НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри

_____ Сергій СТІРЕНКО
(підпис)

“__”_____2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою “Комп’ютерні системи та мережі”
спеціальності 123 “Комп’ютерна інженерія”

на тему: «Картографічний сервіс на базі PostGIS»

Виконав (-ла): студент (-ка) 4 курсу, групи ІО-71
(шифр групи)

Корсун Роман Віталійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник асист. Іваніщев Б.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) д.т.н., проф. Сімоненко В.П. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєктінемає запозичень з праць інших авторів безвідповідних посилань.

Студент _____
(підпис)

Київ – 2021 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“___”_____2021 р.

ЗАВДАННЯ

на бакалаврський дипломний проєкт студента

Корсуна Романа Віталійовича

1. Тема проєкту *Картографічний сервіс на базі PostGIS*
керівник проєкту *асист. Іваніщев Б.В.*,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 11 травня 2021 року № 1139-с
2. Термін здачі студентом закінченого проєкту *1 червня 2021 р.*
3. Вихідні дані до проєкту *технічна документація та теоретичні дані.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються) *Опис та аналіз предметної області, аналіз аналогів у даній області, дослідження технологій для реалізації проєкту, реалізація картографічного сервісу та тестування розробленого проєкту.*
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень) *функціональна схема, принципова схема, структурна схема*

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання_____

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	*Затвердження теми проєкту*	*10.12.2020-15.12.2020*	
2.	*Вивчення та аналіз завдання*	*15.12.2020-15.03.2021*	
3.	*Розробка архітектури та загальної структури системи*	*15.03.2021-25.03.2021*	
4.	*Розробка структур окремих підсистем*	*25.03.2021-5.04.2021*	
5.	*Програмна реалізація системи*	*5.04.2021-15.04.2021*	
6.	*Оформлення пояснювальної записки*	*15.04.2021-31.05.2021*	
7.	*Захист програмного продукту*	*25.04.2021*	
8.	*Передзахист*	*23.05.2021*	
9.	*Захист*	*15.06.2021*	

Студент-дипломник _____
(підпис)

Керівник проєкту _____
(підпис)

Анотація

В бакалаврському дипломному проєкті розглянуто існуючі аналоги картографічних сервісів, а також принцип їх роботи. Було проведено аналіз їхніх сильних та слабких сторін. Розглянуто технології для реалізації власного картографічного сервісу. В результаті роботи було розроблено картографічний сервіс на базі PostGIS та програму для спрощення геоструктур даних. Картографічний сервіс дозволяє переглядати мапу а також взаємодіяти з нею. Розроблений програма зменшує об’єм даних, що зберігаються в базі. Картографічний сервіс побудовано на основі ряду інструментів, а саме Apache для обробки запитів веб-бразуера, Mapnik для візуалізації плиток та база даних PostgreSQL для зберігання даних в ній.

Annotation

In the bachelor's dimploma project considers the existing analogues of cartographic services, as well as the principle of their work. An analysis of their strengths and weaknesses was conducted. Technologies for realization of own cartographic service are considered. As a result, a mapping service based on PostGIS and a program for simplifying data geostructures were developed. The map service allows you to view the map and interact with it. The developed program reduces the amount of data stored in the database. The mapping service is based on a number of tools, namely Apache for processing web browser queries, Mapnik for tile visualization and PostgreSQL database for storing data in it.

Опис альбому

до дипломного проекту

на тему: «Картографічний сервіс на базі PostGIS»

Київ – 2021 року

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпляра	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467100.002 ТЗ***	Картографічний сервіс на	1		
			базі PostGIS			
			Технічне завдання			
	А4	***ІАЛЦ.467100.003 ПЗ***	Картографічний сервіс на	59		
			базі PostGIS			
			Пояснювальна записка			
	А3	***ІАЛЦ.467100.004 Д1***	Картографічний сервіс на	1		
			базі PostGIS			
			Функціональна схема			
	А4	***ІАЛЦ.467100.005 Д2***	Картографічний сервіс на	1		
			базі PostGIS			
			Принципова схема			
	А3	***ІАЛЦ.467100.006 Д3***	Картографічний сервіс на	1		
			базі PostGIS			
			Структурна схема			

Зм	***Лист***	***№ докум.***	***Підп.***	***Дата***	

ІАЛЦ.467100.001 ОА

Розроб.	Корсун Р.В.			*Картографічний сервіс на базі PostGIS* *Опис альбому*	Літ.	Аркуш	Аркушів
Перев.	Іваніщев Б.В.					1	1
					НТУУ “КПІ” ФІОТ ІО-71		
Н. Контр.	Сімоненко В. П.						
Затверд.	Іваніщев Б.В.						

Технічне завдання
до дипломного проекту

на тему: «Картографічний сервіс на базі PostGIS»

Київ – 2021 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Найменування: «Картографічний сервіс на базі PostGIS». Область застосування: використання інтерактивних карт на веб-сайтах різного призначення.

2. ПРИЧИНИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є вдосконалення бази даних картографічного сервісу, що розробляється на базі PostGIS.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література, публікації в спеціалізованих періодичних виданнях, довідники по інтерактивним картам, публікації в мережі Інтернет по даній темі.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- Самодостатність – картографічний сервіс повинен містити тільки необхідні і достатні дані, що використовуються на відповідних веб-сайтах.
- Оптимальний розмір спрощеної бази даних.
- Висока швидкість рендерингу карт, що передбачає якнайшвидший час виконання для отримання однієї плитки.

5.2. Вимоги до програмного забезпечення

- Pycharm 2021.1.1 та вище.
- Операційна система Ubuntu Linux 12.04 та вище.

- Python 3.7 та вище.

5.3. Вимоги до апаратної частини

- Підключення до Internet
- Наявність сучасного двоядерного CPU.
- Оперативна пам’ять не менше 4 Гбайт.
- Вільний простір твердотільного накопичувача не менше 20 Гбайт

6. ЕТАПИ РОЗРОБКИ

	Дата
6.1. Вивчення необхідної літератури	21.02.2021
6.2. Складання і узгодження технічного завдання	08.03.2021
6.3. Написання вступної частини та огляд рішень	22.03.2021
6.4. Розробка архітектури системи	04.04.2021
6.5. Написання програмної частини	18.04.2021
6.6. Тестування та виправлення помилок	19.05.2021
6.7. Оформлення документації дипломного проекту	25.05.2021

Пояснювальна записка
до дипломного проєкту

на тему: «Картографічний сервіс на базі PostGIS»

Київ – 2021 року

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

GIS	(Geographic Information System) Географічна інформаційна система
HTTP	(HyperText Transfer Protocol) Протокол передачі гіпертекстових документів
СУБД	Система управління базами даних
XML	(Extensible Markup Language) Розширювана мова розмітки
JSON	(JavaScript Object Notation) Запис об’єктів JavaScript
API	(Application Programming Interface) Прикладний програмний інтерфейс
WKT	(Well-Known Text) Текстовий формат представлення векторної геометрії та опису систем координат
CSS	(Cascading Style Sheets) Каскадні таблиці стилів
SQL	(Structured Query Language) Мова структурованих запитів
AGG	(Anti-Grain Geometry) Антизернова геометрія
PNG	(Portable Network Graphics) Растровий формат збереження графічної інформації

ВСТУП

Картографія протягом останніх десятиліть постійно розширюється і удосконалюється, перетворюючись із нікому невідомої галузі в широко розповсюджувану технологію, яка стрімко розвивається. Програмне забезпечення для веб-картографії можна описати як сукупність програмних засобів, які здатні вирішувати різні задачі – від вводу та редагування елементарних векторних даних до налаштування параметрів цілих картографічних веб-сервісів в каталозі просторових метаданих.

Актуальність теми

Веб-картографія ознаменувала собою демократизацію доступу широких слоїв населення до географічних даних. Від етапу свого зародження в середині 90-х років і до сьогодення технології веб-картографії пройшли значний шлях свого розвитку. На сьогодні неможливо представити користувача мережі Інтернет без використання картографічних веб-сервісів. В мережі щодня росте кількість веб-сайтів, що використовують інтерактивні карти [1].

Мета і задачі дослідження

Метою даної роботи є розробка картографічного сервісу та вивчення можливостей бази даних PostGIS, а також їх вдосконалення за допомогою зменшення об’єму даних, що зберігаються в цій базі шляхом об’єднання та зменшення геоструктур даних.

Для виконання поставленого завдання дипломної роботи необхідно:

- Опрацювати літературу, що стосується обраної теми дипломної роботи;
- Провести огляд існуючих картографічних сервісів та бібліотек;
- Автоматизувати перебір та спрощення бази даних;
- Створити картографічний сервіс для перевірки розробленого алгоритму спрощення бази даних;
- Зробити висновки за отриманими результатами спрощення бази даних.

Практичне значення

Розроблений картографічний сервіс дозволить ефективніше зберігати дані на сервері шляхом об’єднання та спрощення геоструктур даних. Також із застосуванням автоматизованого перебору бази даних вдасться зменшити розмір даних, що зберігатимуться в базі PostGIS.

Ключові слова

Веб-картографія, тематичні карти, інтерактивні карти, GIS-технології, геоінформаційне картографування, веб-технології.

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ КАРТОГРАФІЧНИХ СЕРВІСІВ

1.1. Опис предметного середовища

Картографічний сервіс – це спеціальна інформаційна система, що представляє просторові дані у вигляді інтерактивної карти. Такі сервіси використовуються для надання місця розташування різних будівель, вулиць, для відображення певної бази маршруту чи просто маршруту з точки А в точку Б.

Картографічний сервіс створює карти, об’єкти та дані атрибутів всередині багатьох типів клієнтського програмного забезпечення [2]. Картографування та розуміння всієї проблеми користувача в цілому є фундаментальною частиною переконання у створенні правильних сервісів для користувачів [3].

Картографічні сервіси представляють іншим користувачам карту, що знаходиться на сервері. Картографічні сервіси спроектовані таким чином, що можуть працювати з великою кількістю сценаріїв в мережі Інтернет. Одним і тим же картографічним сервісом може користуватись один користувач, у веб-застосунку – другий, а в мобільній програмі – третій. Це і є загальні підстави для використання картографічних сервісів[4].

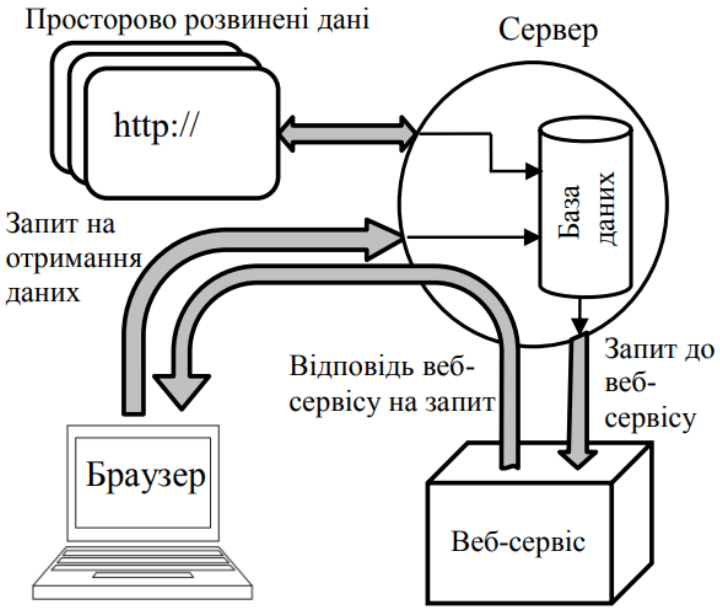


Рис. 1.1 Функціональна схема веб-сервісу

Алгоритм роботи такий: дані на сервер відправляються з просторово рознесених пристроїв за допомогою протоколу HTTP з наступним збереженням у базі даних у організованому вигляді. Задача формування карти області дослідження, її масштабування та рендеринг повністю покладається на картографічний сервіс, який швидко завантажує карти по частинам (у вилгяді «тайлів»).

Для відображення власних даних на картах потрібно використовувати наявні прикладні програмні інтерфейси або можливості інтеграції з СУБД. Користувач генерує запит на відображення даних на веб-сайті, що виконує накопичення та збір інформації з просторово-рознесених пристроїв, після чого автоматично створюється запит на вибірку даних до бази даних. Результат оформлюється у вигляді XML файлу або у форматі JSON. Згодом на основі цих даних на стороні клієнта за допомогою JavaScript виконується накладання даних на карти з прив'язкою їх по координатам. Узагальнену схему використання таких сервісів зображено на рис. 1.1 [5].

У широкому значенні веб-картографія – це сукупність технологій, які зв’язані з створенням різних інтерактивних карт, їх розміщенням та обробкою в мережі Інтернет. В минулому столітті, коли почали з’являтись перші картографічні програми, вони були в основному для вузького кола спеціальностей (геологія, навігація, статистика, землеустрій, бізнес-дані тощо), призначались виключно в професійних цілях і мали неінтерактивний характер.

На даний момент ситуація дуже змінилась: в мережі Інтернет стрімко створюється глобальна, інтерактивна інфраструктура веб-картографії, що має окрім професійних користувачів, мільйони звичайних юзерів-учасників по всьому світу (для прикладу лише спільнота OpenStreetMap налічує близько 400 000 користувачів).

Сучасне програмне забезпечення, доступ до бази даних і можливість швидкої об’єднаної комунікації дозволяють колективно створювати «онлайн» електронні карти, доступні всім з геопросторовою інформацією,

що оновлюється в режимі реального часу. Діапазон застосування надзвичайно широкий: від спеціалізованих до суто побутових потреб. Дуже ефективним цей алгоритм виявився у попередженні й ліквідації надзвичайних ситуацій та гуманітарних катастроф, що спонукало до виникнення і стрімкого збільшення сегменту так званої кризової веб-картографії. Оскільки головною ідеєю кризової веб-картографії є поширена участь користувачів у зборі потрібних даних (краудсорсинг), рушієм її розвитку на даному етапі є недержавні, зазвичай добровільні онлайн-організації та спільноти.

Одночасно з тим, чимдалі масштабним стає використання сучасного потенціалу кризової картографії офіційними структурами – урядами, міжнародними організаціями, рятувальними органами. Для збільшення ефективності інтерактивних карт проектів краудсорсингові дані все частіше наповнюються професійною інформацією такою як супутникові дані[6].

1.2. Теоретичні відомості

Картографія – це наука, яка вивчає, розробляє і створює географічні карти. Вона поділяється на цілий ряд технічно-наукових дисциплін: математичну картографію, картознавство, складання, видавництво карт і оформлення карт.

Географічна інформаційна система (ГІС) – це система, що використовується для збору, зберігання, аналізу та відображення просторових та географічних даних. Ця система поєднує в собі традиційні операції роботи з базами даних, такими як запит і статичний аналіз, з перевагами повноцінного відображення та просторового аналізу, які надає карта.

ГІС характеризують:

- розвинуті аналітичні функції;
- змога керувати великими об’ємами даних;
- інструменти для вводу, обробки та виводу просторових даних;

Прикладний програмний інтерфейс *(англ. Application Programming Interface, API)* – це складова частина серверу, яка отримує запити і відправляє відповіді.

Шар (Map Layer) – покриття, яке розглядається в ситуації його змістовного позначення (рослинність, рельєф, адміністративний поділ тощо). **Шар**, зазвичай, є однаковим не тільки за тематикою, але і по типам об'єктів (лінійні, полігональні, растрові, точкові).

1.3. Огляд наявних аналогів

Google Maps (рис. 1.2) **–** один із найвідоміших картографічних сервісів. Три основні частини Google Maps – це безпосередньо самі карти, знімки з супутника и Google Street View. Карти від Google також надають можливості для використання своїх карт в сторонніх сервісах[7].

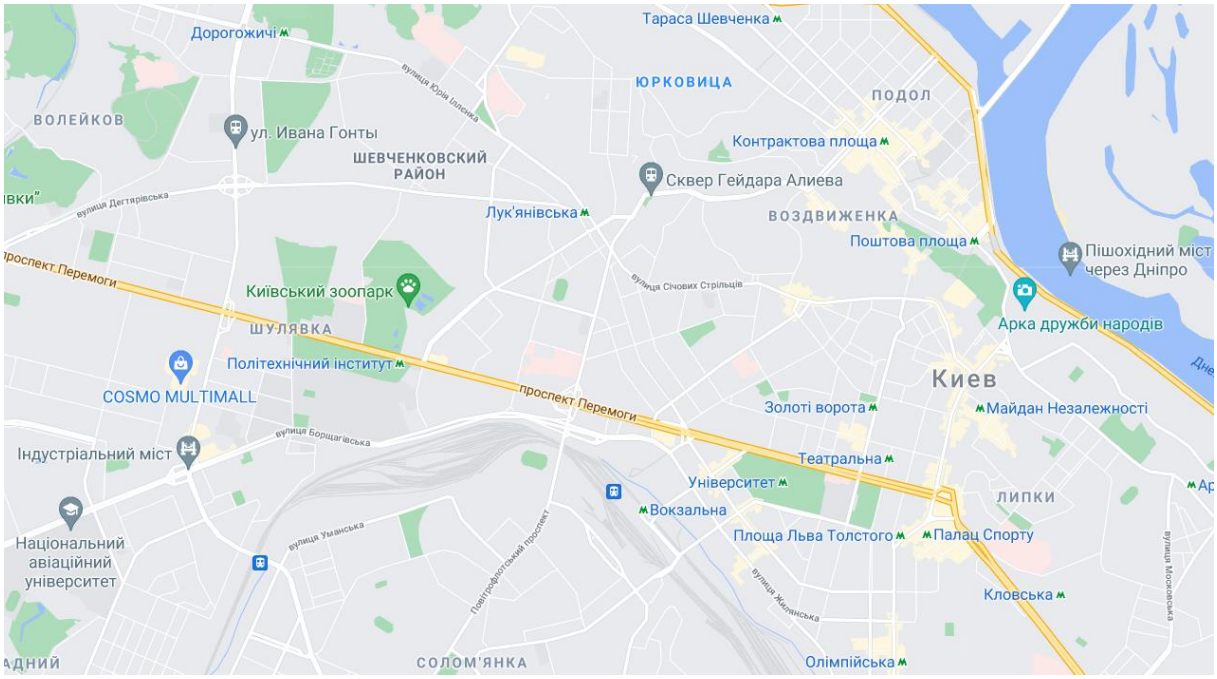


Рис. 1.2 Відображення карт в Google Maps

Більшість користувачів надають перевагу сервісам Google, протягом років повністю занурившись в її екосистему вони починають розуміти, як все-таки зручно, що всі програми інтегровані.

