

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ
БІЗНЕС-АНАЛІТИКИ ПРОДАЖІВ ТОВАРІВ
DEVELOPMENT OF SOFTWARE FOR BUSINESS
ANALYTICS OF PRODUCT SALES

Виконав(ла): студент(ка) 4 курсу, групи КНТ-122
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

МАНДИК О.О.

(ПРІЗВИЩЕ та ініціали)

Керівник ОЛІЙНИК А.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 121 Інженерія програмного забезпечення

(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення

(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.

Сергій СУББОТІН

“ ” 2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

МАНДИКА Олександра Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розроблення програмного забезпечення для бізнес-аналітики продажів товарів. Development of Software for Business Analytics of Product Sales

керівник проєкту (роботи) д.т.н., професор, ОЛІЙНИК Андрій Олексійович,

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

затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування архітектури та структури програмного забезпечення. 3. Розробка програмного забезпечення. 4. Експлуатація, аналіз результатів та тестування системи.

5. Перелік графічного матеріалу (з з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ОЛІЙНИК А.О., професор		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 20 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3–4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування.	6–7 тиждень	Додатки
8	Захист роботи.	8 тиждень	

Студент(ка)

(підпис)

Олександр МАНДИК

(Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Андрій ОЛІЙНИК

(Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
114 с., 6 табл., 38 рис., 3 дод., 20 джерел.

PYTHON, POSTGRESQL, ВЕБЗАСТОСУНОК, БІЗНЕС-АНАЛІТИКА,
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ДАНІ ПРОДАЖІВ.

Об'єкт дослідження – процес розроблення програмного забезпечення для бізнес-аналітики продажів товарів.

Предмет дослідження – методи та засоби оброблення, аналізу та візуалізації даних продажів у програмних системах.

Мета роботи – зменшення часових витрат та трудомісткості процесу підготовки аналітичної звітності підприємства шляхом розроблення програмного забезпечення для автоматизації збору, валідації та візуалізації даних продажів товарів.

Матеріали, методи та технічні засоби: мова програмування Python, вебфреймворк, система керування базами даних PostgreSQL.

Результати. Розроблено архітектурні та проєктні рішення для створення програмного забезпечення бізнес-аналітики продажів. Реалізовано вебзастосунок, що забезпечує імпорт, перевірку, збереження та аналітичне опрацювання даних із формуванням КРІ-дашбордів і звітів. Проведено тестування розробленої програмної системи.

Висновки. Мету роботи досягнуто. Створено програмне забезпечення для бізнес-аналітики продажів із використанням мови програмування Python та СУБД PostgreSQL.

Галузь використання – інформаційні системи аналізу продажів та підтримки прийняття рішень.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 114 pages, 6 tables, 38 figures, 3 appendixes, 20 sources.

PYTHON, POSTGRESQL, WEB APPLICATION, BUSINESS ANALYTICS, SOFTWARE, SALES DATA.

Object of research – the process of developing software for business analytics of product sales.

Subject of research – methods and tools for processing, analysis, and visualization of sales data in software systems.

Purpose of the work – reduction of time costs and labor intensity of the enterprise analytical reporting preparation process through the development of software for automation of collection, validation, and visualization of product sales data.

Materials, methods and technical means: Python programming language, web framework, PostgreSQL database management system.

Results. Architectural and design solutions for creating business analytics software for sales have been developed. A web application has been implemented that provides data import, validation, storage, and analytical processing with the formation of KPI dashboards and reports. Testing of the developed software system has been carried out.

Conclusions. The objective of the work has been achieved. Software for business analytics of sales has been developed using the Python programming language and PostgreSQL DBMS.

Field of application – information systems for sales analysis and decision support.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області	10
1.1 Постановка проблеми аналізу продажів товарів.....	10
1.2 Огляд сучасних систем бізнес-аналітики та аналіз їх можливостей	12
1.3 Вимоги до програмного забезпечення	19
1.4 Висновки за розділом 1	23
2 Проєктування архітектури та структури програмного забезпечення.....	25
2.1 Вибір технологій та інструментів реалізації	25
2.2 Проєктування архітектури програмного забезпечення.....	29
2.3 Проєктування структури бази даних.....	31
2.4 Розробка структурної схеми програмного проєкту	36
2.5 Висновки за розділом 2	38
3 Розробка програмного забезпечення.....	40
3.1 Алгоритм функціонування програмного забезпечення	40
3.2 Опис модулів реалізованої програми.....	42
3.3 Висновки за розділом 3	55
4 Експлуатація, аналіз результатів та тестування системи.....	57
4.1 Призначення та функціональні можливості програми	57
4.2 Умови виконання програмного забезпечення.....	59
4.3 Тестування програмного забезпечення.....	61
4.4 Висновки за розділом 4	74
Висновки	75
Перелік джерел посилання.....	76
Додаток А Технічне завдання.....	78
Додаток Б Фрагмент тексту програми	83
Додаток В Слайди презентації.....	104

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ACID	– Atomicity, Consistency, Isolation, Durability;
CSV	– Comma-Separated Values;
HTTP	– Hyper Text Transfer Protocol;
JSON	– JavaScript Object Notation;
JVM	– Java Virtual Machine;
KPI	– Key Performance Indicators;
MVC	– Model-View-Controller;
SQL	– Structured Query Language;
TCP	– Transmission Control Protocol;
URL	– Uniform Resource Locator;
XLSX	– Excel Spreadsheet File Format;
БД	– база даних;
ПЗ	– програмне забезпечення;
СУБД	– система управління базами даних.

ВСТУП

У сучасних умовах цифровізації економіки підприємства функціонують у середовищі стрімкого зростання обсягів даних, що формуються в процесі господарської діяльності, зокрема у сфері продажів. За оцінками аналітичних досліджень, загальний обсяг даних у світі у 2024 році досяг приблизно 149 зеттабайтів, що відображає безпрецедентний рівень цифровізації бізнес-процесів і підвищує потребу в ефективних інструментах їх оброблення та аналізу [1]. У таких умовах традиційні підходи до оброблення інформації втрачають актуальність, що обумовлює необхідність використання спеціалізованих систем бізнес-аналітики.

Важливість впровадження аналітичних інструментів підтверджується також статистичними даними щодо поширення технологій оброблення даних. Зокрема, станом на 2024–2025 роки близько 50 % організацій у світі вже використовують технології великих даних, а серед великих підприємств цей показник досягає 78 % [2]. Крім того, майже 48 % компаній позиціонують себе як «data-driven», тобто таких, що приймають рішення на основі аналізу даних, причому цей показник демонструє суттєве зростання у порівнянні з попередніми роками [3]. Це свідчить про трансформацію підходів до управління підприємствами, де аналітика стає ключовим фактором конкурентоспроможності.

Поряд із цим, дослідження підкреслюють значний вплив аналітики даних на результати діяльності підприємств. Встановлено, що використання технологій Big Data та аналітичних моделей сприяє підвищенню інноваційності та результативності бізнесу, а також формує основу для прийняття стратегічних управлінських рішень [4]. Також, існує проблема ефективного використання цих технологій, оскільки лише частина проєктів із впровадження аналітики демонструє очікуваний економічний ефект, що обумовлює необхідність розроблення більш структурованих та інтегрованих програмних рішень.

Додатково слід зазначити, що ринок технологій бізнес-аналітики характеризується стабільним зростанням. Зокрема, обсяг світового ринку Big Data оцінювався у понад 300 млрд доларів США у 2023 році з подальшою тенденцією до значного збільшення у наступні роки [5]. Це підтверджує високий попит на програмні засоби, які забезпечують не лише накопичення, але й аналітичну обробку даних, їх візуалізацію та інтерпретацію.

Таким чином, зростання обсягів даних, поширення підходів data-driven управління, а також потреба у підвищенні обґрунтованості управлінських рішень визначають необхідність створення сучасних інформаційних систем бізнес-аналітики. Особливого значення набувають програмні рішення, орієнтовані на аналіз продажів, оскільки саме ця сфера безпосередньо впливає на фінансові результати підприємства та його стратегічний розвиток.

Метою роботи є розроблення програмного забезпечення для автоматизації збору, оброблення та аналізу даних продажів із підтримкою управлінських рішень.

Для досягнення поставленої мети необхідно вирішити такі основні задачі:

- провести аналіз предметної області бізнес-аналітики продажів, визначити проблеми оброблення даних та сформулювати вимоги до програмного забезпечення;
- розробити архітектуру програмного забезпечення та спроектувати структуру бази даних для зберігання і оброблення інформації про продажі;
- реалізувати програмний модуль імпорту, перевірки, оброблення даних і формування аналітичних звітів та KPI-дашбордів;
- виконати тестування програмного забезпечення та проаналізувати результати його функціонування в умовах експлуатації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка проблеми аналізу продажів товарів

Аналіз процесів оброблення даних про продажі дозволяє виявити низку взаємопов'язаних проблем, що ускладнюють отримання достовірної та своєчасної аналітичної інформації. Узагальнення цих проблем доцільно подати у вигляді структурної схеми на рис. 1.1, що відображає їх причинно-наслідкові зв'язки та основні напрямки впливу.



Рисунок 1.1 – Структурна схема проблем аналізу продажів

Основною проблемою є фрагментація даних, що виникає через використання різних джерел інформації, зокрема CSV-файлів та електронних таблиць, що призводить до відсутності єдиного інформаційного середовища та ускладнює консолідацію даних. Важливим фактором також є низька якість вхідних даних, яка проявляється у вигляді помилок, дублікатів та некоректних значень за відсутності процедур їх очищення. Це, у свою чергу, ускладнює аналітичне опрацювання, що включає агрегацію, фільтрацію та порівняння показників, і потребує значних часових витрат при ручній обробці. Додатково виникають труднощі з візуалізацією результатів та повільним формуванням звітів, що обмежує оперативність подання інформації. Сукупність зазначених факторів призводить до затримок у прийнятті управлінських рішень і знижує ефективність використання даних у бізнес-процесах.

Процес аналізу продажів доцільно розглядати як послідовність взаємопов'язаних етапів, що включають імпорт даних із зовнішніх джерел, їх структурну та логічну перевірку, виявлення і усунення дублікатів, збереження у централізованій базі даних, а також подальшу аналітичну обробку із застосуванням операцій агрегації, групування, фільтрації та порівняння. Важливою складовою цього процесу є також візуалізація результатів у вигляді звітів і дашбордів, що забезпечують наочне представлення ключових показників діяльності підприємства.

Вирішення окреслених задач процесу аналізу продажів вимагає визначення вимог до програмного забезпечення, яке повинно забезпечувати автоматизацію зазначених етапів, мінімізувати вплив людського фактора на якість оброблення даних та підвищити оперативність отримання результатів. Особливої уваги потребує забезпечення достовірності даних, що досягається шляхом впровадження механізмів валідації, контролю цілісності та недопущення дублювання записів. Не менш важливим є забезпечення гнучкості аналітичних інструментів, що дозволяють формувати звіти за різними параметрами та адаптувати їх до потреб користувача.

Необхідність розв'язання зазначеної проблеми обумовлена тим, що якість і швидкість оброблення даних про продажі безпосередньо впливають на ефективність управління підприємством. Відсутність автоматизованих інструментів аналітики призводить до затримок у прийнятті рішень, зниження точності прогнозування та втрати потенційних можливостей для оптимізації діяльності. У сучасних умовах конкурентного середовища підприємства потребують інструментів, які забезпечують оперативний доступ до актуальної інформації, дозволяють швидко реагувати на зміни ринку та формувати обґрунтовані управлінські стратегії.

