МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПОЛІСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій,
обліку та фінансів
Кафедра комп'ютерних технологій
і моделювання систем

Кваліфікаційна робота
на правах рукопису

Заверуха Максим Миколайович
(прізвище, ім'я, по батькові здобувача освіти)

УДК 004.4:316.334.2

КВАЛІФІКАЦІЙНА РОБОТА

Розробка інформаційної панелі для дослідження соціально-економічних процесів з використанням ГІС додатків
(тема роботи)

122 "Комп'ютерні науки"
(шифр і назва спеціальності)

Подається на здобуття освітнього ступеня бакалавр

кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____М.М. Заверуха
(підпис, ініціали та прізвище здобувача вищої освіти)

Керівник роботи
Топольницький Павло Петрович
(прізвище, ім'я, по батькові)
к.т.н., доцент
(науковий ступінь, вчене звання)

Житомир – 2025

Висновок кафедри

_____

за результатами попереднього захисту:

_____

Протокол засідання кафедри

_____

№ ____ від «_____» _____ 20____ р.

Завідувач кафедри

_____

_____ _____

_____

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

«_____» _____ 20____ р.

Результати захисту кваліфікаційної роботи

Здобувач вищої освіти _Заверуха Максим Миколайович____ захистив

(прізвище ,ім'я, по батькові)

кваліфікаційну роботу з оцінкою:

сума балів за 100-бальною шкалою _____

за шкалою ECTS _____

за національною шкалою _____

Секретар ЕК

_____ _____

_____

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

АНОТАЦІЯ

Заверуха М.М. Розробка інформаційної панелі для дослідження соціально-економічних процесів з використанням ГІС додатків за спеціальністю 122 – Комп'ютерні науки. – Поліський національний університет, Житомир, 2025.

У роботі розроблено інформаційну панель для інтерактивного аналізу соціально-економічних показників по країнах на основі відкритих даних. Об'єктом дослідження є процеси збирання, агрегації та візуалізації статистичних даних. Метою роботи є створення інструменту для порівняння соціально-економічних індикаторів з географічною прив'язкою. У роботі використано ETL-підхід, кешування, бібліотеки Leaflet.js і D3.js, а також модель ARIMA для прогнозування. Результатом є вебзастосунок, що надає доступ до карти, графіків, таблиць і функції експорту. Реалізовано адаптивний інтерфейс, що дозволяє ефективно працювати на різних пристроях. Отримане рішення може застосовуватись у дослідженнях, освіті та державному управлінні.

Ключові слова: ARIMA, QGIS, GeoJSON, інтерактивна карта, Leaflet.

SUMMARY

Development of a dashboard for the study of socio-economic processes using GIS applications in the specialty 122 - Computer Science - Polissya National University, Zhytomyr, 2025.

The paper develops a dashboard for interactive analysis of socio-economic indicators by country based on open data. The object of research is the processes of collecting, aggregating and visualizing statistical data. The purpose of the study is to create a tool for comparing socio-economic indicators with a geographical reference. The work uses an ETL approach, caching, Leaflet.js and D3.js libraries, and the ARIMA model for forecasting. The result is a web application that provides access to a map, graphs, tables, and an export function. An adaptive interface has been implemented, allowing for efficient work on different devices.

The resulting solution can be used in research, education, and public administration.

Keywords: ARIMA, QGIS, GeoJSON, interactive map, Leaflet.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface (інтерфейс прикладного програмування)

CSV – Comma-Separated Values (формат табличних даних, розділених комами)

ETL – Extract, Transform, Load (процес вилучення, трансформації та завантаження даних)

GeoJSON – Географічний розширений формат для представлення просторових даних у JSON

HTTP – HyperText Transfer Protocol (протокол передавання гіпертексту)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

Open Data – Відкриті дані

OWID – Our World in Data (платформа статистичних даних)

REST – Representational State Transfer (архітектурний стиль веб-сервісів)

TTL – Time To Live (час життя кешованого об'єкта)

UML – Unified Modeling Language (уніфікована мова моделювання)

XML – Extensible Markup Language (розширювана мова розмітки)

HTTPS – HyperText Transfer Protocol Secure (захищений протокол передавання гіпертексту)

ВСТУП

З розвитком концепції Open Data дедалі зростає потреба в зручному доступі до соціально-економічних показників, які розміщені на різних платформах (World Bank, Our World in Data, Wikipedia) у форматах XML, CSV, JSON. Інформація розпорошена, що ускладнює аналіз і порівняння даних між країнами. Повноцінне розгортання реляційної бази даних у таких випадках може бути надмірним, адже оновлення відбувається не частіше ніж раз на добу. Використання кешування із налаштованим терміном життя (TTL) знижує навантаження на зберігання й прискорює доступ до свіжих значень. Отже, створення веб-сервісу, який автоматично отримує дані, обробляє їх і тимчасово зберігає в кеші з терміном дії, а потім відображає у вигляді інтерактивних карт і графіків, є актуальним завданням для аналітиків, науковців і державних установ.

Метою цієї кваліфікаційної роботи є розробка веб-сервісу для автоматизованого збору, кешування та інтерактивного відображення соціально-економічних показників по країнах.

Завдання:

1. Визначити набір показників (ВВП, індекс людського розвитку, безробіття, інфляція тощо) і метод отримання даних із API World Bank (XML/JSON), OWID (CSV) та Wikipedia (API).
2. Формалізувати функціональні вимоги:
   - інтерфейс вибору країни, показника й періоду;
   - побудова інтерактивних карт і графіків з фільтрацією за роками;
   - механізм кешування із TTL для тимчасового зберігання.
3. Описати нефункціональні вимоги: час відповіді API ≤ 200 мс, час рендерингу карти/графіка ≤ 500 мс, HTTPS-з’єднання, ведення журналу (logging).
4. Спроєктувати серверну архітектуру на Flask:
   - ETL-скрипти на Python з налаштуванням TTL;

• реалізація REST-ендпоїнтів для видачі кешованих JSON/GeoJSON.

5. Розробити фронтенд із JavaScript-бібліотеками (Leaflet.js/D3.js для карти, Chart.js для графіків), забезпечити адаптивний дизайн та можливість експорту (PDF).

6. Провести тестування: юніт-тести ETL-скриптів (перевірка парсингу та TTL), інтеграційні тести API (pytest або Postman/Newman) і функціональні тести фронтенду (перевірка рендерингу карти та графіків).

7. Оцінити продуктивність (замір часу відповіді API через time у Python, вимір рендерингу карти через Performance API) і виконати оптимізацію (зменшення розміру GeoJSON, налаштування TTL).

– Об’єкт дослідження – процеси збору, обробки, кешування та візуалізації соціально-економічних даних із зовнішніх API (World Bank, OWID, Wikipedia).

- Предмет дослідження – програмні засоби та алгоритми (Python-скрипти ETL на основі requests, xml.etree.ElementTree, csv, json; механізм кешування із TTL; Flask-сервіс; JavaScript-бібліотеки Leaflet.js/D3.js і Chart.js) для реалізації веб-сервісу, який забезпечує автоматизований збір, тимчасове зберігання та інтерактивне відображення мультиджерелевих показників по країнах.

Методи дослідження

1. Аналіз документації API – вивчення специфікацій World Bank API (формат XML/JSON, параметри запитів, обмеження), OWID (CSV) і Wikipedia MediaWiki API (отримання таблиць у CSV).

2. Проєктування архітектури – побудова UML-діаграм (component, sequence) для моделювання кешу, значення – масив об’єктів.

3. Експериментальна розробка (Agile) – ітеративна реалізація серверного та клієнтського модулів із використанням Git (спринти, демонстрація проміжних результатів).

4. Юніт-тестування – перевірка коректності парсингу XML (xml.etree.ElementTree), CSV та JSON, встановлення TTL у кеші.