Саме з приходом карт і маршрутів громадського транспорту на Google Maps більшість стали користуватись автобусами у великих містах замість метро.

Перелік можливостей:

- керування навігатором за допомогою голосу;
- пошук припаркованого автомобіля;
- можливість будь-кому відправити свої геодані;
- перевірка ймовірного часу прибуття в будь-який час;
- перевірка заторів на дорогах;
- можливість використання офлайн-карт без використання мережі Інтернет (завчасно завантаживши потрібні карти на свій смартфон);

Перевагами є:

- Спільний доступ (можливість ділитись картами з друзями, родиною та колегами).
- Кілька режимів транспортування (Веб-сайт Google Maps дає вказівки щодо поїздок на машині, велосипеді або громадським транспортом).
- Багатство інформації (Карти Google надають схему розташування доріг, міст і населених пунктів, державні межі, географічні об’єкти).
- Google Street View – функція, що дозволяє переглядати вулиці, вітрини магазинів, визначні місця з точки зору водія.
- Внутрішні карти деяких аеропортів, музеїв та інших об’єктів.
- Можливість перегляду різних видів карти, які можна переключати в залежності від потреб.
- Пошук оптимального маршруту при вводі двох чи більше місць, які будуть відвідані.

З недоліків виділяють:

- Обмежена точність - інформація надана картами Google може бути неточною або застарілою, наприклад пішохідні та велосипедні напрямки можуть пропонувати маршрути, на яких відсутні тротуари, пішохідні чи велосипедні доріжки.
- Використання злочинцями - на зображеннях вуличного перегляду злочинці можуть помітити речі, які легко виставити через вікна або відчинені двері, також у заможних кварталах можна виявити будинки

до яких легко проникнути або де припарковані транспортні засоби, які можуть містити дорогі речі.

- Образливий та шокуючий контент, наприклад, камера «Street View» час від часу знімає зображення злочинної поведінки.
- Відсутність деякої інформації - залежно від країни та місця, яке Вас цікавить, можна побачити, що цілі будівлі розмиті в режимі перегляду вулиць. Це робиться за запитом через проблеми конфіденційності, але це також суттєво обмежує корисність інформації.

Bing Maps [8] (рис. 1.3) – це картографічний сервіс, що працює під управлінням Bing Microsoft. Платформа включає плитки карт, API для вбудованих карт, маршрутизацію тощо.

Bing Maps має сильного конкурента, а саме Google Maps. Проте компанія, що розвивається, готова скласти конкуренцію, обіцяючи унікальні особливості.

Карти Bing часто оновлюють і розширюють географічні райони, охоплені їх зображеннями, причому нові оновлення випускаються приблизно щомісяця. Кожен випуск зображень зазвичай містить більше 10 ТБ зображень.

Однак необхідний проміжок часу перед оновленням зображень означає, що аерофотознімки та зображення "Пташиного польоту" для певного місця можуть іноді застарівати на кілька років. Це особливо помітно в місцях, які зазнали бурхливого недавнього розвитку або зазнали інших кардинальних змін після створення знімків, таких як райони, що постраждали від стихійних лих.

Перевагами є:

- Вид з висоти пташиного польоту – відображаються аерофотознімки, зняті з низько літаючих літаків. На відміну від аерофотознімка зверху вниз, знятого супутником, знімки з висоти пташиного польоту робляться під косим кутом 45 градусів, показуючи борти та дахи будівель, що дає краще сприйняття глибини географії.

- StreetSide – копія режиму перегляду вулиць Google, проте помітною відмінністю є те, що Bing Maps розбиває екран, щоб відобразити вид на вулицю зверху та карту внизу сторінки.
- Підтримка GeoJSON: GeoJSON – це один з найбільш часто використовуваних форматів файлів для зберігання та передачі географічних даних. Дані можна легко імпортувати та експортувати.

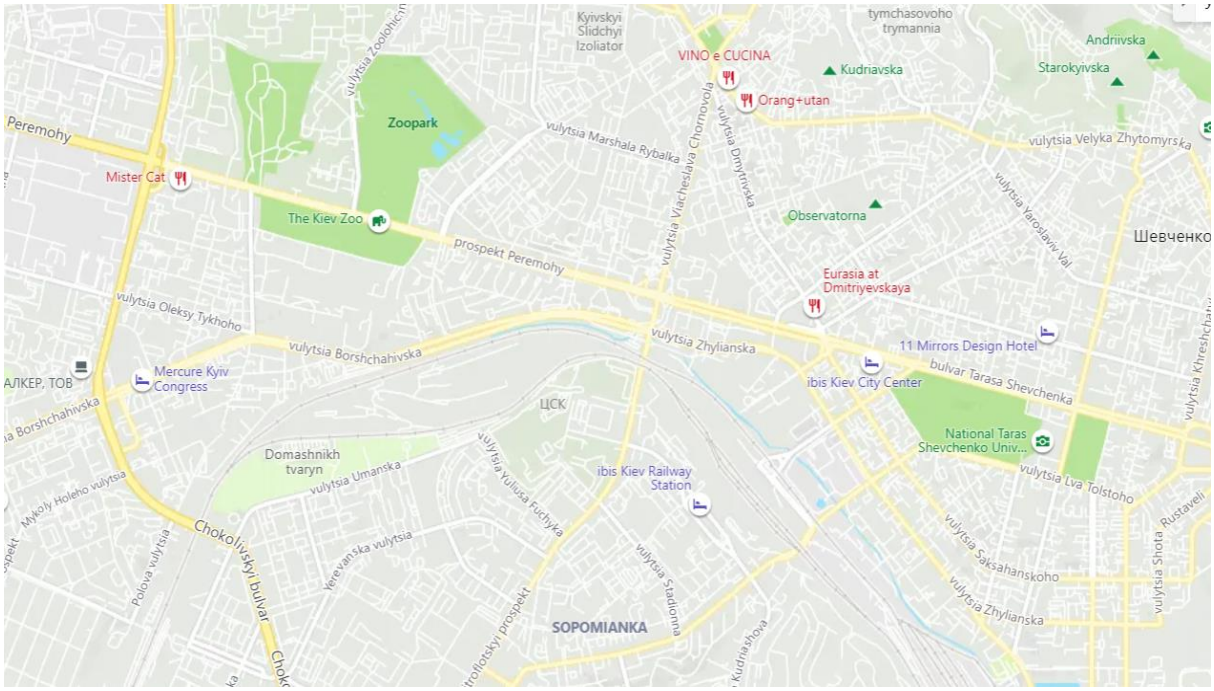


Рис. 1.3 Відображення карт в Bing Maps

Основними недоліками є:

- Відсутнє відображення робочого часу та умов більшості установ або закладів громадського харчування.
- Дуже схожий на Google Maps, проте має набагато обмеженішу функціональність, ніж Google Earth або World Wind.
- Для тривимірного перегляду зображень необхідно завантажити велику кількість даних.

OpenStreetMap (рис. 1.4) – це картографічний сервіс, створений спільнотою волонтерів-картографів, які додають дані про дороги, вулиці, кафе, вокзали тощо. OSM в народі ще називають Вікіпедією картографічного світу.

OSM підтримується некомерційною організацією OpenStreetMap Foundation. Дані OSM є вільно доступними для візуалізації, запиту, завантаження та модифікації за відкритими ліцензіями [9].

Рис. 1.4. Відображення карт в OpenStreetMap

OSM працює у стилі, подібному до Вікіпедії, в якому практично всі функції відкриті для редагування будь-яким членом спільноти користувачів. OSM була заснована в 2004 році і зараз налічує близько двох мільйонів зареєстрованих користувачів. Дані в OSM являються відкритими, достатньо вказати авторські права OpenStreetMap та його учасників.

Серед основних переваг OSM можна виділити:

- Використання даних не вимагає жодних витрат: у деяких регіонах OSM може бути єдиним вільно доступним джерелом векторних даних ГІС із високою роздільною здатністю.
- Вихідні дані доступні для завантаження у похідних картографічних продуктах: тут OSM відрізняється від механізму краудсорсингу, який розгорнули Google Maps, який називається Map Maker. Картографи не можуть завантажувати та повторно використовувати дані Google Map Maker, тоді як OSM доступний кожному, хто має достатньо технічних ноу-хау для отримання даних.

- Оскільки OSM дозволяє людям додавати будь-який тип об’єктів, він може включати багатший та соціально цінніший набір об’єктів, ніж комерційні чи державні карти. Можливості включають дерева, пандуси для інвалідних візків, банки з їжею, крани для питної води тощо.
- Дані OSM є гнучкими і можуть швидко оновлюватися у разі відкриття нового бізнесу, знесення мосту тощо. На відміну від цього, комерційні та державні карти, як правило, оновлюються за фіксованими циклами.

Недоліками є:

- Немає систематичної перевірки якості даних: ви використовуєте дані на свій страх і ризик і повинні бути обережними при використанні інформації OSM для критично важливих функцій.
- Детальність та точність даних OSM залежить від місця: великі міста добре деталізовані, в той час як маленькі села не дуже. Але і ця проблема вирішується за допомогою проекту Missing Maps від OSM.

Основний сервер OSM має обмеження на використання, оскільки компанія не має джерел прибутку, вона не може постачати плитки для комерційного використання у великих масштабах. Краще буде використовувати свій сервер плиток або сервер іншого провайдера.

Leaflet (рис. 1.5) – це бібліотека, яка займає лідируючі позиції серед бібліотек JavaScript з відкритим кодом для інтерактивних карт. Розроблена Володимиром Агафонкіним та підтримується великою громадою волонтерів. [10].

Більше того, бібліотека повністю безкоштовна у користуванні та розповсюджується за 2-пунктною ліцензією BSD. Leaflet розроблена з урахуванням простоти, продуктивності та зручності використання. Бібліотека ефективно працює на всіх основних десктопних та мобільних платформах, може бути розширена безліччю плагінів, має прекрасний,

простий у використанні та добре задокументований API і простий, читабельний вихідний код.

Рис. 1.5 Відображення карт за допомогою Leaflet.js

Leaflet має усі функції, які потрібні більшості розробників, які створюють карти. Крім того, додаткової потужності Leaflet додають його плагіни. Ядро функціонально підтримує лише формат GeoJSON. Тим не менше, підтримка інших форматів, таких як CSV, WKT, TopoJSON, GPX можлива за допомогою плагінів.

Перевагами є:

- Простий у використанні.
- Інтерактивний.
- Може використовуватися з CartoDB, MapBox, Google Maps тощо.
- Відкритий вихідний код.
- Доступно багато плагінів, за допомогою яких можна розширити можливості Leaflet, якщо це потрібно.
- Чудова документація, легко розібратись і почати роботу з Leaflet.

Недоліками є:

- Потрібне вивчення JavaScript – звичайний користувач настільного ПК не зможе відразу почати і створювати карти, для цього потрібно мати базові знання JS.
- Можливо, потрібно буде налаштувати вашу інформацію, за допомогою програмування геоінформаційних систем, наприклад QGIS.

- Можливо, доведеться заплатити за базовий шар карти.
- Завантажується повільніше, ніж Google Maps чи MapBox. Якщо у користувача мала швидкість інтернету, то завантаження карти займе певний час.

OpenLayers – це потужна бібліотека JavaScript з відкритим кодом для створення інтерактивних веб-карт, доступних для перегляду майже в будь-якому веб-браузері. Оскільки це бібліотека на стороні клієнта, вона не вимагає спеціального програмного забезпечення та налаштувань на стороні сервера. Полегшує розміщення динамічної карти на будь-якій веб-сторінці. Ця бібліотека може відображати плитки карт, векторні дані та маркери, завантажені з будь-якого джерела. OpenLayers розроблено для подальшого використання географічної інформації всіх видів.

OpenLayers пропонує більше функціональних можливостей, ніж Leaflet, але і вимагає більше часу для запуску [11]. Наприклад, потрібно використовувати проекції, для того щоб створити лише просту карту. Документація містить QuickStart, навчальні посібники та багато прикладів. Але, на жаль, деякі з них вже застаріли. Документація API добре структурована, але оскільки вона дуже велика, в ній можна легко розгубитись на самому початку. Потужність та гнучкість – дві найсильніші характеристики OpenLayers. Бібліотека має всі необхідні особливості для основної функціональності. OpenLayers підтримує GeoJSON, GeoRSS, KML, GML та картографічні дані з будь-якого джерела, використовуючи стандарти OGC як WMS або WFS.

Основні особливості OpenLayers:

- Плиточні шари – можливість витягнути плитки з OSM, MapBox, Bing, Stamen та будь-якого іншого джерела, яке можна знайти. Також підтримуються служби картографування OGC.
- Векторні шари – можливість візуалізувати векторні дані з GeoJSON, TopoJSON, KML, GML, векторних плиток MapBox та інших форматів.

- Передовий та швидкий – OpenLayers використовує Canvas 2D, WebGL та всі новітні технології HTML5. Створюйте легкі, власні карти лише з потрібними вам компонентами.
- Легко налаштувати та розширити – стилізування елементів керування картою напряму через CSS. Можливість підключитись до різних рівнів API або використовувати сторонні бібліотеки для налаштування та розширення функціональних можливостей.

Перевагами є:

- Швидкість: лише для передачі зображення потрібен час.
- Стабільність: немає потреби у бекенді після генерації плиток, лише для розміщення доступних файлів зображень карти.
- Підтримка великого навантаження: кожна плитка – це лише невелике зображення, яке є достатньо легким і не буде перенавантажувати ваш браузер.
- URL-адреси плиток є попередньо визначеними; отже, їх можна подавати через CDN.

Недоліками є:

- Велике споживання сервера для попередньої обробки (генерації) плиток.
- Споживання великого дискового простору (цілі терабайти для усього світу) коли плитки попередньо згенеровані.
- Візуальні ефекти розриву при безперервному збільшенні карти, оскільки плитки задумані з урахуванням рівнів перегляду. Якщо не використовувати рівні перегляду, зображення будуть розтягуватись і будуть здаватись розмитими.
- OpenLayers обмежений для обслуговування кількох проекцій: набір плиток повинен бути згенерований для кожної проекції.

Таблиця 1.1. Порівняння картографічних сервісів

№	Назва картографічного сервісу	Джерело даних	Відображення елементів на карті	Режим відображення карт	Елементи керування картою	Користувацькі об’єкти поверх карти
1	Google Maps	Має власну базу даних	- карта всього світу, краще відображені великі міста; - об'єкти нормально видно при достатнього збільшенні карти;	- супутник; - схема; - режим перегляду вулиць	- вибір режиму відображення карти; - масштабування; - прокладання маршрутів; - перехід до повноекранного режиму;	+
2	Bing Maps	Має власну базу даних	- карта всього світу, краще відображені великі міста; - об'єкти нормально видно тільки при достатнього збільшенні карти;	- схема; - режим перегляду «з висоти пташиного польоту» - супутник;	- вибір режиму відображення карти; - масштабування; - прокладання маршрутів; - перехід до повноекранного режиму;	+
3	OpenStreet Map	База даних, що редагується спільнотою	- Карта усього світу, яка доповнюється користувачами. - Об'єкти добре видно при незначному збільшенні карти.	- схема	- кнопки переміщення; - масштабування;	Через сторонні бібліотеки
4	Leaflet	Використовує OSM, але може працювати і з іншими базами даних	Може використувати карти OSM, а також карти інших картографічних сервісів.	- схема	-масштабування; - кнопки переміщення;	Через сторонні плагіни
5	OpenLayers	Може працювати з різними базами даних	Може використувати карти OSM, а також карти інших картографічних сервісів.	- схема	-масштабування; - кнопки переміщення;	Через сторонні бібліотеки

ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ

Проведений огляд наявних картографічних сервісів, показав, що:

1. Основним завданням картографічних сервісів є відображення карт для користувачів, прокладання маршруту з початкової точки до кінцевої та надання місця розташування пам’яток.

