Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка WEB сервісу для аналізу ринку вживаних авто України»
(тема роботи)

Виконав: студент 2-ого курсу групи 1ІСТ-22м
(шифр групи)
спеціальності 126 – Інформаційні системи та технології
(шифр та назва спеціальності)
Горбань Денис Андрійович
(ПІБ студента)
Керівник: к.т.н., проф. Євген ПАЛАМАРЧУК
(науковий ступінь, вчене звання/посада, ПІБ керівника)
«4» грудня 2023 р.
Опонент: д.т.н., проф. Володимир ДУБОВОЙ
(науковий ступінь, вчене звання/посада, ПІБ опонента)
«8» грудня 2023 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
(науковий ступінь, вчене звання)
«11» грудня 2023 р.

Вінниця, ВНТУ – 2023 р.

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти II-ий (магістерський)

Галузь знань – 12 – Інформаційні технології

Спеціальність – 126 – Інформаційні системи та технології

Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ

Завідувач кафедри АІІТ

д.т.н., проф. Олег БІСІКАЛО

«20» вересня 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Горбаню Денису Андрійовичу

(ПІБ автора повністю)

1. Тема роботи: Розробка WEB сервісу для аналізу ринку вживаних авто України

   Керівник роботи: к.т.н., проф. Євген ПАЛАМАРЧУК

   Затверджені наказом ВНТУ від «18» 09 2023 року № 247

2. Строк подання роботи студентом: до «05» 12 2023 року
3. Вихідні дані до роботи: стек технологій обробки даних, джерело даних, застосунок для побудови діаграм, ноутбук, середовище розробки PyCharm.
4. Зміст текстової частини: вступ; сучасний стан питання та основні задачі роботи; технології розробки аналітичних сервісів; розробка аналітичного сервісу; тестування створеного функціоналу; економічний розділ; висновки.

5. Перелік ілюстративного (або графічного) матеріалу: життєвий ци
інженерії даних; діаграма ETL конвеєру; діаграма ELT конвеєру; діагра
архітектури сховища даних за схемою Star; діаграма архітектури сховищ
даних за схемою Snowflake; діаграма архітектури з використанням Da
Lake; діаграма структури сховища даних; діаграма етапу збору дани
діаграма етапу доставки даних; діаграма етапу обробки даних; загаль
архітектура конвеєру даних.

6. Консультанти розділів магістерської кваліфікаційної роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-5	к.т.н., проф. Євген ПАЛАМАРЧУК	20.09.23	05.12.23

7. Дата видачі завдання: «20» вересня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	02.10.23	виконано
2	Аналіз та вибір інформаційних технологій обробки даних	09.10.23	виконано
3	Програмна реалізація	14.11.23	виконано
4	Тестування створеного функціоналу	30.11.23	виконано
5	Підготовка економічної частини	02.12.23	виконано
6	Оформлення матеріалів до захисту МКР	05.12.23	виконано

Студент _____ Денис ГОРБАНЬ
(підпис) (прізвище та ініціали)

Керівник роботи _____ Євген ПАЛАМАРЧУК
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.341.1

Горбань Д.А. Розробка WEB сервісу для аналізу ринку вживаних авто України. Магістерська кваліфікаційна робота зі спеціальності 126 – Інформаційні системи та технології, освітньо-професійна програма – Інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2023. 117 с.

На укр. мові. Бібліогр.: 30 назв; рис.: 33; табл.: 13.

У роботі досліджено питання аналітичних інструментів дослідження ринку вживаних автомобілів в Україні. За результатами дослідження було розроблено аналітичний веб сервіс, що має надати користувачам можливість ознайомлюватися з ринком авто в агрегованій та ілюстрованій формі. Для побудови були використані такі складові частини як ELT конвеєр даних, сховище даних та інтерактивний дашборд.

Ключові слова: веб сервіс, аналітична платформа, ETL/ELT конвеєр даних, сховище даних, візуалізація даних.

ABSTRACT

Horban D.A. Development of a WEB service for the analysis of the used car market of Ukraine. Master's thesis in specialty 126 - Information systems and technologies, educational and professional program - Information technologies of data and image analysis. Vinnytsia: VNTU, 2023. 117 p.

The paper examines the issue of analytical tools for researching the used car market in Ukraine. Based on the results of the research, an analytical web service was developed, which should provide users the opportunity to familiarize themselves with the car market in an aggregated and illustrated form. Such components as ELT pipeline, data storage and interactive dashboard were used for construction.

Keywords: web service, analytical platform, ETL/ELT pipeline, data storage, data visualization.

ЗМІСТ

ВСТУП 5

1 СУЧАСНИЙ СТАН ПИТАННЯ ТА ОСНОВНІ ЗАДАЧІ РОБОТИ 7

1.1 Аналіз предметної області 7

1.2 Дослідження існуючих рішень 10

1.3 Методологія розробки сервісу для аналізу даних 13

1.4 Постановка задач розробки 17

1.5 Висновки до розділу 18

2 ТЕХНОЛОГІЇ РОЗРОБКИ АНАЛІТИЧНИХ СЕРВІСІВ 20

2.1 Дослідження технологій обробки даних 20

2.2 Аналіз та вибір основних засобів розробки 32

2.3 Висновки до розділу 43

3 РОЗРОБКА АНАЛІТИЧНОГО СЕРВІСУ 46

3.1 Планування розробки сервісу 46

3.2 Дослідження джерела даних 46

3.3 Побудова структури сховища даних 52

3.4 Аналіз необхідних перетворень даних 56

3.5 Розробка конвеєру обробки даних 57

3.6 Інтеграція та побудова інтерактивних дашбордів 62

3.7 Висновки до розділу 67

4 ТЕСТУВАННЯ СТВОРЕНОГО ФУНКЦІОНАЛУ 69

4.1 Функціональне тестування 69

4.2 Нефункціональне тестування 70

4.3 Структурне тестування 71

4.4 Тестування змін 71

4.5 Тестування розробленого функціоналу 72

4.6 Висновки до розділу 75

5 ЕКОНОМІЧНИЙ РОЗДІЛ 76

5.1 Технологічний аудит розробленого веб сервісу для аналізу ринку вживаних авто України 76

5.2 Розрахунок витрат на розробку аналітичного сервісу 82

5.3 Розрахунок економічного ефекту від можливої комерціалізації розробки 87

ВИСНОВКИ 95

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 97

ДОДАТКИ 100

Додаток А (обов'язковий) Технічне завдання 101

Додаток Б (обов'язковий) Ілюстративна частина 104

Додаток В (обов'язковий) Лістинг програмного забезпечення 108

Додаток Г (обов'язковий) Протокол перевірки магістерської кваліфікаційної роботи на наявність текстових запозичень 117

ВСТУП

Актуальність теми. У сучасному світі все більше людей приймають рішення про купівлю автомобіля, керуючись не лише емоціями та зовнішнім виглядом авто, але й науково-обґрунтованими даними, такими як технічні характеристики, склад та функціональні показники. Купівля автомобіля є значним фінансовим вкладенням і має тривалий термін використання, тому вона вимагає детального аналізу як окремих моделей та їх характеристик, так і ринку в цілому. Пропозиції з ринку вживаних автомобілів в Україні можна бачити на таких онлайн-платформах, як AUTO.RIA, RST та інші. Проте, відсутність інструментів для загального аналізу ринку та візуалізації метрик обмежує можливості покупців зробити обґрунтований вибір та правильно оцінити загальний стан ринку. Тому, створення окремого сервісу стає актуальним завданням для забезпечення доступу до об'єктивної інформації та покращення процесу прийняття рішення про купівлю автомобіля.

Мета і задачі дослідження. Метою роботи є розробка концепції та архітектури сервісу, який запропонує потенційним покупцям можливість ознайомитись з реаліями ринку вживаних автомобілів в Україні та зробити обґрунтований вибір при купівлі.

Для досягнення поставленої мети потрібно розв'язати такі задачі дослідження:

1.Провести аналіз предметної області та виявити функціональності, необхідні для реалізації поставленої мети.

2.Створити загальну архітектуру системи.

3.Визначити технологічні рішення, що будуть застосовані для реалізації.

4.Проробити детальну архітектуру проекту.

5.Розробити програмне забезпечення сервісу та провести його експериментальну апробацію.

6.Проаналізувати економічну складову розробки такого сервісу.

Об'єктом дослідження є ринок вживаних автомобілів в Україні.

Предметом дослідження є створення сервісу, який надає користувачам доступ до агрегованих метрик та даних ринку вживаних автомобілів в Україні.

Методи дослідження. В роботі використані аналітичні методи оброблення інформації та методи математичної статистики.

Практична цінність роботи полягає в тому, щоб надати доступ до агрегованих метрик та даних ринку, що допоможе користувачам приймати обґрунтовані рішення щодо вибору автомобіля. Крім того, розроблений сервіс може стати основою для подальшого розвитку і розширення функціональностей, що може сприяти розвитку індустрії торгівлі вживаними авто в Україні.

Апробація результатів та публікації. За результатами даної роботи було опубліковано тези доповіді на всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» [1].

1 СУЧАСНИЙ СТАН ПИТАННЯ ТА ОСНОВНІ ЗАДАЧІ РОБОТИ

1.1 Аналіз предметної області

Автомобільний транспорт є одним з найважливіших секторів економіки України. Він забезпечує транспортне сполучення між населеними пунктами, сприяє розвитку торгівлі та туризму, а також створює робочі місця.

Ринок автомобілів в Україні складається з двох основних сегментів: нових і вживаних автомобілів. Ринок нових авто представлений автомобілями, що випускаються вітчизняними та іноземними виробниками, і представлені для покупців переважно у автосалонах. Вторинний ринок представлений автомобілями, які були в експлуатації протягом певного часу та можуть бути придбані у власника за помірну ціну на авторинках або через спеціалізовані онлайн-майданчики.

З початку 2022 року і до 1 грудня 2022 року українці придбали 642,3 тисячі легковиків з пробігом на внутрішньому ринку, ще 378 тисячі привезли вживаними з-за кордону. Таким чином, об’єм ринку у 2022 році складає близько 1,1 мільйона угод купівлі-продажу вживаних машин: майже 700 тисяч всередині країни, та понад 400 — імпортних. В Додатку А наведено детальну динаміку ринку автомобілів в Україні впродовж 2022 року [2].

Найпопулярнішими на ринку вживаних авто стали: Volkswagen Passat, ZAZ Lano/Sens, Skoda Octavia, Volkswagen Golf, Renault Megane, Ford Focus, ВАЗ 2109/99, Skoda Fabia, Opel Astra, BMW 5 Series [3].

Український авторинок нових легкових автомобілів минулого року впав на 61,5% порівняно з довоєнним 2021 роком — до близько 40 тисяч штук. Цей показник є найгіршим з 2000 року [4].

Найпопулярнішими моделями на ринку нових авто стали: Toyota RAV4,

Renault Duster, Kia Sportage, Skoda KODIAQ, Toyota Land Cruiser Prado, Volkswagen Touareg, Hyundai Tucson, Kia Ceed, Mitsubishi Outlander, Toyota Camry [5].

Класичні українські авторинки та автосалони поступово переходять в інтернет. Кількість відвідувачів спеціалізованих сайтів росте з кожним роком — і тих, хто купує, і тих, хто продає.

Можна перелічити основні причини, чому українці все частіше купують авто онлайн:

– Величезна кількість пропозицій з усієї України та їх цілодобова доступність.

– Висока конкуренція серед продавців, як наслідок — збалансовані ціни.

– Можливість переглядати оголошення з будь-якого місця та легко спілкуватися через чат на сайті або по телефону.

– Можливість фільтрування оголошень по регіону та місту, що значно полегшує пошук варіантів які знаходяться поруч.

В Україні представлені наступні сайти, що спеціалізуються на продажу та купівлі авто:

– auto.ria.com

– autobazar.ua

– rst.ua

– cars.ua

Провівши аналіз перерахованих вище сайтів по критерію відвідуваності за допомогою сервісу SimilarWeb, можна визначити найпопулярніший онлайн-майданчик для купівлі/продажу авто – AutoRia. На рисунку 1.1 можна бачити що AutoRia займає частку у 82% від сукупних переглядів цих 4 сайтів за останні 3 місяці.

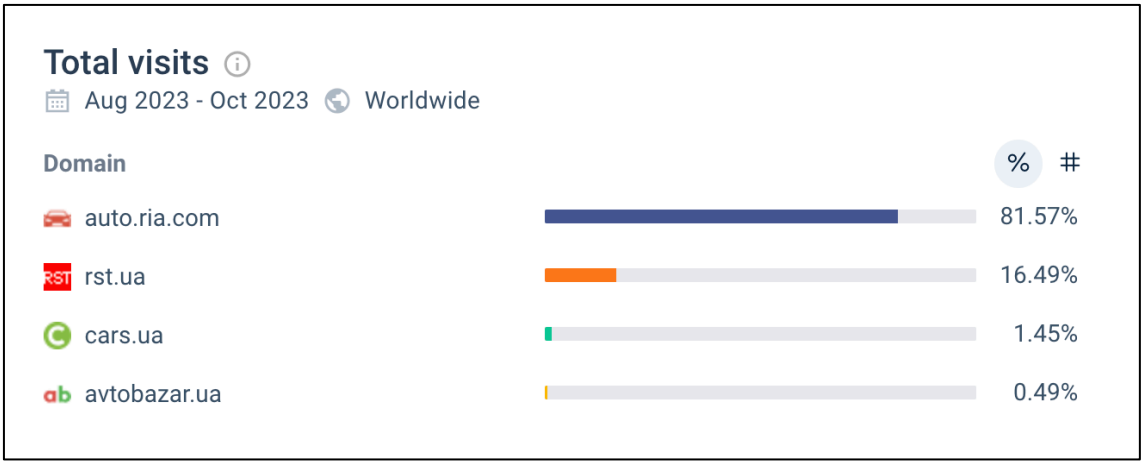


Рисунок 1.1 – Частки відвідувань серед наведених онлайн-майданчиків.

З помісячного графіку відвідувань веб-сторінок (рисунок 1.2), також видно значну перевагу AutoRia за популярністю в порівнянні з конкурентами. За останні 6 місяців, AutoRia відвідало приблизно 119 мільйонів користувачів, в той час як найближчий конкурент – RST – має тільки 25 мільйонів.

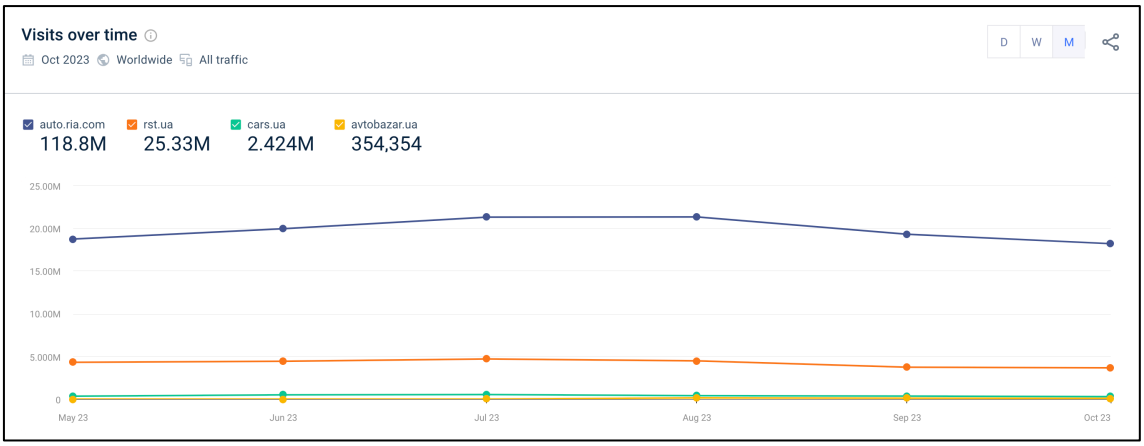


Рисунок 1.2 – Кількість відвідувань веб-сторінок авто-майданчиків помісячно за період Травень – Жовтень 2023 року.

За добу на AutoRia заходить близько півмільйона відвідувачів, а база оголошень про продаж вживаних авто становить майже 300 000 позицій. В день здійснюється до 1 500 угод купівлі-продажу [6].

Отже, за сукупністю наведених вище показників AutoRia є найпопулярнішим онлайн-майданчиком для купівлі/продажу авто в Україні.

1.2 Дослідження існуючих рішень

На сьогоднішній день, сервіс AutoRia є лідером у сфері купівлі/продажу авто в Україні та надає такі функції своїм користувачам:

– Пошук авто. На сайті можна шукати як пропозиції нових авто від автосалонів, так і вживані авто від власників або салонів. Пошук авто є зручним та пропонує багато опцій для фільтрування оголошень, що допоможе підібрати такі авто, які будуть максимально цікаві користувачу. Можна виділити такі основні опції фільтрів: тип транспорту, тип кузова, країна виробник, марка, модель, рік випуску, ціна, регіон, тип палива, тип КПП, об’єм двигуна, пробіг. Приклад інтерфейсу пошуку авто зображено на рисунку 1.3 [7].

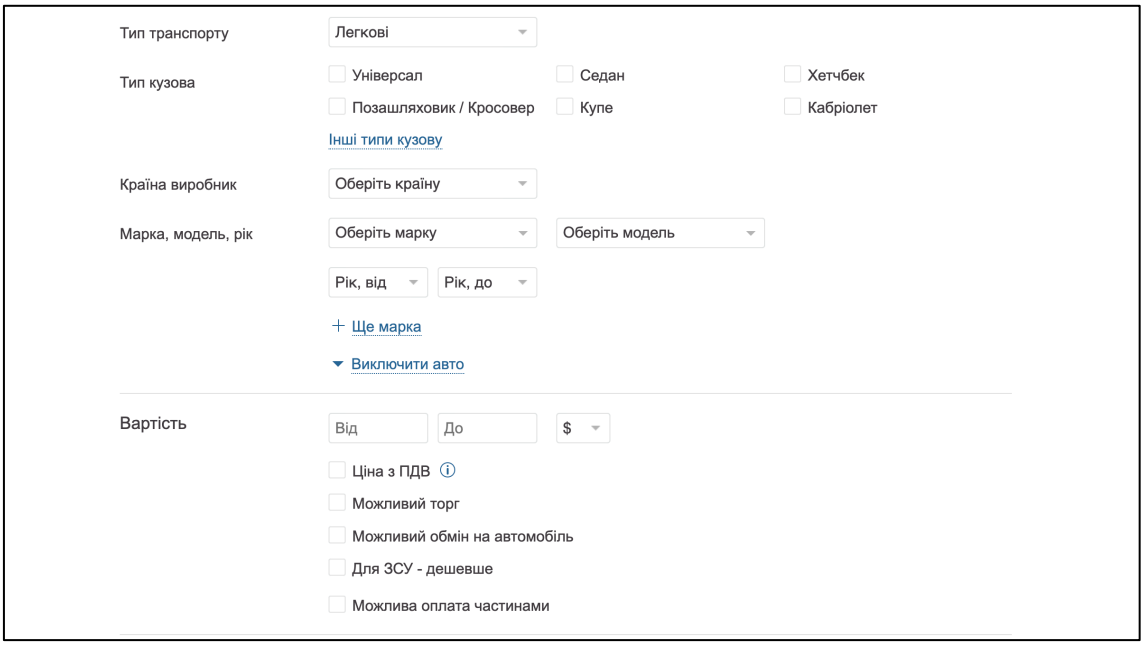


Рисунок 1.3 – Фрагмент інтерфейсу пошуку авто.

– Розміщення оголошень. Зареєстровані користувачі можуть публікувати оголошення про продаж авто. Сайт надає зручний інтерфейс

створення оголошення. Користувач може додати опис, фото та заповнити відповідні характеристики авто. Також, за додаткову плату, можна просувати своє оголошення, для того щоб його побачила більша кількість людей. Приклад інтерфейсу додавання оголошення зображено на рисунку 1.4 [7].

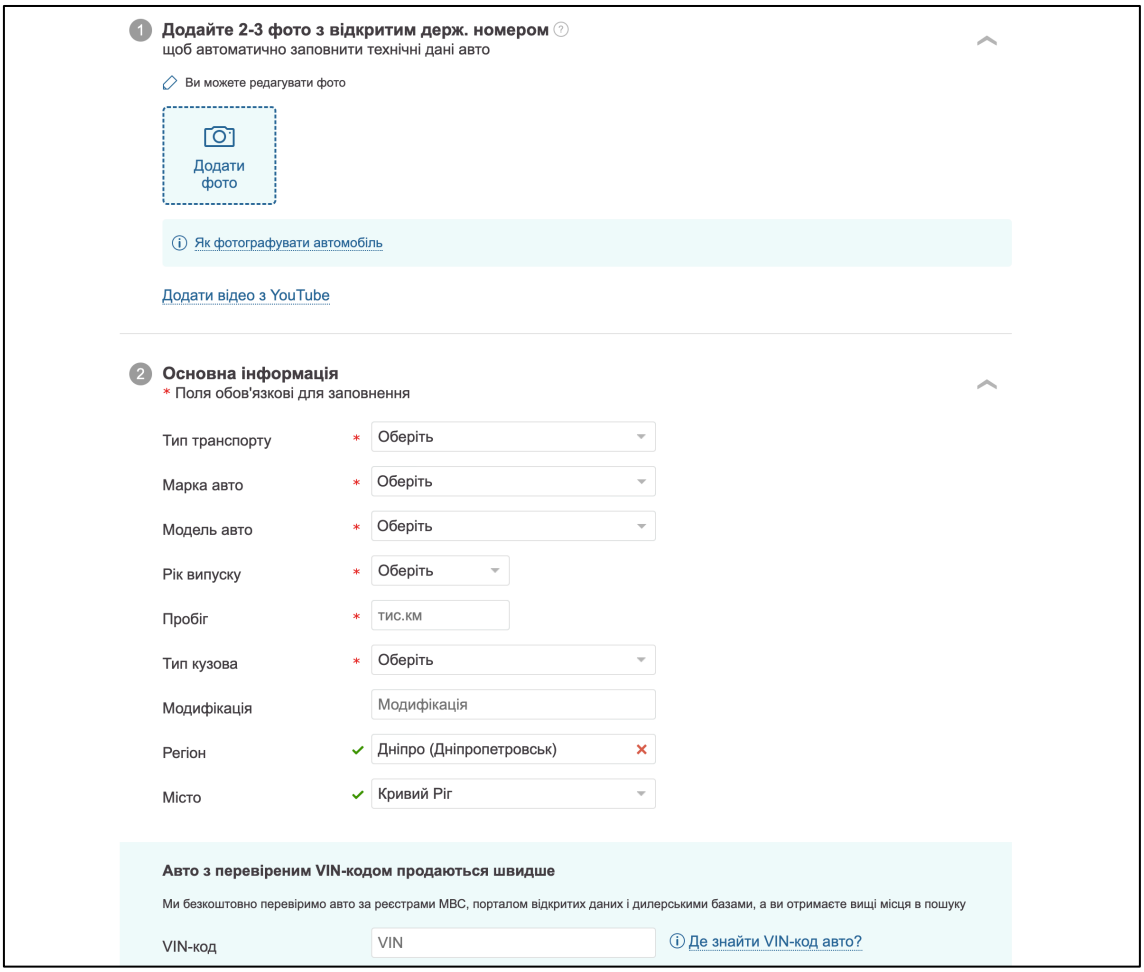


Рисунок 1.4 – Фрагмент інтерфейсу додавання оголошення.

– Калькулятор середньої ціни. Дуже корисна функціональність для тих, хто хоче дізнатися скільки коштує його автомобіль. Також, калькулятор середньої ціни можна використовувати для того, щоб дізнатися середню ціну на конкретне авто, що цікавить. Це один з небагатьох аналітичних інструментів, яким можуть скористатися користувачі. Приклад інтерфейсу калькулятору середньої ціни зображено на рисунку 1.5 [7].

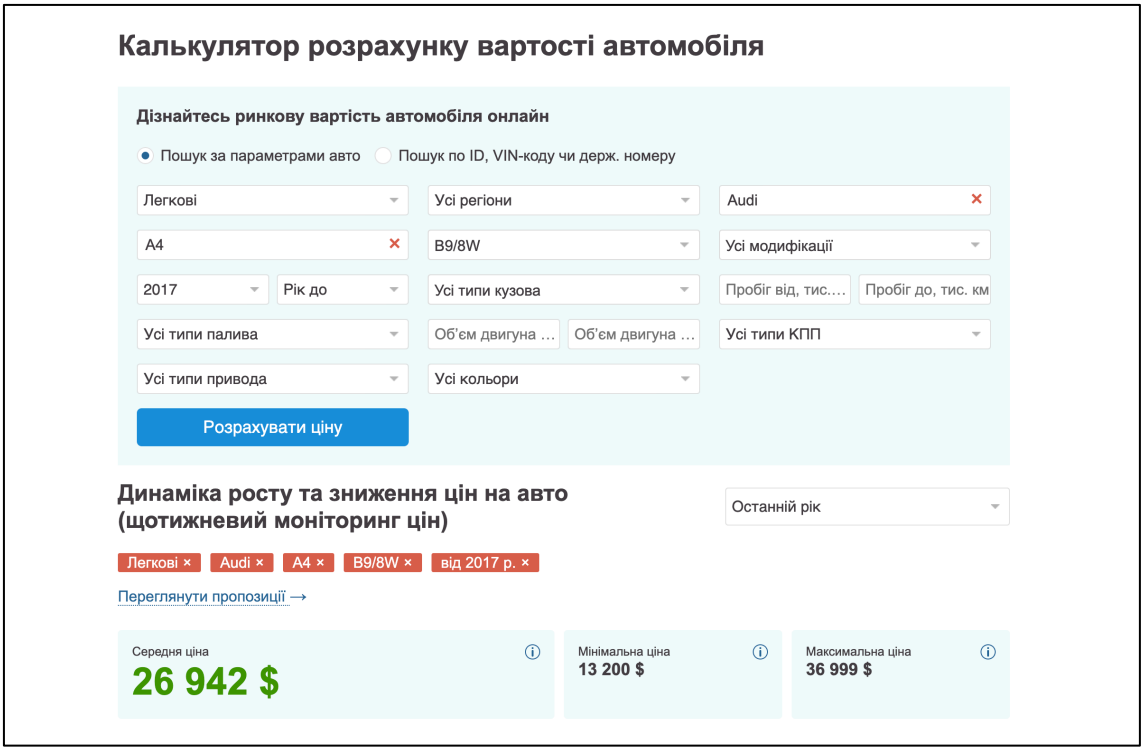


Рисунок 1.5 – Фрагмент інтерфейсу калькулятору середньої ціни.

– Інше. Сайт також надає доступ до каталогу товарів для авто, СТО, автосалонів, а також має окрему сторінку, де автовласники можуть залишити відгук на свою модель авто. Приклади інших доступних сервісів зображено на рисунку 1.6 [7].

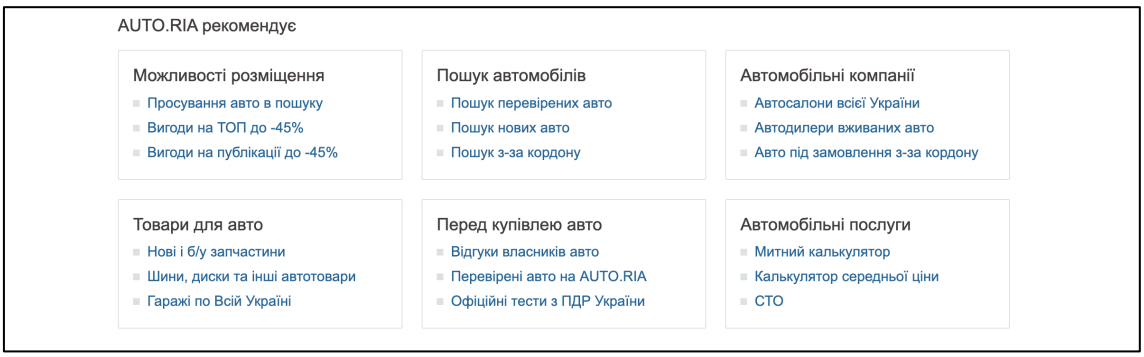


Рисунок 1.6 – Інші доступні сервіси на сайті AutoRia.

Як можна бачити функціональність сайту AutoRia залишається в межах онлайн-майданчику для продажу(купівлі) авто, а також інших супутніх товарів і

послуг. Єдиною аналітичною функціональністю можна вказати калькулятор середньої ціни, але цей інструмент дуже вузько специфікований і, наприклад, не підходить для того випадку, коли користувач не знає що він конкретно хоче, а просто має певні критерії як ціна, рік випуску, пробіг тощо.