5. Інтеграційне тестування – перевірка REST-ендпоїнтів через pytest або Postman/Newman: статус-коди, формати JSON, TTL.

6. Функціональне тестування фронтенду – перевірка рендерингу карти (і графіків, фільтрації даних.

7. Статистичний аналіз даних – обчислення середніх, мінімальних і максимальних показників із кешованих JSON/GeoJSON та порівняння з оригінальними даними API для верифікації цілісності.

8. Оцінка продуктивності – замір часу відповіді серверних ендпоїнтів, часу рендерингу, оптимізація GeoJSON.

Практичне значення отриманих результатів

1. Спрощення інфраструктури: кеш із TTL замість повноцінної БД знижує вимоги до ресурсів і зменшує складність адміністрування.

2. Швидкий доступ до актуальних даних: оновлення кешованих наборів (TTL 24 години або налаштований інтервал) гарантує своєчасне відображення нових показників без довгих затримок.

3. Інтерактивна візуалізація: аналітики, науковці та держустанови можуть швидко формувати карти і графіки для порівняння показників по країнах.

4. Гнучкість і масштабованість: архітектура Back-end і Front-end дозволяє додавати нові джерела даних і розширювати функціонал без суттєвих змін у коді.

5. Навчально-методичне застосування: результати можуть слугувати прикладом у навчальних курсах і практичних семінарах із тем «Open Data», «ETL-процеси», «Веб-розробка».

Кваліфікаційна робота складається з таких розділів:

1. Вступ.
2. Розділ 1. Аналіз предметної області.
3. Розділ 2. Проєктування інформаційної технології.
4. Розділ 3. Реалізація та тестування.
5. Загальні висновки.

6. Список використаних джерел.

7. Додатки (фрагменти коду, UML-діаграми, скриншоти інтерфейсу, таблиці результатів тестування).

Орієнтовний обсяг основного тексту – 18–20 сторінок (без урахування списку джерел та додатків).

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика об’єкта інформатизації та його інформаційні потреби

Об’єктом інформатизації в межах даної роботи є користувачі, які працюють зі статистичними соціально-економічними даними в контексті аналізу за країнами. До таких користувачів належать дослідники, студенти, викладачі, аналітики, журналісти, а також працівники державних органів. Вони потребують швидкого доступу до достовірної, актуальної та уніфікованої інформації з різних міжнародних джерел.

У роботі такі користувачі зазвичай стикаються з необхідністю порівняння статистичних показників між країнами, аналізу динаміки обраного індикатора за роками, побудови візуалізацій у вигляді графіків або картографічних відображень, а також збереження отриманих результатів для подальшого використання. Джерела інформації, які використовуються у цьому процесі (зокрема, World Bank, Our World in Data, Wikipedia), мають відкриті API, але не забезпечують зручного інтерфейсу для безпосередньої роботи з даними без попередньої обробки.[3]

Основні інформаційні потреби об’єкта інформатизації включають:

– Порівняльний аналіз показників по країнах: можливість вибрати показник (наприклад, ВВП на душу населення, рівень безробіття, індекс людського розвитку) та переглянути його значення у різних країнах;

– Аналіз динаміки: відображення зміни показника за роками у вигляді графіка або табличного представлення;

– Інтерактивна візуалізація: використання інтерактивної карти (на основі GeoJSON) для просторового порівняння показників;

– Гнучке фільтрування: можливість вибору країни, показника, часового періоду за допомогою фільтрів, списків та інших елементів керування інтерфейсом;

– Експорт даних: збереження результатів у форматах CSV або GeoJSON для подальшого аналізу або використання в інших системах.

Оскільки сервіс є публічним, його інтерфейс не передбачає авторизації, що додатково знижує поріг входу для користувачів. Водночас реалізація мінімального, інтуїтивно зрозумілого інтерфейсу дозволяє забезпечити доступ до даних без необхідності технічної підготовки або попереднього ознайомлення з документацією джерел.

Таким чином, розробка вебсервісу, що автоматизує доступ до структурованих відкритих соціально-економічних даних, відповідає реальним потребам широкого кола користувачів, які не мають спеціальних технічних інструментів для самостійного оброблення й інтеграції даних з різних джерел.

1.2 Джерела даних та інструменти їх обробки

У рамках розробки вебсервісу використовуються дані з відкритих міжнародних джерел, які надають доступ до статистичних соціально-економічних показників через API або пряме завантаження файлів. Основними джерелами інформації є:

- World Bank – забезпечує доступ до великого набору макроекономічних та соціальних показників. Дані отримуються через API у форматах XML або JSON;
- Our World in Data (OWID) – надає CSV-файли з історичними та поточними глобальними показниками (CO_2, ВВП, смертність, вакцинація тощо);
- Wikipedia / MediaWiki API – джерело для вибіркових табличних даних, зокрема історичних чи класифікаційних, які оновлюються спільнотою.

З огляду на формат і джерело даних, для їх обробки застосовуються відповідні інструменти:

- requests – для виконання HTTP-запитів до зовнішніх API (World Bank, OWID, Wikipedia);

- xml.etree.ElementTree – для парсингу XML-даних, зокрема відповідей від World Bank API;
- csv – для обробки табличних даних OWID та локальних CSV-таблиць;
- json – для роботи з JSON-структурами, зокрема при кешуванні результатів у форматі GeoJSON або передачі у фронтенд;
- стандартні бібліотеки Python (os, shutil, sys, time, urllib.parse) – для навігації по файловій системі, обробки шляхів, параметрів командного рядка та організації кешування;
- logging – для ведення журналу дій та діагностики помилок під час отримання та обробки даних.

Оскільки в роботі не застосовується постійне зберігання даних у базі, замість цього реалізовано тимчасове кешування отриманих результатів. Це дозволяє зберігати оброблені дані у пам’яті (або тимчасових JSON-файлах) із заданим часом життя (TTL), що відповідає характеру оновлення даних у джерелах (раз на день або рідше). Такий підхід спрощує інфраструктуру та зменшує вимоги до ресурсів, зберігаючи при цьому актуальність інформації для користувачів.[6]

Таким чином, набір використовуваних інструментів охоплює повний цикл обробки: отримання → розбір → уніфікація → кешування → передача у візуалізаційний модуль. Обрані засоби дозволяють реалізувати весь процес без створення окремої бази даних або складних ETL-конвеєрів, що відповідає вимогам до легкості й швидкодії сервісу.

1.3. Функціональні вимоги до інформаційної системи

Інформаційна технологія, що розробляється, має забезпечувати користувачеві зручний доступ до порівняльної соціально-економічної статистики у форматі, придатному для аналітики, презентацій або подальшої обробки. Основною вимогою до системи є здатність збирати дані з кількох авторитетних відкритих джерел, тимчасово зберігати їх у кеші, відображати у

візуальній формі та дозволяти користувачеві взаємодіяти з отриманою інформацією.

Оскільки джерела, з якими працює система, містять різні формати даних (XML, CSV, JSON), першим функціональним завданням є їх коректне отримання, обробка та уніфікація. Цей процес реалізується через набір ETL-скриптів на Python, що регулярно витягують дані з API World Bank, OWID та інших, перетворюють їх у внутрішній формат і зберігають у кеші із заданим терміном життя. Кешування дозволяє уникнути повторних запитів до джерел та прискорити взаємодію користувача з інтерфейсом. На поточному етапі TTL встановлено як фіксований – наприклад, 24 години – однак архітектура передбачає можливість зміни цього параметра в майбутньому.[10]

На боці клієнта передбачено три ключові засоби представлення інформації: інтерактивна карта, графік динаміки та таблична форма. Користувач має можливість самостійно обирати країну, показник і часовий інтервал, після чого система виводить відповідні значення в усіх трьох візуальних форматах. Карта реалізована на базі GeoJSON, графіки – через Chart.js. Табличне подання дозволяє переглядати числові значення по роках. Усі елементи інтерфейсу пов’язані між собою: вибір на одному з них синхронізується з іншими.