Таким чином, розроблення програмного забезпечення для бізнес-аналітики продажів виступає як необхідна умова підвищення якості інформаційного забезпечення управління, що дозволяє забезпечити інтеграцію даних, автоматизацію їх оброблення та отримання релевантної аналітичної інформації у зручному для користувача вигляді.

1.2 Огляд сучасних систем бізнес-аналітики та аналіз їх можливостей

Сучасні програмні рішення у сфері бізнес-аналітики охоплюють широкий спектр функціональних можливостей, зокрема інтеграцію даних із різних джерел, їх трансформацію, побудову багатовимірних моделей, виконання аналітичних запитів та формування звітності у зручному для користувача вигляді. Важливою складовою таких систем є засоби візуалізації, що дозволяють представляти результати аналізу у вигляді графіків, діаграм і дашбордів, забезпечуючи наочність та інтерпретованість отриманої інформації.

Разом із тим, існуючі системи бізнес-аналітики відрізняються за рівнем складності, функціональними можливостями, вимогами до ресурсів та умовами використання. Деякі з них орієнтовані на великі корпоративні середовища та потребують значних витрат на впровадження і супровід, тоді як інші є більш легковаговими та доступними, але можуть мати обмежений функціонал. Це

зумовлює необхідність проведення їх аналізу з метою визначення переваг, недоліків та доцільності використання у конкретних умовах.

Система Tableau є сучасним програмним засобом бізнес-аналітики, призначеним для інтерактивного аналізу даних, побудови візуалізацій та формування аналітичних дашбордів (рис. 1.2). Основне призначення системи полягає у перетворенні великих обсягів різномірних даних у наочну та інтерпретовану форму, що дозволяє користувачам швидко виявляти закономірності, тенденції та відхилення у бізнес-процесах, зокрема у сфері продажів товарів [6].

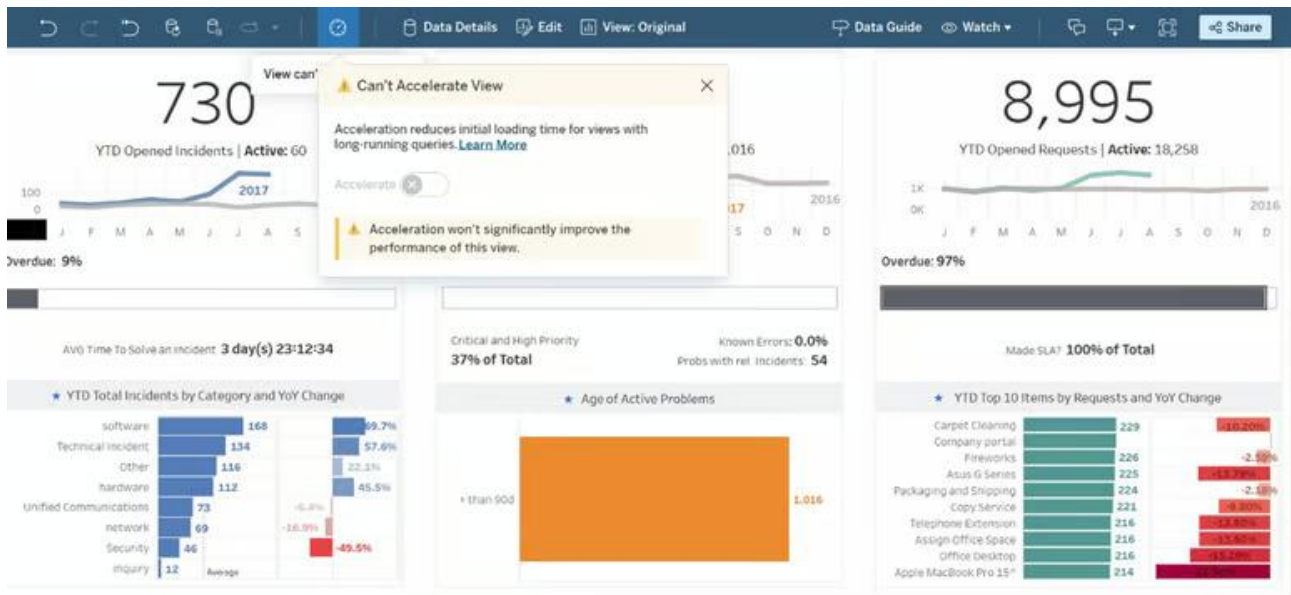


Рисунок 1.2 – Приклад інтерфейсу системи «Tableau» [7]

Архітектурно система побудована за клієнт-серверним принципом і включає декілька взаємопов'язаних компонентів. Центральним елементом виступає Tableau Server або Tableau Cloud, який забезпечує зберігання, оброблення та спільний доступ до аналітичного контенту. На стороні клієнта використовується Tableau Desktop, що виконує функції підключення до джерел даних, їх підготовки, трансформації та побудови візуалізацій [8]. Взаємодія з даними може здійснюватися як у режимі прямого підключення до джерел, так і

через створення витягів (extracts), які оптимізують швидкість оброблення. Для кінцевих користувачів передбачено вебінтерфейс або мобільні застосунки, що забезпечують доступ до дашбордів і звітів у режимі реального часу. Така архітектура дозволяє забезпечити масштабованість, гнучкість інтеграції з різними джерелами даних і ефективну організацію аналітичного процесу.

Переваги Tableau:

- високий рівень візуалізації даних із широкими можливостями побудови інтерактивних дашбордів;
- підтримка підключення до великої кількості джерел даних (бази даних, файли, хмарні сервіси);
- інтуїтивно зрозумілий інтерфейс, що дозволяє працювати без глибоких знань програмування;
- масштабованість і можливість організації спільної роботи через серверні рішення.

Недоліки Tableau:

- висока вартість ліцензії, що обмежує використання для невеликих проєктів;
- обмежені можливості для складної попередньої обробки даних без використання сторонніх інструментів;
- високі вимоги до апаратних ресурсів при роботі з великими обсягами даних;
- обмежена гнучкість у кастомізації порівняно з повністю програмованими рішеннями.

Отже, застосунок Tableau є потужним інструментом бізнес-аналітики, який забезпечує ефективну візуалізацію та аналіз даних, однак його використання є найбільш доцільним у середніх та великих організаціях, де виправдані витрати на впровадження і супровід, а також існує потреба у швидкому формуванні аналітичних звітів без глибокої програмної реалізації.

Система Metabase є програмним засобом бізнес-аналітики з відкритим вихідним кодом, призначеним для спрощення доступу до даних, їх аналізу та візуалізації без необхідності глибоких знань у програмуванні або SQL (рис. 1.3). Основне призначення системи полягає у наданні користувачам можливості швидко отримувати відповіді на бізнес-запити шляхом побудови інтерактивних звітів, дашбордів і запитів до баз даних, що особливо актуально для аналізу продажів та оперативного контролю ключових показників [9].

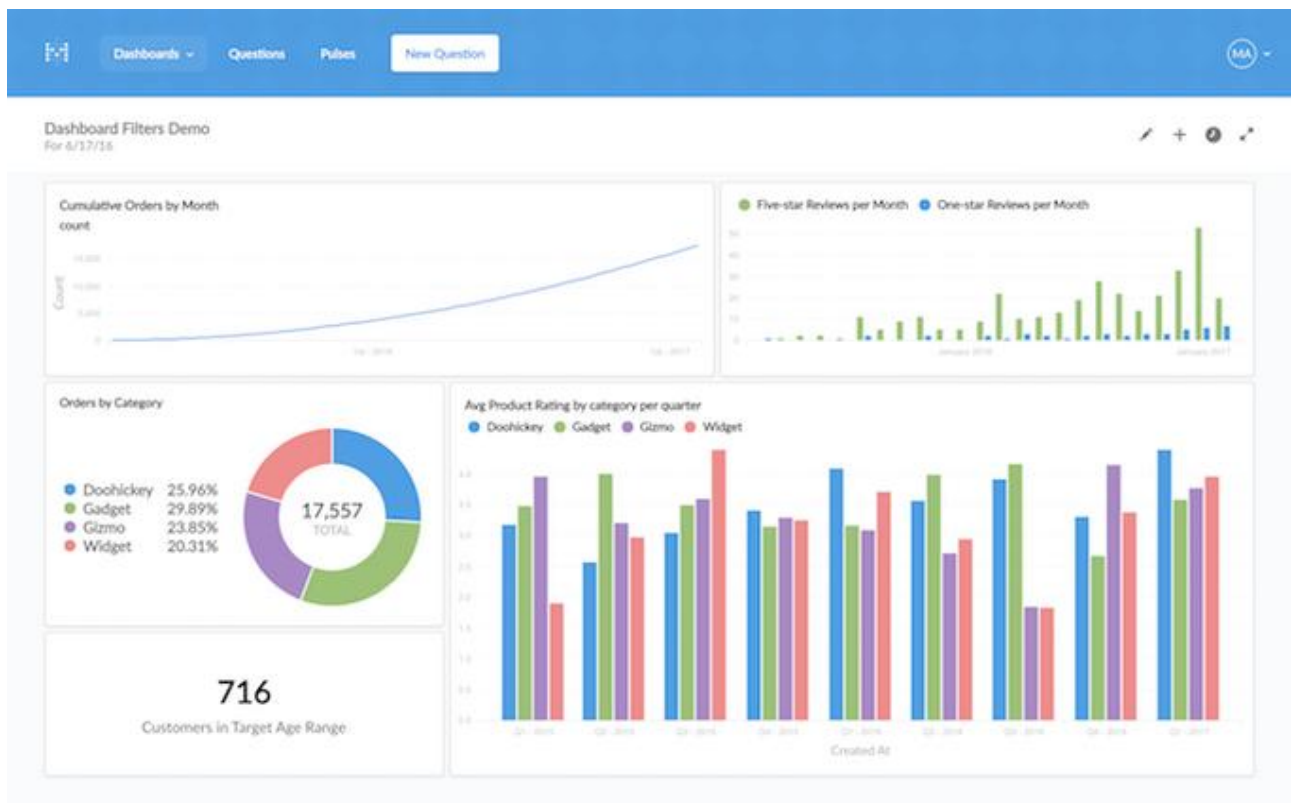


Рисунок 1.3 – Приклад інтерфейсу системи «Metabase» [10]

Архітектура системи побудована за принципом веборієнтованого клієнт-серверного застосунку, де центральним елементом виступає сервер Metabase, що відповідає за оброблення запитів, взаємодію з джерелами даних та формування результатів. Система безпосередньо підключається до різних баз даних, таких як PostgreSQL, MySQL або інші СУБД, виконуючи запити у режимі реального часу без необхідності попереднього копіювання даних [11]. Користувацький доступ

реалізується через вебінтерфейс, який забезпечує побудову запитів за допомогою графічного конструктора або SQL-редактора, а також відображення результатів у вигляді таблиць і візуалізацій. Такий підхід забезпечує простоту розгортання, гнучкість інтеграції та можливість використання системи як у локальному середовищі, так і у хмарній інфраструктурі.

Переваги Metabase:

- відкритий вихідний код і безкоштовність базової версії, що робить систему доступною для навчальних і малих проєктів;
- простота використання завдяки інтуїтивному вебінтерфейсу та можливості роботи без знань SQL;
- швидке підключення до різних баз даних без необхідності складної інтеграції або ETL-процесів;
- легкість розгортання та підтримка роботи як у локальному середовищі, так і в хмарі.