2. Велика кількість картографічних сервісів сприяла вдосконаленню відображень даних на веб-сервісах та їх частоти оновлень, а також до створення спільнотою волонтерів власних картографічних сервісів, для того щоб задовольнити кінцевих користувачів в залежності від їх потреб у використанні картографічних сервісів.

3. Проаналізувавши готові картографічні сервіси стали очевидними ряд недоліків:

- Не всі картографічні сервіси є повністю безкоштовними: наприклад в сервісах Google Maps, якщо перевищити певний ліміт, то за подальше використання їхніх карт на власному сервері доведеться платити;
- Відсутність певної інформації;
- Для роботи з бібліотеками картографічних сервісів потрібно мати базові знання JavaScript;
- Дуже великі об’єми даних, що завантажуються на сервер, якщо брати цілі країни;

4. Дивлячись на згадані вище недоліки, можна визначити основні функції картографічного сервісу:

- Доступність усіх можливостей безкоштовно;
- Приємний та зрозумілий інтерфейс;
- Наповненість інформацією;
- База даних картографічного сервісу повинна бути максимально оптимізована.

РОЗДІЛ 2. ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОЕКТУ

2.1. Структура картографічного сервісу

Розроблюваний картографічний сервіс представляє собою стек програмних застосунків для відображення карт у вигляді:

- PostGIS – це просторове розширення для реляційної бази даних PostgreSQL з відкритим вихідним кодом. PostGIS додає підтримку географічних об’єктів, що дозволяє виконувати запити про розташування в SQL.
- Mapnik – це програма з відкритим вихідним кодом для рендерингу даних з різних джерел, в тому числі і PostGIS в растровий вигляд, тобто картинки. Mapnik написана на C++ і має до мови програмування Python. Використовує бібліотеку AGG і дає можливість згладжувати об’єкти на карті з великою точністю.
- Rednerd, mod_tile – програма та модуль apache, що використовуються для виклику Mapnik, кешування плиток і видачі їх за допомогою apache. mod_tile забезпечує динамічне поєднання ефективного кешування та візуалізації на ходу. Через його динамічну візуалізацію на диску потрібно зберігати лише незначну частину загальної плитки, зменшуючи необхідні ресурси.
- Leaflet.js – JavaScript-бібліотека з відкритим вихідним кодом для відображення інтерактивних карт на сайті. Ефективно працює на всіх пристроях.

2.2. Аналіз відомостей про СУБД

Система управління базами даних (СУБД) представляє собою комплекс програмного забезпечення за допомогою якого можна створювати бази даних (БД) та проводити над ними різні операції: оновлювати, видаляти, вибирати, редагувати і т.д. СУБД гарантує цілісність і безпеку даних, що зберігаються в базі і дозволяє видавати доступ до адміністрування БД.

Ключовими можливостями СУБД є:

- Робота з даними, розміщеними на зовнішніх накопичувачах.
- Робота з даними, що знаходяться в оперативній пам’яті з використанням дискового кешу.
- Ведення звітності, що стосується: резервування, редагування, бекапу даних і т.д.
- Підтримка різних мов баз даних (для роботи і визначення конкретних типів даних).

СУБД складається з:

- Ядра. Підтримує звітність, відповідає за управління даними в оперативній пам’яті і на зовнішніх накопичувачах.
- Процесора мови БД. Дозволяє оптимізувати запити на створення та редагування даних.
- Підсистеми підтримки часу виконання. Дозволяють інтерпретувати програмне забезпечення для підтримки роботи з БД, створювати користувацькі інтерфейси взаємодії з СУБД.
- Допоміжне ПЗ. Набір утиліт, що дозволяють розширити можливості взаємодії з СУБД (в тому числі, і по обслуговуванню).

СУБД розділяють на окремі типи, спираючись на моделі даних, методи представлення доступу до БД і рівню розподілу.

В залежності від моделі даних СУБД бувають:

- мережевими;
- ієрархічними;
- реляційними;
- об’єктно-реляційними;
- об’єктно-орієнтованими;

Відповідно до методу представлення доступу до бази даних СУБД розділяють на:

- вбудовані;
- «клієнт-сервер»;
- «файл-сервер»;

За рівнем розподілу СУБД бувають:

- Розподіленими (складові елементи одної СУБД можуть бути на різних машинах);
- Локальними (Всі елементи СУБД розміщені на одній машині).

СУБД можуть працювати з даними на зовнішніх накопичувачах шляхом відкладеного та безпосереднього запису.

При відкладеному записі зміни в БД записуються в буферах обміну на зовнішніх накопичувачах, доки не наступить:

- Контрольна точка, вказана заздалегідь.
- Нехватка вільного простору для запису на накопичувачі.
- Нехватка оперативної пам’яті для забезпечення роботи буферів.
- Зупинка БД.

СУБД, що за працюють за принципом безпосереднього запису, миттєво записують всі зміни в базах даних на зовнішніх накопичувачах, як поступить підтвердження якої-небудь транзакції. Цей метод розумно використовувати тільки при використанні високоефективної зовнішньої пам’яті.

Для створення картографічного сервісу оптимально буде використовувати об’єктно-реляційну СУБД PostgreSQL з розширенням PostGIS. Оскільки це система управління реляційними базами даних з відкритим вихідним кодом, яка підтримує більшу частину стандарту SQL і пропонує багато сучасних функцій.

Окрім того, користувачі можуть всяко розширювати можливості PostgreSQL, наприклад створюючи свої:

- типи даних;
- функції;
- оператори;
- агрегатні функції;
- методи індексування;
- процедурні мови;

А завдяки вільній ліцензії, PostgreSQL дозволяється безкоштовно використовувати, змінювати і розповсюджувати всім і для будь-яких цілей – особистих, комерційних або навчальних.

2.3. **Аналіз структури системи управління базами даних PostGIS**

PostGIS додає додаткові типи (geometry, geography, raster та інші) до бази даних PostgreSQL. PostGIS також додає функції, оператори та вдосконалення індексу, які застосовуються до цих просторових типів. Ці додаткові функції, оператори, прив'язки індексів та типи збільшують потужність базової СУБД PostgreSQL, роблячи її швидкою, багатофункціональною та надійною просторовою системою управління базами даних.

PostGIS представляє такі можливісті:

- Обробка та аналітичні функції як для векторних, так і для растрових даних для сплайсингу, нарізання, перетворення, перекласифікації та збору/об’єднання з допомогою SQL.
- Алгебра растрових карт для дрібнозернистої обробки растру.
- Функції просторового відтворення SQL, що викликаються, як для векторних, так і для растрових даних.
- Підтримка імпорту/експорту векторних даних шейп-файлу ESRI за допомогою командного рядка та інструментів, що містять графічний інтерфейс, також підтримка великої кількості форматів через інші сторонні інструменти з відкритим вихідним кодом;

- Упакований командний рядок для імпорту растрових даних із багатьох стандартних форматів: GeoTiff, NetCDF, PNG, JPG і т.д.
- Візуалізація та імпорт функцій підтримки векторних даних для стандартних текстових форматів, таких як KML, GML, GeoJSON, GeoHash та WKT за допомогою SQL.
- Підтримка тривимірних об’єктів;
- Підтримка топології мережі;

Об’єкти геоінформаційних систем, що підтримуються PostGIS є надмножиною стандарту «Прості функції», визначеного Консорціумом OpenGIS (OGC). Специфікація OpenGIS визначає два стандартних способи вираження просторових об’єктів: у форматі Well-Known Text (WKT) та у форматі Well-Known Binary (WKB). І WKT, і WKB включають інформацію про тип об’єкта та координати, що використовують об’єкт.

Прикладами текстового подання просторових об’єктів є:

- POINT(0 0);
- POINT Z(1 1 1);
- POINT ZM(1 1 3 4 5 5);
- LINESTRING(1 1, 2 2, 3 4);
- POLYGON((0 0, 5 0, 5 5, 0 5, 0 0), (1 1, 3 1, 3 3, 1 3, 1 1));
- MULTIPOINT((0 0), (2 3));
- MMULTIPOINT Z ((1 1 1),(2 3 4))
- MULTILINESTRING ((1 1,2 2,3 4),(3 4,4 3,6 5))
- MULTIPOLYGON (((0 0,5 0,5 5,0 5,0 0), (1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -3,-3 -3,-3 -1,-1 -1)))
- GEOMETRYCOLLECTION (POINT(3 4),LINESTRING(3 4,4 5))

Специфікація OpenGIS також вимагає, щоб формат внутрішнього зберігання просторових об'єктів містив ідентифікатор просторової системи посилань (SRID). SRID – це унікальний ідентифікатор, що відповідає певній системі координат. SRID необхідний при створенні просторових об'єктів

для вставки в базу даних. Деякі бази даних і просторові типи, наприклад PostGIS Geometry в PostgreSQL використовує заздалегідь указаний набір кодів EPSG і можна використовувати просторові прив’язки тільки з цим ідентифікатором SRID.

2.4. Огляд інструменту управління реляційними базами даних

Для управління та відслідковування змі у базі даних мого проекту було обрано використовувати програмне забезпечення pgAdmin4.

pgAdmin4 (рис 2.1) – провідне програмне забезпечення з відкритим вихідним кодом для PostgreSQL, розроблене для задоволення потреб як новачків так і досвідчених користувачів, представляє потужний графічний інтерфейс, який спрощує створення, обслуговування та використання об’єктів бази даних.

pgAdmin4 підтримує всі функції PostgreSQL, від написання простих SQL-запитів до розробки складних баз даних. Він призначений для запитів до активної бази даних (в режимі реального часу), що дозволяє бути в курсі всіх змін та правок.

Можливості pgAdmin4:

- Автоматичне виявлення та підтримка об’єктів, виявлених під час виконання;
- Live SQL Query Tool з прямим редагуванням даних;
- Підтримка адміністративних запитів;
- Редактор SQL з підсвічуванням синтаксису;
- Перероблений графічний інтерфейс;
- Потужні діалоги управління і інструменти для загальних задач;
- Допоміжні повідомлення про помилки;
- Корисні підказки;
- Інтерактивний помічник та інформація про використання діалогів і інструментів pgAdmin;

Коли відкривається pgAdmin, в інтерфейсі є рядок меню і вікно, розділене на дві панелі: елемент управління «Дерево браузеру» на лівій панелі і браузер з вкладками на правій панелі.

Рис. 2.1 Вступне вікно pgAdmin4.

За допомогою інструменту Query Tool (рис 2.2) можна напряму виконувати запити до бази даних:

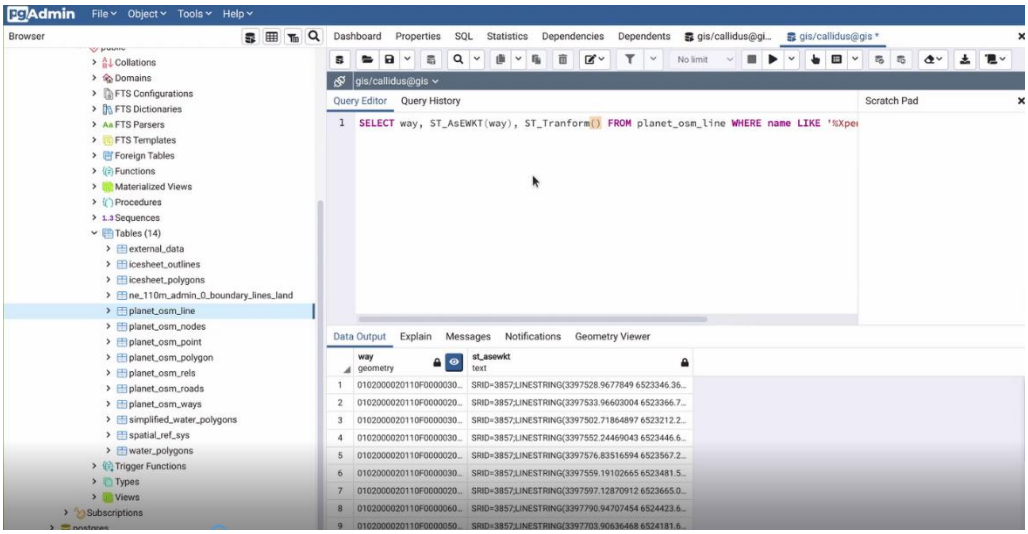


Рис. 2.2 Вікно Query Tool в pgAdmin4.

Приклади деяких функцій, що використовуються в запитах до бази даних будуть описані нижче:

- ST_AsEWKT() – повертає добре відоме текстове представлення геометрії з префіксом SRID.

- ST_Transform() – повертає нову геометрію з координатами, переведеними в іншу систему просторової прив’язки. ST_Transform часто плутають з ST_SetSRID. ST_Transform фактично змінює координати геометрії із одної системи просторової прив’язки в іншу, в той час як ST_SetSRID () просто змінює ідентифікатор SRID геометрії.
- ST_Collect() – збирає геометрії в колекцію геометрій. Результатом буде або Multi*, або GeometryCollection, в залежності від того, чи мають вхідні геометрії однакові або різні типи (рис.2.3).
- ST_Simplify() – повертає «спрощену» версію даної геометрії з використанням алгоритму Дугласа-Пекера. Оскільки спрощення відбувається для кожного об’єкту, то можна також передати GeometryCollection в цю функцію.
- ST_LineMerge() – повертає (набір) LineString (s), сформованих шляхом об’єднання складової лінійної роботи MULTILINESTRING.
- ST_MakeLine() – створює LineString, що містить точки геометрій Point, MultiPoint або LineString.

Рис 2.3. Приклад використання функції ST_Collect().

З виходом pgAdmin 4, починаючи з версії 3.3 постачається також програма Geometry Viewer(рис 2.4), яка буде відображати геометричні (або географічні) дані на OpenStreetMap. За допомогою Query Tool можна написати запит шляхів, та отрмати відображення запиту в Geometry Viewer:

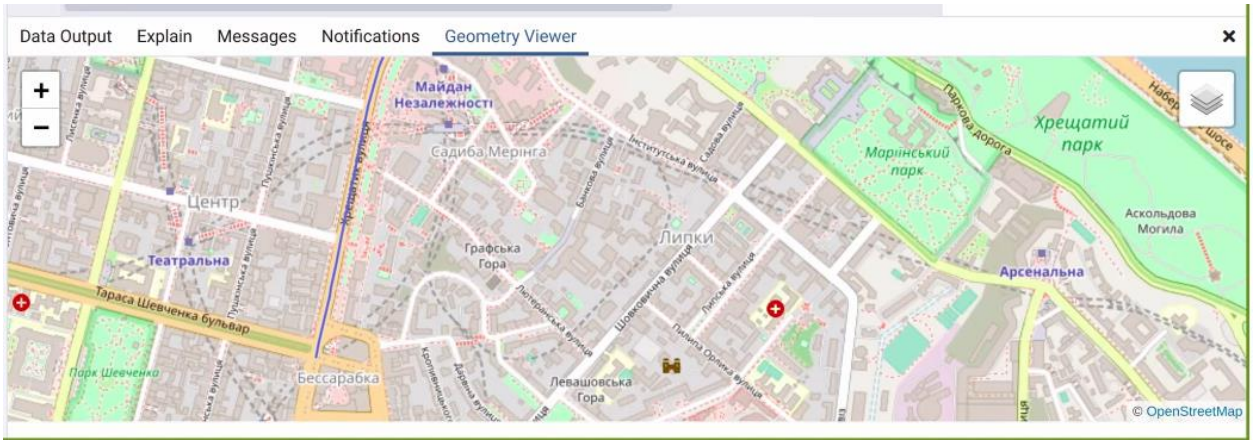


Рис. 2.4 Відображення шляхів на карті в Geometry Viewer.

2.5. Огляд мови програмування Python та середовища розробки PyCharm

Python (рис 2.5) – це проста в освоєнні та потужна мова програмування. Її мовні конструкції, а також об’єктно орієнтований підхід мають за мету допомогти програмістам писати чіткий логічний код для малих та великих проектів. Елегантний синтаксис і динамічна типізація Python разом з її інтерпретованою природою роблять її ідеальною мовою для написання сценаріїв та швидкої розробки програм у багатьох галузях на більшості платформ.

Python розроблена голандським програмістом Гвідо ван Россумом в 1990 році. Отримала таку назву в честь популярного британського серіалу в жанрі комедії 1970-х років «Літаючий цирк Монті Пайтона».

Програми написані на Python, можуть бути як простими скриптами, так і потужними комплексними програмами.

Інтерпретатор Python і велика стандартна бібліотека знаходяться у вільному доступі у вихідній або двійковій формі для всіх основних платформ на веб-сайті Python і можуть вільно розповсюджуватись. Цей же сайт містить

дистрибутиви і вказівки на безкоштовні сторонні модулі, програми та інструменти Python, а також додаткову документацію.

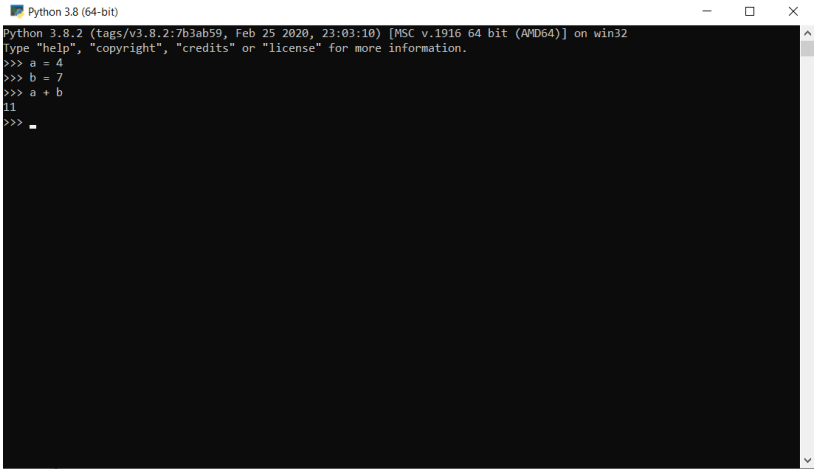


Рис 2.5. Вікно інтерпретатора Python в консолі Windows.

