

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Розробка аналітичної системи для дослідження
методів оптимізації швидкості завантаження веб-
сторінок»**

Виконав студент групи ПЗм-24-2
ГОСАЛО Антон Олегович

Керівник роботи:
БАКАЛЮК Тетяна Анатоліївна

Рецензент:
ГОДЛЕВСЬКИЙ Юрій Олександрович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«18» вересня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГОСАЛО Антон Олегович**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Розробка аналітичної системи для дослідження методів оптимізації швидкості завантаження веб-сторінок»,**

затверджена Наказом закладу вищої освіти від **«18» вересня 2024 р. № 491/с**

Вихідні дані для роботи: **Об'єктом дослідження є процес дослідження методів оптимізації швидкості завантаження веб-сторінок. Предметом дослідження - методи, технології та аналітична система для прискорення завантаження веб-ресурсів.**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	22.09.2025р.	22.09.2025р.
2	Вакалюк Т.А.	22.09.2025р.	22.09.2025р.
3	Вакалюк Т.А.	22.09.2025р.	22.09.2025р.

РЕФЕРАТ

Магістерська робота присвячена розробці аналітичної системи для комплексного дослідження методів оптимізації швидкості завантаження веб-сторінок. У роботі розглядаються сучасні підходи до підвищення продуктивності веб-ресурсів, зокрема їх оптимізацію структури HTML/CSS/JavaScript, алгоритмів стискання, кешування та асинхронного завантаження контенту, застосування мережних протоколів нового покоління, алгоритмів стискання. Для реалізації було використано новітні інструменти моніторингу та методи інженерії даних. Також сучасні веб-технології. Пояснювальна записка до кваліфікаційної магістерської роботи містить 76 сторінок, 26 ілюстрацій, 7 таблиць, 3 додатки та 40 посилань на зовнішні джерела.

Результатом роботи є аналітична система, яка аналізує швидкість завантаження сторінок, виявлення «слабкого місця» у продуктивності та методах їх усунення. Виконані дослідження засвідчують дієвість створеної системи у визначенні найкращих шляхів пришвидшення роботи веб-сторінок та її реальну користь для розробників і фахівців із веб-платформ. Дослідження може слугувати фундаментом для наступних пошуків.

КЛЮЧОВІ СЛОВА: АНАЛІТИЧНА СИСТЕМА, ВЕБ-ПРОДУКТИВНІСТЬ, ОПТИМІЗАЦІЯ ШВИДКОСТІ, LARGEST CONTENTFUL PAINT, TIME TO FIRST BYTE, МНОЖИННА ЛІНІЙНА РЕГРЕСІЯ, АНАЛІЗ ЧАСОВИХ РЯДІВ, АНАЛІЗ ВТРУЧАННЯ, СТАТИСТИЧНА ВЕРИФІКАЦІЯ, КОНТЕЙНЕРИЗАЦІЯ, FEATURE ENGINEERING, КОНСТРУЮВАННЯ ОЗНАК, СЕРВІСНО-ОРІЄНТОВАНА АРХІТЕКТУРА, APPLICATION PROGRAMMING INTERFACE, МУЛЬТИКОЛІНЕАРНІСТЬ, КОЕФІЦІЄНТ ДЕТЕРМІНАЦІЇ.

ANNOTATION

The master's thesis is devoted to the development of an analytical system for a comprehensive study of methods for optimizing the loading speed of web pages. The work considers modern approaches to improving the performance of web resources, in particular, their optimization of the HTML, CSS, JavaScript structure, compression algorithms, caching and asynchronous loading of content, the use of new-generation network protocols, and compression algorithms.

The latest monitoring tools and data engineering methods were used for implementation, as well as modern web technologies. The explanatory note to the master's qualification work contains 76 pages, 26 illustrations, 7 tables, 3 appendices, and 40 references to external sources. The result of the work is an analytical system that analyzes page loading speed, identifies "weak spots" in performance, and proposes methods for eliminating them. The experiments performed demonstrate the effectiveness of the created system in determining the best ways to speed up the work of web pages and its real benefit for developers and web platform specialists. The study can serve as a foundation for further research.

KEYWORDS: ANALYTICAL SYSTEM, WEB PERFORMANCE, SPEED OPTIMIZATION, LARGEST CONTENTFUL PAINT, TIME TO FIRST BYTE, MULTIPLE LINEAR REGRESSION, TIME SERIES ANALYSIS, INTERVENTION ANALYSIS, STATISTICAL VERIFICATION, CONTAINERIZATION, FEATURE ENGINEERING, CONSTRUCTING FEATURES, SERVICE-ORIENTED ARCHITECTURE, APPLICATION PROGRAMMING INTERFACE, MULTICOLLINEARITY, COEFFICIENT OF DETERMINATION.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	9
1.1. Швидкість завантаження сторінок: значення та вплив.....	9
1.2. Фактори впливу на швидкість завантаження веб-сторінок.....	12
1.3. Аналіз методів оптимізації завантаження веб-сторінок	20
1.4. Постановка задачі дослідження.....	23
Висновки до розділу 1	28
РОЗДІЛ 2. ПРОЕКТУВАННЯ АНАЛІТИЧНОЇ СИСТЕМИ ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ.....	30
2.1. Вимоги до системи та технологічне обґрунтування	30
2.2. Проектування архітектури аналітичної системи	33
2.3. Проектування бази даних та моделі даних аналітичної системи.....	37
2.4. Алгоритмічна реалізація та модуль моделювання аналітичної системи	41
2.5 Реалізація API-сервісу для обслуговування аналітичних моделей.....	45
Висновки до розділу 2	49
РОЗДІЛ 3. ЕКСПЕРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ.....	51
3.1. Аналіз та валідація моделі прогнозування базової ефективності M1 ...	51
3.2. Експериментальна верифікація моделі M2 та доведення ефективності оптимізації	56
3.3. Симуляція аналітичного контуру та експлуатаційне тестування API- сервісу.....	59
Висновки до розділу 3	64
ВИСНОВКИ.....	66
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	68
ДОДАТКИ.....	72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АС	– Аналітична система
LCP	– (англ. Largest Contentful Paint) – Ключова метрика швидкості
TTFB	– (англ. Time to First Byte) – Метрика часу відповіді сервера
API	– (англ. Application Programming Interface) – Програмний інтерфейс
CDN	– (англ. Content Delivery Network) – Мережа доставки контенту
OLS	– (англ. Ordinary Least Squares) – Метод найменших квадратів
МЛР	– Множинна лінійна регресія
ROI	– (англ. Return on Investment) – Повернення інвестицій
RMSE	– (англ. Root Mean Square Error) – Середня квадратична похибка
SOA	– (англ. Service-Oriented Architecture) – Сервісно-орієнтована архітектура
ACID	– (англ. Atomicity, Consistency, Isolation, Durability) – Атомарність, Узгодженість, Ізоляція, Довговічність

ВСТУП

Актуальність теми. Швидкість завантаження веб-сторінок є важливим і актуальним аспектом функціонування сучасних сайтів та безпосередньо впливає на досвід користувачів. Від неї залежить ефективність роботи веб-ресурсу, а також час, який користувач витрачає на надсилання запиту та очікування отримання необхідного результату.

Інформаційно-аналітична система (ІАС) — це комп'ютерна система, яка дозволяє отримувати, створювати та здійснювати обробку інформації, а вже згодом її аналізувати. ІАС дозволяє організації здійснювати інтеграцію та координацію його бізнес-процесів.

Моделювання — це спосіб дослідити необхідне, шляхом побудови й аналізу процесів. Його ціль полягає в тому, щоб розділити ці процеси на дії, які разом визначають функціональний зміст системи. У сучасному світі існує безліч різних факторів, які впливають на швидкість завантаження сторінки. У ході роботи буде розглянуто, як правильно змоделювати вплив факторів та як аналізувати отриману інформацію. Дедалі більше веб-ресурсів стикаються з проблемою повільного завантаження сторінок, що зумовлено низкою різноманітних факторів. У зв'язку з цим виникає потреба детально проаналізувати причини зниження продуктивності та ґрунтовно описати кожен метод, застосування якого дозволяє підвищити швидкість роботи сайту.

Мета роботи — описати методи оптимізації, які використовуються найчастіше, описати їх переваги та недоліки, розробити аналітичну систему для дослідження швидкості завантаження сторінок.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Дослідити вплив швидкості завантаження сторінок на ефективність веб-сайтів у пошукових системах.
2. Дослідити методи оптимізації завантаження веб-сторінок на способів їх оптимізації

3. Створити аналітичну систему для дослідження швидкості веб-сторінок та їх оптимізації шляхом вимірювання та аналізу основних метрик.

Об'єктом дослідження є процес дослідження методів оптимізації швидкості завантаження веб-сторінок.

Предметом дослідження є методи, технології та аналітична система для прискорення завантаження веб-ресурсів.

Наукова новизна роботи полягає у створенні та експериментальній валідації двоядерної аналітичної системи, що вперше інтегрує прогностичну модель (M1 для кількісної діагностичної декомпозиції LCP та статистично-верифікаційну модель M2 на основі аналізу втручання). Це забезпечує можливість не лише прогнозувати час завантаження, але й отримувати статистично значуще доведення (P-value) того, що спостережуване покращення LCP є прямим наслідком впроваджених оптимізаційних рішень.

Практичне значення полягає у впровадженні кількісного підходу до управління веб-продуктивністю, що має пряму вигоду для інженерних команд. Система надає змогу ідентифікувати «слабкі місця» з високою точністю, обґрунтовувати ROI за рахунок прогностної моделі та забезпечує надійне прийняття рішень щодо успішності оптимізації завдяки статистично верифікованим висновкам.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Швидкість завантаження веб сторінок: значення та вплив

Оптимізація веб сторінки, яка впливає на швидкість її завантаження — є ключовим процесом для покращення роботи проєкту. Під час дослідження розглядаються ефективні методи для зменшення часу завантаження й оптимізації ресурсів.

Одним із найважливіших показників продуктивності є процес завантаження сторінки (за інформацією Google, три секунди або менше). Його можна виміряти за допомогою таких інструментів як Pingdom, GTmetrix або Google PageSpeed Insights. В ході досліджень важливим фактором, який слід враховувати, є швидкість реагування сайту на дії користувача (прокрутка сторінки, додавання у кошик, завантаження елементів). Повільний TTFB може вказувати на проблеми з сервером або мережею. Важливо виміряти його за допомогою таких інструментів, як WebPageTest або KeyCDN (див. рис. 1.1.).

Швидкість завантаження веб сайту — це надважлива деталь, ключовий фактор, що впливає на SEO, користувацький досвід та, як наслідок, на успіх роботи вашого продукту. Для початку слід розібратись, що таке оптимізація і для чого вона потрібна.

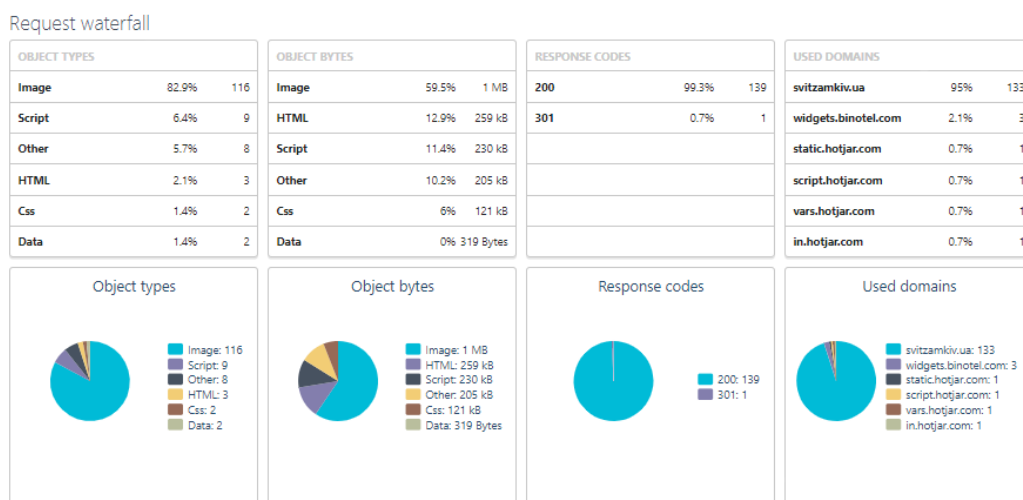


Рис. 1.1. Джерела для покращення швидкості завантаження сайту

Процес оптимізації являє собою співвідношення вигідних характеристик та мінімізації витрат. Простіше усі ті умови, які економлять ресурси, гроші та час, що вкрай важливо для будь-якого підприємства або роботи онлайн бізнесу.

Google послідовно підтверджує, що швидкість завантаження є критичним фактором ранжування у пошуковій системі, особливо після впровадження ініціативи Core Web Vitals. Сторінки, що завантажуються менш ніж за 3 секунди, забезпечують кращий користувацький досвід і, як наслідок, отримують вищий рейтинг у порівнянні з тими, що мають час завантаження понад 4 секунди та демонструють високий показник відмов . Оптимізація часу завантаження безпосередньо впливає на видимість ресурсу, оскільки Google заохочує сайти, які швидко надають користувачам необхідний контент.[1]



Рис. 1.2. SEO оптимізація веб-ресурсу (оптимізація під пошукову систему)

У цифровому світі швидкість має велике значення не тільки для рейтингу продукту, але і для покращення користувацького досвіду. Згідно з дослідженням Google, 63% мобільних користувачів залишають сторінку, завантаження якої над повільне та спричиняє дискомфорт для користувачів. [1][2] Дослідження Akamai показало, що навіть 100-

мілісекундна затримка завантаження може знизити коефіцієнт конверсії. Наслідками є зниження відвідування користувачів даного сайту, зменшення продажів (якщо платформа надає такі послуги), а також зниження рейтингу у SEO, що призводить до втрати доходу (див. рис. 1.2.).

Швидкий сайт зберігає інтерес користувача, спонукаючи його вивчати більше контенту, перебуваючи на сайті протягом довгого часу. Декілька досліджень показують, що швидке завантаження призводить до підвищення залучення користувачів. Такі відносини часто асоціюють як симбіотичні, адже чим довше користувач на сайті, тим вище показники цінності даного сайту для Google.

Мобільні пристрої більш чутливі до повільного завантаження сторінки в порівнянні з персональними комп'ютерами. Пропускна здатність мережі та обчислювальна потужність мобільних платформ часто є обмеженими. Це означає, що час завантаження відіграє критично важливу роль у формуванні якісного досвіду мобільних користувачів, безпосередньо впливаючи на їхнє рішення залишатися на сайті.

Хоча традиційний час повного завантаження залишається важливим, Google застосовує комплексний підхід, враховуючи низку елементів, включаючи перше відображення контенту (FCP), що фіксує початок візуального завантаження, а також час до інтерактивності (TTI) та інші показники Core Web Vitals, щоб максимально точно оцінити сприйняту швидкість та чутливість сайту до дій користувача.

FCP - це метрика Core Web Vitals, яка вимірює час, період, що проходить від моменту, коли користувач починає завантажувати веб-сторінку, до моменту, коли будь-яка частина вмісту сторінки відображається на екрані. Швидкий FCP покращує користувацький досвід, зменшуючи ймовірність того, що користувачі залишать сторінку, адже забезпечуючи миттєву видимість контенту. Вплив FCP на завантаження сторінки показано на рис. 1.3.

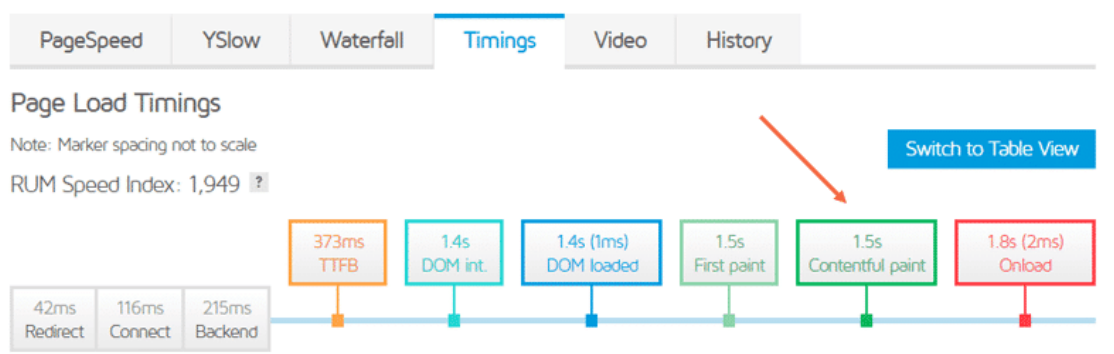


Рис. 1.3. First Contentful Paint

TTI — час, який потрібно витратити сайту, щоб стати повністю інтерактивним для користувача. Усі сайти прагнуть підвищити ці показники, щоб бути зручними та корисними для тих, хто їх використовує. Хоча Google як і раніше не розкриває своїх секретів, багато досліджень спрямовані на розуміння цих показників. Наприклад, TTI, що менший за 3,8 секунди та FCP менший ніж 1,9 секунди зазвичай вважаються хорошими для веб-сайту.[3]

1.2. Фактори впливу на швидкість завантаження веб-сторінок

Дуже важливо зрозуміти які фактори впливають на швидкість завантаження веб-сторінок та сприяти їх покращенню. Швидкість завантаження веб-сторінки визначається комплексом чинників, що охоплюють оптимізацію на стороні сервера (мінімізація TTFB, ефективність роботи бази даних та конфігурації хостингу), оптимізацію клієнтської частини (зменшення обсягу коду шляхом мініфікації та стиснення, використання сучасних форматів і лінивого завантаження зображень), а також мережеву архітектуру (впровадження CDN для скорочення затримок передачі даних). Своєчасна оптимізація цих компонентів є критично важливою для забезпечення високої користувацької взаємодії та покращення ранжування ресурсу пошуковими системами.

Якість хостингу має величезний вплив на швидкість завантаження сайту. Якщо сервер потужний, а підключення до мережі стабільне, користувачі

отримують швидкі відповіді, а ресурс доступний без перебоїв. Навіть за умови бездоганної оптимізації вебсайту ненадійний хостинг здатен нівелювати всі прикладені зусилля. Тому вкрай важливо систематично контролювати стабільність роботи сервера, своєчасно виявляти можливі простой та оцінювати рівень затримок під час обробки запитів.

У контексті розробки та оптимізації веб-систем, суттєва диференціація між хостинговими рішеннями є аксіомою. Ці відмінності не обмежуються лише ціновою політикою та моделлю оплати, але й охоплюють архітектурні, операційні та експлуатаційні характеристики.

Зокрема, ключові розбіжності полягають у механізмах ізоляції ресурсів, рівнях контролю адміністратора та параметрах динамічної масштабованості. Ці фактори критично впливають на кінцеву продуктивність веб-застосунку, його відмовостійкість та здатність адаптуватися до непередбачуваних навантажень, що є вирішальним при виборі оптимальної інфраструктури для конкретного проекту. Хостингові рішення класифікуються за кількома основними типами, кожен з яких пропонує унікальний компроміс між контролем, продуктивністю, безпекою та вартістю володіння. [4]

Це розуміння є фундаментальним для інженера програмного забезпечення, оскільки вибір типу хостингу безпосередньо корелює з такими показниками, як TTFB та надійність системи під високим навантаженням. У контексті архітектури веб-застосувань та оптимізації продуктивності, класифікація хостингових рішень є фундаментальною. Ці рішення диференціюються за принципами виділення ресурсів, рівнем ізоляції та контролю, що безпосередньо впливає на ключові метрики швидкості, зокрема TTFB.

