки підтримці стандартів OGC і широким можливостям налаштування, QGIS Server може бути використаний для створення складних геоінформаційних систем, забезпечуючи високу продуктивність і надійність.

2.3. КЛІЄНТСЬКІ БІБЛІОТЕКИ

2.3.1. LEAFLET

Leaflet [15] — це легка і проста у використанні бібліотека JavaScript, яка використовується для створення інтерактивних карт у веб-додатках, розроблена в Україні. Посилання на документацію та дистрибутив бібліотеки доступні на сторінці проєкту Leaflet (рис. 4). Завдяки своїй гнучкості і потужним можливостям Leaflet став популярним інструментом для розробки геопорталів, які надають користувачам можливість взаємодії з геопросторовими даними через Веб-інтерфейс [16].

Бібліотека Leaflet пропонує широкий спектр можливостей, що робить її популярним вибором для розробників інтерактивних карт.

Легкість і швидкодія Leaflet досягається завдяки малому розміру, всього близько 39KB (gzip), що робить цю бібліотеку майже ідеальним рішенням для використання на Веб-сайтах з обмеженими ресурсами. Завдяки оптимізованому рендерингу та

кешуванню, Leaflet забезпечує високу швидкість роботи навіть на мобільних пристроях.

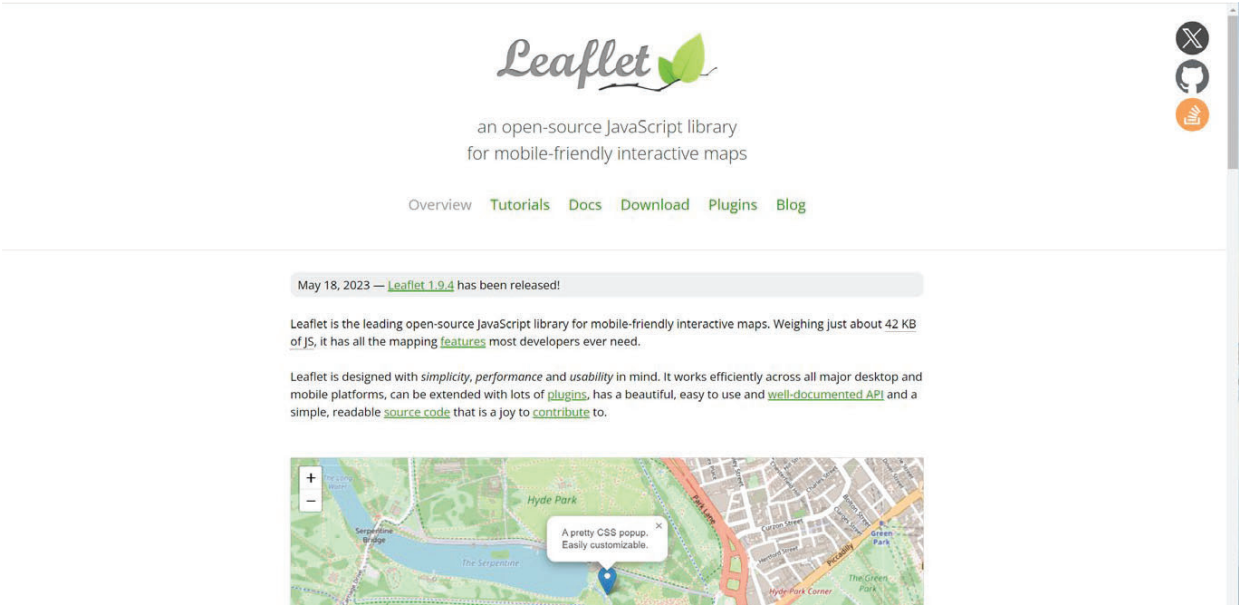


Рис. 4. Стартова сторінка проєкту Leaflet

Підтримка різноманітних форматів даних дозволяє працювати з векторними шарами, такими як GeoJSON, які легко візуалізувати і стилізувати. Leaflet також підтримує додавання картографічних зображень з популярних картографічних сервісів, таких як OpenStreetMap, Google Maps, Mapbox тощо. Крім того, бібліотека надає можливість кластеризації, що дозволяє об'єднувати велику кількість точок даних у кластери для покращення візуалізації.

Інтерактивність є однією з ключових особливостей Leaflet. Бібліотека підтримує різноманітні події, такі як клацання миші, наведення курсору, масштабування, що дозволяє створювати динамічні та інтерактивні карти. Leaflet також дозволяє додавати інтерактивні елементи (попапи і тултіпи) для відображення інформації при натисканні на об'єкти на карті, а також легко управляти шарами і контролерами, що дає змогу користувачам перемикатися між різними наборами даних.

Масштабованість та розширюваність Leaflet забезпечується завдяки великій кількості доступних плагінів, які розширюють функціональні можливості бібліотеки, додаючи підтримку графіків, маршрутизації, анімацій і багато іншого. Крім того, бібліотека пропонує широкі можливості для кастомізації, включаючи створення власних іконок, стилів для ліній і полігонів.

Відкритий код і спільнота є важливими аспектами Leaflet. Це проєкт з відкритим кодом, що робить його доступним для

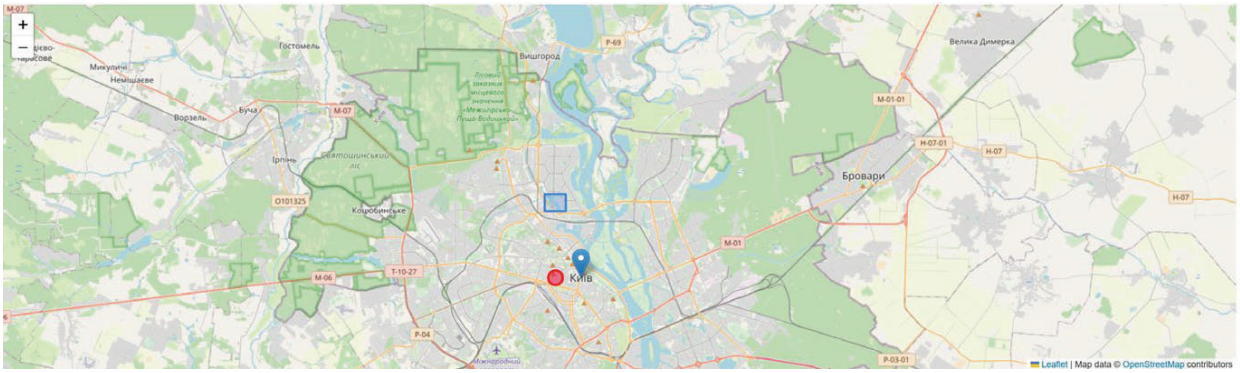
модифікацій і вдосконалень від спільноти розробників. Завдяки широкій та активній спільноті користувачів і розробників, створюються додаткові інструменти та забезпечується підтримка і розвиток бібліотеки.

Приклади використання Leaflet для створення геопорталів. Розглянемо приклад розробки простого Веб-додатка з картою на базі Open Street Map. Почнемо з декларації DOCTYPE для HTML. У заголовковій частині (блок **head**) підключені метатеги для встановлення кодування сторінки (UTF-8) та налаштування для адаптивного дизайну (viewport). Підключено CSS бібліотеки Leaflet для стилізації карти та налаштування висоти елемента карти через стиль.

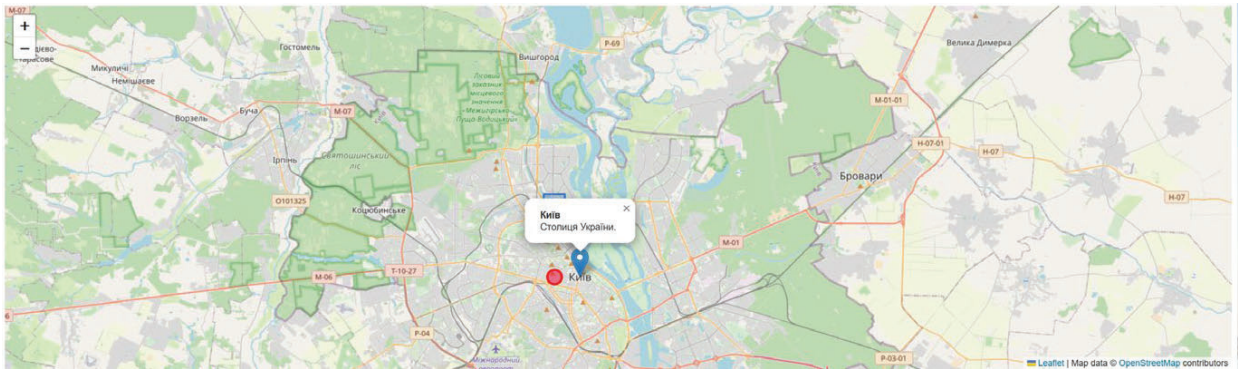
Тіло сторінки (блок **body**) включає div елемент з ідентифікатором «map», який буде використовуватись як контейнер для відображення карти. Підключається JavaScript файл бібліотеки Leaflet, яка використовується для роботи з картами. В **блоці JavaScript коду** ініціалізується карта з центром на координатах Києва та заданим рівнем масштабування. Додається шар картографічного зображення з OpenStreetMap для відображення карти. Потім додаються різні геооб'єкти: маркер, круг і полігон, кожен з яких має власне спливаюче вікно із текстом. Результат представлений на рис. 5.

```
<!DOCTYPE html>
<html>
<head>
   <title>Simple Map</title>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
   <style>
      #map { height: 500px; }
   </style>
</head>
<body>
   <div id="map"></div>

   <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
   <script>
      // Ініціалізація карти
```

```
        var map = L.map('map').setView([50.4501, 30.5234], 13); // Координати Києва

        // Додавання картографічного зображення з OpenStreetMap
        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
            maxZoom: 19,
            attribution: 'Map data © <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
        }).addTo(map);

        // Додавання маркера
        var marker = L.marker([50.4501, 30.5234]).addTo(map)
            .bindPopup('<b>Київ</b><br>Столиця України.')
            .openPopup();

        // Додавання круга
        var circle = L.circle([50.4501, 30.5], {
            color: 'red',
            fillColor: '#f03',
            fillOpacity: 0.5,
            radius: 500
        }).addTo(map).bindPopup('Це круг.');

        // Додавання полігона
        var polygon = L.polygon([
            [50.49, 30.49],
            [50.50, 30.49],
            [50.50, 30.51],
            [50.49, 30.51]
        ]).addTo(map).bindPopup('Це полігон.');
    </script>
</body>
</html>
```



(а)



(б)

Рис. 5. Створення карти OSM з кількома маркерами (а) та текстовим описовим атрибутом (б)

Використання даних в форматі GeoJSON (масив точкових даних) для візуалізації можна представити наступним чином.

```
<!DOCTYPE html>
<html>
<head>
   <title>GeoJSON Example</title>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
   <style>
      #map { height: 500px; }
   </style>
</head>
<body>
   <div id="map"></div>

   <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
   <script>
```

```
        // Ініціалізація карти
        var map = L.map('map').setView([50.4501, 30.5234], 13);

        // Додавання каргографічного зображення з OpenStreetMap
        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
            maxZoom: 19,
            attribution: 'Map data © <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
        }).addTo(map);

        // GeoJSON дані
        var geojsonData = {
            "type": "FeatureCollection",
            "features": [
                {
                    "type": "Feature",
                    "properties": {
                        "name": "Місце A",
                        "popupContent": "Це місце A!"
                    },
                    "geometry": {
                        "type": "Point",
                        "coordinates": [30.5234, 50.4501]
                    }
                },
                {
                    "type": "Feature",
                    "properties": {
                        "name": "Місце B",
                        "popupContent": "Це місце B!"
                    },
                    "geometry": {
                        "type": "Point",
                        "coordinates": [30.5250, 50.4520]
                    }
                }
            ]
        };

        // Додавання GeoJSON до карти
        L.geoJSON(geojsonData, {
```

```
        onEachFeature: function (feature, layer) {
            if (feature.properties && feature.properties.popupContent) {
                layer.bindPopup(feature.properties.popupContent);
            }
        }
    }).addTo(map);
  </script>
</body>
</html>
```

Цей HTML-документ є представленням Веб-сторінки з інтерактивною картою, яка відображає дані в форматі GeoJSON за допомогою бібліотеки Leaflet. У заголовній частині (head) частині аналогічно попередньому прикладу задається заголовок сторінки, встановлюється кодування символів (UTF-8) і налаштовується адаптивність сторінки для мобільних пристроїв, а також підключається CSS-файл бібліотеки Leaflet для стилізації карти, а також задається стиль для елемента з картою, встановлюючи висоту карти на 500 пікселів.