Основними перевагами Python є:

- Велика кількість бібліотек – Python завантажується з великою кількістю бібліотек та містить код для різних цілей, таких як регулярні вирази, створення документації, модульне тестування, веб-браузери, багатопотоковість, бази даних, CGI, електронна пошта, обробка зображень та багато іншого. Таким чином не потрібно писати весь повний код вручну.
- Підвищена продуктивність – простота мови велика кількість бібліотек роблять програмістів більш продуктивними ніж мови на кшталт Java або C++.
- Простота та легкість – при роботі з Java, можливо, прийдеться створити клас для виводу «Hello World». Але в Python потрібно просто застосувати оператор виводу print(). Її також легко вивчити, зрозуміти та написати код.
- Читабельність – оскільки це не така багатослівна мова, читання коду на Python в багатьох аспектах схоже на читання тексту англійською мовою.
- Об’єктно-орієнтований – ця мова програмування підтримує парадигми як процедурного, так і об’єктно-орієнтованого програмування. В той час як функції допомагають нам повторно

використовувати код, класи та об’єкти дозволяють нам моделювати реальний світ. Клас дозволяє інкапсулювати дані і функції в одне ціле.

- Безкоштовний та з відкритим вихідним кодом – Python знаходиться у вільному доступі. Завантажується з великою кількістю бібліотек, які можуть допомогти у вирішенні поставлених задач.
- Інтерпретований – це інтерпретована мова програмування, оскільки оператори виконуються один за другим, відлагодження коду – простіше, ніж в компільованих мовах програмування.

Недоліки Python:

- Повільний – код на Python виконується построково. Але оскільки Python інтерпретується це часто призводить до повільного виконання написаних програм.
- Це мова програмування високого рівня, яка не підходить для написання апаратних програм;
- Для деяких задач неявне виділення пам’яті може бути недоліком;

Переваги Python над іншими мовами:

- Потрібно писати менше коду – майже всі задачі, що виконуються на Python потребують менше коду, якщо та ж задача виконується на інших мовах програмування.
- Доступний – Python повністю безкоштовний, тому окремі користувачі, невеликі компанії або масштабні організації можуть використовувати безкоштовні доступні ресурси для створення програм. Python популярний та широко використовується, тому він забезпечує найкращу підтримку спільноти. Щомісячне опитування Github в 2020 році показало, що Python обігнала Java в найпопулярнішій категорії мов програмування.
- Python для всіх – код може працювати на будь-якій машині, будь-то Linux, Mac або Windows.

- Python - універсальна мова, яка може використовуватися в багатьох сферах. Вона логічна і її легко вивчити, навіть якщо у вас нульовий досвід програмування.

Враховуючи переваги Python, не дивно, що багато великих IT-компаній використовують Python:

- Amazon і Spotify використовують Python для аналізу даних користувачів, продажу інформації та розробки персоналізованих рекомендацій.
- Уолт Дісней використовує цю мову як мову сценаріїв для анімації.
- YouTube та Instagram повністю написані цією мовою.
- Netflix створив свою реферальну службу з нуля в Python.
- Autodesk створює анімацію у своєму редакторі анімації Maya 3D за допомогою Python.
- Студія Pixar також використовує цю мову.

Серед недоліків цієї мови є повільна швидкість роботи та її залежність від системних бібліотек. Однак як малі, так і великі компанії успішно долають ці обмеження та застосовують їх у своїх проектах. Можна написати будь-яку програму, якщо ви знаєте, як програмувати на цій мові. Ви можете перейти від початкового рівня до просунутого. Маючи досвід, ви можете стати експертом і кодувати різні речі, такі як ігри, операційні системи, графічні інтерфейси користувача, програми штучного інтелекту та багато іншого.

PyCharm (рис 2.6) – це спеціальне інтегроване середовище розробки Python (IDE), що представляє широкий спектр необхідних інструментів для розробників Python, тісно інтегрованих для створення зручного середовища для продуктивної розробки Python.

PyCharm – доступний у трьох виданнях:

- Community (безкоштовна та з відкритим кодом): для розумної та інтелектуальної розробки Python, що містить допомогу по коду, рефакторинг, візуальне налогодження та інтеграцію контролю версій.

- Professional (платний) – для професійної розробки на Python, включає в себе в можливості Community-видання, а також віддалені конфігурації, розгортання, підтримку популярних фреймворків, таких як Django або Flask, підтримка баз даних, наукові інструменти та інструменти для обробки великих даних.
- Edu (безкоштовна та з відкритим кодом): для вивчення мови програмування Python та суміжних технологій з інтегрованими освітніми засобами.

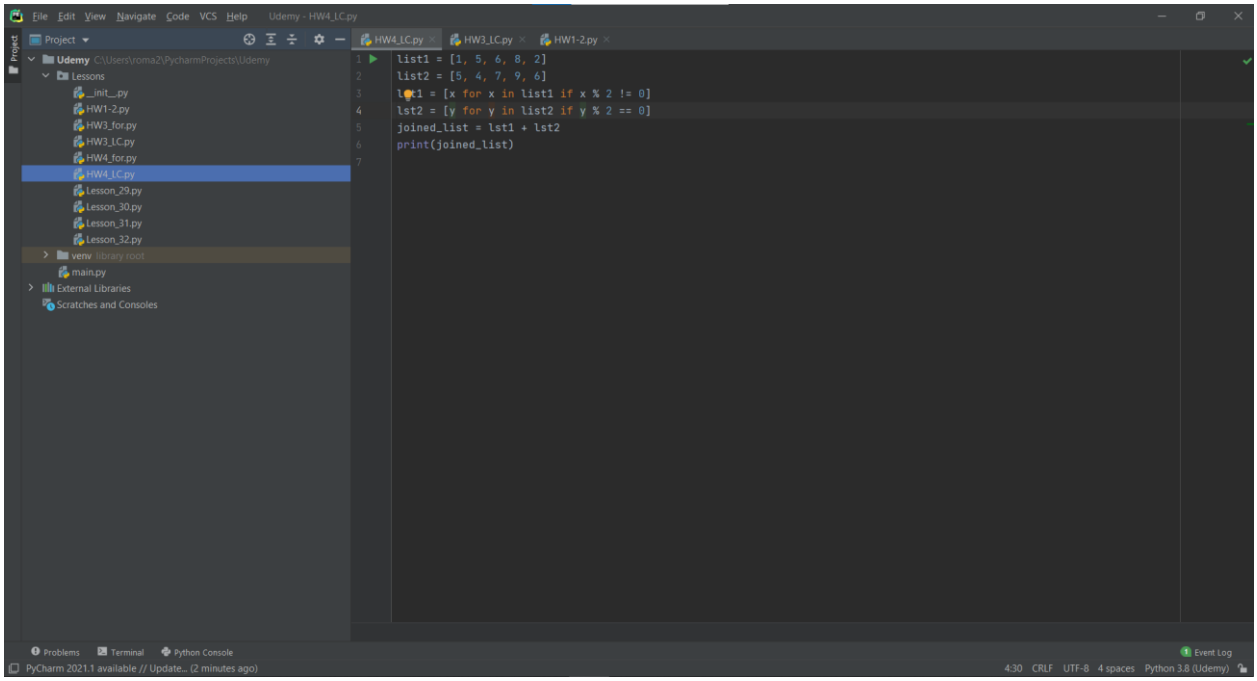


Рис. 2.6. Вікно середовища розробки PyCharm.

2.6. Огляд інструменту для відображення даних з PostGIS

Mapnik (рис 2.7) – це інструментарій з відкритим кодом для візуалізації карт. Поміж іншим, використовується для візуалізації чотирьох основних шарів на веб-сайті OpenStreetMap. Він підтримує різноманітні формати геопросторових даних та забезпечує гнучкі варіанти стилів для проектування багатьох різних видів карт.

Mapnik може виводити зображення карти у різні графічні формати - PNG, JPEG, SVG та PDF. Основне використання Mapnik у OpenStreetMap включає рендерінг багатьох мільйонів плиток карт, які відображаються в інтерфейсі JavaScript Slippy Map (рис. 2.8).

Mapnik дозволяє налаштувати всі картографічні аспекти карти - функції даних, піктограми, шрифти, кольори, візерунки та навіть певні ефекти, такі як псевдо-3D-будівлі та тіні. Це все контролюється визначенням джерел даних та правил стилю, найчастіше мовою XML, специфічною для Mapnik.

Правила стилю Mapnik, що використовуються для шару плитки OSM Standard, є відкритими і можуть бути використані як основа для власних візуалізацій даних OSM. Доступні й інші стилі, наприклад, стиль гуманітарної карти.

Існує ряд зовнішніх інструментів, які можуть допомогти у створенні стилів Mapnik. Вони пропонують мови стилів, які є більш компактними та легшими для читання та запису, ніж вбудована мова Mapnik XML. TileMill і Kosmtik є devolpment середовища з використанням CartoCSS як стиль препроцесора (для перетворення CSS-як CartoCSS в стилі Mapnix XML).

Mapnik може використовувати дані з різних джерел: він може безпосередньо обробляти дані OSM, бази даних PostGIS , файли форм тощо.

PostGIS - це найпоширеніший підхід до надання даних OSM за допомогою Mapnik. OSM може бути завантажений за допомогою такого інструменту, як Osmosis, osm2pgsql або Imposm, і доступ до нього здійснюється за допомогою запитів SQL та функцій ГІС, визначених у стилі Mapnik. Цей підхід може бути використаний для більш розширених візуалізацій і є основним джерелом даних, що використовується стандартним рівнем OpenStreetMap.

Shapefiles - це загальний формат зберігання та обміну географічними даними. На додаток до PostGIS, стандартний стиль OpenStreetMap використовує кілька файлів фігур для відображення карти. Наприклад, наземні маси малюються за допомогою файлів форм.

Mapnik може відображати файли **GeoTIFF** як растрові зображення. Це зазвичай використовується для рельєфних карт. Подібно до GeoTIFF, Mapnik може також відображати **растрові зображення**, які не містять

інформації про геокодування. Обмежувальне поле для цих зображень повинно бути вказано через окремі параметри.

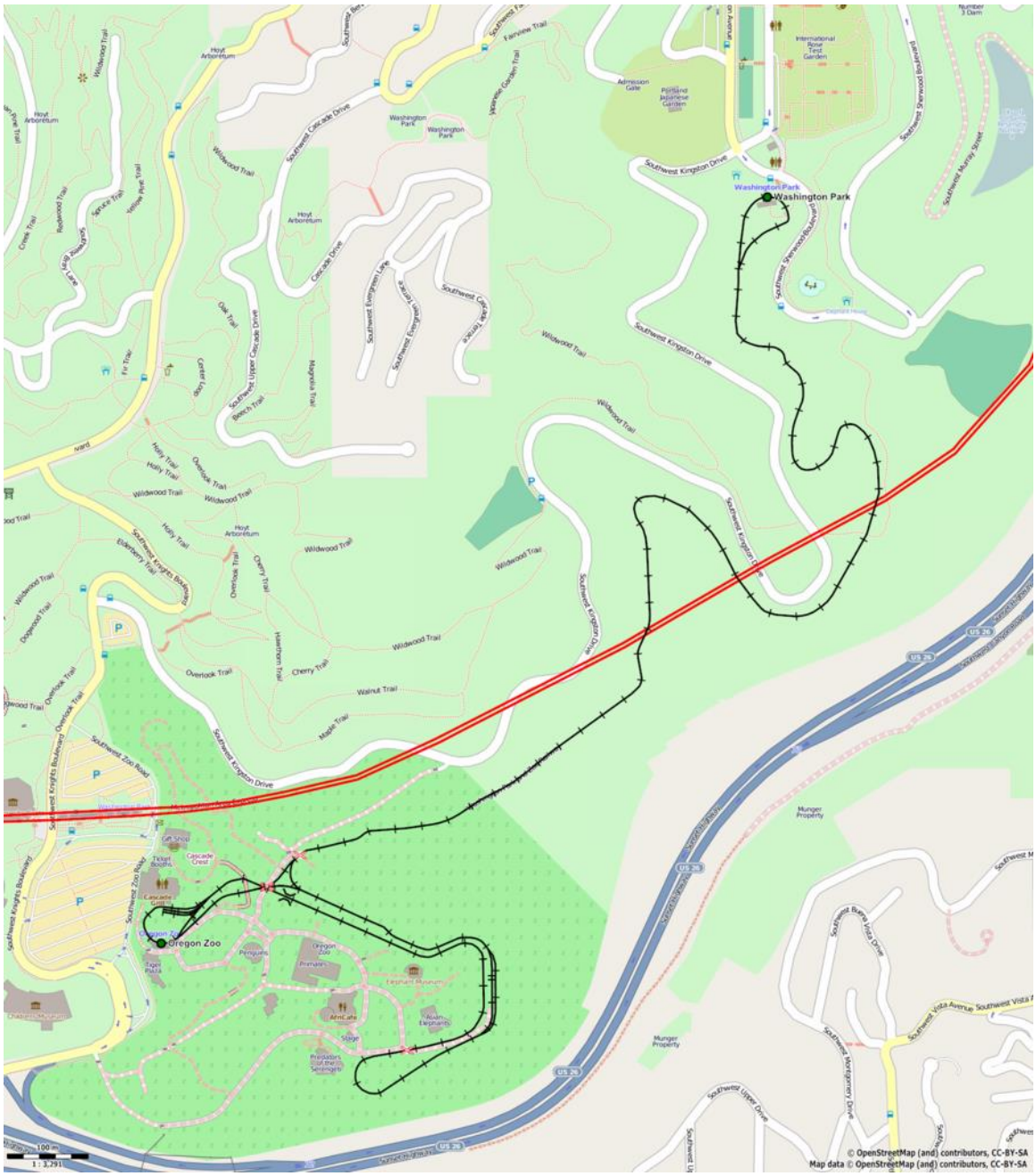


Рис. 2.7. Приклад стандартного стилю OpenStreetMap, що використовує механізм Mapnik.

```
import mapnik
m = mapnik.Map(256,256)
mapnik.load_map(m, "path/to/file.xml")
m.zoom_all()
mapnik.render_to_file(m, "the_image.png")
```

Рис. 2.8. Код на мові програмування Python для відображення плиток карт.

2.7. Огляд технології для рендерингу карт

mod_tile та rednerd – програмне забезпечення для візуалізації і обслуговування плиток растрових карт (рис. 2.9).

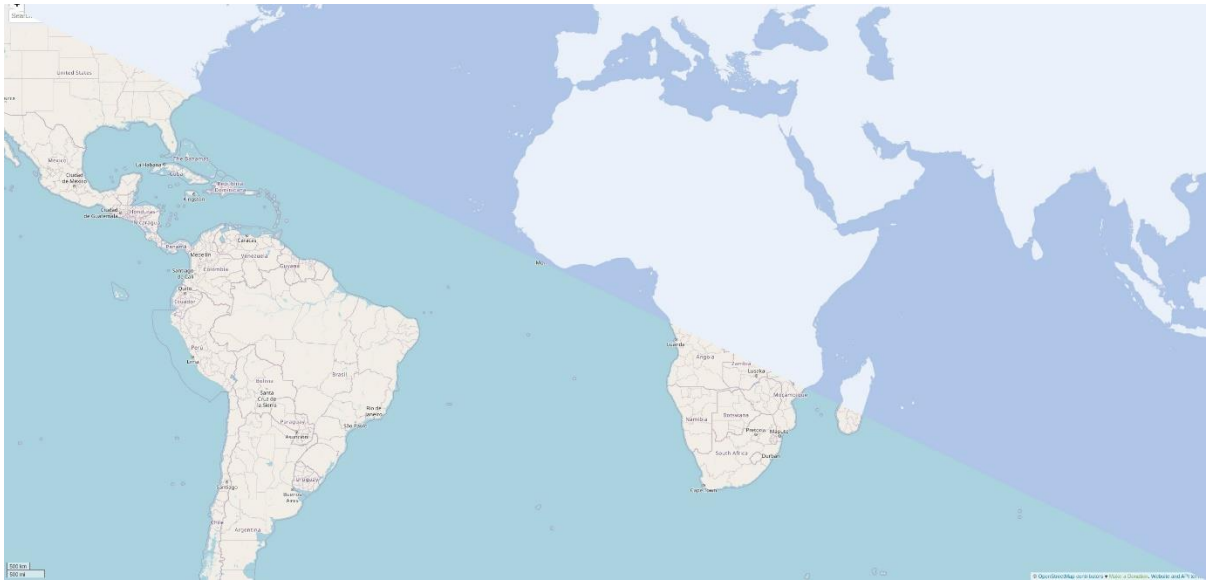


Рис. 2.9. Відображення карт до рендерингу та після нього.

mod_tile – користувацький модуль Apache, який відповідає за обслуговування плиток і вимагає візуалізації плиток, якщо вони ще не доступні в кеші або якщо вони змінились з того часу.