В ході цілеспрямованого пошуку у мережі інтернет аналітичних інструментів для ринку вживаних авто України не вдалося знайти існуючі рішення. У зв'язку з цим постає проблема, а саме - відсутність інтерактивних статистичних або аналітичних інструментів дослідження ринку авто України.

Коли постає питання купівлі авто, людина точно має декілька критеріїв для вибору – це може бути ціна, пробіг, тип КПП або палива тощо. Не всі люди знаються на машинах, тому критерій моделі або марки може бути відсутнім. Маючи вищезгадані вхідні дані, на даний момент, єдиний варіант самостійного дослідження ринку авто буде перегляд тисяч оголошень з зазначеними критеріями для фільтрації. І це все просто для того, щоб зрозуміти які моделі, та з якими характеристиками доступні в рамках вказаних користувачем обмежень. Такий підхід потребує багато часу та натхнення для перегляду такої кількості оголошень, що є неефективним.

Виходом із цієї ситуації може бути звернення до консультантів, які знаються на ринку і можуть підібрати варіанти відповідно до побажань клієнта. Проте це все ще не дає гарантій оптимального вибору, адже консультант може нав'язувати свої вподобання, мати інтерес продати якесь конкретне авто тощо.

Наявність веб-сервісу для аналізу ринку авто України з можливістю огляду статистики в розрізі по різних критеріях суттєво покращило б процес вибору авто та зробило б його більш інформаційно підкріпленим.

1.3 Методологія розробки сервісу для аналізу даних

Процес розробки рішень спрямованих на збирання, обробку та аналіз даних прийнято називати дата інжинірингом (з англ. *Data Engineering*). Такого

роду рішення дають людям можливість знаходити практичне застосування даних.

Без заглиблення у технологічні аспекти, процес дата інжинірингу можна пояснити за допомогою такої абстракції як життєвий цикл інженерії даних (рисунок 1.7) [8].

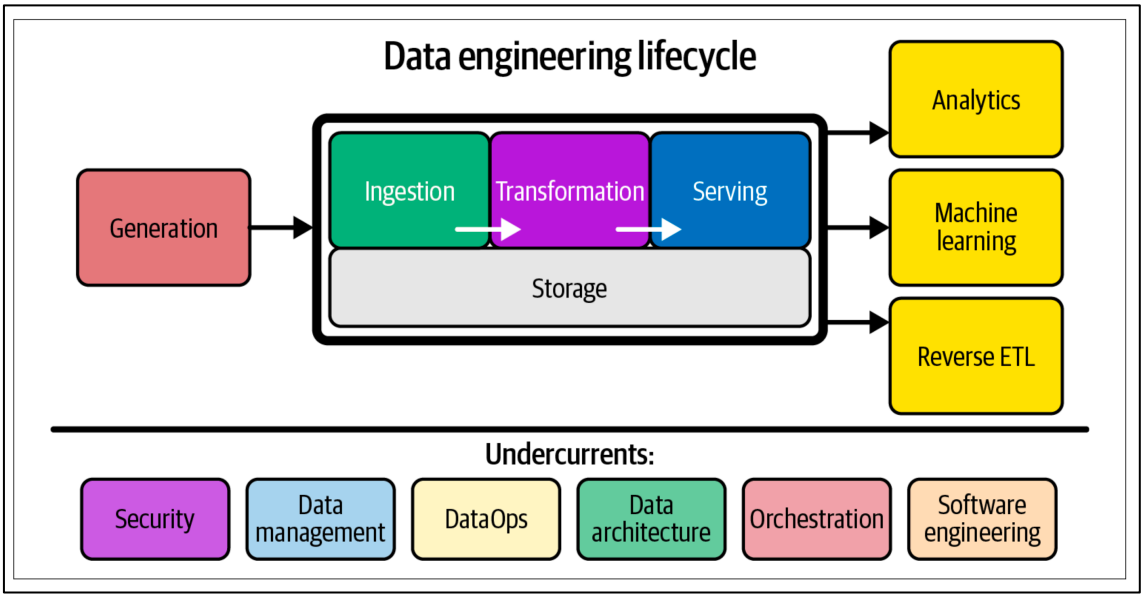


Рисунок 1.7 – Життєвий цикл інженерії даних.

Початкова фаза життєвого циклу інженерії даних характеризується процесом генерації даних. На цьому етапі необроблені дані або створюються в екосистемі організації, або надходять із зовнішніх джерел. Основна увага приділяється створенню моделей даних, що можуть бути як у структурованій, так і у неструктурованій формі. Генерація даних є основою для подальшої обробки і трансформації даних.

Після фази генерації життєвий цикл переходить до завантаження даних. Цей етап передбачає збирання та завантаження даних із різних джерел у місце тимчасового зберігання даних. Сервіси, розгорнуті для збору даних, можуть працювати за пакетною або потоковою схемою, залежно від часових вимог до

обробки даних. Збір та завантаження забезпечують безперервний потік даних, який буде оброблено на наступних етапах.

Наступним етапом життєвого циклу інженерії даних є трансформація, тобто дані повинні бути змінені з їх початкової форми на щось корисне для подальших сценаріїв використання. Без належних перетворень дані залишатимуться інертними та не будуть корисними для звітів, аналізу чи ML. Як правило, на етапі трансформації дані починають створювати цінність для кінцевих користувачів.

Останньою фазою життєвого циклу інженерії даних є доставка даних. На цьому етапі оброблені дані стають доступними для споживання кінцевими користувачами або подальшими програмами. Доставка включає: надання даних через визначені інтерфейси; забезпечення доступності даних; оперативне реагування на аналітичні запити користувачів. Ефективність етапу доставки залежить від якісної інтеграції оброблених даних в екосистему організації, що має сприяє прийняттю швидких та обґрунтованих рішень, а також покращувати стратегічне розуміння стану бізнесу.

Дані мають цінність, коли вони використовуються в практичних цілях. Аналітика є найбільш розповсюдженою метою побудови систем обробки даних в організаціях. Після того як дані зібрано та оброблено, їх можна використовувати для створювання звітів або дашбордів, а також для проведення точкового аналізу даних. У той час як основна частина аналітики раніше охоплювала BI (з англ. *Business Intelligence*), тепер вона включає й інші аспекти, такі як операційна аналітика та вбудована аналітика (рисунок 1.8) [8].

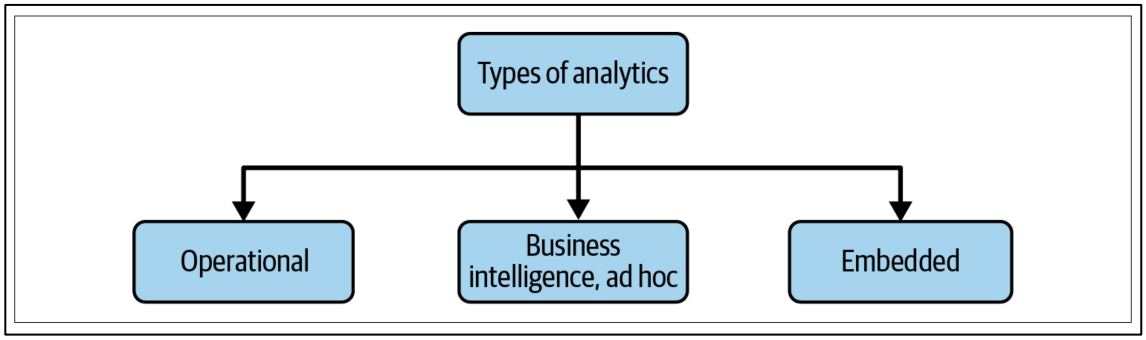


Рисунок 1.8 – Типи аналітики.

Спеціалісти з BI збирають дані, щоб описати минулий і поточний стан бізнесу. BI вимагає використання бізнес-логіки для представлення вхідних даних. Бізнес-логіка часто застосовується до даних на етапі трансформації життєвого циклу розробки даних, але підхід застосування бізнес-логіки при використані даних стає все більш популярним. Для цілей BI, дані зберігаються в чистому, але досить необробленому вигляді. Система BI всередині має репозиторій з бізнес-логікою, яка використовується для запитів до сховища даних, щоб звіти та інформаційні панелі відповідали бізнес-вимогам.

Операційна аналітика зосереджується на дрібних операційних деталях бізнесу, сприяючи швидкому реагуванню на ті чи інші показники. Операційна аналітика може представляти стан складу у реальному часі або інформаційну панель для моніторингу працездатності веб-сайту чи програми. У цьому випадку дані споживаються в режимі реального часу або безпосередньо з вихідної системи, або з конвеєра потокових даних. Практичне застосування операційної аналітики відрізняється від традиційної BI, оскільки операційна аналітика зосереджена на сьогоденні й не обов’язково стосується історичних тенденцій.

На практиці аналітика, яка надається клієнтам на платформі SaaS і є вбудованою, має окремий набір вимог і ускладнень. Внутрішня BI має обмежену аудиторію та зазвичай надає обмежену кількість уніфікованих представлень. Контроль доступу є критичним, але не особливо складним. Якщо ж поглянути на вбудовану аналітику, то там кількість запитів на звіти та відповідне

навантаження на аналітичні системи різко зростає. Контроль доступу значно складніший і критичніший. Компанії можуть обслуговувати окремі аналітичні дані багатьох тисяч клієнтів. Кожен клієнт повинен бачити свої і тільки свої дані. Витік даних між клієнтами буде вважатися масовим порушенням довіри, що призведе до уваги ЗМІ та значної втрати клієнтів. Саме через наведені вище ускладнення вбудована аналітика позначається окремо від традиційної BI аналітики.

Підсумовуючи все вищесказане, життєвий цикл обробки даних забезпечує структурований каркас для ефективного управління даними організації. Ключові етапи — генерація, завантаження, трансформація, доставка та аналітика — підкреслюють комплексний підхід до обробки даних від їх створення до кінцевого використання. Цей життєвий цикл слугує цінним посібником для навігації в складних процесах обробки даних, підтримки прийняття обґрунтованих рішень і стратегічних ідей.

1.4 Постановка задач розробки

Ціль аналітичного сервісу для аналізу ринку вживаних автомобілів в Україні - забезпечити користувачів всебічним розумінням автомобільного ринку в конкретний момент часу. Перш за все, основною метою є створення інтерактивної та легко доступного сервісу, що дозволяє аналізувати та порівнювати різні аспекти ринку, від цінової політики до технічних характеристик автомобілів, представлених на ринку. Однією з головних функціональностей є забезпечення користувачам можливості розглядати агреговані метрики в розрізі по цінах, пробігу та інших ключових параметрах автомобілів.

На шляху розробки такого сервісу потрібно буде вирішити наступні задачі:

Збирання даних. В рамках розробки аналітичного сервісу, важливо створити механізм збирання актуальних та достовірних даних з релевантних джерел.

Обробка даних. Обробка даних є ключовим етапом, що визначає якість та ефективність кінцевого продукту. В процесі обробки, дані потрібно очистити від зайвих деталей та структурувати.

Зберігання даних. Дані мають зберігатися у базі даних, яка буде формувати основу для подальшого аналізу та вивчення даних. Схема бази даних має відповідати цілям кінцевого використання – аналітичний тип запитів, а також має містити всі необхідні поля для зручної фільтрації та формування зрізів.

Візуалізація даних. Візуалізація графіків та діаграм не лише робить складні дані більш зрозумілими, але і надає користувачам можливість швидше розуміти ключові характеристики ринку. В ході роботи потрібно побудувати інтерактивні графіки, що дозволять користувачам досліджувати різні аспекти ринку в розрізі по цінах, пробігу та інших ключових параметрах автомобілів.

1.5 Висновки до розділу

Автомобільний сектор відіграє життєво важливу роль в українській економіці, сприяючи транспортним зв’язкам, підтримуючи торгівлю, туризм і створюючи можливості для працевлаштування. Український автомобільний ринок складається з двох основних сегментів: нових і вживаних автомобілів. Станом на 2022 рік ринок вживаних автомобілів значно перевершує ринок нових автомобілів з понад 1 мільйоном угод, що підкреслює високий попит на вживані автомобілі серед українських автомобілістів.

Було визначено, що кількість користувачів, які використовують спеціалізовані онлайн-платформи для пошуку та продажу (або купівлі) авто постійно зростає. Популярність таких онлайн-платформ збільшується завдяки широкому вибору авто, конкурентним цінам, зручності та можливості

переглядати оголошення та спілкуватися з продавцями в рамках однієї платформи. AutoRia є провідним гравцем на ринку України у сфері онлайн-продажу автомобілів та інших пов'язаних послуг, охоплюючи 82% загальних відвідувань веб-сайтів серед ключових конкурентів. Маючи приблизно півмільйона відвідувачів щодня та майже 300 000 оголошень, AutoRia забезпечує до 1500 продажів на день, зміцнюючи свою позицію як найпопулярніший автомобільний онлайн-майданчик в Україні.

Було розглянуто методологію побудови рішень для обробки та аналізу даних шляхом детального аналізу такої абстракції як життєвий цикл інженерії даних. За допомогою цієї абстракції, було послідовно наведено фази інженерії даних без заглиблення у технологічні деталі. У розділі підкреслюється, що цінність даних матеріалізується, коли вони застосовуються в практичному контексті, при цьому аналітика стає основною метою побудови систем обробки даних. Життєвий цикл обробки даних служить цінним посібником для навігації складними процесами обробки даних, підтримки прийняття обґрунтованих рішень і стратегічних ідей.

В результаті аналізу предметної області та методології побудови аналітичних сервісів, було поставлено ряд задач, що потрібно вирішити в ході виконання роботи. Такі задачі включають: збір даних із відповідних джерел; обробку даних для забезпечення якості та ефективності; зберігання даних у структурованій базі даних, адаптовані до типів аналітичних запитів; візуалізацію даних за допомогою інтерактивних графіків.

2 ТЕХНОЛОГІЇ РОЗРОБКИ АНАЛІТИЧНИХ СЕРВІСІВ

2.1 Дослідження технологій обробки даних

Аналітичні платформи є основою для організацій, які прагнуть підкріплювати процес прийняття рішень даними та знаходити практичне застосування даним. Основоположні елементи цих платформ охоплюють різноманітні технології, процеси та компоненти, які разом сприяють створенню та підтримці якісного аналітичного процесу. Ключові компоненти такої структури включають:

Джерела даних. У контексті інженерії даних джерело даних відноситься до джерела або місця, з якого дані збираються для обробки, аналізу та зберігання. Джерела даних можуть бути різними, включаючи різні типи систем, платформ і форматів. У таблиці 2.1 наведено описано декілька типів джерел даних, проте в загальному їх значно більше. Нижче буде коротко розглянуто згадані у таблиці типи джерел даних.

Реляційна база даних зазвичай представлена базою даних застосунку або сервісу. Стандартним прикладом є база даних, яка зберігає стан банківських рахунків. У міру здійснення транзакцій і платежів клієнта програма оновлює баланс банківського рахунку. Як правило, база даних програми — це система онлайн-обробки транзакцій (з англ. *OLTP – On-Line Transaction Processing*) — база даних, яка читає та записує окремі записи даних із високою швидкістю. По суті, бази даних OLTP добре працюють як серверні програми, коли тисячі або навіть мільйони користувачів можуть одночасно взаємодіяти з програмою, одночасно оновлюючи та записуючи дані. Системи OLTP менш підходять для використання у випадку коли один запит повинен сканувати величезну кількість даних.

На відміну від системи OLTP, система онлайн-аналітичної обробки (з англ. *OLAP – On-Line Analytical Processing*) створена для виконання великих

аналітичних запитів і зазвичай неефективна при обробці пошуку окремих записів. Наприклад, сучасні бази даних, що орієнтовані на зберігання даних у стовпцях, оптимізовані для сканування великих обсягів даних за рахунок відмови від індексів для покращення масштабованості та продуктивності сканування. Будь-який запит у такого роду базу даних зазвичай передбачає сканування мінімального блоку даних, часто розміром 100 МБ або більше. Спроба шукати тисячі окремих елементів за секунду в такій системі спровокує значні проблеми з продуктивністю.

Таблиця 2.1 – Джерела даних.

Джерело даних	Опис
Реляційна база даних	Традиційні бази даних зі структурованими даними, організованими в таблиці та попередньо визначені схеми.
Файли	Файли, що містять дані, часто у форматах CSV, TXT, JSON.
API	Інтерфейси, що дозволяють системам програмно спілкуватися та обмінюватися даними.
Черги повідомлень	Безперервні потоки даних у реальному часі з таких джерел, як Apache Kafka, RabbitMQ.

Файл — це послідовність байтів, яка зазвичай зберігається на диску. Програми часто записують дані у файли. Файли можуть зберігати локальні параметри, події, журнали, зображення та аудіо. Основні типи форматів вхідних файлів — це Excel, CSV, TXT, JSON і XML. Ці файли мають свої особливості та можуть бути структурованими (Excel, CSV), напівструктурованими (JSON, XML, CSV) або неструктурованими (TXT, CSV).

Інтерфейси прикладного програмування (з англ. *API - Application Programming Interface*) є стандартним способом обміну даними між системами.

API є стандартним і поширеним способом обміну даними в мережі, для платформ SaaS і між внутрішніми системами компаній. В Інтернеті існує багато типів API-інтерфейсів, але найбільш популярними є ті, що побудовані на основі HTTP: REST, GraphQL та інші.

Черга повідомлень — це механізм для асинхронного надсилання даних між окремими системами за допомогою моделі публікації та підписки. Як видно з рисунку 2.1, дані публікуються в черзі повідомлень і доставляються одному або кільком підписникам. Сервіс-підписник підтверджує отримання повідомлення, видаляючи його з черги.

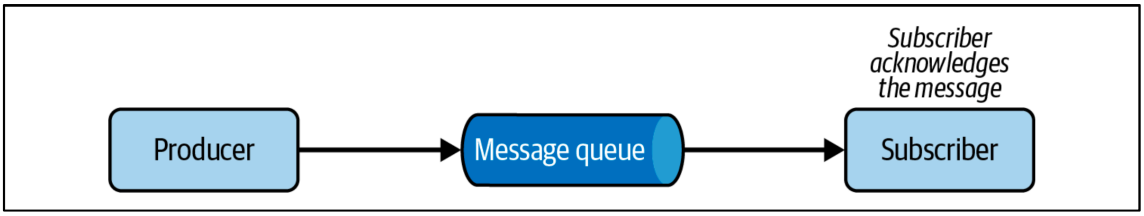


Рисунок 2.1 – Принцип роботу черги повідомлень. [7]

Черги повідомлень дозволяють програмам і системам бути відокремленими один від одного і широко використовуються в архітектурах мікросервісів. Черга повідомлень буферизує повідомлення для обробки тимчасових стрибків навантаження та робить повідомлення довговічними через розподілену архітектуру з реплікацією. Черги повідомлень є критично важливим компонентом для відокремлених мікросервісів і архітектур, керованих бізнес-подіями.

ETL та ELT конвеєри. На рисунку 2.2 зображено діаграму будови ETL конвеєру.

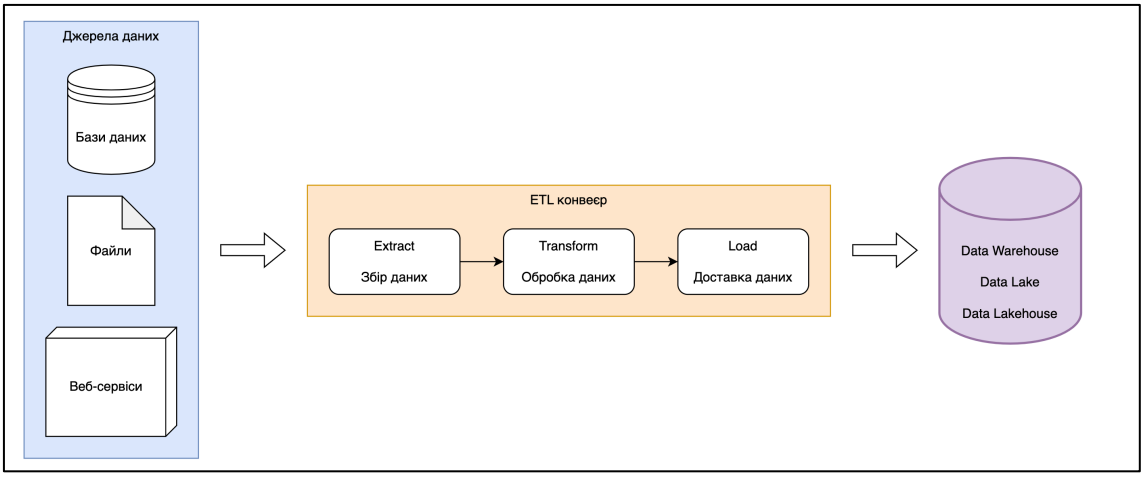


Рисунок 2.2 – Діаграма ETL конвеєру.

Конвеєр ETL (англ. *ETL - Extract, Transform, Load*) — це систематичний процес інтеграції даних, який передбачає вилучення необроблених даних із різних джерел, перетворення їх у потрібний формат і завантаження в цільове місце призначення для подальшого аналізу та прийняття рішень.

На етапі збору, необроблені дані збираються з різних систем, які можуть включати бази даних, файли, API або інші сховища даних. Збір даних зазвичай передбачає надсилання запитів до баз даних, читання файлів або отримання необхідних даних з API.

Після збору необроблені дані проходять процес обробки, щоб перетворити їх у структурований і придатний для використання формат. Обробка може включати очищення та перевірку даних, підстановку відсутніх значень, агрегування інформації та застосування бізнес-правил для задоволення вимог цільової системи.

Оброблені дані доставляються в цільову систему, якою може бути сховище даних, озеро даних або інше середовище зберігання. Доставка передбачає вставлення оброблених даних у таблиці, файли або системи зберігання у місці доступному для кінцевого користувача. Цей етап може включати стратегії інкрементальної доставки оновлень або повне оновлення наборів даних.

Традиційний ETL конвеєр покладається на спеціалізовану зовнішню систему, що виконує задачі збору, обробки та очищення даних під час підготовки їх до цільової схеми. Потім готові дані будуть доставлені до сховища даних, де можна буде виконувати задачі бізнес-аналітики. Сам шаблон ETL був обумовлений обмеженнями як вхідної, так і цільової систем. Фаза збору, як правило, була головним вузьким місцем, оскільки обмеження вхідної бази даних обмежували швидкість отримання даних. Крім того, перетворення виконуються на окремому кластері, оскільки сховище даних мало обмежені ресурси як для зберігання, так і для обробки.

Наступний етап розвитку – це ELT конвеєр. Оскільки продуктивність і ємність сховищ даних зросла, стало можливим просто витягувати необроблені дані з вхідної системи, імпортувати їх у сховище даних із мінімальним перетворенням, а потім очищати та трансформувати безпосередньо в системі сховища даних. Діаграму ELT конвеєра зображено на рисунку 2.3.

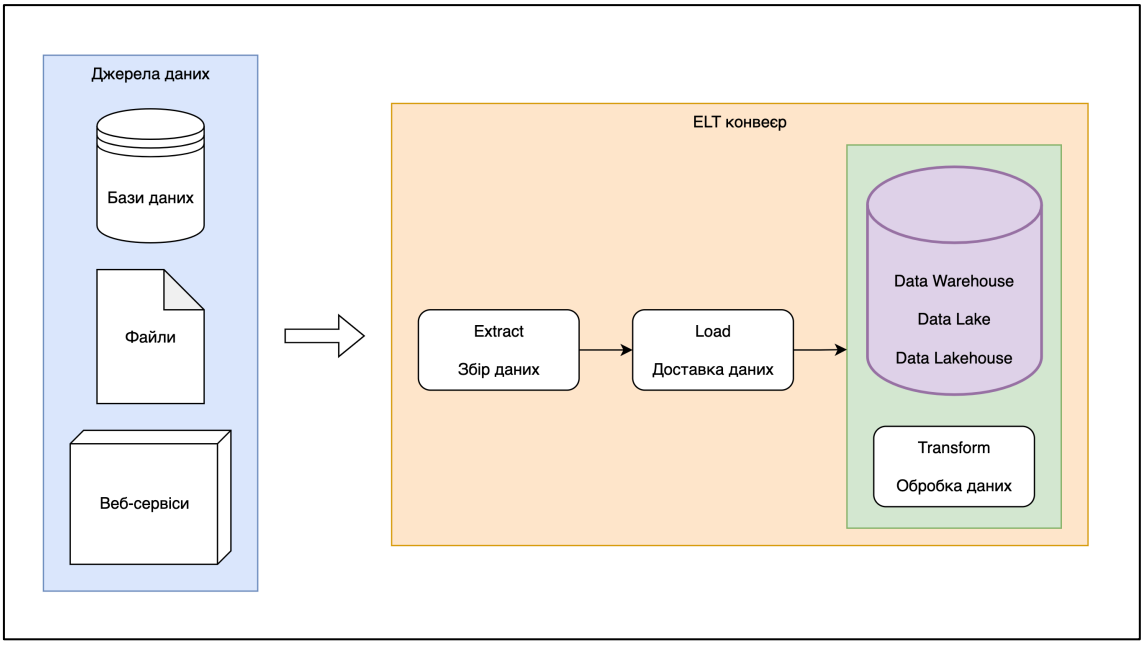


Рисунок 2.3 – Діаграма ELT конвеєру даних.

Конвеєр ELT — це альтернативний підхід до обробки даних, коли перетворення відбуваються в цільовому сховищі даних, а не в процесі попередньої обробки даних. Основними кроками конвеєра ELT є:

– Збір даних. Подібно до ETL, етап завантаження даних передбачає збір необроблених даних із різноманітних вхідних систем, які можуть включати бази даних, файли, API або інші джерела.

– Доставка даних. На етапі доставки даних, зібрані дані завантажуються безпосередньо в цільове сховище даних.

– Етап обробки даних в ELT конвеєрі відбувається в цільовому сховищі даних, де необроблені дані трансформуються в структурований і придатний для використання формат. Даний етап може включати такі завдання: Завдання: застосування логіки трансформації та бізнес-правил; використання SQL або інших мов для виконання перетворень даних; збагачення, агрегація або форматування даних у цільовому сховищі.

Сховища даних. У світі дана інженерії існує 3 основні архітектурні підходи до побудови цільового сховища даних:

– Data Warehouse (з англ. «Сховище даних»);

– Data Lake (з англ. «Озеро даних»);

– Data Lakehouse (з англ. «Озероподібне сховище даних»).

Data Warehouse (також відомий як DW, DWH) — це сховище, яке систематично й цілеспрямовано накопичує, організовує та зберігає великі обсяги структурованих даних, отриманих із різних джерел. Функціонуючи як комплексне та інтегроване рішення для зберігання даних, сховища даних насамперед спрямовані на покращення ефективності запитів, аналізу та звітності для процесів прийняття рішень. Архітектура DW зазвичай побудована на базі реляційної моделі, використовуючи такі методи, як індексування та розділення для оптимізації продуктивності запитів. Процеси ETL є невід’ємними компонентами робочих процесів сховища даних, гарантуючи, що дані очищаються, трансформуються та стандартизуються перед завантаженням у

сховище. Крім того, у сховищах даних часто використовуються методи багатовимірного моделювання, наприклад схеми зірки (англ. *Star schema*) або сніжинки (англ. *Snowflake schema*), для підвищення ефективності пошуку даних у аналітичних сценаріях використання.

Схема Зірка - це фундаментальна концепція в інженерії даних, що представляє собою техніку багатовимірного моделювання, яка широко використовується при розробці реляційних баз даних для систем підтримки прийняття рішень. Для цієї концепції характерна центральна таблиця фактів (англ. *Fact table*), оточена таблицями вимірів (англ. *Dimension tables*), утворюючи структуру, що нагадує зірку. Приклад такої архітектури в деталях зображено на рисунку 2.4.