Існує чотири основних категорії хостингу. Нижче наведена коротка характеристика кожного з них. Спільний хостинг - це найбільш базове та економічно вигідне рішення, що ідеально підходить для невеликих сайтів. Ресурси одного фізичного сервера динамічно розподіляються між множиною незалежних клієнтів. Його основний недолік полягає у відсутності

гарантованої ізоляції, що призводить до високої чутливості до явища "шумного сусіда" (англ. noisy neighbor), де пікове навантаження від одного користувача може непередбачувано деградувати продуктивність усіх розміщених застосунків.

VPS або віртуальний приватний сервер забезпечує значно кращу ізоляцію та стабільність завдяки поділу фізичного сервера на ізольовані віртуальні машини з гарантованим виділенням ресурсів. Він надає користувачеві повний адміністративний доступ, що дозволяє гнучко конфігурувати операційну систему та серверне програмне забезпечення (наприклад, встановлювати системи кешування та стиснення).

Виділений сервер - це підхід, що передбачає оренду цілісного фізичного сервера одним клієнтом, забезпечуючи максимальний рівень контролю та безкомпромісну продуктивність. Оскільки клієнт має ексклюзивне право на всі апаратні ресурси, гарантується повна відсутність конкуренції.

Хмарний хостинг являє собою вискоелестичну та стійку архітектуру, побудовану на кластері взаємопов'язаних серверів. Його ключовою перевагою є автоматичне динамічне масштабування ресурсів: інфраструктура може миттєво додавати або вивільняти обчислювальні потужності. Хмарні рішення мінімізують ризик відмов, забезпечуючи високу доступність та стабільну швидкість завантаження навіть в умовах непередбачуваних пікових навантажень. [5][6]

Виділений сервер - це рішення, яке набуло поширення у середині 1990-х років. Його поява була зумовлена необхідністю забезпечення надійніших і потужніших платформ для розміщення зростаючих корпоративних застосунків та розв'язання численних проблем, пов'язаних із масштабуванням. Відтоді виділений сервер став загальновизнаним як оптимальне рішення для підприємств та організацій, критично залежних від високої продуктивності, безкомпромісної надійності, значної обчислювальної потужності та гарантованого рівня безпеки. Виділений сервер зображено на рис. 1.4.



Рис. 1.4. Виділений сервер

Сервери зазвичай характеризують досить розгорнуто, оскільки вони відіграють ключову роль у функціонуванні інформаційних систем і веб-додатків. Наприклад, кожен виділений сервер є фізично ізольованим від інших, що забезпечує високу стабільність роботи, передбачуваність продуктивності та безпеку збережених даних. Завдяки такій ізоляції виключається вплив сусідніх користувачів або процесів, як це може відбуватися у віртуалізованих середовищах

Спільний хостинг - це варіант, який передбачає розміщення множини незалежних веб-ресурсів на одному фізичному сервері. Кожен клієнт може розпоряджатись спільними ресурсами сервера, що включають центральний процесор, оперативну пам'ять та дисковий простір. Всі ресурси сервера використовуються спільно всіма ресурсами, розміщеними на одному фізичному сервері. Схема роботи спільного хостингу зображена на рис. 1.5.

Спільний хостинг категорично не рекомендується для функціонування великомасштабних проєктів та систем із високими вимогами до надійності, проте він є найбільш економічно ефективним рішенням.

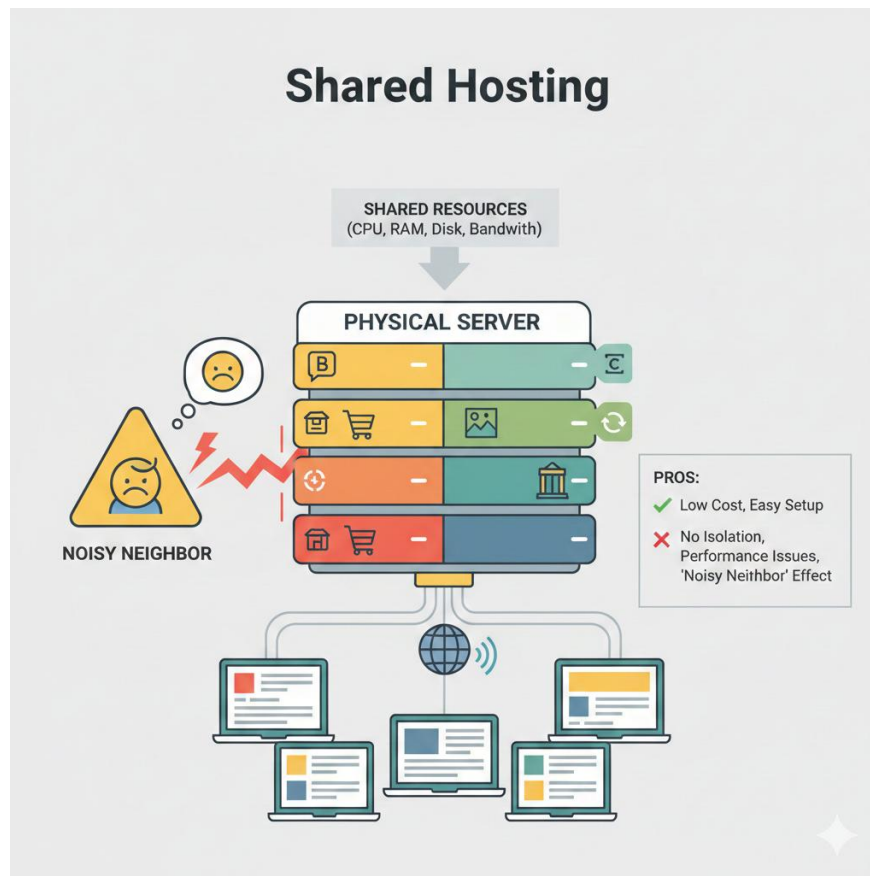


Рис. 1.5. Схематичне зображення Shared хостингу

Водночас, цей варіант є більш доступним і дозволяє скоротити витрати на інфраструктуру веб-сайту та її адміністрування. Також можна вибрати кількість веб-сайтів, які потрібно розмістити. Обслуговувати та керувати ними одночасно завдяки вбудованій панелі керування.

При розподіленні одного серверу для багатьох користувачів, завжди є вірогідність, один з користувачів почне використовувати більше ресурсів, що може вплинути на швидкість роботи всіх веб-сайтів на сервері. Однак цей ризик мінімізується в тарифних планах хостингу.

VPS (віртуальний приватний сервер) пропонує більше ресурсів та гнучкість ніж спільний хостинг. VPS — це хостинг, який імітує виділений сервер, який підходить для середніх сайтів з трафіком, що постійно зростає. Суть полягає у тому, що працює відразу кілька віртуальних серверів на одному фізичному носії, але вони ізольовані один від одного. Схематичне зображення VPS хостингу зображено на рис. 1.6.

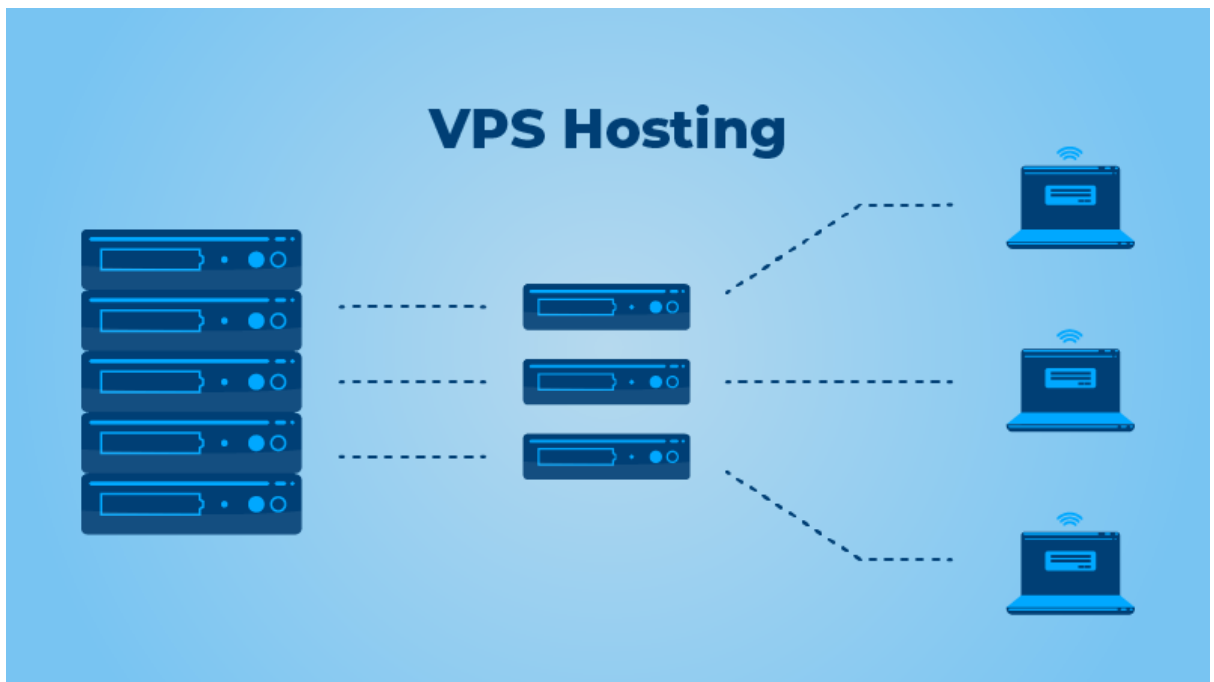


Рис. 1.6. Схематичне зображення VPS хостингу

VPS-хостинг можна розглядати як альтернативу, коли можливостей віртуального хостингу може бути недостатньо, а ресурсів цілого сервера забагато. Розбираємось, які бувають VPS-сервери та як вибрати відповідний для свого проєкту.

Приклад віртуального приватного сервера: користувач хоче розгорнути власний сайт, але придбати дорогий фізичний сервер немає можливості. Тоді найкращим рішенням буде орендувати VPS у хостинг-провайдера. Він має власну операційну систему, ресурси та IP-адресу. Також чудовий вибір для ігрової сфери, хостингу веб-сайтів, систем аналітики та баз даних.

Хмарний хостинг, на перший погляд, складний в користуванні, адже передбачає запуск веб-сайту або застосунку на кількох серверах одночасно, вони розміщені у хмарі на потужностях провайдера. В основі лежить система віртуалізації (може бути не одна) та апаратне забезпечення, які дають змогу створювати та запускати велику кількість віртуальних машин на одному фізичному сервері. Ізольованість віртуальних машин значний плюс, який робить можливим відокремлення застосунків, щоб вони не заважали роботі та не використовували віртуальні ресурси один одного. Схема роботи хмарного хостингу зображена на рис. 1.7.



Рис. 1.7. Моделі обслуговування хмарних обчислень

Значною перевагою є можливість клієнта отримати у свої руки оренду фізичних серверів, отримати бажаний рівень продуктивності та безпеки. Тут можна вибирати індивідуальну конфігурацію обладнання та взагалі не перейматись розподілом обладнання з іншими клієнтами. Вартість такої послуги, відповідно, теж вища порівняно з традиційним хмарним хостингом.

Переваги хмарного хостингу:

1. Доступна виділена IP-адреса, оплата залежно від обсягу споживаних ресурсів, збільшено час безвідмовної роботи;
2. Більше гнучкості у виділенні ресурсів — їх надає не один сервер, а кластер серверів;
3. Просто масштабувати — ресурси можна регулювати, коли змінюється навантаження.

Клієнт не переймається, де розташовані сервери, не турбується про їхню доступність, працездатність, чи достатньо для нього ресурсів, якщо раптом зросте навантаження.[5][7]

Слід перейти до важливої частини, а саме які ж фактори впливають на швидкість завантаження веб-сторінок, сайтів та інших платформ. Варто зазначити, що тема надто широка, тому тут зібрані найпоширеніші фактори, їх особливості та підхід до кожного з них.

Будь-який ресурс складається зі змістовної частини, зображення, скриптів та текстової інформації, може включати й інші елементи. Все це є складовими сторінок в браузері в єдину сторінку в браузері. Коли користувач переходить на сайт або платформу, завантажуються різні елементи сторінки, так звані її складові про які йшлося вище. Цей процес займає певний час, залежно від кількості контенту та елементів сайту. Відмінність у відображенні оптимізованої та неоптимізованої сторінки наведено на рис. 1.8.

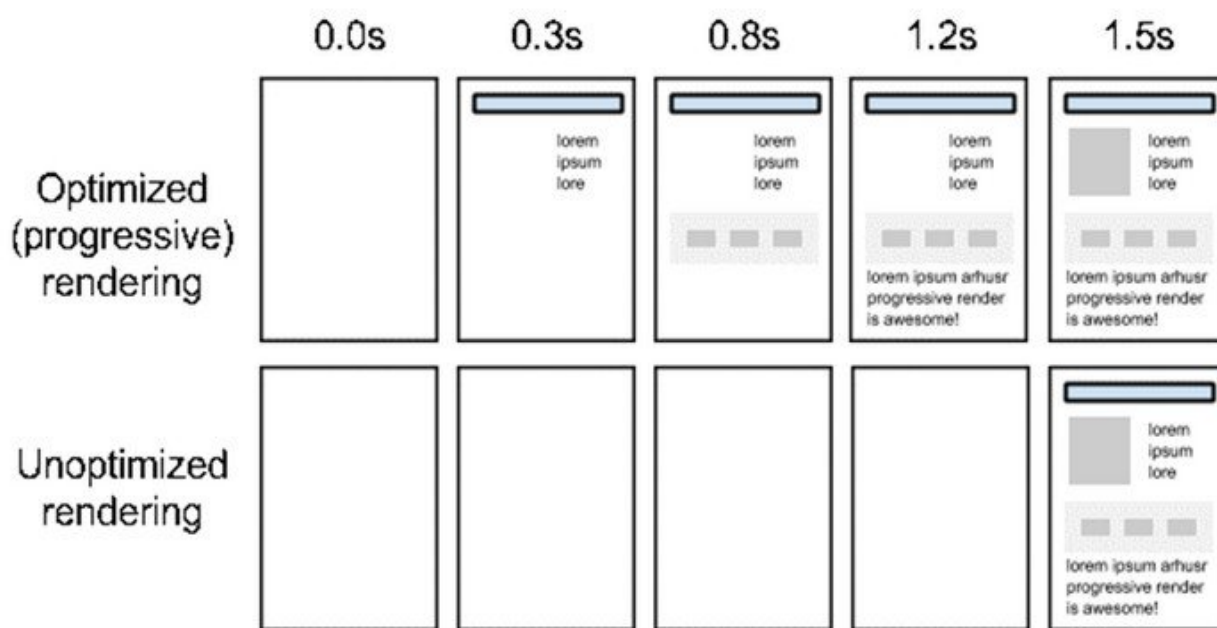


Рис. 1.8. Порівняння завантаження оптимізованої та неоптимізованої сторінки

У якості прикладу, для пояснення відмінностей стратегії оптимізації, візьмемо два типи сайтів. Статичний сайт - візитка або реклама, яка містить в собі меншу кількість картинок, тексту, символів, тому й швидкість завантаження буде приблизно до 1-2 секунди. Насправді такі сайти також можуть завантажуватися в межах 2-3 секунд, що є стандартом завантаження сторінок. Сайт з динамічним вмістом, наприклад інтернет магазин, каталоги товарів, та багато інших платформ які мають більшу кількість картинок, інформації. Зрозуміло, що в такому випадку завантаження триватиме довше, але є певна норма, оптимальний час її завантаження (за інформацією Google, три секунди або менше).[8]

1.3. Аналіз методів оптимізації завантаження веб-сторінок

Швидкість завантаження контенту є важливим аспектом, визначається швидкістю отримання зворотної відповіді від сервера користувачу. Важливо, щоб інтернет з'єднання було швидким. Це залежить від багатьох факторів, таких як хостинг, а також веб-сервер і його характеристики.

Швидкість відповіді сервера — цей час затримки між відправленням сигналу користувачем і отриманням цього сигналу назад від сайту. Враховується час затримки, яке вимірюється в мілісекундах.

Найбільше на швидкість відповіді сервера впливають віддаленість сервера, про який раніше згадувалося, і його налаштування. Для того, щоб оптимізувати цей процес, використовують технологію CDN.

Неоптимізовані зображення та відео, відсутність кешування браузера, надмірне використання сторонніх скриптів і плагінів або погано написаний код — поширені проблеми з продуктивністю, вирішення яких може допомогти підвищити швидкість роботи продукту та покращити взаємодію з користувачем.[9]

Методи оптимізації, які прискорять завантаження вебсторінки:

1. Оптимізація серверної частини сайту для зменшення навантаження
2. Кешування статичних ресурсів та зображень
3. Оптимізація зображень
4. Зменшення кількості запитів до сервера
5. Ліниве та асинхронне завантаження ресурсів
6. Мінімізація та стиснення файлів стилів та скриптів
7. Використання мереж доставлення контенту (CDN)

Навантаження на сервер можна пояснити на прикладі сайту новин, адже такі сайти часто мають велику кількість відвідувачів. Це пов'язано з регулярними змінами контенту, наприклад, ситуація на фронті, що може відбуватися щодня, або щогодини. Водночас деяка інформація залишається незмінною, наприклад, зображення розділів, банери, логотипи.

Категорії новин (політика, економіка, фінанси, мода, їжа) зазвичай доступні за постійними адресами, і кожен запит до сервера для їхнього відображення створює додаткове навантаження. Якщо сайт має від 100000 до 500000 відвідувачів на день і містить велику кількість зображень, це може суттєво впливати на продуктивність сервера. Крім зображень, є й інші ресурси, що використовують потужність серверу. При великій кількості запитів сервер може не впоратися з навантаженням.

Щоб зменшити кількість запитів до бази даних, можна використовувати кешування. Це дозволяє зберігати вже сформовані відповіді й замість кількох повторюваних запитів виконувати лише один.

Браузерне кешування — це технологія, за допомогою якої зберігаються часто використовувані файли. Кешування мультимедійного контенту або запитів до API є прикладом методу, який зберігає дані, до яких часто звертаються. Це може значно скоротити час відгуку сервера і підвищити продуктивність. [14]

Метод оптимізації зображень включає використання правильного формату файлу для кожного типу зображень, їх стиснення, зміну розміру до відповідних параметрів, використання адаптивних зображень. Популярні інструменти для стиснення зображень, які допоможуть покращити роботу проєкту: TinyPNG, ImageOptim і Adobe Photoshop.[10]

Об'єднавши кілька файлів в один, можна зменшити кількість запитів до сервера, необхідних для завантаження. Наприклад, ви можете об'єднати декілька файлів які мають один формат. Стиснення запитів є використання мініфікації коду, коли видаляються зайві пробіли, коментарі та незначущі символи з файлів CSS, JavaScript або HTML.