Важливим є також те, що система не потребує авторизації: вона створена як публічний інструмент для широкого кола користувачів. При цьому інтерфейс має бути адаптивним, тобто коректно працювати на пристроях із різною шириною екрана – настільних ПК, планшетах та смартфонах.

Функція експорту передбачає можливість збереження результатів у форматах CSV або GeoJSON. Це дає змогу використовувати дані у зовнішніх аналітичних системах або додавати їх до звітів і публікацій. Експорт реалізовано як через інтерфейс, так і безпосередньо через API, що розширює можливості застосування сервісу.

Оскільки система працює з періодичним автоматичним збором даних, важливою є наявність логування процесів. Усі етапи отримання, парсингу та збереження даних фіксуються з використанням стандартного механізму логування Python, що дозволяє виявляти помилки або затримки в роботі ще до появи проблем у взаємодії з користувачем. Логування на клієнтській стороні не реалізовано, однак за потреби така функція може бути додана для збору статистики використання.

У підсумку, система повинна бути простою у використанні, візуально зрозумілою, технічно гнучкою до змін і масштабованою в разі зростання кількості користувачів або підключення нових джерел.[2]

Висновки до розділу 1

У першому розділі проведено аналіз предметної області, у межах якого встановлено об'єкт інформатизації – користувачів, які працюють зі статистичними соціально-економічними даними. Визначено основні інформаційні потреби цих користувачів, серед яких – доступ до актуальних показників, можливість порівняльного аналізу, перегляд динаміки та візуалізація інформації у зручному форматі.

Оцінено особливості використання відкритих джерел статистичних даних, таких як World Bank, Our World in Data та Wikipedia, а також інструменти для їх обробки та уніфікації. Обґрунтовано доцільність використання механізму кешування з фіксованим терміном життя як ефективної альтернативи повноцінним базам даних у задачах періодичного збору даних.

На основі встановлених інформаційних потреб сформульовано функціональні вимоги до інформаційної системи, яка має забезпечити отримання, обробку, візуалізацію та експорт даних, при цьому залишаючись доступною, адаптивною й масштабованою.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

2.1 Загальна архітектура системи

Архітектура інформаційної технології базується на розподіленому підході, що передбачає поділ системи на окремі функціональні компоненти. Кожен компонент виконує визначену роль: збір, попередню обробку, зберігання або візуалізацію даних. Такий підхід забезпечує модульність, спрощує супровід і дозволяє легко адаптувати систему до нових вимог без суттєвих змін її структури.

Основу серверної логіки становить окремий ETL-скрипт, написаний мовою Python. Його завдання – отримання соціально-економічних даних із кількох відкритих джерел, включаючи API World Bank (у форматі XML), CSV-файли платформи OWID та таблиці з Wikipedia. Після завантаження дані проходять етап трансформації та агрегування: показники уніфікуються за структурою, ключі країн синхронізуються, часові ряди формуються у стандартному вигляді. Для цього використовується набір вбудованих бібліотек Python: requests, xml.etree.ElementTree, csv, json, а також додаткові службові модулі (os, shutil, time, logging).[8]

Паралельно відбувається генерація просторових даних у форматі GeoJSON, які відповідають обраним країнам і територіям. Сирі геодані, що надходять із відкритих джерел (наприклад, у форматі shapefile або GeoPackage), попередньо обробляються в QGIS. У цьому середовищі здійснюється нормалізація атрибутів, приведення систем координат до WGS 84, редукція зайвої деталізації меж та перевірка відповідності кодів ISO країн. QGIS не використовується безпосередньо у роботі вебсервісу, однак відіграє технічну роль у забезпеченні коректної просторової прив’язки під час підготовки даних.

Оброблені дані – як табличні, так і просторові – зберігаються у кеші у вигляді JSON- і GeoJSON-файлів. Кеш формується для кожного набору

(показник + країна + рік) та має фіксований термін життя (TTL), орієнтовно 24 години. Оновлення кешу може виконуватись вручну або в автоматичному режимі за розкладом (через системний планувальник завдань, наприклад cron).

Фронтенд-частина системи реалізована за допомогою HTML, CSS та JavaScript. Основну взаємодію з користувачем забезпечують візуальні компоненти: карта, графік та таблиця. Для реалізації карти використовується бібліотека Leaflet.js, для графіків – Chart.js. Інтерфейс дозволяє користувачеві вибрати країну, показник і часовий період, після чого виконується AJAX-запит до кешованих даних, і результат виводиться в інтерактивному форматі.

Передача даних здійснюється через Flask-сервер, що виконує дві функції: роздачу статичних файлів (HTML, CSS, JS) і обслуговування простих API-запитів до кешованих JSON-файлів. Уся логіка обробки зосереджена на стороні ETL-скрипту, що дозволяє зберігати серверну частину легкою та масштабованою. Загальна схема побудови системи подана на рисунку 2.1.

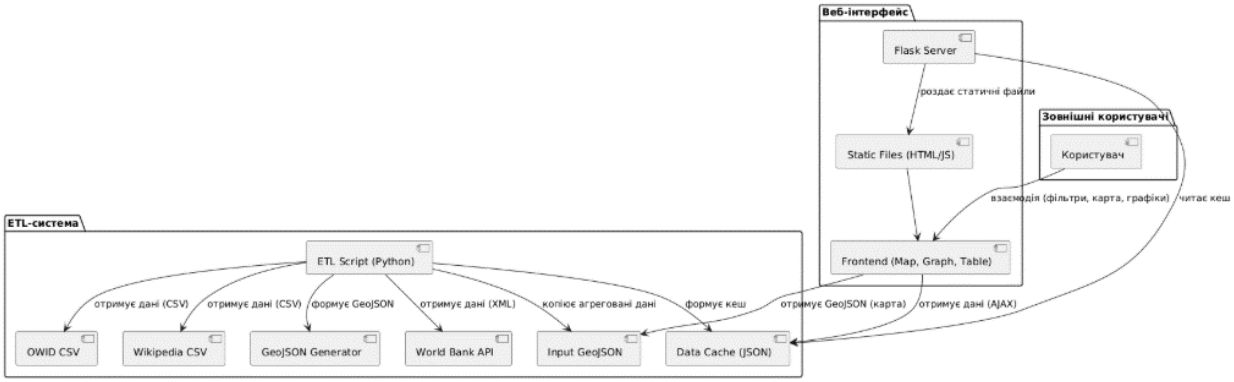


Рис. 2.1 – UML діаграма архітектури інформаційної панелі з використанням ETL, кешу та вебінтерфейсу

Представлена архітектура дозволяє ефективно розмежувати функціональну відповідальність між компонентами системи. Збір, обробка й прогнозування даних відбуваються окремо від візуалізації, що дає змогу підтримувати систему у розподіленому режимі, масштабувати її за потреби

та спрощувати впровадження нових джерел або форматів даних. Така побудова забезпечує гнучкість, стабільність роботи і придатність до подальшого розширення функціоналу.

2.2 Логіка функціонування та обмін даними

Функціонування інформаційної системи базується на послідовній обробці даних та взаємодії з користувачем у режимі читання з кешу. Архітектура системи поділена на три основні рівні: модуль збору та підготовки даних (ETL), механізм збереження проміжних результатів (кеш) і фронтенд-інтерфейс для візуального подання інформації.

Першим етапом є запуск ETL-модуля, реалізованого як окремий скрипт на мові Python. Скрипт здійснює підключення до джерел відкритих даних – API Світового банку (у форматі XML), CSV-файлів OWID та таблиць Wikipedia. Після отримання дані проходять етап попередньої обробки: вирівнювання форматів, синхронізація ідентифікаторів країн, перевірка цілісності, відкидання неповних записів. Потім дані агрегуються за роками та країнами і формуються в кешовані файли у форматі JSON. Паралельно генеруються файли формату GeoJSON, які використовуються на карті.