Функції, що надаються mod_tile:

- Висока продуктивність подачі плитки.
- Управління запитами динамічної візуалізації.
- Підтримка декількох різних стилів.
- Термін придатності та рендеринг окремих плиток.
- Динамічне налаштування заголовку cache_expiry.
- Обмеження індивідуальних клієнтів (за IP-адресою).
- Ряд графіків «мунінів» для моніторингу стану та працездатності сервера.

renderd – програма, яка відображає плитки карти за допомогою *mapnik*. Примає на себе запити від mod_tile і надає їх у файлову систему. Вона також

відповідає за чергу запитів, щоб запобігти перенавантаженню сервера, якщо одночасно надходить багато запитів.

Демон (програма) візуалізації реалізує механізм черги з декількома рівнями пріоритету, щоб забезпечити найшвидший перегляд карт, враховуючи доступні ресурси візуалізації. Найвищим пріоритетом є рендеринг плиток, що ще не знаходяться в кеші плиток, два рівні пріоритету для рендерінгу застарілих плиток на льоту та дві черги пакетного візуалізації. Розмір головної фонової черги визначається під час компіляції.

Після рендерингу плитки вона кешується на жорсткому диску для швидокого обслуговування. Оскільки дані карт постійно оновлюються, вдосконалюються та змінюються, потрібен механізм для правильного закінчення терміну дії кешу, щоб забезпечити повторне відображення оновлених плиток.

Існує кілька можливих механізмів для закінчення терміну дії та оновлення плиток (або через mod_tile, або шляхом безпосереднього повторного відображення кешованих плиток), які часто працюється наступним чином:

- Термін діє mod_tiles працює, порівнюючи позначку часу відтворених плиток з датою останнього повного імпорту даних у базу даних.
- Якщо мітку часу планети оновити, усі плитки автоматично позначаються як такі, що потребують повторного відтворення, і вони будуть надіслані до серверної панелі візуалізації під час наступного відвідування.

Крім того, існує ряд допоміжних програм, які можуть безпосередньо спілкуватись з renderd, подаючи плитки на рендеринг. Це часто використовується для рендерингу плиток із низьким масштабуванням на періодичній основі у фоновому режимі, оскільки було б надто затратно оновлювати їх через звичайний термін дії на основі різниці.

Модуль Apache під назвою mod_tile покращує звичайні механізми обслуговування файлів Apache, а саме:

1) Коли термін дії плиток закінчився, він просить демон візуалізації відтворити (або повторно відтворити) плитку.

2) Переформатування шляху до файлу до хеш-макета.

3) Розставляє пріоритети при отриманні запитів залежно від доступних ресурсів на сервері та від того, наскільки вони застарілі.

4) Термін придатності плитки. Він підраховує, коли плитка буде наступною, можливо, рендериться, і додає відповідні заголовки терміну дії кешу HTTP.

Tirex – це набір програм для запуску сервера плиток, був розроблений для серверів плиток OpenStreetMap, але може і використовуватись і для інших карт. Виконує приблизно ті самі завдання, що і добре відомий renderd, але набагато гнучкіший.

Особливості Tirex:

- Розбиває управління чергою та рендеринг плитки на різні модулі.
- Підтримує декілька фонових зображень візуалізації плитки.
- Гнучка черга пріорітетів, яка може обробляти тисячі робочих місць.
- Чудова консоль стану, яка дозволяє переглядати статистику та те, що роблять черги та візуалізатори.
- Рендеринг серверного менеджера, який повторно запускає невдалі рендеринги.
- Використовує багатопроцесорну обробку для використання всіх центральних процесорів серверу.
- Легко встановити, оскільки існує безліч пакетів Debian / Ubuntu для всіх частин системи.

Основна відмінність класичного рендерингу renderd від Tirex полягає в тому, що в Tirex фактичний рендеринг та управління чергою строго відокремлені. Є один демон, який управляє лише запитами, чергами, пріорітетами і т.д., і є інший набір демонів, які не роблять нічого окрім рендеринга плитки. Зв'язок між ними здійснюється за допомогою UDP.

На відміну від renderd, Tirex підтримує лише метатайлову візуалізацію, а не візуалізацію окремих плиток (у renderd це прапор компіляції).

ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ

У цьому розділі проведено загальний аналіз відомостей про СУБД, також розглянуто структуру бази даних PostGIS, що дозволяє працювати з просторовими об’єктами. Проведено аналіз інструменту управління з базами даних pgAdmin4 та розглянуто його можливості. Наведено переваги та недоліки мови програмування Python, на якій буде написана програма для оптимізації бази даних. А також, розглянуто спеціальне середовище розробки (IDE) PyCharm. Було проведено аналіз програми для відображення даних Mapnik та розглянуто стилі Mapnik і джерела даних, які Mapnik може обробляти. Розглянуто технології для рендерингу карт та наведені особливості їх використання.

Загалом структура мого картографічного сервісу виглядає наступним чином:

- PostGIS – розширення для PostgreSQL для зберігання векторної графіки;
- Mapnik – бібліотека для відображення даних, отриманих з PostGIS;
- renderd + mod_tile – програма та модуль apache, що використовується для виклику Mapnik та рендерингу і кешування тайлів і видачі їх за допомогою apache.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ КАРТОГРАФІЧНОГО СЕРВІСУ

3.1 Аналіз структури картографічного сервісу

Картографічний сервіс на базі PostGIS – це сукупність програм та бібліотек, які спільно цілісний сервіс для відображення карт. Існує безліч способ його створення, і майже всі компоненти мають альтернативи, що мають специфічні переваги та недоліки. Мною було обрано стек технологій, який складається з 5 основних компонентів: mod_tile, renderd, mapnik, osm2pgsql та база даних PostgreSQL та розширенням для неї – PostGIS.

Mod_tile – це модуль Apache, який обслуговує кешовані плитки та вирішує, які плитки потребують повторного відтворення – або тому, що вони ще не кешовані, або тому, що вони застарілі. Renderd надає пріоритетну систему черги для різних типів запитів для управління та зменшення навантаження від запитів рендерингу. Mapnik – це бібліотека програмного забезпечення, яка виконує власне рендеринг і використовується renderd.

Для побудови даних компонентів необхідно встановити всі необхідні залежності:

- libboost-all-dev – файли розробки бібліотек Boost C++
- git – система контролю версій
- tar – утиліта для роботи з архівами формату .tar
- unzip – утиліта для розпаковки архівів формату .zip
- wget – утиліта для завантаження файлів через консоль Linux
- bzip2 – утиліта для стиснення даних без втрат, використовуючи алгоритм Барроуза-Уілера.
- build-essential – мета-пакет, що включає в собі компілятора GNU, налогоджувач GNU та інші бібліотеки та інструменти розробки, необхідні для компіляції програмного забезпечення.
- autoconf – утиліта для створення конфігураційних скриптів.

- libtool – представляє загальні сервіси для збору бібліотек.
- libxml2 – файли розробки для бібліотеки XML GNOME.
- libgeos-dev, libgeos++-dev – пакети, що містять заголовки і бібліотеки необхідні для розробки програм з використанням GEOS.
- libpq-dev – файли заголовків та статична бібліотека для компіляції програм на мові C для зв'язування з бібліотекою libpq для зв'язку з серверною частиною бази даних PostgreSQL.
- libbz2-dev – бібліотека стиснення за алгоритмом Барроуза-Уілера.
- libproj-dev – бібліотека картографічної проекції.
- munin-node – пакет, що містить демон для вузлів, що контролюються.
- munin – пакет, що містить збирач, який періодично опитує всі вузли мережі, про які йому відомо, використовує дані для створення даних графіків та HTML-сторінок, придатних для перегляду за допомогою обраного бразуера.
- protbuf-c-compiler – компілятор буферів протоколів C.
- libfreetype6-dev – механізм шрифтів FreeType 2.
- libtiff5-dev – бібліотека, що забезпечує підтримку формату файлу зображень тегів (TIFF), широко використовуваного формату для зберігання даних зображень.
- libicu-dev – бібліотека, що містить файли розробки для міжнародних компонентів Unicode.
- libgdal-dev – бібліотека абстракції геопросторових даних.
- libcairo2-dev – пакет, що містить бібліотеки розробки, файли заголовків, необхідні для програм, які хочуть компілюватись з Cairo.
- libcairomm-1.0-dev – оболонки C++ для Cairo.
- apache2 – веб-сервер, що примає запити від веб-браузерів і видає їм HTTP-відповідь, разом з HTML-сторінкою, зображеннями або іншими даними.
- apache2-dev – заголовки розробки веб-серверу Apache HTTP.

- libagg-dev – графічний інструментарій Anti-Grain Geometry.
- liblua5.2-dev – файли розробки для мови Lua версії 5.2.
- ttf-unifont – пакет, що містить в собі шрифти «Unifont» і «Unifont Sample».
- lua5.1 – мова програмування призначена для розширення програм.
- liblua5.1-0-dev – файли розробки для мови програмування Lua

Список утиліт та ПЗ вище включає в себе веб-сервер Apache та “carto”, який використовується для перетворення таблиць стилів Carto-CSS у щось, що “mapnik” може як відображати у вигляді карти.

CartoCSS – це препроцесор таблиці стилів Mapnik, розроблений MapBox. Таблиці стилів CartoCSS відрізняються від MapCSS кількома пунктами: найважливішим є те, що базове представлення даних не прив’язане до даних OSM або будь-якого іншого джерела даних. Хоча CartoCSS може працювати з будь-якою структурою даних, блок «OSM» таблиці стилів на основі від однієї схеми, як Imposm або стовпової, може бути замінений на інші таблиці стилів на основі однієї і тієї ж схеми.

3.2 Розробка картографічного сервісу

На операційній системі Ubuntu є заздалегідь упаковані версії як PostGIS та і PostgreSQL, тому я просто установлюю їх через менеджер пакетів Ubuntu.

$ *sudo apt install postgresql postgresql-contrib postgis postgresql-12-postgis-3 postgresql-12-postgis-3-scripts*

PostgreSQL – це база даних, в якій я буду зберігати картографічні дані, а PostGIS розширює додаткову графічну підтримку.

Після встановлення необхідного програмного забезпечення створюємо базу даних, за допомогою термінального клієнту *psql* для роботи з PostgreSQL створюємо потрібні нам розширення postgis та hstore;

Створена база даних та розширення postgis і hstore для неї, а також вставлені таблиці геометрії geometry_columns та spatial_ref_sys зображені на рис. 3.1.

```
postgres@romeo-VirtualBox:~$ createdb -E UTF8 -O romeo gis
postgres@romeo-VirtualBox:~$ psql
psql (12.6 (Ubuntu 12.6-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# \c gis
You are now connected to database "gis" as user "postgres".
gis=# CREATE EXTENSION postgis;
CREATE EXTENSION
gis=# CREATE EXTENSION hstore;
CREATE EXTENSION
gis=# ALTER TABLE geometry_columns OWNER TO romeo;
ALTER TABLE
gis=# ALTER TABLE spatial_ref_sys OWNER TO romeo;
ALTER TABLE
gis=#
```

Рис. 3.1 Створена база даних за допомогою командного рядка Linux

Далі встановлюємо *osm2pgsql* – програмне забезпечення для імпорту даних OpenStreetMap у базу даних PostgreSQL / PostGIS. Це важлива частина багатьох ланцюжків рендеринга, що обробляє дані OSM.

Особливості osm2pgsql:

- Перетворення файлів OSM у БД PostgreSQL.
- Перетворення тегів у стовпці, можна налаштувати у файлі стилю.
- Здатний безпосередньо читати .gz, .bz2, .pbf та .o5m.
- Підтримка вибору вихідної проекції.
- Настроювані назви таблиць.
- Підтримка типу поля hstore для зберігання повного набору тегів в одному полі бази даних, якщо потрібно.

Встановлюємо та перевіряємо працездатність Mapnik за допомогою інтерпретатора Python (рис 3.2). Якщо інтерпретатор не видав помилок, значить бібліотека Mapnik була знайдена інтерпретатором.

```
romeo@romeo-VirtualBox: ~
romeo@romeo-VirtualBox:~$ python3
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information
.
>>> import mapnik
>>>
```

Рис. 3.2. Перевірка роботи Mapnika за допомогою інтерпретатора.

Далі встановлюємо mod_tile та компілюємо його вихідний код за допомогою команд:

$ mkdir ~/src

$ cd ~/src

$ git clone -b switch2osm git://github.com/SomeoneElseOSM/mod_tile.git

$ cd mod_tile

$ ./autogen.sh

$ ./configure

$ make

$ sudo make install

$ sudo make install-mod_tile

$ sudo ldconfig

Тепер коли встановлено все необхідне програмне забезпечення, треба завантажити таблицю налаштування стилів. Я використав стиль, який застосовується на «стандартній» карті на веб сайті openstreetmap.org. Його обрано, оскільки він добре задокументований і повинен працювати в будь-якій точці світу(у тому числі в місцях з нелатинськими назвами точок). Зберігаємо деталі таблиці стилів в кореневій папці ~/src та встановлюємо відповідну версію компілятора «carto» за допомогою менеджера пакетів npm (рис. 3.3).

```
romeo@romeo-VirtualBox:~$ sudo npm install -g npm
/usr/local/bin/npm -> /usr/local/lib/node_modules/npm/bin/npm-cli.js
/usr/local/bin/npx -> /usr/local/lib/node_modules/npm/bin/npx-cli.js
+ npm@7.15.0
added 254 packages from 920 contributors in 6.36s
romeo@romeo-VirtualBox:~$ sudo npm install -g carto

changed 64 packages, and audited 65 packages in 4s

1 package is looking for funding
  run `npm fund` for details

4 vulnerabilities (2 low, 1 moderate, 1 high)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
romeo@romeo-VirtualBox:~$ carto -v
1.2.0
```

Рис. 3.3. Установка менеджера пакетів npm
та компілятора таблиці стилів carto

Потім ми перетворюємо проект картографії на те, що Mapnik може зрозуміти:

$ carto project.mml > mapnik.xml

Тепер на сервері є таблиця стилів Mapnik XML (рис.3.4) за адресою */home/romeo/src/openstreetmap-carto/mapnik.xml*

```
<Rule>
  <MaxScaleDenominator>200000</MaxScaleDenominator>
  <MinScaleDenominator>100000</MinScaleDenominator>
  <Filter><![CDATA[([feature] = 'shop_mall') and ([way_pixels] >= 64)]]></Filter>
  <PolygonSymbolizer fill="#dddddd" />
</Rule>
```

Рис 3.4. Код файлу mapnik.xml, що описує
правило стилю для торгового центру

Rule – правило застосування стилю. Воно може бути задано як глобально так і мати логіку певного фільтра.

MaxScaleDenominator – вказує максимальний масштаб карти, в якому повинен бути відтворений символ або об’єкт.

MinScaleDenominator – вказує мінімальний масштаб мапи, в якому повинен відображитись символ або об’єкт.

PolygonSymbolizer – секція, в якій задається символіка об’єкта, в цьому випадку – торгового центру (полігон).

Наступним кроком буде завантаження даних карти України з сайту Geofabrik у форматі .pbf та вливання цих даних в БД. (рис. 3.5.)

```
romeo@romeo-VirtualBox:~$ osm2pgsql -d gis --create --slim  -G --hstore --tag-transform
-script ~/src/openstreetmap-carto/openstreetmap-carto.lua -C 4000 --number-processes 4
-S ~/src/openstreetmap-carto/openstreetmap-carto.style ~/data/europe/ukraine-latest.osm
.pbf
```

Рис. 3.5 Завантаження даних карти України в БД PostgreSQL.

Опис ключів:

- *-d gis* - це база даних, з якою можна працювати;
- *--create* – ключ, за допомогою якого, дані завантажуються у порожню базу;
- *--slim* – ключ, для «тонких» таблиць, які працюють для рендерингу.
- *-G* – визначає, як обробляються мультиполігони.

- *--hstore* – дозволяє використовувати теги, для яких немає явних стовпців бази даних, для рендерингу.
- *--tag-transform-script* – визначає скрипт lua, який використовується для обробки тегів.
- *-C 4000* – об’єм виділеної оперативної пам’яті для osm2pgsql для процесу імпорту.
- *--number-processes 4* – кількість процесорів, які використовуються для імпорту даних.

З версії openstreetmap-carto 5.3.0 потрібно деякі додаткові індекси створювати вручну і хоча більшість даних, що використовуються для створення картографічного сервісу, надходять безпосередньо з файлу даних OpenStreetMap, все таки потрібні деякі *Shapefiles* для меж країни з низьким масштабуванням. Імена, що використовуються для місць на карті України, пишуться не латинськими символами, тому потрібно також встановити шрифти для коректного відображень назв на карті.

Налаштовуємо renderd

Конфігураційним файлом для «renderd» є “/usr/local/etc/renderd.conf”. Змінимо у ньому параметр *num_threads=4* (відповідає кількості ядер на сервері). Та вказуємо шлях до mapnik.xml відповідно до користувача базою даних(рис 3.6).