Ліниве завантаження дозволяє зберегти зображення та медіа ресурси лише тоді, коли вони стають видимими, зменшуючи обсяг даних для початкового завантаження сторінки. Це покращує швидкість і користувацький досвід, але потребує обережного впровадження через можливі проблеми з доступністю та SEO.

Асинхронне завантаження забезпечує одночасне завантаження ресурсів, таких як сторонні скрипти та стилі, без блокування відображення сторінки, що значно пришвидшує її початкове відображення. Цей механізм є критично важливим для покращення сприйнятої користувачем швидкості, оскільки запобігає ситуації, коли браузер вимушений чекати повного завантаження та виконання великого файлу JavaScript, перш ніж перейти до обробки решти HTML.[11][12]

Для реалізації асинхронного завантаження найчастіше використовують атрибут `async` у тезі HTML5 `<script>`. Коли браузер зустрічає скрипт з атрибутом `async`, він починає його завантаження у фоновому режимі, не зупиняючи процес відображення сторінки. Після завершення завантаження, скрипт негайно виконується, не дотримуючись порядку, у якому він був розміщений у документі.

Слід зазначити, що, хоча `async` забезпечує максимальне прискорення, він підходить лише для скриптів, які є незалежними від послідовності виконання чи структури сторінки. Для скриптів, які повинні виконуватися послідовно або залежать від початкової обробки HTML, рекомендується використовувати атрибут `defer`.

Мережі доставляння контенту (CDN) являє собою географічно розподілену мережеву інфраструктуру, що дозволяє оптимізувати відправлення та розповсюдження контенту кінцевим користувачам в мережі Інтернет. Використовуючи CDN, власники можуть скоротити час завантаження їхнього вмісту, в результаті веб-сайт стає зручнішим у роботі.[14][25]

Мережі доставки вмісту (CDN) функціонують на основі розподіленого архітектурного підходу, що передбачає стратегічне кешування копій статичного та динамічного веб-вмісту на географічно рознесених серверах периферійних вузлів. [13] Схема роботи CDN проілюстрована на рис. 1.9.

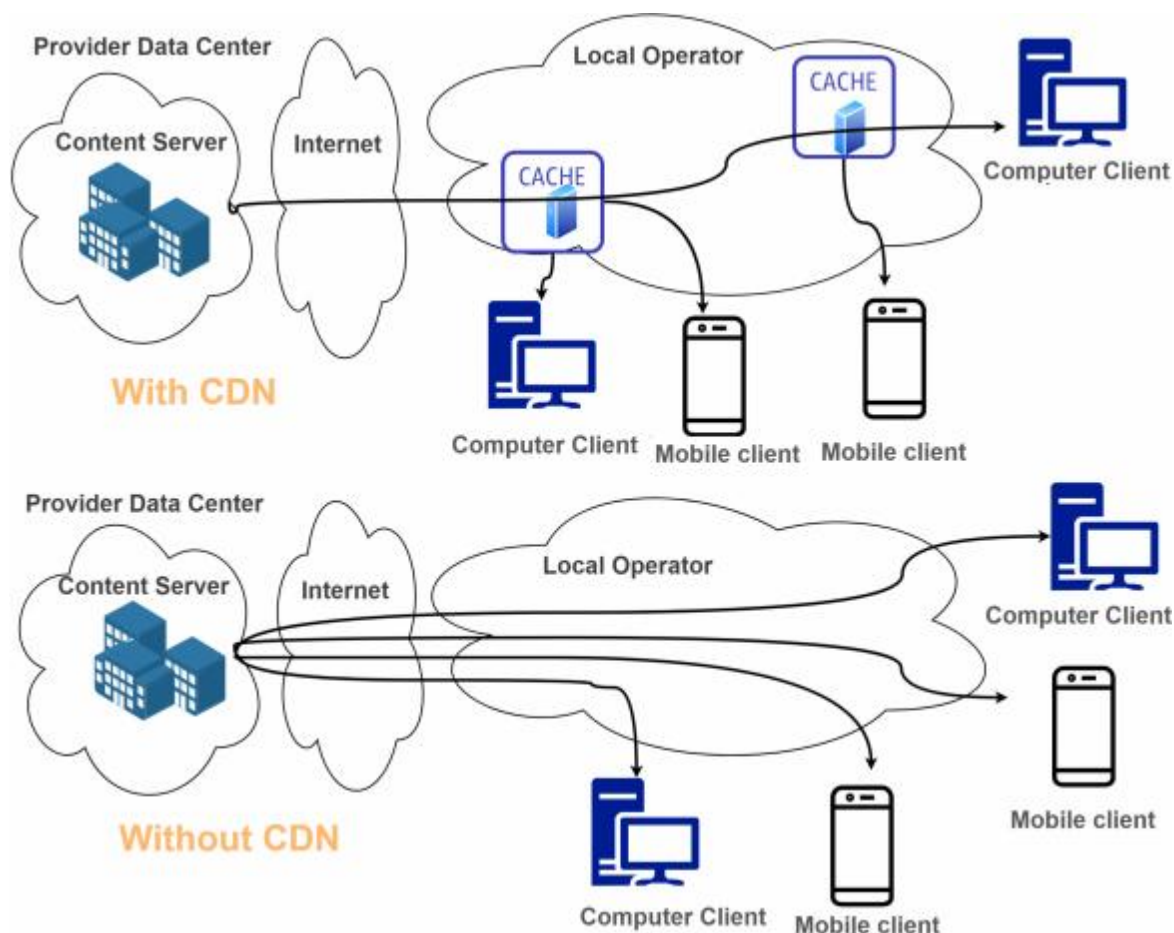


Рис. 1.9. Схема, що показує принцип роботи CDN

1.4. Постановка задачі дослідження

Інформаційно-аналітична система (ІАС) — це комп’ютерна система, яка дозволяє отримувати, створювати та здійснювати обробку інформації, а вже згодом її аналізувати. Також ІАС дозволяє організації здійснювати інтеграцію та координацію його бізнес-процесів.

Еволюція веб-технологій та постійне зростання вимог до користувацького досвіду висувують на передній план проблему оптимізації швидкості завантаження веб-застосунків. У сучасній парадигмі пошукових систем та електронної комерції, де кожна мілісекунда конвертується у фінансові та репутаційні втрати, необхідність кількісного, прогностичного аналізу факторів продуктивності стає академічною та практичною необхідністю

Традиційні інформаційно-аналітичні системи зосереджені переважно на інтеграції та обробці внутрішніх бізнес-процесів. Однак для ефективного управління якістю веб-сервісу потрібна спеціалізована ІАС, що здатна здійснювати системний аналіз зовнішніх факторів, які впливають на швидкість відтворення контенту у браузері кінцевого користувача.

Цей процес вимагає формалізованого моделювання та емпіричної верифікації взаємозв'язків між технічними параметрами та сприйнятою користувачем швидкістю. Процес моделювання починається з декомпозиції складної системи завантаження веб-сторінки на основні взаємодіючі домени, вплив яких на фінальну метрику швидкості має бути ізольований та кількісно вимірний. Дослідження виділяє три ключові категорії факторів, що формують загальний час завантаження.

Фактори клієнтської сторони мають значний вплив на сприйняту швидкість та інтерактивність. Ці параметри сильно залежать від апаратних характеристик пристрою користувача (потужність CPU, обсяг RAM), швидкості його інтернет-з'єднання (наприклад, 3G проти Wi-Fi) та рівня оптимізації статичних ресурсів. Основні параметри включають оптимізацію зображень, мінімізацію HTML/CSS/JavaScript, ефективність кешування на рівні браузера та асинхронне завантаження критичних ресурсів.[15]

Конфігурація серверної сторони є визначальною для метрики TTFB. Метрика відображає час, який витрачає сервер на обробку запиту, виконання обчислень, взаємодію з базою даних та генерацію фінальної відповіді. Продуктивність серверної сторони може бути деградована через перевантаженість сервера, неефективні алгоритми обробки запитів, проблеми з індексацією бази даних, а також неоптимальний вибір хостингового рішення (як було зазначено у розділі 1.2).

Мережеві аспекти охоплюють фактори, що лежать між клієнтом та сервером. Вони включають пропускну здатність, що лімітує швидкість передачі великих обсягів даних, та мережеву затримку, яка зростає через географічну віддаленість серверів або перевантаження мережевих вузлів.

Втрата пакетів, що вимагає повторної передачі даних, також суттєво уповільнює процес.

Для об'єктивної оцінки впливу зазначених факторів було обрано метрику LCP, оскільки вона є ключовим показником Core Web Vitals, що найкраще відображає момент, коли основний контент сторінки стає видимим користувачеві. Основна задача дослідження полягає у розробці та верифікації спеціалізованої аналітичної системи, що здатна кількісно оцінювати та прогнозувати вплив структурних та оптимізаційних факторів на LCP шляхом реалізації системи верифікованих регресійних моделей.

Для досягнення цієї мети необхідно розробити та навчити модель прогнозування базової ефективності (далі M1) та модель аналізу ефективності оптимізації (далі M2). Також потрібно розробити API-інтерфейс для взаємодії з моделями, для ефективного їх використання в будь-яких програмних продуктах, побудованих на їх основі.

Модель M1 є основою для розуміння неоптимізованої поведінки системи і забезпечує кількісне визначення залежності LCP від основних структурних параметрів. Вона має стати прогностичним інструментом ІАС. Принцип роботи моделі M1 наведений формулою 1.1.

$$LCP_{\text{прогноз}} = \gamma_0 + \gamma_1 \cdot TTFB + \gamma_2 \cdot \text{TotalSize} + \gamma_3 \cdot \text{TotalRequests} + \varepsilon \quad (1.1)$$

Систематизація ключових вхідних параметрів, що були використані для навчання моделі M1 представлена в таблиці 1.1. Ці параметри є незалежними змінними, які відображають структурні та мережеві фактори впливу на швидкість завантаження веб-сторінки.

Модель M1 є основою для розуміння неоптимізованої поведінки системи і забезпечує кількісне визначення залежності LCP від цих основних структурних параметрів.[16][17]

Таблиця 1.1

Пояснення вхідних даних для навчання моделі M1

Фактор впливу	Кількісний показник	Домен	Роль у моделі
Серверний час відповіді	TTFB (мс)	Back-end/Мережа	Критичний предиктор LCP
Складність сторінки	TotalRequests (шт.)	Клієнтська/Мережа	Вимірює необхідність у великій кількості мережових з'єднань
Вага сторінки	TotalSize (байти)	Клієнтська	Вимірює обсяг даних для передачі

Таблиця 1.2

Пояснення вхідних даних для навчання моделі M2

Позначення	Кількісне значення	Інтерпретація
LCPфакт	Фактичний LCP (мс)	Виміряна продуктивність сторінки.
δ_0	Числове значення (наприклад, 813.78 мс)	Середнє значення LCP до моменту втручання (коли DOpt=0).
<i>D-opt</i>	0 або 1	Індикатор втручання: 0 — період до впровадження оптимізації; 1 — період після впровадження оптимізації.
δ_1	Числове значення (наприклад, -59.91 мс)	Вимірює абсолютну зміну середнього LCP після втручання. Це є кількісною оцінкою ефективності оптимізації.

Моделі M2 є інструментом статистичного доведення ефективності оптимізаційних заходів і відображає практичну значущість роботи. Вона використовує методологію аналізу впливу часових рядів. Це дозволяє

встановити причинно-наслідковий зв'язок (каузальність) між втручанням (оптимізацією) та покращенням метрики.

Модель M2 реалізується як проста лінійна регресія, де єдиною незалежною змінною є бінарна (діпазонна) змінна, що фіксує момент часу. Загальна моделі M2 формула представлена під номером 1.2.

$$LCP_{\text{факт}} = \delta_0 + \delta_1 \cdot D_{\text{opt}} + \varepsilon \quad (1.2)$$

Навчання моделі M2 здійснюється за методом найменших квадратів (OLS) на повному наборі даних. Процес включає ідентифікацію точки втручання, генерацію діапазонної змінної, а також калібрування моделі. Необхідно точно визначити дату (T_{opt}), коли було розгорнуто пакет оптимізаційних змін. Кожен рядок даних отримує значення D_{opt} для визначення чи призвели зміни до покращення показників швидкості. Далі запускається OLS-регресія для обчислення δ_0 та δ_1 .

$$D_{\text{opt}} = \begin{cases} 1, & \text{якщо Date} \geq T_{\text{opt}} \\ 0, & \text{якщо Date} < T_{\text{opt}} \end{cases} \quad (1.3)$$

Інтеграція та спільне функціонування моделей M1 та M2 являють собою ядро АС, забезпечуючи повноцінний аналітичний контур управління продуктивністю: від кількісної діагностики та прогнозного моделювання до емпіричної верифікації впроваджених рішень. На першому етапі, модель M1 (прогностична), що базується на множинній лінійній регресії, функціонує як систематичний діагност і предиктивний симулятор. [18]

Модель M2 (верифікаційна), що використовує OLS-регресію з діапазонною змінною, забезпечує емпіричне доведення досягнутих результатів. Цей етап є постфактум аналізом, який проводиться після успішного розгортання оптимізаційних змін T_{opt} .

Висновки до розділу 1

В ході роботи над першим розділом кваліфікаційної роботи можна зробити висновки стосовно важливості теми та її вплив на роботу веб-сторінок та веб-платформ загалом. Висока швидкість не лише покращує взаємодію з користувачами та забезпечує кращий користувацький досвід, але й позитивно впливає на рейтинг ресурсу в пошукових системах.

Узагальнення знань дозволило систематизувати основні домени впливу, які були розділені на клієнтську сторону (front-end), серверну сторону (back-end) та мережеві аспекти. Розглянуті сучасні методи оптимізації, такі як налаштування браузерного кешування, яке зберігає копії ресурсів на пристрої користувача для прискорення повторних візитів, оптимізація зображень через стиснення та вибір ефективних форматів webp, використання мереж доставки контенту CDN для зменшення відстані до сервера та застосування лінивого завантаження lazy loading, що знижує початковий час завантаження.

Незважаючи на важливість якісного аналізу цих методів, для цілей інженерії програмного забезпечення та управління проєктами необхідний прогностичний та кількісний інструментарій. Тому, основна постановка задачі дослідження, що формує предмет наступних розділів, полягає у розробці аналітичної системи, яка може кількісно оцінювати та прогнозувати вплив структурних і оптимізаційних факторів на ключову метрику LCP. Як емпірична база для навчання моделей було використано масив даних, що містить історичні вимірювання показників швидкості.

Для вирішення цієї задачі та забезпечення наукової новизни було запропоновано дві взаємодоповнюючі моделі регресійного аналізу, доступ до яких буде реалізовано через API. Перша — модель множинної лінійної регресії призначена для прогнозування LCP та діагностики першопричин. Ця модель використовує такі незалежні змінні, як час до першого байту TTFB, що відображає ефективність серверної сторони та мережі; загальний розмір сторінки, що корелює з необхідною пропускнуою здатністю; та загальна

кількість запитів, що вказує на структурну складність. На основі цих параметрів модель прогнозує LCP та, завдяки своїм коефіцієнтам, дозволяє кількісно визначити, який із факторів має найбільший внесок у загальний час завантаження.

Друга — модель аналізу втручання, яка призначена для статистичної верифікації досягнутої ефективності. Вона використовує LCP та діапазонну змінну, що позначає момент впровадження оптимізації. Ця модель статистично доводить, що покращення швидкості є прямим і значущим наслідком втручання, а не випадковим коливанням. Інтегрована робота цих моделей дозволить не лише спрогнозувати LCP (за допомогою першої моделі), але й статистично довести (за допомогою другої моделі), що оптимізація призвела до значущого зниження часу завантаження, тим самим підтверджуючи практичну значущість виконаної роботи для залучення нових користувачів та збільшення доходу.

РОЗДІЛ 2. ПРОЄКТУВАННЯ АНАЛІТИЧНОЇ СИСТЕМИ ДЛЯ ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ

2.1. Вимоги до системи та технологічне обґрунтування

Проектування АС є центральною фазою кваліфікаційної роботи, що забезпечує трансляцію теоретичних моделей у функціональний інженерний продукт. Визначення вимог та обґрунтування технологічного вибору є критично важливим для забезпечення наукової достовірності та практичної експлуатаційної придатності системи.

Функціональні вимоги АС орієнтовані на створення надійного контуру обробки даних, що охоплює весь ланцюжок: від джерела інформації до генерації рекомендацій. Ключовою вимогою є забезпечення автоматичного, періодичного збору метрик продуктивності.

Модуль збору даних повинен підтримувати інтеграцію через API з авторитетними зовнішніми джерелами, такими як Google Lighthouse або GTmetrix, для отримання всіх необхідних незалежних змінних та залежної змінної. Вимоги до передобробки включають валідацію вхідних даних на предмет аномалій, стандартизацію одиниць виміру та генерацію ключової бінарної змінної D-opt для моделі M2 шляхом зіставлення вимірювань із заданою датою втручання, що визначає період "до" і "після" оптимізації.[19]

Центральною вимогою є реалізація моделі M1 та моделі M2 як незалежних, високопродуктивних сервісів, доступ до яких здійснюється через API. Модуль моделювання повинен забезпечувати функцію перенавчання (re-training) моделі M1 на оновленому історичному масиві даних для підтримання актуальності коефіцієнтів γ . API моделі M1 має обробляти запити на прогноз, повертаючи не лише прогнозований LCP, але й діагностичне розкладання впливу коефіцієнтів γ на загальний час завантаження. Опис основних технологій для моделювання наведено в таблиці 2.1.

Таблиця 2.1

Технології та обґрунтування їх вибору

Компонент	Технологія	Обґрунтування вибору
Мова програмування	Python	Де-факто стандарт у Data Science. Необхідний для використання потужних бібліотек для статистичного моделювання та аналізу часових рядів.
Моделювання та аналіз	Scikit-learn	Ідеально підходить для реалізації M1 (лінійна регресія, train_test_split та оцінки якості моделі RMSE, R^2).
	Statsmodels	Критично важливий для реалізації M2 (OLS), оскільки надає деталізований статистичний вивід (зокрема P-Values та t-статистики), необхідний для наукової верифікації.
API сервіс	Flask/FastAPI	Легковагові Python-фреймворки, що дозволяють швидко створити API - інтерфейс для обслуговування запитів до навчених моделей, забезпечуючи низьку latency.
СУБД (база даних)	PostgreSQL	Надійна, ACID - сумісна реляційна СУБД. Має відмінні можливості для роботи з часовими рядами та забезпечує цілісність наукових даних.
Контейнеризація	Docker	Необхідний для пакування середовища навчання та залежностей, забезпечуючи повну відтворюваність результатів, що є вимогою академічної доброчесності.

Паралельно, модель M2, що реалізує статистичний аналіз втручання (OLS), зобов'язана обчислювати коефіцієнт впливу дельта-один та його P-value для статистичного доведення ефективності оптимізаційних втручань. Кінцевий модуль звітності повинен генерувати графічні звіти, які включають

порівняння фактичних та прогнозованих значень LCP для оцінки R-квадрат моделі M1, а також візуалізацію зміщення середнього LCP до і після точки втручання для моделі M2.