Другий етап починається з відкриття вебінтерфейсу користувачем у браузері. Фронтенд, написаний із використанням HTML, CSS та JavaScript (Leaflet.js, D3.js), виконує асинхронні запити до кешованих файлів, розміщених у директорії static. Запити реалізуються через стандартний механізм AJAX і не потребують окремих API-ендпоїнтів, оскільки система працює з попередньо збереженими файлами. Дані завантажуються у відповідні компоненти інтерфейсу: карта (GeoJSON), графік і таблиця (JSON). Весь процес відбувається без перезавантаження сторінки.[5]

Робота Flask-сервера в системі обмежується роздачею статичних файлів та кешованих даних. Будь-яка логіка обробки або динамічних запитів із боку сервера відсутня – система працює за принципом «збір → кеш →

візуалізація». У разі, якщо кеш застарілий або відсутній, інтерфейс все одно завантажується, однак виводяться неповні або попередні дані.

Алгоритм функціонування системи умовно поділяється на дві послідовності: серверна обробка (ETL) та клієнтська взаємодія (візуалізація). Цей процес подано на рисунку 2.2.

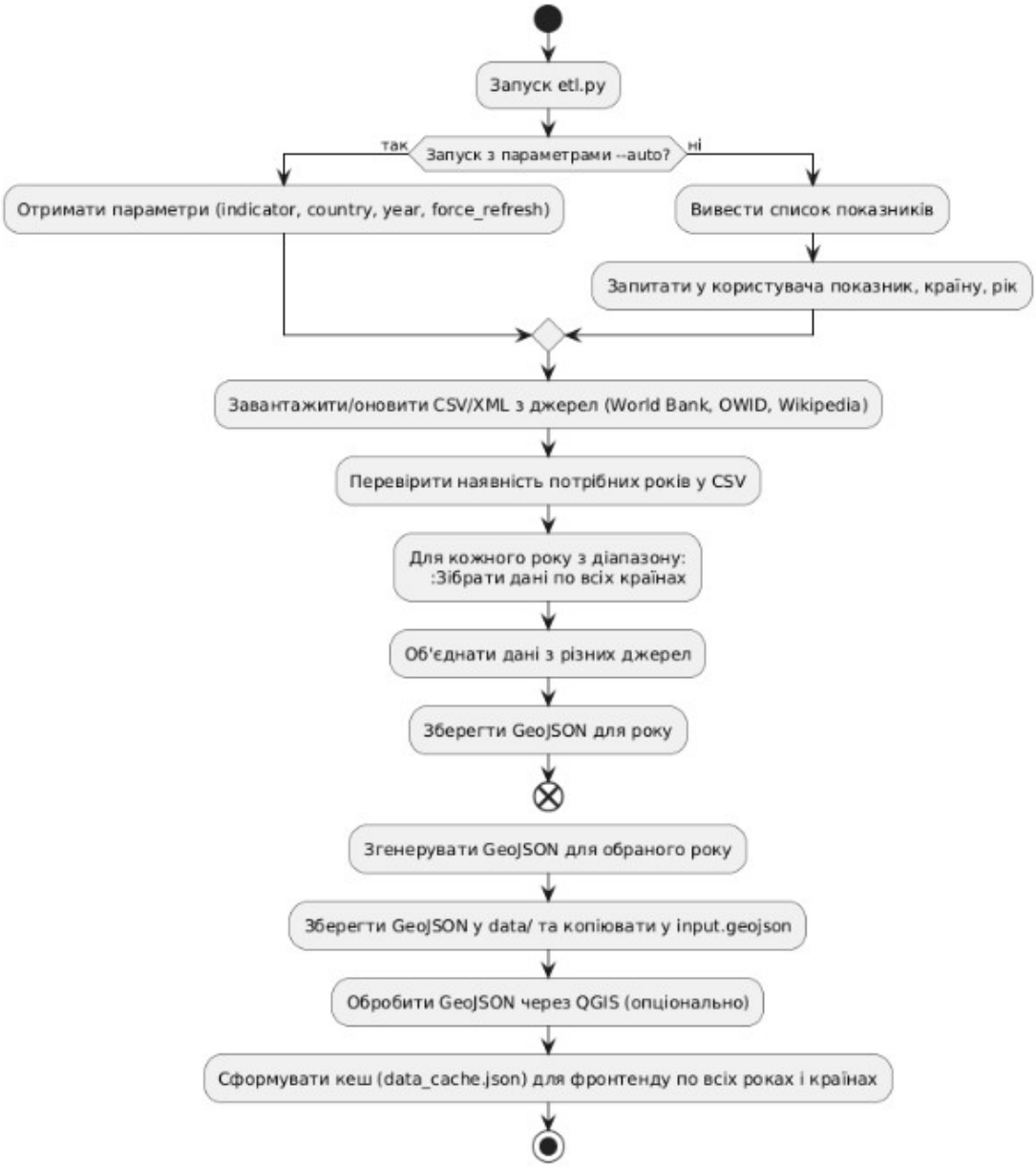


Рис. 2.2 – ETL діаграма активностей

У такий спосіб система забезпечує повну автоматизацію процесу збору та виводу соціально-економічної інформації без необхідності динамічного рендерингу на стороні сервера. Логіка побудови з орієнтацією на кеш і

незалежність між компонентами дає змогу легко масштабувати рішення, переносити його на інші платформи та адаптувати до нових джерел даних.

2.3 Моделювання даних та функціональні блоки

У цьому підпункті представлено результати інфологічного моделювання ключових структур даних і описано функціональні блоки системи з урахуванням ролі кожного елемента в обробці та відображенні інформації.

Інфологічне моделювання кешу

Кешована інформація зберігається у файлі static/data_cache.json як плоский словник, де кожний запис відповідає комбінації показника, ISO-коду країни та року (наприклад, "gdp_UKR_2000": 1260.5). Така структура забезпечує швидкий доступ до окремих значень за ключем, без необхідності додаткової навігації по вкладених масивах. У разі відсутності даних для певного року значення присвоюється null. Це зображено на логічній моделі кешу (рис. 2.3), де клас Cache містить атрибут entries: Map<String, Float>; ключі мають формат indicator_country_year, а значення – числовий показник.

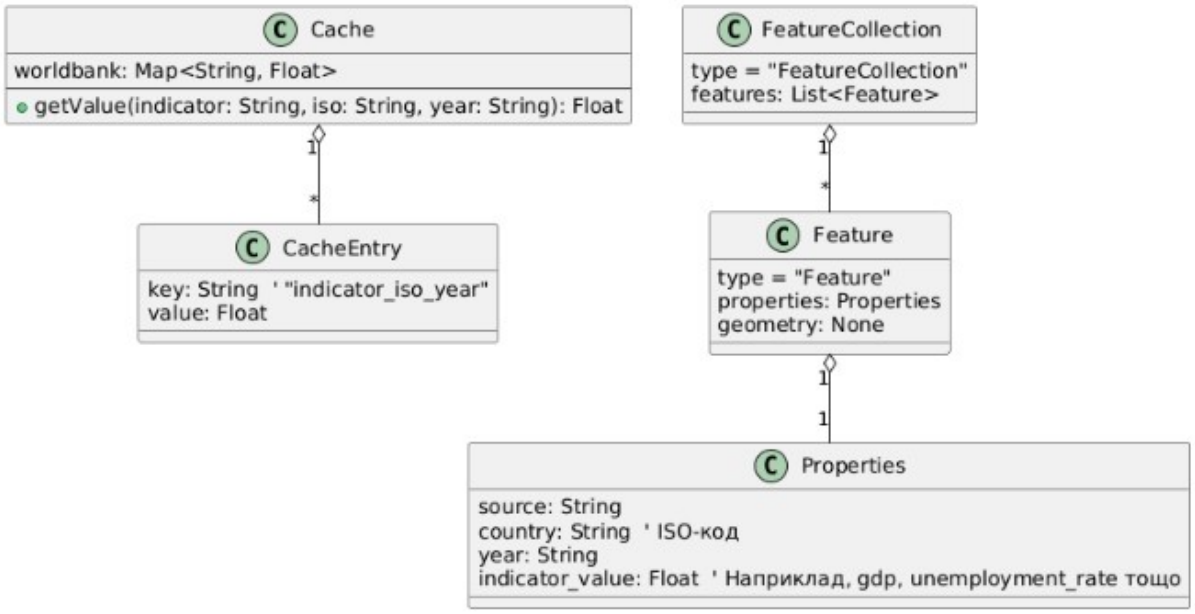


Рис. 2.3 – UML діаграма класів «Логічна модель кешу»