Недоліки Metabase:

- обмежені можливості складної аналітики та трансформації даних порівняно з більш потужними BI-системами;
- менш розвинені інструменти візуалізації у порівнянні з комерційними рішеннями;
- обмежений функціонал у безкоштовній версії для корпоративного використання (керування доступом, масштабування);
- продуктивність може знижуватися при роботі з великими обсягами даних без додаткової оптимізації.

Отже, Metabase є доцільним рішенням для базового рівня бізнес-аналітики, особливо у випадках, коли важлива швидкість впровадження, простота використання та мінімальні витрати, однак для складних аналітичних задач і масштабних корпоративних систем може потребувати доповнення іншими інструментами або переходу до більш функціонально розвинених платформ.

Система Zoho Analytics є сучасним хмарним програмним засобом бізнес-аналітики, призначеним для збору, інтеграції, оброблення та візуалізації даних з метою підтримки управлінських рішень (рис. 1.4). Основне призначення платформи полягає у забезпеченні можливості аналізу даних із різних джерел, зокрема файлів, баз даних і хмарних сервісів, із подальшим формуванням інтерактивних звітів і дашбордів, що дозволяє оцінювати показники діяльності підприємства, включаючи продажі товарів, у реальному або наближеному до реального часу [12].

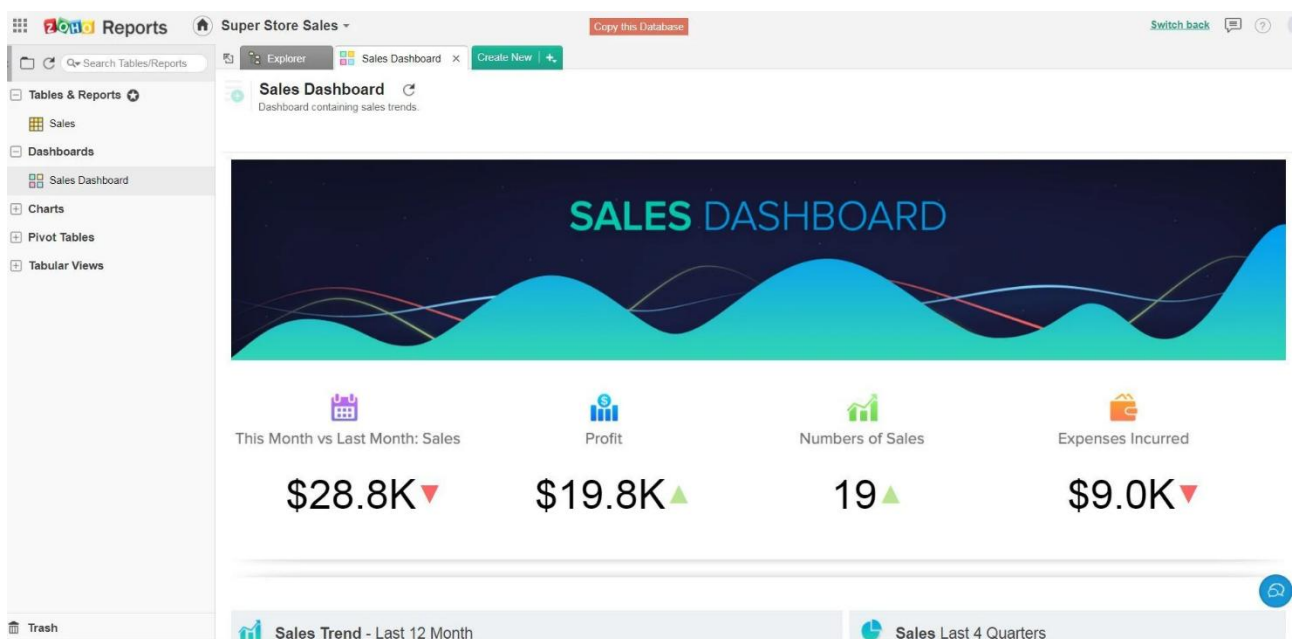


Рисунок 1.4 – Приклад інтерфейсу системи «Zoho Analytics» [13]

Архітектура системи базується на хмарній клієнт-серверній моделі, де всі обчислювальні та зберігальні ресурси розміщуються у хмарній інфраструктурі Zoho. Серверна частина відповідає за збереження даних, їх трансформацію, оброблення аналітичних запитів і керування доступом користувачів. Інтеграційний рівень забезпечує підключення до зовнішніх джерел даних, а також автоматичне оновлення інформації за заданими інтервалами [14]. Користувацький доступ реалізується через вебінтерфейс або мобільні застосунки, що дозволяє створювати звіти за допомогою вбудованих інструментів або SQL-запитів і

відображати результати у вигляді графіків, таблиць і дашбордів. Така архітектура забезпечує високу доступність, масштабованість і зручність використання системи без необхідності розгортання локальної інфраструктури.

Переваги Zoho Analytics:

- хмарна архітектура, що забезпечує доступ до системи без необхідності розгортання локальної інфраструктури;
- широкі можливості інтеграції з різними джерелами даних, включаючи файли, СУБД та хмарні сервіси;
- наявність вбудованих інструментів автоматизації оновлення даних та формування звітів;
- доступний інтерфейс із підтримкою як візуального конструктора, так і SQL-запитів.

Недоліки Zoho Analytics:

- залежність від стабільного інтернет-з'єднання через хмарну модель роботи;
- обмеження у кастомізації та розширенні функціоналу порівняно з open-source рішеннями;
- можливі затримки при обробленні великих обсягів даних у хмарному середовищі;
- повний функціонал доступний лише у платних тарифах.

Отже, Zoho Analytics є зручним і функціональним інструментом для побудови бізнес-аналітики в умовах хмарної інфраструктури, який забезпечує швидке розгортання та інтеграцію з різними джерелами даних, однак його використання найбільш доцільне у випадках, коли допустима залежність від хмарного середовища та обмежень, пов'язаних із ліцензуванням.

Проведений аналіз сучасних систем бізнес-аналітики показує, що більшість із них орієнтовані або на корпоративний сегмент із високою вартістю впровадження, або на універсальні рішення з надлишковим функціоналом, що

ускладнює їх використання у навчальних та малих проєктах. Це обумовлює доцільність розроблення нового програмного забезпечення, яке буде орієнтоване на спрощення процесів роботи з даними про продажі та забезпечення зрозумілого інтерфейсу користувача. Запропоноване рішення має передбачати лише необхідний функціонал для імпорту, перевірки, зберігання та аналізу даних без перевантаження складними інструментами. Важливою перевагою повинна стати легкість впровадження без складних налаштувань та залежності від дорогих ліцензійних продуктів. Таким чином, створення безкоштовного або умовно безкоштовного програмного засобу з базовими аналітичними можливостями є обґрунтованим і доцільним для забезпечення доступності бізнес-аналітики широкому колу користувачів.

1.3 Вимоги до програмного забезпечення

Вимоги до програмного забезпечення визначають функціональні можливості системи, її обмеження та характеристики якості. Для системи бізнес-аналітики продажів такі вимоги мають охоплювати роботу з великими обсягами даних, забезпечення їх коректності, аналітичну обробку та подання результатів у зручному для користувача вигляді. За результатами аналізу предметної області та технічного завдання вимоги доцільно розділити на функціональні та нефункціональні.

Функціональні вимоги визначають перелік можливостей, які має реалізовувати програмне забезпечення для виконання поставлених задач:

- забезпечення автентифікації користувачів та розмежування доступу за ролями (адміністратор, аналітик);
- імпорт даних про продажі з файлів форматів CSV та XLSX;
- перевірка структури та коректності імпортованих даних;
- виявлення та недопущення дублювання записів;
- збереження оброблених даних у базі даних;

- ведення журналу імпорту із фіксацією результатів оброблення;
- аналітична обробка даних із можливістю агрегації, фільтрації та групування;
- формування табличних звітів за різними параметрами;
- побудова графічних звітів і KPI-дашбордів;
- збереження параметрів звітів і дашбордів;
- експорт результатів аналізу у формати CSV або Excel;
- застосування фільтрів за датою, категорією, товаром, клієнтом, регіоном, каналом продажу та способом оплати;
- відображення повідомлень про результати імпорту, помилки та дублікати.

Окрім функціональних можливостей, важливу роль відіграють нефункціональні вимоги, які визначають якісні характеристики системи та умови її використання:

- забезпечення зрозумілого та веборієнтованого інтерфейсу;
- висока надійність роботи системи при повторних імпортах даних;
- забезпечення цілісності та достовірності даних;
- захист даних шляхом хешування паролів та контролю доступу;
- масштабованість системи та можливість подальшого розширення функціоналу;
- забезпечення достатньої продуктивності при обробленні значних обсягів даних;
- простота розгортання та супроводу;
- підтримка роботи з різними джерелами даних;
- забезпечення логування дій користувачів і процесів імпорту;
- стабільність роботи системи в умовах експлуатації.

Сукупність наведених вимог формує основу для подальшого проєктування архітектури програмного забезпечення та визначає напрямки його реалізації.

Для більш детального розуміння функціональних можливостей розроблюваної системи доцільно представити взаємодію користувача з програмним забезпеченням у вигляді діаграми варіантів використання. Такий підхід дозволяє формалізувати основні сценарії роботи та визначити перелік операцій, доступних для кожної ролі. На рис. 1.5 наведено діаграму варіантів використання для ролі адміністратора.