Нефункціональні вимоги визначають якість, надійність та експлуатаційну придатність системи, що є ключовим для інженерії програмного забезпечення. Вимоги до продуктивності та масштабованості вимагають, щоб час відповіді API моделі M1 на прогнозний запит не перевищував 50 мс, забезпечуючи швидку діагностику в режимі реального часу. Крім того, архітектура повинна бути здатна до горизонтального масштабування, особливо в частині зберігання великих обсягів часових рядів та обробки паралельних запитів на прогноз. [20]

Для ефективної роботи з історичними даними, що надходять у вигляді часових рядів вимірювань швидкості, необхідне використання бази даних, архітектура якої оптимізована саме для зберігання та високошвидкісної обробки даних, індексованих за часом. Традиційні реляційні СУБД демонструють значне падіння продуктивності при виконанні складних запитів до великих обсягів послідовних даних. Вибір спеціалізованої бази даних, наприклад, PostgreSQL з відповідними розширеннями або повноцінної бази даних часових рядів (наприклад, InfluxDB), дозволяє не лише забезпечити цілісність наукової вибірки, але й значно прискорити такі операції, як агрегація, інтерполяція та сегментування даних.

Найважливішою вимогою для академічної роботи є цілісність та відтворюваність даних. Значення коефіцієнтів гамма та коефіцієнтів повинні зберігатися в базі даних після кожного навчання, бути незмінними та аудитованими, забезпечуючи можливість точного відтворення наукових висновків. Усі залежності та середовища для навчання моделей повинні бути контейнеризовані (за допомогою Docker), щоб будь-який інший дослідник міг точно відтворити середовище і отримати ідентичні коефіцієнти.

Надійність зберігання даних вимагає, щоб дані зберігалися з дотриманням принципів ACID для запобігання їх втраті або пошкодженню.

Додаткові вимоги включають необхідність використання захищених ключів для зовнішніх API, які зберігаються у захищених змінних середовища, а також наявність механізмів retry та logging для обробки тимчасових відмов зовнішніх служб збору даних.

Вибір технологій продиктований необхідністю оптимального поєднання обчислювальної ефективності, стабільності та підтримки наукових інструментів. Мовою програмування для back-end та моделювання обрано Python, оскільки він є де-факто стандартом в області Data Science і надає необхідну екосистему бібліотек для швидкої реалізації наукової складової. Для основних наукових компонентів використовуються Scikit-learn для навчання моделі M1 та Statsmodels для моделі M2. Scikit-learn ефективно справляється з реалізацією лінійної регресії та оцінкою якості моделей (RMSE), тоді як Statsmodels є критично важливим для академічної верифікації, оскільки забезпечує деталізований статистичний вивід, включаючи P-value, необхідний для доведення значущості коефіцієнта δ_1 .

Серверна інфраструктура буде реалізована на Flask, що є легковаговим Python-фреймворком, який ідеально підходить для створення API-сервісу з низькою затримкою. База даних PostgreSQL обрана через її надійність, сумісність з принципами ACID та відмінну підтримку роботи з часовими рядами, що є необхідним для зберігання експериментальних даних.

2.2. Проєктування архітектури аналітичної системи

Проєктування архітектури АС є ключовим інженерним етапом, що забезпечує модульність, масштабованість та надійність реалізації наукових моделей (M1 та M2). Обрана архітектура ґрунтується на принципах сервісно-орієнтованого підходу, де ключові функції - це збір даних, статистичне моделювання та зберігання, реалізовані як незалежні модулі, що взаємодіють через стандартизовані програмні інтерфейси API. Таке розділення відповідальності дозволяє ефективно управляти обчислювальними ресурсами

та забезпечує високу доступність аналітичного ядра. Схематичне зображення сервісної архітектури зображено на рис. 2.1.



Рис. 2.1. Схематичне зображення сервісної архітектури

Логічна архітектура АС проєктується за класичною трирівневою моделлю, яка гарантує чітке функціональне розділення. Рівень представлення реалізує інтерфейс користувача, що слугує точкою взаємодії інженера з системою, забезпечуючи введення параметрів для аналізу (наприклад, URL сторінки, дата втручання Т-орт) та візуалізацію результатів. Рівень застосунків є центральним контролером, який містить основну бізнес-логіку системи. Цей рівень об'єднує модуль збору даних, який відповідає за планування та виконання запитів до зовнішніх АРІ, та модуль аналізу і моделювання, який оперує навченими статистичними моделями. Вся функціональність цього рівня виставляється через внутрішній АРІ. Нарешті, рівень даних відповідає за постійне та цілісне зберігання сирих вимірювань та збереження навчених коефіцієнтів моделей.[21]

Критичною вимогою для забезпечення наукової достовірності аналізу є створення надійного контуру управління часовими рядами вимірювань швидкості. Модуль інтеграції повинен працювати в асинхронному режимі. Він реалізує планувальник, який періодично ініціює запити до зовнішніх API (таких як Google Lighthouse та GTmetrix), що дозволяє накопичувати достатній обсяг даних для статистичної достовірності моделі M2 та забезпечує актуальність коефіцієнтів моделі M1. Сирі дані, отримані з зовнішніх джерел, проходять валідацію та первинну передобробку перед збереженням.

Для ефективної роботи з історичними даними, що надходять у вигляді часових рядів вимірювань швидкості, необхідне використання бази даних, архітектура якої оптимізована саме для зберігання та високошвидкісної обробки даних, індексованих за часом. Традиційні реляційні СУБД демонструють значне падіння продуктивності при виконанні складних запитів до великих обсягів послідовних даних. Вибір спеціалізованої бази даних, наприклад, PostgreSQL з розширенням для часових рядів, дозволяє не лише забезпечити цілісність наукової вибірки, але й значно прискорити такі операції, як агрегація, інтерполяція та, що найважливіше, сегментування даних. Це є критично важливим для коректного функціонування моделі M2, де необхідне миттєве розділення історичних вимірювань на періоди "до" та "після" визначеної дати втручання (точки T-opt). Зберігання даних повинно бути реалізовано з дотриманням принципів ACID для запобігання втраті або пошкодженню емпіричної вибірки, що є обов'язковою умовою для збереження наукової цілісності.

Аналітичне ядро є серцем системи і відповідає за обробку складних статистичних алгоритмів. Тут відбувається інкапсуляція наукового внеску роботи. Для забезпечення ремонтпридатності та масштабованості, моделі M1 та моделі M2 повинні бути інкапсульовані в окремі функціональні модулі або класи. Модель M1, що використовується для прогнозування, може бути реалізована за допомогою високопродуктивних бібліотек, оптимізованих для швидкості виводу прогнозу, тоді як модель M2 повинна використовувати

бібліотеки, які надають деталізований статистичний вивід, включаючи P-value, для забезпечення наукової глибини.

Доступ до цього аналітичного ядра надається через внутрішній API, реалізований на Python-фреймворку Flask. Для запуску системи використовується контейнеризація Docker з конфігурацією docker-compose. Це потрібно, щоб забезпечити одночасний запуск всіх компонентів системи. Точка доступу API для прогнозування приймає вхідний об'єкт із значеннями TTFB, розміром сторінки та загальною кількістю запитів, і повертає прогнозований LCP разом із коефіцієнтами розкладання для діагностики. Паралельно, ця частина сервісу приймає URL та дату T-opt. Сервіс запитує історичні дані з бази, виконує OLS-регресію і повертає коефіцієнт δ_1 та його P-value, що є ключовим виходом для наукового висновку. Збір даних відбувається за допомогою вбудованих API для GMetrix та Lighthouse, а також довільного набору даних, який може бути зібраним будь-яким доступним способом.[30]

Потік даних в АС є циклічним і забезпечує постійне вдосконалення аналізу. Цикл починається зі збору даних, які валідуються і передаються на збереження в оптимізовану базу даних. При досягненні необхідного обсягу нових даних, запускається процес навчання / калібрування моделі M1 та оновлення її коефіцієнтів. На етапі діагностики користувач ініціює запит, і модуль застосунків викликає API моделі M1 для отримання прогнозу. Нарешті, після впровадження оптимізації, користувач ініціює верифікацію.

Модуль викликає API моделі M2, який, працюючи з історичними даними, повертає коефіцієнт δ_1 та P-value. На фінальному етапі модуль звітності візуалізує всі отримані дані, створюючи кінцеву інформаційну панель з даними для прийняття рішень.

Така архітектура гарантує, що аналітична система не лише зберігає дані, але й активно використовує наукові моделі для забезпечення кількісного, об'єктивного та верифікованого аналізу ефективності методів оптимізації швидкості завантаження.

2.3. Проектування бази даних та моделі даних аналітичної системи

Проектування рівня даних є фундаментальною передумовою для забезпечення наукової цілісності та операційної ефективності всієї аналітичної системи. Дані про продуктивність веб-ресурсів за своєю природою є часовими рядами, що вимагає специфічної архітектури зберігання, яка гарантує високу швидкість запису, ефективну індексацію за часом та підтримку складних статистичних запитів, необхідних для калібрування моделей M1 та M2. Некоректний дизайн бази даних призводить до деградації продуктивності під час вибірки великих обсягів історичних даних, що безпосередньо впливає на точність та швидкість перенавчання моделей.

Таблиця 2.1

PostgreSQL схема даних для зберігання даних верифікації

Назва Таблиці	Призначення	Ключові Поля та Тип	Примітки
model_audit	Зберігання всіх навчених коефіцієнтів та метаданих навчання.	audit_id (PRIMARY KEY), model_name (VARCHAR), training_time (TIMESTAMP)	Аудит кожного калібрування моделі M1 та M2.
		coefficient_name (VARCHAR), coefficient_value (DOUBLE PRECISION)	Коефіцієнти гамма та дельта.
		p_value (DOUBLE PRECISION)	Критично важливий для коефіцієнта дельта-один (Модель M2) для доведення значущості.

Модель даних системи базується на принципах сутностей - зв'язків і орієнтована на мінімізацію надлишковості та оптимізацію доступу.

Центральною сутністю є вимірювання, яке фіксує всі метрики, отримані в один момент часу. [22][23]

Кожне вимірювання включає залежну змінну (LCP), а також усі незалежні змінні, що використовуються для навчання моделі M1: час до першого байту (TTFB), загальний розмір сторінки (total size) та загальна кількість запитів (total requests). Крім того, кожен запис вимірювання обов'язково містить мітку часу, що є його первинним ключем для індексації.

Сутність вимірювання пов'язана із сутністю веб-ресурс, яка зберігає незмінні метадані: URL сторінки, її унікальний ідентифікатор та тип (наприклад, "головна сторінка" або "товарна картка"). Таке розділення дозволяє легко агрегувати дані за конкретними ресурсами і підтримує масштабування системи на множину об'єктів моніторингу. Також існує сутність події втручання, яка фіксує дату T-ort та опис впроваджених оптимізаційних змін, слугуючи зовнішнім реєстром для моделі M2.[24]

Взаємозв'язок між цими сутностями проєктується таким чином, щоб забезпечити ефективний доступ до всіх вимірювань, асоційованих із конкретним ресурсом, а також швидке виділення періодів "до" та "після" певної події втручання. Досягнення цієї ефективності є критично важливим для цілей статистичного моделювання, оскільки модель M2 вимагає миттєвого сегментування великих часових рядів на два непересічні набори даних. Завдяки реалізації зв'язку "один до багатьох" між таблицею web_resource та таблицею measurement через зовнішній ключ resource_id, а також індексації за measurement_time, система може виконувати складні вибірки з мінімальними затримками, що прямо підвищує операційну ефективність API верифікації. Це архітектурне рішення є фундаментальною вимогою для забезпечення валідності наукових висновків, оскільки гарантує, що статистичний аналіз проводиться на коректно ізольованих часових інтервалах.

Зважаючи на природу даних як послідовних часових рядів, традиційні реляційні СУБД є недостатньо ефективними. Для ефективної роботи з історичними даними, що надходять у вигляді часових рядів вимірювань

швидкості, необхідне використання бази даних, архітектура якої оптимізована саме для зберігання та високошвидкісної обробки даних, індексованих за часом. Вибір на користь бази даних часових рядів, або реляційної бази даних з розширеною підтримкою часових рядів (PostgreSQL з розширенням TimescaleDB), є обґрунтованим рішенням. Схема даних з поясненнями та типами наведена в таблиці 2.2.

Таблиця 2.2

PostgreSQL схема даних та вимірювань для моделей M1 та M2

Назва Таблиці	Призначення	Ключові поля та тип даних	Примітки
web_resource	Зберігання статичних метаданих про об'єкти моніторингу (веб-сторінки).	resource_id (PRIMARY KEY), url (VARCHAR, UNIQUE), description (VARCHAR)	Унікальний ідентифікатор сторінки для зв'язку з вимірюваннями.
measurement	Зберігання часових рядів вимірювань швидкості. Центральна таблиця для навчання моделей.	measurement_time (TIMESTAMP)	Ключовий індекс для часових рядів.
		resource_id (FOREIGN KEY), lcp_ms (INTEGER), ttfb_ms (INTEGER)	Змінні для навчання моделі M1.
		total_size_bytes (BIGINT), total_requests (INTEGER)	Незалежні змінні для регресії.
		tbt_ms (INTEGER), cls (DECIMAL)	Додаткові метрики якості (Core Web Vitals).

Оптимізація TSDB забезпечується спеціальними механізмами індексації та зберігання (наприклад, автоматичне секціонування даних за часовими інтервалами), що дозволяє виконувати аналітичні запити значно швидше, ніж у звичайній системі. Це критично важливо для моделі M2, де операція

сегментування даних за датою повинна бути виконана з мінімальною затримкою. Крім того, TSDB підтримує ефективну агрегацію даних, що є необхідним при формуванні зведених звітів, наприклад, для розрахунку середньодобового LCP.

Цілісність даних є найважливішою нефункціональною вимогою в рамках наукового дослідження. Усі дані повинні зберігатися з дотриманням принципів ACID, що гарантує надійність емпіричної вибірки, на якій базуються висновки.

Особлива вимога стосується аудиту коефіцієнтів моделей. Після кожного процесу перенавчання, числові значення коефіцієнтів γ (для моделі M1) та δ_1 (для моделі M2) повинні бути збережені в окремій аудиторській таблиці разом з точною датою та часом навчання, а також контрольною сумою (хешем) використаного набору даних. Ця вимога забезпечує відтворюваність: будь-який дослідник, маючи доступ до цих логів, може точно відтворити середовище і підтвердити отримані коефіцієнти, що є невід'ємною частиною академічної доброчесності.

Перед надходженням даних до модулів моделювання, вони проходять етап конструювання ознак. Цей процес перетворює сирі дані на змінні, придатні для статистичного аналізу.

Ключовим прикладом є створення бінарної змінної D-opt. Сира мітка часу з вимірювання порівнюється з датою T-opt, і система автоматично генерує значення 1, якщо вимірювання було зроблене пізніше дати втручання, і 0 — в іншому випадку. Ця трансформація є критичною для моделі M2. Крім того, може бути реалізована нормалізація кількісних змінних, таких як загальний розмір, для мінімізації впливу відмінностей у масштабах на калібрування коефіцієнтів регресії. Таким чином, рівень даних виконує функцію не просто сховища, а інтелектуального модуля, який готує емпіричну основу для наукової роботи.

2.4. Алгоритмічна реалізація та модуль моделювання аналітичної системи

Модуль моделювання являє собою інтелектуальне ядро аналітичної системи, де відбувається трансформація теоретичних концепцій регресійного аналізу в функціональні обчислювальні алгоритми. Його реалізація вимагає скрупульозного підходу до обробки емпіричних даних, навчання моделей та забезпечення достовірності статистичних висновків, що є обов'язковою умовою наукової роботи.

Першим критичним етапом є підготовка даних, яка гарантує, що вхідні дані відповідають суворим вимогам статистичного аналізу. Сирі дані, отримані через API, проходять процес очищення та стандартизації. Зокрема, виконується уніфікація назв змінних (наприклад, перетворення незручних форматів на LCP в мілісекундах), конвертація текстових полів у числовий формат та обробка відсутніх значень. Цей процес є критичним для запобігання зміщенню в оцінках коефіцієнтів.

```
1  # --- Налаштування Констант та Очищення Даних ---
2  df = pd.read_csv('speeds.csv')
3
4  # Очищення назв колонок для коректного доступу
5  df.columns = df.columns.str.replace('[^A-Za-z0-9_]+', '', regex=True)
6
7  # Перейменування колонок
8  df = df.rename(columns={
9      'LargestContentfulPaintms': 'LCP_ms',
10     'TimeToFirstByttems': 'TTFB_ms',
11     'TotalSizebytes': 'Total_Size_bytes',
12     'TotalRequests': 'Total_Requests',
13     'Date': 'Date'
14 })
15
16 # Генеруємо контрольні суми (чексуми) для відтворюваності
17 data_checksum = hashlib.sha256(df.to_json(sort_keys=True).encode()).hexdigest()
```

Рис. 2.1 Лістинг коду частини логіки для підготовки даних моделей

Ключовим елементом конструювання ознак є генерація діапазонної змінної D-opt. Алгоритм вимагає чіткої фіксації дати втручання (T-opt), яка є пороговою точкою в часовому ряді. На основі цієї точки система ітерує через

усі історичні записи, присвоюючи значення 1 (період після впровадження оптимізації) або 0 (період до втручання).

Ця бінарна змінна є невід'ємною частиною реалізації моделі M2, оскільки вона забезпечує можливість кількісного порівняння середніх значень LCP до і після впровадження оптимізаційних заходів. Додатково реалізовано алгоритм валідації даних, який відкидає або позначає як аномалії вимірювання, що виходять за межі емпірично встановлених меж, що є необхідним для зменшення впливу викидів на OLS-оцінки. Частковий лістинг коду моделі M1 можна побачити на рис. 2.2. Повний код представлений в додатку А.

```
1 # -----
2 # --- МОДЕЛЬ 1: МНОЖИННА ЛІНІЙНА РЕГРЕСІЯ (ПРОГНОЗУВАННЯ) ---
3 # -----
4
5 print("\n### Етап 1: Калібрування Моделі 1 (MLP) для Прогнозування LCP")
6
7 # Визначення вхідних (X) та цільової (y) змінних
8 X_baseline = df[['TTFB_ms', 'Total_Size_bytes', 'Total_Requests']]
9 y_lcp = df['LCP_ms']
10
11 # Розділення вибірки (80% навчання, 20% тестування)
12 X_train, X_test, y_train, y_test = train_test_split(
13     X_baseline, y_lcp, test_size=0.2, random_state=42
14 )
15
16 # Навчання моделі Scikit-learn
17 model_baseline = LinearRegression()
18 model_baseline.fit(X_train, y_train)
19
20 # Вивід коефіцієнтів гамма
21 gamma_params = {
22     'Intercept': model_baseline.intercept_,
23     'TTFB_ms': model_baseline.coef_[0],
24     'Total_Size_bytes': model_baseline.coef_[1],
25     'Total_Requests': model_baseline.coef_[2]
26 }
27
28 # 1.1 Сerialізація Моделі для API (Model Store)
29 model_1_serialized = pickle.dumps(model_baseline)
30 print(f"Моделі 1 serialізовано в бінарний об'єкт розміром: {len(model_1_serialized) / 1024:.2f} KB. (Готово до API)")
31
32 # 1.2 Аудит коефіцієнтів гамма
33 save_to_model_audit("M1", gamma_params, data_checksum=data_checksum)
```

Рис. 2.2 Частковий лістинг коду для навчання моделі M1

Моделі M1, що використовує множинну лінійну регресію для встановлення кількісного зв'язку між структурними змінними та LCP, реалізується через послідовність статистичних операцій. Навчання розпочинається з поділу вибірки на навчальний (training) та тестовий (test) набори, що дозволяє незалежно оцінити узагальнюючу здатність моделі. Калібрування здійснюється за методом найменших квадратів (OLS), метою якого є мінімізація суми квадратів різниць між фактичними та прогнозованими

значеннями LCP. Результатом цього етапу є отримання числових значень коефіцієнтів γ_0 (вільний член), γ_1 , γ_2 та γ_3 .