У body частині сторінки міститься контейнер з ідентифікатором «map», який використовується для відображення карти.

JavaScript код ініціалізує карту, встановлюючи її центр на координатах Києва з рівнем масштабування 13. Потім на карту додається шар картографічного зображення з OpenStreetMap, який використовується як базова карта. Далі в коді визначається об'єкт GeoJSON, який містить два об'єкти з типом «Feature», що мають тип геометрії «Point» з відповідними координатами. Кожен об'єкт також містить властивості, включаючи ім'я та текст для спливаючого вікна (popup). Останній блок коду додає GeoJSON дані на карту за допомогою функції L.geoJSON. При додаванні кожного об'єкта на карту, для кожного з них створюється спливаюче вікно, якщо об'єкт має властивість popupContent. Нарешті, всі об'єкти додаються на карту. Результат виконання коду представлений на рис. 6.

Leaflet дозволяє легко нарощувати функціонал шляхом підключення плагінів з додатковими функціями, наприклад для підтримки побудови маршрутів. Плагін Leaflet Routing Machine [17] дозволяє додавати можливості побудови маршруту на карту. Прокладемо маршрут між точками A та B.

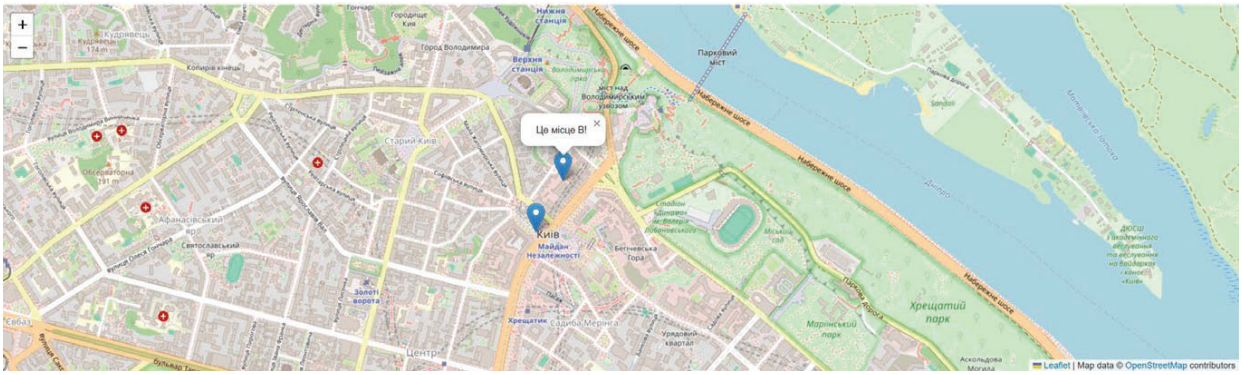


Рис. 6. Побудова карти OSM з кількома маркерами в JSON-форматі

```
<!DOCTYPE html>
<html>
<head>
    <title>Routing Example</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
    <link rel="stylesheet" href="https://unpkg.com/leaflet-routing-machine/dist/leaflet-routing-machine.css" />
    <style>
        #map { height: 500px; }
    </style>
</head>
<body>
    <div id="map"></div>

    <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
    <script src="https://unpkg.com/leaflet-routing-machine/dist/leaflet-routing-machine.js"></script>
    <script>
        // Ініціалізація карти
        var map = L.map('map').setView([50.4501, 30.5234], 13);

        // Додавання каргографічного зображення з OpenStreetMap
        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
            maxZoom: 19,
            attribution: 'Map data © <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
        }).addTo(map);
```

```
        // Маршрутизація
        L.Routing.control({
            waypoints: [
                L.latLng(50.4501, 30.5234),
                L.latLng(50.4547, 30.5238)
            ],
            routeWhileDragging: true
        }).addTo(map);
    </script>
</body>
</html>
```

HTML-документ починається з декларації DOCTYPE. У заголовковій частині (**head**) визначені метатеги для встановлення кодування сторінки (UTF-8) та забезпечення адаптивності дизайну для мобільних пристроїв. Підключено два CSS файли: один для базової стилізації карти через бібліотеку Leaflet, а інший для додавання стилів для маршрутизації з використанням бібліотеки Leaflet Routing Machine. Стиль задає висоту контейнера карти.

Аналогічно попередньому прикладу тіло сторінки (body) містить div елемент з ідентифікатором «map», який буде використовуватися як контейнер для карти. Підключаються JavaScript файли бібліотек Leaflet та Leaflet Routing Machine для роботи з картою та реалізації функціоналу маршрутизації.

Блок JavaScript коду ініціалізує карту з центром на Києві (координати) та заданим рівнем масштабування. Додається шар зображення з OpenStreetMap для відображення базової карти. Додається маршрутизатор (L.Routing.control) на карту з початковою і кінцевою точками маршруту, дозволяючи користувачам змінювати маршрут перетягуванням його на карті. Графічний вигляд створеного маршруту представлений на рис. 7.

Leaflet дозволяє досить легко налагоджувати використання шарів і контролерів.

```
<!DOCTYPE html>
<html>
<head>
    <title>Layers Example</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

```
    <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
    <style>
        #map { height: 500px; }
    </style>
</head>
<body>
    <div id="map"></div>

    <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
    <script>
        // Ініціалізація карти
        var map = L.map('map').setView([50.4501, 30.5234], 13);

        // Зображення OpenStreetMap
        var osmLayer = L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
            maxZoom: 19,
            attribution: 'Map data © <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
        });

        // Зображення Google Maps
        var googleLayer = L.tileLayer('http://{s}.google.com/vt/lyrs=m&x={x}&y={y}&z={z}', {
            maxZoom: 20,
            subdomains: ['mt0', 'mt1', 'mt2', 'mt3'],
            attribution: 'Map data © <a href="https://www.google.com/maps">Google Maps</a>'
        });

        // Додавання зображення на карту
        osmLayer.addTo(map);

        // Маркери
        var markerA = L.marker([50.4501, 30.5234]).bindPopup('Місце A');
        var markerB = L.marker([50.4547, 30.5238]).bindPopup('Місце B');

        // Група маркерів
        var markersGroup = L.layerGroup([markerA, markerB]);

        // Контролери шарів
```

```
        var baseLayers = {
            'OpenStreetMap': osmLayer,
            'Google Maps': googleLayer
        };

        var overlays = {
            'Маркери': markersGroup
        };