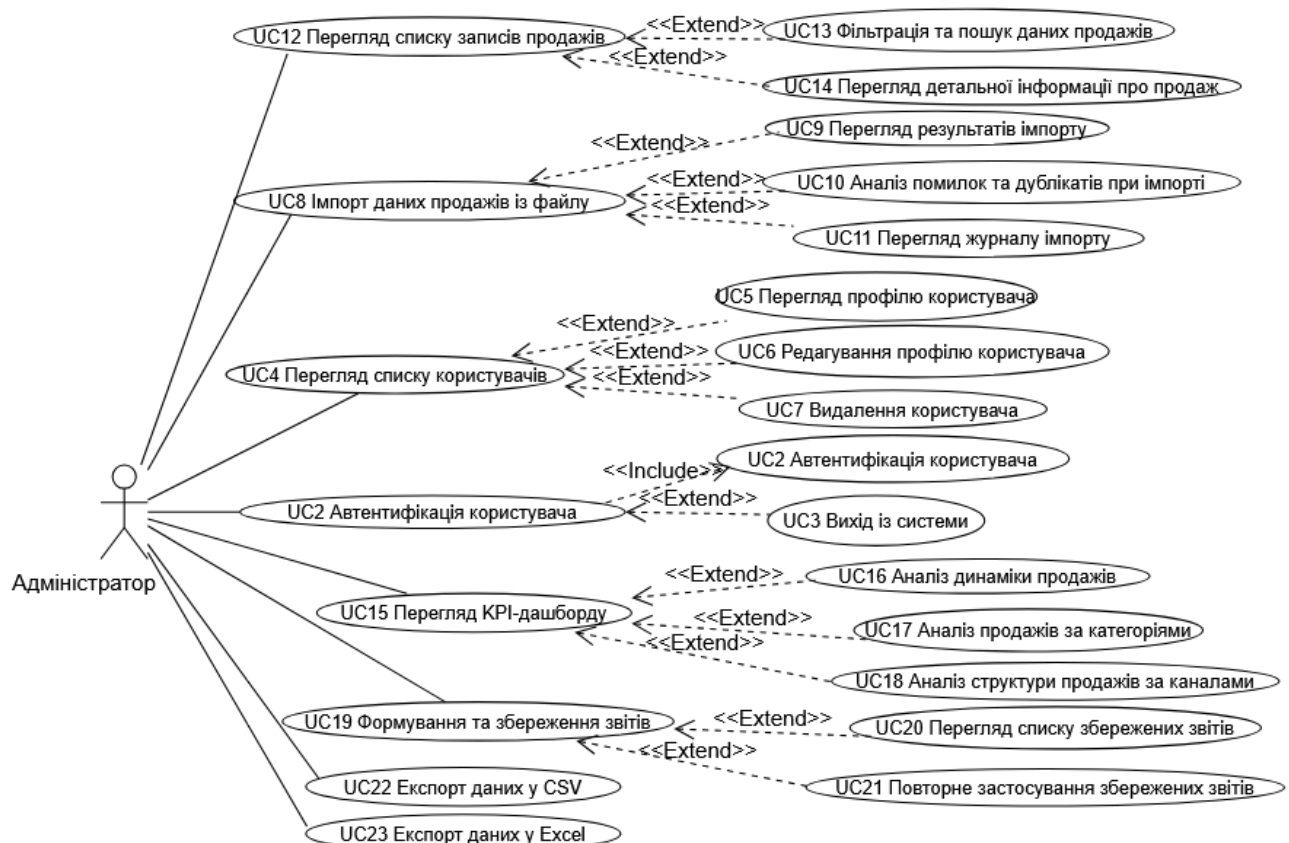


Рисунок 1.5 – Діаграма варіантів використання для ролі «адміністратор»

Адміністратор взаємодіє із системою через базові функції автентифікації та виходу, після чого отримує доступ до розширеного функціоналу управління даними та користувачами. Йому доступні операції перегляду списку користувачів із можливістю переходу до перегляду, редагування та видалення профілів. Значна частина функціоналу пов'язана з імпортом даних продажів, що включає завантаження файлів, перегляд результатів імпорту, аналіз помилок і дублікатів, а

також перегляд журналу імпорту. Адміністратор також має можливість працювати з даними продажів, переглядаючи їх список, здійснюючи фільтрацію, пошук і перегляд детальної інформації. Аналітичний блок представлений функціями перегляду KPI-дашборду та виконання різних видів аналізу, зокрема динаміки продажів, структури за категоріями та каналами. Додатково є можливості формування, збереження та повторного використання звітів, а також експорту даних у формати CSV та Excel, що забезпечує повний цикл роботи адміністратора з аналітичною інформацією.

На відміну від адміністратора, аналітик має обмежений набір функціональних можливостей, орієнтований переважно на аналіз даних без втручання у процеси їх завантаження або управління користувачами (рис. 1.6).

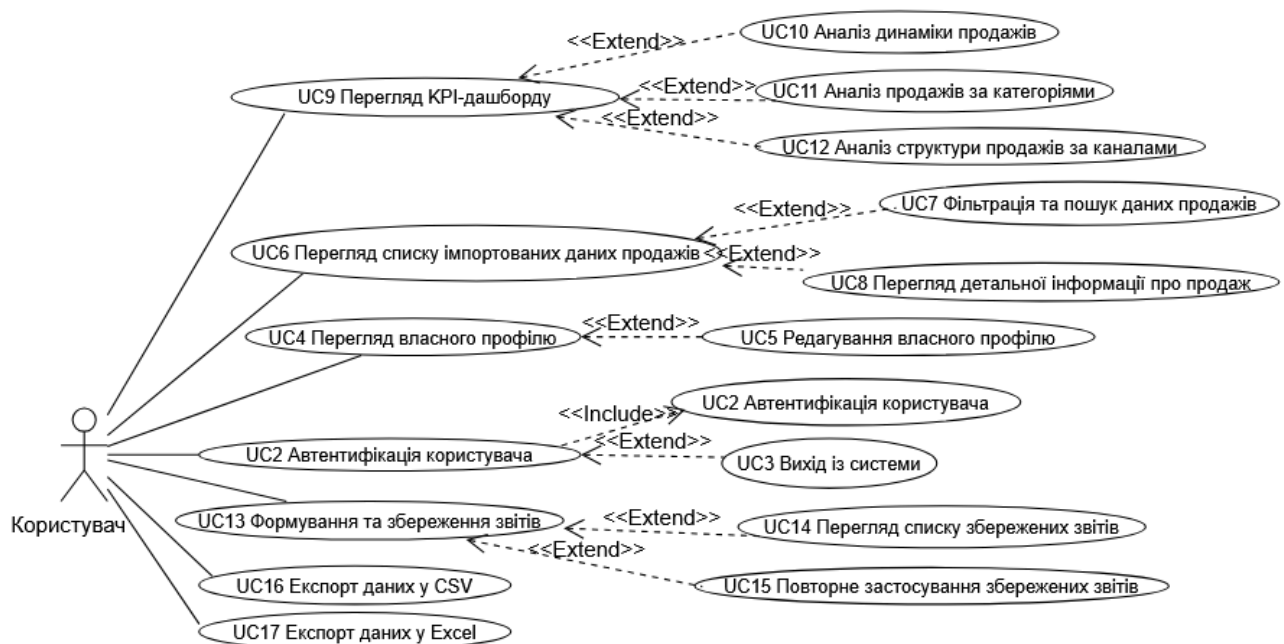


Рисунок 1.6 – Діаграма варіантів використання для ролі «аналітик»

Взаємодія із системою розпочинається з автентифікації, після чого користувач отримує доступ до перегляду KPI-дашборду та аналітичних функцій, що включають аналіз динаміки продажів, структури за категоріями та каналами. Аналітик має можливість переглядати список імпортованих даних продажів,

здійснювати їх фільтрацію та пошук, а також переходити до детального перегляду окремих записів. Окрім цього, передбачено доступ до власного профілю з можливістю його редагування, що забезпечує персоналізацію роботи із системою. Важливою складовою функціоналу є формування, збереження та повторне використання звітів, а також експорт результатів аналізу у формати CSV та Excel.

1.4 Висновки за розділом 1

У рамках даного розділу виконано комплексний аналіз предметної області бізнес-аналітики продажів, що дозволило сформулювати цілісне уявлення про особливості оброблення та використання даних у процесі прийняття управлінських рішень. Розглянуто основні проблеми аналізу продажів, пов'язані з фрагментацією джерел даних, низькою якістю вхідної інформації, складністю аналітичного опрацювання та обмеженими можливостями візуалізації, що узагальнено у вигляді структурної схеми. Це дало змогу конкретизувати задачу оброблення даних як необхідність створення інтегрованого програмного рішення, здатного забезпечити повний цикл аналітичної обробки інформації.

У процесі огляду сучасних систем бізнес-аналітики, зокрема Tableau, Metabase та Zoho Analytics, проаналізовано їх функціональні можливості, архітектурні особливості, а також переваги і обмеження з точки зору застосування для задач аналізу продажів. Отримані результати дозволили встановити, що існуючі рішення або мають надлишковий функціонал і високу вартість, або характеризуються обмеженими можливостями, що обґрунтовує доцільність розроблення власного програмного продукту, орієнтованого на простоту використання та доступність.

На основі проведеного аналізу сформовано вимоги до програмного забезпечення, які охоплюють як функціональні аспекти, пов'язані з імпортом, перевіркою, зберіганням і аналітичною обробкою даних, так і нефункціональні характеристики, що визначають надійність, зручність використання, безпеку та

масштабованість системи. Додатково формалізовано сценарії взаємодії користувачів із системою у вигляді діаграм варіантів використання для ролей адміністратора та аналітика, що дозволило чітко визначити межі доступу та перелік доступних операцій.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір технологій та інструментів реалізації

Враховуючи необхідність оброблення табличних даних, побудови вебінтерфейсу, інтеграції з базою даних та формування аналітичних звітів, доцільно розглянути сучасні мови програмування та відповідні інструменти розробки. Аналіз доступних технологій дозволяє визначити найбільш придатні засоби для створення ефективного, масштабованого та зручного у використанні програмного рішення.

Мова програмування Python є високорівневою інтерпретованою мовою, яка підтримує об'єктно-орієнтований, процедурний та функціональний підходи до розробки [15]. Вона характеризується наявністю широкого набору бібліотек для роботи з даними, зокрема для оброблення табличної інформації та виконання аналітичних обчислень. Python активно використовується для створення вебзастосунків, автоматизації процесів та реалізації систем аналізу даних.

Мова програмування Java є об'єктно-орієнтованою мовою загального призначення, яка орієнтована на розробку масштабованих і платформонезалежних застосунків [16]. Вона використовує віртуальну машину Java Virtual Machine (JVM), що забезпечує виконання програм на різних операційних системах без змін у коді. Java широко застосовується у корпоративних системах, вебзастосунках та сервісах із високими вимогами до надійності та продуктивності.

Мова програмування Ruby є динамічною інтерпретованою мовою, яка підтримує об'єктно-орієнтовану парадигму та орієнтована на швидку розробку програмного забезпечення [17]. Вона має простий та виразний синтаксис, що сприяє зменшенню обсягу коду та підвищенню читабельності програм. Ruby часто використовується для створення вебзастосунків, зокрема із застосуванням фреймворку Ruby on Rails.

Для обґрунтування вибору мови програмування доцільно виконати їх порівняльний аналіз за ключовими характеристиками, що безпосередньо впливають на процес розроблення програмного забезпечення бізнес-аналітики. Узагальнені результати порівняння мов програмування Python, Java та Ruby наведено у таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика мов програмування

Характеристика	Python	Java	Ruby
Тип мови	Інтерпретована	Компілюється у байткод (JVM)	Інтерпретована
Синтаксис	Простий	Строгий	Простий
Швидкість розробки	Висока	Середня	Висока
Продуктивність виконання	Середня	Висока	Нижча за Python
Наявність бібліотек для аналізу	Дуже широка (Pandas, NumPy)	Обмежена	Обмежена
Підтримка веброзробки	Django, Flask	Spring	Ruby on Rails
Підтримка роботи з даними	Розвинена	Середня	Обмежена
Простота навчання	Висока	Середня	Висока
Інтеграція з БД	Широка	Широка	Широка
Вимоги до ресурсів	Низькі	Високі	Середні

Проведений порівняльний аналіз мов програмування показує, що Python найбільш повно відповідає вимогам розроблюваної системи бізнес-аналітики продажів. Це зумовлено наявністю широкого набору спеціалізованих бібліотек для оброблення табличних даних, що дозволяє ефективно реалізувати функції імпорту, перевірки та аналітичного опрацювання інформації. Важливою перевагою є також простий і лаконічний синтаксис, який забезпечує високу

швидкість розробки та зменшує складність супроводу програмного коду. Додатково Python має розвинені засоби для створення вебзастосунків, що дозволяє реалізувати інтерактивний інтерфейс користувача без використання додаткових складних технологій. Сукупність зазначених характеристик робить дану мову доцільною для реалізації системи, орієнтованої на оброблення та аналіз даних про продажі.