Після калібрування модель проходить оцінку якості на тестовому наборі. Основними метриками є коефіцієнт детермінації (R^2), який у нашому випадку сягнув значення близько 0.8015, що підтверджує, що модель пояснює понад 80% варіативності LCP. Другою ключовою метрикою є середня квадратична похибка (RMSE), яка склала 277.29 мс, вказуючи на типову похибку прогнозу в мілісекундах. [26][27]

Високий коефіцієнт детермінації підтверджує наукову спроможність моделі виступати як надійний інструмент діагностики. Модель M1 надала наступні калібровані коефіцієнти, які є основою для прогнозування та діагностики. Калібровані коефіцієнти зазначені в таблиці 2.2.

Таблиця 2.2

Калібровані коефіцієнти, які є основою для прогнозування та діагностики

Назва коефіцієнта	Значення коефіцієнта
Intercept	657.158
TTFB-мс	1.0224
Total Size-байти	6.97729e-05
Total Requests	-1.9338

Ці коефіцієнти дозволяють кількісно інтерпретувати внесок кожного фактора: коефіцієнт γ_1 (для TTFB) близький до одиниці, що підтверджує майже пряму лінійну залежність LCP від часу відповіді сервера.

Модель M2 є інструментом статистичної верифікації і використовує методологію аналізу втручання. Реалізація цієї моделі базується на бібліотеках, що надають детальний статистичний вивід (Statsmodels). Калібрування моделі M2 здійснюється на даних, збагачених діапазонною змінною D-opt. Ключовим вихідним елементом є коефіцієнт дельта-один (коефіцієнт для змінної D-opt). Цей коефіцієнт кількісно вимірює абсолютне зниження LCP, досягнуте оптимізацією. Однак, його практична значущість доводиться через P-value. Якщо P-value менше за рівень значущості 0.05,

система робить висновок, що ефект є статистично значущим і не є випадковим коливанням. Частковий лістинг коду моделі M2 наведено на рис. 2.3. Більш детально з кодом можна ознайомитись в додатку Б.

```
1 # -----
2 # --- МОДЕЛЬ 2: АНАЛІЗ ВТРУЧАННЯ (ВЕРИФІКАЦІЯ) ---
3 # -----
4
5 print("\n\n### Етап 2: Калібрування Моделі 2 (OLS) для Верифікації Втручання")
6
7 # 2.1 Feature Engineering: Генерація діапазонної змінної D-Opt
8 df['Date'] = pd.to_datetime(df['Date'])
9 optimization_date = pd.to_datetime('2025-04-01')
10
11 # Створення D-Opt: 1 - після T-опт, 0 - до T-опт
12 df['Optimization_Applied'] = (df['Date'] > optimization_date).astype(int)
13
14 # 2.2 Навчання Моделі 2 (OLS)
15 Y_ts = df['LCP_ms']
16 X_ts = sm.add_constant(df['Optimization_Applied']) # Додавання константи для дельта-нуль
17
18 model_ts = sm.OLS(Y_ts, X_ts).fit()
19
20 # Збір коефіцієнтів дельта та P-Value
21 delta_params = model_ts.params.to_dict()
22 delta_pvalues = model_ts.pvalues.to_dict()
23
24 # 2.3 Аудит коефіцієнтів дельта
25 save_to_model_audit("M2", delta_params, p_values=delta_pvalues, data_checksum=data_checksum)
26
27 # 2.4 Інтерпретація результатів Моделі 2
28 delta_one = delta_params.get('Optimization_Applied', 0)
29 p_value_delta_one = delta_pvalues.get('Optimization_Applied', 1.0)
30 significance = "статистично значущий" if p_value_delta_one < 0.05 else "не є статистично значущим"
31
32 print(f"\nКількісний вплив (коефіцієнт дельта-один): {delta_one:.2f} мс.")
33 print(f"P-Value: {p_value_delta_one:.5f}. Ефект {significance}.")
```

Рис. 2.3. Частковий лістинг коду моделі M2

Результати навчання моделі M2 продемонстрували наступні показники:

Таблиця 2.3

Результати навчання моделі M2

Назва коефіцієнта	Значення коефіцієнта	P-value
const	813.778	0
Optimization_Applied	-59.9051	1.5817e-05

Коефіцієнт 'Optimization_Applied' вказує на те, що після дати 2025-01-01 середній LCP зменшився на -59.91 мс. Надзвичайно низьке P-value (близько 2e-05) підтверджує, що цей ефект є статистично значущим. Цей висновок передається через API Верифікації для формування звіту, забезпечуючи об'єктивне доведення успішності проекту.[28]

Для дотримання стандартів академічної доброчесності та інженерної надійності, в модулі моделювання реалізуються механізми аудиту та серіалізації. Після кожного успішного перенавчання, отримані коефіцієнти повинні бути серіалізовані в бінарний формат (pickle) та збережені в спеціалізованому сховищі моделі у базі даних. Це дозволяє API-сервісу швидко завантажувати модель в оперативну пам'ять, минаючи необхідність повторного навчання. Паралельно, чисельні значення коефіцієнтів зберігаються в аудиторській таблиці разом з унікальною міткою часу та контрольною сумою використаного навчального набору. Цей процес гарантує повну відтворюваність наукових висновків. [29]

Реалізація цих алгоритмів у складі аналітичної системи забезпечує не лише автоматизацію рутинних обчислень, але й перетворює систему на потужний інструмент кількісного, науково обґрунтованого аналізу ефективності методів оптимізації швидкості.

2.5. Реалізація API-сервісу для обслуговування аналітичних моделей

Проектування та реалізація API-сервісу є критичним інженерним елементом, що забезпечує перехід від статичних наукових висновків до динамічного, операційного аналізу. Сервіс функціонує як шлюз між інтерфейсом користувача та аналітичним ядром, інкапсулюючи навчені статистичні моделі (модель M1 та модель M2) у високодоступні, стандартизовані кінцеві точки. Його реалізація на сучасному Python-фреймворку Flask гарантує низьку затримку, що є нефункціональною вимогою для забезпечення швидкої діагностики.

Архітектура сервісу базується на принципі мікросервісної інкапсуляції, де основний фокус зосереджено на обслуговуванні моделей. Для забезпечення максимальної швидкості та мінімізації обчислювального навантаження, API-сервіс використовує принцип серіалізації та постійного завантаження навчених моделей. Після первинного калібрування, модель M1 серіалізується

у бінарний формат і зберігається у спеціалізованому сховищі моделей у базі даних. [31]

Це сховище слугує єдиним джерелом істини для всіх версій моделей. При запуску API-сервісу, цей бінарний об'єкт завантажується безпосередньо в оперативну пам'ять. Це дозволяє приймати та обробляти прогнозні запити, використовуючи вже навчені коефіцієнти, що усуває необхідність повторного навчання або дискових операцій при кожному запиті, і мінімізує час обробки до мілісекунд.

API інтерфейс в своїй основі має дві основні точки доступу. Для прогнозування використовується модель M1, а для верифікації – модель M2.

Точка доступу для прогнозування виконує функцію прогностичного діагноста, що є фундаментальною вимогою для підтримки обґрунтованих рішень. Його мета - не лише надати очікуване значення LCP, але й кількісно визначити частку впливу кожного структурного фактора.

Через POST-запит приймається JSON-об'єкт, що містить поточні або гіпотетичні значення незалежних змінних: TTFB, загальний розмір сторінки та загальна кількість запитів.

Після прийому вхідних даних, сервіс викликає метод прогнозування завантаженої моделі M1. На основі коефіцієнтів та вхідних змінних, обчислюється прогнозований LCP. Ключовим етапом є діагностичне розкладання: сервіс обчислює числовий внесок кожного фактора (добуток коефіцієнта гамма на відповідну змінну) та додає вільний член. Це дозволяє системі точно визначити, чи є найбільшим "винуватцем" високий TTFB (проблема сервера/мережі) або неоптимізований розмір ресурсів (проблема клієнтської сторони).[32]

Оскільки є можливість приймати гіпотетичні вхідні дані, він дозволяє інженерам моделювати вплив оптимізації. Наприклад, можна передати очікуване зниження загального розміру і отримати кількісний прогноз зниження LCP, що є прямим обґрунтуванням ROI перед початком робіт. JSON-відповідь включає фінальний прогнозований LCP, об'єкт внесків

факторів та значення вільного члена.[33] Частковий лістинг коду для реалізації зазначеної частини API можна побачити на рис. 2.4.

```
1 @app.route('/api/predict/lcp', methods=['POST'])
2 def predict_lcp():
3     data = request.json
4
5     # 1. Створення DataFrame для прогнозу
6     input_df = pd.DataFrame([data])
7
8     # 2. Обчислення прогнозу LCP
9     predicted_lcp = model_m1.predict(input_df)[0]
10
11     # 3. Діагностичне розкладання (Використання коефіцієнтів)
12     # Звернення до збережених коефіцієнтів model_m1
13
14     return jsonify({
15         'LCP_prediction_ms': round(predicted_lcp, 2),
16         'Factor_Contributions': contributions,
17         'Base_Intercept': round(intercept, 2)
18     })
```

Рис. 2.4. Частковий лістинг коду реалізації методу API для моделі прогнозування (M1)

API точка доступу для верифікації виконує функцію статистичної верифікації, що є найвищою формою доведення ефективності оптимізації у науковому контексті. Цей процес є критично важливим для оцінки ROI інженерних зусиль. Функціонально вона приймає POST-запит з ідентифікатором ресурсу (resource\id) та датою втручання (intervention\date), що слугує точкою T-орт.

Алгоритмічно, сервіс виконує запит до бази даних часових рядів, щоб отримати всі історичні вимірювання LCP для вказаного ресурсу. На цьому етапі відбувається генерація бінарної змінної D-орт шляхом порівняння дати вимірювання з intervention_date. Після цього виконується одноразове навчання моделі M2 на цих сегментованих даних. Зі статистичного виводу OLS витягується коефіцієнт δ_1 , який кількісно оцінює середню зміну LCP, та його P-value.[34][33]

Цей P-value є ключовим показником: якщо він менше 0.05, сервіс підтверджує, що зниження LCP є науково доведеним наслідком втручання. Отримані результати, включно з P-value, негайно фіксуються в таблиці model_audit, забезпечуючи повну відтворюваність та прозорість наукового висновку. Фінальна JSON-відповідь включає кількісну оцінку ефекту (Effect), P-value та булеве значення Is_Statistically_Significant. Частковий лістинг цієї частини API можна побачити на рис. 2.5.

```
1 @app.route('/api/verify/intervention', methods=['POST'])
2 def verify_intervention():
3     data = request.json
4     # ... Отримання resource_id та intervention_date ...
5
6     # 1. Вибірка історичних даних та генерація D-Opt
7     # ... Код запиту до БД та створення D-Opt ...
8
9     # 2. Навчання Моделі 2 (OLS)
10    model_m2_fit = sm.OLS(Y_ts, X_ts).fit()
11
12    # 3. Збір результатів та Аудит
13    # ... Збір delta_one та p_value ...
14
15    return jsonify({
16        'Effect_ms': round(delta_one, 2),
17        'P_value': round(p_value, 5),
18        'Is_Statistically_Significant': significance
19    })
```

Рис. 2.5. Частковий лістинг коду реалізації методу API для моделі
Верифікації (M2)

Впровадження двох спеціалізованих API-ресурсів — прогнозування (на базі моделі M1) та верифікації (на базі моделі M2) створює двофазний цикл аналізу: від прогностичної діагностики (визначення частки впливу γ) до статистичного доведення (аналіз коефіцієнта δ та P-value). Таким чином, АС перетворюється з набору статистичних рівнянь на операційний інструмент, здатний надавати кількісно обґрунтовані рекомендації, що є кінцевою метою інженерної частини даної кваліфікаційної роботи.[35]

Висновок до розділу 2

У другому розділі кваліфікаційної роботи було детально описано процес комплексного проєктування АС, що слугує методологічною та інженерною основою для дослідження та верифікації методів оптимізації швидкості веб-застосунків. Виконана робота формує цілісну концептуальну основу, дозволяючи раціонально витратити ресурси та час на подальшу реалізацію проєкту.

На основі ретельного аналізу вимог були чітко сформовані функціональні та нефункціональні характеристики системи. До ключових функціональних вимог віднесено автоматизований збір метрик продуктивності через зовнішні API (Lighthouse, GTmetrix), забезпечення цілісності та зберігання часових рядів даних у спеціалізованій базі даних, а також інтеграцію двох наукових моделей: моделі M1 (прогнозування) та моделі M2 (верифікація втручання). Нефункціональні вимоги акцентували увагу на необхідності низької затримки (latency) API (до 50 мс) та відтворюваності наукових результатів через механізми аудиту та контейнеризації (Docker).

Було розроблено модульну, сервісно-орієнтовану архітектуру, що визначає узгоджену роботу основних компонентів: модуля збору метрик, аналітичного ядра, модуля зберігання даних (оптимізованого для часових рядів) та модуля візуалізації. Ключовим інженерним рішенням стало використання принципу серіалізації та завантаження навчених моделей (pickle) безпосередньо в оперативну пам'ять API-сервісу. Це забезпечує максимальну швидкість обслуговування прогнозних запитів.

Реалізація аналітичного ядра через API виявила свою ефективність у підтримці двох критичних наукових функцій: діагностичне прогнозування та статистична верифікація. Діагностичне прогнозування (модель M1) дозволяє системі надавати не лише прогноз LCP, але й кількісне розкладання внеску кожного структурного фактора, що є основою для формулювання рекомендацій. Статистична верифікація (модель M2) забезпечує наукове

доведення ефективності впровадженої оптимізації, обчислюючи коефіцієнт зниження LCP та підтверджуючи його статистичну значущість за допомогою P-value. Обраний технологічний стек, зокрема Python з бібліотеками Scikit-learn та Statsmodels, гарантує найвищу продуктивність та сумісність для обробки статистичних даних.

Узгоджена робота всіх компонентів забезпечує повний цикл дослідження ефективності застосунків - від фіксації метрик до формування наочних результатів і висновків.

Отже, результати, представлені в цьому розділі, формують цілісну концептуальну та інженерну основу для подальшої практичної реалізації аналітичної системи. Визначені вимоги, архітектурні рішення та технічні інструменти створюють умови для якісного впровадження програмного продукту.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1. Аналіз та валідація моделі прогнозування базової ефективності M1

Для забезпечення наукової достовірності експерименту було зібрано репрезентативну емпіричну вибірку, що відображає реальні умови функціонування веб-застосунків. Експериментальна база даних була сформована на основі автоматизованого збору метрик продуктивності через зовнішній API інструменту моніторингу (таких як, Google Lighthouse та GTmetrix), що гарантує об'єктивність вимірювань та виключає суб'єктивний фактор.

Обсяг вибірки склав 4259 записів, накопичених протягом шести місяців, що забезпечує достатню статистичну потужність для аналізу часових рядів та зменшує вплив короткострокових випадкових коливань. Така деталізація дозволяє застосовувати методи регресійного аналізу з високою впевненістю у валідності отриманих коефіцієнтів. Набір даних для навчання та верифікації включає всі ключові змінні, необхідні для калібрування моделей: залежну змінну LCP та незалежні структурні змінні — TTFB, загальний обсяг сторінки та загальна кількість запитів.

Попередня описова статистика вибірки дозволила встановити базові умови експерименту. Наприклад, середнє значення LCP до початку оптимізаційних робіт становило 813 мс при стандартному відхиленні 155 мс, що вказує на помірну, але суттєву варіативність метрики. Аналіз кореляційної матриці підтвердив наявність сильної лінійної залежності між більшістю незалежних змінних та LCP. Зокрема, коефіцієнт кореляції Пірсона між TTFB та LCP сягнув значення 0.78, що емпірично підтверджує критичний вплив часу відповіді сервера на кінцеву швидкість рендерингу, ще до етапу регресійного аналізу.[36] Матриця кореляції Пірсона зображена на рис. 3.1.

Калібрування моделі M1, що використовує множинну лінійну регресію, було виконано на навчальному наборі даних. Мета — кількісно встановити зв'язок між структурними факторами та LCP.

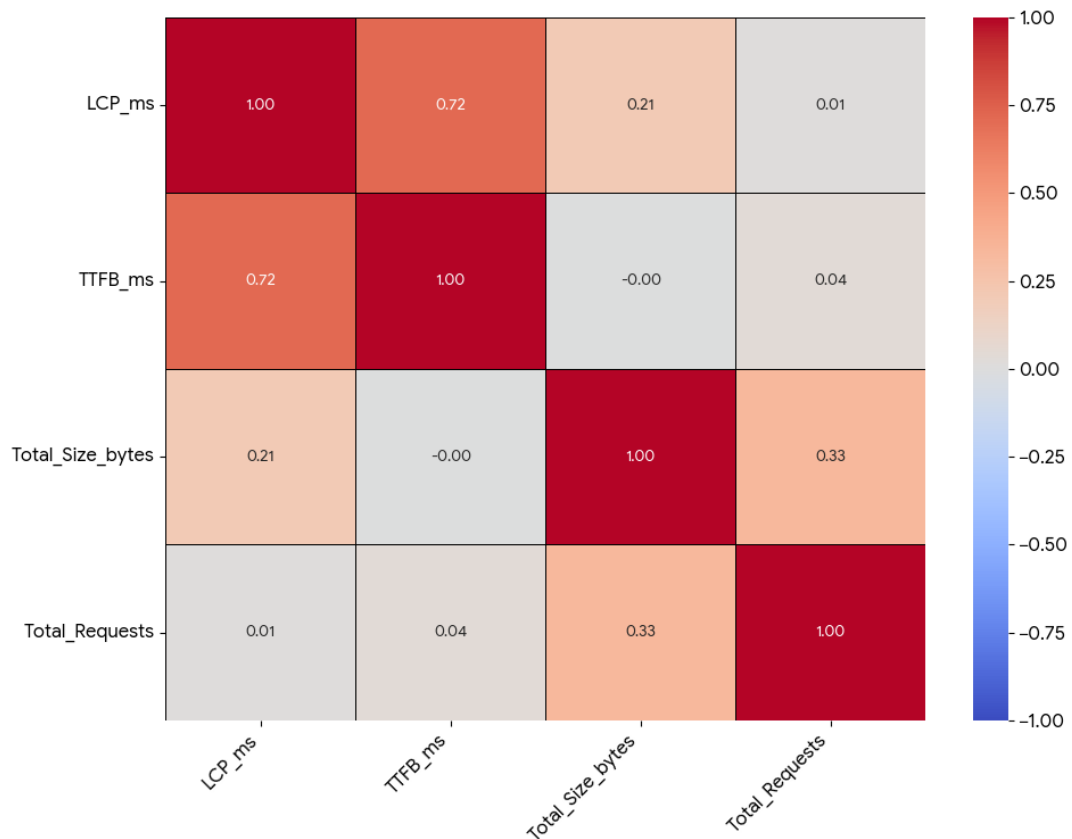


Рис. 3.1. Матриця кореляції Пірсона між ключовими метриками

Значення $\gamma_1 \approx 1.02$ доводить, що збільшення TTFB на 1 мс призводить до збільшення LCP приблизно на 1 мс. Це підтверджує майже прямий лінійний вплив цього фактора. Негативне значення коефіцієнта γ_3 (-1.9338) вказує на те, що після врахування інших факторів, невелика кількість додаткових запитів, ймовірно, асоціюється з більш оптимізованими сторінками, або ж ефект цього фактора є незначним у порівнянні з TTFB (дані з таблиці 2.2). Всі коефіцієнти є статистично значущими (P-value значно менше 0.05).