        // Додавання контролера шарів
        L.control.layers(baseLayers, overlays).addTo(map);
    </script>
</body>
</html>
```

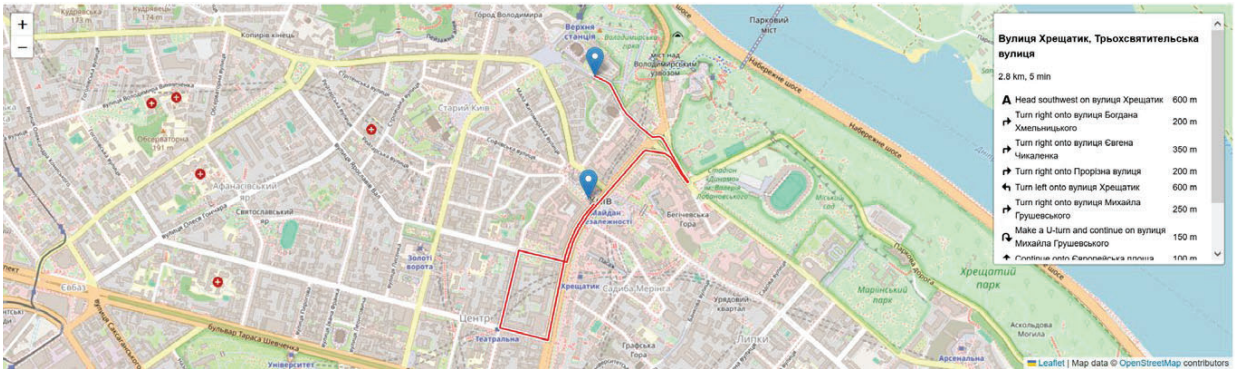


Рис. 7. Прокладання маршруту між двома точками із використанням плагіну

Більша частина коду (заголовкова та тіло сторінки) аналогічні попереднім прикладам, але є відмінність у частині JavaScript. Створюються два шари картографічних зображень: OpenStreetMap і Google Maps, кожен з яких має свої параметри конфігурації, такі як максимальний рівень масштабування і атрибуція. Далі створюються два маркери з координатами в Києві. Кожний маркер має прив'язане спливаюче вікно, яке відображає текст при натисканні на маркер. Маркери об'єднуються в групу, яку можна використовувати як один шар. І описується контролер шарів, в якому визначаються базові шари (OpenStreetMap і Google Maps) і оверлейний шар (група маркерів). Контролер шарів додається на карту, даючи користувачам можливість перемикатися між цими шарами (рис. 8).

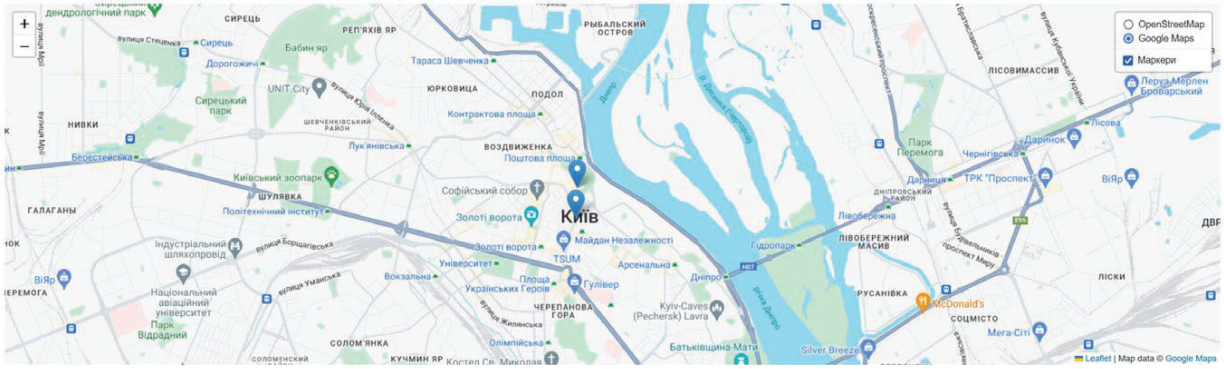


Рис. 8. Додавання шарів на карту

Виділимо основні **переваги та недоліки використання Leaflet.** Бібліотека Leflet є доволі цікавим та легким у використанні рішенням для розробки користувацького інтерфейсу геопорталу. Вона має свої переваги та недоліки, які наведені нижче.

Переваги. Leaflet легко інтегрується з іншими JavaScript-фреймворками, такими як React, Angular і Vue.js, що дозволяє створювати складні Веб-додатки з інтерактивними картами. Завдяки оптимізації та кешуванню, бібліотека забезпечує високу продуктивність навіть на мобільних пристроях і в умовах обмежених ресурсів. Leaflet пропонує великий вибір інструментів для створення кастомних рішень, які можуть задовольнити потреби різних проєктів. Крім того, активна спільнота розробників постійно оновлює бібліотеку та створює нові плагіни та інструменти.

Недоліки. Попри всі переваги, Leaflet може мати деякі обмеження у функціональності в порівнянні з іншими бібліотеками, наприклад OpenLayers, особливо коли йдеться про роботу з великими наборами даних. Також деякі складні функції, наприклад, обробка 3D-даних або підтримка специфічних форматів, можуть вимагати використання додаткових плагінів або розширень.

В цілому ж **Leaflet** — це потужна і гнучка бібліотека для створення інтерактивних карт, яка ідеально підходить для розробки геопорталів помірної складності. Завдяки легкості інтеграції, широкій підтримці стандартів і наявності великої кількості плагінів, Leaflet може бути ефективним рішенням для різних веб-проєктів, що потребують візуалізації геопросторових даних.

2.3.2. OPENLAYERS

OpenLayers [18] — це потужна бібліотека JavaScript з відкритим кодом, яка використовується для створення динамічних Веб-карт та інтерактивних геопорталів. Дистрибутив бібліотеки та документація доступні та сторінці Openlayers (рис. 9). OpenLayers забезпечує широкі можливості для роботи з різноманітними геопросторовими даними і підтримує великий набір функцій, включаючи візуалізацію векторних і растрових шарів, роботу з різними проєкціями, інтеграцію з геопросторовими сервісами та багато іншого. Завдяки своїй потужності і гнучкості, OpenLayers є відмінним вибором для розробників, які створюють геоінформаційні системи [19].

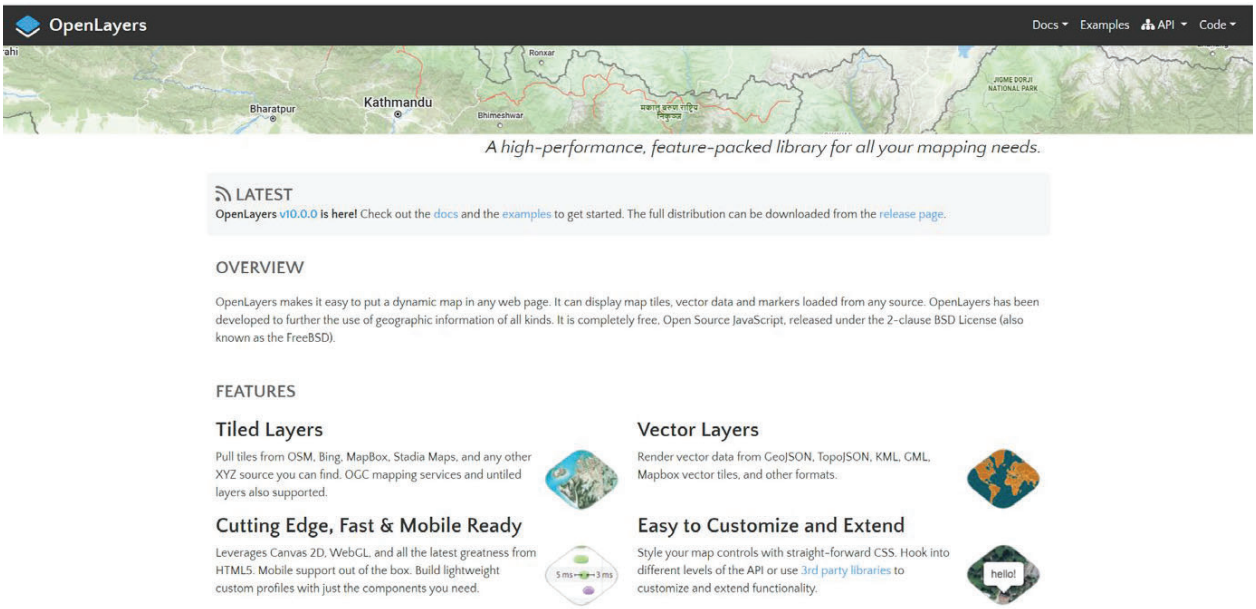


Рис. 9. Стартова сторінка проєкту OpenLayers

OpenLayers пропонує широкий спектр можливостей для роботи з геопросторовими даними.

Бібліотека **підтримує роботу з векторними форматами**, такими як GeoJSON, KML, GML, GPX, IGC, TopoJSON, а також із **растровими даними**, дозволяючи використовувати різноманітні картографічні сервіси, зокрема WMS, WMTS, TMS та інші.

OpenLayers має **вбудовану підтримку більшості проєкцій і систем координат**, а також забезпечує легке перетворення координат між різними системами координат, що спрощує роботу з даними з різних джерел.

Бібліотека дозволяє **відстежувати та обробляти різноманітні події на карті**, такі як клацання миші, панорамування,

масштабування та зміна вигляду карти. Крім того, є можливість додавання інтерактивних елементів та інформаційних вікон для відображення деталей про об’єкти на карті. OpenLayers надає доволі широкі **можливості налаштування стилів** для векторних шарів, включаючи маркери, лінії та полігони, а також дозволяє додавати анімації об'єктів на карті для створення динамічних візуалізацій.

Бібліотека легко **інтегрується з популярними геосервісами**, такими як Google Maps, Bing Maps, Mapbox, а також підтримує стандарти OGC, включаючи WMS, WFS, WCS, що розширює її можливості в контексті роботи з геоданими.

OpenLayers **оптимізована для роботи з великими наборами даних**, що забезпечує високу продуктивність і масштабованість. Підтримка кешування елементів картографічного зображення допомагає знизити навантаження на сервер і покращити швидкість завантаження карт.

Реалізуємо просте відображення карти за допомогою OpenLayers з OSM-зображенням.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>OpenLayers OpenStreetMap Example</title>
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/ol@v7.4.0/ol.css"
    />
    <style>
      .map {
        height: 500px;
        width: 100%;
      }
    </style>
  </head>
  <body>
    <h2>Карта OpenStreetMap за допомогою OpenLayers</h2>
    <div id="map" class="map"></div>
    <script type="module">
      import Map from "https://cdn.jsdelivr.net/npm/ol@v7.4.0/Map.js";
```

```
    import View from "https://cdn.jsdelivr.net/npm/ol@v7.4.0/View.js";
    import TileLayer from "https://cdn.jsdelivr.net/npm/ol@v7.4.0/layer/Tile.js";
    import OSM from "https://cdn.jsdelivr.net/npm/ol@v7.4.0/source/OSM.js";

    // Створення растрового шару з OpenStreetMap
    const osmLayer = new TileLayer({
      source: new OSM(),
    });

    // Створення карти
    const map = new Map({
      target: "map", // Ідентифікатор елемента HTML, в якому буде відображена карта
      layers: [osmLayer], // Додаємо шар з OSM
      view: new View({
        center: [3398015, 6524165], // Центр карти у форматі EPSG:3857 (Web Mercator)
        zoom: 12, // Початковий масштаб
      }),
    });
  </script>
 </body>
</html>
```

У заголовковій частині (**head)** встановлюється кодування символів (UTF-8) і налаштовується адаптивний дизайн для мобільних пристроїв. Також підключається CSS-файл бібліотеки OpenLayers для стилізації карти. У тілі сторінки (**body**) є заголовок і контейнер з ідентифікатором «map», який буде використовуватися для відображення карти. Карта має висоту 500 пікселів і ширину 100%, щоб заповнювати весь доступний простір на сторінці.

JavaScript код підключається як модуль і імпортує необхідні класи з бібліотеки OpenLayers: Map, View, TileLayer та OSM. Створюється шар картографічного зображення, який використовує OpenStreetMap як джерело даних. Потім створюється карта, яка відображається в контейнері з ідентифікатором «map». Карта містить шар з OpenStreetMap (рис. 10) і задається початковий вигляд з центром на визначених координатах у системі координат EPSG:3857 (Web Mercator) і з початковим масштабом 12.

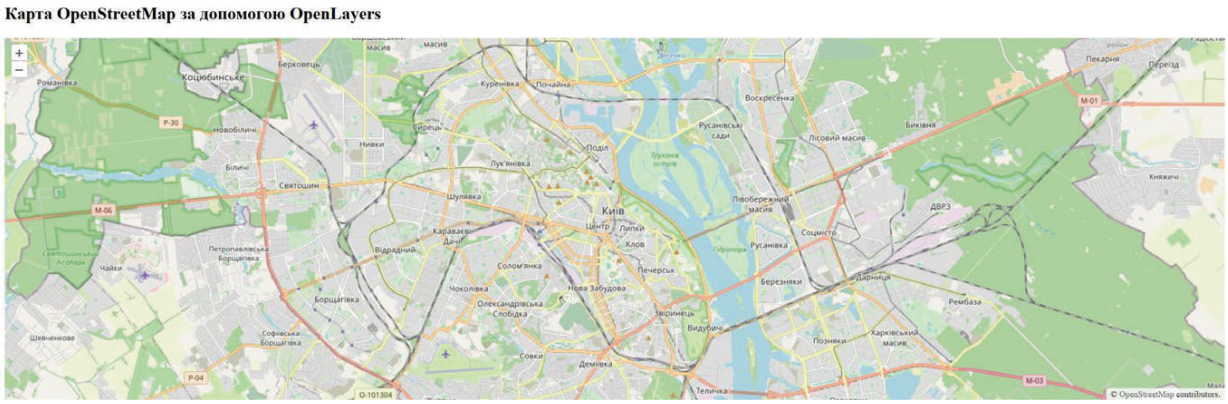


Рис. 10. Відображення базової карти засобами OpenLayers

Відобразимо дані в GeoJSON-форматі засобами OpenLayers:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>OpenLayers GeoJSON Example for Kyiv</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/ol3/4.6.5/ol.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/ol3/4.6.5/ol.js"></script>
    <style>
        .map {
            height: 100vh;
            width: 100%;
        }
    </style>
</head>
<body>
    <div id="map" class="map"></div>
    <script>
        // Додамо повідомлення для налагодження
        console.log('Initializing map...');

        // Створення карти
        var map = new ol.Map({
            target: 'map',
            layers: [
```

```
            new ol.layer.Tile({
                source: new ol.source.OSM()
            })
        ],
        view: new ol.View({
            center: ol.proj.fromLonLat([30.5234, 50.4501]), // Координати Києва
            zoom: 12
        })
    });

    console.log('Map initialized.');

    // GeoJSON об'єкти
    var geojsonObject = {
        'type': 'FeatureCollection',
        'features': [
            {
                'type': 'Feature',
                'geometry': {
                    'type': 'Point',
                    'coordinates': [30.5134, 50.4501] // Координати точки в Києві
                },
                'properties': {
                    'name': 'Kyiv Center'
                }
            },
            {
                'type': 'Feature',
                'geometry': {
                    'type': 'Point',
                    'coordinates': [30.5216, 50.4476] // Координати іншої точки в Києві
                },
                'properties': {
                    'name': 'Another Point'
                }
            }
        ]
    };
```

```
        console.log('GeoJSON object created.');

        // Створення вектора джерела даних
        var vectorSource = new ol.source.Vector({
            features: new ol.format.GeoJSON().readFeatures(geojsonObject,
{
                featureProjection: 'EPSG:3857'
            })
        });

        console.log('Vector source created.');

        // Створення векторного шару
        var vectorLayer = new ol.layer.Vector({
            source: vectorSource
        });

        console.log('Vector layer created.');