Вибір системи керування базами даних є важливим етапом проектування, оскільки саме вона забезпечує зберігання, цілісність та ефективний доступ до даних, що використовуються у процесі аналітики продажів. При виборі СУБД необхідно враховувати вимоги до продуктивності, масштабованості, підтримки складних запитів та інтеграції з обраними технологіями розробки. Для обґрунтованого рішення доцільно розглянути сучасні системи керування базами даних, які широко застосовуються у вебзастосунках та аналітичних системах.

СУБД PostgreSQL є об'єктно-реляційною СУБД з відкритим вихідним кодом, яка підтримує складні запити, транзакції та розширені механізми роботи з даними [18]. Вона забезпечує високий рівень відповідності стандартам SQL та підтримує розширюваність за рахунок користувацьких функцій і типів даних. PostgreSQL широко використовується у системах, що потребують оброблення великих обсягів інформації та виконання аналітичних операцій.

СУБД MySQL є реляційною СУБД, яка орієнтована на швидке виконання запитів і простоту використання у вебзастосунках [19]. Вона підтримує стандартні операції роботи з даними, забезпечує надійність зберігання та має велику кількість інструментів для адміністрування. MySQL часто застосовується у проєктах із середніми вимогами до складності оброблення даних.

СУБД Firebird є реляційною СУБД з відкритим вихідним кодом, яка відзначається компактністю та низькими вимогами до ресурсів [20]. Вона підтримує транзакції, тригери та збережені процедури, що дозволяє реалізовувати бізнес-логіку. Firebird часто використовується у вбудованих або локальних системах, де важлива автономність і простота розгортання.

Для обґрунтування вибору системи керування базами даних доцільно виконати їх порівняльний аналіз за ключовими характеристиками, що визначають ефективність зберігання, оброблення та аналізу даних у розроблюваній системі. Результати порівняння PostgreSQL, MySQL та Firebird наведено у таблиці 2.2.

Таблиця 2.2 – Порівняльна характеристика СУБД

Характеристика	PostgreSQL	MySQL	Firebird
Тип СУБД	Об'єктно-реляційна	Реляційна	Реляційна
Відповідність стандарту SQL	Висока	Середня	Середня
Підтримка складних запитів	Розширена	Обмежена	Обмежена
Робота з великими обсягами даних	Ефективна	Середня	Обмежена
Масштабованість	Висока	Висока	Середня
Підтримка транзакцій	Повна (ACID)	Повна (InnoDB)	Повна
Розширюваність	Висока	Обмежена	Обмежена
Підтримка JSON	Повноцінна (JSON, JSONB)	Обмежена	Відсутня
Аналітичні можливості	Розвинені	Обмежені	Обмежені
Інтеграція з Python	Висока	Висока	Середня
Продуктивність складних операцій	Висока	Середня	Середня
Вимоги до ресурсів	Середні	Низькі	Низькі
Підтримка спільноти	Дуже активна	Дуже активна	Активна

Проведений порівняльний аналіз показує, що PostgreSQL найбільш повно відповідає вимогам розроблюваної системи бізнес-аналітики продажів. Це

обумовлено високим рівнем підтримки складних запитів і розвиненими аналітичними можливостями, що є необхідним для оброблення великих обсягів даних і формування звітів. Додатково важливим є повна відповідність стандартам SQL та підтримка розширених типів даних, зокрема JSON, що забезпечує гнучкість у роботі з різнорідною інформацією. PostgreSQL також характеризується високою масштабованістю та можливістю розширення функціоналу, що дозволяє адаптувати систему до зростання обсягів даних і ускладнення задач. Сукупність цих характеристик робить дану СУБД доцільною для реалізації системи, орієнтованої на аналітичне опрацювання даних про продажі.

2.2 Проектування архітектури програмного забезпечення

На етапі розроблення системи бізнес-аналітики продажів важливим є визначення архітектурного підходу, який забезпечить логічну організацію компонентів системи, їх взаємодію та можливість подальшого розширення функціоналу. Архітектура повинна враховувати вимоги до оброблення даних, розмежування відповідальностей між компонентами, а також забезпечення зручності супроводу та модифікації програмного коду. З огляду на необхідність створення веборієнтованого застосунку доцільним є використання шаблону проектування, що дозволяє чітко відокремити логіку оброблення даних від представлення та взаємодії з користувачем.

Для реалізації зазначених вимог обрано архітектурний підхід Model-View-Template, який широко застосовується у вебфреймворку Django та забезпечує ефективну організацію структури застосунку. На рис. 2.1 наведено загальну схему взаємодії компонентів у межах цієї архітектури.

Архітектура Model-View-Template передбачає розподіл програмного забезпечення на три основні складові, кожна з яких виконує окрему функцію у процесі оброблення запитів. Користувач ініціює взаємодію із системою шляхом

формування HTTP-запиту, який надходить до вебсерверної частини Django. Далі відбувається маршрутизація запиту через компонент URL, який визначає відповідну функцію оброблення, тобто View. Компонент View виступає центральною ланкою оброблення запиту, оскільки він отримує дані від користувача, формує запит до моделі та забезпечує підготовку результату для відображення.

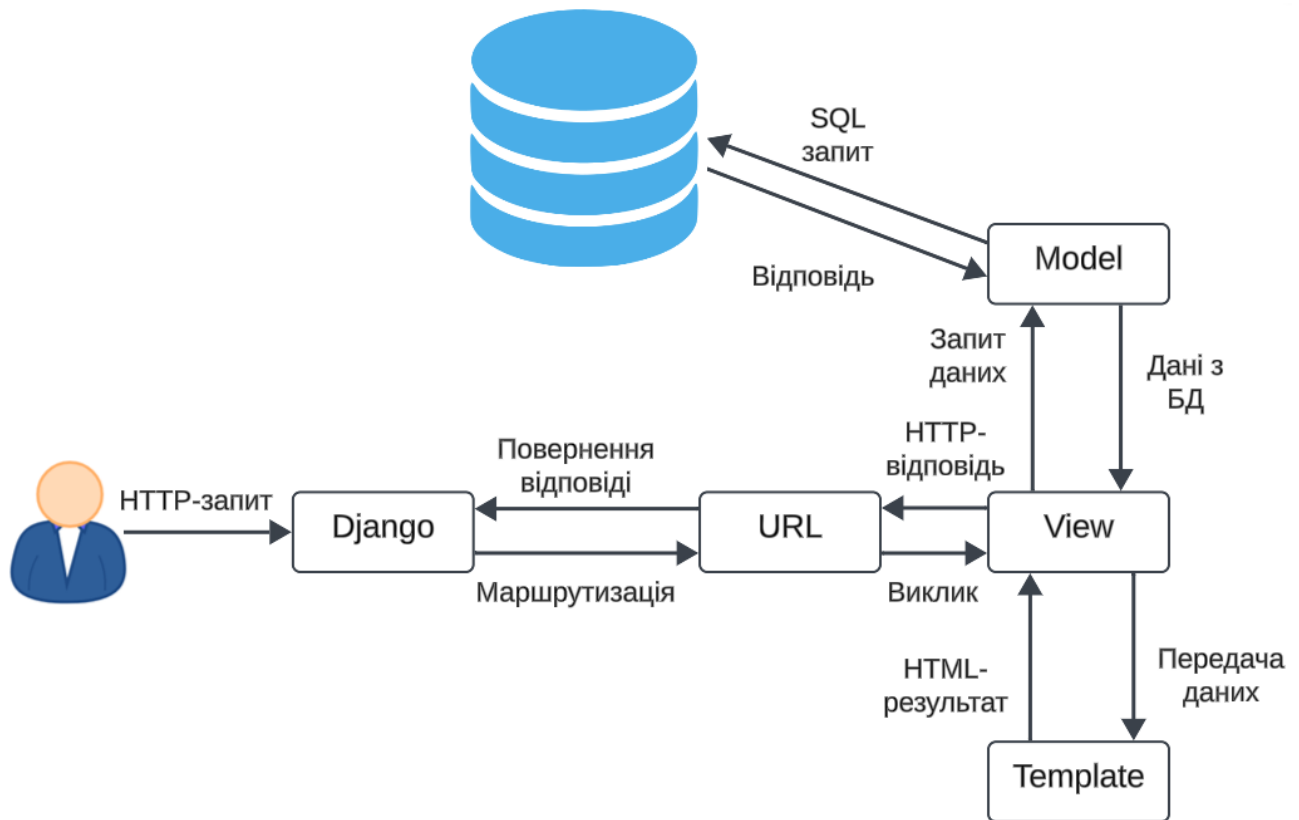


Рисунок 2.1 – Model View Template архітектура

Модель (Model) відповідає за роботу з базою даних і реалізує логіку доступу до даних. Вона формує SQL-запити до СУБД, отримує результати та повертає їх до View у структурованому вигляді. Таким чином забезпечується абстракція роботи з даними та ізоляція бізнес-логіки від інших компонентів системи. Після отримання необхідної інформації View передає її до шаблону (Template), який відповідає за формування HTML-сторінки та представлення даних у зручному для користувача вигляді.

Шаблон виконує функцію візуалізації, використовуючи передані дані для побудови інтерфейсу, що відображається у браузері користувача. Сформований результат повертається через View та URL-маршрутизацію у вигляді HTTP-відповіді до клієнта. Такий підхід забезпечує чітке розмежування відповідальностей між компонентами, підвищує модульність системи та спрощує її супровід, що є особливо важливим для систем, орієнтованих на оброблення та аналіз великих обсягів даних.

2.3 Проєктування структури бази даних

На етапі розроблення програмного забезпечення бізнес-аналітики продажів важливим є формування логічної та фізичної структури бази даних, яка забезпечує ефективне зберігання, оброблення та доступ до інформації. Структура бази даних повинна враховувати особливості предметної області, зокрема необхідність роботи з великими обсягами даних про продажі, забезпечення їх цілісності, унікальності та можливості подальшого аналітичного опрацювання. При цьому доцільно застосувати підхід, що поєднує елементи нормалізації для службових даних і денормалізації для основної таблиці продажів, що дозволяє спростити побудову звітів і підвищити продуктивність аналітичних запитів.

Виходячи з визначених функціональних вимог та логіки роботи системи, структура бази даних формується таким чином, щоб забезпечити підтримку процесів імпорту, зберігання, перевірки та аналізу даних, а також ведення журналів і збереження налаштувань звітів. У рамках проєктування передбачено використання обмеженої кількості таблиць, що дозволяє зберегти простоту реалізації та зручність подальшого супроводу системи. До складу бази даних входять такі основні таблиці:

Таблиця «sales_data» призначена для зберігання імпортованих даних про продажі товарів у підготовленому для аналітичного опрацювання вигляді, включаючи інформацію про товари, клієнтів, параметри продажу та службові

атрибути, необхідні для контролю цілісності даних і відстеження процесу імпорту (табл. 2.3).

Таблиця 2.3 – Атрибути сутності «sales_data»

№	Найменування	Тип даних	Призначення
1	sale_data_id	SERIAL	Ідентифікатор запису
2	sale_date	DATE	Дата здійснення продажу
3	product_name	VARCHAR(200)	Назва проданого товару
4	category_name	VARCHAR(100)	Назва категорії товару
5	customer_name	VARCHAR(150)	Ім'я або назва клієнта
6	region_name	VARCHAR(100)	Регіон здійснення продажу
7	sales_channel	VARCHAR(50)	Канал продажу (онлайн, офлайн тощо)
8	payment_method	VARCHAR(50)	Спосіб оплати
9	quantity	INTEGER	Кількість проданих одиниць товару
10	unit_price	NUMERIC(12,2)	Ціна за одиницю товару
11	discount_percent	NUMERIC(5,2)	Відсоток знижки
12	total_amount	NUMERIC(14,2)	Загальна сума продажу з урахуванням знижки
13	import_batch_id	VARCHAR(100)	Ідентифікатор пакету імпорту
14	row_hash	VARCHAR(255)	Контрольний хеш для виявлення дублікатів
15	created_at	TIMESTAMP	Дата та час додавання запису
16	user_id	INTEGER	Ідентифікатор користувача, який виконав імпорт

Таблиця «users» призначена для зберігання облікових записів користувачів системи, включаючи їх ідентифікаційні дані, параметри доступу та службову інформацію, що забезпечує реалізацію механізмів автентифікації та розмежування прав доступу (табл. 2.4).