Інфологічне моделювання GeoJSON

Файл GeoJSON містить усі країни в одному об’єкті FeatureCollection. Кожний елемент включає у властивості (properties) ISO-код країни (country),

рік (year), значення показника (наприклад, gdp), а також джерело даних (source). Геометрія (geometry) може бути null у базовій версії, якщо передбачається лише семантичне поєднання атрибутів. Відповідну структуру схематично зображено на рис. 2.4.

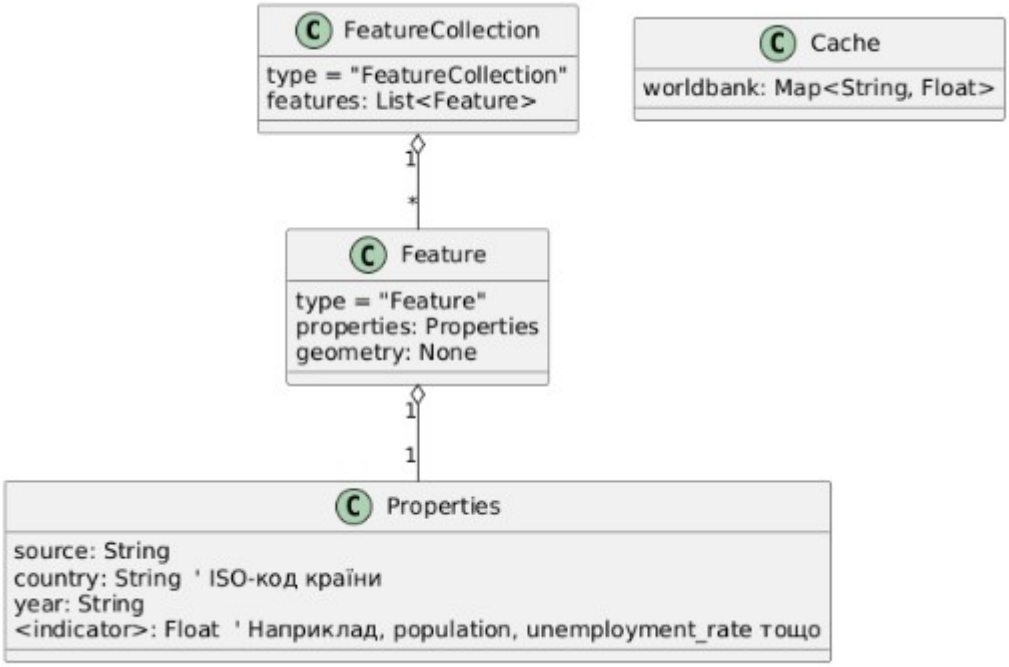


Рис. 2.4 – UML діаграма клас «Структура GeoJSON»

Для аналізу й прогнозування динаміки соціально-економічних показників у системі застосовується модель ARIMA (AutoRegressive Integrated Moving Average). Завдяки здатності враховувати тренди, сезонність і випадкові коливання, ARIMA дозволяє отримувати короткострокові прогнози значень на основі історичних даних. Загальний вигляд моделі ARIMA(p, d, q) подано формулою:

¿

де L – оператор зсуву, φ_i і $\varnothing_i$– коефіцієнти моделі, ε_t – білий шум. Диференціювання¿ усуває нестаціонарність ряду.

У системі ARIMA інтегровано в ETL-модуль як окремий крок після агрегації історичних значень. Вихідним масивом для побудови моделі стає ряд попередньо очищених даних, де пропуски замінено на null, а часові ряди приведено до регулярної шкали. За допомогою бібліотеки statsmodels у

Python обчислюються оптимальні параметри $\{\varphi_i\}\{\varphi_i\},\{\varnothing_i\}\{\varnothing_i\}$ і d для обраного показника. Результати прогнозу (наприклад, на три-п’ять років уперед) зберігаються в окремий JSON-файл і передаються у фронтенд для відображення поруч із реальними даними.[8]

ARIMA обрано з таких причин:

– здатність моделювати як трендові, так і сезонні компоненти;
– робота з неповними даними (пропущені значення можуть бути інтерпольовані або залишені як null);
– простота інтеграції у Python через готові інструменти (у тому числі statsmodels).

Рис. 2.5 – Алгоритм побудови прогнозу за допомогою ARIMA

Функціональні блоки системи

Система складається з чотирьох основних модулів:

1. ETL-модуль (python/etl.py):

Цей модуль запускається вручну або за розкладом (через cron). Він поєднує кілька етапів: отримання сирих даних із API World Bank (XML), CSV

OWID та Wikipedia, уніфікацію кодів країн (усі дані приводяться до ISO-кодів), заповнення відсутніх років значенням null, формування кешу у вигляді плоского словника JSON та створення GeoJSON із властивостями source, country, year і самими значеннями показника. Підготовка сирих геоданих спочатку проходить у QGIS для нормалізації атрибутів і переведення до WGS 84, після чого готові GeoJSON-файли передаються на сервер.

2. Кеш (файлова система):

Результати ETL зберігаються у двох файлах: data_cache.json (словникова структура) і data_map.geojson (архітектура геопросторових даних). Кеш забезпечує швидкий доступ до даних без повторних запитів до зовнішніх API, а TTL налаштований на 24 години. Якщо в момент запуску фронтенду файл не оновлено, користувач отримає попередні дані.

3. Flask-сервер:

Сервер відповідає лише за роздачу статичних ресурсів: HTML-сторінок, JavaScript-скриптів, CSS, а також файлів data_cache.json і data_map.geojson. Він не виконує обробки запитів чи агрегації, тому залежить від актуальності кешу. Мінімальна конфігурація включає кореневий маршрут (/) для інтерфейсу та прямий доступ до директорії static.

4. Фронтенд-інтерфейс (HTML/CSS/JS):

Клієнтська частина побудована із застосуванням Leaflet.js для карти та D3.js або Chart.js для графіків. Інтерфейс пропонує фільтри для вибору країни, показника й року. Після вибору параметрів виконується AJAX-запит до відповідних JSON- або GeoJSON-файлів, результат передається компонентам карти, графіка та таблиці. Адаптивний дизайн забезпечує коректне відображення на настільних ПК, планшетах і телефонах.

Рис. 2.5 – Схема функціональних блоків системи

З огляду на пріоритетність джерел, при агрегації даних ETL-модуль віддає перевагу World Bank; якщо для певної країни/року дані відсутні, він звертається до OWID або Wikipedia (як «fallback»). У результаті кожен запис у кеші має відповідний пріоритет: первинне значення або null, якщо даних немає ніде.