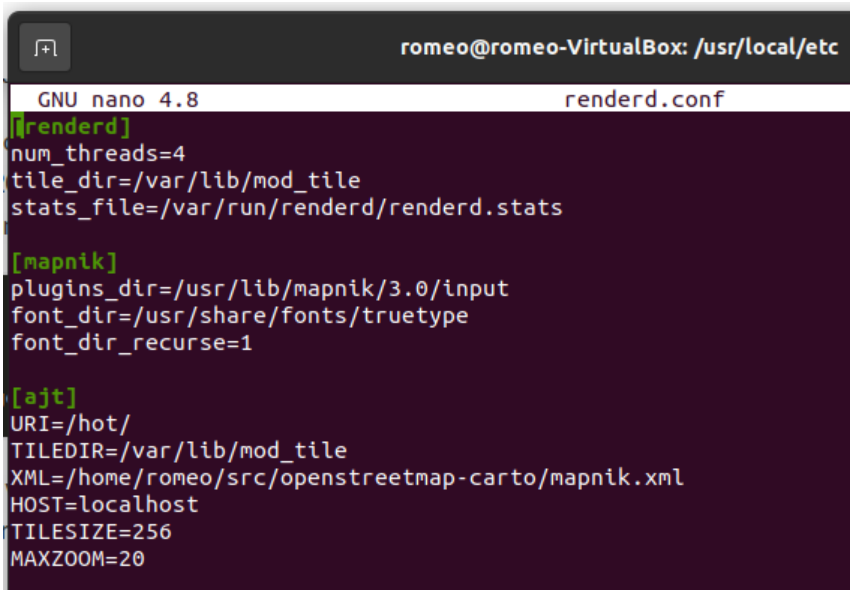


Рис. 3.6 Налаштування конфігураційного файлу renderd.conf

Налаштування Apache

Створюємо директорії /var/lib/mod_tile (міститься інформація про mod_tile) та /var/run/renderd (міститься загальна інформація про стан з моменту останнього запуску, входу в систему і т.д.). Надаємо права доступ до папок для користувача на сервері.

Тепер потрібно сказати Apache про “mod_tile” в каталог */etc/apache2/conf-available/mod_tile.conf* додаємо наступний рядок «LoadModule tile_module /usr/lib/apache2/modules/mod_tile.so» (без лапок).

Зараз треба сказати Apache про “renderd” в каталог */etc/apache2/sites-available/000-default.conf* між рядками “ServerAdmin” та “DocumentRoot” додаємо рядки зі шляхами до файлу конфігурації, ім’ям сокета і таймаутом перед тим як відмовитись від рендерингу плиток та таймаут перед тим, як відмовити у рендерингу плитки, яка в іншому випадку відсутня. (рис 3.7)

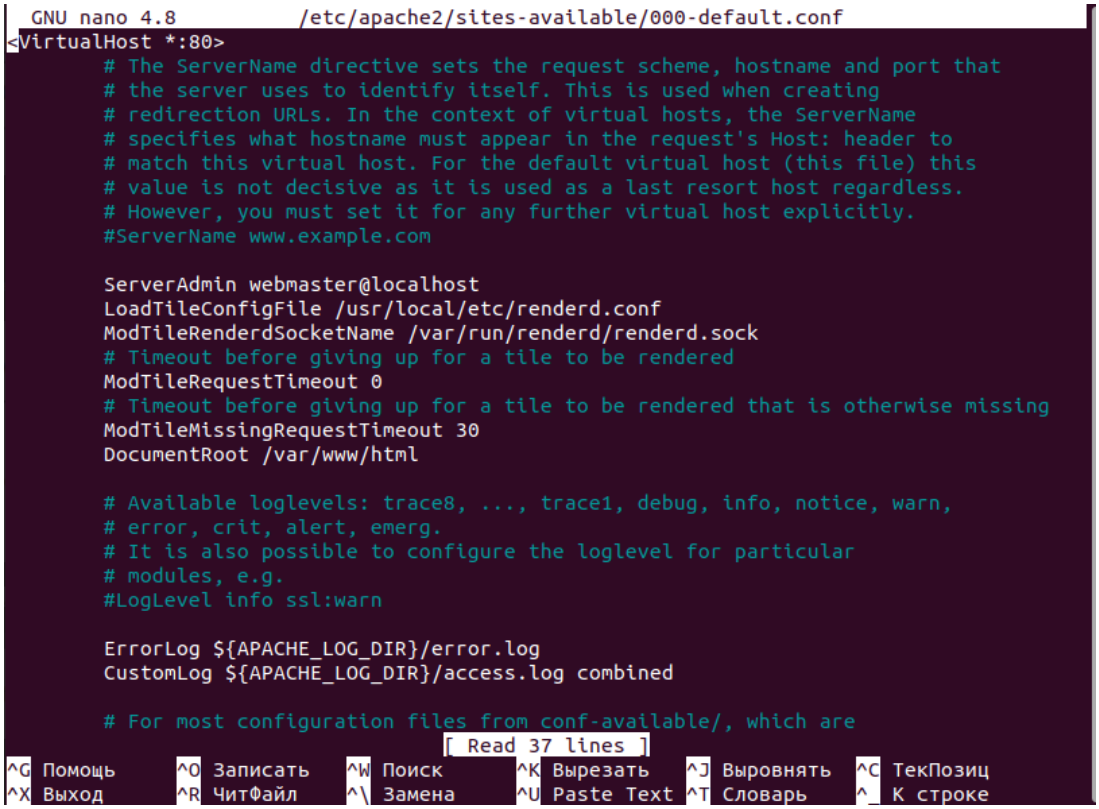


Рис. 3.7. Конфігураційний файл Apache, що стосується рендерингу

Перезавантажуємо веб-сервер Apache (рис 3.8) та перевіряємо правильність налаштування переглядом початкової сторінки за адресою http://10.0.2.15./index.html (10.0.2.5 - адреса локального серверу).

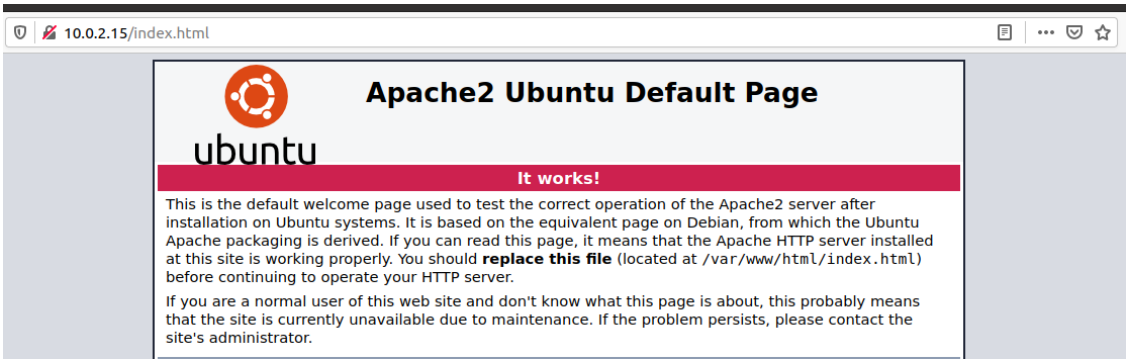


Рис. 3.8. Відображення працездатності та правильного налаштування серверу Apache

Запуск renderd вперше

Далі ми запускаємо renderd, щоб спробувати відтворити деякі плитки. Запускаємо через консоль рендеринг плиток (рис. 3.9) за допомогою команди:

$ renderd -f -c /usr/local/etc/renderd.conf

В браузері переходимо за адресою: http://10.0.2.15/hot/0/0/0.png

У веб-браузері відображається карта світу, а в командному рядку записи “DEBUG: START TILE” та “DEBUG: DONE TILE”, що свідчить про початок рендерингу плитки та його завершення.

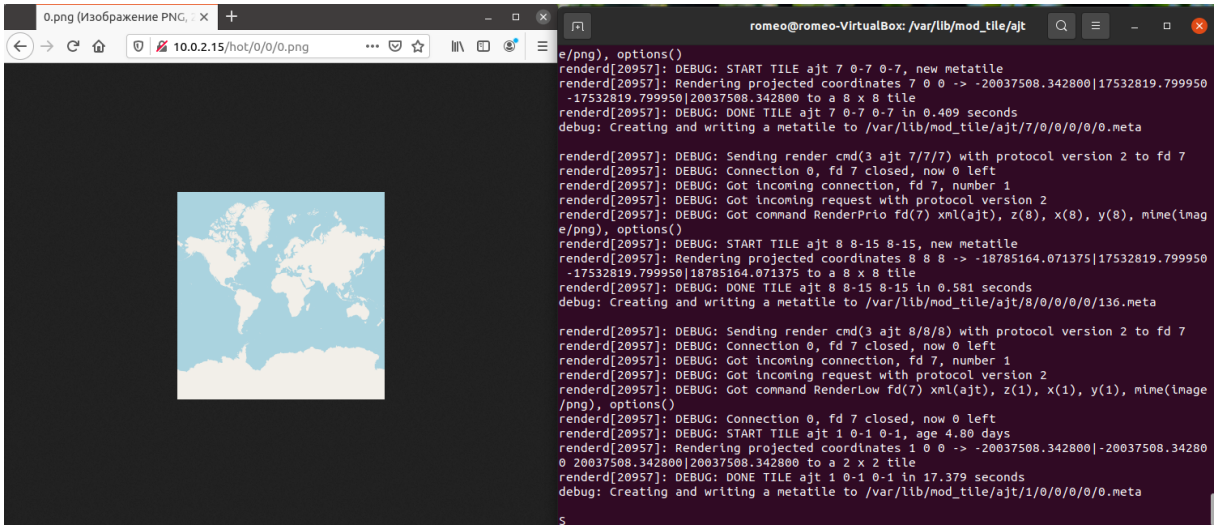


Рис. 3.9 Відображення карти світу після рендерингу плитки та відповідних повідомлень в консолі про початок та заверження рендерингу.

Запуск renderd у фоновому режимі

Далі я налаштовую “renderd” для роботи у фоновому режимі (рис.3.10). Спочатку потрібно відредагувати файл: *«~/src/mod_tile/debian/renderd.init»*

так, щоб «RUNASUSER» було встановлено для поточного адміністратора серверу.

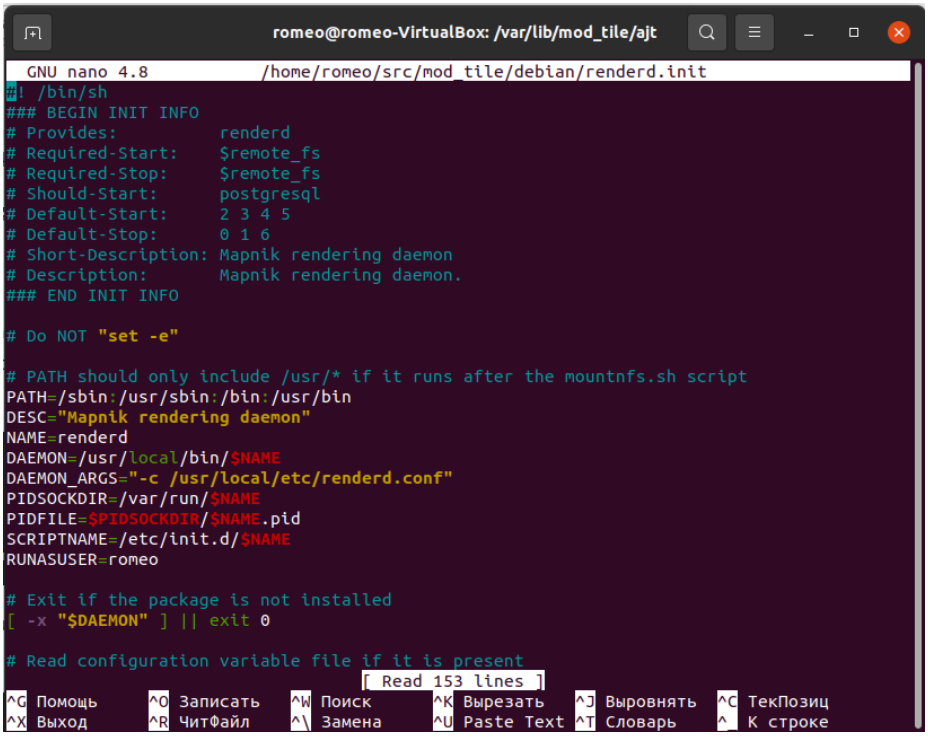


Рис. 3.10. Вікно конфігурації файлу ініціалізації renderd.init

Файл *“renderd.service”* є службовим файлом *“systemd”*. Версія, яка використовується в моєму картографічному сервісі, просто викликає команди init старого стилю. Перевіряємо працездатність та вводимо команду для запуску автоматичного щоразу після входу в систему:

$ sudo /etc/init.d/renderd start

$ sudo systemctl enable renderd

3.3 Розробка скрипта для об’єднання геоструктур даних в базі PostGIS

Скрипт для зменшення об’єму даних в БД PostgreSQL написаний на мові програмування Python, у середовищі розробки PyCharm 2021.1.1 Professional з використанням бібліотеки для роботою з базою даних PostgreSQL psycopg2.

Psycopg – найпопулярніший адаптер бази даних PostgreSQL для мови програмування Python. Основними його особливостями є повна реалізація специфікації Python DB API 2.0 та безпека потоків (декілька потоків можуть використовувати одне і те ж з’єднання). Psycopg був розроблений для

багатопотокових додатків, які створюють і руйнують велику кількість курсорів і роблять достатню кількість одночасних INSERT`s або UPDATE`s.

Psycopg 2 в основному реалізована на мові C як обгортка libpq, звідси випливає ефективність обробки бази даних та її безпека. Він має курсори на стороні клієнта та сервера, асинхронний зв'язок та повідомлення, підтримує "COPY TO / COPY FROM". Більшість типів даних Python підтримуються нестандартно і адаптовані у відповідності до типів даних PostgreSQL. Адаптація може бути розширена та налаштова завдяки гнучкій системі адаптації об'єктів.

Перш ніж взаємодіяти з базою даних PostgreSQL, з нею необхідно зв'язатись (рис. 3.11). Визначаємо функцію create_connection() для підключення до бази даних PostgreSQL.

```
def create_connection(db_name, db_user, db_password, db_host, db_port):
    connection = None
    try:
        connection = psycopg2.connect(
            database=db_name,
            user=db_user,
            password=db_password,
            host=db_host,
            port=db_port,
        )
        print("Connection to PostgreSQL DataBase successful!")
    except OperationalError as e:
        print(f"The error '{e}' occurred")
    return connection
```

Рис. 3.11. Функція підключення до бази даних PostgreSQL.

Підключення виконується через інтерфейс psycopg2.connect(). Далі я використовую написану мною функцію:

*connection = create_connection("gis", "postgres", "***", "127.0.0.1", "5432")*

Тут gis – назва бази даних, postgres – користувач, *** - пароль користувача, 127.0.0.1 – IP-адреса, 5432 – порт хоста серверу.

Використання бібліотеки *psycopg2* в функції *execute_query()* також має на меті роботу з *cursor*. Функція описана нижче на рис. 3.12 може бути

використана для організації таблиць, вставки, змін та видалення записів в базі даних PostgreSQL.

```
def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        print("QUERY EXECUTED SUCCESSFULLY!")
    except Error as e:
        print(f"The error '{e}' occurred")
```

Рис. 3.12 Код функції для зміни записів в БД

Для вибору записів з таблиць PostgreSQL я використовую cursor.execute(), потім метод fetchall() для вибору записів із таблиці. На рис. 3.13. відображається функція для виконання запиту на читання записів.

```
def execute_read_query(connection, query):
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()
        return result
    except OperationalError as e:
        print(f"The error '{e}' occurred")

select_names_primary = "SELECT name" \
                       " FROM planet_osm_line" \
                       " WHERE highway = 'primary' and name LIKE '%вулиця%'" \
                       "GROUP BY name" \
                       "HAVING count (*) > 1"

names_primary = execute_read_query(connection, select_names_primary)
print("SELECT NAMES FOR PRIMARY HIGHWAY SUCCESSFUL!")
```

Рис. 3.13. Код для виконання запиту на читання записів колонки name з таблиці planet_osm_line.

Щоб отримати дані з таблиці, потрібно виконати запит. Для цього призначений SQL-оператор SELECT. Він складається з декількох частин: вибірки (в якій перечисляються стовпці, які повинні бути отримані), списка таблиць (в ньому перечисляються таблиці, з яких будуть отримані дані) та необов'язкові умови (визначають обменження, наприклад, такі як WHERE, LIKE, GROUP BY).

У частині WHERE вказується логічний вираз (перевірка істинності), який слугує фільтром рядків: в кінцевому результаті опиняються тільки ті рядки, для яких цей вираз істина. У виразі можуть бути присутні звичайні логічні оператори (AND, OR, NOT).

LIKE повертає true, якщо рядок відповідає заданому шаблону, а знак відсотку (%) підміняє будь-яку послідовність символів.

Рядки запитів таблиці, що пройшли фільтр WHERE, можна згрупувати за допомогою GROUP BY, а в результаті можна залишити тільки потрібні групи рядків, використовуючи HAVING.