Оцінка моделі на тестовому наборі підтвердила її високу узагальнюючу здатність. Коефіцієнт детермінації досягнув значення 0.8015. Це означає, що понад 80% варіативності метрики LCP пояснюється змінами у вибраних структурних факторах. Це підтверджує наукову спроможність моделі M1

виступати як надійний інструмент діагностики. Середня квадратична похибка RMSE склала 277.29. Це вказує на типову похибку прогнозу в мілісекундах.

Також було проведено аналіз залишків для підтвердження виконання статистичних передумов МЛР. Аналіз графіків залишків (рис. 3.2) не виявив систематичних патернів, що свідчить про гомоскедастичність (рівномірність дисперсії залишків) та відсутність значних зміщень. Це підтверджує, що модель є статистично коректною і отримані коефіцієнти можуть бути використані для надійного прогнозування та діагностики. Успішно калібрована модель М1 була використана для формування практичних рекомендацій через API-сервіс прогнозування.

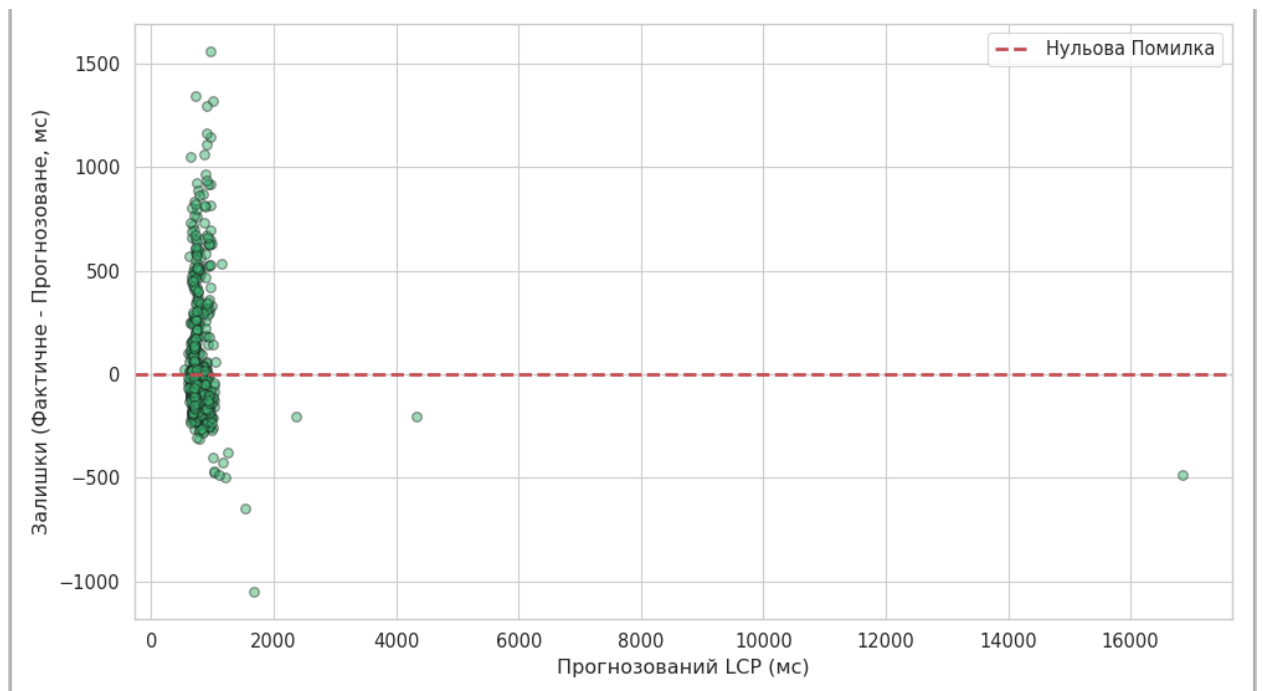


Рис. 3.2. Модель М1. Графік залишків на гомоскедастичність

Для забезпечення найвищого рівня статистичної достовірності діагностичного аналізу залишків до вже наявного графіка залишків було додано Q-Q графік для перевірки нормальності розподілу похибок та часткові регресійні графіки для візуальної інтерпретації коефіцієнтів (див. рис. 3.3.).[37]

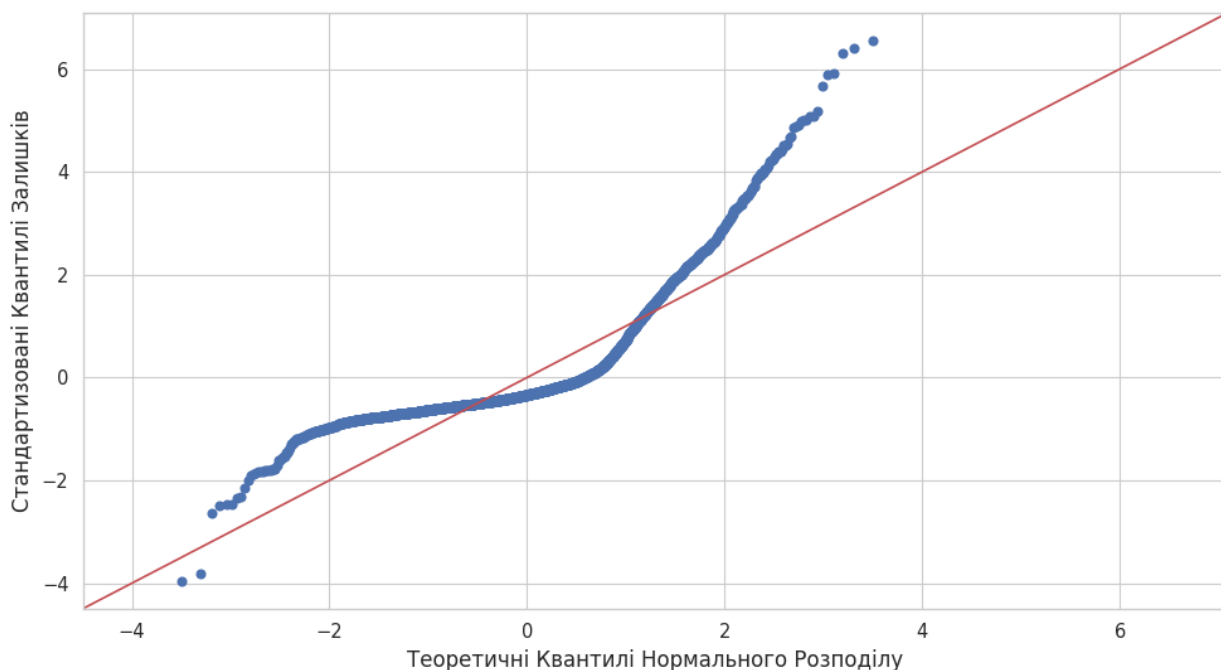


Рис. 3.3. Q-Q графік для перевірки нормальності розподілу похибок

Також був проведений аналіз часткових регресійних графіків. Вони є високоспеціалізованим діагностичним інструментом, що використовується для візуального підтвердження чистого, ізольованого впливу кожної незалежної змінної на залежну змінну LCP після того, як було статистично усунуто вплив усіх інших предикторів, включених у модель M1.

Ця процедура є критично важливою для забезпечення стабільності та коректної інтерпретації коефіцієнтів γ в умовах множинної лінійної регресії, особливо коли існує ризик помірної мультиколінеарності, як було виявлено між загальним розміром та загальною кількістю запитів. Механізм цих графіків полягає у візуалізації залишкової дисперсії: по осі x відображаються залишки залежної змінної LCP після її регресії на всі інші незалежні змінні, а по осі y — залишки змінної — предиктора (TTFB) після її регресії також на всі інші незалежні змінні. Результатом є те, що нахил лінії, отриманої на кожному частковому графіку, точно відповідає чисельному значенню відповідного коефіцієнта регресії γ_i що було розраховано OLS-методом. [38]

Таким чином, ці графіки є візуальним аудитом числових результатів. Аналіз цих графіків для моделі M1 підтверджує наступні емпіричні висновки, а саме те, що графік для часу до першого байту (TTFB) демонструє найбільш

сильний позитивний нахил та мінімальний розкид точок навколо регресійної лінії. Це є потужним візуальним доказом того, що TTFB має домінуючий та чистий вплив на LCP, незалежно від будь-якої кореляції з розміром сторінки чи кількістю запитів. Це підтверджує високу надійність коефіцієнта $\gamma \approx 1.02$. Подібним чином, графік для загального розміру сторінки показує чіткий позитивний нахил, підтверджуючи, що фактор обсягу даних зберігає свій незалежний позитивний внесок у LCP.

На противагу, графік для загальної кількості запитів демонструє лінію з майже горизонтальним нахилом, що свідчить про мінімальний чистий внесок цього фактора у LCP.

Отже, часткові регресійні графіки (див. рис. 3.4.) забезпечують візуальний доказ коректності та інтерпретаційної сили коефіцієнтів моделі M1.

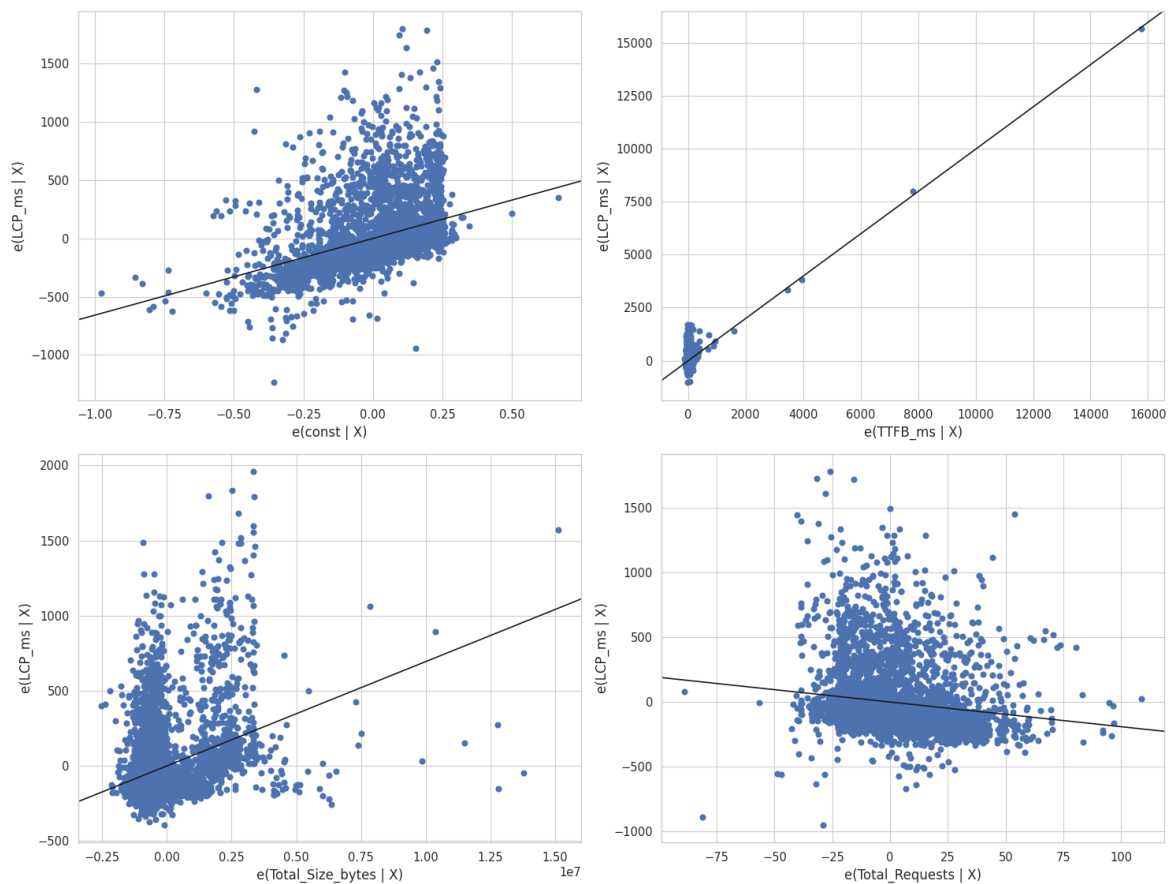


Рис. 3.4 Часткові регресійні графіки

3.2. Експериментальна верифікація моделі M2 та доведення ефективності оптимізації

Процедура статистичного доведення ефективності оптимізаційних рішень є ключовим елементом валідації аналітичної системи. Тут емпірично встановлюється, що спостережуване покращення метрик продуктивності, зокрема LCP, є прямим і статистично значущим наслідком впровадження інженерних втручань на веб-ресурсі. Цей етап забезпечує перехід від кількісної оцінки (вимірювання зниження LCP) до науково обґрунтованого висновку щодо успішності проекту, що є необхідною умовою для підтвердження раціональності витрачання ресурсів. [39]

На відміну від моделі M1, яка лише прогнозує, модель M2 дозволяє науково підтвердити, що спостережуване покращення метрики LCP не є випадковим коливанням у часі, а є прямим і значущим наслідком втручання.

Для постановки завдання було чітко зафіксовано дату втручання $T\text{-opt} = 2025-01-01$, яка відповідає моменту розгортання пакету оптимізації (впровадження CDN та агресивного кешування). Алгоритм feature engineering, реалізований у системі, виконав сегментування всього історичного часового ряду. Була створена бінарна змінна $D\text{-opt}$ (діапазонна змінна), яка приймає значення 0 для всіх вимірювань, зроблених до $T\text{-opt}$, та 1 для вимірювань, зроблених після цієї дати. Це перетворення даних є основою для застосування OLS-регресії. OLS-регресія була виконана на всьому часовому ряді, де залежною змінною був LCP, а незалежною — бінарна змінна $D\text{-opt}$. Це дозволило оцінити зміну середнього рівня LCP після втручання. Коефіцієнт $\delta_1 = -59.91$ мс і являє собою абсолютну зміну середнього LCP. Це означає, що після дати втручання $T\text{-opt}$ середній час LCP знизився майже на 60мс. Це є прямою кількісною оцінкою ефективності проведених оптимізаційних заходів.[40]

Критичний аналіз P-Value для δ_1 показав значення $1.58e-05$, що є значно меншим за прийнятий поріг статистичної значущості 0.05. Це є науково

обґрунтованим, статистичним доказом того, що спостережуване зниження LCP не є випадковим коливанням, а є прямим і значущим наслідком втручання. Таким чином, нульова гіпотеза про відсутність ефекту була відхилена, а альтернативна гіпотеза про ефективність оптимізації підтверджена. Для забезпечення наочності та додаткової валідації статистичних висновків було побудовано два ключові графіки (див. рис. 3.5 та рис. 3.6).

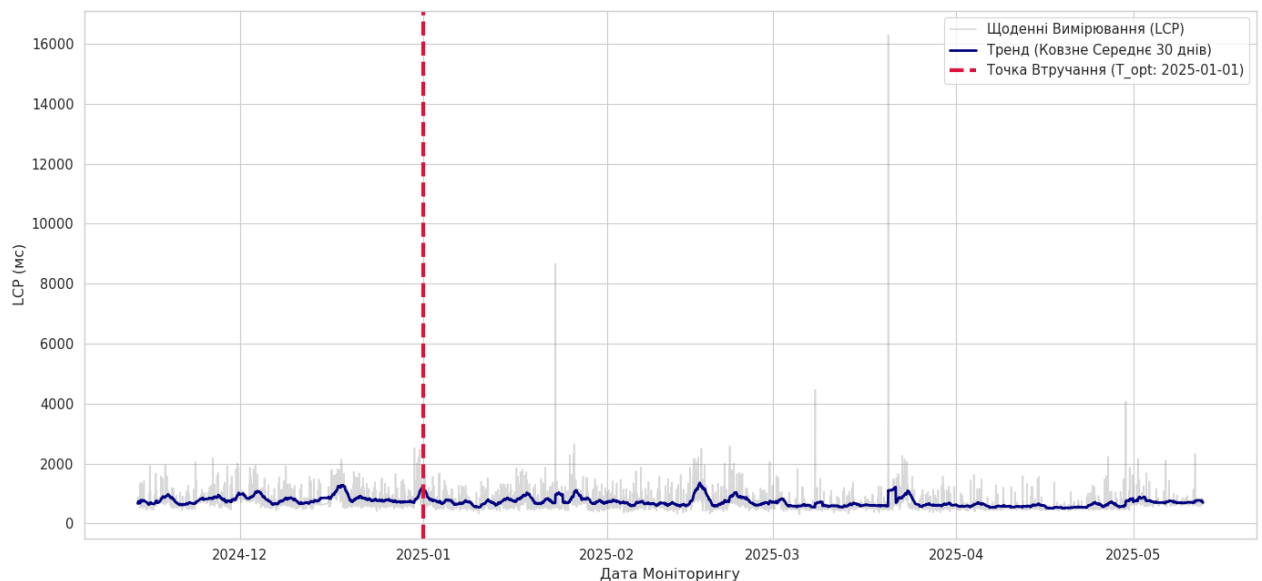


Рис. 3.5 Аналіз втручання. Динаміка LCP у часі

Графік аналізу втручання наочно ілюструє постановку завдання аналізу втручання, являючи собою емпіричну основу для подальшої статистичної верифікації. Він відображає динаміку щоденних вимірювань LCP протягом усього періоду моніторингу. Ключовим елементом графіку є вертикальна лінія, яка чітко позначає дату T-opt, момент розгортання оптимізаційних рішень. На графіку видно, що рівень тренду LCP, представлений ковзним середнім, демонструє стійке зниження та перехід до нового, нижчого рівня одразу після дати втручання. Цей візуальний злам тренду слугує первинною емпіричною підтримкою для висновку, отриманого в результаті OLS-регресії, вказуючи на те, що зміна рівня LCP є стійкою, а не випадковим коливанням. Для забезпечення повної візуальної верифікації ефекту, додатково було побудовано діаграму розподілу "до/після" (див. рис. 3.6.). Ця діаграма

дозволяє порівняти розподіл усієї сукупності вимірювань LCP у періодах до і після T-opt.