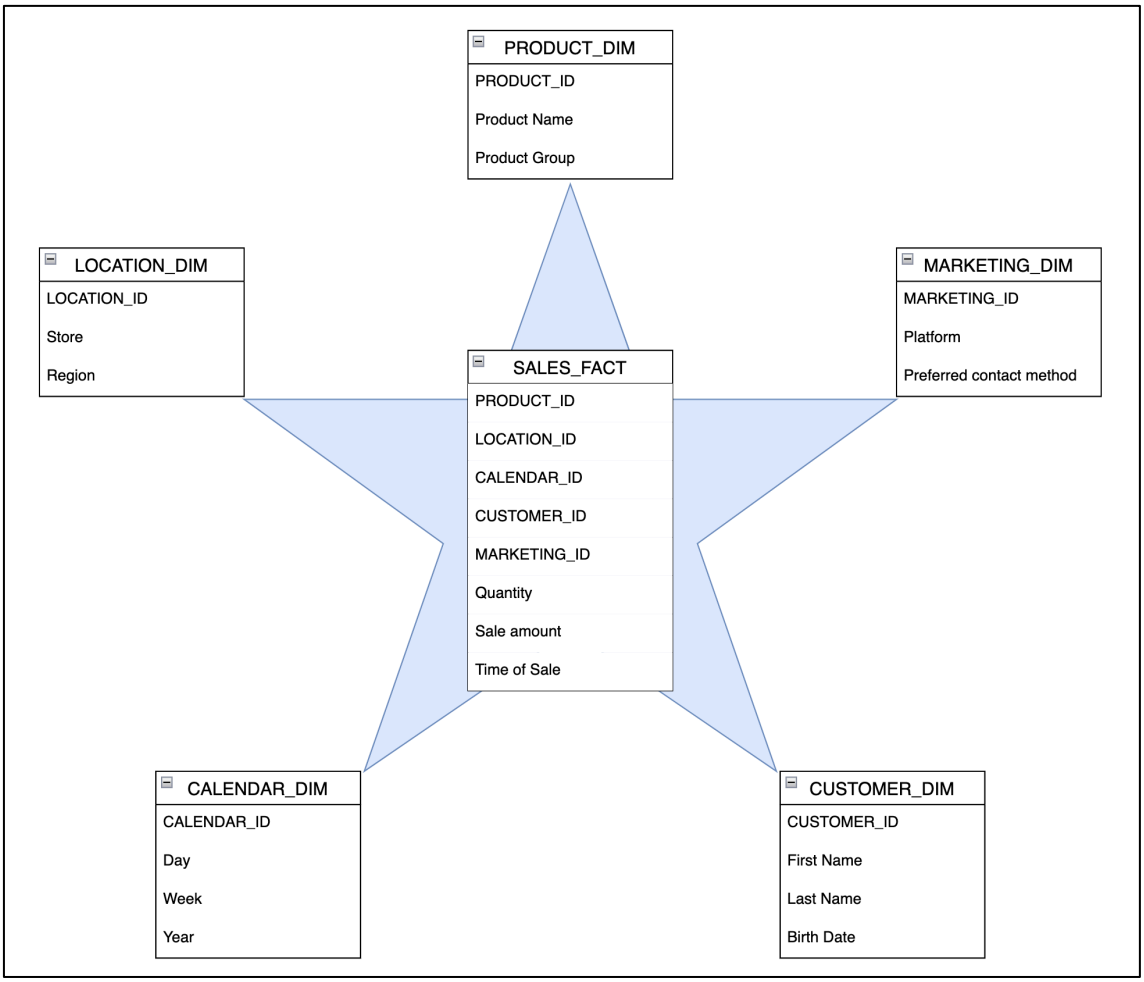


Рисунок 2.4 – Приклад архітектури сховища даних за схемою Зірка.

Таблиця фактів містить кількісні числові дані, які служать центром для аналізу, тоді як таблиці розмірів надають описові атрибути, які полегшують контекстуалізацію та категоризацію. Ця схема підвищує ефективність запитів і звітів, оскільки мінімізує кількість з'єднань, необхідних для складних аналітичних запитів, підвищуючи загальну продуктивність системи. Схема Зірка є прикладом балансу між простотою та аналітичною потужністю, сприяючи її поширеності у архітектурі сховищ даних і рішень бізнес-аналітики.

Схема Сніжинка - це теж відома парадигма в інженерії даних і багатовимірному моделюванні, відрізняється від схеми Зірка шляхом подальшої нормалізації таблиць вимірів, створюючи таким чином ієрархічну та взаємопов'язану структуру. У цій схемі таблиці вимірів розкладаються на підвиміри, що призводить до розгалуження, схожого на сніжинку. Нормалізація даних у схемі Сніжинка зменшує повторюваність інформації, підвищує цілісність даних і сприяє більш нормалізованій формі зберігання. Однак така підвищена нормалізація може призвести до більш складних структур запитів і потенційних наслідків для продуктивності порівняно зі схемою Зірка. Схема Сніжинка підходить для сценаріїв, де цілісність даних і нормалізація мають пріоритет над спрощеними структурами запитів, і її розгортання є звичайним у середовищах, де пріоритетом є зменшення повторюваності в атрибутах вимірів. Приклад схеми Сніжинка зображено на рисунку 2.5.

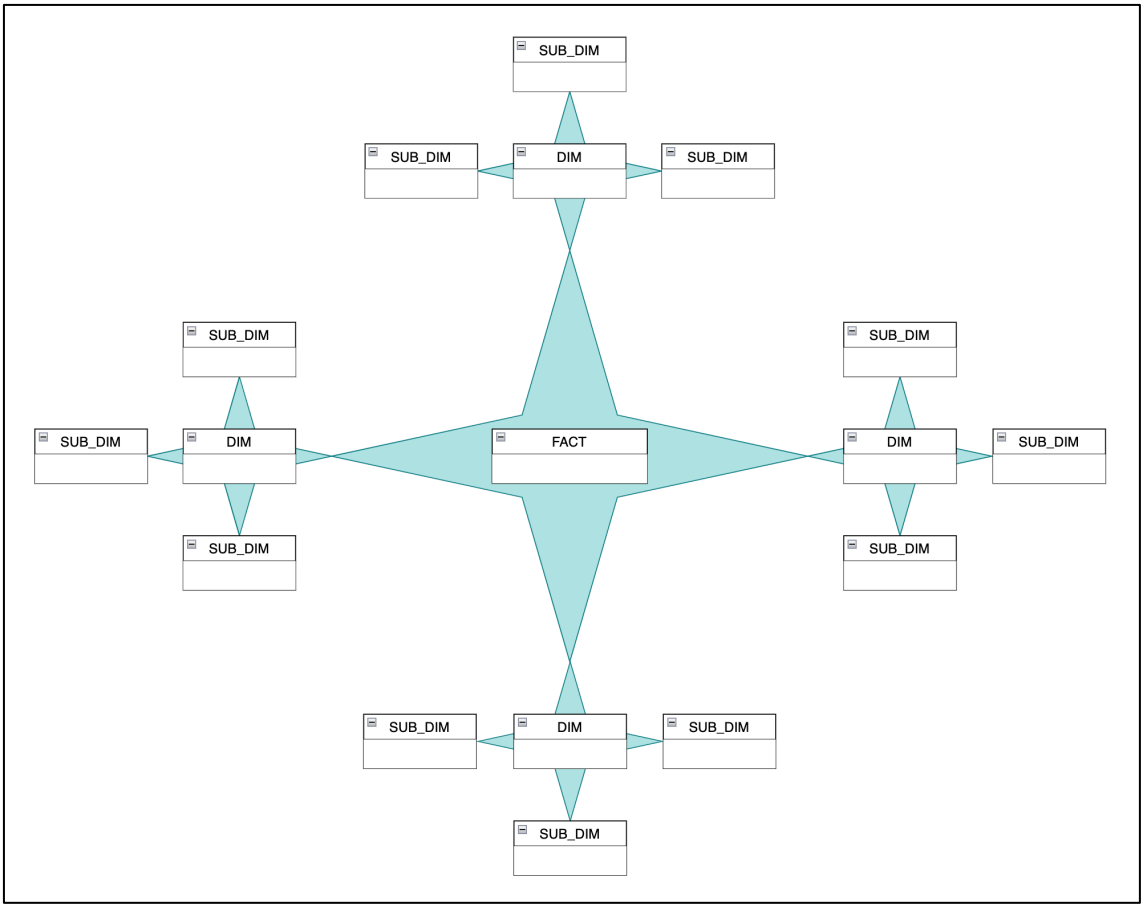


Рисунок 2.5 – Приклад архітектури сховища даних за схемою Сніжинка.

Сховища даних відіграють ключову роль у підтримці потреб організацій у бізнес-аналітиці, надаючи структуровану та консолідовану платформу для зберігання історичних і поточних даних. Використання інструментів OLAP додатково розширює можливості виконання комплексного аналізу, ідентифікації тенденцій і підтримки прийняття рішень. Сховища даних сприяють покращенню доступності, узгодженості та надійності даних, тим самим сприяючи більш обґрунтованому підходу до прийняття рішень на підприємства. Крім того, сховища даних дотримуються принципів управління даними, забезпечуючи якість, цілісність та безпечність даних протягом всього їх життєвого циклу.

Data Lake - конструкція сучасної інженерії даних, є розширеним і гнучким сховищем даних, призначеним для розміщення великих і різноманітних наборів даних у їх необроблених, неструктурованих або напівструктурованих формах.

На відміну від традиційних парадигм зберігання даних, Data Lake відрізняє гнучкість схеми під час читання, дозволяючи зберігати різнорідні дані без необхідності попередньо визначених структур. Маючи масштабованість як визначальний атрибут, Data Lake можуть ефективно обробляти величезні обсяги даних шляхом горизонтального масштабування ресурсів. Data Lake часто використовують розподілені структури зберігання та обробки, сприяючи їх адаптивності при обробці великих даних. Крім того, Data Lake є екосистемами, які сприяють спільному дослідженню та аналізу даних, дозволяючи користувачам отримувати інформацію з цілого спектру джерел даних. Використання механізмів керування метаданими в Data Lake покращує можливість виявлення даних, відстеження походження та управління. На рисунку 2.6 зображено приклад архітектури конвеєру даних з використанням Data Lake як цільового сховища.

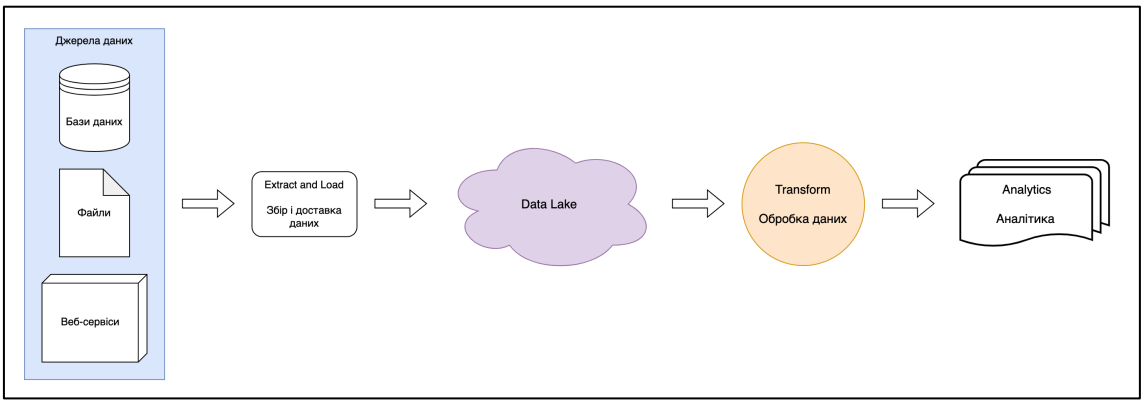


Рисунок 2.6 – Приклад архітектури з використанням Data Lake.

Архітектура Data Lake часто інтегрується з різноманітними інструментами обробки даних, полегшуючи перетворення даних «на льоту» та аналітику. Data Lake підтримує як пакетну обробку даних, так і обробку даних у реальному часі, що дозволяє організаціям отримувати практичну користь із динамічних наборів даних. Хоча відсутність примусових схем дозволяє включати необроблені дані, це також вимагає надійного керування метаданими для полегшення ефективного

управління даними. Багатогранна природа Data Lake робить його основоположним елементом у сучасних архітектурах, що дозволяє організаціям використовувати потенціал своїх ресурсів для розширеної аналітики та машинного навчання. Тим не менш, такі проблеми, як забезпечення якості даних, безпека та складність керування метаданими, є властивими для Data Lake і вимагаю системних рішень.

Data Lakehouse - нова парадигма в сучасній інженерії даних, являє собою синтез моделей озера даних і традиційного сховища даних, що прагне поєднати їхні сильні сторони. Ця гібридна архітектура поєднує гнучкість і масштабованість озера даних із можливостями структурованої обробки сховища даних. У Data Lakehouse необроблені дані можуть співіснувати зі структурованими підібраними даними, пропонуючи уніфіковану платформу для варіативного зберігання, перетворення та аналітики даних. Ця модель вирішує проблеми накопичення даних і підтримує як схему під час читання для дослідницького аналізу, так і схему під час запису для структурованої ефективної аналітики.

Відмінною особливістю Data Lakehouse є його акцент на підтримці пакетної обробки даних і обробки даних у реальному часі, завдяки чому враховується динамічний характер сучасних джерел даних. Завдяки використанню хмарних технологій і розподілених обчислювальних ресурсів, Data Lakehouse дають можливість організаціям виконувати масштабну аналітику та бездоганно інтегрувати інформацію в бізнес-операції. Як і з озером даних, управління метаданими відіграє вирішальну роль у такому сховищі даних.

Незважаючи на покращення гнучкості та аналітичних можливостей, модель Data Lakehouse створює складності в оркестрації, забезпеченні безпеки та оптимізації ресурсів. Організації, які використовують архітектуру о Data Lakehouse, повинні долати труднощі, пов’язані з балансуванням співіснування необроблених і оброблених даних; забезпеченням ефективності аналітичних запитів і підтримкою якості даних; забезпеченням стандартів безпеки. Будучи

парадигмою, що розвивається, Data Lakehouse має значні перспективи у застосування на великих підприємствах, що прагнуть отримати максимум практичного застосування своїм даним.

Аналітичні сценарії використання. Дані з цільового сховища даних використовуються в різних підрозділах організації для прийняття обґрунтованих рішень, аналізу та звітності. Декілька ключових областей, де зазвичай використовуються дані зі сховища наведено нижче:

– Бізнес-аналітика. Сховища даних займають центральне місце в ініціативах бізнес-аналітики, забезпечуючи структуроване й оптимізоване середовище для аналізу історичних і поточних даних. Інструменти бізнес-аналітики підключаються до сховища даних для створення комплексних звітів, інформаційних панелей і візуалізацій.

– Аналітика та звітність. Спеціалісти з аналітики використовують сховища даних для виконання складних запитів і аналізу їх результатів. Структурований характер даних сприяє ефективній побудові звітності, дозволяючи організаціям отримати уявлення про тенденції, закономірності та показники ефективності.

– Системи підтримки прийняття рішень. Сховища даних служать основоположним елементом для систем підтримки прийняття рішень, забезпечуючи необхідну інфраструктуру даних для керівників і менеджерів найвищої ланки для прийняття обґрунтованих стратегічних рішень.

– Дослідження даних. Аналітики та дослідники даних досліджують дані, що зберігаються в сховищі, щоб виявити закономірності, кореляції та аномалії. Можливість надсилати запити й аналізувати дані в структурованому форматі підвищує ефективність розвідувального аналізу даних.

– Оперативна звітність. Операційні команди отримують доступ до сховища даних для рутинної та ситуативної звітності, моніторингу ключових показників ефективності та оцінки робочого стану різних бізнес-процесів.

– Прогнозування та предикативна аналітика. Сховища даних

відіграють вирішальну роль у підтримці прогнозування та прогнозної аналітики. Надаючи історичні та поточні дані, вони дозволяють організаціям будувати моделі та прогнозувати майбутні тенденції.

– Управління взаємовідносинами з клієнтами (англ. *CRM - Customer Relationship Management*). Відділи маркетингу та продажів використовують сховища даних для аналізу поведінки, уподобань і тенденцій клієнтів. Ця інформація використовується для посилення взаємодії з клієнтами, оптимізації маркетингових стратегій і покращення результатів продажів.

– Фінансовий аналіз. Фінансові відділи використовують сховища даних для фінансової звітності, бюджетування та прогнозування. Структуровані фінансові дані, що зберігаються в сховищі, допомагають аналізувати доходи, витрати та загальну фінансову ефективність організації.

– Підготовка навчальних даних. Моделі машинного навчання вимагають високоякісних навчальних даних. Сховища даних служать надійним джерелом для підготовки наборів даних, які використовуються для навчання алгоритмів машинного навчання. Можливість витягувати відповідні дані зі сховища даних полегшує створення різноманітних і репрезентативних навчальних наборів.

2.2 Аналіз та вибір основних засобів розробки

2.2.1 Вибір засобів розробки ETL конвеєру

Якщо розглянути структуру ETL конвеєру даних, можна побачити, що він представлений послідовністю пов’язаних між собою програм. У найпростішому випадку такий конвеєр може складатися з 3 програм:

– Програма для збору та завантаження даних;

– Програма для перетворення даних;

– Програма для доставки даних до сховища даних.

Беручи до уваги наведену вище інформацію, наступним кроком буде визначення засобів розробки ETL конвеєру, тобто тих програм, з яких він буде складатися.

Коли мова заходить про створення програми, то перший інструмент, який спадає на думку – мова програмування. У сфері обробки даних використовується багато мов, проте можна виділити декілька основних: Python, Scala та Java.

Python, динамічно типізована мова програмування високого рівня, стала популярним інструментом у сфері розробки даних завдяки своїй універсальності, зручності читання та розгалуженій екосистемі. Python, відомий своєю простотою, створює середовище, сприятливе для швидкого розвитку та легкого обслуговування робочих процесів обробки даних. Його об’єктно-орієнтована парадигма в поєднанні з можливостями функціонального програмування забезпечує гнучку основу для вирішення різноманітних завдань обробки даних.

Надійна екосистема Python включає бібліотеки та фреймворки, призначені для завдань інженерії даних. Pandas, бібліотека обробки даних, сприяє ефективній обробці та трансформації даних. Apache Spark, інтегрований з PySpark, забезпечує масштабовану та розподілену обробку даних, тоді як Dask забезпечує паралельне обчислення для обробки великих наборів даних. Крім того, інтеграція Python із базами даних SQL і системами NoSQL покращує його застосування в різних сценаріях зберігання та збору даних.

Популярність Python в інженерії даних посилюється його широким впровадженням у сферах машинного навчання та науки про дані. Python служить об’єднуючим інструментом для всього конвеєра даних, від збору даних і попередньої обробки до навчання моделі та її розгортання.

Розробка, керована спільнотою, відіграє ключову роль в успіху Python у розробці даних. Широка підтримка спільноти забезпечує постійний приплив бібліотек, інструментів і документації, сприяючи спільному вирішенню проблем і обміну знаннями.

Незважаючи на свої сильні сторони, Python стикається з проблемами, що пов’язані з продуктивностю при виконані інтенсивних обчислювальних завдань. Це може пом’якшуватися за рахунок інтеграції з оптимізованими бібліотеками або переходом критичних для продуктивності компонентів на мови нижчого рівня.

Підводячи підсумок, перевага Python у розробці даних пояснюється його зручним синтаксисом, універсальною екосистемою та співтовариством, що позиціонує його як зручну мову для навігації у складних процесах обробки та аналізу сучасних даних.

Scala - статично типізована функціональна мова програмування, що стала вибором професіоналів у галузі обробки даних, поєднуючи виразність функціонального програмування з надійністю статично типізованої мови. Лаконічність Scala та можливість взаємодії з JVM (англ. *Java Virtual Machine*) роблять її гарним вибором для розробки масштабованих та ефективних конвеєрів обробки даних. Цю мову цінують за її здатність легко інтегруватися з існуючими бібліотеками та фреймворками Java, сприяючи плавному переходу для організацій із усталеними екосистемами Java.

Парадигми функціонального програмування в Scala, такі як незмінність і функції вищого порядку, сприяють зрозумілості коду та зручності обслуговування, що є важливими атрибутами для складних завдань інженерії даних. Підтримка цією мовою принципів як об’єктно-орієнтованого, так і функціонального програмування підвищує гнучкість у розробці робочих процесів даних.

Роль Scala в обробці великих даних заслуговує особливої уваги, так як найбільш використовуваний фреймворк у інженерії даних – Apache Spark – написаний мовою Scala.. Лаконічний синтаксис і виразні конструкції Scala сприяють розробці читабельних програм Spark, забезпечуючи ефективну обробку даних у масштабі.

Паралелізм, що властивий Scala, добре відповідає вимогам інженерії даних, сприяючи розробці високопродуктивних і масштабованих конвеєрів обробки даних. Його модель для паралельного програмування та підтримка паралельних колекцій сприяють ефективному використанню багатоядерних процесорів.

Система типів мови підвищує надійність і правильність робочих процесів розробки даних, дозволяючи розробникам виявляти помилки під час компіляції, а не під час виконання. Крім того, сильний акцент Scala на незмінності узгоджується з найкращими практиками щодо забезпечення узгодженості та надійності даних у сценаріях розподілених обчислень.

Незважаючи на свої сильні сторони, Scala має складнішу криву навчання порівняно з більш поширеними мовами.

Активна спільнота та постійний розвиток Scala, прикладом якого є Scala Center і Lightbend, ще більше зміцнюють її позицію як ключової мови у сфері інженерії даних. Таким чином, поєднання функціональних і об’єктно-орієнтованих парадигм Scala, сумісність із JVM і придатність для обробки великих даних роблять її потужною мовою для створення надійних і масштабованих рішень обробки даних.

Java, широко поширена та універсальна мова програмування, зарекомендувала себе як непохитний вибір у сфері розробки даних. Надійна та статично типізована природа Java сприяє надійності та зручності підтримки кодових баз інженерії даних, узгоджуючи вимоги складних і критично важливих робочих процесів.

Розгалужена екосистема мови є наріжним каменем її успіху в розробці даних із такими фреймворками, як Apache Hadoop і Apache Flink, які використовують можливості Java для масштабованої та розподіленої обробки даних. Крім того, безперебійна інтеграція Java з безліччю баз даних і технологій зберігання даних зміцнює її позицію як універсального інструменту для завдань інженерії даних.

Паралелізм, важливий для ефективної обробки великих наборів даних і паралельної обробки, є сильною стороною Java. Її багатопотокові можливості дозволяють розробникам створювати паралельні програми, що є життєво важливим аспектом у контексті робочих процесів обробки даних.

Хоча синтаксис Java може бути більш багатослівним порівняно з іншими мовами, її акцент на явному та читабельному коді узгоджується з принципами зручності обслуговування коду з часом. Об'єктно-орієнтована парадигма в Java сприяє організації коду та багаторазовому використанню, важливим атрибутам для розробки модульних і розширюваних систем обробки даних.

Продуктивність Java, доповнена компіляцією Just-In-Time (JIT), робить її вигідною для інтенсивних обчислювальних завдань у інженерії даних. Про зріле та усталене становище мови в галузі свідчить її безперервна еволюція під керівництвом Java Community Process (JCP) і широкої спільноти Java.

Підсумовуючи, характеристики Java, такі як незалежність від платформи, надійність, розгалужена екосистема, підтримка паралелізму та продуктивність, роблять її чудовою мовою для побудови масштабованих і надійних рішень обробки даних. Її незмінна популярність і адаптивність підкреслюють її важливість у вирішенні нових викликів сучасної обробки даних і аналітики.

Порівняльний аналіз згаданих вище мов програмування проведено у таблиці 2.2, що знаходиться нижче.

Таблиця 2.2 – Порівняльний аналіз мов Python, Java та Scala.

Критерій	Python	Java	Scala
Зручність синтаксису	Синтаксис Python дуже читабельний і стислий, що підвищує швидкість розробки.	Синтаксис Java багатослівний, проте дуже структурований і зручний для читання та подальшої підтримки.	Синтаксис Scala є лаконічним. Він балансує між виразністю і зручністю читання.

Продовження таблиці 2.2

Критерій	Python	Java	Scala
Насиченість екосистеми	Python має велику екосистемою з потужними бібліотеками та фреймворками для різноманітних завдань інженерії даних.	Велика екосистема Java включає потужні фреймворки (наприклад, Apache Hadoop, Apache Flink) і повну інтеграцію з базами даних.	Scala використовує екосистему Java, пропонуючи взаємодію та чудово підходить для обробки великих даних за допомогою таких фреймворків, як Apache Spark.
Складність вивчення	Простота Python робить його зручним інструментом для початківців.	Java з більш структурованим підходом може бути складнішою у навчанні, особливо для початківців.	Крива навчання Scala може бути складною, особливо для тих, хто новачок у концепціях функціонального програмування.
Продуктивність	Інтерпретована природа Python може призвести до падіння продуктивності у завданнях, що потребують обчислень.	Java зі своєю компіляцією Just-In-Time (JIT) забезпечує стабільну продуктивність для широкого спектру завдань інженерії даних.	Scala, компілюючи байт-код Java, успадковує переваги продуктивності Java і добре підходить для паралельної обробки.

За сукупністю таких характеристик як зручність синтаксису, насиченість екосистеми та складність вивчення, на погляд автора, перевагу має мова Python. Ця мова ідеально підходить для реалізації проектів малої та середньої складності. Простота мови та насиченість її екосистеми допоможуть швидко перейти від вивчення мови та інструментів до безпосередньої реалізації майбутньої бізнес-логіки обробки даних.

Переважна більшість проектів, що працюють з великим масивом даних використовують для їх обробки фреймворк Apache Spark. Тому необхідно також розглянути особливості цього інструменту.

Apache Spark – фреймворк для розподіленого обчислення з відкритим кодом. Він став ключовим інструментом у сфері обробки та аналітики великих даних. Spark полегшує обробку великих масивів даних у відмовостійкий та розподілений спосіб. Його основна абстракція, Resilient Distributed Datasets (RDD), забезпечує відновлення після помилок і ефективну паралельну обробку, створюючи основу для надійних і масштабованих перетворень даних.

Варто згадати PySpark – API для Apache Spark, що забезпечує повну інтеграцію можливостей Spark із мовою програмування Python. PySpark надає високорівневий інтерфейс для обробки та аналізу даних, що дає змогу розробникам Python використовувати можливості Spark для розподіленної обробки даних.

2.2.2 Вибір імплементації реляційної бази даних

Традиційний сховище даних, виду Data warehouse, будується на основі реляційної бази даних. Найважливішим критерієм до бази даних, що буде використовуватися як сховище даних – це підтримка ефективної обробки OLAP (англ. *On-Line Analytical Processing*) запитів. Популярними рішеннями, що підтримують такий сценарій використання на сьогоднішній день є:

- PostgreSQL;
- Microsoft SQL Server;
- Oracle Database.

PostgreSQL, система керування реляційними базами даних з відкритим кодом, набула популярності як життєздатне рішення для сховищ даних завдяки своїй надійній архітектурі, розширюваності та повному набору функцій. Відомий своєю сумісністю з ACID і підтримкою складних запитів, PostgreSQL

служить надійною основою для створення сховищ даних, які вимагають цілісності та узгодженості даних. Прикладом розширюваності PostgreSQL є його підтримка користувальницьких функцій, типів даних і процедурних мов, що забезпечує гнучкість адаптації системи до конкретних вимог сховища даних.

Здатність PostgreSQL обробляти великі обсяги даних і підтримка паралельної обробки сприяють його ефективності у використанні в якості сховища даних. Крім того, підтримка PostgreSQL як OLTP, так і OLAP запитів, робить його універсальним вибором для користувачів, що мають різноманітні потреби в обробці даних.

Так як проект PostgreSQL є відкритим, це сприяє створенню середовища для співпраці, забезпечуючи безперервне вдосконалення та адаптацію до нових проблем. Дотримання стандартів SQL і сумісність з різними мовами програмування ще більше зміцнюють позицію PostgreSQL як вибору для побудови та підтримки сховища даних, для якого важливі надійність, розширюваність та продуктивність.

Microsoft SQL Server – комплексна система керування реляційними базами даних, виділяється як надійне рішення для сховища даних завдяки своїй складній архітектурі та багатим можливостям. SQL Server, відомий своєю бездоганною інтеграцією з екосистемою Microsoft, пропонує великий набір інструментів для проектування сховищ даних і керування ними. Його компонент Analysis Services підтримує OLAP запити, забезпечуючи ефективне виконання запитів і аналіз великих наборів даних у сховищі даних. Можливості паралельної обробки SQL Server ще більше підвищують ефективність роботи зі складним аналітичним навантаженням. Проект Microsoft SQL Server акцентує свою увагу на масштабованості, безпеці та інтегрованості.