Для вставки записів в таблиці баз даних передається SQL-запит із заповнювачами і списком записів методу *execute()*. Кожен запис в списку повинен являти собою кортеж, значення якого відповідають значенням стовпця в таблиці БД. Вставка користувацьких записів в таблицю planet_osm_line зображено на рис. 3.14.

```
names_primary = execute_read_query(connection, select_names_primary)
print("SELECT NAMES FOR PRIMARY HIGHWAY SUCCESSFUL!")

for i in names_primary:
    select_ways_primary = "SELECT ST_LineMerge(ST_Collect(way))" \
                          " FROM planet_osm_line" \
                          f" WHERE name LIKE '{i[0]}' AND highway = 'primary'"
    ways_primary = execute_read_query(connection, select_ways_primary)
    cursor = connection.cursor()
    cursor.execute("INSERT INTO planet_osm_line(name, highway, way) VALUES (%s, %s, %s)",
                   (names_primary, 'primary', ways_primary[0]))
    connection.commit()
print("NEW DATA INSERT FOR PRIMARY HIGHWAY SUCCESSFUL!")
```

Рис. 3.14 Код вставки даних в таблицю *planet_osm_line*.

Список імен вулиць вибраний запитом зображеним на рис. 3.13 передається в цикл, де використовується для вставки даних в подальшому. Запит *select_ways_primary* збирає шляхи вулиць з однаковими назвами в одну геометрію Multilinestring за допомогою методу ST_Collect, а функція ST_LineMerge об’єднує дані всередині Multilinestring та повертає їх у звичному для таблиці planet_osm_line форматі LineString.

Потім методом *cursor.execute()* вставляємо нові дані в поля name, highway, way таблиці planet_osm_line взяті з попередніх запитів, поле highway

= ‘primary’ залишається незмінним. Значення розділені трьома елементами заповнювачами (%s), що відповідає трьом користувацьким записам.

Видалення старих записів(рис. 3.15) код здійснюється за допомогою SQL-оператора DELETE та методу *execute_query()*.

```
delete_old_record_primary = "DELETE FROM planet_osm_line" \
                            " WHERE name LIKE '%вулиця%'" \
                            " AND highway = 'primary' AND z_order > 320"
execute_query(connection, delete_old_record_primary)
print("OLD DATA DELETE FOR PRIMARY HIGHWAY SUCCESSFUL!")
```

Рис. 3.15. Код видалення старих записів з БД

Аналогічним методом і спрощуються записи в базі даних з highway – secondary, tertiary та residential.

ВИСНОВОК ДО ТРЕТЬОГО РОЗДІЛУ

У даному розділі було проаналізовано структуру картографічного сервісу, який включає в себе 5 основних компонентів та безліч залежностей. Було створено картографічний сервіс на базі PostgreSQL з розширенням для геометричних даних PostGIS. Встановлено та налаштовано Mapnik, mod_tile, компілятор «carto», osm2pgsql, за допомогою якого імпортовано дані карти України в БД,. Створено таблицю стилів Mapnik XML. Налаштовано веб-сервер Apache та запущено скрипт renderd для рендерингу карт у фоновому режимі. Також було розроблено скрипт для об’єднання геоструктур даних в базі PostGIS. Скрипт написаний на мові програмування Python у середовищі розробки PyCharm з використанням можливостей бібліотеки psycopg2.

РОЗДІЛ 4. ТЕСТУВАННЯ КАРТОГРАФІЧНОГО СЕРВІСУ ТА РОЗРОБЛЕНОГО СКРИПТА.

4.1 Тестування картографічного сервісу.

Для того, щоб побачити карту використаємо html-файл “sample_leaflet.html”, що знаходиться за шляхом ~/src/mod_tile/extra. На рисунку 4.1. зображена карта України з файлу “sample_leaflet.html” у 6 зумі.

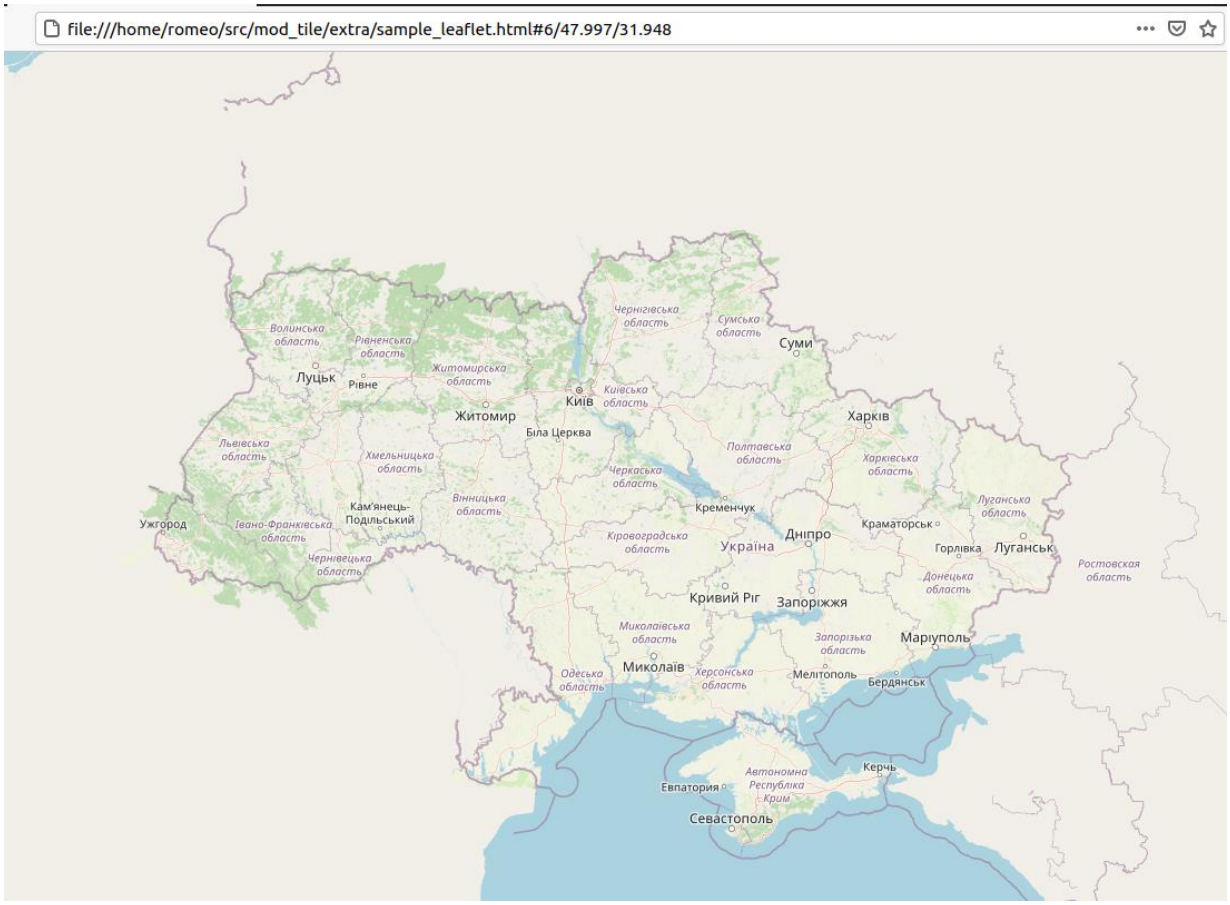


Рис 4.1. Карта України, що відображена у файлі “sample_leaflet.html”

Для того щоб переконатись, що дані дійсно рендеряться відобразимо логи в консоль: *$ tail -f /var/log/syslog | grep " TILE ".*

На рисунку 4.2 можна побачити старт рендерингу плитки і її завершення кожного разу коли відбувається запит тайлів. Якщо збільшувати зум, то можна побачити все більше і більше запитів. Рендеринг плиток після 11-12 зуму займає багато часу, щоб відтворити їх вперше, але якщо вони будуть потрібні наступного разу знову, то вони нададуться майже миттєво.

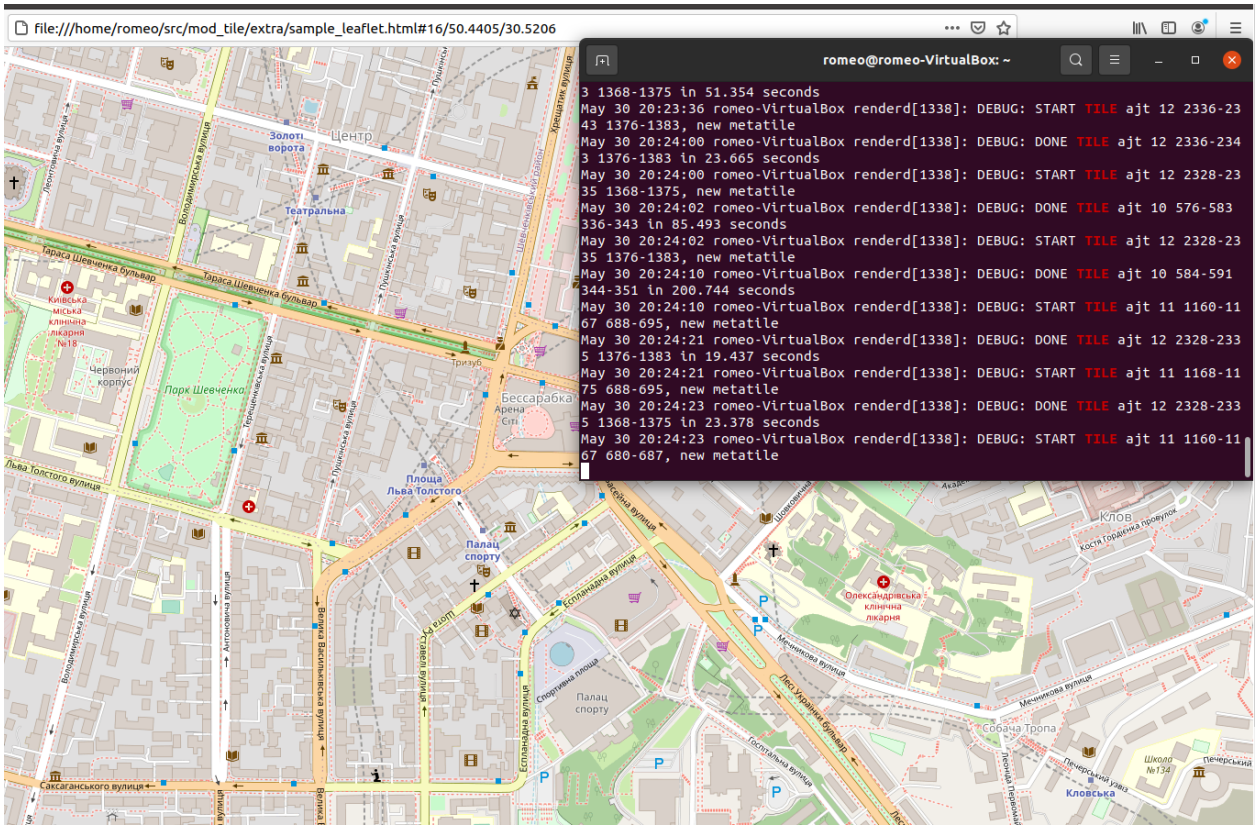


Рис. 4.2 Карта м. Київ після рендерингу та лог рендерингу талйів в консолі

4.2 Тестування розробленого скрипта для змешення об’єму даних в базі PostGIS.

Розмір стандартної таблиці *planet_osm_line* бази даних PostGIS в якій не проводили жодні зміни становить 1384 Мбайт. На рисунку 4.3. зображено дані, що відображають у програмі адміністрування бази даних pgAdmin 4.

Table name	Size
external_data	32 kB
icesheet_outlines	85 MB
icesheet_polygons	73 MB
ne_110m_admin_0_boundary_lines_l...	88 kB
planet_osm_line	1384 MB
planet_osm_nodes	5233 MB
planet_osm_point	317 MB
planet_osm_polygon	3491 MB
planet_osm_rels	207 MB
planet_osm_roads	141 MB
planet_osm_ways	6340 MB
simplified_water_polygons	33 MB
spatial_ref_sys	7248 kB
water_polygons	1043 MB

Рис. 4.3. Розміри таблиць в базі даних PostgreSQL

Для запуску скрипта потрібна мова програмування Python, яка вбудована майже в кожну UNIX-систему. А також додатково потрібно буде завантажити менеджер пакетів Python *pip* та бібліотеку *psycopg2*.

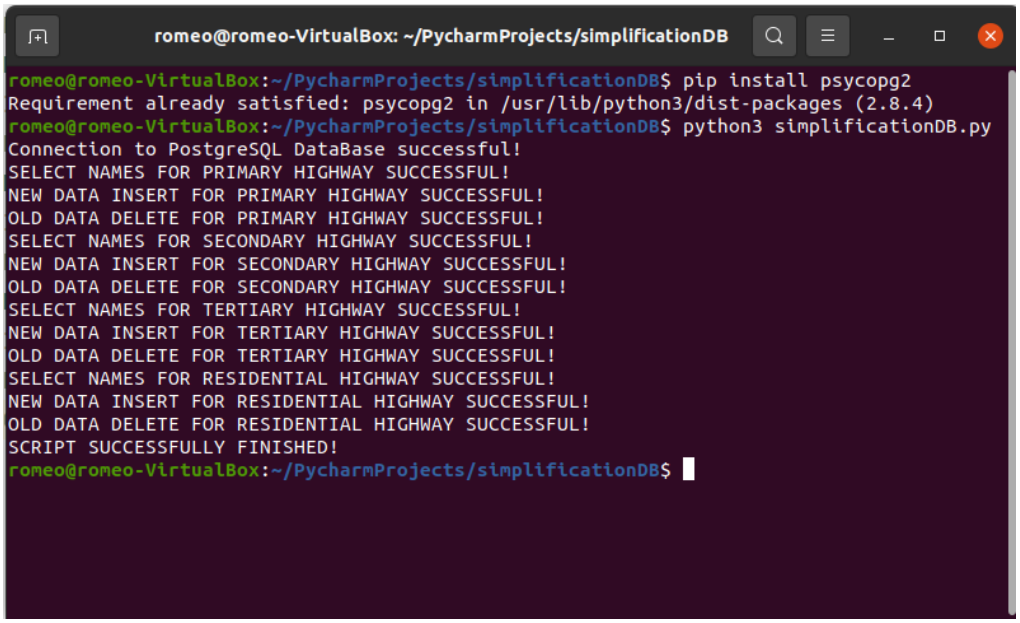


Рис. 4.4. Запуск скрипта для спрощення геоструктур даних в БД

Після установки необхідних залежностей для роботи скрипта, можна його запустити з папки проекту і побачити, що все працює і скрипт успішно завершив свою роботу (рис. 4.4). Тепер перевіримо розмір бази даних після об’єднання (рис. 4.5) та виберемо запис про вулицю Хрещатик, щоб перевірити що дані справді об’єднались (рис.4.6).

Table name	Size
external_data	32 kB
icesheet_outlines	85 MB
icesheet_polygons	73 MB
ne_110m_admin_0_boundary_lines_l...	88 kB
planet_osm_line	1309 MB
planet_osm_nodes	5233 MB
planet_osm_point	317 MB
planet_osm_polygon	3491 MB
planet_osm_rels	207 MB
planet_osm_roads	141 MB
planet_osm_ways	6340 MB
simplified_water_polygons	33 MB
spatial_ref_sys	7248 kB
water_polygons	1043 MB

Рис. 4.5 Розмір таблиць в базі даних після спрощення.

З рисинку 4.5 видно, що розмір таблиці *planet_osm_line* змінився і став 1309 Мбайт, після об’єднання записів, вулиці яких мають ідентичні назви та однакове поле *highway*, розмір таблиці *planet_osm_line* зменшився на 5,4%.