Висновки до розділу 2

У Розділі 2 представлено загальну архітектуру інформаційної системи, яка складається з чотирьох функціональних блоків: ETL-модуля, кешу, Flask-сервера та фронтенд-інтерфейсу. Показано, що ETL-модуль відповідає за одержання, уніфікацію та агрегацію даних із джерел (World Bank, OWID, Wikipedia), формує кеш у вигляді плоского словника JSON і готує GeoJSON для карти. Кешування у файловій системі з TTL 24 години дозволяє

прискорити доступ і знизити навантаження на зовнішні API. Flask-сервер виконує роль роздачі статичних файлів, не здійснюючи динамічної обробки, що спрощує його конфігурацію та забезпечує масштабованість. Фронтенд-інтерфейс, реалізований на HTML/CSS/JS (Leaflet.js, D3.js), динамічно підвантажує дані через AJAX і відображає їх у вигляді інтерактивної карти, графіків і таблиць. Алгоритми функціонування системи побудовані за принципом «ETL → кеш → візуалізація», що гарантує незалежність між компонентами та можливість подальшого розширення.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис середовища розробки та інструкція з розгортання

Розробка інформаційної панелі здійснювалася на платформі Windows 10/11 із використанням Python версії 3.10 (сумісність з 3.8+ забезпечена) та віртуального середовища (venv). Для реалізації серверної частини застосовано фреймворк Flask (версія 2.x), що виконує роль простого файлового сервера для статичних ресурсів і кешованих даних. У фронтенді використані бібліотеки Leaflet.js (не нижче 1.9) для геопросторової візуалізації та D3.js (версія 7+) для побудови інтерактивних графіків. Редактором коду обрано Visual Studio Code, а для роботи з геоданими за попередньої обробки застосовувався QGIS.[7]

Для налаштування проєкту виконуються такі кроки:

1. У командному рядку необхідно перейти у папку з проєктом і створити віртуальне середовище:

```
python -m venv venv
```

Після цього активувати його командою .\venv\Scripts\activate (Windows) або source venv/bin/activate (Linux).

2. Інсталювати залежності, зокрема Flask і requests, перелік яких міститься у файлі requirements.txt:

```
pip install -r requirements.txt
```

Якщо планується робота з XML через модуль lxml, слід додати його до requirements.txt та виконати інсталяцію:

pip install lxml

3. Переконатися, що структура директорій відповідає стандарту Flask:

app.py або server.py (основний файл для запуску сервера),

каталог static/, який містить підкаталоги css/, js/, data/ (з кешованими JSON та GeoJSON),

каталог templates/ із файлами HTML (якщо використовується рендеринг через Jinja; у базовій версії достатньо статичних HTML, що розміщені у static/).

4. Запуск ETL-скрипту здійснюється у кореневій папці проєкту через команду:

python etl.py --auto <indicator> <country> <year>

де параметри indicator, country та year визначають, який показник і для якої країни слід оновити. Сценарій працює в локальному середовищі, формує файли static/data/data_cache.json і static/data/data_map.geojson. Для автоматичного оновлення навколо доби можна налаштувати cron (Linux) чи Планувальник завдань (Windows), викликавши той самий python etl.py за розкладом.

5. Після формування кешу для запуску вебсервера в тому самому терміналі потрібно виконати:

set FLASK_APP=app.py (Windows)
export FLASK_APP=app.py (Linux)
flask run

За замовчуванням сервер почує запити на http://localhost:5000/. Він роздаватиме статичні HTML- та JS-файли і надаватиме доступ до data_cache.json і data_map.geojson.

6. Фронтенд-інтерфейс не потребує окремого збірника; достатньо відкрити у браузері http://localhost:5000/index.html (або аналогічний шлях до головного HTML). При кожному завантаженні сторінки JavaScript-логіка перевіряє наявність актуальних файлів у static/data/ і динамічно відображає дані.

Таким чином, середовище розгортання включає локальний сервер Flask, підготовлені дані ETL-модулем і файли фронтенду, які гарантують коректну роботу інформаційної панелі як на Windows, так і на Linux.

3.2 Опис інтерфейсу та керівництво користувачу

Інтерфейс інформаційної панелі складається із трьох взаємопов’язаних компонентів: карти, графіка динаміки та табличного подання. Його метою є забезпечити швидкий доступ до порівняльного аналізу показників по країнах.

– Після завантаження головної HTML-сторінки користувач бачить верхню панель із випадаючими списками для вибору:

– Показник (наприклад, ВВП, індекс людського розвитку, рівень безробіття),

– Країна (ISO-коди зі словника всіх країн),

– Рік (діапазон від найдавнішого до найактуальнішого року в кеші).

Далі межа екрана ділиться на дві частини: велику ліву область займає інтерактивна карта, а праворуч розташовано графік динаміки та під ним – таблицю з цифровими значеннями.

Карта реалізована за допомогою Leaflet.js. Після вибору показника і року через AJAX підвантажується файл static/data/data_map.geojson. Кожна країна на карті підсвічується кольором, який відповідає інтенсивності значення показника (градація блідих відтінків при низьких значеннях до

насичених – при високих). Клацання на країну відкриває спливаюче вікно з інформацією: назва країни, рік, значення показника та джерело даних. Приклад відображення карти можна побачити на рисунку 1.

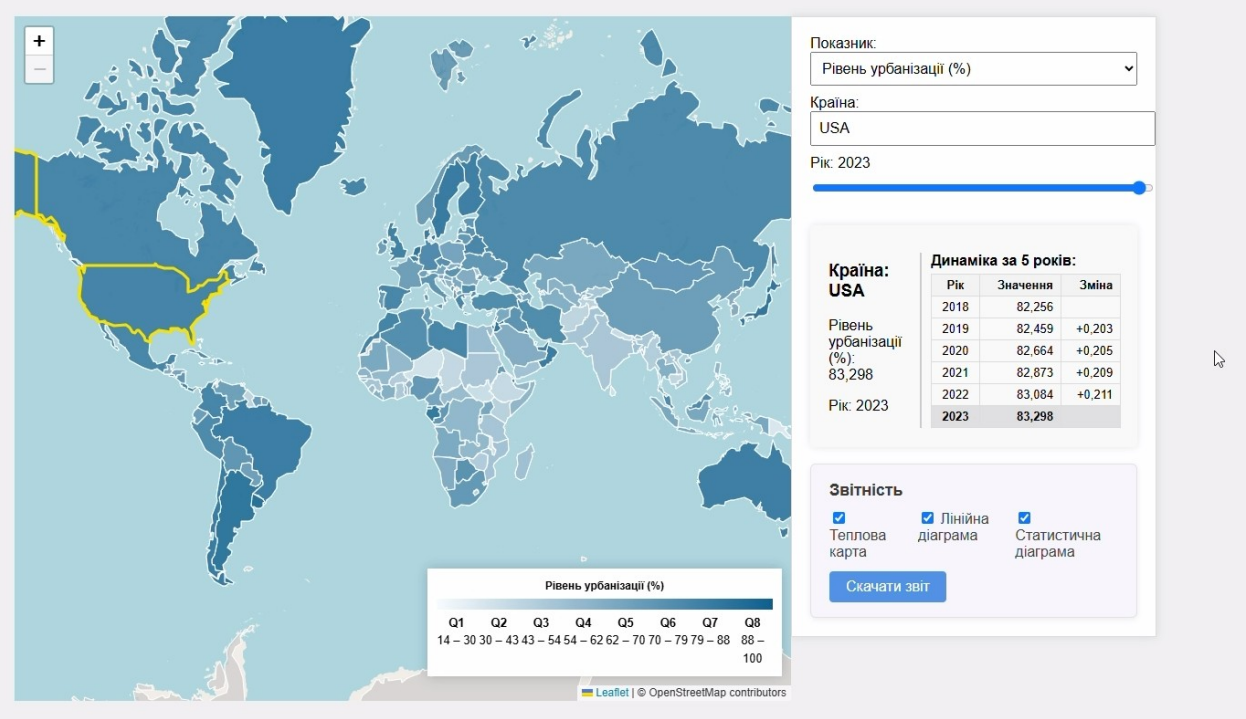
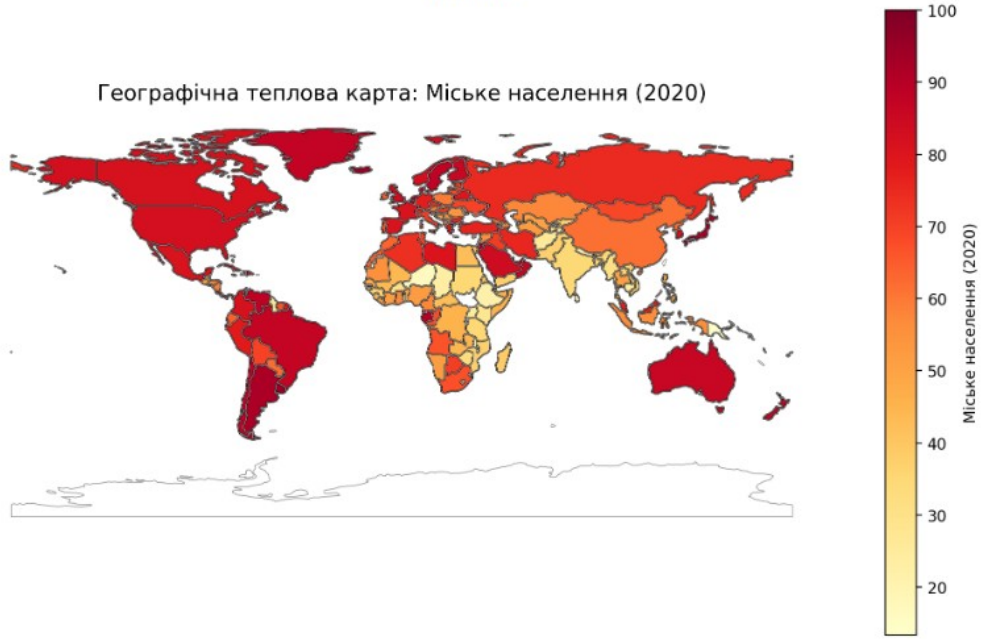