Oracle Database – система керування реляційними базами даних, що може виступати у якості сховища даних. Відома своєю здатністю обробляти величезні набори даних, Oracle Database використовує розширені функції, такі як розподілення, матеріалізовані види та паралельна обробка, оптимізуючи

виконання складних аналітичних запитів у середовищі сховища даних. Підтримка OLAP ще більше сприяє ефективному багатовимірному аналізу даних. Oracle Database включає такі функції, як стиснення та індексування, для підвищення ефективності зберігання та швидкості пошуку, сприяючи оптимізованій роботі сховища даних. Oracle Database постійно оновлюється та підтримується однойменною корпорацією Oracle.

У таблиці 2.3 виконано порівняння розглянутих баз даних.

Таблиця 2.3 – Порівняльний аналіз баз даних PostgreSQL, Oracle Database та Microsoft SQL Server.

Критерій	PostgreSQL	Oracle Database	Microsoft SQL Server
Ліцензування	Відкрита, безкоштовна ліцензія	Комерційна ліцензія з різними опціями.	Комерційна ліцензія з різними опціями.
Підтримка OLAP	Потужна підтримка OLAP із розширеннями та плагінами.	Надійна підтримка OLAP через Oracle OLAP.	Комплексна підтримка OLAP із інтеграцією Analysis Services та Power BI.
Підтримка паралельних обчислень	Підтримує паралельну обробку для певних операцій, підвищуючи продуктивність аналітичних навантажень.	Пропонує складний механізм паралельного виконання запитів, який оптимізує обробку великих наборів даних на кількох процесорах.	Підтримує паралельну обробку, використовуючи паралелізм для виконання запитів і завантаження даних

Проаналізувавши наявні рішення, оптимальним виглядає база даних PostgreSQL. Вона має безкоштовну ліцензію, підтримує OLAP, а також може використовувати паралельні обчислення при аналітичних запитах. Її конкуренти мають більш потужні інструменти підтримки OLAP запитів та паралельних

обчислень, проте мають комерційну ліцензію та опції, які необхідно оплачувати протягом всього періоду використання.

2.2.3 Вибір інструментів інтерактивної візуалізації даних

Інтерактивні дашборди – фундаментальний компонент візуалізації даних у бізнес-аналітиці, вони є динамічними та орієнтованими на користувача інтерфейсами для дослідження та розуміння складних наборів даних. Ці дашборди об’єднують різноманітні візуалізації даних, такі як діаграми, графіки та таблиці, надаючи зацікавленим сторонам інформацію про ключові показники ефективності та тенденції. Інтерактивний характер дашбордів дозволяє користувачам безпосередньо взаємодіяти з даними, полегшуючи пошуковий аналіз даних і прийняття рішень. Використовуючи такі функції, як фільтри, деталізація та параметризація, інтерактивні дашборди дають змогу користувачам адаптувати свої дослідження. Їх роль виходить за рамки простого представлення даних, охоплюючи такі функції, як фільтрація даних, сортування та формування зрізів. Загалом інтерактивні дашборди слугують інструментами для перетворення даних у практичну інформацію, що сприяє прийняттю обґрунтованих рішень.

З популярних на сьогоднішній день сервісів, що пропонують такі інструменти як інтерактивні дашборди, можна виділити наступні:

– Tableau. Tableau — це відома платформа візуалізації даних і бізнес-аналітики, яка дає змогу користувачам створювати інтерактивні та проникливі дашборди. Завдяки зручному для користувача інтерфейсу з функцією перетягування та скидання Tableau підтримує широкий спектр з’єднувачів даних, що забезпечує бездоганну інтеграцію з різноманітними джерелами даних. Її надійні функції підходять як новачкам, так і досвідченим користувачам, полегшуючи динамічну візуалізацію для ефективного дослідження та аналізу даних.

– Power BI. Power BI, продукт у складі Microsoft Power Platform, є комплексним інструментом бізнес-аналітики, який полегшує створення інтерактивних дашбордів і звітів. Пропонуючи зручний інтерфейс і багату екосистему з'єднувачів, Power BI дозволяє користувачам підключатися до різних джерел даних, аналізувати дані та ділитися інформацією між організаціями. Його повна інтеграція з продуктами і хмарними службами Microsoft сприяє створенню згуртованого середовища аналітики даних.

– Google Looker Studio. Google Looker Studio – це платформа для дослідження даних і бізнес-аналітики, яка наголошує на простоті та орієнтованому на користувача дизайні. Будучи частиною Google Cloud, Looker Studio пропонує широкий спектр з'єднувачів для бездоганної інтеграції з різними базами даних і хмарними сервісами. Зосереджуючись на співпраці та доступності, Looker Studio дозволяє користувачам створювати інтерактивні інформаційні панелі та звіти та ділитися ними в екосистемі Google Cloud.

Таблиця 2.4 – Порівняльний аналіз Tableau, Power BI та Looker Studio.

Критерій	Tableau	Power BI	Google Looker Studio
З'єднувачі даних	Велика бібліотека з'єднувачів, що підтримує різні джерела даних, бази даних і формати файлів.	Багата екосистема з'єднувачів для різноманітних джерел даних, у тому числі хмарних служб і локальних баз даних.	Широкий вибір з'єднувачів для бездоганної інтеграції з різними базами даних, хмарними службами та сторонніми програмами.

Продовження таблиці 2.4

Критерій	Tableau	Power BI	Google Looker Studio
Легкість використання	Інтуїтивно зрозумілий інтерфейс з функцією перетягування та зручним дизайном, що робить його доступним як для початківців, так і для досвідчених користувачів.	Power BI має зручний інтерфейс з інтуїтивно зрозумілою навігацією, що полегшує швидке створення звітів і дашбордів.	Має дизайн орієнтований на користувача, з акцентом на простоті, що робить його доступним для користувачів із різними технічними знаннями.
Інтеграція з іншими засобами	Tableau добре інтегрується з різними сторонніми інструментами та платформами, пропонуючи надійні варіанти підключення для покращеної співпраці.	Power BI легко інтегрується з продуктами Microsoft, службами Azure та іншими програмами сторонніх розробників, забезпечуючи злагоджені робочі процеси.	Бездоганно інтегрується з Google Workspace та іншими службами Google Cloud, створюючи єдину екосистему.

Виходячи з того, що в процесі роботи не планується інтеграція з сервісами Microsoft та Google, а по іншим критеріям Tableau нічим не поступається, буде використано саме Tablea як сервіс для побудови інтерактивних дашбордів.

2.3 Висновки до розділу

У цьому розділі основна увага була зосереджена на підходах та технологіях, що застосовуються при побудові сучасних аналітичних платформ.

Ключові структурні компоненти таких платформ включають джерела даних, ETL конвеєри, сховища даних та певні аналітичні застосунки.

Було проаналізовано різні типи джерел даних, такі як реляційні бази даних, файли, API та черги повідомлень. Було виділено відмінності між системами OLTP, призначеними для високошвидкісного читання та запису окремих записів даних, і системами OLAP, оптимізованими для великих аналітичних запитів.

Визначено особливості та відмінності будови ETL та ELT конвеєрів даних. Традиційний конвеєр ETL покладається на спеціалізовану зовнішню систему, яка виконує завдання збору, обробки та очищення даних для підготовки та доставки до сховища даних, в той час як конвеєр ELT завантажує досить сирі дані у сховище, а вже в ньому здійснює основні перетворення.

Було досліджено різні підходи до побудови цільового сховища даних. За місцем і роллю сховища у конвеєрі даних, було виділено такі три концепції як Data Warehouse, Data Lake та Data Lakehouse. Для Data Warehouse було висвітлено два популярних шаблони проектування внутрішньої структури сзовища – Star та Snowflake.

Було зроблено декілька суттєвих висновків щодо вибору мови програмування для розробки ETL конвеєру. Визначено Python як ідеальний вибір завдяки його простоті, зручності та багатій екосистемі. Також відзначено його сумісність з різноманітними джерелами даних та базами даних, що робить його універсальним вибором для ETL конвеєру.

Проведено аналіз реляційних баз даних, зокрема Microsoft SQL Server, PostgreSQL та Oracle Database. За результатом порівняння було вибрано PostgreSQL, так як вона має безкоштовну ліцензію без обмеження будь-якого функціоналу.

У ході аналізу інструментів для побудови інтерактивних дашбордів були розглянуті три основних варіанти: Tableau, Google Looker Studio та Power BI. Всі три варіанти мають схожі якості, проте було вибрано Tableau як варіант зі

зручним інтерфейсом та без цільової інтеграції з певними сервісами, як у випадку з продуктами Microsoft та Google.

3 РОЗРОБКА АНАЛІТИЧНОГО СЕРВІСУ

3.1 Планування розробки сервісу

Для правильної побудови сервісу з обробки даних треба крок за кроком аналізувати та вирішувати задачі, що будуть з’являтися при порівнянні вхідних даних з тим, якої структури даних необхідно досягти для подальшого використання у аналітиці. З цих міркувань побудова сервісу буде розбита на етапи:

- Дослідження джерела даних;
- Побудова структури сховища даних;
- Аналіз необхідних перетворень даних;
- Розробка конвеєру обробки даних;
- Інтеграція та побудова інтерактивних дашбордів.

3.2 Дослідження джерела даних

Виходячи з теоретичної бази наданої у попередньому розділі, першою задачею, що постає в процесі розробки – це визначення джерела даних.

У таблиці 3.1 наведено показники відвідуваності найпопулярніших онлайн-майданчиків з продажу авто.

Таблиця 3.1 – Відвідуваність сайтів онлайн-майданчиків з продажу авто.

Веб-сторінка	Частка відвідувань за останні 3 місяці	Кількість відвідувань
auto.ria.com	81,5 %	58 млн.
rst.ua	16,5 %	12 млн.

Продовження таблиці 3.1

Веб-сторінка	Частка відвідувань за останні 3 місяці	Кількість відвідувань
cars.ua	1,5 %	1 млн.
avtobazar.ua	0,5 %	350 тис.

Так як онлайн-майданчик AutoRia має найбільшу кількість відвідувань і залишає своїх конкурентів далеко позаду за цим показником, необхідно розглянути можливість збору даних про авто, що продаються саме з цієї веб-сторінки.

AutoRia пропонує надання окремого API для користувачів, що зацікавлені в інтеграції з їх сервісом. Для того, щоб отримати доступ до API, необхідно зареєструватися та вибрати план підписки. Серед інших опцій, також є варіант безкоштовної підписки з такими обмеженнями: 30 запитів на годину, 1000 всього на місяць. На рисунку 3.1 можна бачити деякі з варіантів підписок.

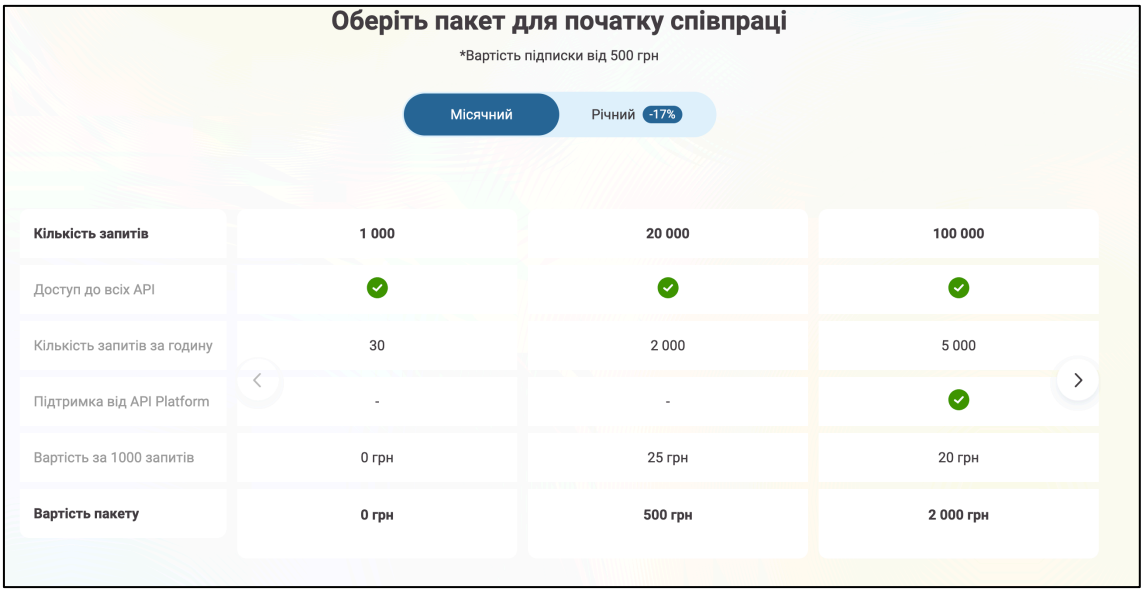


Рисунок 3.1 – Варіанти підписок на AutoRia API.

Опції підписки, що не потрапили на фрагмент екрану на рисунку 3.1, представлені за цінами 6 тис. та 9 тис. грн. Варіант за 6 тис. грн. включає 500 тисяч запитів на місяць з обмеженням у 10 тис. запитів на годину. Найдорожчий тариф підриски, що коштує 9 тис. грн., включає вже 1 мільйон запитів та обмеження у 20 тисяч запитів на годину. Так як ETL конвеєр буде виконуватися за певним графіком, наприклад кожні 3 місяці, і в час збору буде надсилати велику кількість запитів, для потреб такого аналітичного сервісу краще підійде тариф з найбільшою пропускною спроможністю і загальним числом доступних запитів.

API веб-сайту AutoRia надає такі можливості:

– Отримати списки наявних технічних характеристик авто, марок, моделей тощо.

– Провести пошук пропозицій за встановленими критеріями.

– Отримати розрахунки середньої ціни на авто з зазначеними параметрами.

– Переглядати деталі, редагувати розміщені користувачем оголошення.

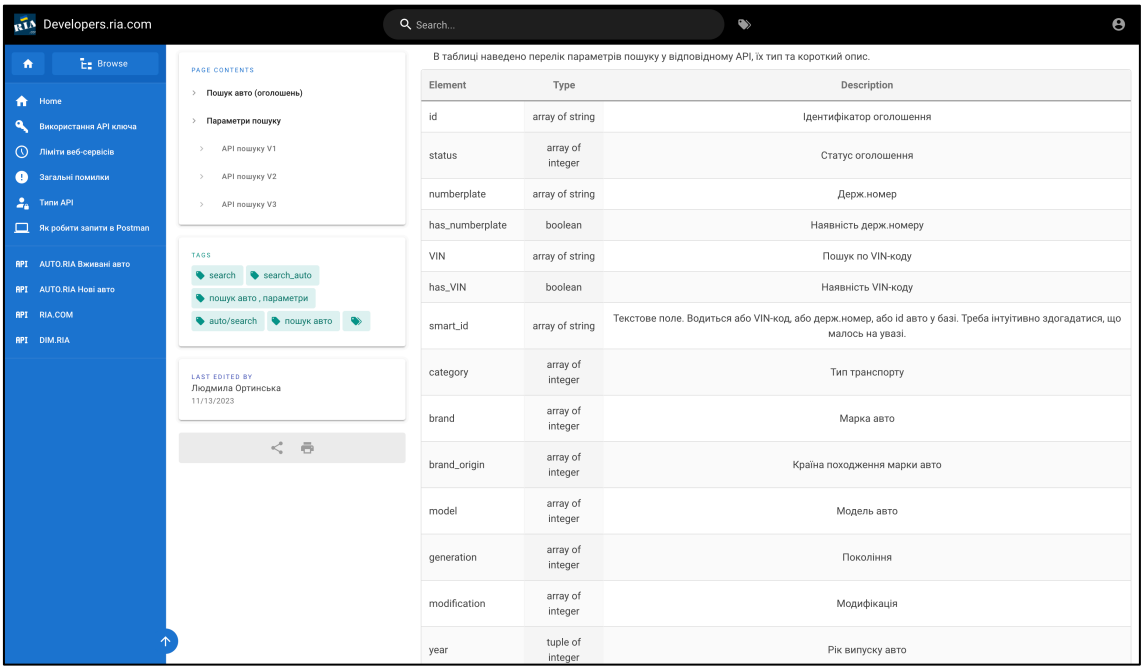
Для цілей сервісу, найкраще підходить API пошуку за зазначеними опціями. За допомогою цього API ми зможемо отримувати інформацію про кількість пропозицій, що відповідають таким критеріям як марка, модель, рік випуску, коробка передач, тип палива тощо. Маючи таку інформацію, можна буде виводити агреговані метрики та візуалізації. Приклад відповіді на запит пошуку зображено на рисунку 3.2. У полі «count» можна бачити кількість пропозицій, що було знайдено.

```
[
 {
  "additional_params": {
    "lang_id": 2,                                    // Російська мова
    "page": 0,                                       // Порядковий номер сторінки
    "view_type_id": 0,
    "target": "search",
    "section": "auto",                               // Пошук по авто
    "catalog_name": "",
    "elastica": true,
    "nodejs": true
  },
  "result": {                                        // Результат пошуку
    "search_result": {
      "ids": [                                       // id оголошень
        ....
        ....
        ....

      ],
      "count": 11,                                   // Кількість id оголошень доступних
      "last_id": 0,
      "qs": {
        "fields": [
          "_id"
        ],
        "size": 50,                                  // Кількість id оголошень, що відоб
        "from": 0,
 ]
```

Рисунок 3.2 – Приклад відповіді на запит до API пошуку авто.

Нижче на рисунку 3.3 наведено фрагмент переліку можливих параметрів фільтрації оголошень.



Developers.ria.com

Search...

Browse

Home
Використання API ключа
Ліміти веб-сервісів
Загальні помилки
Типи API
Як робити запити в Postman
API AUTO.RIA Вживані авто
API AUTO.RIA Нові авто
API RIA.COM
API DIM.RIA

PAGE CONTENTS
Пошук авто (оголошень)
Параметри пошуку
API пошуку V1
API пошуку V2
API пошуку V3

TAGS
search
search_auto
пошук авто , параметри
auto/search
пошук авто

LAST EDITED BY
Людмила Ортинська
11/13/2023

В таблиці наведено перелік параметрів пошуку у відповідному API, їх тип та короткий опис.

Element	Type	Description
id	array of string	Ідентифікатор оголошення
status	array of integer	Статус оголошення
numberplate	array of string	Держ.номер
has_numberplate	boolean	Наявність держ.номеру
VIN	array of string	Пошук по VIN-коду
has_VIN	boolean	Наявність VIN-коду
smart_id	array of string	Текстове поле. Водиться або VIN-код, або держ.номер, або id авто у базі. Треба інтуітивно здогадатися, що малось на увазі.
category	array of integer	Тип транспорту
brand	array of integer	Марка авто
brand_origin	array of integer	Країна походження марки авто
model	array of integer	Модель авто
generation	array of integer	Покоління
modification	array of integer	Модифікація
year	tuple of integer	Рік випуску авто

Рисунок 3.3 – Наявні параметри фільтрації оголошень.

Розширений список доступних параметрів включає: марка авто, модель авто, рік випуску авто, тип кузову, коробка передач, тип палива, пробіг, ціна.

Інший метод збору даних з веб-сторінок – це веб-скрейпінг. Для оцінки можливості використання даного методу треба дослідити структуру веб-сторінок AutoRia та визначити на якій сторінці та які елементи можна відслідковувати та зчитувати їх значення. Після огляду кількох сторінок, було виявлено, що для цілей проекту може підійти сторінка пошуку авто за критеріями. З цією сторінкою можна буде працювати подібно до API, тобто передавати параметри пошуку та отримувати кількість наявних пропозицій на ринку. Приклад сторінки з виділеним елементом що відповідає за відображення інформації про кількість пропозицій наведено на рисунку 3.4.

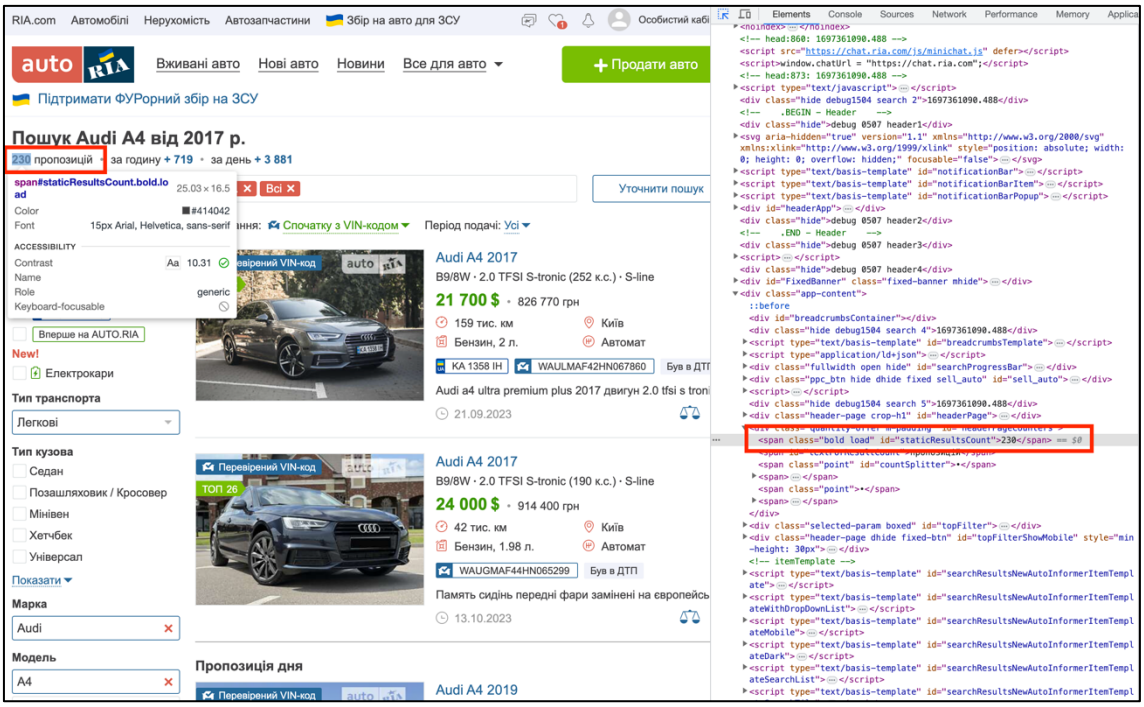


Рисунок 3.4 – Приклад веб-сторінки пошуку авто.

Передача параметрів пошуку буде відбуватися у параметрах пошукової стрічки, як у випадку з API, так і з веб-скрейпінгом.

Використання наданого API над веб-скрейпінгу дає явні переваги, які сприяють більш ефективному та надійному процесу збору даних. По-перше, API

призначені для забезпечення структурованого доступу до даних, забезпечуючи стандартизований і передбачуваний формат для пошуку інформації. Цей структурований підхід покращує цілісність даних, зменшуючи ймовірність помилок, пов'язаних із аналізом та інтерпретацією неструктурованого веб-вмісту.

По-друге, API дотримуються визначеного набору ендпоінтів і протоколів, сприяючи більш стабільному процесу інтеграції. Цей стандартизований інтерфейс спрощує реалізацію та зменшує складність вилучення даних у порівнянні зі спеціальним характером веб-скрейпінгу, коли зміни в структурі веб-сайту можуть порушити процес збору даних.

Крім того, API спеціально розроблені для обміну даними, сприяючи більш етичному та законному процесу пошуку даних. Навпаки, веб-скрейпінг може викликати юридичні проблеми, пов'язані з порушеннями умов обслуговування та прав інтелектуальної власності, що робить використання API більш надійним рішенням.

API також підвищують масштабованість, оскільки вони часто підтримують механізми розбиття на сторінки та пакетування, полегшуючи пошук великих наборів даних у систематичний спосіб. Ця масштабованість є особливо вигідною в сценаріях, коли веб-скрейпінг може зіткнутися з обмеженнями, накладеними серверами веб-сайтів, або викликати проблеми, пов'язані з надмірною кількістю запитів.

Також, API зазвичай надають вичерпну документацію з детальним описом доступних ендпоінтів, параметрів запитів і форматів відповідей. Ця документація служить цінним ресурсом для розробників, забезпечуючи ясність і легкість впровадження в порівнянні з підходом проб і помилок, який часто пов'язують із веб-скрейпінгу.

Надійність пошуку даних на основі API додатково підсилюється механізмами керування версіями, які підтримують сумісність із наборами даних і бізнес-логікою, що піддаються змінам. З іншого боку, веб-скрейпінг вимагає

постійного моніторингу та адаптації до змін на веб-сайті, створюючи додаткові витрати на обслуговування.

API сприяють підвищенню продуктивності, мінімізуючи обсяг даних, що передаються між сервером і клієнтом. Вони дозволяють вибірково отримувати дані, зменшуючи використання смуги пропускання та затримку порівняно з невибірковим вилученням даних, властивим веб-збиранню.

Підсумовуючи, можна сказати, що використання наданого AutoRia API замість веб-скрейпінгу узгоджується з найкращими практиками збору даних, пропонуючи переваги пов’язані з структурованістю даних, стабільністю, відповідністю, масштабованістю, надійністю та продуктивністю. Ці фактори разом сприяють більш надійному та стійкому підходу до процесу збору даних.

3.3 Побудова структури сховища даних

Нижче наведено список доступних параметрів для фільтрації оголошень:

- марка авто,
- модель авто,
- рік випуску авто,
- тип кузову,
- коробка передач,
- тип палива,
- пробіг,
- ціна.

Завдяки різноманітному поєднанню цих параметрів можна отримувати різні значення такої метрики як кількість або оголошень, що є на даний момент на AutoRia. Метрика відображає кількісні характеристику пропозиції авто на ринку за заданими параметрами. Саме цю метрику буде прийнято за наявні

фактичні дані та буде використано у побудові відповідної таблиці з фактами у сховищі даних.

В результаті огляду наявних параметрів та метрики, буде створено наступні об’єкти у сховищі даних:

– Таблиця OFFERS_FACT – для зберігання даних про кількість пропозицій за сукупністю параметрів.

– Таблиця BODY_TYPE_DIM – для зберігання даних про наявні типи кузовів.

– Таблиця GEARBOX_DIM – для зберігання даних про наявні типи коробки передач.

– Таблиця FUEL_DIM – для зберігання даних про наявні типи палива.

– Таблиця MANUFACTURER_DETAILS_DIM – для зберігання даних про виробника.

– Таблиця MILEAGE_DIM – для зберігання даних про діапазони пробігу.

На рисунку 3.5 зображено діаграму, що показує структуру сховища даних та зв’язки між об’єктами.

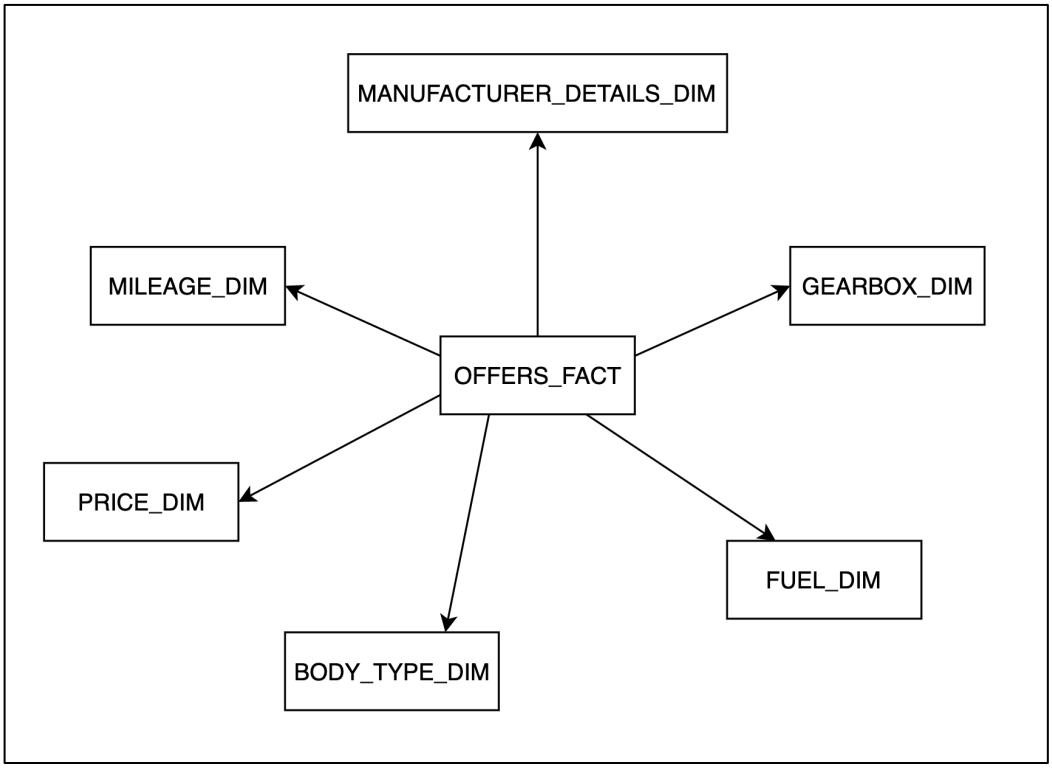


Рисунок 3.5 – Діаграма структури сховища даних.