Таблиця 2.4 – Атрибути сутності «users»

№	Найменування	Тип даних	Призначення
1	user_id	SERIAL	Ідентифікатор користувача
2	username	VARCHAR(100)	Унікальне ім'я користувача для входу в систему
3	email	VARCHAR(150)	Унікальна електронна адреса користувача
4	password_hash	VARCHAR(255)	Хешоване значення пароля користувача
5	role_name	VARCHAR(50)	Назва ролі користувача (адміністратор, аналітик)
6	is_active	BOOLEAN	Ознака активності облікового запису
7	created_at	TIMESTAMP	Дата та час створення облікового запису

Таблиця «saved_reports» призначена для збереження параметрів сформованих звітів та дашбордів, що дозволяє повторно використовувати налаштування аналітики та забезпечує персоналізацію роботи користувачів із системою (табл. 2.5).

Таблиця 2.5 – Атрибути сутності «saved_reports»

№	Найменування	Тип даних	Призначення
1	saved_report_id	SERIAL	Ідентифікатор збереженого звіту
2	report_name	VARCHAR(200)	Назва звіту

Продовження таблиці 2.5

№	Найменування	Тип даних	Призначення
3	report_type	VARCHAR(100)	Тип звіту або візуалізації
4	filter_json	JSONB	Збережені параметри фільтрації у JSON-форматі
5	created_by	VARCHAR(150)	Ім'я користувача, який створив звіт
6	created_at	TIMESTAMP	Дата та час створення звіту
7	is_shared	BOOLEAN	Ознака доступності звіту для інших користувачів
8	user_id	INTEGER	Ідентифікатор користувача, якому належить звіт

Таблиця «import_logs» призначена для зберігання інформації про процеси імпорту даних, включаючи параметри завантажених файлів, результати оброблення та статистику щодо помилок і дублікатів, що дозволяє контролювати якість даних та відстежувати історію імпортів (табл. 2.6).

Таблиця 2.6 – Атрибути сутності «import_logs»

№	Найменування	Тип даних	Призначення
1	import_log_id	SERIAL	Ідентифікатор запису імпорту
2	file_name	VARCHAR(255)	Назва імпортованого файлу
3	uploaded_by	VARCHAR(150)	Ім'я користувача, який виконав завантаження
4	uploaded_at	TIMESTAMP	Дата та час завантаження файлу
5	total_rows	INTEGER	Загальна кількість рядків у файлі
6	imported_rows	INTEGER	Кількість успішно імпортованих записів
7	duplicate_rows	INTEGER	Кількість виявлених дублікатів
8	invalid_rows	INTEGER	Кількість некоректних рядків

Продовження таблиці 2.6

№	Найменування	Тип даних	Призначення
9	status_name	VARCHAR(50)	Статус виконання імпорту
10	comment_text	TEXT	Опис помилок
11	user_id	INTEGER	Ідентифікатор користувача, який виконав імпорт

Сформована структура таблиць дозволяє перейти до представлення фізичної моделі бази даних, яка відображає взаємозв'язки між сутностями та організацію зберігання даних на рівні СУБД. Така модель забезпечує наочне уявлення про зв'язки між таблицями, зокрема залежності між користувачами, даними продажів, журналами імпорту та збереженими звітами. На рис. 2.2 наведено фізичну модель бази даних, побудовану на основі структури таблиць.

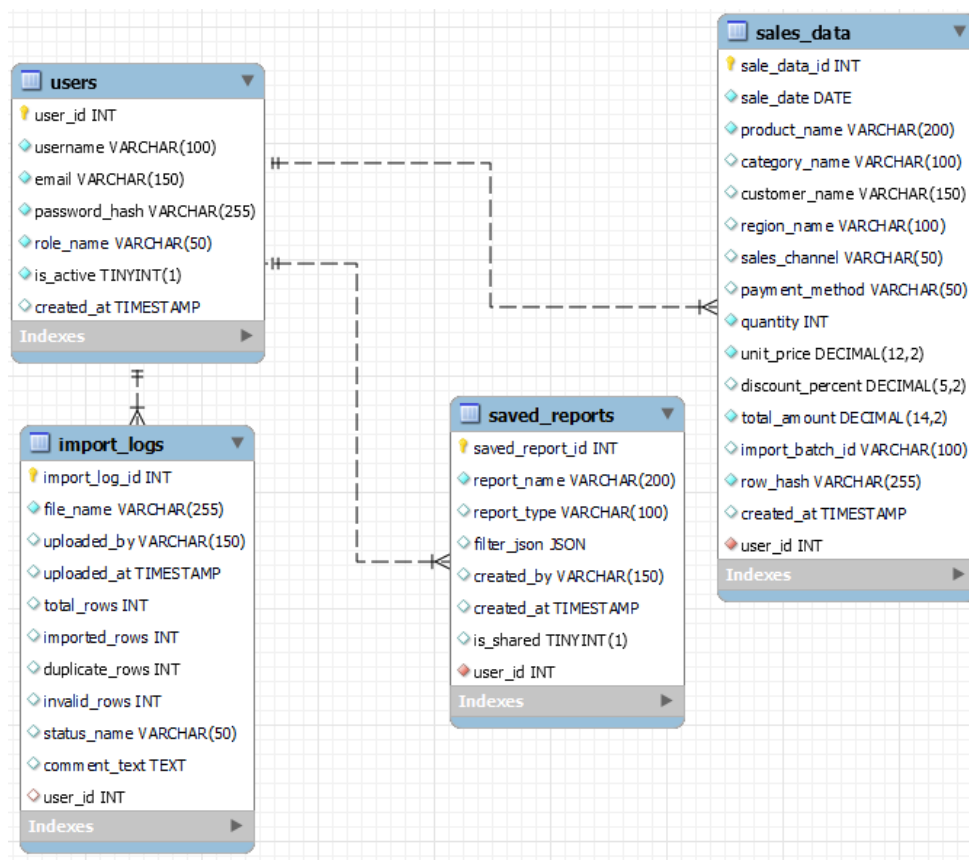


Рисунок 2.2 – Структура бази даних

Представлена модель дозволяє чітко простежити зв'язки між сутностями через зовнішні ключі, що забезпечує цілісність і узгодженість даних. Вона також створює основу для ефективного виконання запитів та аналітичних операцій, оскільки структура оптимізована для роботи з даними продажів. Використання такої моделі сприяє підвищенню продуктивності системи та спрощує її подальший супровід і розширення.

2.4 Розробка структурної схеми програмного проєкту

Розробка структурної схеми програмного проєкту. На етапі реалізації програмного забезпечення важливим є визначення внутрішньої структури системи, яка відображає взаємозв'язки між її основними компонентами та рівнями оброблення даних. Структурна схема дозволяє узагальнити архітектурні рішення, визначити логіку взаємодії модулів і забезпечити цілісне уявлення про функціонування програмного проєкту. Такий підхід сприяє впорядкуванню процесу розроблення, спрощує супровід системи та забезпечує можливість її подальшого розширення.

Для наочного представлення організації програмного забезпечення доцільно використати структурну схему, яка відображає розподіл системи на окремі модулі та рівні. На рис. 2.3 наведено структурну схему програмного проєкту, що реалізує систему бізнес-аналітики продажів.

Представлена схема демонструє, що програмний проєкт побудований за багаторівневою архітектурою, у межах якої виділяються рівень інтерфейсу користувача, серверна частина застосунку та рівень роботи з даними. На верхньому рівні розташовано вебінтерфейс, який реалізується за допомогою HTML-сторінок та компонентів Bootstrap і забезпечує взаємодію користувача із системою через форми, таблиці, графіки та KPI-дашборди. Паралельно

функціонує Django-застосунок, який виступає центральним елементом оброблення запитів і координує роботу всіх модулів системи.

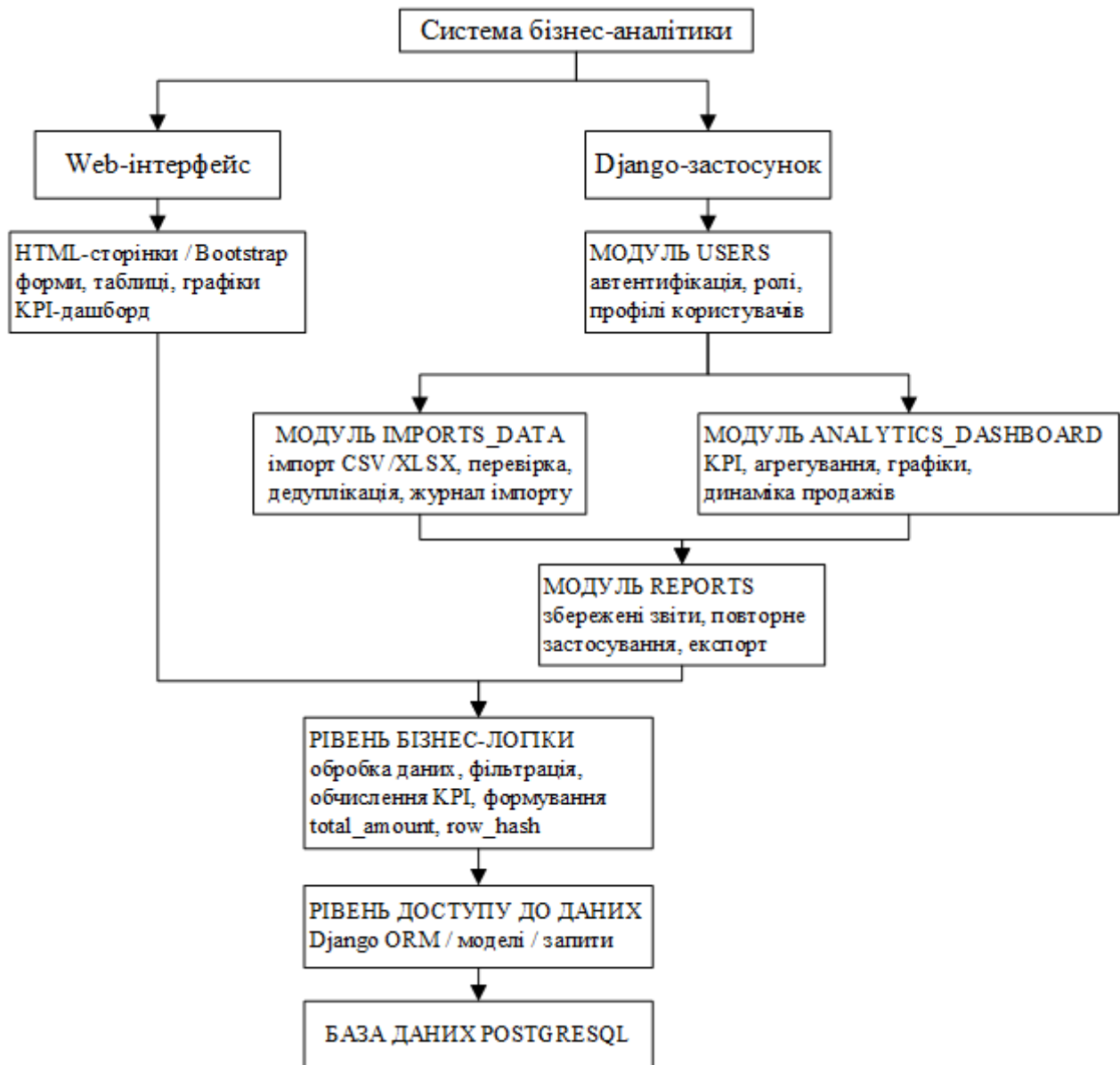


Рисунок 2.3 – Структурна схема програмного проєкту

У межах серверної частини виділено окремі функціональні модулі, кожен з яких відповідає за визначену групу задач. Модуль USERS забезпечує автентифікацію користувачів, управління ролями та профілями, що реалізує механізми безпеки та розмежування доступу. Модуль IMPORTS_DATA відповідає за імпорт даних із файлів CSV та XLSX, їх перевірку, виявлення

дублікатів і ведення журналу імпорту. Модуль `ANALYTICS_DASHBOARD` реалізує обчислення ключових показників, агрегацію даних і побудову візуалізацій, що дозволяє здійснювати аналіз динаміки продажів. Модуль `REPORTS` забезпечує формування, збереження, повторне використання звітів і експорт результатів аналізу.