Рис. 3.1 – Інформаційна панель з картою, вибором показника, країни та року

Графік динаміки побудований із застосуванням D3.js. Він отримує масив значень із файлу static/data/data_cache.json за ключем <indicator>_<country>_<year>. Потім будує лінійний графік, де по осі абсцис відкладається час (роки), а по осі ординат – значення показника. Наведення мишкою на точку графіка відображає точне значення у спливаючій підказці. Приклад графіку динаміки показника та математичне пояснення ARiMA можна побачити на рисунку 2.

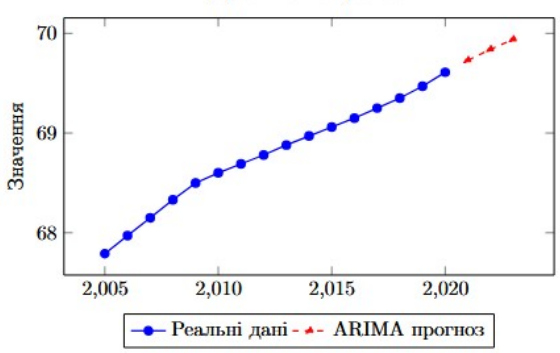
Географічна теплова карта показника

Карта ілюструє розподіл значень показника по країнах за обраний рік. Чим яскравіший колір, тим вище значення.



Динаміка показника та прогноз (ARIMA)

Діаграма ілюструє зміну значення показника за роками для обраної країни. Пунктирна лінія — прогноз ARIMA.



Математичне пояснення ARIMA

ARIMA (AutoRegressive Integrated Moving Average) — це модель для прогнозування часових рядів. Вона описується рівнянням:

$$y_t = c + \phi_1 y_{t-1} + \theta_1 \varepsilon_{t-1} + \varepsilon_t$$

де y_t — значення ряду, ϕ_1 — коефіцієнт авторегресії, θ_1 — коефіцієнт ковзного середнього, ε_t — випадкова похибка. У цьому звіті використовується ARIMA(1,1,1): 1 лаг авторегресії, 1 диференціювання, 1 лаг ковзного середнього.

1

Рис. 3.2 – Графік динаміки показника та прогноз (ARIMA)

Табличне подання сформоване за допомогою стандартного HTML-елемента <table>, наповненого через JavaScript. У таблиці розміщено два

стовпці (рік і значення), що дозволяє користувачу швидко переглянути цифрові дані. Під час вибору іншого показника чи країни дані в таблиці оновлюються автоматично.

Кроки користувача для отримання інформації:

– Відкрити веб-браузер та перейти за адресою http://localhost:5000/index.html.

– У верхніх елементах керування обрати потрібний показник із випадаючого списку.

– Обрати країну за ISO-кодом.

– Задати цікавий інтервал років або вибрати один рік.

– Інтерфейс автоматично формує AJAX-запити до data_cache.json

– data_map.geojson і відображає карту з підсвіченими країнами.

На графіку та в таблиці показується динаміка показника за вибраний період.

При потребі користувач може експорт із відповідних кнопок: натиснувши «Завантажити звіт» отримати звіт в форматі PDF з даними за показником.

Особливості адаптивності. Інтерфейс адаптовано за допомогою CSS-медіа-запитів: на планшетах і мобільних пристроях карта займає всю ширину екрану, а графік і таблиця згортаються за допомогою вкладених табів або випадаючого меню, що забезпечує зручність перегляду на невеликих екранах.

Приклад гістограми розподілу значень показника можна побачити на рисунку 3.

Прогноз будується на основі попередніх значень ряду.

Гістограма розподілу значень

Гістограма відображає розподіл значень показника по країнах за обраний рік. Кожен стовпчик відповідає певному інтервалу значень.

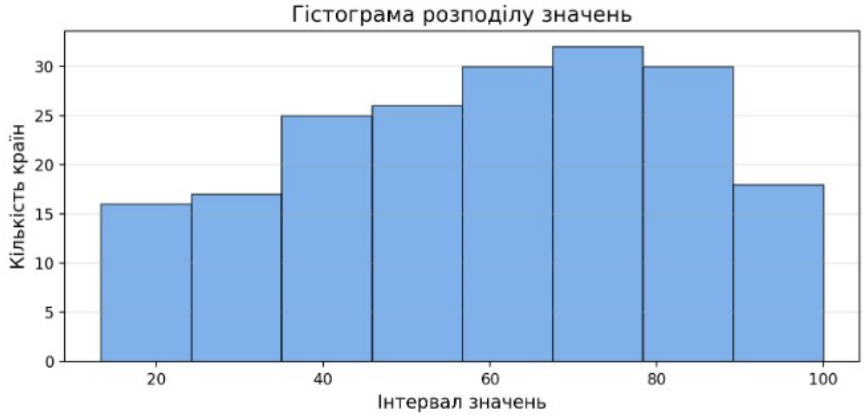


Рис. 3.3 – Гістограма розподілу значень

Таким чином, інтерфейс забезпечує інтуїтивну взаємодію, дозволяючи швидко переходити від карти до графіка та таблиці, а експортні функції дають змогу завантажити дані для подальшого використання.

3.3 Тестування

Тестування інформаційної системи включало перевірку коректності роботи ETL-модуля, надійності роздачі даних через Flask і коректного відображення інформації в інтерфейсі.

Юніт-тести для ETL-скрипту виконувалися з використанням pytest. Основну увагу приділено тестуванню окремих функцій:

Перевірка завантаження даних із API World Bank: функція повертає очікуваний формат словника при імітованих відповідях.