На діаграмі зображено схему даних за шаблоном Зірка, який було детально описано в попередніх розділах. У центрі структури знаходиться таблиця з фактами, а навколо неї таблиці що містять деталі фактів, і таким чином можуть уточнювати їх. Така схема дозволяє зручно та ефективно виконувати запити аналітичного типу без їх перевантаження з’єднаннями, що часто трапляється при нормалізації даних.

Наступним кроком буде уточнення діаграми шляхом додавання атрибутів. На рисунку 3.6 наведено деталізовану діаграму побудови сховища даних. Вимірні таблиці поєднані з таблицею фактів зв’язком багато-до-багатьох. Це тип зв’язку між двома таблицями, де кожен запис в одній таблиці може бути пов’язаний з кількома записами в іншій таблиці, і навпаки.

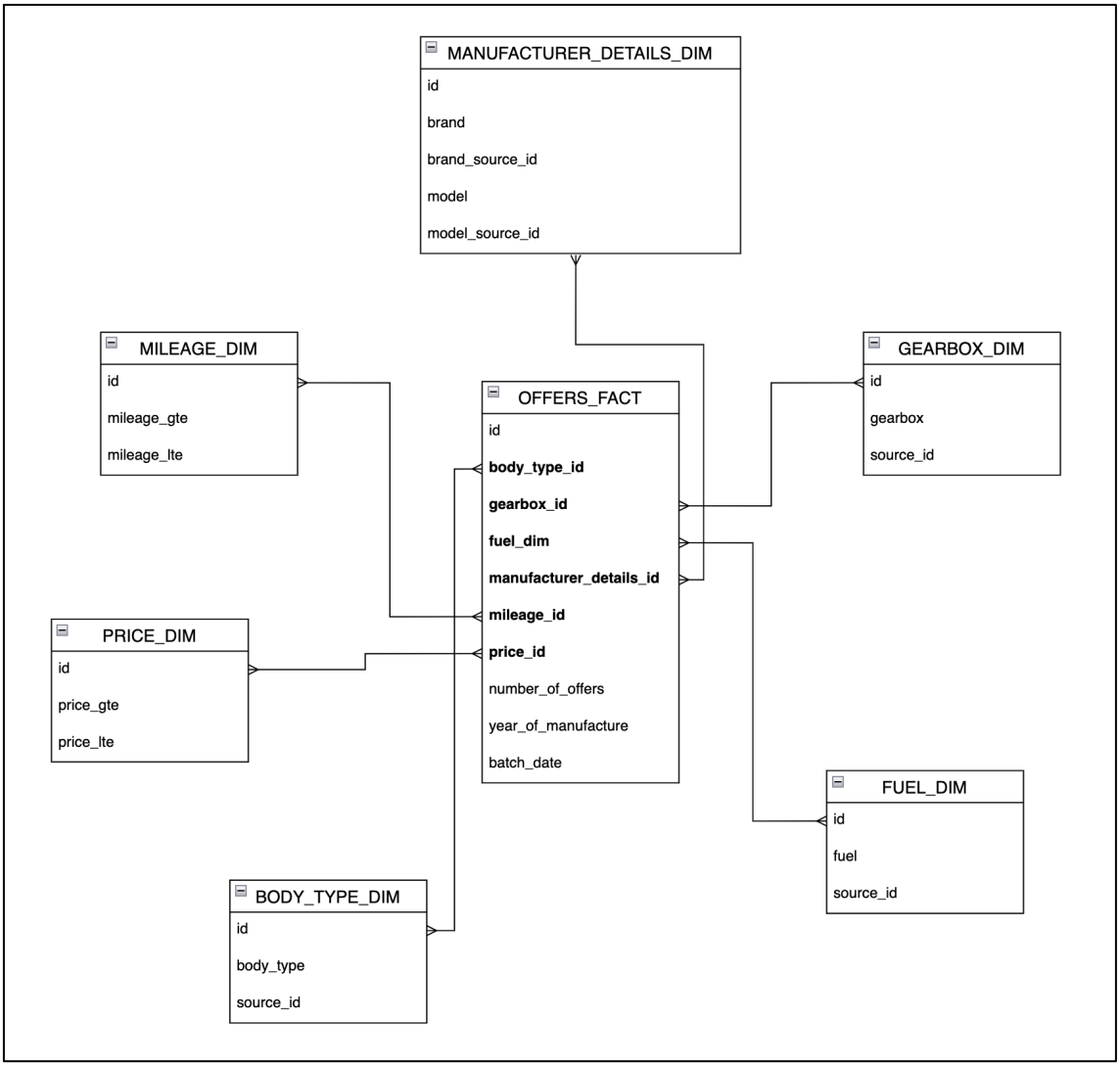


Рисунок 3.6 – Деталізована діаграма структури сховища даних.

Таблиця OFFERS_FACT має такі атрибути: унікальний ідентифікатор, кількість пропозицій, рік випуску авто, дата завантаження пакету даних та зовнішні ключі до вимірних таблиць. Атрибут «рік випуску авто» було вирішено розмістити у фактичній таблиці задля оптимізації продуктивності роботи з таблицею. Винесення цього атрибуту у окрему вимірну таблицю додало би додаткове з’єднання таблиць, що вплинуло би на продуктивність системи, адже рік випуску авто є популярним параметром для фільтрації пропозицій, проте не додало би ніякої практичної користі.

Таблиця MANUFACTURER_DETAILS_DIM містить атрибути: унікальний ідентифікатор, назва бренду, унікальний ідентифікатор бренду з

джерела даних, назва моделі, унікальний ідентифікатор моделі з джерела даних. Додавання унікальних ідентифікаторів, що використовуються на стороні джерела даних пов’язані з потребою з’єднання вхідних даних з даними, що містяться у сховищі.

Таблиця GEARBOX_DIM містить такі атрибути: унікальний ідентифікатор, назва типу коробки передач, унікальний ідентифікатор коробки передач з джерела даних.

Таблиця FUEL_DIM містить такі атрибути: унікальний ідентифікатор, назва типу палива, унікальний ідентифікатор палива з джерела даних.

Таблиця BODY_TYPE_DIM містить такі атрибути: унікальний ідентифікатор, назва типу кузова, унікальний ідентифікатор кузова з джерела даних.

Таблиця MILEAGE_DIM містить такі атрибути: унікальний ідентифікатор, значення початку діапазону значень пробігу (включно), значення закінчення діапазону значень пробігу (включно). Для цілей оптимізації, було вирішено групувати значення пробігу у діапазони з кроком у 50 тисяч кілометрів.

Таблиця PRICE_DIM містить атрибути: унікальний ідентифікатор, значення початку діапазону значень ціни (включно), значення закінчення діапазону значень ціни (включно). Для цілей оптимізації, було вирішено групувати значення ціни у діапазони з кроком у 1 тисячу дол. США.

3.4 Аналіз необхідних перетворень даних

Для того, щоб визначити задачі конвеєру даних, потрібно провести аналіз вхідних даних та цільової структури даних, яку має вже спроектоване сховище даних.

Вхідні дані представлені атрибутами, що містяться у відповідях, що надсилає API сервіс AutoRia на відповідні запити. При аналізі вхідних даних було виявлено, що немає необхідності у складних перетворюваннях. Вхідні дані

повністю переносяться у сховище даних у визначені таблички. Єдина відмінність – це назви полів, що не має суттєвого впливу на роботу системи. У таблиці 3.2 наведено матрицю відповідності атрибутів у вхідних даних з атрибутами у сховищі даних.

Таблиця 3.2 – Таблиця відповідності атрибутів джерела даних з атрибутами у сховищі даних.

Атрибут вхідних даних	Відповідний атрибут у сховищі даних
bodyStyle	BODY_TYPE_DIM.source_id
marka_id	MANUFACTURER_DETAILS_DIM.brand_source_id
model_id	MANUFACTURER_DETAILS_DIM.model_source_id
s_yers	OFFERS_FACT.year_of_manufacture
po_yers	OFFERS_FACT.year_of_manufacture
price_ot	PRICE_DIM.price_gte
price_do	PRICE_DIM.price_lte
type	FUEL_DIM.source_id
gearbox	GEARBOX_DIM.source_id
result.search_result.count	OFFERS_FACT.number_of_offers

Підсумовуючи, в ході дослідження схем вхідних та цільових даних не було виявлено необхідності у перетворювані даних. Це матиме безпосередній вплив на архітектуру конвеєру даних, адже виключається елемент важких обчислень.

3.5 Розробка конвеєру обробки даних

В результаті попередніх досліджень було виявлено, що відсутня необхідність у складних перетвореннях даних на проміжку від їх збору до завантаження у сховище даних.

Для такого випадку конвеєр ELT виглядає гарним вибором, адже має такі переваги:

1. Простота етапу доставки даних. ELT спрощує процес обробки даних шляхом прямого збору даних і завантаження їх у цільову систему зберігання даних, таку як сховище даних, без проміжних кроків перетворення. Ця простота вигідна, коли трансформації мінімальні або зовсім відсутні.

2. Зменшення накладних витрат часу. ELT усуває потребу в проміжному етапі перетворення, зменшуючи витрати часу на налаштування та підтримку окремої інфраструктури для обробки даних.

3. Зменшення грошових витрат. ELT буде економічно ефективнішим, оскільки зменшується потреба в проміжних інструментах і процесах трансформації.

ELT конвеєр містить 3 етапи: збір (англ. *extract*), доставка(англ. *load*) та трансформація (англ. *transform*). Далі буде детально розглянуто та спроектовано кожний окремий етап конвеєру.

Збір даних. Так як джерелом даних є API від AutoRia, необхідно створити компонент-збирач (англ. *Extractor*), логіка якої буде включати в себе наступні кроки:

- Надіслати запит до API
- Отримати відповідь від серверу
- Вибрати необхідні атрибути та записати їх в об’єкт
- Додати об’єкт до результуючого списку
- Записати результуючий список на диск

З наведених вище задач відразу постає інша задача – розробка плану запитів, за яким програма-збирач буде виконувати запити. Для того, щоб скласти план запитів, необхідно заздалегідь знати всі варіанти параметрів для створення певного списку з їх комбінацій.

Для цього треба створити ще компонент-планер (англ. *Planner*), що буде виконуватися перед збирачем та буде формувати план API запитів. Задачами цього компоненту будуть:

– Підключитися до сховища даних
– Завантажити дані з DIM таблиць
– Створити список з комбінаціями параметрів
– Передати список комбінацій компоненту-збирачу

Для підтримки такої функціональності необхідно попередньо заповнити DIM таблиці даними. Така процедура буде необхідна тільки перед першим запуском конвеєру. Для попереднього наповнення DIM таблиць треба створити програму-заповнювач (англ. *Preloader*), що виконає наступні кроки:

– Надіслати запит до API параметрів, щоб отримати всі його можливі значення
– Отримати відповідь від серверу
– Підключитися до сховища даних
– Заповнити DIM таблиці даними із відповіді

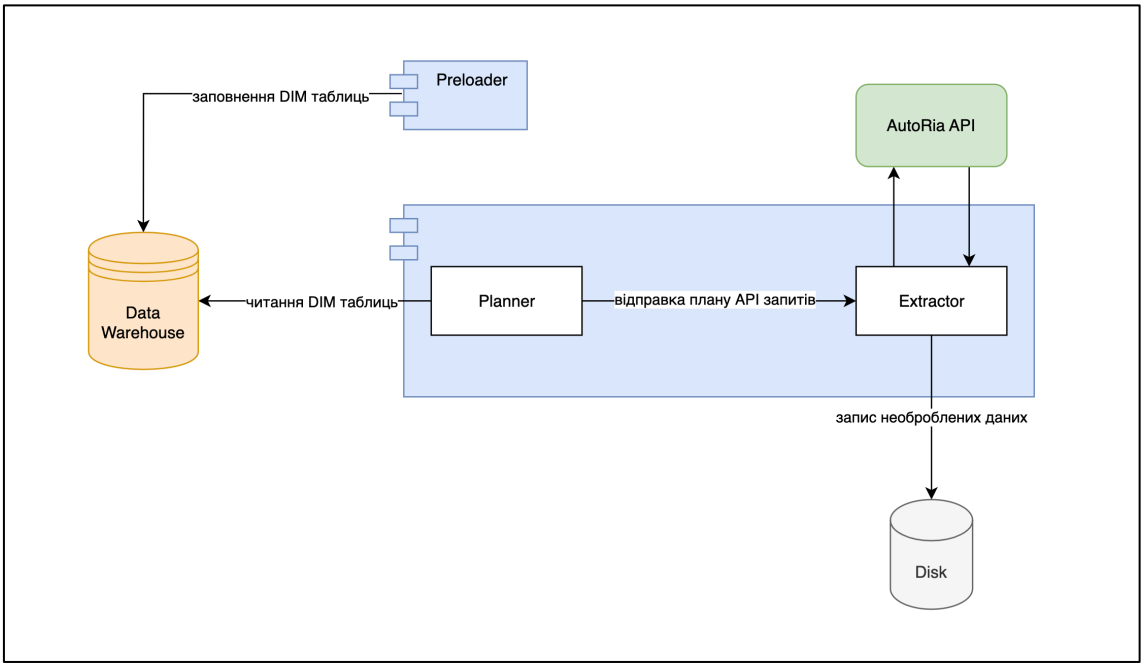


Рисунок 3.7 – Діаграма етапу збору даних.

На рисунку 3.7 зображено загальну архітектуру етапу збору даних. Можна бачити 2 окремих модуля. Перший, що має назву Preloader, виконує функцію заповнення даними DIM таблиць у сховищі даних. Інший, має в своєму складі 2 окремих компоненти Planner та Extractor. Planner створює план API запитів для Extractor, а він в свою чергу виконує ці запити та зберігає необроблені дані на диск.

Доставка даних. Після того як дані було зібрано на збережено на диск, настає наступний етап конвеєру, а саме завантаження даних до сховища даних.

На даному етапі вже доступні необроблені дані на диску. Для того, щоб завантажити дані до таблиць необхідно як мінімум замінити унікальні ідентифікатори AutoRia, що надходили від API, на внутрішні, що були створені при заповнені DIM таблиць. Такого роду задачі вирішуються через додавання проміжних таблиць всередині сховища даних, тобто необроблені дані будуть завантажуватися не у фінальну OFFERS_FACT таблицю, а у проміжну, наприклад, OFFERS_FACT_LDG. На рис.3.8 зображено архітектуру етапу даставки даних до сховища даних.

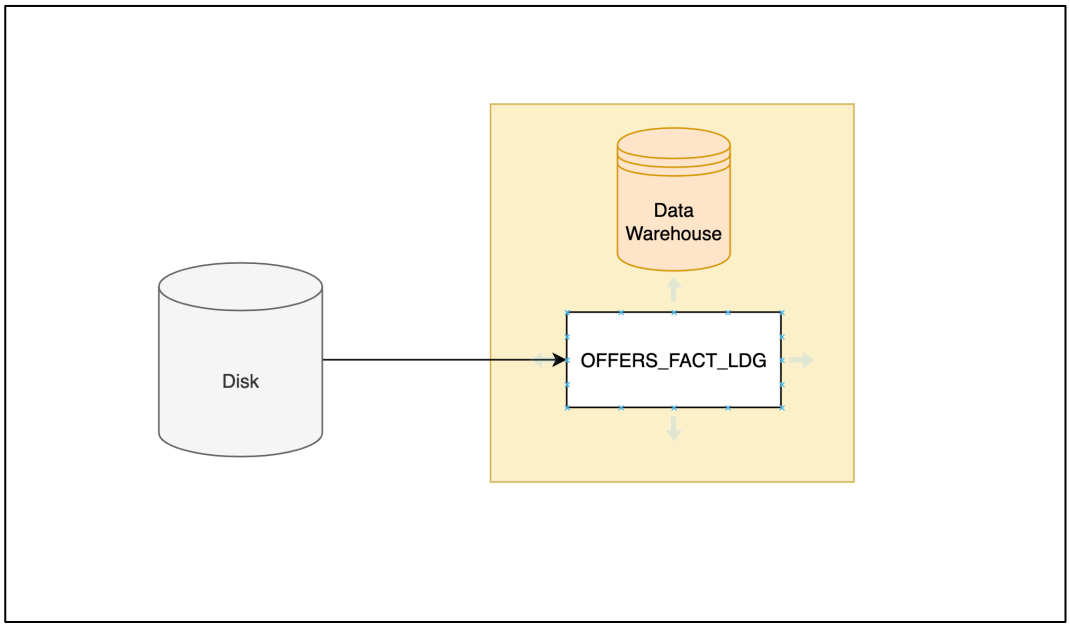


Рисунок. 3.8 – Діаграма етапу доставки даних.

На діаграмі видно, що дані рухаються з диску до сховища даних, а саме проміжної таблиці OFFERS_LDG.

Обробка даних. На останньому етапі ELT конвеєру виконуються фінальні перетворення та дані завантажуються у цільові таблиці. Для даних, що надійшли до OFFERS_FACT_LDG таблиці, необхідне певне збагачення, а саме додавання внутрішніх ідентифікаторів DIM таблиць, які будуть використовуватися при завантажені даних у таблицю OFFERS_FACT. На рисунку 3.9 зображено етап обробки даних.

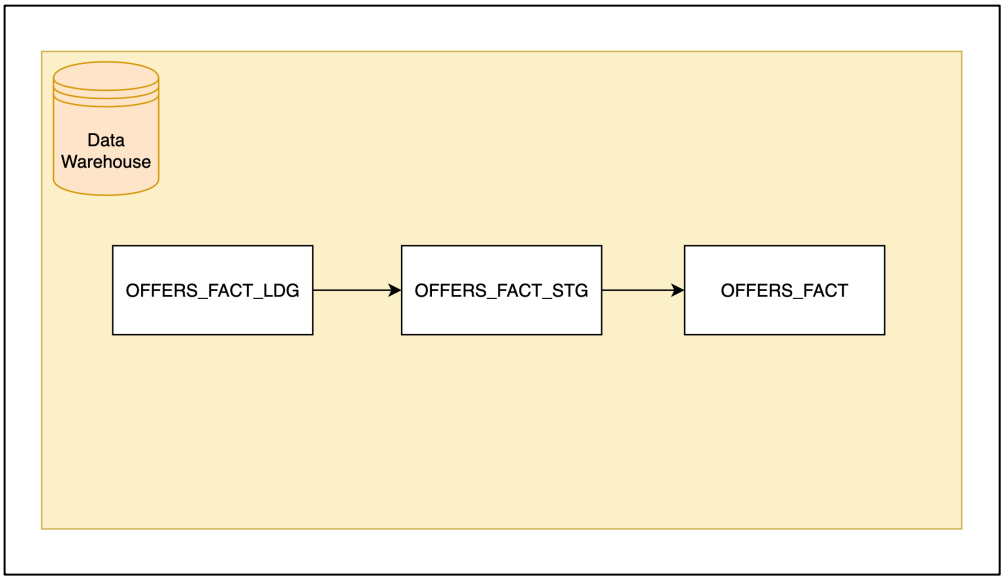


Рисунок 3.9 – Діаграма етапу обробки даних.

На діаграмі відображено рух даних від таблиці OFFERS_FACT_LDG до OFFERS_FACT. Збагачення даних відбувається на етапі завантаження даних до OFFERS_FACT_STG таблиці, а після цього вже повністю готові дані надходять до фінальної таблиці.

Попередньо вже було розглянуто та детально описано окремі елементи конвеєру даних. Для кращого розуміння системи створено загальну діаграму конвеєру (рисунок 3.10), на якій позначено всі етапи руху даних.

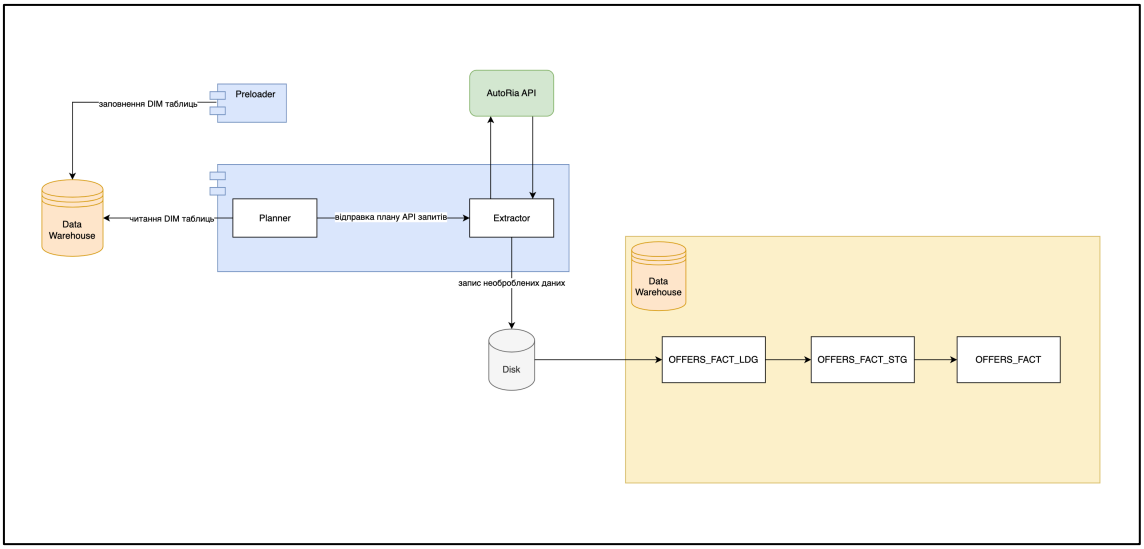


Рисунок 3.10 – Загальна архітектура конвеєру даних.

Дані збираються з API AutoRia та записуються на диск. З диску дані прямують до проміжної таблиці сховища даних, а потім, проходячи через певні перетворення потрапляють до фінальної OFFERS_FACT таблиці та стають доступними для кінцевого використання.

3.6 Інтеграція та побудова інтерактивних дашбордів

Tableau пропонує безкоштовну версію, що має назву Tableau Public. Платформа вимагає створення профілю для доступу до створення дашбордів та використання всього функціоналу. На рисунку 3.11 зображено екран створення профілю. Після заповнення всіх полів треба натиснути на кнопку створити профіль та підтвердити реєстрацію у листі, що буде вислано на електронну пошту.

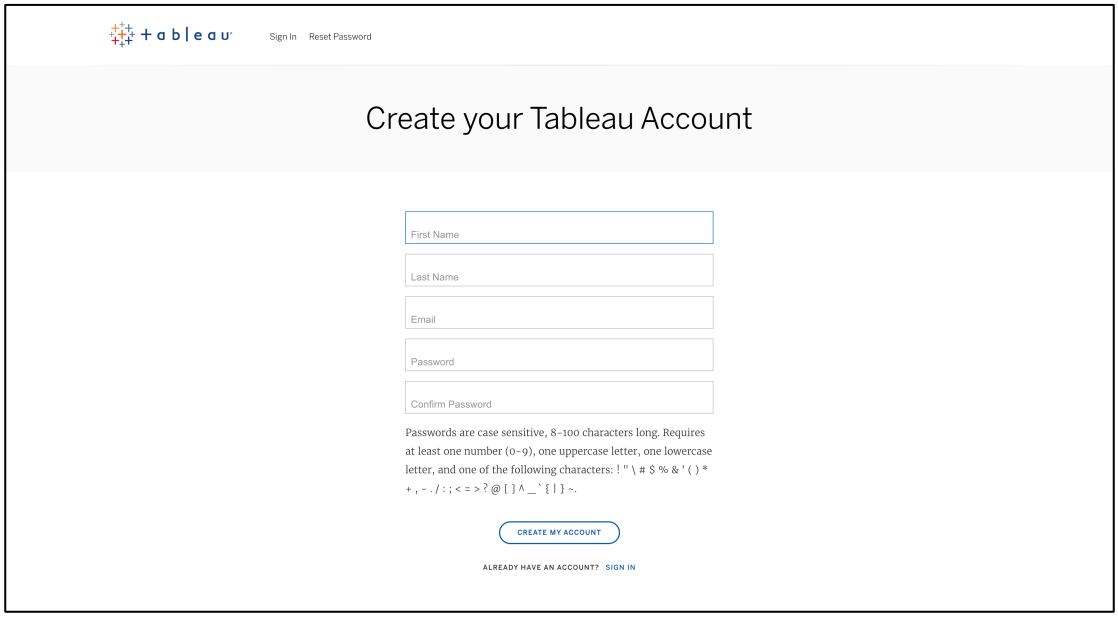


Рисунок 3.11 – Екран створення профілю Tableau.

Після входу на платформу стане доступний стартовий екран. Для того створити дашборд треба натиснути на кнопку «Create», що знаходиться у лівому верхньому кутку веб-сторінки (рисунок 3.12).

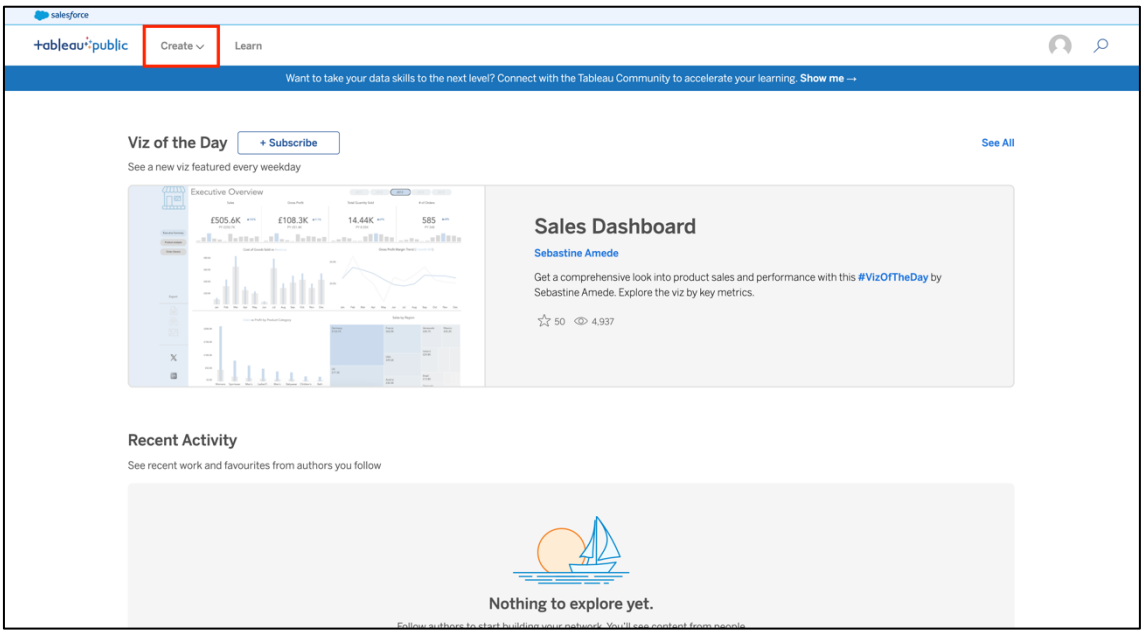


Рисунок 3.12 – Стартовий екран Tableau Public.

Для створення доступні дві опції: створення онлайн дашборду та завантаження настільної версії застосунку Tableau для роботи локально.

Першим завданням у побудові дашборду буде створення з’єднання до сховища даних. Для цього треба надати адресу серверу та дані для автентифікації (рисунок 3.13).