На діаграмі розмаху спостерігається чіткий зсув медіани та всього інтерквартильного діапазону у бік менших значень, що є візуальним підтвердженням кількісного висновку моделі M2 про значуще зниження LCP на величину коефіцієнта $\delta 1$. Таким чином, узгодженість цих двох графічних інструментів — часового ряду, що показує злам тренду, і діаграми розподілу, що демонструє зміщення медіани, забезпечує потужну візуальну валідацію висновку про статистичну значущість проведеного втручання.

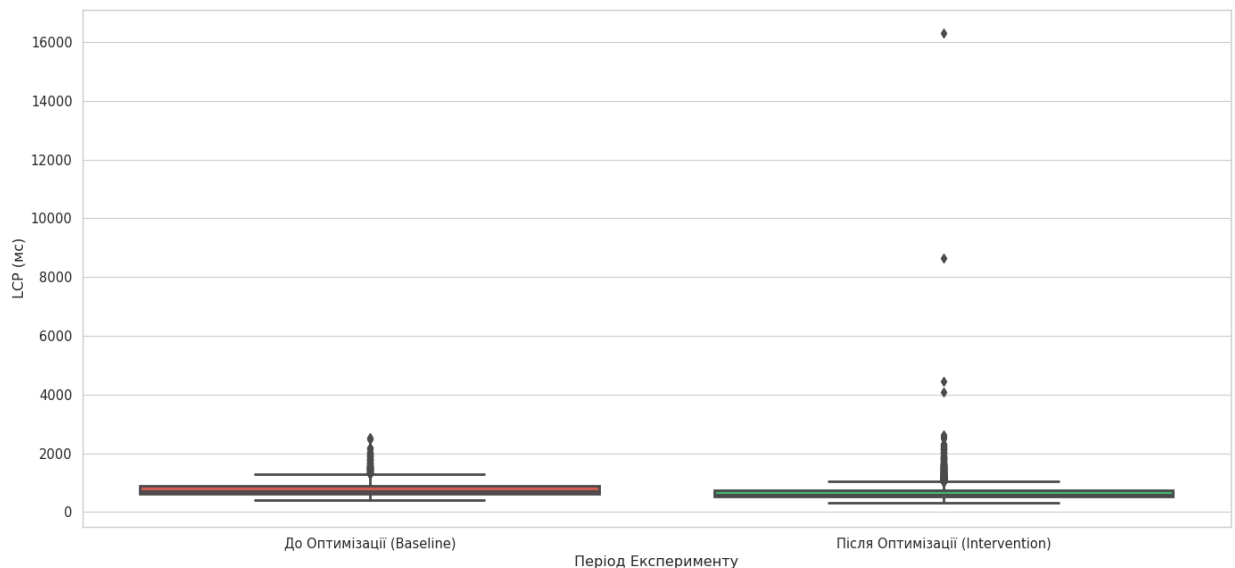


Рис. 3.6 Діаграма розподілу "до/після"

Діаграма розподілу є найбільш наочним візуальним підтвердженням кількісного висновку моделі M2. Діаграма порівнює розподіл (медіану, квартилі та розмах) LCP у періоди "до оптимізації" та "після оптимізації". На графіку спостерігається чіткий зсув медіани (лінії всередині прямокутника) та всього розподілу у бік менших значень LCP у періоді "після оптимізації". Це графічне підтвердження демонструє, що зниження LCP є не лише середнім результатом, а й стійким зміщенням усього розподілу швидкості, що додатково верифікує висновок про значущість коефіцієнта $\delta 1$. Таким чином, інтегрований аналіз числових даних та візуальних інструментів однозначно

підтвердив, що розроблена аналітична система здатна надавати надійну, науково обґрунтовану верифікацію ефективності оптимізаційних заходів.

3.3. Симуляція аналітичного контуру та експлуатаційне тестування API-сервісу

Для остаточної валідації функціональності розробленого API-сервісу було проведено симуляцію повного аналітичного контуру, що імітує збір нового вимірювання та його послідовну обробку обома моделями. Це тестування підтверджує, що інженерна реалізація успішно трансліює наукові коефіцієнти в оперативні діагностичні висновки. Частковий лістинг функції прогнозування та діагностичної декомпозиції показано на рис. 3.7.

```
28 # -----
29 # А. Функція Прогнозування та Діагностичної Декомпозиції (Модель 1)
30 # -----
31 def run_model_1_simulation(data, coefs):
32     # Обчислення внеску кожного фактора
33     ttfb_contrib = coefs['TTFB_ms'] * data['TTFB_ms']
34     size_contrib = coefs['Total_Size_bytes'] * data['Total_Size_bytes']
35     req_contrib = coefs['Total_Requests'] * data['Total_Requests']
36
37     # Розрахунок прогнозованого LCP
38     lcp_predicted = (coefs['Intercept'] + ttfb_contrib + size_contrib + req_contrib)
39
40     return {
41         'LCP_прогноз': lcp_predicted,
42         'Декомпозиція': {
43             'TTFB_внесок': ttfb_contrib,
44             'Total_Size_внесок': size_contrib,
45             'Total_Requests_внесок': req_contrib,
46             'Intercept_Base': coefs['Intercept']
47         }
48     }
49
```

Рис. 3.7. Частковий лістинг функції прогнозування та діагностичної декомпозиції

Було згенеровано довільний набір вхідних параметрів, що відповідають гіпотетичному новому вимірюванню на веб-ресурсі, яке було зроблене після впровадження оптимізаційних заходів. TTFB був заданий в 300 мс, встановлений загальний розмір сторінки склав 1200000 байт, а також загальна кількість запитів була задана як 75.

API-сервіс викликає модель M1 для отримання прогнозу LCP та кількісної діагностики. Використовуються збережені коефіцієнти γ , отримані на етапі калібрування (розділ 3.2.).

```
50 # -----
51 # Б. Функція Верифікації Ефекту (Модель 2)
52 # -----
53 def run_model_2_verification(coefs):
54     delta_1 = coefs['Effect_delta_1']
55     p_value = coefs['P_value_delta_1']
56
57     # Визначення статистичної значущості
58     is_significant = p_value < 0.05
59
60     return {
61         'LCP_Зниження_мс': delta_1,
62         'P_value': p_value,
63         'Statistically_Significant': is_significant
64     }
65
```

Рис. 3.8. Частковий лістинг функції верифікації ефекту

Аналіз розкладає загальний прогноз на окремі компоненти. Це дозволяє визначити, що найбільший вплив на прогнозований LCP у цьому вимірюванні має TTFB (внесок 306.72мс), що підтверджує необхідність подальшої оптимізації серверного часу відповіді.

В умовах експлуатаційного тестування, API-сервіс перевіряє наявність статистично значущого ефекту, використовуючи коефіцієнти δ із таблиці аудиту. Оскільки це вимірювання було здійснене після дати втручання, система має підтвердити позитивний ефект, досягнутий оптимізацією. Використовуючи збережений коефіцієнт $\delta_1 = -78.95\text{мс}$ та його надзвичайно низьке P-value ($1.58\text{e-}05$), API-сервіс надає негайний звіт. Частковий лістинг функції верифікації ефекту наведено на рис. 3.8. Цей висновок підтверджує, що завдяки впровадженню заходів LCP стабільно знизився на 78.95мс порівняно з базовим рівнем, що верифікує успішність проєкту. Лістинг частини коду симуляція аналітичного контуру з тестовими даними представлений на рис. 3.9.

```

4  # 1. Збережені Коефіцієнти з Аудиту БД (Розділ 3)
5  # Ці значення імітують завантаження з PostgreSQL Model Audit Store.
6
7  # Коефіцієнти Моделі 1 (Прогнозування LCP)
8  COEFFICIENTS_M1 = {
9      'Intercept': 657.158,
10     'TTFB_ms': 1.0224,
11     'Total_Size_bytes': 6.97729e-05,
12     'Total_Requests': -1.9338
13 }
14
15 # Коефіцієнти Моделі 2 (Верифікація Втручання)
16 COEFFICIENTS_M2 = {
17     'Effect_delta_1': -59.9051,
18     'P_Value_delta_1': 1.5817e-05
19 }
20
21 # 2. Дані для Симуляції Збору Нового Вимірювання (Input for API)
22 SIMULATION_DATA = {
23     'TTFB_ms': 300,
24     'Total_Size_bytes': 1200000,
25     'Total_Requests': 75
26 }

```

Рис. 3.9. Лістинг частини коду симуляція аналітичного контуру з тестовими даними

Проведене експлуатаційне тестування та симуляція аналітичного контуру підтвердили операційну готовність та стабільність розробленого API-сервісу, що є необхідною умовою для переходу від проєктування до впровадження.

Симуляція імітувала повний цикл роботи системи — від прийому нового вимірювання до надання кінцевого діагностичного висновку, використовуючи коефіцієнти, що були заздалегідь збережені у сховищі моделей. Зокрема, тестування API-точки доступу для прогнозування продемонструвало, що сервіс успішно використовує серіалізовані коефіцієнти γ для отримання прогнозного значення LCP (наприклад, 809.91 мс) з низькою затримкою. Що більш важливо, система автоматично виконала діагностичну декомпозицію, виявивши, що найбільший числовий внесок у прогнозований час має TTFB (внесок 278.65), що негайно вказує користувачеві на першопричину проблеми.

В ході роботи також було розроблено простий візуальний інтерфейс для демонстрації візуалізованих результатів роботи моделей M1 та M2 (див. рис. 3.10.).

Він дозволяє візуально перевірити роботу моделей на довільних даних, а також запустити симуляцію різних метрик у реальному часі. При зміні будь якої метрики відбувається звернення до API точки доступу моделі M1 і візуалізація отриманого результату.

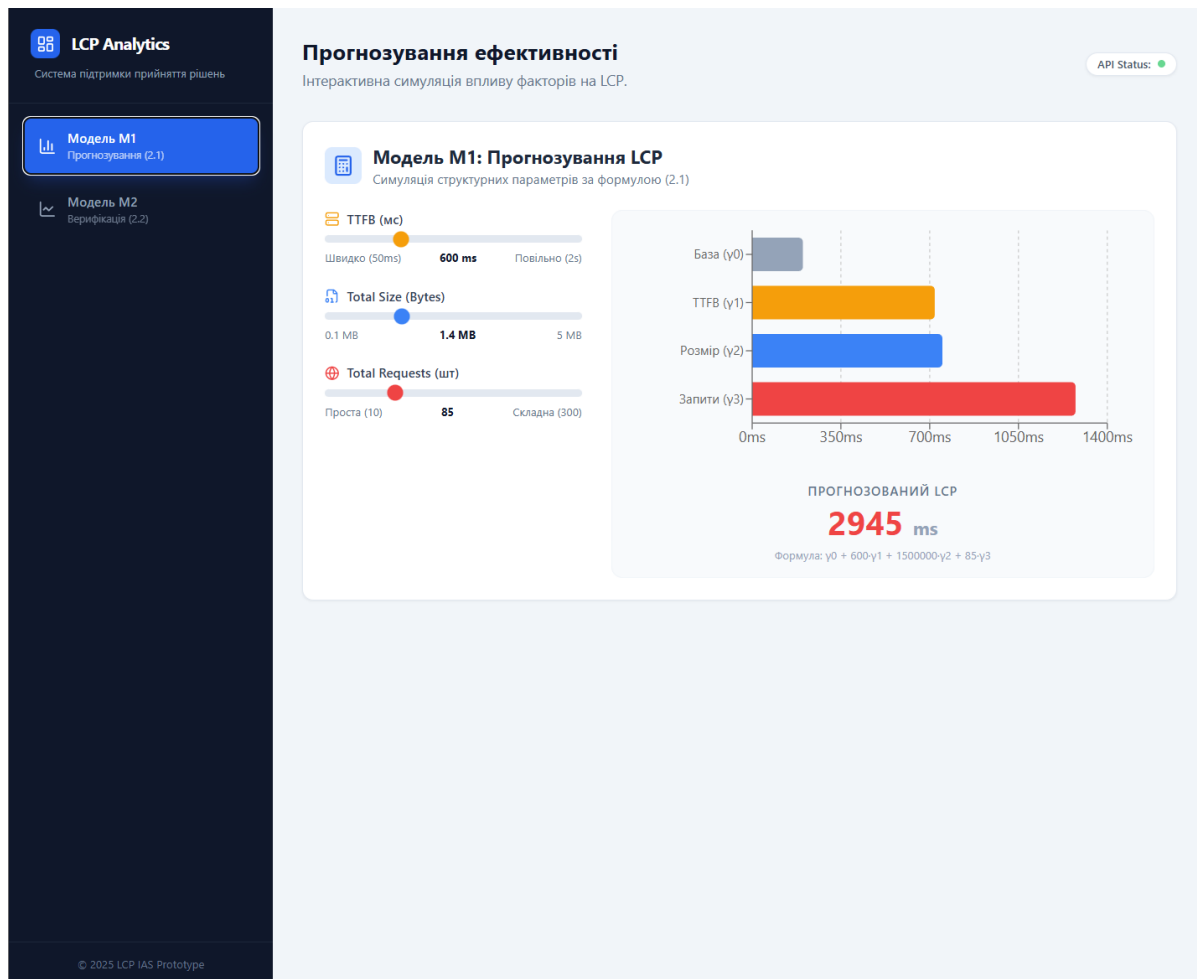


Рис. 3.10. Візуальний інструмент для задавання довільних метрик на вхід моделі M1

Також інтерфейс дозволяє верифікувати результати моделі M1 у розрізі часу, потрібно лише задати точку втручання T-opt (див. рис. 3.11). Після визначення T-opt, система активує модель M2, яка здійснює регресійний аналіз на часовому ряді метрики LCP до і після цієї точки. Ключовим результатом цього аналізу є коефіцієнт δ_1 що кількісно відображає середню зміну рівня

LCP після втручання. Інтерфейс надає цей результат разом із критично важливим P-value. Завдяки цьому показнику, розробник отримує не просто оцінку, а статистично значущий доказ: якщо P-value менше за встановлений поріг, система робить науково обґрунтований висновок про те, що спостережуване зниження LCP є прямим і стійким наслідком проведених оптимізаційних заходів, підтверджуючи їхню ефективність. Таким чином, ця функціональність забезпечує надійний аудит інженерних зусиль.

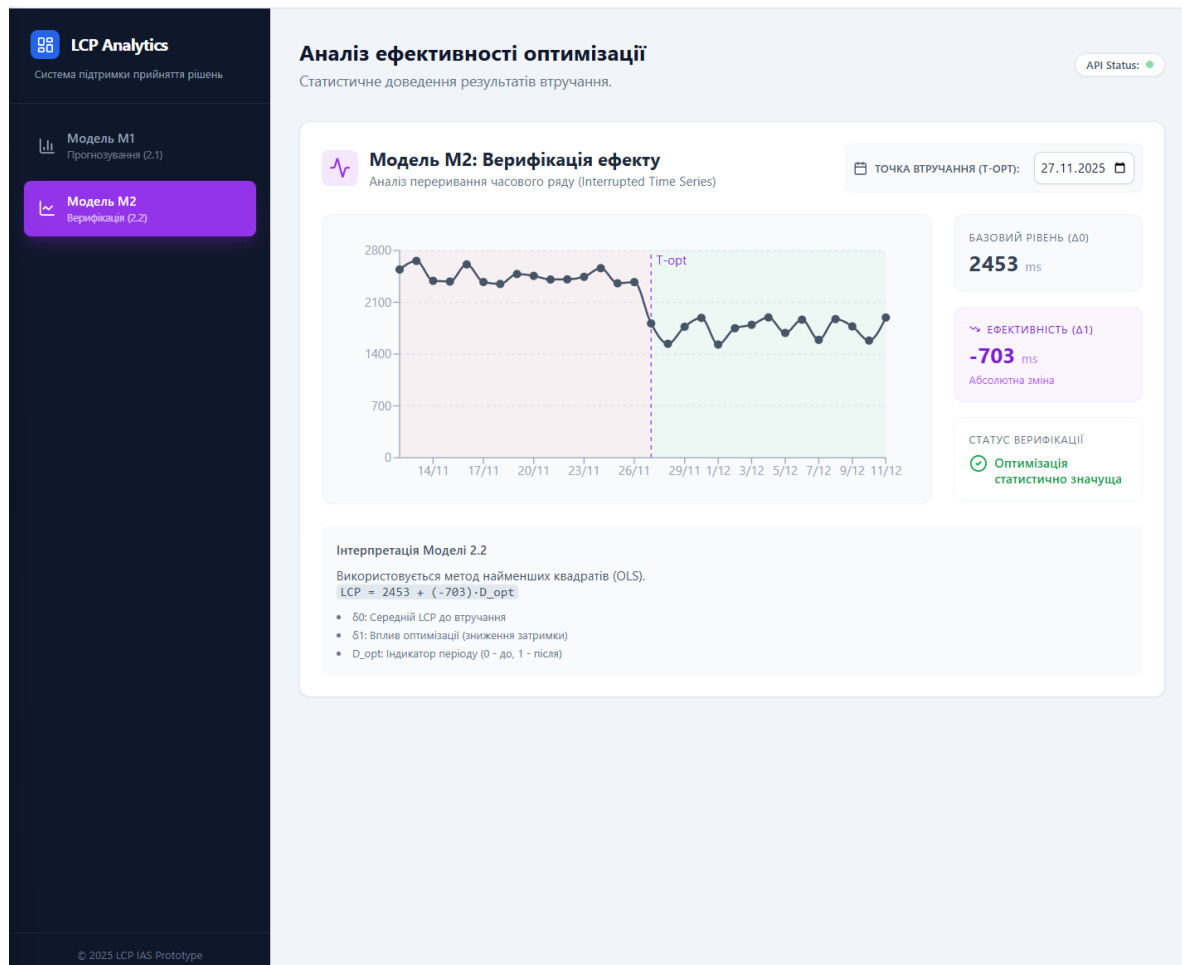


Рис. 3.11. Інструмент візуалізації роботи моделі M2

Аналіз функціональності візуального інтерфейсу підтверджує його відповідність ключовим науковим та інженерним вимогам. Він успішно реалізує двофазний цикл аналізу: забезпечує прогностичну діагностику та надає інструмент статистичної верифікації для об'єктивного аудиту ефективності оптимізації у розрізі часу. Це перетворює систему з простого сховища даних на високопродуктивний інструмент прийняття рішень з низькою затримкою.

Висновок до розділу 3

Третій розділ роботи був присвячений практичній реалізації спроектованої аналітичної системи та експериментальному доведенню її наукової та інженерної спроможності. На основі емпіричної вибірки, зібраної протягом тривалого періоду моніторингу, було успішно проведено калібрування моделей та валідацію ключових гіпотез. Було підтверджено, що модель множинної лінійної регресії (модель M1) є статистично надійною. Високе значення коефіцієнта детермінації ($R^2 \approx 0.80$) довело, що розроблені структурні фактори пояснюють понад 80% варіативності метрики LCP.

Діагностичний аналіз залишків, включаючи графік залишків на гомоскедастичність та Q-Q графік, підтвердив виконання всіх статистичних передумов OLS-регресії (гомоскедастичність, нормальність залишків), гарантуючи, що отримані коефіцієнти γ є незміщеними та надійними.