        // Додавання векторного шару до карти
        map.addLayer(vectorLayer);

        console.log('Vector layer added to map.');
    </script>
</body>
</html>
```

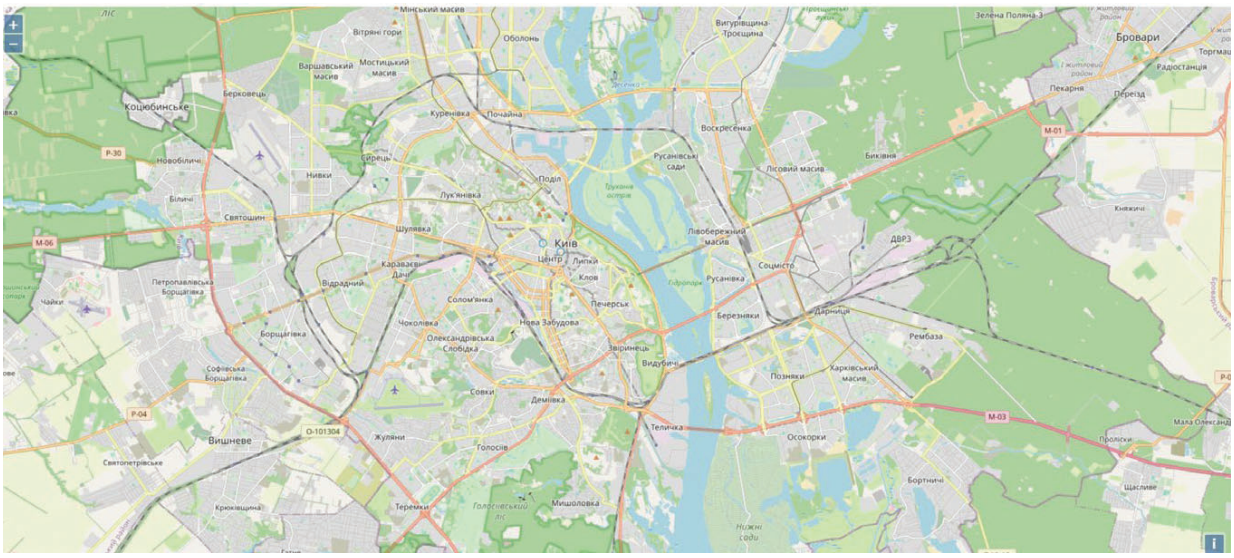


Рис. 11. Відображення векторних елементів в GeoJson форматі засобами OpenLayers

В даному прикладі створюється Веб-сторінка, яка відображає інтерактивну карту з GeoJSON даними за допомогою бібліотеки OpenLayers. У заголовковій частині **(head)** встановлюється кодування символів (UTF-8), налаштовується адаптивний дизайн для мобільних пристроїв, підключається CSS-файл для стилізації карти та JavaScript-файл бібліотеки OpenLayers.

У тілі сторінки **(body)** знаходиться контейнер з ідентифікатором map, в якому буде відображатися карта. CSS стиль задає висоту карти на весь екран (100vh) і ширину на 100%.

JavaScript код починається з виведення повідомлення в консоль для налагодження. Далі створюється карта з базовим шаром OpenStreetMap, яка центрована на координатах Києва та має початковий масштаб 12. Координати перетворюються у систему координат EPSG:3857 (Web Mercator). Створюється GeoJSON об'єкт, який містить два точкових об'єкти з їхніми координатами і властивостями. Потім цей GeoJSON об'єкт зчитується і перетворюється у векторні об'єкти OpenLayers з проєкцією EPSG:3857. Створюється векторний шар, який використовує зчитані векторні об'єкти як джерело даних. Цей шар додається на карту. Кожен крок супроводжується виведенням відповідного повідомлення в консоль для полегшення налагодження коду.

До **переваг** OpenLayers можна віднести **широкий функціонал**, який надає максимальний набір інструментів для роботи з геопросторовими даними, дозволяючи реалізовувати складні рішення в геоінформаційних системах. Завдяки підтримці стандартів OGC (Open Geospatial Consortium), OpenLayers **легко інтегрується з іншими геоінформаційними системами та сервісами**. Бібліотека **забезпечує велику гнучкість у налаштуванні та стилізації карт**, що дозволяє створювати кастомні рішення під конкретні потреби проєктів. Вона також **здатна ефективно обробляти великі набори даних**, забезпечуючи високу продуктивність. Активна спільнота розробників постійно вдосконалює бібліотеку, додаючи нові функції та виправляючи недоліки.

Проте OpenLayers має і **недоліки**. У порівнянні з іншими бібліотеками, такими як Leaflet, вона може бути **складнішою в налаштуванні та використанні**, що може вимагати більше часу на вивчення. Крім того, OpenLayers має більший розмір, ніж інші подібні бібліотеки, що може вплинути на швидкість завантаження карт у веб-додатках, особливо за умов обмеженої пропускної здатності мережевого з'єднання. Деякі функції можуть також не

підтримуватися старими версіями браузерів, що може створювати проблеми з сумісністю.

OpenLayers — це потужна бібліотека для створення геопорталів та інтерактивних карт, яка надає великий функціонал для роботи з різними типами геопросторових даних. Вона є відмінним вибором для проєктів, які вимагають високої продуктивності та гнучкості в налаштуванні, а також інтеграції з геосервісами і стандартами OGC. Однак, її використання може вимагати більше часу на освоєння та налаштування, ніж у випадку з іншими бібліотеками, такими як Leaflet.

2.3.3. MAPBOX GL JS

Mapbox [20] є досить зручною платформою для створення інтерактивних карт і геопорталів. Вона забезпечує розробників інструментами для інтеграції картографічних сервісів у Веб- та мобільні додатки. Відзначається досить повною та актуальною документацією, доступною на сайті проєкту (рис. 12а). Mapbox використовує бібліотеку **Mapbox GL JS** для рендерингу векторних карт з високою продуктивністю. Для використання необхідний токен, який можна створити у користувацькому профілі (рис. 12б).

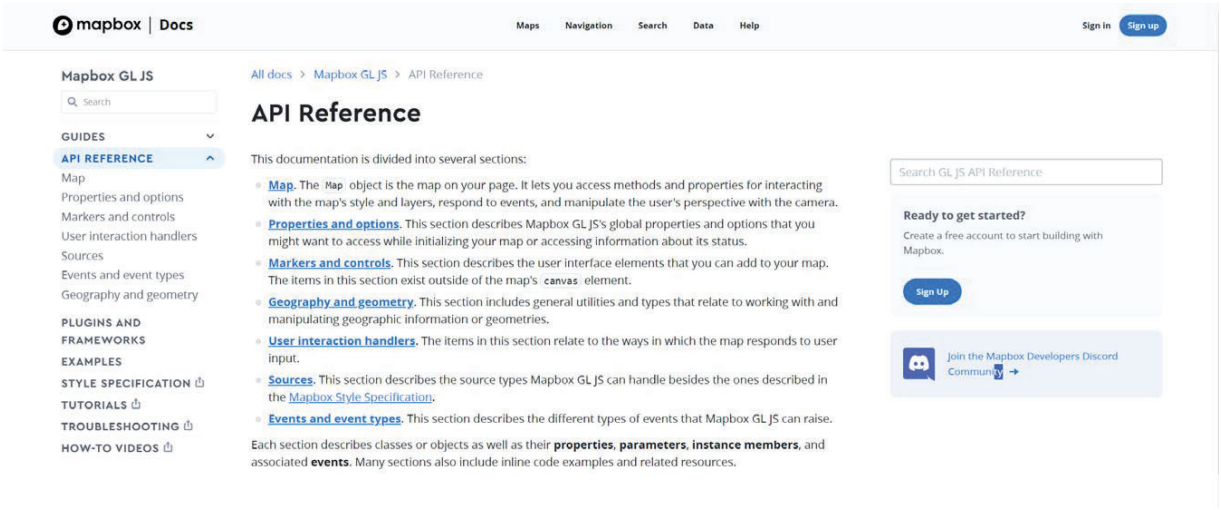
Основні можливості Mapbox: Mapbox пропонує кілька основних можливостей, серед яких підтримка векторних карт з забезпеченням високої продуктивності і деталізації карт. Користувачі можуть легко налаштовувати стилі карт для створення унікального вигляду. Доступні також функції геокодування для пошуку місць і адрес, а також інтеграція навігаційних маршрутів. Крім того, Mapbox дозволяє інтегрувати та візуалізувати власні дані у форматі GeoJSON. Даний продукт є комерційним і його використання в безкоштовному режимі має суттєві обмеження [19].

Розглянемо приклад використання Mapbox GL JS для відображення карти та відображення даних у форматі GeoJSON. Для роботи з Mapbox необхідний токен, який можна отримати, зареєструвавшись на офіційному сайті проєкту.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

```
    <title>Mapbox GL JS Example</title>
    <script src="https://api.mapbox.com/mapbox-gl-js/v2.3.1/mapbox-gl.js"></script>
    <link href="https://api.mapbox.com/mapbox-gl-js/v2.3.1/mapbox-gl.css" rel="stylesheet" />
    <style>
        body { margin: 0; padding: 0; }
        #map { position: absolute; top: 0; bottom: 0; width: 100%; }
        .map-overlay { position: absolute; background: white; padding: 10px; font-family: sans-serif; }
        .map-overlay .mapboxgl-ctrl { margin-bottom: 10px; }
        .map-overlay .mapboxgl-ctrl-group { display: inline-block; }
    </style>
</head>
<body>
    <div id="map"></div>
    <div class="map-overlay" id="menu">
        <input id="streets-v11" type="radio" name="rtoggle" value="streets-v11" checked="checked">
        <label for="streets-v11">Streets</label>
        <input id="outdoors-v11" type="radio" name="rtoggle" value="outdoors-v11">
        <label for="outdoors-v11">Outdoors</label>
        <input id="satellite-v9" type="radio" name="rtoggle" value="satellite-v9">
        <label for="satellite-v9">Satellite</label>
        <input id="light-v10" type="radio" name="rtoggle" value="light-v10">
        <label for="light-v10">Light</label>
        <input id="dark-v10" type="radio" name="rtoggle" value="dark-v10">
        <label for="dark-v10">Dark</label>
    </div>
    <script>
        // Вставте свій Mapbox API ключ сюди
        mapboxgl.accessToken = 'YOUR_MAPBOX_ACCESS_TOKEN';
        // Створення карти
        var map = new mapboxgl.Map({
            container: 'map',
            style: 'mapbox://styles/mapbox/streets-v11', // початковий стиль карти
            center: [30.5234, 50.4501], // Координати Києва
            zoom: 12
        });
```