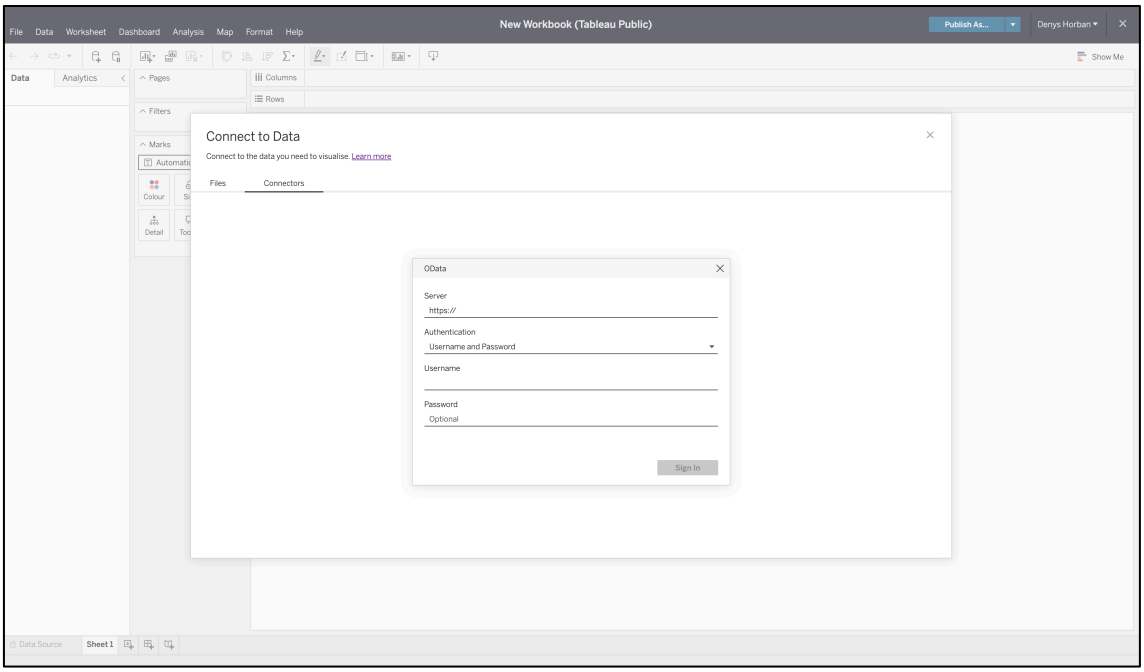


Рисунок 3.13 – Екран налаштування джерела даних.

Наступним етапом буде перевірка завантажених даних. На рисунку 3.14 зображено структуру таблиць, що було завантажено, а нижче видно самі дані з таблиці OFFERS_FACT.

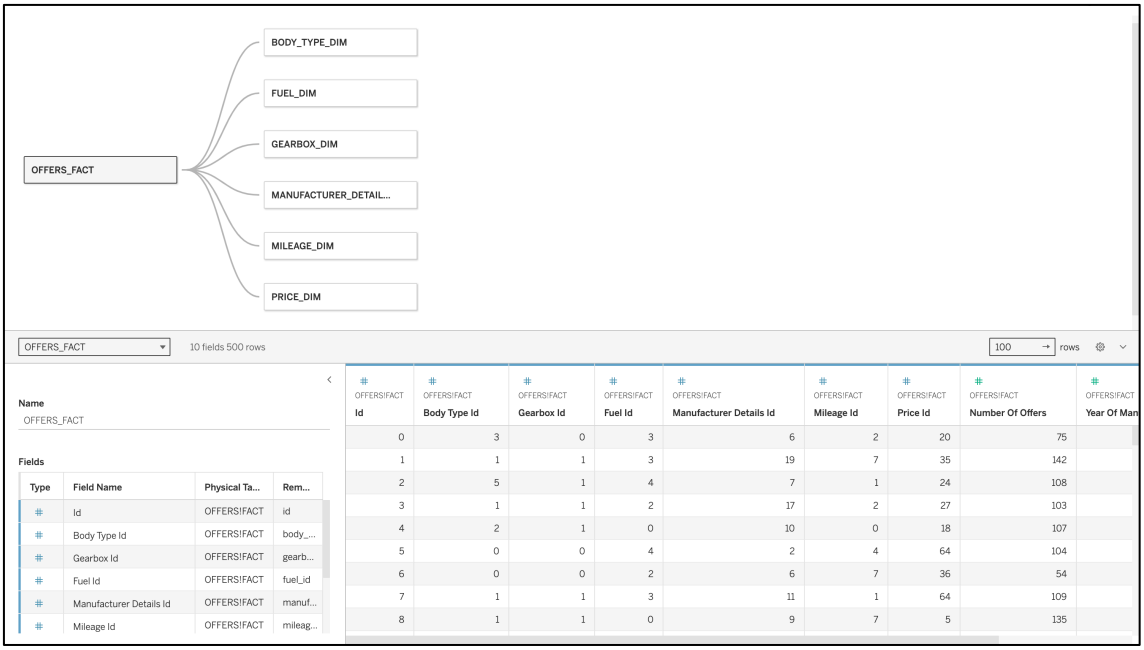


Рисунок 3.14 – Перегляд завантажених даних на платформі Tableau.

Далі буде створено декілька графіків та діаграм для включення до дашборду. На рисунку 3.15 зображено процес розробки графіку.

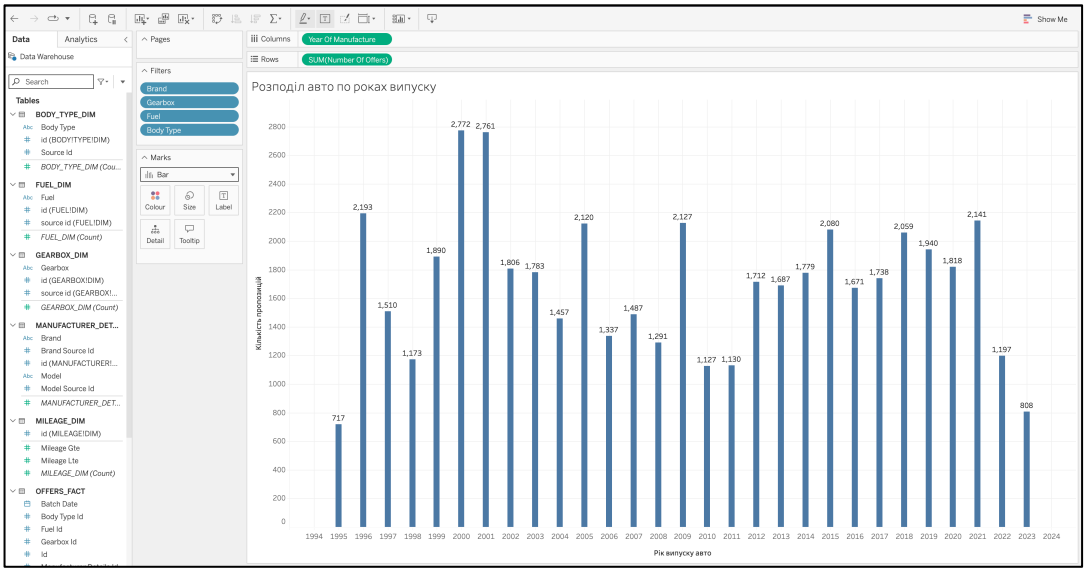


Рисунок 3.15 – Процес розробки графіку на платформі Tableau.

Даний графік буде відображати кількість авто що продаються в розподілі по роках їх випуску. Дані щодо пропозицій будуть агрегуватися по роках шляхом

додавання. Також, графік буде адаптуватися під чотири види фільтрів, які вибрано у відповідному віконці «Filters».

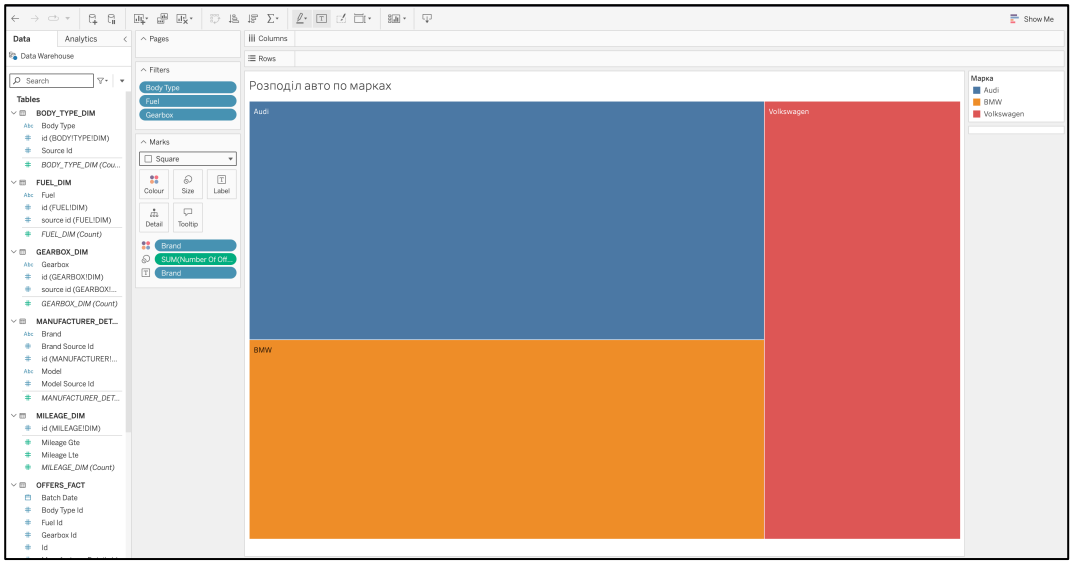


Рисунок 3.16 – Процес створення прямокутної діаграми на платформі Tableau.

У доповнення до графіку буде додано ще 2 таблиці та 1 діаграму. Таблиці будуть містити інформацію про частку пропозицій в розрізі по типи кузову та палива. Діаграма буде мати форму прямокутника з замальованими площами, що відповідають частці тієї чи іншої марки відносно всієї сукупності пропозицій. На рисунку 3.16 наведено процес створення такої діаграми.

У вікні створення дашборду з лівої сторони будуть доступні створені таблиці, діаграми та графіки. Шляхом їх перетягування на робочу зону вони будуть додані до дашборду. Як видно з рисунку 3.17, графік було розміщено вгорі, а відразу під ним знаходяться таблиці та діаграма розподілу по марках.

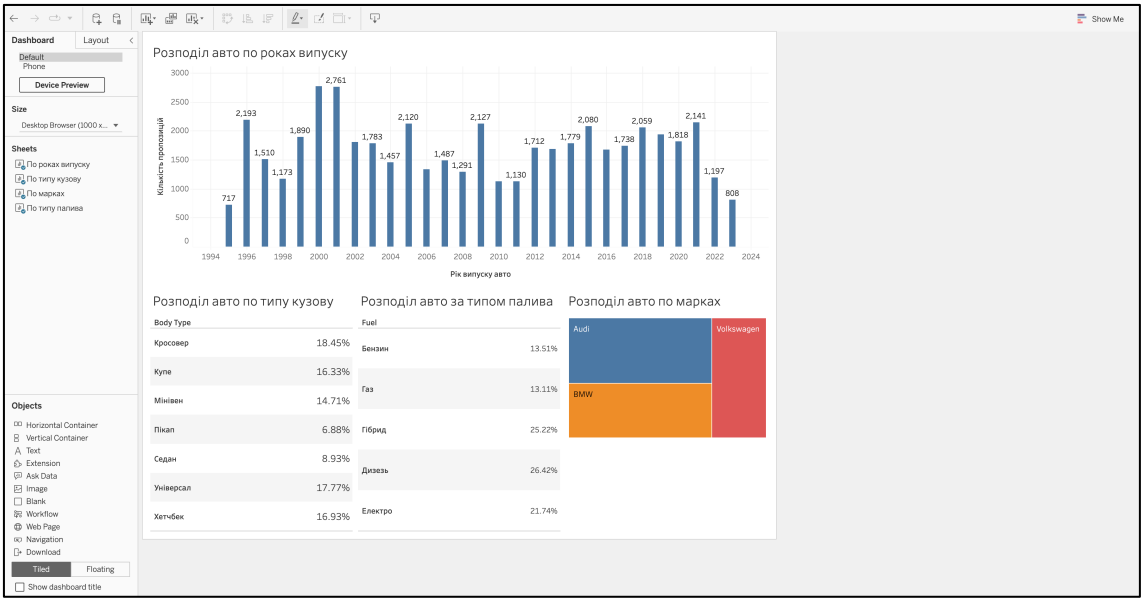


Рисунок 3.17 – Створення дашборду на платформі Tableau.

Таким чином, на платформі Tableau був створений інтерактивний дашборд, заснований на інформації зі сховища даних, які, у свою чергу, були завантажені з сервісу AutoRia.

3.7 Висновки до розділу

У розділі було підкреслено важливість методичного підходу до побудови сервісу обробки даних і відповідно визначено план реалізації аналітичного сервісу.

Було досліджено варіанти взаємодії з джерелом даних – веб-сервісом AutoRia. Було виявлено два способи збору даних: використання API та веб-скрейпінг. Використання готового API над веб-скрейпінгу дає явні переваги, які сприяють більш ефективному та надійному процесу збору даних. API призначені для забезпечення структурованого доступу до даних, забезпечуючи стандартизований і передбачуваний формат для пошуку інформації. Цей структурований підхід покращує цілісність даних, зменшуючи ймовірність

помилок, пов’язаних із аналізом та інтерпретацією неструктурованого веб-вмісту.

На основі вхідного набору даних було створено структуру сховища даних за шаблоном Star. Центральною таблицею фактів є OFFERS_FACT, що оточено вимірними таблицями, які містять додаткову інформацію про факт, то можуть легко бути з’єднані завдяки зовнішнім ключам. В ході порівняння було побудовано таблицю відповідності атрибутів між вхідним набором даних та структурою сховища даних. За результатами аналізу цієї таблиці було визначено ступінь складності перетворень вхідних даних.

Було побудовано конвеєр даних за схемою ELT, так як такий підхід дозволив спростити реалізацію та повністю задовольняє вимогам. Було детально описано кожний етап конвеєру та надано відповідні діаграми, що показують рух даних у зрозумілій та простій формі.

Кінцевим місцем використання вже готових даних стала платформа інтерактивних дашбордів Tableau. Дані зі сховища були завантажені на платформу та опрацьовані засобами платформи. Було створено декілька графічних елементів, а саме таблиці, графік та діаграма, що потім були поєднані в процесі створення дашборду.

4 ТЕСТУВАННЯ СТВОРЕНОГО ФУНКЦІОНАЛУ

Тестування програмного забезпечення є одним з етапів життєвого циклу розробки програмного забезпечення. Задача тестування – це перевірка коректності, функціональності та надійності розробленої програмної системи. Процес тестування передбачає систематичне виконання програм для тестування компонентів або систем з метою виявлення помилок. Однією з основних цілей тестування є виявлення дефектів або розбіжностей між очікуваними та фактичними результатами. Процес тестування має різні підходи, включаючи ручний та автоматизований, кожен з яких має свої сценарії застосування.

Ручне тестування – це підхід, що передбачає дії людини, спрямовані на виконання тестових сценаріїв та документування результатів. Він пропонує якісну оцінку досвіду користувача та інтуїтивне дослідження потенційних проблем. Автоматизоване тестування покладається на спеціалізовані інструменти для виконання повторюваних і складних програмних тестів.

Також, тестування розділяють на такі підтипи:

- функціональне тестування (functional testing);
- нефункціональне тестування (non-functional testing);
- структурне тестування (structural testing);
- тестування змін (change related testing).

4.1 Функціональне тестування

Функціональне тестування є важливим етапом у розробці програмного забезпечення, що спрямований на перевірку певних функціональних можливостей системи відповідно до попередньо визначених вимог. Цей процес тестування систематично оцінює різні компоненти та аспекти програмного забезпечення, щоб переконатися в його правильній поведінці та функціональності. Він передбачає оцінку окремих функцій та особливостей

програми, таких як перевірка вхідних даних, взаємодія з інтерфейсом користувача, маніпулювання даними та генерація вихідних даних.

Функціональне тестування охоплює кілька методів, включаючи модульне тестування, коли окремі одиниці або компоненти тестуються ізольовано, та інтеграційне тестування, яке оцінює взаємодію між цими одиницями. Крім того, тестування системи використовується для перевірки відповідності програмного забезпечення загальним функціональним вимогам, гарантуючи, що інтегровані компоненти функціонують бездоганно разом.

4.2 Нефункціональне тестування

Нефункціональне тестування є невід'ємним аспектом тестування програмного забезпечення, який зосереджується на оцінці атрибутів, що виходять за межі конкретних функцій системи. Його метою є оцінка нефункціональних аспектів програмного забезпечення, зокрема продуктивності, надійності, зручності використання та безпеки.

Тестування продуктивності є важливою частиною нефункціонального тестування, яке зосереджується на визначенні швидкості реагування, масштабованості та стабільності програмного забезпечення за різних навантажень і умов.

Тестування надійності передбачає перевірку здатності програмного забезпечення працювати стабільно та надійно протягом тривалого періоду часу, часто у формі стрес-тестування, щоб оцінити його стійкість у складних умовах.

Тестування зручності використання оцінює зручність програмного забезпечення, гарантуючи, що воно забезпечує інтуїтивно зрозумілу та ефективну роботу користувача.

Тестування безпеки, ще один важливий нефункціональний компонент тестування, зосереджується на виявленні вразливостей і слабких місць у заходах безпеки програмного забезпечення, захисту від потенційних загроз.

4.3 Структурне тестування

Структурне тестування, також відоме як тестування білої скриньки, — це метод, який досліджує внутрішню логіку та структуру програмного забезпечення, ретельно вивчаючи його код, архітектуру та дизайн. Основна мета — оцінити внутрішні компоненти програмного забезпечення, переконавшись, що кожен елемент функціонує належним чином і що загальна структура підтримує бажані функції.

Перевірка гілок логіки є ще одним важливим аспектом, який вивчає виконання різних гілок логіки, що містяться у коді та виконуються при виконані тих чи інших умов. Покриття гілок робить структурне тестування ще одним кроком вперед, аналізуючи унікальні шляхи виконання програми. Ця ретельна перевірка внутрішньої структури програми допомагає виявити потенційні дефекти, перевірити правильність алгоритмів і забезпечити передбачувану поведінку при різних сценаріях.

Структурне тестування є особливо цінним на ранніх стадіях розробки, допомагаючи виявити проблеми на рівні коду та полегшуючи ефективне налагодження. Надаючи розуміння тонкощів внутрішньої роботи програмного забезпечення, структурне тестування сприяє створенню надійних програмних систем, які можна легко підтримувати.

4.4 Тестування змін

Тестування, пов’язане зі змінами, яке часто називають регресійним тестуванням, є критичним етапом життєвого циклу розробки програмного забезпечення, який зосереджується на перевірці функціональності системи після внесення змін, удосконалень або оновлень. Його основна мета полягає в тому,

щоб внесені зміни не вплинули негативно на існуючі функції програмного забезпечення.

Цей підхід до тестування передбачає повторне виконання раніше проведених тестів для виявлення будь-яких ненавмисних побічних ефектів або дефектів, що виникають через внесені зміни. Мета полягає в тому, щоб перевірити коректність системи та підтримувати цілісність програмного забезпечення в цілому, незважаючи на поточні зміни. Тестування, пов’язане зі змінами, має ключове значення для надання високоякісного програмного забезпечення та мінімізації ризику виникнення нових проблем під час розвитку проекту програмного забезпечення.

4.5 Тестування розробленого функціоналу

Протягом всього циклу розробки до функціональності аналітичного сервісу застосовувалися різні типи тестування. Весь функціонал було піддано регресійному тестуванні при завершенні його розробки. Всі етапи тестування наведено у таблиці 4.1.

Таблиця 4.1 Тестування розробленого функціоналу

Цикл розробки	Тип тестування	Результат тестування
Ініціалізація проекту	Функціональне тестування	Підключення до сховища даних, створення таблиць
Підключення API	Функціональне тестування Структурне тестування	Отримання відповіді від API marks, bodystyles, gearboxes, search.

Продовження таблиці 4.1

Цикл розробки	Тип тестування	Результат тестування
Розробка Extract частини конвеєру	Функціональне тестування	Готова Extract частина конвеєру даних
Розробка Load частини конвеєру	Функціональне тестування	Готова Load частина конвеєру даних
Розробка Transform частини конвеєру	Функціональне тестування	Готова Transform частина конвеєру даних
Виявлення проблем та їх усунення	Повторне тестування	Усунуто помилки в схемі таблиць у сховищі даних
Перевірка роботи функціоналу після імплементації конвеєру даних	Регресійне тестування Тестування стабільності системи	Виявлення проблем та робота над їх усуненням
Розробка Tableau дашборду	Функціональне тестування	Готовий дашборд на платформі Tableau
Усунення проблем після регресійного тестування	Повторне тестування	Усунуто помилки інтерпретації даних в окремих діаграмах Tableau
Рефакторинг коду	Структурне тестування Регресійне тестування	Конфігурація додаткових змінних, оптимізація кодової бази

Тестування під час розробки сервісу можна умовно поділити на 3 логічні частини:

– Тестування збору та доставки даних. Такі процеси як збір і доставка даних виконуються Python застосунками. Під час розробки цих окремих компонентів створювалися юніт-тести, що документували у коді логіку процесів

та очікуваний результат. На рисунку 4.1 наведено результат виконання всіх юніт-тестів для створених Python компонентів.

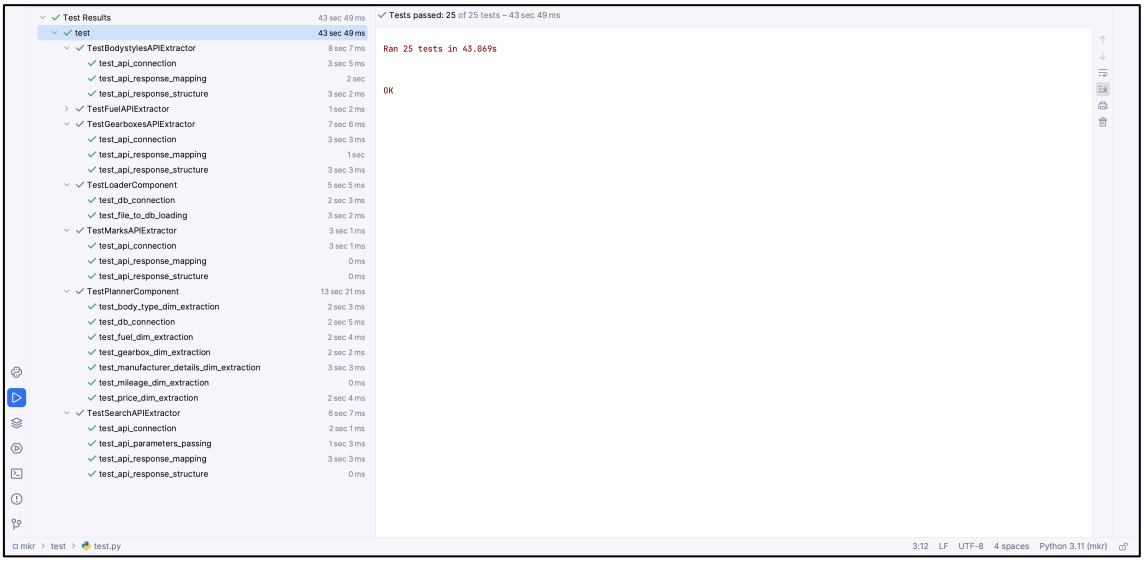


Рисунок 4.1 – Результат виконання юніт-тестів.

– Тестування обробки даних у сховищі. Цей процес включав у себе виконання SQL скриптів та ручну валідацію результатів у цільових таблицях бази даних базуючись як на порівнянні як агрегованих показників, таких як count, distinct count, так і безпосередній перегляд декількох записів та даних що там містяться.

– Тестування інтеграції бази даних та дашборду. Після створення підключення до бази даних, було виконано налаштування зв’язків між таблицями та перевірено відповідність даних, що відображаються у Tableau з тими, що містяться у сховищі даних. На рисунку 4.2 зображено процес налаштування зв’язків між таблицями у відповідності зі схемою даних.

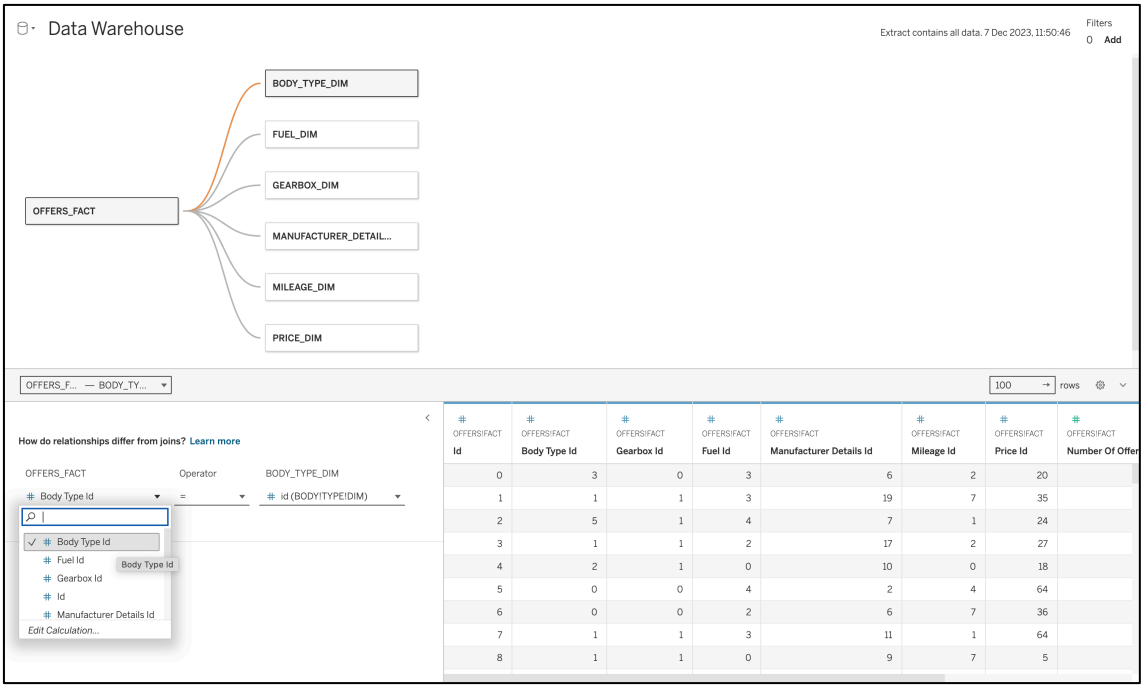


Рисунок 4.2 – Налаштування зв’язків між таблицями.

4.6 Висновки до розділу

У цьому розділі докладно розглядено різні типи тестування програмного забезпечення, які використовуються протягом життєвого циклу розробки програмного забезпечення. Описані методи тестування охоплюють як функціональні, так і нефункціональні аспекти, пропонуючи комплексну перевірку продуктивності програмного забезпечення.

Для оцінки розробленої функціональності систематично застосовувалися різні підходи до тестування, що забезпечило надійний контроль якості на всіх етапах розробки.

5 ЕКОНОМІЧНИЙ РОЗДІЛ

5.1 Технологічний аудит розробленого веб сервісу для аналізу ринку вживаних авто України

У сучасному світі все більше людей приймають рішення про покупку автомобіля не тільки на основі емоцій і зовнішнього вигляду автомобіля, але й на основі аналітичних обґрунтованих даних, таких як технічні характеристики, склад і функціональні показники. Придбання автомобіля – це значні фінансові інвестиції з тривалим періодом використання, що вимагає глибокого аналізу як окремих моделей та їх характеристик, так і ринку в цілому. Пропозиції ринку вживаних автомобілів в Україні можна побачити на таких онлайн-платформах, як AUTO.RIA, RST та ін. Однак відсутність інструментів для всебічного аналізу ринку та візуалізації показників обмежує можливості покупців робити усвідомлений вибір і точно оцінювати загальний стан ринку. Тому створення спеціального сервісу стає актуальним завданням для забезпечення доступу до об’єктивної інформації та покращення процесу прийняття рішень щодо покупки автомобіля.

Тому перед виконаною магістерською кваліфікаційною роботою було поставлено мету: розробити аналітичний веб сервіс, що запропонує потенційним покупцям можливість ознайомитись з реаліями ринку вживаних автомобілів в Україні та зробити обґрунтований вибір при купівлі.

Для цього було: досліджено ринок онлайн авто-майданчиків України; детально вивчено принципи розробки аналітичних сервісів; розроблено конвеєр ELT та побудовано дашборд; проведено тестування проекту протягом всього циклу розробки.

В результаті було розроблено сервіс, що надає можливість користувач ознайомитися з агрегованими метриками ринку у інтерактивний спосіб за допомогою зручного дашборду.