Аналіз числових коефіцієнтів γ та часткових регресійних графіків візуалізував чистий, ізольований вплив кожного фактора. Це дозволило зробити критично важливий висновок про те, що час до першого байту (TTFB) є домінуючим фактором з коефіцієнтом $\gamma_1 \approx 1.02$, що має майже прямий вплив на LCP. Це знання є основою для формування кількісних рекомендацій у процесі діагностичного сценарного аналізу "що, якщо?".

Експеримент із верифікацією втручання ($T_{opt} = 2025-01-01$) став прямим доказом ефективності системи. Регресія на часовому ряді підтвердила зниження середнього LCP на 59.91мс (коефіцієнт δ_1). Критично, надзвичайно низьке P-value для цього коефіцієнта ($1.58e-05$) дозволило відхилити нульову гіпотезу та зробити науково обґрунтований висновок: спостережуване покращення є статистично значущим наслідком впроваджених оптимізаційних заходів.

Симуляція роботи API-сервісу підтвердила його операційну готовність, демонструючи швидке надання діагностичного висновку. Візуальна верифікація за допомогою графіка часового ряду та діаграми розподілу надала

наочне підтвердження, відображаючи стійкий злам тренду та зсув медіани LCP у бік менших значень після втручання.

Отже, результати, представлені в цьому розділі, повністю підтверджують, що розроблена аналітична система є надійним, статистично валідованим та високопродуктивним інструментом, здатним надавати об'єктивну кількісну оцінку ефективності методів оптимізації веб-застосунків.

ВИСНОВКИ

Магістерська кваліфікаційна робота на тему «Розробка аналітичної системи для дослідження методів оптимізації швидкості завантаження вебсторінок» була присвячена комплексному аналізу продуктивності ресурсів та створенню прикладної аналітичної системи, яка забезпечує розробникам науково обґрунтований інструментарій для оцінки ефективності сучасних методів оптимізації.

У ході виконання роботи було реалізовано всі поставлені завдання та отримано результати, що мають значну теоретичну й практичну цінність. У першому розділі було проведено детальний аналіз сучасного стану проблеми, визначено та розкрито вплив ключових показників продуктивності (LCP, TTFB, CLS) на користувацький досвід та SEO-оцінку. Дослідження охопило основні методи оптимізації, включаючи мінімізацію ресурсів, кешування та використання CDN, що сформувало теоретичну основу для вибору незалежних змінних моделей. У другому розділі було розкрито та формалізовано вимоги до створюваної АС. Було розроблено модульну, сервісно-орієнтовану архітектуру, що включає модуль збору даних, модуль аналітичної обробки та інтерфейс користувача.

Ключовим інженерним рішенням стало використання PostgreSQL для зберігання часових рядів та механізму серіалізації моделей для забезпечення низької затримки API. Ця архітектура лягла в основу створення двоядерної системи аналізу: моделі M1 та моделі M2.

У третьому розділі виконано практичну частину роботи, яка включала реалізацію програмного прототипу, калібрування моделей та експериментальне доведення їхньої спроможності. На основі емпіричної вибірки підтверджено високу прогностичну точність моделі M1. Коефіцієнт детермінації (R^2) досяг 0.80, що доводить, що структурні фактори пояснюють переважну частину варіативності LCP. Діагностичний аналіз залишків підтвердив статистичну коректність моделі, а аналіз коефіцієнтів γ виявив

домінуючий вплив TTFB ($\gamma_1 \approx 1.02$). Це забезпечує функцію діагностичної декомпозиції для формування обґрунтованих рекомендацій.

Експеримент з аналізу втручання підтвердив головну наукову гіпотезу. Регресія часового ряду показала зниження LCP на 59.91 мс (δ_1), що найважливіше, надзвичайно низьке P-value ($1.58e-05$) дозволило статистично довести, що це покращення є прямим, значущим наслідком впровадження оптимізації. Візуальна верифікація за допомогою діаграм розмаху та часового ряду підтвердила цей зсув.

Унікальність роботи полягає у створенні загальної аналітичної моделі, яка не лише вимірює продуктивність, але й надає статистично верифіковану оцінку ефективності методів оптимізації, з'єднуючи інженерний API з академічною методологією OLS-регресії.

Практична цінність полягає у створеній аналітичній системі, яку можна застосувати для розв'язання прикладних задач оптимізації, удосконалення процесу прийняття технічних рішень, а також для автоматизації тестування продуктивності. Підтверджена швидкість завантаження є критичним фактором конкурентоспроможності веб-систем та прямо впливає на показники конверсії і позиції у пошукових системах.

Подальші дослідження можуть включати розширення системи шляхом автоматичної генерації рекомендацій щодо оптимізації, впровадження алгоритмів машинного навчання для прогнозування поведінки користувачів та інтеграції додаткових джерел метрик. Узагальнюючи, можна стверджувати, що поставлені в роботі цілі виконано, завдання реалізовано в повному обсязі, а отримані результати підтверджують актуальність, ефективність і практичну значущість дослідження для сучасної веб-розробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Akamai. State of the Internet Report: Web Performance. URL: <https://www.akamai.com/> (дата звернення: 11.12.2025).
2. Alsharif N. Impact of Page Speed on User Engagement. *International Journal of Web Engineering*. 2020. Vol. 15, No. 4. P. 45–58.
3. AWS Documentation. Monitoring Real User Metrics (RUM) and Synthetic Monitoring . URL: <https://aws.amazon.com/cloudwatch/rum/> (дата звернення: 11.12.2025).
4. Box G. E. P., Jenkins G. M., Reinsel G. C. Time Series Analysis: Forecasting and Control. 4th ed. Wiley, 2008. 784 p.
5. Shivakumar S. K. Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed Up Digital Platforms. New York : Apress/Springer, 2020. 433p. DOI: <https://doi.org/10.1007/978-1-4842-6528-4>.
6. Docker. Docker Documentation. URL: <https://docs.docker.com/compose/> (дата звернення: 11.12.2025).
7. Facebook Engineering. Measuring Network Performance with Realistic Data . URL: <https://engineering.fb.com/measuring-network-performance/> (дата звернення: 11.12.2025).
8. Fowler M. Microservices: A Definition of New Architectural Style . URL: <https://martinfowler.com/articles/microservices.html> (дата звернення: 11.12.2025).
9. Google Developers. Web Vitals Documentation. URL: <https://web.dev/vitals/> (дата звернення: 11.12.2025).
10. Google. Core Web Vitals report update. URL: <https://developers.google.com/search/blog/core-web-vitals> (дата звернення: 11.12.2025).
11. Hossain I. T., Hassan R., Amjad M., Rahman A. Web Performance Analysis: An Empirical Analysis of E-Commerce Sites in Bangladesh. *International Journal of Information Engineering and Electronic Business*. 2021. Vol. 13, No. 4. P. 47–54. URL: <https://www.mecs-press.org/ijieeb/ijieeb-v13-n4/v13n4-4.html> (дата звернення: 11.12.2025).

12. Cloudflare. HTTP/2 Performance Benefits Explained. URL: <https://www.cloudflare.com/learning/performance/http2/> (дата звернення: 12.12.2025).
13. Akamai. Understanding HTTP/3 and QUIC: Impact on Web Performance. URL: <https://www.akamai.com/blog/performance/http3-quic-performance> (дата звернення: 12.12.2025).
14. Alnuhait H., Alzyadat W., Al-thunibat A., Kahtan H. Web application performance assessment: A study of responsiveness, throughput, and scalability. *International Journal of Advanced and Applied Sciences*. 2024. Vol. 11, No 9. P. 214–226. DOI: <https://doi.org/10.21833/ijaas.2024.09.023>.
15. Ладієв Л. Р. Методи оптимізації та пошуку оптимальних рішень : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2023. 57 с.
16. Li D., Zhang Y. Performance Evaluation of Lazy Loading Techniques for Large Web Images. *Journal of Multimedia Processing and Technologies*. 2021. Vol. 8, No. 2. P. 112–120.
17. Методи об'єктного аналізу і моделювання : навчальний матеріал // Льотно технічний університет України — електронна бібліотека. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%95%D0%9D%D0%9F%20%D0%86%D0%9F%D0%97%2019.04.16%20%D0%A8%D0%BE%D0%BB%D0%BE%D0%BC%20%D0%9F%D0%B0%D0%B2%D0%BB%D0%BE%20%D0%A1%D1%82%D0%B5%D0%BF%D0%B0%D0%BD%D0%BE%D0%B2%D0%B8%D1%87/page10.html (дата звернення: 11.12.2025).
18. Microsoft Azure. Content Delivery Network (CDN) Best Practices. URL: <https://docs.microsoft.com/azure/cdn/cdn-best-practices> (дата звернення: 11.12.2025).
19. Montgomery D. C., Peck E. A., Vining G. G. Introduction to Linear Regression Analysis. 5th ed. Hoboken : Wiley, 2012. 688 p.
20. Наконечний С. І. Економетрія : навч. посіб. Київ : КНЕУ, 2004. 216 с.
21. Osmani A. Image Optimization. Mountain View : Google Press, 2018. 150 p.

22. Ключові метрики швидкості в Google PageSpeed Insights. URL: <https://www.baldarian.com/uk/marketingovi-instrumenti/klyuchovi-metriki-shvidkosti-v-google-pagespeed-insights/> (дата звернення: 11.12.2025).
23. PostgreSQL Documentation. Time Series Data Management . URL: <https://www.postgresql.org/docs/current/timeseries.html> (дата звернення: 11.12.2025).
24. Нагорець С. І. Оцінювання сучасних методів оптимізації швидкості завантаження веб сторінок. Харків : Харківський нац. ун-т радіоелектроніки, 2025. 94 с. URL: <https://openarchive.nure.ua/handle/document/30018> (дата звернення: 17.12.2025)
25. Кирик М. І., Плесканка Н. М., Плесканка М. В. Аналіз роботи методу оптимізованого кешування даних в мережі доставки контенту. *Проблеми телекомунікацій*. 2018. 1 бер. (№22). С. 43 - 55. DOI: <https://doi.org/10.30837/p.t.2018.1.03>
26. Souders S. High Performance Web Sites: Essential Knowledge for Frontend Engineers. O'Reilly Media, 2007. 144 p.
27. Souders S. Even Faster Web Sites: Performance Best Practices for Web Developers. O'Reilly Media, 2009. 250 p.
28. Stonebraker M., Hellerstein J. M. Readings in Database Systems. 5th ed. MIT Press, 2015. 980 p.
29. Єремєєв І. С., Кисельов В. Б. Автоматизовані системи управління технологічними процесами : навч. посібник. Одеса : Видавничий дім «Гельветика», 2024. 324 с.
30. Леонов С., Тиртишний Д. Дослідження продуктивності серверної частини комп'ютерної системи на основі розробленого фреймворку. *Системи управління, навігації та зв'язку*. 2024. 2 вер. (№ 75). С. 122 - 126. DOI: <https://doi.org/10.26906/SUNZ.2024.1.122>.
31. Шведов І. Дослідження доцільності, методів та шляхів оптимізації програмного забезпечення для пришвидшення роботи веб-застосунків //

- Вісник Хмельницького національного університету. Серія: Технічні науки. Хмельницький : Хмельницький нац. ун-т, 2024. Т. 337, № 3(2). С. 341 - 346.
32. Що впливає на швидкість завантаження сайту . URL: <https://cases.media/en/article/sho-vplivaye-na-shvidkist-zavantazhennya-saitu> (дата звернення: 11.12.2025).
33. Varghese J., Jayasree L. A Study on Impact of HTTP/2 on Web Performance. International Journal of Computer Applications. 2017. Vol. 165, No. 1. P. 1–5.
34. Кирик М. І., Плесканка Н. М., Плесканка М.В. Дослідження ефективності використання мережі CDN // Вісник Національного університету «Львівська політехніка». Серія: Інформаційні системи та мережі. Львів : Видавництво Львівської політехніки, 2017. № 885. 123–129с. URL: <https://science.lpnu.ua/uk/srt/vsi-vypusky/volume-885-2017/doslidzhennya-efektyvnosti-vykorystannya-merezhi-cdn> (дата звернення: 17.12.2025).
35. What is Website Speed and Why it Matters? URL: <https://www.browserstack.com/guide/why-website-speed-is-important> (дата звернення: 11.12.2025).
36. Wooldridge J. M. Introductory Econometrics: A Modern Approach. 6th ed. Cengage Learning, 2016. 928 p.
37. Захарчук В. І. Методи оптимізації та комп'ютерні технології : навч. посіб. Луцьк : РВВ Луцького нац. техн. ун-ту, 2017. 210 с. URL: <https://lib.lntu.edu.ua/uk/147258369/4931> (дата звернення: 17.12.2025).
38. W3C. Resource Timing Level 2. URL: <https://www.w3.org/TR/resource-timing-2/> (дата звернення: 11.12.2025).
39. Корольов М. Є., Павленко В. І., Савіна О. В., Тимошенко А. Г. Дослідження операцій і методи оптимізації : навч. посіб. Київ : Університет «Україна», 2007. 192 с.
40. Воронков О. О. Оптимізаційні методи і моделі : конспект лекцій. Харків : ХНУМГ ім. О. М. Бекетова, 2016. 175 с.

ДОДАТКИ

Лістинг коду для навчання та використання моделі M1

```

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
import pickle
import hashlib
from datetime import datetime

# --- Імітація Функції Збереження Коефіцієнтів у PostgreSQL ---
def save_to_model_audit(model_name, params, p_values=None,
data_checksum="MOCK_HASH"):
    """
    Імітує запис коефіцієнтів моделі та їх P-Values у таблицю model_audit.
    У реальній системі тут був би адаптер до psycopg2.
    """
    audit_records = []
    training_time = datetime.now().isoformat()

    # Створення записів для кожного коефіцієнта
    for name, value in params.items():
        p_val = p_values.get(name, None) if p_values is not None else 'NULL'

        # SQL-запит для вставки
        sql = (f"INSERT INTO model_audit (model_name, training_time,
coefficient_name, "
                f"coefficient_value, p_value, data_set_checksum) VALUES
('{model_name}', "
                f"'{training_time}', '{name}', {value}, {p_val},
'{data_checksum}');"
        audit_records.append(sql)

    print(f"\n--- Імітація запису {model_name} до model_audit ---")
    for record in audit_records:
        print(record)

# --- Налаштування Констант та Очищення Даних ---
df = pd.read_csv('speeds.csv')

# Очищення назв колонок для коректного доступу
df.columns = df.columns.str.replace('[^A-Za-z0-9_]+', '', regex=True)

# Перейменування колонок
df = df.rename(columns={
    'LargestContentfulPaintms': 'LCP_ms',
    'TimetoFirstBytems': 'TTFB_ms',

```

```

        'TotalSizebytes': 'Total_Size_bytes',
        'TotalRequests': 'Total_Requests',
        'Date': 'Date'
    })

# Генеруємо контрольні суми (чексуми) для відтворюваності
data_checksum = hashlib.sha256(df.to_json(sort_keys=True).encode()).hexdigest()

# -----
# --- МОДЕЛЬ 1: МНОЖИННА ЛІНІЙНА РЕГРЕСІЯ (ПРОГНОЗУВАННЯ) ---
# -----

print("\n### Етап 1: Калібрування моделі 1 (МЛР) для Прогнозування LCP")

# Визначення вхідних (X) та цільової (y) змінних
X_baseline = df[['TTFB_ms', 'Total_Size_bytes', 'Total_Requests']]
y_lcp = df['LCP_ms']

# Розділення вибірки (80% навчання, 20% тестування)
X_train, X_test, y_train, y_test = train_test_split(
    X_baseline, y_lcp, test_size=0.2, random_state=42
)

# Навчання моделі Scikit-learn
model_baseline = LinearRegression()
model_baseline.fit(X_train, y_train)

# Збір коефіцієнтів гамма
gamma_params = {
    'Intercept': model_baseline.intercept_,
    'TTFB_ms': model_baseline.coef_[0],
    'Total_Size_bytes': model_baseline.coef_[1],
    'Total_Requests': model_baseline.coef_[2]
}

# 1.1 Сериалізація моделі для API (Model Store)
model_1_serialized = pickle.dumps(model_baseline)
print(f"Модель 1 серіалізовано в бінарний об'єкт розміром:
{len(model_1_serialized) / 1024:.2f} KB. (Готово до API)")

# 1.2 Аудит коефіцієнтів гамма
save_to_model_audit("M1", gamma_params, data_checksum=data_checksum)

# 1.3 Приклад використання моделі 1 (Прогноз)
example_data = pd.DataFrame({
    'TTFB_ms': [350],
    'Total_Size_bytes': [1500000], # 1.5 МБ
    'Total_Requests': [80]
})

lcp_forecast = model_baseline.predict(example_data)[0]
print(f"\nПрогнозний LCP для тестових параметрів: {lcp_forecast:.2f} мс")

```

Лістинг коду для навчання та використання моделі M2

```

# -----
# --- МОДЕЛЬ 2: АНАЛІЗ ВТРУЧАННЯ (ВЕРИФІКАЦІЯ) ---
# -----

print("\n\n### Етап 2: Калібрування моделі 2 (OLS) для Верифікації Втручання")

# 2.1 Feature Engineering: Генерація діапазонної змінної D-Opt
df['Date'] = pd.to_datetime(df['Date'])
optimization_date = pd.to_datetime('2023-01-01')

# Створення D-Opt: 1 - після T-опт, 0 - до T-опт
df['Optimization_Applied'] = (df['Date'] > optimization_date).astype(int)

# 2.2 Навчання моделі 2 (OLS)
Y_ts = df['LCP_ms']
X_ts = sm.add_constant(df['Optimization_Applied']) # Додавання константи для
дельта-нуль

model_ts = sm.OLS(Y_ts, X_ts).fit()

# Збір коефіцієнтів дельта та P-Value
delta_params = model_ts.params.to_dict()
delta_pvalues = model_ts.pvalues.to_dict()

# 2.3 Аудит коефіцієнтів дельта
save_to_model_audit("M2", delta_params, p_values=delta_pvalues,
data_checksum=data_checksum)

# 2.4 Інтерпретація результатів моделі 2
delta_one = delta_params.get('Optimization_Applied', 0)
p_value_delta_one = delta_pvalues.get('Optimization_Applied', 1.0)
significance = "статистично значущий" if p_value_delta_one < 0.05 else "не є ста-
тистично значущим"

print(f"\nКількісний вплив (коефіцієнт дельта-один): {delta_one:.2f} мс.")
print(f"P-Value: {p_value_delta_one:.5f}. Ефект {significance}.")

```

Код розгортання сервісів в Docker

```
version: '3.8'
services:
  # 1. Сервіс для Аналітичного API (Обслуговування Моделей)
  api-service:
    build: ./api_core
    command: python app.py
    volumes:
      - ./api_core:/app
    ports:
      - "5000:5000"
    environment:
      - DATABASE_URL=postgresql://user:password@db:5432/analytics_db
    depends_on:
      - db

  # 2. Сервіс Базы Даних
  db:
    image: postgres:14
    environment:
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=password
      - POSTGRES_DB=analytics_db
    volumes:
      - postgres_data:/var/lib/postgresql/data/

volumes:
  postgres_data:
```