```
        // Додавання контролу перемикання шарів
        var layerList = document.getElementById('menu');
        var inputs = layerList.getElementsByTagName('input');

        function switchLayer(layer) {
            var layerId = layer.target.value;
            map.setStyle('mapbox://styles/mapbox/' + layerId);
        }
        for (var i = 0; i < inputs.length; i++) {
            inputs[i].onclick = switchLayer;
        }
    </script>
</body>
</html>
```



(а)

(б)

Рис. 12. Стартова сторінка проєкту Mapbox (а) та інтерфейс для створення токенів доступу (б)

В наведеному прикладі створюється інтерактивна карта за допомогою Mapbox GL JS JS (рис. 13). У частині head визначені основні метатеги для налаштування кодування сторінки та забезпечення підтримки мобільних пристроїв. Підключаються JavaScript і CSS файли з бібліотеки Mapbox GL JS для роботи з картами.

У частині body сторінки знаходиться контейнер з ідентифікатором map, який використовується для відображення карти, а також блок елементів з класом map-overlay, що містить перемикачі для вибору різних стилів карт.

Код JavaScript починається з визначення API ключа Mapbox (mapboxgl.accessToken), який необхідний для доступу до сервісу Mapbox. Потім створюється карта з початковим стилем streets-v11, центром з координатами Києва та початковим рівнем масштабування 12.

Також додається функціонал для перемикання шарів карти. Блок перемикачів в елементі з ідентифікатором 'menu' дозволяє користувачам вибирати різні стилі карт. При виборі іншого шару викликається функція switchLayer, яка змінює стиль карти на вибраний.

Рис. 13. Вивід списку базових шарів для території м. Києва

Тепер додамо декілька векторних точок до попереднього прикладу.

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Mapbox GL JS Example</title>
  <script src="https://api.mapbox.com/mapbox-gl-js/v2.3.1/mapbox-gl.js"></script>
  <link href="https://api.mapbox.com/mapbox-gl-js/v2.3.1/mapbox-gl.css" rel="stylesheet" />
  <style>
    body { margin: 0; padding: 0; }
    #map { position: absolute; top: 0; bottom: 0; width: 100%; }
    .map-overlay { position: absolute; background: white; padding: 10px; font-family: sans-serif; }
    .map-overlay .mapboxgl-ctrl { margin-bottom: 10px; }
    .map-overlay .mapboxgl-ctrl-group { display: inline-block; }
  </style>
</head>
<body>
  <div id="map"></div>
  <div class="map-overlay" id="menu">
    <input id="streets-v11" type="radio" name="rtoggle" value="streets-v11" checked="checked">
    <label for="streets-v11">Streets</label>
    <input id="outdoors-v11" type="radio" name="rtoggle" value="outdoors-v11">
    <label for="outdoors-v11">Outdoors</label>
    <input id="satellite-v9" type="radio" name="rtoggle" value="satellite-v9">
    <label for="satellite-v9">Satellite</label>
    <input id="light-v10" type="radio" name="rtoggle" value="light-v10">
    <label for="light-v10">Light</label>
    <input id="dark-v10" type="radio" name="rtoggle" value="dark-v10">
    <label for="dark-v10">Dark</label>
  </div>
  <script>
    // Вставте свій Mapbox API ключ сюди
    mapboxgl.accessToken = 'YOUR_MAPBOX_ACCESS_TOKEN';
```

```
    // Створення карти
    var map = new mapboxgl.Map({
        container: 'map',
        style: 'mapbox://styles/mapbox/streets-v11', // початковий стиль карти
        center: [30.5234, 50.4501], // Координати Києва
        zoom: 12
    });

    // Додавання контролю перемикання шарів
    var layerList = document.getElementById('menu');
    var inputs = layerList.getElementsByTagName('input');
    function switchLayer(layer) {
        var layerId = layer.target.value;
        map.setStyle('mapbox://styles/mapbox/' + layerId);
    }

    for (var i = 0; i < inputs.length; i++) {
        inputs[i].onclick = switchLayer;
    }
    // Додавання векторних даних
    map.on('load', function () {
        // Додавання джерела даних
        map.addSource('places', {
            'type': 'geojson',
            'data': {
                'type': 'FeatureCollection',
                'features': [
                    {
                        'type': 'Feature',
                        'geometry': {
                            'type': 'Point',
                            'coordinates': [30.5234, 50.4501]
                        },
                        'properties': {
                            'title': 'Kyiv Center',
                            'description': 'Center of Kyiv'
                        }},
                    {
```

```
                'type': 'Feature',
                'geometry': {
                    'type': 'Point',
                    'coordinates': [30.5176, 50.4476]
                },
                'properties': {
                    'title': 'Another Point',
                    'description': 'Another point in Kyiv'
                }}]}
        });
        // Додавання шару даних
        map.addLayer({
            'id': 'places',
            'type': 'circle',
            'source': 'places',
            'paint': {
                'circle-radius': 10,
                'circle-color': '#007cbf'
            }});

        // Додавання pop-up при кліку на точку
        map.on('click', 'places', function (e) {
            var coordinates = e.features[0].geometry.coordinates.slice();
            var title = e.features[0].properties.title;
            var description = e.features[0].properties.description;
            new mapboxgl.Popup()
                .setLngLat(coordinates)
                .setHTML('<strong>' + title + '</strong><p>' + description +
'</p>')
                .addTo(map);
        });

        // Зміна курсору на "пойнтер" при наведенні на точку
        map.on('mouseenter', 'places', function () {
            map.getCanvas().style.cursor = 'pointer';
        });

        // Зміна курсору назад при виході з точки
        map.on('mouseleave', 'places', function () {
```

```
            map.getCanvas().style.cursor = '';
        }); });
    </script>
</body>
</html>
```

В наведеному прикладі аналогічно попередньому створюється інтерактивна карта за допомогою Mapbox GL JS, яка дозволяє користувачам перемикатися між різними стилями карт і взаємодіяти з векторними даними. У частині head визначені основні метатеги для налаштування кодування сторінки (UTF-8) та забезпечення адаптивності на мобільних пристроях. Далі підключаються файли CSS і JavaScript з бібліотеки Mapbox GL JS для роботи з картами, а також задаються стилі, щоб карта займала весь екран, а елементи інтерфейсу займали певні позиції відносно розміру екрану.

У частині body для відображення карти (#map) використано контейнер, а також блок перемикачів, які дозволяють користувачам змінювати стилі відображення карт (наприклад, Streets, Outdoors, Satellite).

Код JavaScript починається з визначення API ключа для Mapbox для доступу до сервісу. Далі створюється карта з початковим стилем streets-v11, центром з координатами Києва та масштабом 12. Код також додає функціонал для перемикання шарів карти через перемикачі. При виборі іншого стилю карти викликається функція switchLayer, яка змінює стиль карти на вибраний. При завантаженні карти додається джерело даних типу GeoJSON, яке містить дві точки з їх координатами та описом JS (рис. 14). Ці дані додаються як шар «circle» з визначеними параметрами (радіус кола та його колір). При натисканні на одну з точок з'являється спливаюче вікно з інформацією про цю точку. Також додається функціонал для зміни курсору миші на «pointer» при наведенні на точку, що покращує інтерфейс користувача.

Переваги та недоліки Mapbox. Переваги Mapbox GL JS включають високу продуктивність завдяки використанню WebGL, що забезпечує плавність роботи навіть з великими обсягами даних. Mapbox Studio дозволяє створювати власні стилі карт, які можна використовувати в додатках, а також налаштовувати карти за допомогою JSON стилів. Mapbox пропонує широкий набір інструментів для роботи з картами, таких як Mapbox Navigation SDK і Mapbox Geocoding API. Бібліотека підтримує різні формати

даних, включаючи GeoJSON, KML, GPX та Shapefiles, і дозволяє інтегруватися з іншими системами через REST API. Крім того, Mapbox дозволяє додавати інтерактивні елементи та анімації на карти, а також забезпечує високу доступність сервісів завдяки хмарній інфраструктурі, що дозволяє обробляти великі обсяги даних і запити. Якісна документація та активна спільнота користувачів надають численні приклади та підтримку.

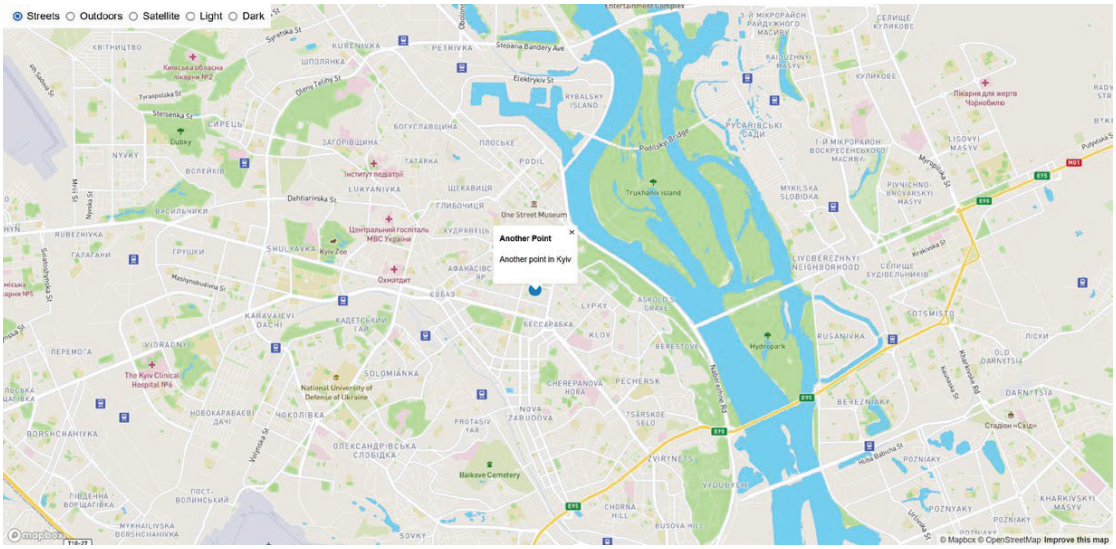


Рис. 14. Виведення векторних даних засобами Mapbox

Недоліки Mapbox включають вартість використання, яка може бути значною для великих проєктів через модель тарифікації на основі кількості запитів та обсягу даних. Безкоштовний тарифний план має обмеження, що може стати проблемою для масштабних проєктів. Використання Mapbox залежить від стабільного інтернет-з'єднання, що може викликати затримки або проблеми з доступністю в регіонах з поганим інтернет-з'єднанням. Налаштування може бути складним для новачків через великий набір функцій та можливостей, що потребує часу для навчання. Mapbox має обмежену підтримку офлайн-карт у веб-версії, що потребує додаткових налаштувань для створення офлайн-карт для мобільних додатків. Крім того, використання Mapbox залежить від стабільності та політики стороннього провайдера, що може змінювати умови використання або тарифи.

Mapbox є цікавим комерційним інструментом для створення інтерактивних карт та геопорталів, що надає широкий спектр можливостей для налаштування та інтеграції. Однак, вартість та складність можуть бути недоліками для деяких проєктів. При

виборі Mapbox важливо враховувати вимоги проєкту, можливі альтернативи та бюджетні можливості.

2.4. ГОТОВІ ДО ВИКОРИСТАННЯ РІШЕННЯ (MAPBENDER)

Mapbox [20] - **Mapbender** [21] — це Веб-GIS (геоінформаційна система), розроблена для управління геопросторовими даними та сервісами, яка дозволяє створювати, редагувати та публікувати геопросторові продукти у вигляді наборів тематичних шарів. Mapbender є частиною OSGeo (Open Source Geospatial Foundation) проєктів і забезпечує користувачів інструментами для роботи з даними через веб-інтерфейс [22]. Архітектурна схема MapBender представлена на рис. 15.

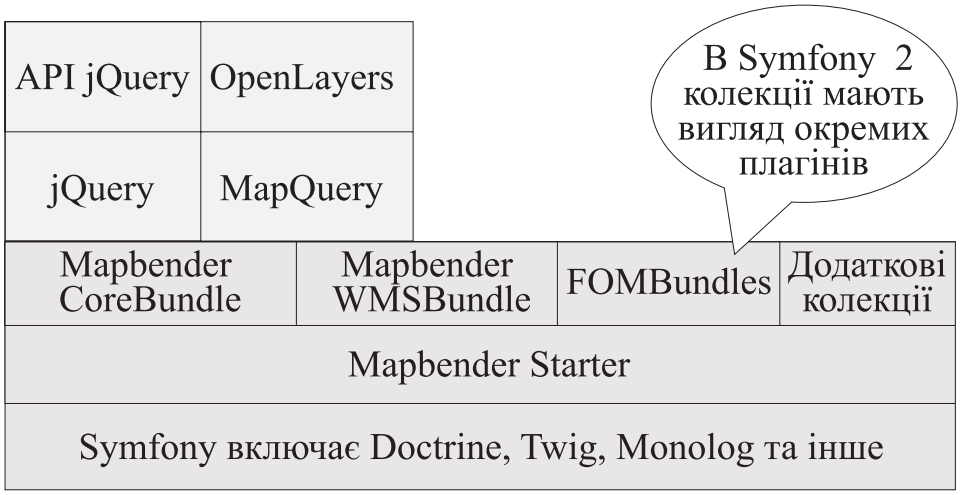


Рис. 15. Внутрішня архітектура Mapbender

Архітектура Mapbender складається з наступних блоків.