Нижче розташовано рівень бізнес-логіки, який відповідає за оброблення даних, їх фільтрацію, обчислення показників ефективності та формування похідних значень, таких як загальна сума продажу та контрольні хеші для виявлення дублікатів. Наступним є рівень доступу до даних, що реалізується засобами Django ORM і забезпечує взаємодію між програмною логікою та базою даних через моделі та запити. Завершальним рівнем є база даних PostgreSQL, яка виконує функцію зберігання всієї інформації системи.

2.5 Висновки за розділом 2

У рамках даного розділу виконано проектування архітектури та структури програмного забезпечення системи бізнес-аналітики продажів, що дозволило сформувати цілісну технічну основу для подальшої реалізації. Проведено порівняльний аналіз мов програмування та систем керування базами даних, у результаті чого обрано Python як засіб реалізації серверної логіки та PostgreSQL як СУБД для зберігання та оброблення даних. Визначено архітектурний підхід на основі шаблону Model-View-Template, що забезпечує розмежування відповідальності між компонентами системи.

Розроблено структуру бази даних, що включає основні таблиці для зберігання даних. Це дозволило сформувати фізичну модель даних, орієнтовану на забезпечення цілісності, унікальності та ефективності виконання аналітичних запитів. Побудовано структурну схему програмного проекту, яка відображає розподіл системи на функціональні модулі та рівні оброблення даних, що забезпечує чітку організацію програмного рішення.

Отримані результати дозволяють перейти до наступного розділу, у якому буде реалізовано програмне забезпечення, зокрема розроблено алгоритми функціонування системи, описано модулі застосунку та забезпечено практичну реалізацію визначених архітектурних і структурних рішень.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Алгоритм функціонування програмного забезпечення

Функціонування системи бізнес-аналітики продажів базується на послідовній обробці даних, починаючи від їх завантаження, перевірки та збереження і завершуючи формуванням аналітичних звітів та візуалізацій. Це дозволяє забезпечити цілісність даних, автоматизацію аналітичних процесів і оперативне отримання результатів для підтримки управлінських рішень. Важливою складовою алгоритму є розмежування доступу до функціоналу системи відповідно до ролей користувачів, що забезпечує безпеку та логічну організацію роботи із системою. Додатково це сприяє підвищенню керованості системи, оскільки кожен користувач працює лише з тими функціями, які відповідають його повноваженням. Така організація дозволяє мінімізувати ризики помилок при роботі з даними та забезпечує стабільність функціонування системи в умовах багатокористувацького середовища. Крім того, послідовність оброблення даних забезпечує узгодженість усіх етапів аналітичного процесу, що є важливим для отримання достовірних результатів.

У межах системи передбачено дві ролі користувачів – адміністратор та аналітик, що зумовлено необхідністю розподілу функцій між процесами керування даними та їх аналітичного опрацювання. Адміністратор відповідає за імпорт, контроль якості даних, управління користувачами та загальний контроль роботи системи, тоді як аналітик використовує вже підготовлені дані для побудови звітів і аналізу показників. Такий розподіл дозволяє уникнути дублювання функцій, підвищити безпеку та забезпечити ефективну організацію роботи користувачів. Він також сприяє оптимізації робочих процесів, оскільки кожна роль виконує чітко визначений набір задач. У результаті забезпечується більш раціональне використання ресурсів системи та підвищується продуктивність роботи користувачів.

Для узагальнення логіки роботи системи та відображення послідовності виконання основних операцій доцільно представити алгоритм її функціонування у вигляді блок-схеми. На рис. 3.1 наведено алгоритм функціонування програмного забезпечення бізнес-аналітики продажів.

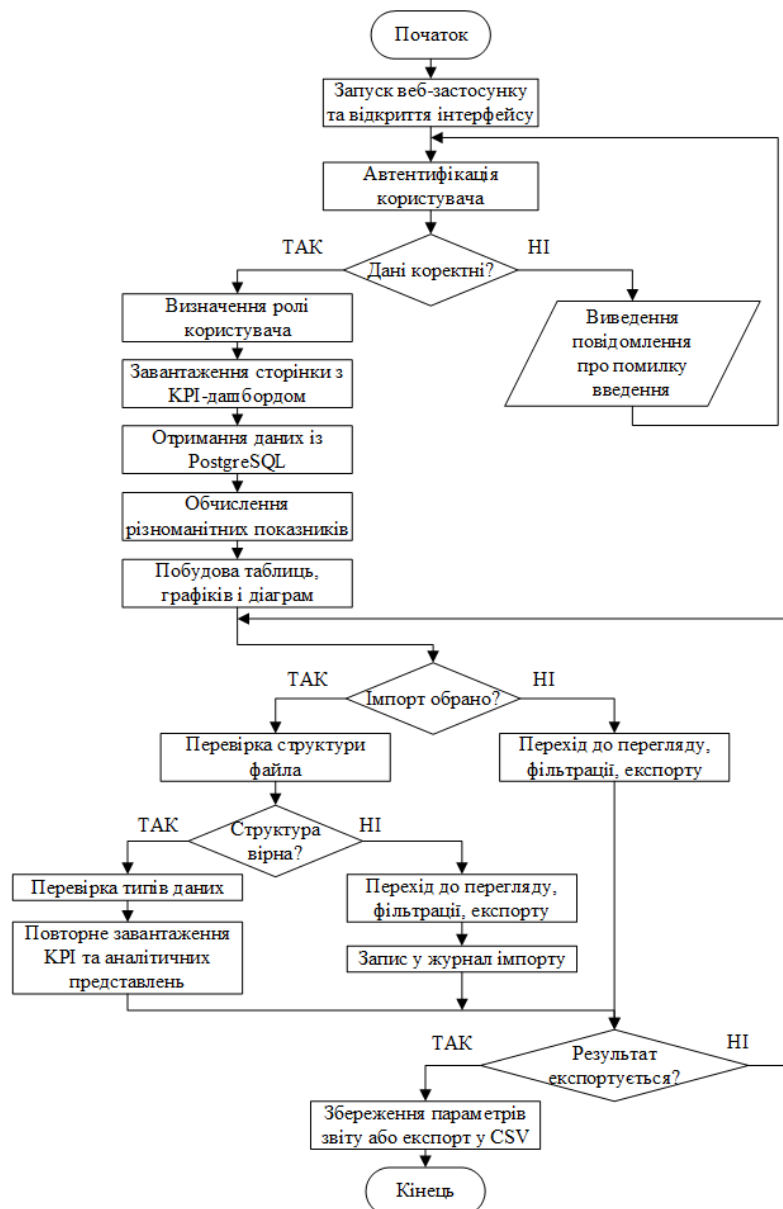


Рисунок 3.1 – Алгоритм функціонування програмного забезпечення

Робота системи розпочинається із запуску вебзастосунку та відкриття інтерфейсу користувача, після чого виконується процедура автентифікації. У разі введення коректних облікових даних здійснюється визначення ролі користувача,

що впливає на доступний функціонал, після чого відбувається завантаження сторінки з KPI-дашбордом. Система формує запити до бази даних PostgreSQL, отримує необхідні дані та виконує обчислення показників, що є основою для побудови таблиць, графіків і діаграм, які відображаються користувачеві.

Подальша логіка роботи залежить від вибору користувача щодо виконання імпорту даних. Якщо імпорт ініційовано, система перевіряє структуру файлу, після чого визначає її відповідність встановленим вимогам. У випадку коректної структури здійснюється перевірка типів даних, що забезпечує їх узгодженість та достовірність, після чого відбувається оновлення аналітичних представлень і повторне завантаження KPI. Якщо ж структура файлу не відповідає вимогам, система переходить до стандартного режиму роботи з даними, зокрема перегляду, фільтрації та експорту, одночасно фіксуючи результат імпорту у журналі.

У разі, якщо користувач не обирає імпорт, система одразу переходить до операцій перегляду та аналізу даних, що включає фільтрацію, формування звітів і підготовку результатів для експорту. Завершальним етапом алгоритму є перевірка необхідності експорту результатів, після чого система або зберігає параметри звіту, або виконує експорт даних у формат CSV. Таким чином, алгоритм відображає повний цикл функціонування програмного забезпечення – від оброблення запитів користувача до формування та збереження результатів аналітичної діяльності.

3.2 Опис модулів реалізованої програми

Опис модулів реалізованої програми. На етапі реалізації програмного забезпечення важливим є детальний розгляд функціональних модулів, які забезпечують виконання основних задач системи бізнес-аналітики продажів. Кожен модуль відповідає за окрему частину функціоналу, що дозволяє досягти модульності, спростити підтримку та забезпечити гнучкість у подальшому розширенні системи. Особливу роль відіграє модуль управління користувачами,

який реалізує механізми автентифікації, авторизації та роботи з профілями, забезпечуючи безпечний доступ до системи.

Для ілюстрації реалізації зазначеного функціоналу доцільно розглянути приклад програмного коду, який відповідає за оброблення процесів входу, виходу та перегляду профілю користувача. На рис. 3.2 наведено фрагмент коду, що реалізує відповідні компоненти системи.

```
class UserLoginView(LoginView):
    template_name = "users/login.html"
    authentication_form = LoginForm
    redirect_authenticated_user = True

class UserLogoutView(LogoutView):
    next_page = "login"

class ProfileView(LoginRequiredMixin, TemplateView):
    template_name = "users/profile.html"
```

Рисунок 3.2 – Оброблення процесів входу, виходу та перегляду профілю користувача

Наведений фрагмент демонструє реалізацію класів, які відповідають за автентифікацію користувача, завершення сеансу роботи та доступ до сторінки профілю. Клас, що відповідає за вхід у систему, використовує стандартні можливості фреймворку для оброблення авторизації, визначає шаблон інтерфейсу та форму введення облікових даних, а також забезпечує автоматичне перенаправлення користувача у разі, якщо він вже автентифікований. Клас виходу із системи виконує завершення користувацької сесії та визначає сторінку, на яку здійснюється перенаправлення після виходу.