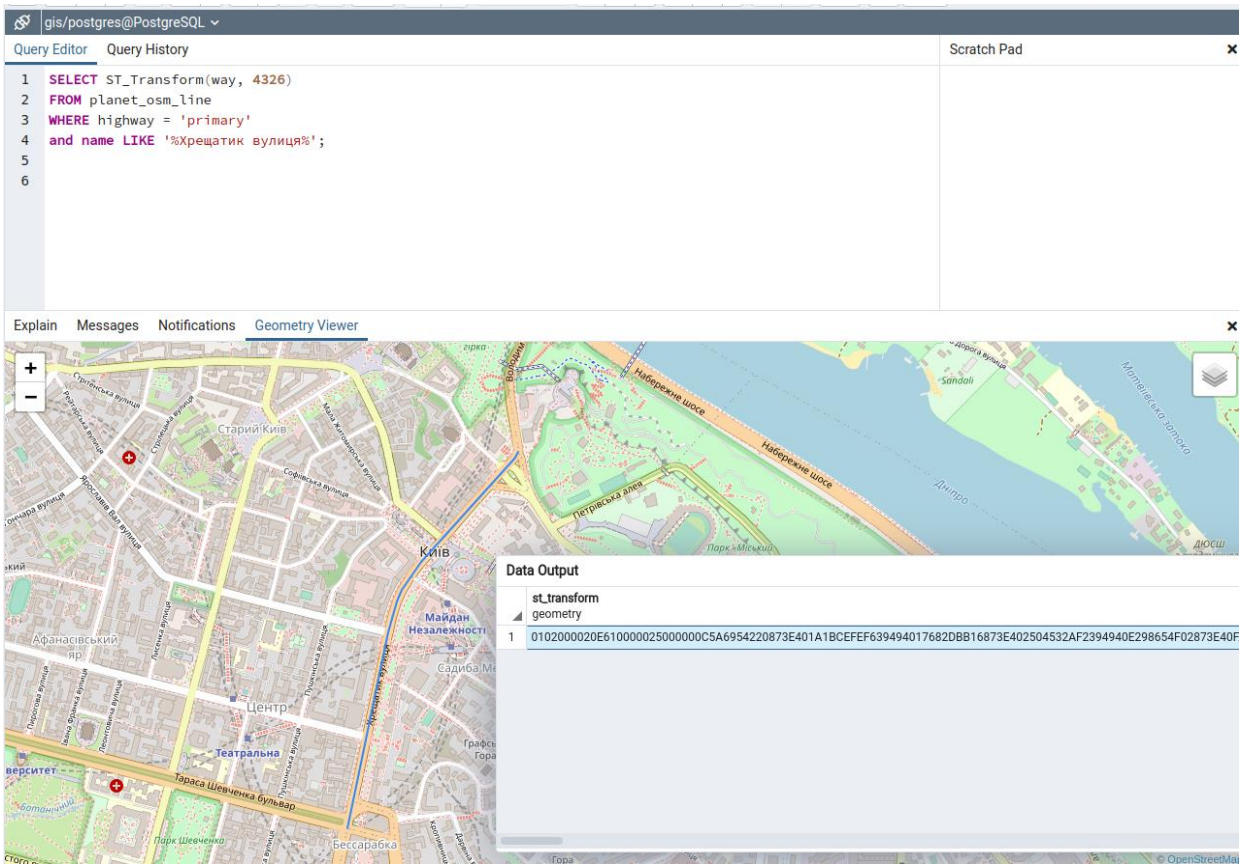


Рис. 4.6 Вибір запису з БД про вул. Хрещатик
та перегляд її структури в Geomerty Viewer

На карті м. Києва видно суцільну вулицю Хрещатик та один єдиний про неї в базі даних PostgreSQL. Це свідчить про те, що скрипт працює правильно і поставлена задача виконана.

ВИСНОВОК ДО ЧЕТВЕРТОГО РОЗДІЛУ

В даному розділі представлено використання розробленого в рамках бакалаврської роботи картографічного сервісу на базі PostGIS. Було запущено скрипт для спрощення геоструктур даних.

В якості результату продемонстровано карту України та м. Київ і різницю в розмірі таблиці planet_osm_line в базі PostGIS. В базі зберігаються тільки унікальні записи про вулиці. Об’єм даних в таблиці зменшився на 5,4%, що демонструє працездатність програми.

ВИСНОВКИ

Метою даної бакалаврської дипломної роботи є вдосконолення бази даних картографічного сервісу, що розроблений на базі PostGIS. Популярність інтернету призвела до появи великої кількості різноманітних картографічних сервісів, які потрібно весь час підтримувати актуальною інформацією, щоб користувачі були задоволені і користувались сервісами і надалі.

В першому розділі проаналізовано існуючі картографічні сервіси. Під час аналізу було наведено їх можливості, переваги та недоліки. Виконано порівняльний аналіз картографічних сервісів.

В другому розділі йде мова про структуру картографічного сервісу, технології для його реалізації, мову програмування та бібліотеки для розробки програми для змешення об’єму БД. Було обрано стек технологій: PostGIS, Mapnik, mod_tile та renderd для реалізації картографічного сервісу. Для розробки скрипта використовується середовище розробки PyCharm, мова програмування Python та бібліотека psycopg2.

Третій розділ містить детальний опис реалізації картографічного сервісу та скрипта для спрощення геоструктур даних, щоб була відповідність поставленим задачам.

В чертвертому розділі продемонстровано зовнішній вигляд картографічного сервісу в браузері та роботу скрипта. Показано, що скрипт вдосконалив базу даних.

Отже, мета диплопного проекту досягнена. Розроблений в даній бакалаврській роботі картографічний сервіс та скрипт відповідає всім поставленим задачам.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] Chiriac V., Ciclicci V. GEODESY, CARTOGRAPHY, AND AERIAL PHOTOGRAPHY. 2020. 91,2020, № 91.

[2] Що таке картографічний сервіс? [Електронний ресурс] – Режим доступу: https://enterprise.arcgis.com/ru/server/latest/publish-services/windows/what-is-a-map-service.htm

[3] Service mapping. [Електронний ресурс] – Режим доступу: https://services.blog.gov.uk/2020/09/01/service-mapping-a-step-by-step-guide/

[4] Основні причини використання картографічних сервісів. [Електронний ресурс] – Режим доступу: https://enterprise.arcgis.com/ru/server/latest/publish-services/windows/common-reasons-for-using-map-services.htm

[5] Аналіз технологій web-картографування для представлення земельно-кадастрових даних в Інтернеті [Електронний ресурс]. – Режим доступу : http://www.nbuv.gov.ua/portal/natural/Nvngu/2011_2/gavr.pdf

[6] Сучасна веб-картографія. [Електронний ресурс] – Режим доступу: https://niss.gov.ua/doslidzhennya/nacionalna-bezpeka/suchasna-veb-kartografiya-ta-ii-vikoristannya-u-poperedzhenni-y

[7] Карти Google – Вікіпедія. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Карти_Google

[8] Карти Bing. [Електронний ресурс] – Режим доступу: https://ru.joecomp.com/gt-guide-bing-maps

[9] Switch2OSM. [Електронний ресурс] – Режим доступу: https://switch2osm.org/

[10] Leaflet – an open-source JavaScript library for interactive maps. [Електронний ресурс] – Режим доступу: https://leafletjs.com/

[11] OpenLayers. [Електронний ресурс] – Режим доступу: https://openlayers.org/

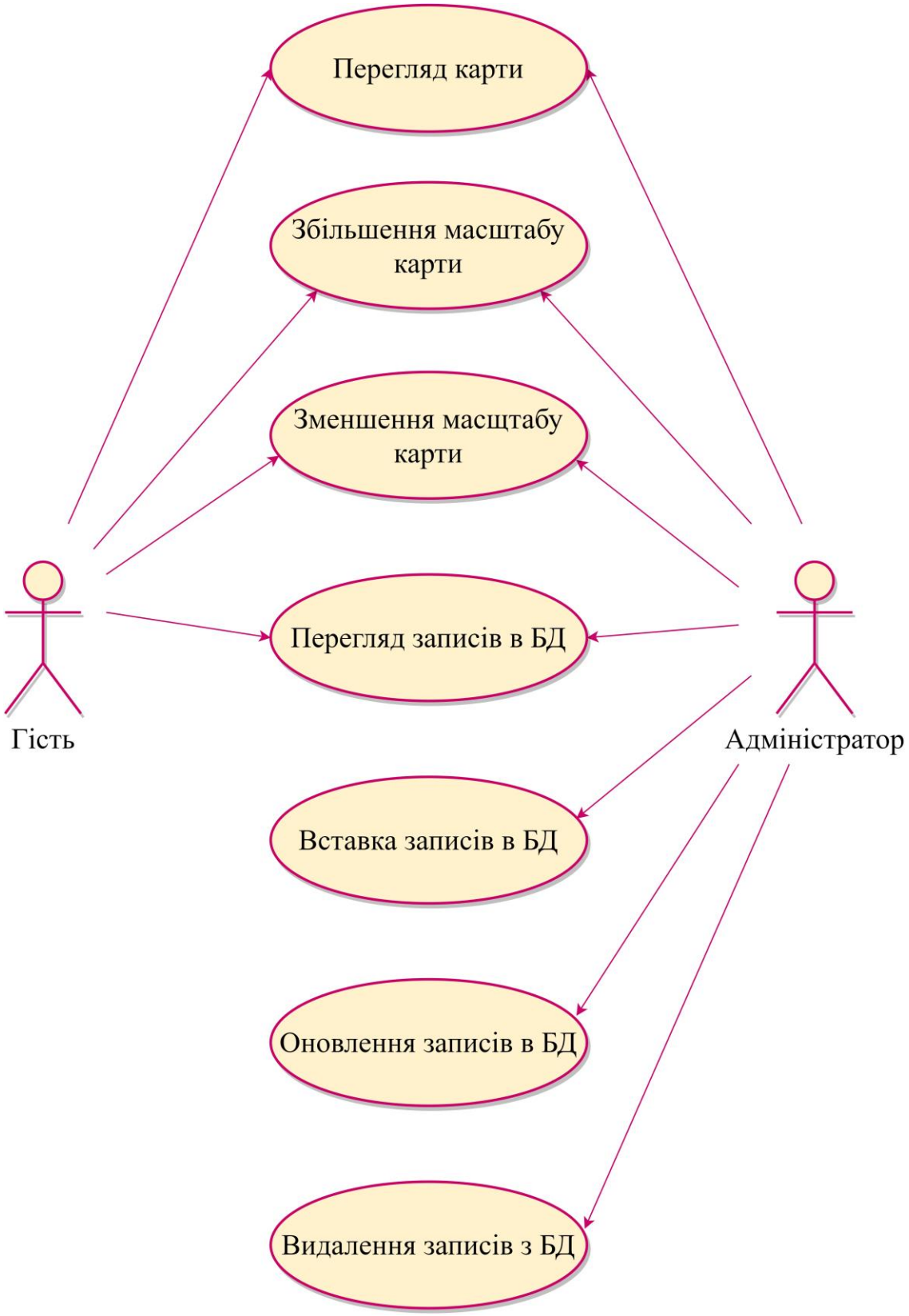
Додаток А

ФУНКЦІОНАЛЬНА СХЕМА

до дипломного проєкту

на тему: «Картографічний сервіс на базі PostGIS»

Київ – 2021 року



					ІАЛЦ.467100.004 Д1		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		*Корсун Р.В.*			*Картографічний сервіс на базі PostGIS* *Функціональна схема*	Літ.	Арк. 1 / Аркушів 1
Перевірив		*Іваніщев Б.В.*					
Реценз.						*«КПІ імені Ігоря.Сікорського» ФІОТ, гр. ІО-71*	
Н. Контр.		*Сімоненко В.П.*					
Затв.		*Іваніщев Б.В.*					

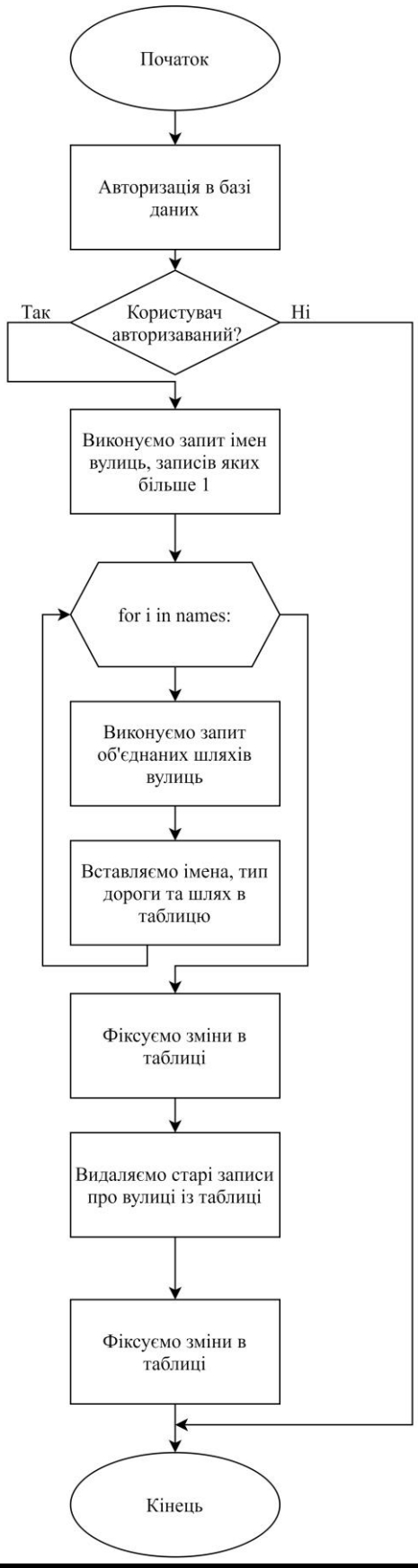
Додаток Б

ПРИНЦИПОВА СХЕМА

до дипломного проєкту

на тему: «Картографічний сервіс на базі PostGIS»

Київ – 2021 року



					ІАЛЦ.467100.005 Д2		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		*Корсун Р.В.*			*Картографічний сервіс на базі PostGIS* *Принципова схема*	*Літ.*	*Арк.* 1 / *Аркушів* 1
Перевірив		*Іваніщев Б.В.*					
Реценз.						*«КПІ імені Ігоря.Сікорського» ФІОТ, гр. ІО-71*	
Н. Контр.		*Сімоненко В.П.*					
Затв.		*Іваніщев Б.В.*					

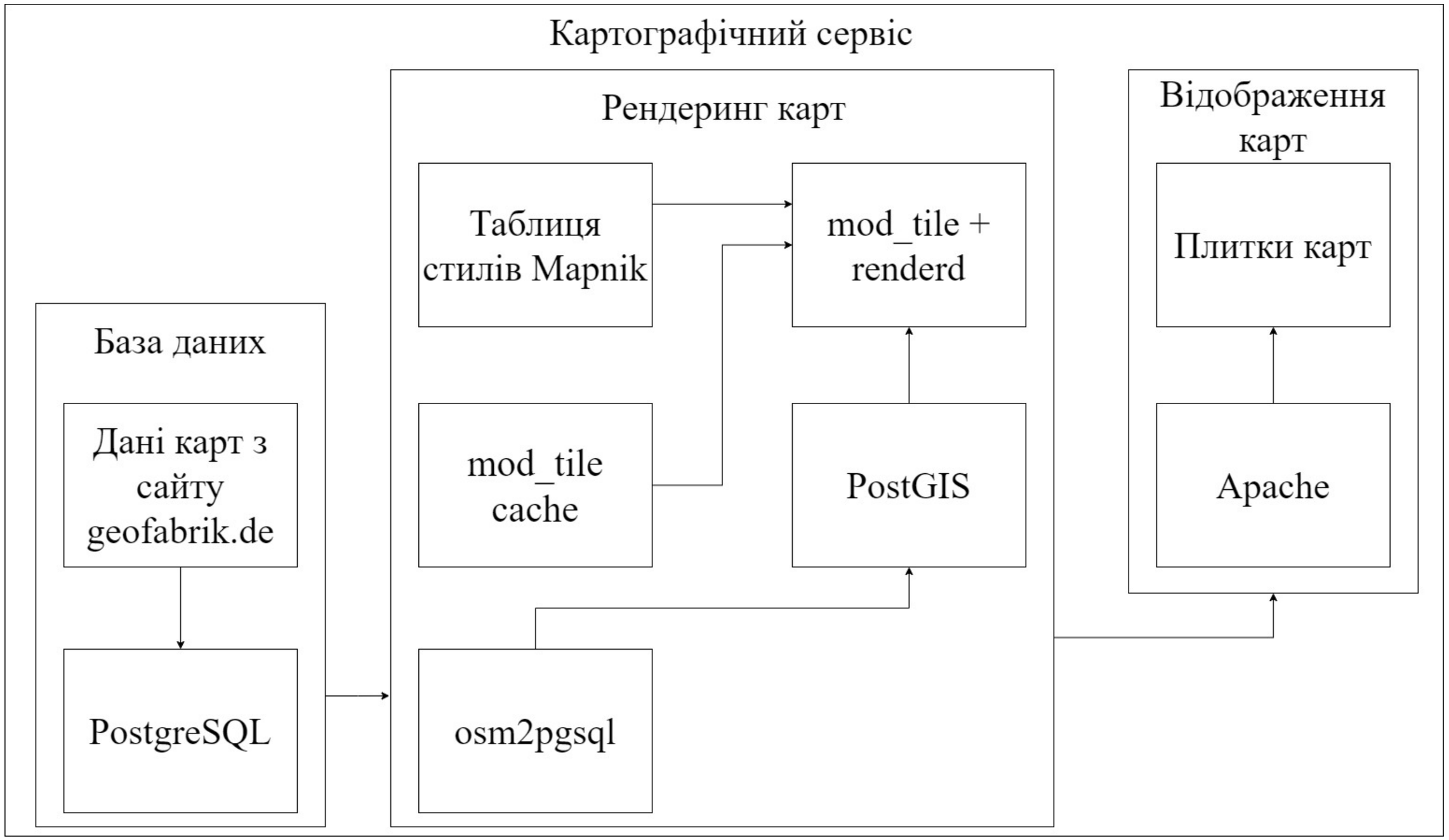
Додаток В

СТРУКТУРНА СХЕМА

до дипломного проєкту

на тему: «Картографічний сервіс на базі PostGIS»

Київ – 2021 року



					ІАЛЦ.467100.006 ДЗ	
Змн	*Арк.*	*№ докум.*	*Підпис*	*Дат*	*Картографічний сервіс на базі PostGIS*	*Літ.* / *Маса* / *Масш.*
Розроб.		Корсун Р. В.				
Перевір.		Іваніщев Б.В.				*Аркуш* / *Аркушів*
Т. Контр.						
Н. Контр.		Сімоненко В. П.			*Структурна схема*	НТУУ «КПІ», ФІОТ, ІО-71
Затверд.		Іваніщев Б.В.				