1. Клієнтська частина (frontend) забезпечує інтерфейс для кінцевих користувачів, дозволяючи їм взаємодіяти з Веб-картами. Вона побудована на основі стеку наступних технологій і складається з кількох ключових компонентів.

HTML задає структуру Веб-сторінок, створюючи основний каркас інтерфейсу користувача, тоді як **CSS** відповідає за стилізацію, забезпечуючи привабливий та зручний дизайн. **JavaScript** додає інтерактивність і динамічність, використовуючи такі бібліотеки, як **jQuery**, що полегшує роботу з DOM (Document Object Model) та обробку подій, і **OpenLayers**, яка дозволяє відображати, панорамувати, масштабувати та взаємодіяти з картографічними даними.

Клієнтська частина також включає **віджети** — інтерактивні компоненти, що надають різноманітні функції для роботи з картами. Серед них навігаційні віджети для панорамування, масштабування і повертання карти, інструменти для малювання та вимірювання точок, ліній і полігонів на карті, віджети для вибору та налаштування видимості шарів, інструменти для відображення інформації про об'єкти на карті, легенди та масштабні лінійки, а також інструменти для пошуку та геокодування.

Інтерактивні карти в Mapbender створюються за допомогою бібліотеки OpenLayers, що дозволяє користувачам переглядати картографічні дані в різних проєкціях та масштабах, накладати кілька шарів даних і налаштовувати їх прозорість, а також використовувати різні базові карти, такі як OpenStreetMap, Google Maps та Bing Maps.

Клієнтська частина також дозволяє взаємодіяти з різними картографічними сервісами для отримання та відображення геопросторових даних. Перш за все до них слід віднести WMS для обміну картографічними зображеннями, WFS для обміну геопросторовими об'єктами та WMTS для обміну картографічними фрагментами через Інтернет.

Користувачі можуть налаштовувати інтерфейс клієнтської частини Mapbender відповідно до своїх потреб. Зокрема, можна додавати, видаляти або налаштовувати віджети, вибирати шари даних, які відображатимуться за замовчуванням, та їхнього порядку відображення. Також забезпечується персоналізація інтерфейсу, дозволяючи зберігати налаштування для кожного користувача для створення персоналізованого досвіду.

2. Серверна частина (backend). Mapbender використовує популярні веб-сервери для обробки HTTP-запитів. **Apache** є широко використовуваним веб-сервером, відомим своєю надійністю та гнучкістю, тоді як **Nginx** забезпечує високу продуктивність і ефективне використання ресурсів.

PHP є основною мовою програмування для серверної частини Mapbender. Вона обробляє запити від клієнтської частини, взаємодіє з базою даних і формує відповіді. Mapbender побудований на Symfony, популярному PHP фреймворку, який забезпечує структуровану архітектуру та набір інструментів для розробки веб-додатків.

Symfony-контролери відповідають за обробку HTTP-запитів, виконання бізнес-логіки та повернення відповідей клієнтській частині. Вони обробляють конкретні запити, виконують необхідні дії, такі як отримання даних з бази даних або виклик геосервісів,

і формують відповіді. Маршрутизація визначає, які контролери оброблятимуть певні URL-запити, забезпечуючи правильний розподіл запитів та логіку роботи додатку.

Mapbender використовує **реляційні бази даних** для зберігання конфігурацій та геопросторових даних. Основною системою управління базами даних є PostgreSQL, яка забезпечує надійне зберігання та доступ до даних, а розширення PostGIS додає підтримку геопросторових даних та запитів.

Серверна частина Mapbender **взаємодіє з різними геосервісами** для отримання та надання геопросторових даних. Це включає стандарти WMS для обміну картографічними зображеннями, WFS для обміну геопросторовими об'єктами та WMTS для обміну фрагментами картографічних зображень через Інтернет.

Mapbender надає **API для інтеграції** з іншими системами та розширення функціональних можливостей. **REST API** забезпечує програмний доступ до функцій Mapbender, що дозволяє автоматизувати завдання та інтегрувати Mapbender з іншими додатками. Система підтримує розширення через плагіни та модулі, що дозволяє додавати нові функції або інтегрувати сторонні сервіси.

Mapbender забезпечує надійні механізми для аутентифікації та авторизації користувачів. Система дозволяє створювати користувачів та призначати їм ролі з відповідними правами доступу, а також забезпечує детальні налаштування прав доступу для різних функцій та даних, що гарантує безпеку та конфіденційність.

Mapbender також включає інструменти збору службової інформації та моніторингу. Журнали подій містять записи про дії користувачів та системи, що допомагає у відстеженні та розв'язанні проблем. Інструменти для моніторингу продуктивності серверної частини дозволяють своєчасно виявляти та вирішувати проблеми з продуктивністю.

До переваг клієнтської частини Mapbender можна віднести інтерактивність та вдалий користувацький досвід.

Mapbender пропонує інтуїтивний інтерфейс, який розроблений для зручності використання кінцевими користувачами. Він забезпечує простоту навігації та легкий доступ до основних функцій. Адаптивний дизайн дозволяє користуватися Mapbender на різних пристроях, включаючи комп'ютери, планшети та смартфони, забезпечуючи однаковий досвід використання незалежно від розміру екрана.

Інтерактивні функції включають динамічне завантаження даних, що відображаються в реальному часі, а також можливість взаємодії з картами через кліки, перетягування, масштабування та інші дії.

Клієнтська частина Mapbender спроєктована для легкої інтеграції з іншими системами та сервісами. Вона підтримує загальноприйняті стандарти, такі як WMS, WFS, WMTS, що забезпечує сумісність з іншими геопросторовими системами та інструментами. Крім того, API надає програмний доступ до функцій Mapbender, дозволяючи розробникам створювати власні рішення та інтеграції.

До переваг серверної частини Mapbender можна віднести наступні особливості. Дане ПЗ є проєктом з відкритим кодом, що означає, що користувачі мають доступ до вихідного коду і можуть його модифікувати, налаштовувати під свої потреби та розповсюджувати. Безкоштовність використання робить Mapbender доступним для різних організацій та проєктів незалежно від їх бюджету.

Mapbender також відзначається гнучкістю та масштабованістю. Його модульна архітектура дозволяє додавати нові функції та розширення без значних змін у базовій структурі. Mapbender може бути налаштований для роботи з великими обсягами даних та високими навантаженнями, що робить його придатним для проєктів різного масштабу.

Він **підтримує основні стандарти для роботи з геопросторовими даними**, зокрема стандарти OGC, такі як WMS, WFS, та WMTS. Це забезпечує сумісність з іншими ГІС-системами та сервісами, а також полегшує інтеграцію з різними базами даних та геосервісами.

Mapbender використовує Symfony як основний фреймворк для серверної частини, що забезпечує стабільність, безпеку та багатий набір інструментів для розробки веб-додатків. Він також має вбудовані механізми для управління доступом, що дозволяють налаштовувати права доступу для різних користувачів та ролей.

Керувати налаштуваннями та адмініструвати Mapbender можна через зручний Веб-інтерфейс, який дозволяє легко управляти користувачами, правами доступу, шарами даних та іншими параметрами.

Mapbender забезпечує інтеграцію та розширюваність завдяки наявності REST API для програмного доступу до своїх функцій, що дозволяє інтегрувати його з іншими системами та автоматизувати процеси. Система підтримує створення та

використання плагінів і модулів, що дозволяє додавати нові функції та інтеграції без зміни основного коду.

Висока продуктивність Mapbender досягається завдяки використанню сучасних технологій та оптимізованих алгоритмів, які забезпечують ефективну роботу навіть з великими обсягами даних. Система також підтримує кешування, що зменшує навантаження на сервер та прискорює відображення даних.

Більшу частину операцій по керування таким геопорталом можна зробити у зручному Веб-інтерфейсі. На рисунках (рис. 16-18) нижче представлені окремі елементи керування, доступні в Mapbender.

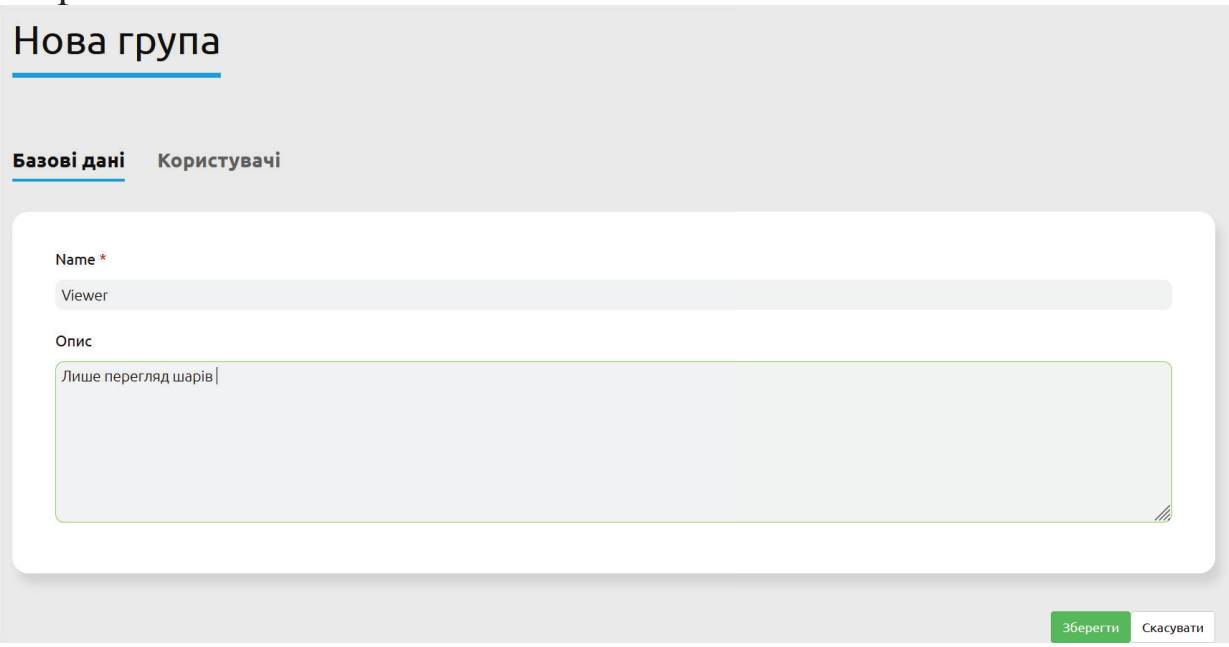


Рис. 16. Створення групи Viewer

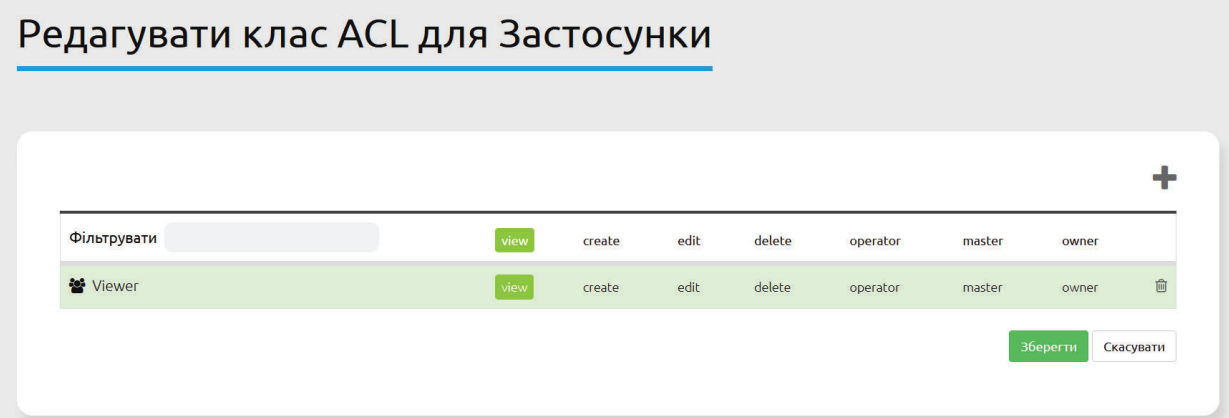


Рис. 17. Робота зі списками контролю доступу для групи Viewer

Рис. 18. Додавання нового джерела даних з доступом по протоколу WMS

В межах проєкту e-shape “EO in support of agricultural activities: new horizons” [23] на базі Mapbender було розгорнуто портал, який є практичною демонстрацією можливостей цього додатку. Для дашборду визначено набір користувацьких інструментів (рис. 19) та продемонстровано можливості зміни стилів для зовнішнього вигляду дашборду (рис. 20). Створений дашборд отримує дані по WMS протоколу з низки налаштованих джерел (рис. 21).