Тест парсингу CSV-файлів OWID та таблиць Wikipedia: на основі контрольного набору тестових рядків перевіряється правильність розбору, приведення до ISO-кодів і формування ключів (indicator_country_year).

Перевірка логіки TTL: створювався тестовий словник із датами модифікації файлу, після чого перевірялося, чи коректно повертається повідомлення про застарілість даних або про наявність актуального кешу.

Тести на генерацію GeoJSON: перевірка відповідності структури FeatureCollection, що містить правильні поля у properties (source, country, year, indicator).

Для перевірки серверної частини (Flask) застосовувалися ручні тести в Postman. Кожен запит на http://localhost:5000/static/data_cache.json та http://localhost:5000/static/data_map.geojson перевірявся на коректність структури відповіді та відсутність статусу 404 або помилок формату. Також перевірялася швидкість відповіді (орієнтовно не більше ніж 100 мс на локальному середовищі).

Фронтенд-інтерфейс тестувався переважно вручну в браузері Google Chrome та Firefox. Основні перевірки включали:

– Клавіатурна навігація між фільтрами та елементами карти.
– Коректність AJAX-запитів до data_cache.json та data_map.geojson при зміні фільтрів (відображення в консолі браузера статусу 200 або коректних даних).
– Зміна кольору регіонів на карті відповідно до вибраного показника.
– Перевірка побудови графіка динаміки: правильне відображення точок та осей.
– Функція експорту: через кнопку «Завантажити звіт» формувався файл, який відкривався в форматі PDF та містив дані про показник та країну, а ще деякі статистичні дані
– Перевірка адаптивності: елементи інтерфейсу коректно масштабувалися при зміні розміру вікна браузера.

Для автоматизації фронтенд-тестів за потреби передбачено використання Jest (для unit-тестування окремих JavaScript-функцій) та Cypress (для e2e-

тестів), але в поточному обсязі досить ручних перевірок, оскільки функціональність інтерфейсу не зазнала змін протягом розробки.

Приклад тестового сценарію (ETL-модуль)

1. Запустити etl.py без параметрів (автоматичний режим), перевірити, що після завершення з'являється файл static/data/data_cache.json.
2. Уміст файлу порівняти із контрольним JSON (невеликим набором даних з топ-3 країн за одним індикатором) – обов'язкові ключі gdp_USA_2020, gdp_CHN_2020, gdp_IND_2020, значення повинні відповідати відомим даним.
3. Запустити etl.py --auto gdp U S A 2021 (тестовий режим), перевірити оновлення лише ключів, пов'язаних із США і 2021 роком.
4. Завантажити у браузері http://localhost:5000/index.html, переконатися, що карта підсвічує США у 2021 році кольором, що відповідає показнику ВВП.

Висновки до розділу 3

У Розділі 3 описано середовище розробки, порядок встановлення та налаштування системи, а також наведено керівництво користувачу щодо роботи з інтерфейсом. Описано, як створити віртуальне середовище, інсталювати залежності (Flask, requests та ін.), запускати ETL-скрипт для формування кешу й GeoJSON, а також запускати Flask-сервер для роздачі статичних файлів. Інтерфейс складається з інтерактивної карти, графіка та таблиці, що реагують на вибір показника, країни і року, а також дозволяють експортувати дані у форматах CSV і GeoJSON. Тестування ETL-модуля виконано за допомогою pytest, перевірено правильність завантаження, обробки й агрегації даних, а також відповідність формату кешу. Сервер Flask перевірено вручну через Postman, а фронтенд протестовано в браузері (Google Chrome/Firefox) на коректне відображення карти, графіків і таблиці. Завдяки такому підходу забезпечено стабільну роботу системи, коректність даних і відповідність вимогам до продуктивності.

ВИСНОВКИ

Розробка інформаційної панелі для дослідження соціально-економічних процесів із використанням QGIS та веб-технологій відповідає сучасним потребам оперативного аналізу даних: інтеграція інформації з різних відкритих джерел (World Bank, OWID, Wikipedia) дозволила зібрати ключові показники уніфікованих форматів (XML, CSV, JSON) та забезпечити їх швидке отримання і візуалізацію. На основі аналізу предметної області визначено, що основними користувачами є дослідники, аналітики, викладачі й державні структури, які потребують порівняння даних за країнами, відстеження динаміки, інтерактивної карти та можливості експорту результатів.

Конструкторська частина передбачала створення модульної архітектури: окремий ETL-модуль на Python, що збирає, уніфікує й агрегує дані, формує кеш у вигляді плоского словника JSON і GeoJSON для карти, а також реалізує прогнозування методом ARIMA; файли кешу зберігаються з TTL, що мінімізує затримки й навантаження над джерелами; Flask-сервер відповідає лише за роздачу статичних ресурсів; фронтенд на HTML/CSS/JS (Leaflet.js, D3.js) динамічно відображає карту, графік і таблицю за допомогою AJAX-запитів.

Підсумкове тестування підтвердило коректність роботи всіх компонентів: ETL-скрипт правильно обробляє дані й формує необхідні файли, сервер стабільно роздає JSON/GeoJSON, а браузерний інтерфейс без помилок візуалізує інформацію і дозволяє експортувати її в CSV та GeoJSON. Отриманий результат – гнучка й масштабована система, яка може бути розгорнута локально або в хмарі та розширена новими показниками й джерелами для прийняття обґрунтованих управлінських рішень.

ДЖЕРЕЛА

1. Flask Documentation : веб-сайт. URL: https://flask.palletsprojects.com (дата звернення: 05.06.2025).

2. Leaflet Quick Start Guide : веб-сайт. URL: https://leafletjs.com/examples/quick-start/ (дата звернення: 09.06.2025).

3. GeoJSON Specification (RFC 7946) : веб-сайт. Open Geospatial Consortium. URL: https://datatracker.ietf.org/doc/html/rfc7946 (дата звернення: 10.06.2025).

4. World Bank Indicators API Documentation : веб-сайт. URL: https://datahelpdesk.worldbank.org/knowledgebase/articles/889392 (дата звернення: 11.05.2025).

5. COVID-19 dataset by Our World in Data : веб-сайт. URL: https://github.com/owid/covid-19-data (дата звернення: 10.06.2025).

6. Statsmodels.tsa.arima.model.ARIMA: веб-сайт. URL: https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html (дата звернення: 01.06.2025).

7. How to Create an ARIMA Model for Time Series Forecasting in Python :веб-сайт.MachineLearningMastery.URL: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/ (дата звернення: 01.06.2025).

8. Time Series Analysis Using ARIMA From StatsModels : веб-сайт. nbshare.io. URL: https://nbshare.io/notebook/63595349/time-series-analysis-using-arima/ (дата звернення: 15.04.2025).

9. Introduction to D3.js : веб-сайт. URL: https://d3js.org/ (дата звернення: 10.06.2025).

10. A Beginner's Guide to Creating a Map Using Leaflet.js : веб-сайт. SitePoint. URL: https://www.sitepoint.com/creating-simple-map-leaflet/ (дата звернення: 10.05.2025).

11. Working with Projections — QGIS Documentation : веб-сайт. URL: https://docs.qgis.org/3.28/en/docs/user_manual/working_with_projections/index.html (дата звернення: 10.06.2025).

12. Купач Т.Г., Гринюк О.Ю. Проектування геоінформаційних систем : навч. посібник. Київ: КНУ, 2021.

13. Часковський О., Андрейчук Ю., Ямелинець Т. Застосування ГІС у природоохоронній справі на прикладі відкритої програми QGIS. Львів: ЛНУ ім. І. Франка, 2021.

14. Лейберюк О. М. Інтерактивні веб-карти // Український географічний журнал, 2016. № 2. С. 34–40.

15. Непошивайленко Н. О. Основи геоінформаційних систем : навчальний курс. Кам’янське: ДДТУ, 2018.