Для встановлення комерційного потенціалу розробленого веб сервісу було запрошено 3-х експертів. Встановлення комерційного потенціалу розробленого веб сервісу було здійснено за критеріями, наведеними в таблиці 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання технічного рівня та комерційного потенціалу будь-якої розробки і їх бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Практична здійсненність					
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Запрошені експерти оцінили розроблений аналітичний веб сервіс таким чином (див. таблицю 5.2):

Таблиця 5.2 – Результати технологічного аудиту розробленого мобільного додатка (за шкалою оцінювання 0-1-2-3-4)

Критерії	Експерт		
	1	2	3
	Бали, що їх виставили експерти:		
1	4	3	3
2	3	3	4
3	4	3	4
4	4	4	4
5	3	3	3

Продовження таблиці 5.2

Критерії	Експерт		
	1	2	3
	Бали, що їх виставили експерти:		
6	4	3	4
7	4	4	3
8	3	3	3
9	4	4	3
10	4	4	3
11	3	3	4
12	2	4	3
Сума балів	$СБ_1 = 42$	$СБ_2 = 41$	$СБ_3 = 41$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{42 + 41 + 41}{3} = \frac{124}{3} = 41,3$		

Встановлення комерційного потенціалу розробленого веб сервісу буде здійснюватися на основі рекомендацій, наведених в таблиці 5.3.

Таблиця 5.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 41,3 балів, то це свідчить, що розроблений нами мобільний додаток та програмний інтерфейс має рівень комерційного потенціалу, який вважається «високим».

Це пояснюється тим, що така функціональність ще не прдставлена на ринку, та потенційно буде цікавою для широкої аудиторії.

5.2 Розрахунок витрат на розробку аналітичного сервісу

При виконанні роботи були зроблені певні витрати.

Зокрема:

А) Основна заробітна плата $З_о$ розробників, яка визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t \text{ грн}, \quad (5.1)$$

де М – місячний посадовий оклад розробника, грн; приймемо, що

М = (6700…23000) грн/місяць;

$Т_р$ – число робочих днів в місяці; приймемо $Т_р$ = 20 день;

t – число днів роботи розробників (6 годин роботи на день).

Зроблені розрахунки зведемо до таблиці 5.4:

Таблиця 5.4 – Основна заробітна плата розробників

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи	20000	1000	20 годин	≈ 3333

Продовження таблиці 5.4

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн
2. Магістрант-студент-виконавець	6700	335	75 днів	≈ 25125
3. Консультант з економічної частини	18000	900	1,5 години	≈ 225
Загалом				$З_о$ = 28 683 грн

Б) Додаткова заробітна плата $З_д$ розробників розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о \qquad (5.2)$$

Приймемо, що $\alpha = 0{,}11$. Тоді отримаємо:

$$З_д = 0{,}11 \times 28683 = 3155{,}13 \approx 3156 \text{ грн.}$$

В) Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \qquad (5.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. $\beta = 22\%$. Тоді:

$$НЗН_{зп} = (28683 + 3156) \times 0{,}22 = 7004{,}36 \approx 7005 \text{ грн.}$$

Г) Амортизація основних засобів А, які використовувались під час виконання цієї роботи:

$$A = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ грн}, \qquad (5.4)$$

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань. Для випадку можна прийняти, що $Н_а$ = (2,5...25)%;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 5.5.

Таблиця 5.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання тощо	44500	25	2,1 (при 85% використанні	≈ 1946
2. Приміщення університету, кафедри	21700	3,5	2,25 при 90% використанні	≈ 133
Всього				2079

Д) Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ грн.} \quad (5.5)$$

де $Н_i$ – витрати матеріалу *і*-го найменування, кг; $Ц_i$ – вартість матеріалу *і*-го най-менування; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); $В_i$ – маса відходів матеріалу *і*-го найменування; $Ц_в$ – ціна відходів матеріалу *і*-го найменування; n – кількість видів матеріалів.

Е) Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ грн} \quad (5.6)$$

де $Н_i$ – кількість комплектуючих *і*-го виду, шт.; $Ц_i$ – ціна комплектуючих *і*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих.

Під час виконання роботи загальні витрати на матеріали та комплектуючі склали приблизно 1000 грн.

Ж) Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} \quad (5.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2023 р. В ≈ 4,5 грн/кВт;

П – установлена потужність обладнання, кВт; П = 1,0 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 380 годин;

$К_п$ – коефіцієнт використання потужності; $К_п < 1 = 0,9$.

$К_д$ – коефіцієнт корисної дії, $К_д = 0,8$.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,5 \cdot 1,0 \cdot 380 \cdot 0,9}{0,8} = 1923,75 \approx 1924 \text{ грн.}$$

И) Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о \quad (5.8)$$

Для випадку отримаємо:

$$В_{інш} = 1,2 \times 28683 = 34419,60 \approx 34420 \text{ грн.}$$

К) Сума всіх попередніх статей витрат складає витрати на виконання роботи безпосередньо розробником-магістрантом – В.

$$В = 28683 + 3155 + 7005 + 2079 + 1000 + 1924 + 34420 = 78266 \text{ грн.}$$

Л) Загальні витрати на розроблення мобільного додатка та програмного інтерфейсу $В_{заг}$ становлять:

$$В_{заг} = \frac{В}{\beta} \quad (5.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи. Можна прийняти, що, $\beta \approx 0,90$, оскільки робота практично завершена.

Тоді:

$$В_{заг} = \frac{78\,266}{0,9} = 86\,962,22 \text{ грн або приблизно 87 тис. грн.}$$

Тобто прогнозовані загальні витрати на розробку аналітичного сервісу становлять приблизно 87 тисяч гривень.

5.3 Розрахунок економічного ефекту від можливої комерціалізації розробки

Економічний ефект від впровадження та можливої комерціалізації розробленого аналітичного сервісу пояснюється його новизною, так як аналогічні рішення ще поки не представлені на ринку.

Розробка може бути реалізована за принципом річної підписки на послуги сервісу.

Аналіз ринку також показав, що потенційна кількість користувачів може сягати декількох тисяч. Для розрахунків приймемо 5 користувачів, що відразу захочуть спробувати такий сервіс. Разом з тим, проведений аналіз показав, що кількість зацікавлених користувачів буде зростати, тобто можна очікувати зростання попиту на розробку принаймні протягом 3-х років після її впровадження.

Тобто, якщо розробка буде впроваджена з 1 січня 2024 року, то її результати будуть виявлятися протягом 2024-го, 2025-го та 2026-го років.

Прогноз зростання попиту на розробку складає по роках:

а) 2024 р. – приблизно +1000 користувачів;

б) 2025 р. – +1500 користувачів до базового року;

в) 2026 р. – +600 користувачів до базового року (оскільки можуть з’явитися ще кращі розробки, що створені компаніями AutoRia або RST на базі власних продуктів).

Можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від виведення розробки на ринок, становитиме:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}) \qquad (5.10)$$

де $\Delta Ц_o$ – покращення основного якісного показника від впровадження результатів розробки у цьому році. Для випадку це є ціна річної підписки на сервіс $\Delta Ц_o$ = 500 грн.

N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 5 шт.;

ΔN – покращення основного кількісного показника від впровадження результатів розробки.

Таке покращення становитиме по роках, відповідно: у 2024 році – +1000 користувачів, у 2025 році +1500 користувачів, та у 2026 році +600 користувачів;

$Ц_o$ – основний якісний показник (тобто ціна), який визначає обсяг діяльності у році після впровадження результатів розробки, грн; $Ц_o$= 500 грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для випадку n = 3;

λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ = (0,2...0,5); візьмемо ρ = 0,5;

υ – ставка податку на прибуток. У 2023-25 роках υ = 18%.

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора протягом першого року від можливого впровадження розробки (2024 р.) становитиме:

$$\Delta П_1 = [500 \cdot 5 + 500 \cdot 1000] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 171\,681 \text{ грн.}$$

Можливе зростання чистого прибутку $\Delta П_2$ для потенційного інвестора від можливого впровадження розробки протягом другого (2025) року складе:

$$\Delta П_2 = [500 \cdot 5 + 500 \cdot 1500] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 257\,094 \text{ грн.}$$

Можливе зростання чистого прибутку $\Delta П_3$ для потенційного інвестора від можливого впровадження розробки протягом третього (2026) року складе:

$$\Delta П_3 = [500 \cdot 5 + 500 \cdot 600] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) \approx 103\,351 \text{ грн.}$$

Приведена вартість зростання всіх чистих прибутків від можливого впровадження розробки становитиме:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t} \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки. Для випадку т = 3 роки;

τ – ставка дисконтування. Приймемо τ = 0,10 (10%);

t – період часу від моменту початку розроблення мобільного додатка до моменту отримання можливих чистих прибутків потенційним інвестором.

Тоді приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації розробки, складе:

$$ПП = \frac{171681}{(1+0,1)^2} + \frac{257094}{(1+0,1)^3} + \frac{103\,351}{(1+0,1)^4} \approx 141886 + 193159 + 70591 \approx 406 \text{ тис. грн.}$$

Теперішня вартість інвестицій PV (вартість стартапу), що повинні бути вкладені для реалізації розробки: PV = (1,0…5) × $В_{заг}$.

Для випадку PV = (1,0…5) × 87 = 1,46 × 87 = 127 тис. грн.

Абсолютний ефект від можливих вкладених інвестицій $Е_{абс}$.

$$Е_{абс} = ПП – PV \quad (5.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для інвестора від можливого впровадження розробки, грн;

PV – теперішня вартість інвестицій PV = 127 тис. грн.

Абсолютний ефект від можливого впровадження розробки складе:

$$Е_{абс} = 406 – 127 = 279 \text{ тис. грн.}$$

Оскільки $Е_{абс} > 0$, то комерціалізація розробки може бути доцільною.

Далі розрахуємо внутрішню дохідність $Е_в$ вкладених інвестицій:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \quad (5.13)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 279 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 127 тис. грн;

$Т_ж$ – життєвий цикл розробки, роки.

$Т_ж$ = 4 років (2023-й, 2024-й, 2025-й, 2026-й роки)

Для випадку отримаємо:

$$Е_в = \sqrt[4]{1 + \frac{279}{127}} - 1 = \sqrt[4]{1 + 2,2} - 1 = \sqrt[4]{3,2} - 1 = 1,337 - 1 = 0,337 = 33,7\%$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією розробки.

Мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f \quad (5.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022-2023 роках в Україні d = (0,10...0,12);

f – показник, що характеризує ризикованість вкладень;

f = (0,1...0,50). Приймемо f = 0,22.

Для випадку отримаємо:

$$\tau_{мін} = 0,1 + 0,22 = 0,32 \text{ або } \tau_{мін} = 32\%.$$

Оскільки величина $E_в = 33,7\% > \tau_{мін} = 32\%$, то потенційний інвестор у принципі може бути зацікавлений у фінансуванні та комерціалізації розробки.

Далі розраховуємо термін окупності коштів, вкладених у можливу комерціалізацію розробленого нами мобільного та програмного інтерфейсу.

Термін окупності $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в} \quad (5.15)$$

Для випадку термін окупності $T_{ок}$ коштів становитиме:

$$T_{ок} = \frac{1}{0,337} = 2,97 \text{ років} > 3 \text{ років},$$

що свідчить про потенційну доцільність комерціалізації розробленого нами мобільного додатка та програмного інтерфейсу.

Далі проведено моделювання залежності величини внутрішньої дохідності вкладених потенційних інвестицій від рівня інфляції в країні.

Якщо рівень інфляції в країні зросте до 20%, то:

$$ПП = \frac{171681}{(1+0,2)^2} + \frac{257094}{(1+0,2)^3} + \frac{103\,351}{(1+0,2)^4} \approx 119223 + 148782 + 49929 \approx 318 \text{ тис. грн.}$$

Тоді абсолютний ефект від можливого впровадження розробки за три роки складе:

$$Е_{абс} = 318 – 127 = 191 \text{ тис. грн.}$$

Внутрішня дохідність $Е_в$ вкладених інвестицій становитиме:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (5.16)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 191 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 127 тис. грн.

Для випадку отримаємо:

$$E_в = \sqrt[4]{1 + \frac{191}{127}} - 1 = \sqrt[4]{1 + 1,5} - 1 = \sqrt[4]{3,2} - 1 = 1,257 - 1 = 0,257 = 25,7\%$$

Оскільки величина $Е_в$ = 25,7% < $\tau_{мін}$ = 32%, то потенційний інвестор може бути НЕ зацікавлений у фінансуванні та комерціалізації розробки.

Зроблені розрахунки у вигляді графіків наведено на рисунку 5.1.

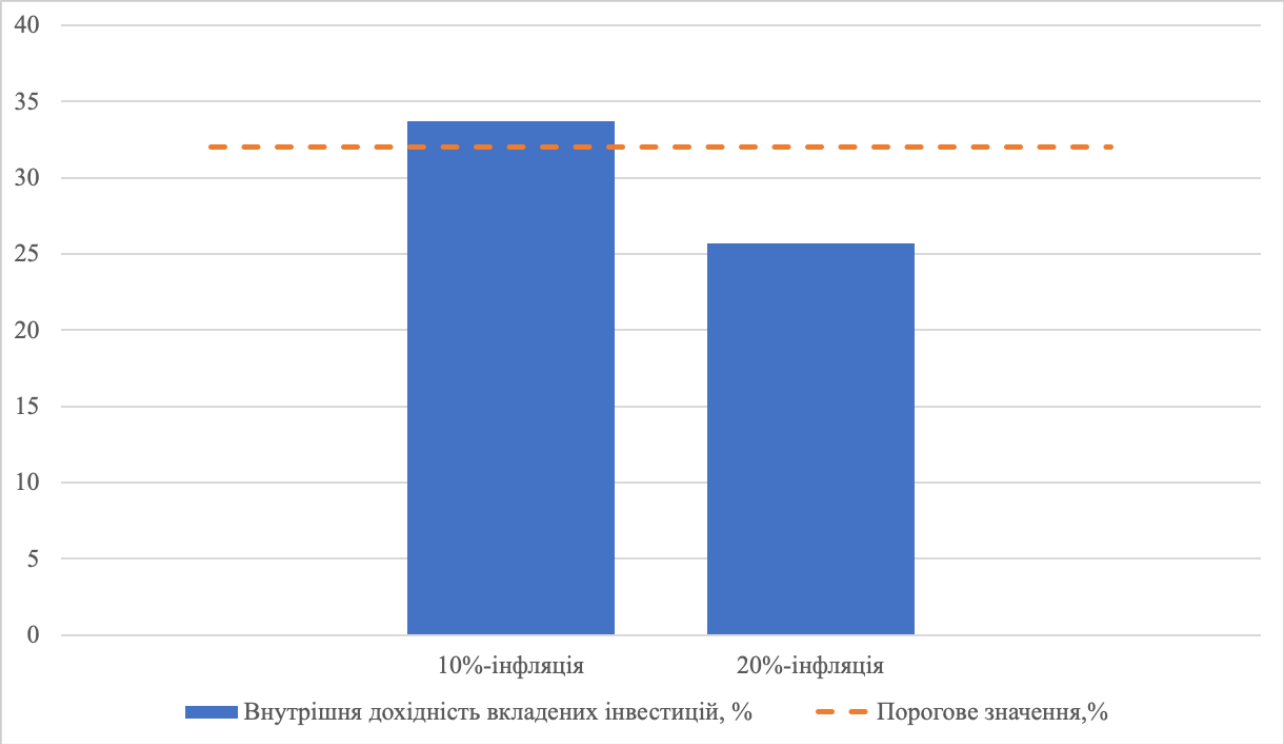


Рисунок 5.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій від рівня інфляції в країні.

Аналіз діаграм на рисунку 5.1 показує, що при рівні інфляції в 10% величина внутрішньої дохідності інвестицій становить $E_в$ = 33,7%, що більше порогового значення $\tau_{мін}$ = 32% і тому комерціалізація розробки може бути доцільною. При рівні інфляції в 20% величина внутрішньої дохідності інвестицій, вкладених в комерціалізацію розробки, становить всього $E_в$ = 25,7%, що менше порогового значення $\tau_{мін}$ = 32%, і тому комерціалізація розробки потенційним інвестором може бути проблематичною.

Остаточне рішення з цього питання потребує проведення додаткових розрахунків (можливо – зниження рівня ризикованості вкладень тощо).

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю:

Таблиця 5.6 – Результати виконаної економічної частини МКР.

Показники	Задані у ТЗ	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку	Не більше 100 тис. грн	87 тис. грн.	Досягнуто
2. Абсолютний ефект від впровадження розробки, тисяч грн	Не менше 250 тисяч грн.	279 тисяч грн (при 10%-інфляції)	Виконано
3. Внутрішня дохідність інвестицій, %	не менше 30%	33,7%	Досягнуто
4. Термін окупності інвестицій, роки	до 3-ти років	2,97 років	Виконано

Таким чином, основні техніко-економічні показники розробленого аналітичного веб сервісу, що може бути використаний при прийнятті рішень щодо купівлі автомобіля або при аналізі ринку в цілому, були виконані.

ВИСНОВКИ

В даній магістерській кваліфікаційній роботі було здійснено аналіз існуючих онлайн-майданчиків з продажу автомобілів в Україні. За кількістю відвідувань веб-сторінки в інтернеті, сервіс AutoRia став лідером серед автомайданчиків, маючи 82% переглядів від загальної кількості переглядів подібних сервісів. Було проаналізовано функціонал, що надається користувачам та виявлено, що AutoRia та інші автомайданчики не надають користувачам аналітичні інструменти для аналізу ринку вживаних авто.

Було розглянуто методи побудови сучасної аналітичної платформи. Такі платформи складаються з джерела даних, конвеєрів ETL (або ELT), сховища даних, а також аналітичних застосунків, що використовують оброблені дані. Проаналізовано типи джерел даних, включаючи реляційні бази даних, файли, API та черги повідомлень, підкреслюючи відмінності між системами OLTP і OLAP. Також, було розглянуто конвеєри ETL і ELT, та продемонстровано відмінність між ними. Детально описано три ключові концепції у сфері зберігання оброблених даних — Data Warehouse, Data Lake і Data Lakehouse — з акцентом на їхні унікальні характеристики.

В процесі розробки було розглянуто варіанти взаємодії із веб-сервісом AutoRia, визначаючи API та веб-скрейпінг як два методи збору даних. Використання готового API над веб-скрейпінгом надає явні переваги, підвищуючи ефективність і надійність збору даних.

На основі вхідного набору даних було створено структуру сховища даних за шаблоном Star. Центральною таблицею фактів стала таблиця OFFERS_FACT, що оточена вимірними таблицями, які містять додаткову інформацію про факт, та можуть легко бути з’єднані завдяки зовнішнім ключам. Було побудовано конвеєр даних за схемою ELT, так як такий підхід дозволив спростити реалізацію та повністю задовольняє вимогам. Було детально описано кожний етап конвеєру та надано відповідні діаграми, що показують рух даних у зрозумілій та простій

формі. Кінцевим аналітичним застосунком стала платформа інтерактивних дашбордів Tableau.

Для оцінки розробленої функціональності систематично застосовувалися різні підходи до тестування, що забезпечило надійний контроль якості на всіх етапах розробки.

В результаті розрахунку економічного ефекту від можливої комерціалізації даної розробки було обраховано наступні економічні показники: витрати на розроблення сервісу для аналізу ринку вживаних авто України - 87 тис. грн; Абсолютний економічний ефект від впровадження розробки - 279 тис. грн; Внутрішню дохідність потенційних інвестицій - 33,7 %; Термін окупності потенційних інвестицій - 2,97 років.

У результаті виконання даної роботи було розроблено аналітичний веб сервіс, що надає можливість користувачам ознайомитися з агрегованими метриками ринку у інтерактивний спосіб за допомогою зручного дашборду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Горбань Д.А. Особливості проектування веб-сервісу для аналізу ринку авто України. Всеукраїнська науково-практична інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи». Вінниця : ВНТУ, 2023.

2. Український авторинок: що впливало у 2022 та чого чекати у 2023? Прогноз експертів. Інститут досліджень авторинку. URL: https://eauto.org.ua/news/231-ukrajinskiy-avtorinok-shcho-vplivalo-u-2022-ta-chogo-chekati-u-2023-prognoz-ekspertiv (дата звернення: 01.11.2023).

3. Які вживані авто купували українці у 2022 році: найпопулярніші марки та моделі. ФОКУС. URL: https://focus.ua/uk/auto/543590-kakie-bu-avto-pokupali-ukraincy-v-2022-godu-samye-populyarnye-marki-i-modeli (дата звернення: 01.11.2023).

4. Скільки автодилерам вдалося реалізовати нових авто у 2022 році. AutoConsulting. URL: https://autoconsulting.ua/article.php?sid=52925 (дата звернення: 01.11.2023).

5. Найпопулярніші нові автомобілі, які українці купували у 2022 році. Finance.ua. URL: https://finance.ua/ua/insurance/avtomobili-jaki-kupuvaly-ukrainci-v-2022 (дата звернення: 01.11.2023).

6. Автосайти України. URL: https://signs.fm/uk/auto/ukrainian-sites/ (дата звернення: 01.11.2023).

7. Офіційний сайт AutoRIA. URL: https://auto.ria.com/uk/ (дата звернення: 01.11.2023).

8. Офіційний сайт RST. URL: https://rst.ua/ukr/ (дата звернення: 01.11.2023).

9. Офіційний сайт Cars. URL: https://cars.ua/ (дата звернення: 01.11.2023).

10. Офіційний сайт Avtobazar. URL: https://avtobazar.ua/ (дата звернення: 01.11.2023).

11. Офіційний сайт similarweb. URL: https://www.similarweb.com/corp/pro/ (дата звернення: 01.11.2023).

12. Joe Reis, Matt Housley. Fundamentals of Data Engineering: Plan and Build Robust Data Systems. O'Reilly Media, Inc., 2022. 447 p.

13. Martin Kleppmann. Designing Data-Intensive Applications. O'Reilly Media, Inc., 2017. 559 p.

14. What is a data pipeline? IBM. URL: https://www.ibm.com/topics/data-pipeline (дата звернення: 05.11.2023).

15. Understanding Data Pipelines: Architecture and Workflow. Medium. URL: https://medium.com/@halfcapsule/understanding-data-pipelines-architecture-and-workflow-4f7ab1e864e6 (дата звернення: 05.11.2023).

16. Adrian Booth, Jeff Hart, Stuart Sim. Building a great data platform. McKinsey&Company. URL: https://www.mckinsey.com/~/media/McKinsey/Industries/Electric%20Power%20and%20Natural%20Gas/Our%20Insights/Building%20a%20great%20data%20platform/Building-a-great-data-platform-final.pdf (дата звернення: 05.11.2023).

17. Star and snowflake schemas. IBM. URL: https://www.ibm.com/docs/en/db2/9.7?topic=schemas-star-snowflake (дата звернення: 05.11.2023).

18. Paul Crickard. Data Engineering with Python: Work with massive datasets to design data models and automate data pipelines using Python. Packt Publishing, 2020.

19. James Densmore. Data Pipelines Pocket Reference: Moving and Processing Data for Analytics. O'Reilly Media, Inc., 2021. URL: https://www.oreilly.com /library/view/data-pipelines-pocket/9781492087823/ (дата звернення: 05.11.2023).

20. Mark Lutz. Learning Python. 5th ed. O'Reilly Media, 2013. 1643 p.

21. Python and REST APIs: Interacting With Web Services. URL: https://realpython.com/api-integration-in-python/ (дата звернення: 13.11.2023).

22. Regina O. Obe, Leo S. Hsu. PostgreSQL: Up and Running, 3rd Edition. O'Reilly Media, Inc., 2017.

23. John Viescas, Douglas Steele, Ben Clothier. Effective SQL: 61 Specific Ways to Write Better SQL. Addison-Wesley Professional, 2016.

24. Mauricio Aniche. Effective Software Testing: A developer's guide. Manning, 2020.

25. David Sale. Testing Python: Applying Unit Testing, TDD, BDD and Acceptance Testing. Wiley, 2014.

26. Офіційний сайт Tableau. URL : https://www.tableau.com/ (дата звернення: 17.12.2023).

27. Sumit Gupta, Sylvester Pinto, Shweta Sankhe-Savale. The Tableau Workshop: A practical guide to the art of data visualization with Tableau. Packt Publishing, 2022.

28. Barry W. Boehm. Software Engineering Economics. Prentice Hall. 800 p.

29. В.О. Козловський, О.Й. Лесько, В.В.Кавецький. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. Вінниця : ВНТУ, 2021. 42 с.

30. . Бісікало О.В, Іванов Ю.Ю., Маслій Р.В. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальностей: 126 – «Інформаційні системи та технології», 151 – «Автоматизація та комп’ютерно-інтегровані технології», 174 – «Автоматизація, комп’ютерно-інтегровані технології та робототехніка». Вінниця : ВНТУ, 2023. 63 с.

ДОДАТКИ

Додаток А (обов'язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
_____д.т.н., проф. Олег БІСІКАЛО
«__» _____2023 року

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
РОЗРОБКА WEB СЕРВІСУ ДЛЯ АНАЛІЗУ РИНКУ ВЖИВАНИХ АВТО УКРАЇНИ
08-31.МКР.006.02.003 ТЗ

Керівник: к.т.н., проф.
_____ Євген ПАЛАМАРЧУК
«__» _____ 2023 р.
Розробив студент гр. 1ІСТ-22м
_____ Денис ГОРБАНЬ
«__» _____ 2023 р.

Вінниця 2023

1. Назва та галузь застосування.
Розробка WEB сервісу для аналізу ринку вживаних авто України. Інформаційні системи та технології.

2. Підстава для проведення робіт.
Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2023р., та індивідуальне завдання на МКР, затверджене протоколом №__ засідання кафедри АІІТ від «__» _____ 2023р.

3. Мета та призначення роботи.
Метою роботи є розробка сервісу, який запропонує потенційним покупцям можливість ознайомитись з реаліями ринку вживаних автомобілів в Україні та зробити обґрунтований вибір при купівлі.

4. Джерела розробки:

1) Martin Kleppmann. Designing Data-Intensive Applications. O'Reilly Media, Inc., 2017. 559 p.
2) Joe Reis, Matt Housley. Fundamentals of Data Engineering: Plan and Build Robust Data Systems. O'Reilly Media, Inc., 2022. 447 p.
3) Python and REST APIs: Interacting With Web Services. URL: https://realpython.com/api-integration-in-python/ (дата звернення: 13.11.2023).

5. Показники призначення
5.1 Мінімальні вимоги до техніки:
- ОС: Ubuntu 20.04;
- Процесор: Intel Core i5 3 GHz або вище;
- Об’єм оперативної пам’яті: 8 Gb або вище;
- SSD диск: 120 Gb SATA2 або вище.
5.2 Середовище розробки та запуску PyCharm.
5.3 СУБД PostgreSQL.

Вихідні дані для проведення робіт:
Стек технологій обробки даних, джерело даних, застосунок для побудови діаграм, ноутбук, середовище розробки PyCharm.
Методи дослідження:
В роботі використані аналітичні методи оброблення інформації та методи математичної статистики.
Результати роботи програми:
Конвеєр даних успішно збирає та доставляє дані до сховища, де вони обробляються та завантажуються до фінальних таблиць. Інтерактивний

дашборд використовує дані з фінальних таблиць та візуалізує їх.

6. Економічні показники
До економічних показників входять:
- витрати на розробку – до 100 тис. грн.
- мінімальна дохідність – не менше 250 тис. грн.
- внутрішня дохідність – не менше 30%
- термін окупності – не більше 3 років

7. Стадії розробки:

а) Дослідження предметної галузі	20.09.23 – 02.10.23
б) Обгрунтування вибору технологічного стеку	03.10.23 – 09.10.23
в) Розробка програмного забезпечення	10.10.23 – 14.11.23
г) Тестування розробленого програмного забезпечення	15.11.23 – 30.11.23
д) Підготовка економічної частини	01.12.23 – 12.12.23
е) Оформлення пояснювальної записки	13.12.23 – 16.12.23

8. Порядок контролю та приймання

Рубіжний контроль провести до «___» _____ 2023 р.
Попередній захист МКР провести «___» _____ 2023 р.
Захист МКР провести до «___» _____ 2023 р.