Портал містить деякі геопросторові продукти, які оцінюють індикатор цілі сталого розвитку 2.4.1 «Частка сільськогосподарських площ під продуктивним та сталим сільським господарством» та 15.3.1 «Частка деградованої землі від загальної площі землі» (рис. 22). Також створений портал містить низку растрових геопросторових продуктів, зокрема LAI (Leaf Area Index) для території України (рис. 23).

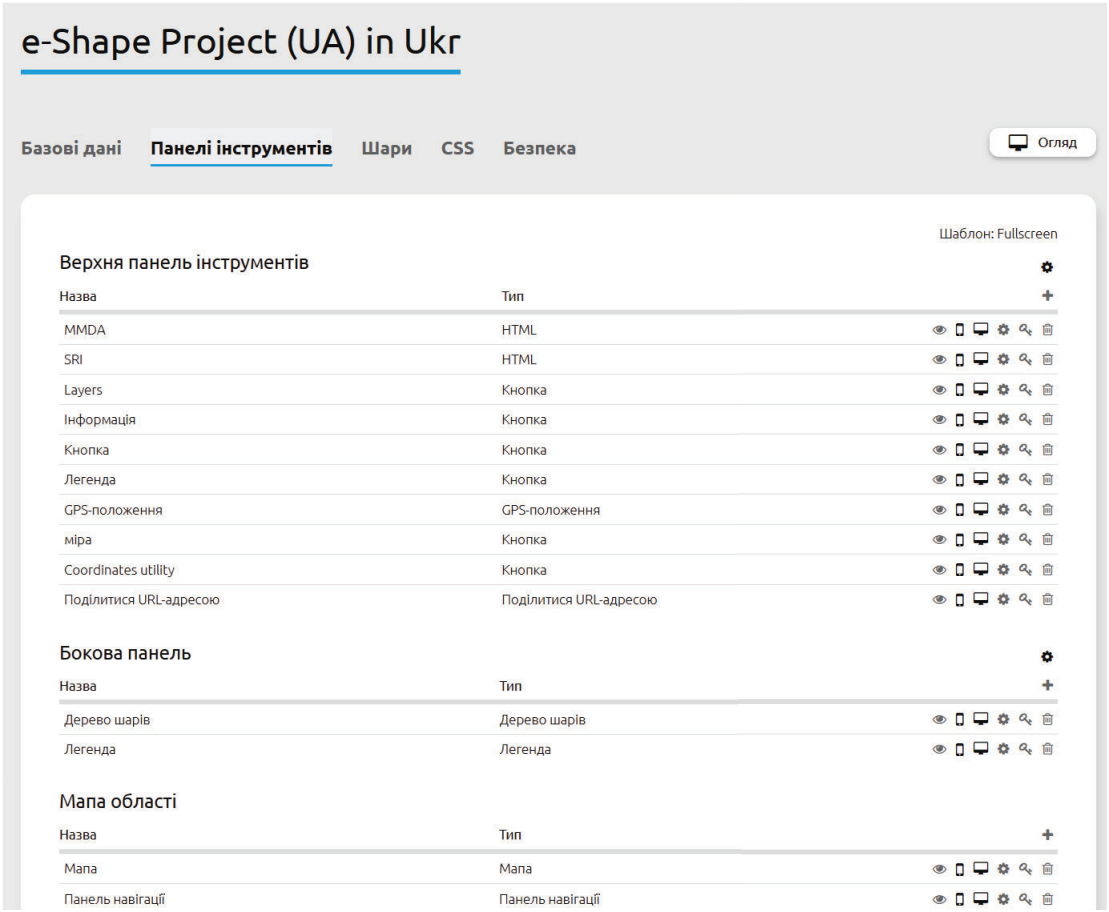


Рис. 19. Налаштування віджетів для дашборду

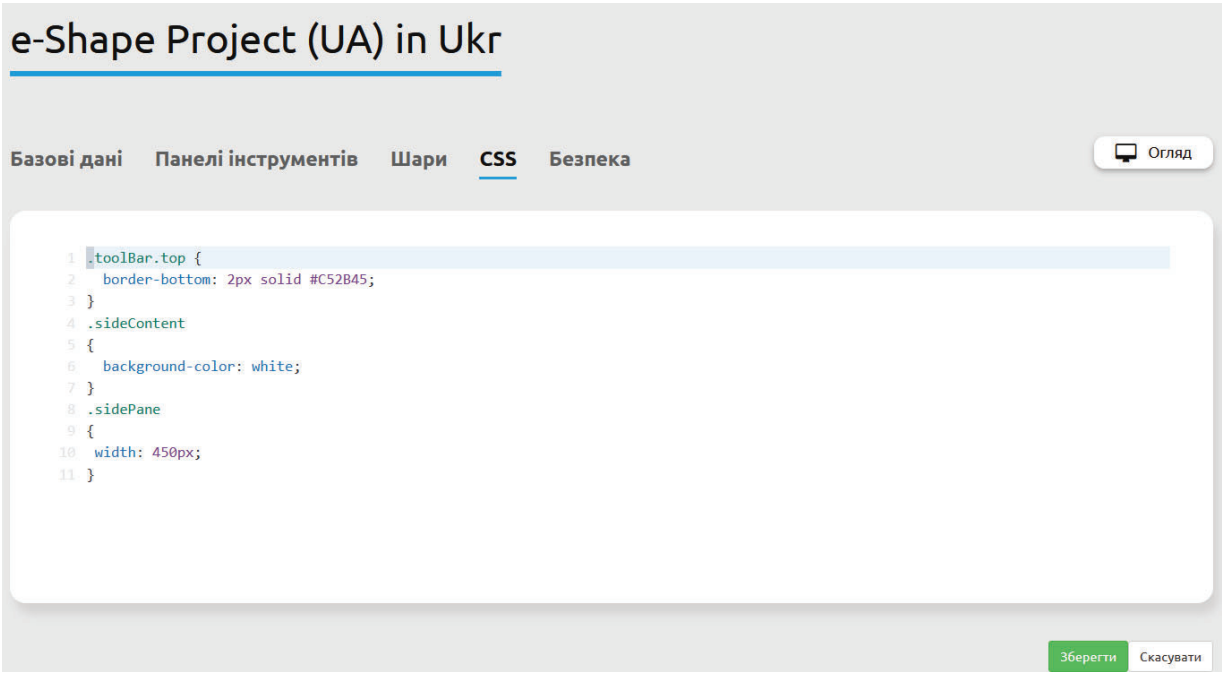


Рис. 20. Використання CSS для зміни зовнішнього вигляду дашборду

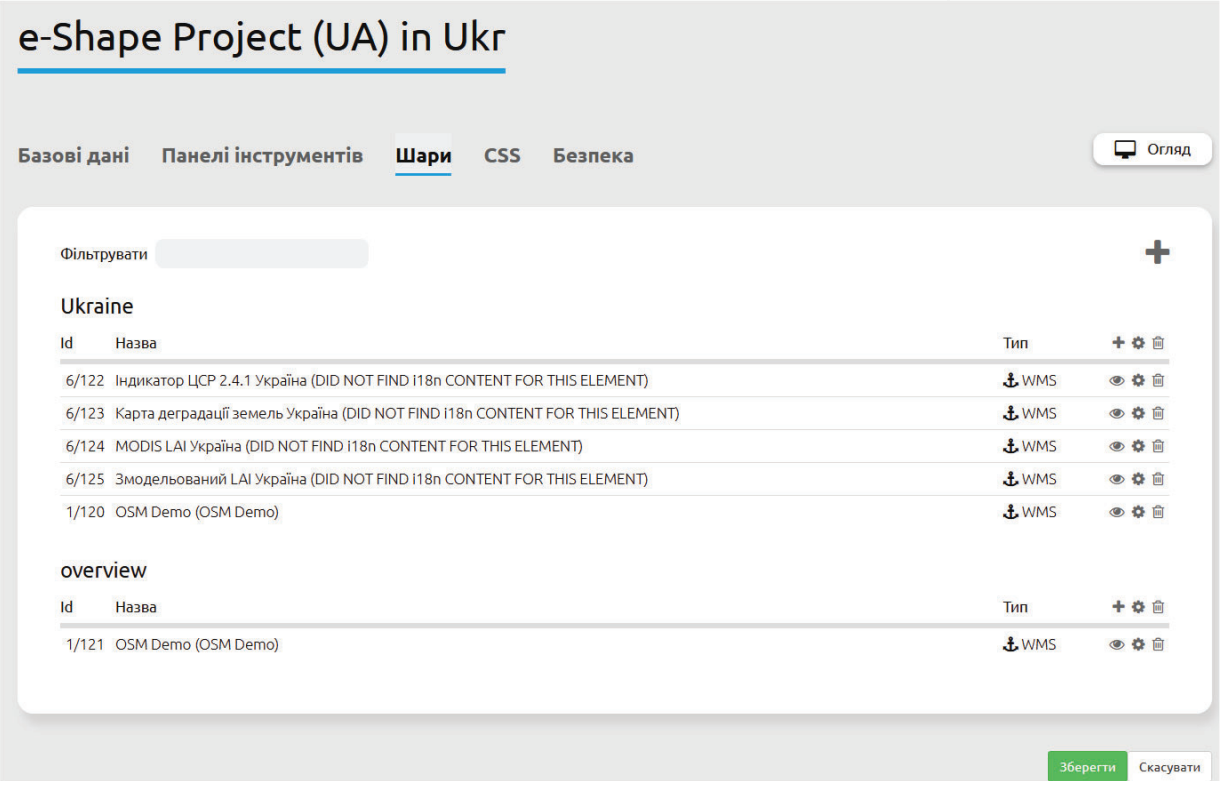


Рис. 21. Налаштування шарів даних дошборду

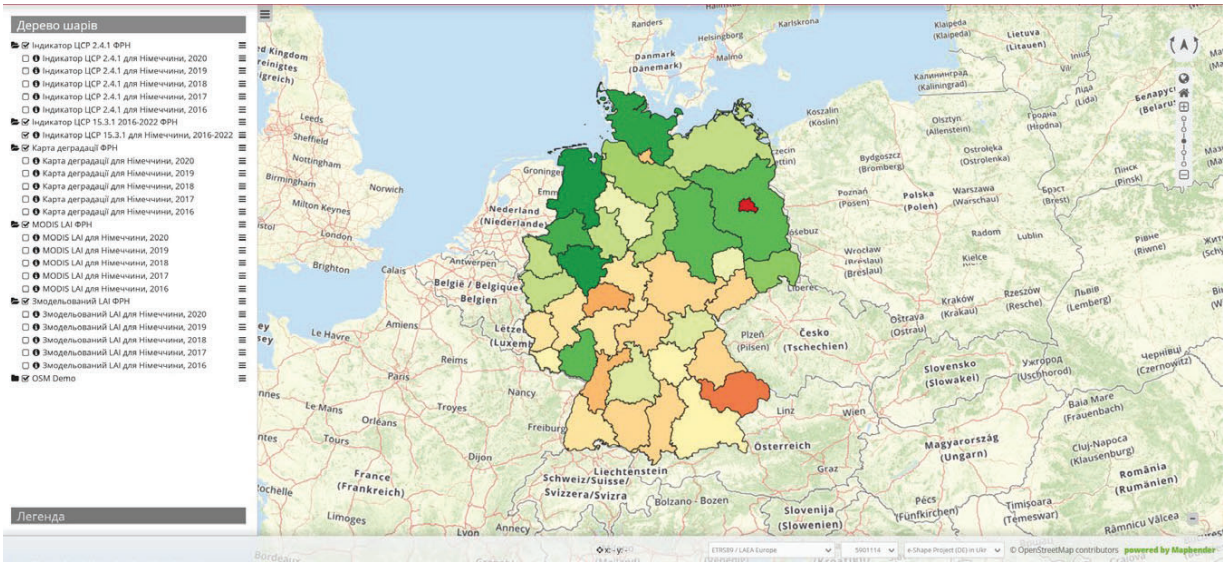


Рис. 22. Відображення векторних даних на дашборді

Mapbender є доволі зручним і готовим для використання клієнтським рішенням для створення геопорталів, але при виборі його як основи накладаються певні архітектурні та функціональні обмеження, які можуть ускладнювати значну кастомізацію проєкту під власні потреби або вимагати для цього значно більше зусиль, ніж вже розглянуті аналоги (Leaflet, OpenLayers тощо).

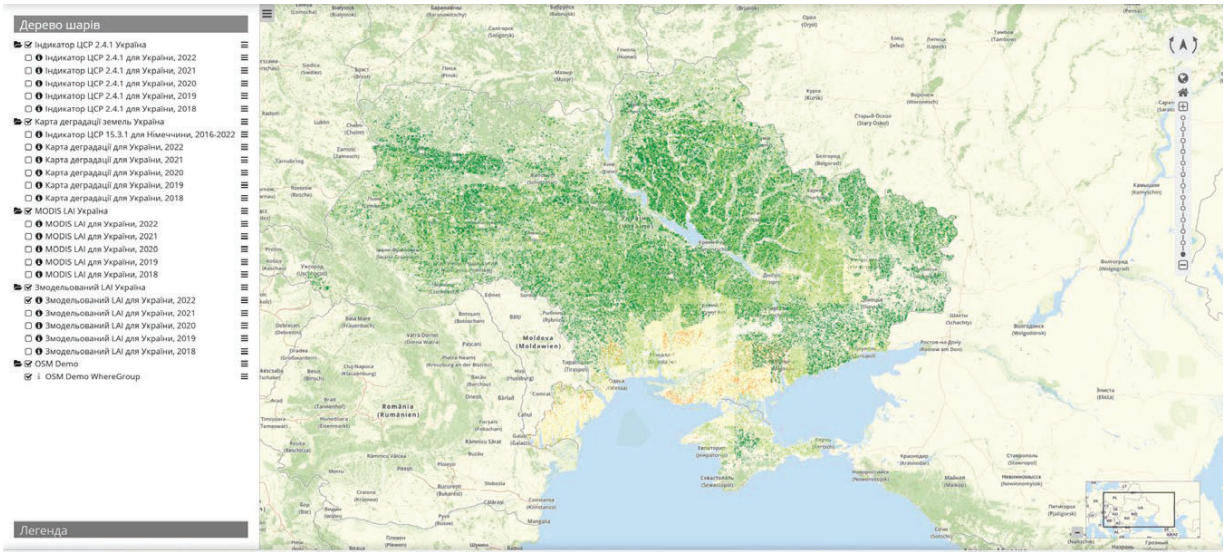


Рис. 23. Відображення растрових даних на дашборді

ВИСНОВКИ

Технології створення геопорталів грають ключову роль у розвитку сучасних систем для роботи з геопросторовими даними. Вони дозволяють ефективно візуалізувати, аналізувати та управляти геоданими, надаючи користувачам зручні інструменти для доступу до картографічної інформації. У цьому розділі розглянуто такі рішення, як Geoserver, MapServer, QGIS Server, Leaflet, OpenLayers, Mapbox та Mapbender, кожне з яких має свої переваги і підходить для певних типів проєктів.

Geoserver є одним із найбільш популярних серверних рішень для роботи з геопросторовими даними. Він підтримує численні формати даних і стандарти OGC, що робить його ідеальним рішенням для інтеграції з іншими системами. Geoserver дозволяє легко публікувати та обробляти великі обсяги геоданих, забезпечуючи високий рівень гнучкості та масштабованості. Його функції, включаючи можливість створення складних стилів для карт і підтримку великих наборів даних, роблять його популярним вибором для організацій, які працюють з великими і складними геопросторовими проєктами.

MapServer є ще одним серверним рішенням, яке відзначається високою продуктивністю і можливістю роботи з широким спектром форматів даних. Він підходить для проєктів, де потрібна ефективна обробка та відображення великих обсягів геоданих. MapServer підтримує стандарти OGC, що дозволяє інтегрувати його з іншими ГІС-системами та інструментами. Його здатність працювати з різноманітними джерелами даних і високий

рівень продуктивності роблять його відмінним вибором для проєктів з високими вимогами до швидкості обробки та відображення даних.

QGIS Server є серверною частиною популярного настільного ГІС-рішення QGIS. Він дозволяє легко публікувати картографічні дані, створені в QGIS Desktop, у Веб-додатках. QGIS Server інтегрується з QGIS Desktop, що робить його зручним інструментом для тих, хто вже використовує QGIS для створення карт. Ця технологія забезпечує зручний та інтуїтивно зрозумілий інтерфейс для публікації карт, а також підтримує основні геопросторові стандарти, що робить його відповідним вибором для проєктів, де потрібно швидко створити веб-карти з мінімальними витратами на налаштування.

На стороні клієнта існує кілька бібліотек для роботи з картами у Веб-додатках, які мають різні можливості та різну складність використання.

Leaflet є легкою та простою у використанні бібліотекою, яка ідеально підходить для створення базових картографічних додатків. Вона має невеликий розмір, що забезпечує швидке завантаження і високу продуктивність навіть на мобільних пристроях. Leaflet підтримує широкий спектр плагінів, які розширюють її функціональність, роблячи її популярним вибором для невеликих та середніх проєктів, де потрібна інтерактивність і легкість у використанні.

OpenLayers є значно складнішою бібліотекою, яка забезпечує ширшу функціональність порівняно з Leaflet. Вона підтримує роботу з різними форматами даних і проєкціями, що зумовлює її відповідність для складних картографічних проєктів. OpenLayers надає гнучкість у налаштуванні та дозволяє створювати кастомні рішення для різних типів проєктів. Завдяки підтримці основних стандартів OGC, вона легко інтегрується з серверними рішеннями, такими як Geoserver і MapServer.

Mapbox є потужною комерційною платформою для створення інтерактивних карт, яка використовує векторні карти для забезпечення високої продуктивності і деталізації. Вона надає інструменти для кастомізації карт, що дозволяє створювати унікальні картографічні рішення. Mapbox також пропонує різноманітні API для геокодування, навігації та інших гео-функцій, що робить її популярним вибором для проєктів, де потрібна висока інтерактивність і деталізація карт. Завдяки підтримці хмарної інфраструктури, Mapbox забезпечує високу

доступність і масштабованість, що робить її придатною для великих проєктів.

Mapbender є готовим рішенням для створення геопорталів, яке надає користувачам зручний веб-інтерфейс для роботи з картами. Вона інтегрується з різними геосервісами, такими як WMS, WFS та WMTS, і забезпечує легкість у налаштуванні та використанні. Mapbender підтримує розширення через плагіни та модулі, що дозволяє додавати нові функції та інтеграції без зміни основного коду. Це робить Mapbender відмінним вибором для організацій, які шукають простий та ефективний інструмент для створення та адміністрування геопорталів.

Все це представляє собою різноманітний набір інструментів для створення геопорталів. Вибір відповідної технології залежить від специфічних вимог проєкту, таких як обсяг даних, необхідність у кастомізації, інтерактивність і бюджет. Використання цих інструментів дозволяє створювати ефективні, функціональні та масштабовані рішення для роботи з геопросторовими даними.

ПЕРЕЛІК ПОСИЛАНЬ

1. Шелестов, А. Ю., Куссуль, Н. М., Скакун, С. В., Кравченко, О. М., Волошин, С. В., & Загородній, Є. В. (2011). Геоінформаційна система моніторингу для сільськогосподарського підприємства. *Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка*, (13), 121-125.

2. Шелестов, А. Ю., Кравченко, А. Н., Скакун, С. В., Волошин, С. В., & Куссуль, Н. Н. (2013). Информационная система агромониторинга на основе геопространственных данных. *Кибернетика и системный анализ.*

3. Шелестов, А. Ю., Куссуль, Н. М., Волошин, С. В., Скакун, С. В., Кравченко, О. М., & Колотій, А. В. (2011). Геоінформаційна система фермера. *Наука та інновації*, *7*(3), 25-29.

4. Corti, P., Bartoli, F., Fabiani, A., Giovando, C., Kralidis, A. T., & Tzotsos, A. (2019). *GeoNode: an open source framework to build spatial data infrastructures* (No. e27534v1). PeerJ Preprints.

5. Kulawiak, M., Dawidowicz, A., & Pacholczyk, M. E. (2019). Analysis of server-side and client-side Web-GIS data processing methods on the example of JTS and JSTS using open data from OSM and geoportal. *Computers & Geosciences*, *129*, 26-37.

6. Jiang, H., van Genderen, J., Mazzetti, P., Koo, H., & Chen, M. (2020). Current status and future directions of geoportals. *International Journal of Digital Earth*.

7. Сайт GeoServer. https://geoserver.org/

8. Сайт Spring Framework. https://spring.io/projects/spring-framework

9. Сайт GeoTools. https://www.geotools.org/

10. Gui, Z., Cao, J., Liu, X., Cheng, X., & Wu, H. (2016). Global-scale resource survey and performance monitoring of public OGC web map services. *ISPRS International Journal of Geo-Information*, *5*(6), 88.

11. Сайт MapServer. https://mapserver.org/

12. Сайт GDAL/OGR. https://gdal.org/en/latest/faq.html#what-is-this-ogr-stuff

13. Сайт QGIS Server. https://docs.qgis.org/3.34/en/docs/server_manual/index.html

14. Khan, S., & Mohiuddin, K. (2018). Evaluating the parameters of ArcGIS and QGIS for GIS Applications. *Int. J. Adv. Res. Sci. Eng*, *7*, 582-594.

15. Сайт Leaflet. https://leafletjs.com/

16. Horbiński, T., & Lorek, D. (2022). The use of Leaflet and GeoJSON files for creating the interactive web map of the preindustrial state of the natural environment. *Journal of Spatial Science*, *67*(1), 61-77.

17. Сайт плагіну для Leaflet. https://www.liedman.net/leaflet-routing-machine/

18. Сайт OpenLayers. https://openlayers.org/

19. Zunino, A., Velбzquez, G., Celemнn, J. P., Mateos, C., Hirsch, M., & Rodriguez, J. M. (2020). Evaluating the performance of three popular Web mapping libraries: A case study using Argentina's life quality index. *ISPRS International Journal of Geo-Information*, *9*(10), 563.

20. Сайт Mapbox. https://docs.mapbox.com/mapbox-gl-js/api/

21. Сайт MapBender. https://mapbender.org/en/

22. Kuzin, V., Musial, J., & Shelestov, A. (2022, December). EO4ua initiative: Scientific European support of Ukrainian scientific community. In *2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT)* (pp. 1-5). IEEE

23. Сайт проєкту e-shape. https://e-shape.eu/index.php/showcases/pilot-1-6-service-for-sdg-2-4-1-and-15-3-1-indicators-assessment