Розробив студент групи 1ІСТ-22м _____ Денис ГОРБАНЬ

Додаток Б (обов'язковий)

Ілюстративна частина

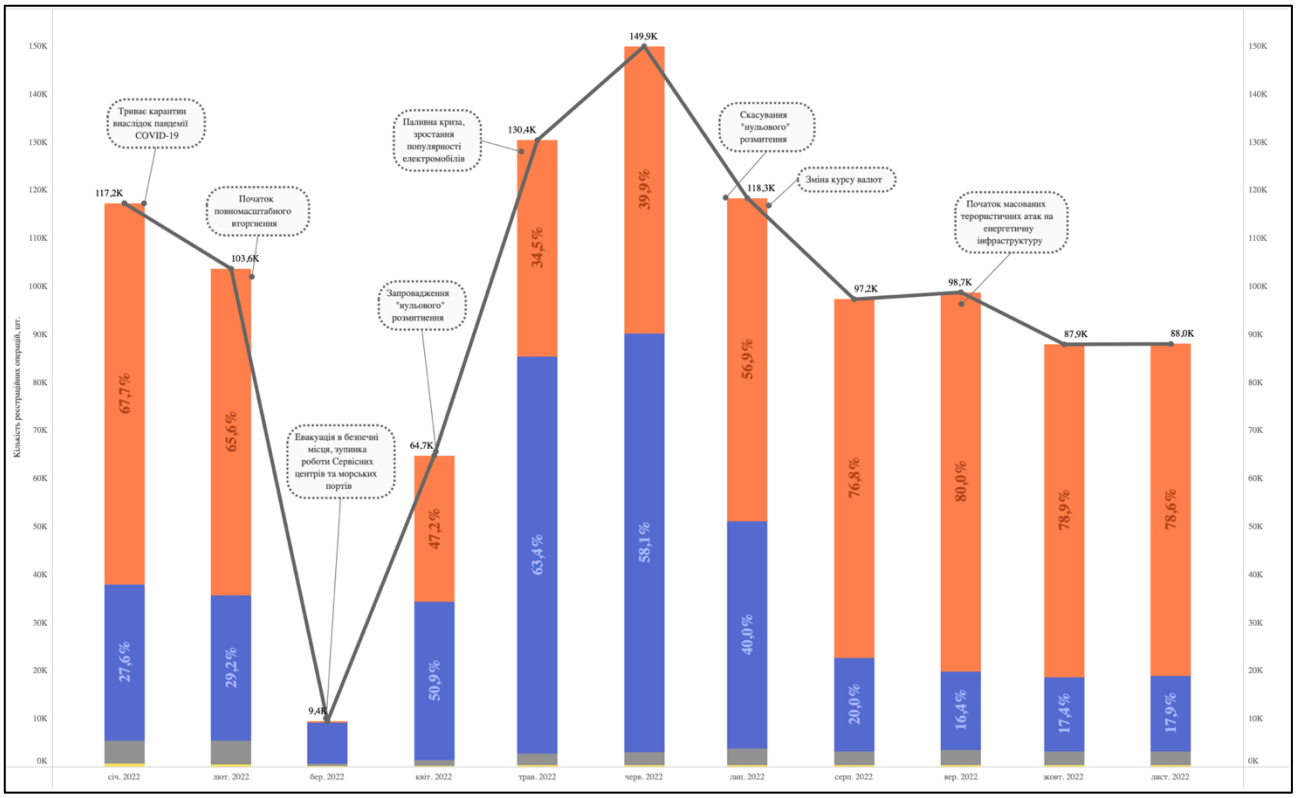


Рисунок Б.1 – Графік динаміки операцій на ринку авто у 2022 році.

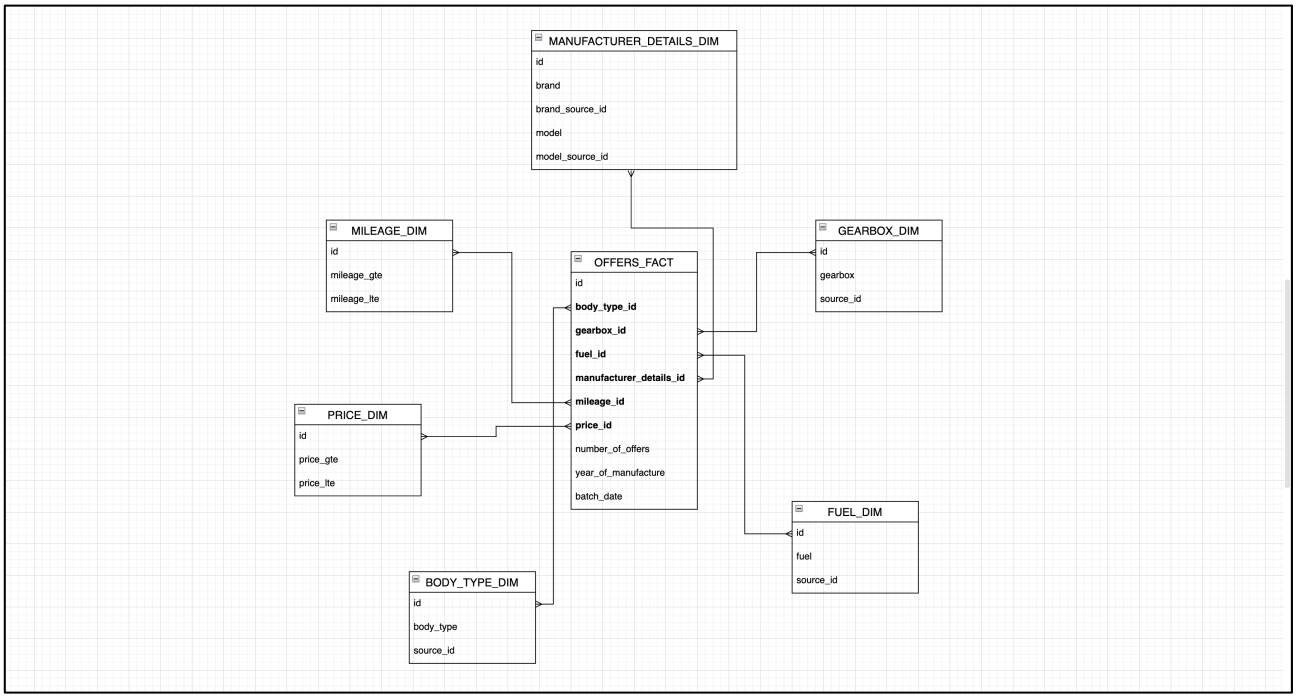


Рисунок Б.2 – Схема сховища даних.

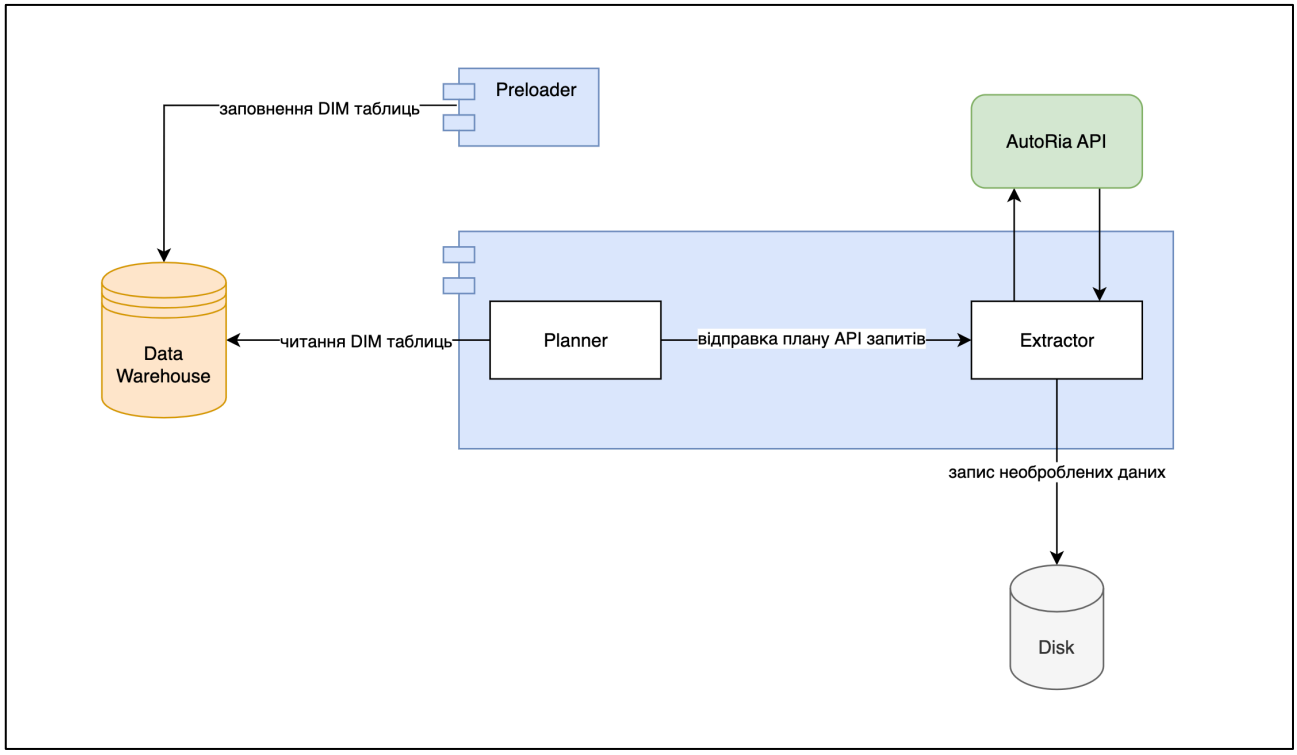


Рисунок Б.3 – Діаграма етапу збору даних.

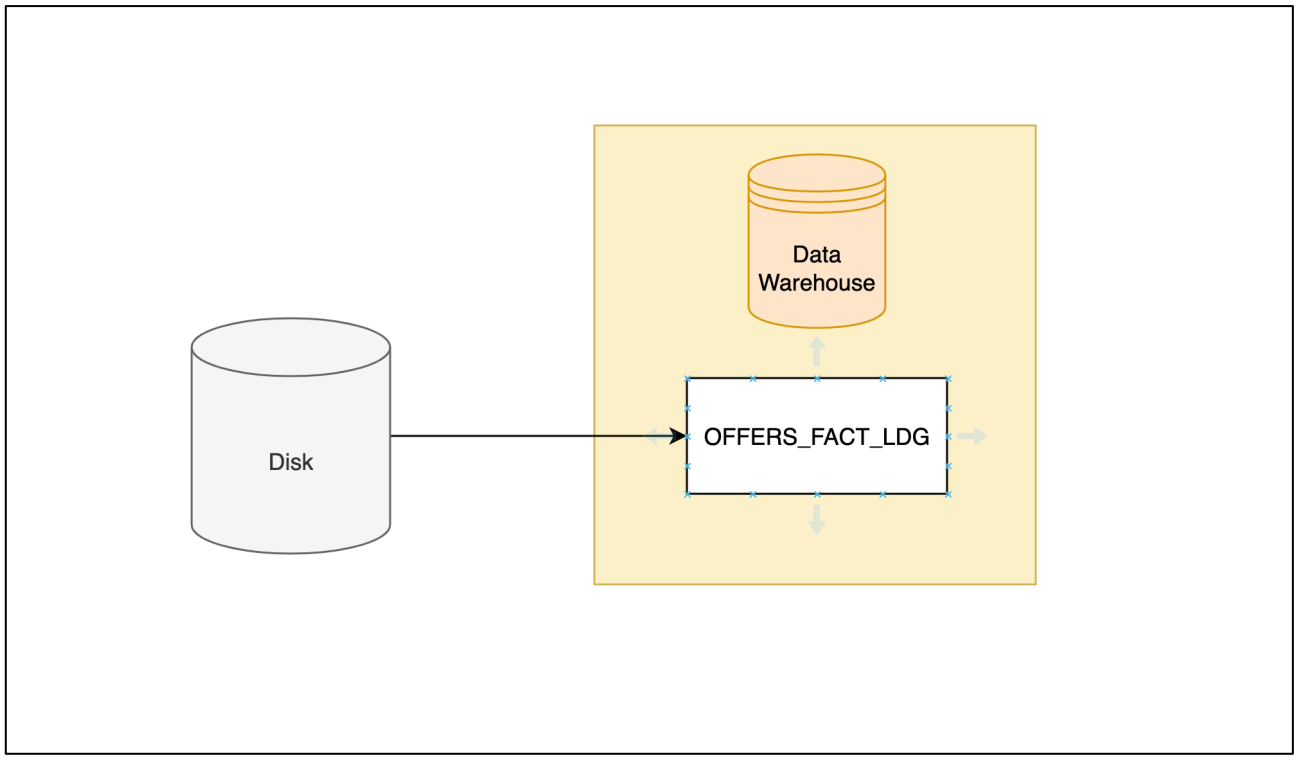


Рисунок Б.4 – Діаграма етапу доставки даних.

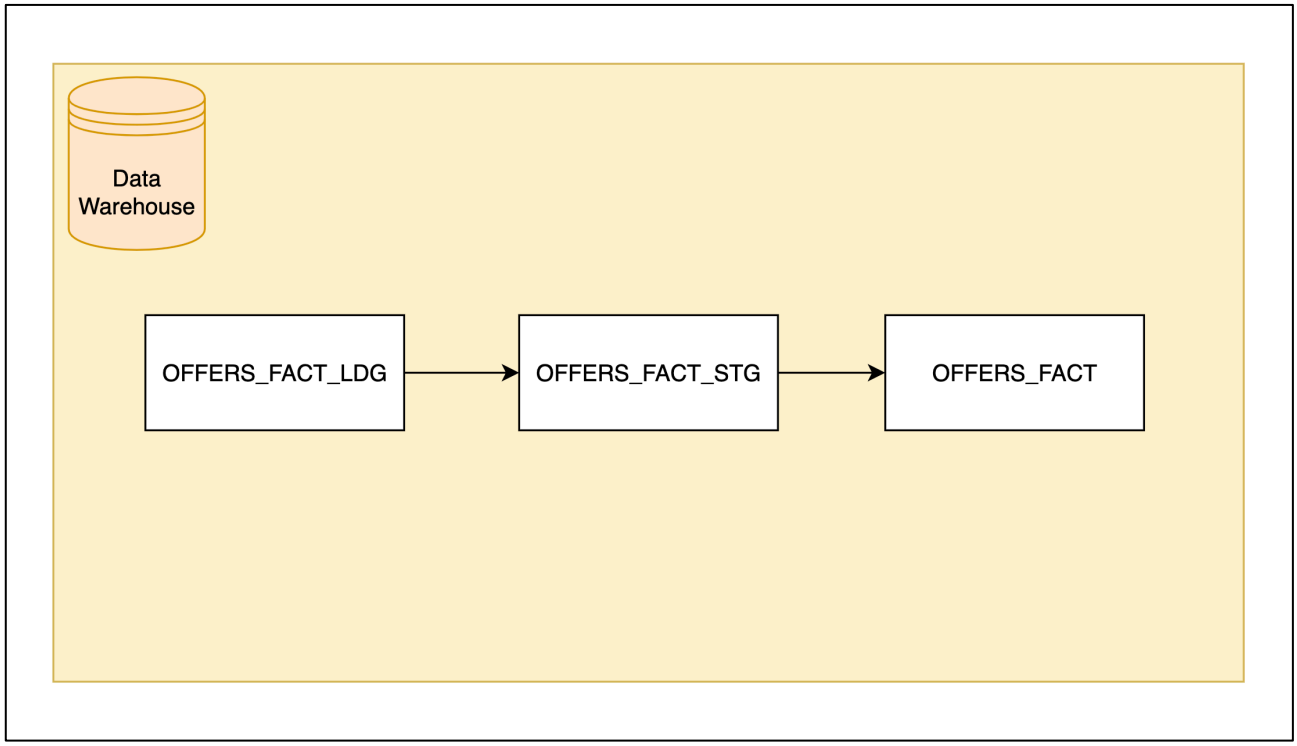


Рисунок Б.5 – Діаграма етапу трансформації даних.

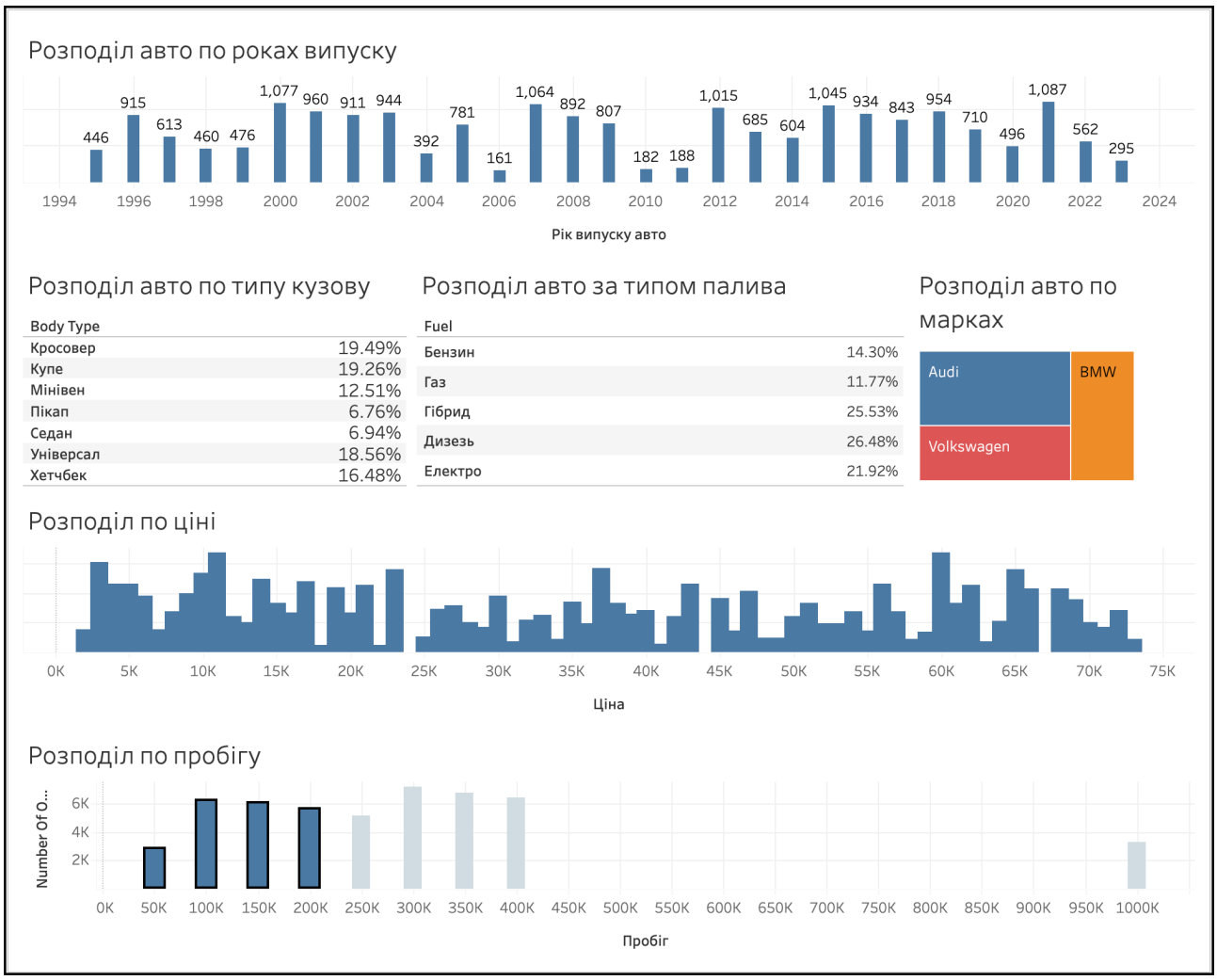


Рисунок Б.6 – Вигляд інтерактивного дашборду.

Додаток В (обов'язковий)

Лістинг програмного забезпечення

```
from dataclasses import dataclass
from typing import List

@dataclass
class ResponseItem:
    name: str
    value: int



@dataclass
class WrappedResponse:
    response_items: List[ResponseItem]

    def __post_init__(self):
        self.marks = [ResponseItem(**item) for item in self.response_items]

class OffersSummaryDto:
    def __init__(self, body_type_source_id, gearbox_source_id,
                 fuel_source_id, brand_source_id, model_source_id,
                 mileage_from, mileage_to, price_from, price_to,
                 number_of_offers, year_of_manufacture):
        self.body_type_source_id = body_type_source_id
        self.gearbox_source_id = gearbox_source_id
        self.fuel_source_id = fuel_source_id
        self.brand_source_id = brand_source_id
        self.model_source_id = model_source_id
```

```
        self.mileage_from = mileage_from
        self.mileage_to = mileage_to
        self.price_from = price_from
        self.price_to = price_to
        self.number_of_offers = number_of_offers
        self.year_of_manufacture = year_of_manufacture

import requests
import json

from src.wrapped_response import WrappedResponse



class MarksAPIExtractor:
    URL = 'https://developers.ria.com/auto/categories/1/marks'

    def __init__(self, *, endpoint, api_key):
        self.endpoint = endpoint
        self.api_key = api_key

    # Response example:
    # [
    #     {"name": "Acura", "value": 98},
    #     {"name": "Adler", "value": 2396},
    #     {"name": "Autobianchi", "value": 102}
    #     ...
    # ]
    def __get_response(self):
        params = {'API_KEY': self.api_key}
```

```
        response_payload = requests.get(url=MarksAPIExtractor.URL,
params=params).text
        return response_payload

    def __parse_response(self, response):
        wrapped_response = f'{{"response_items": {response} }}'
        wrapped_response_json = json.load(wrapped_response)
        response_items = WrappedResponse(**wrapped_response_json).response_items

        marks = []
        for name, source_id in response_items:
            marks.append((name, source_id))

        return marks

    def get_marks(self):
        response = self.__get_response()
        marks = self.__parse_response(response)
        return marks

import json

import requests

from src.wrapped_response import WrappedResponse


class BodyTypeAPIExtractor:
    URL = 'https://developers.ria.com/auto/categories/1/bodystyles'
```

```
    def __init__(self, *, endpoint, api_key):
        self.endpoint = endpoint
        self.api_key = api_key

    # Response example:
    # [
    #     {"name":"Седан","value":3},
    #     {"name":"Позашляховик / Кросовер","value":5},
    #     {"name":"Пікап","value":9}
    #     ...
    # ]
    def __get_response(self):
        params = {'API_KEY': self.api_key}
        response_payload = requests.get(url=BodyTypeAPIExtractor.URL,
params=params).text
        return response_payload

    def __parse_response(self, response):
        wrapped_response = f'{{"response_items": {response} }}'
        wrapped_response_json = json.load(wrapped_response)
        response_items = WrappedResponse(**wrapped_response_json).response_items

        body_types = []
        for name, source_id in response_items:
            body_types.append((name, source_id))

        return body_types
```

```
    def get_body_types(self):
        response = self.__get_response()
        body_types = self.__parse_response(response)
        return body_types

CREATE TABLE MANUFACTURER_DETAILS_DIM (
  id INTEGER PRIMARY KEY,
  brand TEXT NOT NULL,
  brand_source_id INTEGER NOT NULL,
  model TEXT NOT NULL,
  model_source_id INTEGER NOT NULL
);

CREATE TABLE GEARBOX_DIM (
  id INTEGER PRIMARY KEY,
  gearbox TEXT NOT NULL,
  source_id INTEGER NOT NULL
);

CREATE TABLE FUEL_DIM (
  id INTEGER PRIMARY KEY,
  fuel TEXT NOT NULL,
  source_id INTEGER NOT NULL
);

CREATE TABLE BODY_TYPE_DIM (
  id INTEGER PRIMARY KEY,
  body_type TEXT NOT NULL,
  source_id INTEGER NOT NULL
```

```
);

CREATE TABLE PRICE_DIM (
  id INTEGER PRIMARY KEY,
  price_gte INTEGER NOT NULL,
  price_lte INTEGER NOT NULL
);

CREATE TABLE MILEAGE_DIM (
  id INTEGER PRIMARY KEY,
  mileage_gte INTEGER NOT NULL,
  mileage_lte INTEGER NOT NULL
);

CREATE TABLE OFFERS_FACT_LDG (
  body_type_source_id INTEGER NOT NULL,
  gearbox_source_id INTEGER NOT NULL,
  fuel_source_id INTEGER NOT NULL,
  brand_source_id INTEGER NOT NULL,
  model_source_id INTEGER NOT NULL,
  mileage_from INTEGER NOT NULL,
  mileage_to INTEGER NOT NULL,
  price_from INTEGER NOT NULL,
  price_to INTEGER NOT NULL,
  number_of_offers INTEGER NOT NULL,
  year_of_manufacture INTEGER NOT NULL
);

CREATE TABLE OFFERS_FACT_STG (
```

```
  body_type_id INTEGER REFERENCES BODY_TYPE_DIM (id),
  gearbox_id INTEGER REFERENCES GEARBOX_DIM (id),
  fuel_id INTEGER REFERENCES FUEL_DIM (id),
  manufacturer_details_id INTEGER REFERENCES
MANUFACTURER_DETAILS_DIM (id),
  mileage_id INTEGER REFERENCES MILEAGE_DIM (id),
  price_id INTEGER REFERENCES PRICE_DIM (id),
  number_of_offers INTEGER NOT NULL,
  year_of_manufacture INTEGER NOT NULL,
  batch_date DATETIME NOT NULL
);

CREATE TABLE OFFERS_FACT (
  id INTEGER PRIMARY KEY,
  body_type_id INTEGER REFERENCES BODY_TYPE_DIM (id),
  gearbox_id INTEGER REFERENCES GEARBOX_DIM (id),
  fuel_id INTEGER REFERENCES FUEL_DIM (id),
  manufacturer_details_id INTEGER REFERENCES
MANUFACTURER_DETAILS_DIM (id),
  mileage_id INTEGER REFERENCES MILEAGE_DIM (id),
  price_id INTEGER REFERENCES PRICE_DIM (id),
  number_of_offers INTEGER NOT NULL,
  year_of_manufacture INTEGER NOT NULL,
  batch_date DATETIME NOT NULL
);

DELETE FROM OFFERS_FACT_STG ALL;

INSERT INTO OFFERS_FACT_STG
```

```
SELECT
  (SELECT BODY_TYPE_DIM.id
  FROM OFFERS_FACT_LDG
  JOIN BODY_TYPE_DIM ON BODY_TYPE_DIM.source_id =
OFFERS_FACT_LDG.body_type_source_id
  ) AS body_type_id,
  (SELECT GEARBOX_DIM.id
  FROM OFFERS_FACT_LDG
  JOIN GEARBOX_DIM ON GEARBOX_DIM.source_id =
OFFERS_FACT_LDG.gearbox_source_id
  ) AS gearbox_id,
  (SELECT FUEL_DIM.id
  FROM OFFERS_FACT_LDG
  JOIN FUEL_DIM ON FUEL_DIM.source_id =
OFFERS_FACT_LDG.fuel_source_id
  ) AS fuel_id,
  (SELECT MANUFACTURER_DETAILS_DIM.id
  FROM OFFERS_FACT_LDG,
  JOIN MANUFACTURER_DETAILS_DIM ON
(OFFERS_FACT_LDG.brand_source_id =
MANUFACTURER_DETAILS_DIM.brand_source_id AND
OFFERS_FACT_LDG.model_source_id =
MANUFACTURER_DETAILS_DIM.model_source_id)
  ) AS manufacturer_details_id,
  (SELECT MILEAGE_DIM.id
  FROM OFFERS_FACT_LDG,
  JOIN MILEAGE_DIM ON (OFFERS_FACT_LDG.mileage_from =
MILEAGE_DIM.mileage_gte AND OFFERS_FACT_LDG.mileage_to =
MILEAGE_DIM.mileage_lte)
```

```
  ) AS mileage_id,
  (SELECT PRICE_DIM.id
  FROM OFFERS_FACT_LDG,
  JOIN PRICE_DIM ON (OFFERS_FACT_LDG.price_from =
PRICE_DIM.price_gte AND OFFERS_FACT_LDG.price_to =
PRICE_DIM.price_lte)
  ) AS price_id,
  number_of_offers,
  year_of_manufacture,
  CURRENT_TIMESTAMP AS batch_date
FROM OFFERS_FACT_LDG;

INSERT INTO OFFERS_FACT
SELECT *
FROM OFFERS_FACT_STG;
```

Додаток Г (обов'язковий)

Протокол перевірки магістерської кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: «Розробка WEB сервісу для аналізу ринку вживаних авто України»

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра АІІТ

Показники звіту подібності Unicheck

Оригінальність 91,4 % Схожість 8,6 %

Аналіз звіту подібності (відмітити потрібне)

• Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

○ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

○ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Роман МАСЛІЙ
(підпис)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Денис ГОРБАНЬ
(підпис)

Керівник роботи _____ Євген ПАЛАМАРЧУК
(підпис)