Окремо реалізовано клас для роботи з профілем користувача, який забезпечує доступ лише для авторизованих користувачів, що досягається шляхом використання відповідного механізму перевірки доступу. Даний компонент

відповідає за відображення інформації профілю через визначений шаблон, забезпечуючи інтеграцію з інтерфейсом системи.

Наведений фрагмент програмної реалізації на рис. 3.3 описує форму автентифікації користувача, яка використовується для введення облікових даних під час входу в систему. У межах цієї реалізації визначаються два основних поля – ім'я користувача та пароль, які супроводжуються відповідними текстовими підписами для зручності сприйняття інтерфейсу. Особливістю є налаштування візуального представлення полів введення, зокрема застосування стилів для забезпечення узгодженості з дизайном вебінтерфейсу та використання підказок, що полегшують взаємодію користувача із системою.

```
class LoginForm(AuthenticationForm):
    username = forms.CharField(
        label="Ім'я користувача",
        widget=forms.TextInput(attrs={
            "class": "form-control",
            "placeholder": "Введіть ім'я користувача"
        })
    )

    password = forms.CharField(
        label="Пароль",
        widget=forms.PasswordInput(attrs={
            "class": "form-control",
            "placeholder": "Введіть пароль"
        })
    )
```

Рисунок 3.3 – Реалізація форми автентифікації користувача в системі

Поле введення імені користувача реалізовано у вигляді текстового елемента, який дозволяє вводити ідентифікаційні дані, тоді як поле пароля використовує спеціалізований тип введення, що приховує символи для підвищення безпеки. Такий підхід забезпечує базовий рівень захисту облікових даних та відповідає стандартним вимогам до реалізації механізмів автентифікації.

Крім того, використання спеціалізованої форми дозволяє інтегрувати процес перевірки введених даних із загальною логікою системи.

Фрагмент програмної реалізації на рис. 3.4 описує модель даних, яка використовується для збереження інформації про процеси імпорту даних у системі бізнес-аналітики продажів. У межах цієї моделі визначено набір атрибутів, що характеризують параметри завантаженого файлу, включаючи його назву, користувача, який виконав імпорт, а також дату і час виконання операції. Окрім цього, передбачено збереження статистичних показників, що відображають загальну кількість рядків у файлі, кількість успішно імпортованих записів, виявлених дублікатів і некоректних рядків, що дозволяє оцінити якість оброблення даних.

```
class ImportLog(models.Model):
    file_name = models.CharField(max_length=255, verbose_name="Назва файлу")
    uploaded_by = models.CharField(max_length=150, verbose_name="Користувач, який завантажив")
    uploaded_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата і час завантаження")
    total_rows = models.PositiveIntegerField(default=0, verbose_name="Усього рядків")
    imported_rows = models.PositiveIntegerField(default=0, verbose_name="Імпортовано рядків")
    duplicate_rows = models.PositiveIntegerField(default=0, verbose_name="Дублікатів")
    invalid_rows = models.PositiveIntegerField(default=0, verbose_name="Некоректних рядків")
    status_name = models.CharField(
        max_length=20,
        choices=ImportStatus.choices,
        default=ImportStatus.SUCCESS,
        verbose_name="Статус імпорту"
    )
    comment_text = models.TextField(blank=True, null=True, verbose_name="Коментар")
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name="import_logs",
        verbose_name="Користувач"
    )
```

Рисунок 3.4 – Реалізація моделі журналу імпорту даних у системі

Особливу увагу приділено фіксації статусу імпорту, який визначається за допомогою заздалегідь встановлених значень і відображає результат виконання операції, а також можливості збереження додаткового коментаря для уточнення деталей або опису помилок. Модель також передбачає встановлення зв'язку з користувачем системи, що забезпечує ідентифікацію автора імпорту та дозволяє організувати персоналізований доступ до журналу операцій.

Наведений фрагмент програмної реалізації на рис. 3.5 описує обробник запиту, який відповідає за імпорт файлу з даними продажів у систему та попередню ініціалізацію процесу його оброблення. Доступ до цього функціоналу обмежений лише користувачами з адміністративними правами, що забезпечується перевіркою ролі перед виконанням основної логіки, у разі невідповідності користувач отримує повідомлення про відмову в доступі та перенаправляється на іншу сторінку.

```
@login_required
def import_sales_file_view(request):
    if request.user.role_name != "ADMIN":
        messages.error(request,
            "У вас немає прав доступу до сторінки імпорту.")
        return redirect("home")
    import_result = None
    if request.method == "POST":
        form = SalesImportForm(request.POST, request.FILES)
        if form.is_valid():
            uploaded_file = form.cleaned_data["file"]
            file_info = save_uploaded_file(uploaded_file)
            import_log = ImportLog.objects.create(
                file_name=file_info["original_name"],
                uploaded_by=request.user.username,
                total_rows=0,
                imported_rows=0,
                duplicate_rows=0,
                invalid_rows=0,
                status_name=ImportStatus.SUCCESS,
                comment_text="Файл завантажено.
                Виконується перевірка структури та вмісту.",
                user=request.user
```

Рисунок 3.5 – Ініціалізація процесу імпорту файлу продажів та створення запису в журналі імпорту

У процесі виконання обробляється запит на завантаження файлу, після чого відбувається ініціалізація форми та перевірка коректності введених даних, включаючи сам файл. У разі успішної валідації виконується збереження файлу у файловій системі або іншому сховищі, а також формується службова інформація

про нього, яка буде використана на наступних етапах оброблення. Паралельно створюється запис у журналі імпорту, де фіксуються основні параметри операції, такі як назва файлу, користувач, що виконав завантаження, початкові значення статистики оброблення та поточний статус процесу.

Рис. 3.6 відображає метод реалізує оброблення запиту, пов'язаного з відображенням журналу імпорту даних у системі, і доступний лише для авторизованих користувачів. У процесі виконання формується вибірка записів журналу, яка включає інформацію про всі імпортовані файли разом із пов'язаними даними про користувачів, що дозволяє отримати повний контекст виконаних операцій. Записи впорядковуються за датою завантаження у спадному порядку, що забезпечує відображення найактуальніших імпортів у верхній частині списку.

```
@login_required
def import_logs_list_view(request):
    logs_qs =
    ImportLog.objects.select_related("user").all().order_by("-uploaded_at")

    paginator = Paginator(logs_qs, 15)
    page_number = request.GET.get("page")
    page_obj = paginator.get_page(page_number)

    context = {
        "page_title": "Журнал імпорту",
        "page_obj": page_obj,
    }
    return render(request, "imports_data/import_logs_list.html", context)
```

Рисунок 3.6 – Відображення журналу імпорту з підтримкою пагінації

Для оптимізації відображення великої кількості даних використовується механізм пагінації, який дозволяє розбити результати на окремі сторінки з обмеженою кількістю записів на кожній. Це підвищує зручність перегляду інформації та зменшує навантаження на інтерфейс користувача. На основі отриманих даних формується контекст, що передається до шаблону, який відповідає за відображення журналу імпорту у вигляді структурованого списку.

Наведений фрагмент програмної реалізації на рис. 3.7 описує обробник запиту, який відповідає за формування та відображення списку даних продажів із можливістю їх фільтрації за різними параметрами. На початковому етапі формується вибірка записів із бази даних разом із пов'язаними сутностями, що дозволяє отримати розширену інформацію про користувача та джерело імпорту, після чого дані впорядковуються за датою продажу та ідентифікатором у спадному порядку для відображення найактуальніших записів.

```
@login_required
def sales_data_list_view(request):
    sales_qs = SalesData.objects.select_related("user",
        "import_log").all().order_by("-sale_date", "-id")
    form = SalesFilterForm(request.GET or None)
    if form.is_valid():
        date_from = form.cleaned_data.get("date_from")
        date_to = form.cleaned_data.get("date_to")
        category_name = form.cleaned_data.get("category_name")
        region_name = form.cleaned_data.get("region_name")
        sales_channel = form.cleaned_data.get("sales_channel")
        payment_method = form.cleaned_data.get("payment_method")
        if date_from:
            sales_qs = sales_qs.filter(sale_date__gte=date_from)
        if date_to:
            sales_qs = sales_qs.filter(sale_date__lte=date_to)
        if category_name:
            sales_qs = sales_qs.filter(category_name__icontains=category_name.strip())
        if region_name:
            sales_qs = sales_qs.filter(region_name__icontains=region_name.strip())
        if sales_channel:
            sales_qs = sales_qs.filter(sales_channel__icontains=sales_channel.strip())
        if payment_method:
            sales_qs = sales_qs.filter(payment_method__icontains=payment_method.strip())
```

Рисунок 3.7 – Формування та фільтрація списку даних продажів за заданими параметрами

Ініціалізується форма фільтрації, яка обробляє параметри, передані у запиті, і у разі їх коректності виконується поступове уточнення вибірки. Зокрема, застосовуються умови відбору за часовим інтервалом, категорією товару, регіоном, каналом продажу та способом оплати, що дозволяє користувачеві гнучко формувати необхідний набір даних. Кожен параметр перевіряється на наявність значення, після чого до запиту додаються відповідні умови фільтрації, що забезпечує динамічне формування результатів залежно від введених критеріїв.

На рис. 3.8 метод описує обробник запиту, який відповідає за формування та відображення головної сторінки KPI-дашборду, доступної лише для авторизованих користувачів. На початковому етапі здійснюється ініціалізація форм для фільтрації даних та збереження звітів, після чого виконується обробка вхідних параметрів, зокрема визначення часового інтервалу, за яким буде здійснюватися аналіз. У разі коректності введених даних відбувається їх вилучення та передача до функцій, що забезпечують обчислення ключових показників.

```
@login_required
def dashboard_home_view(request):
    form = DashboardFilterForm(request.GET or None)
    save_form = SaveReportForm()
    date_from = None
    date_to = None
    if form.is_valid():
        date_from = form.cleaned_data.get("date_from")
        date_to = form.cleaned_data.get("date_to")
    kpis = get_dashboard_kpis(date_from=date_from, date_to=date_to)
    trend_data = get_sales_trend_data(date_from=date_from, date_to=date_to)
    category_data = get_sales_by_category_data(date_from=date_from, date_to=date_to)
    channel_data = get_sales_by_channel_data(date_from=date_from, date_to=date_to)
    context = {
        "page_title": "KPI-дашбордин",
        "form": form,
        "save_form": save_form,
        "kpis": kpis,
        "trend_labels_json": json.dumps(trend_data["labels"], ensure_ascii=False),
        "trend_values_json": json.dumps(trend_data["values"]),
        "category_labels_json": json.dumps(category_data["labels"], ensure_ascii=False),
        "category_values_json": json.dumps(category_data["values"]),
        "channel_labels_json": json.dumps(channel_data["labels"], ensure_ascii=False),
        "channel_values_json": json.dumps(channel_data["values"]),
    }
    return render(request, "analytics_dashboard/dashboard_home.html", context)
```

Рисунок 3.8 – Формування та відображення KPI-дашборду з урахуванням параметрів фільтрації

Подальша логіка передбачає отримання агрегованих аналітичних даних, які включають основні KPI, динаміку продажів у часі, розподіл за категоріями товарів та каналами збуту. Отримані результати структуруються у вигляді наборів значень і міток, які додатково перетворюються у формат JSON для подальшого використання у візуалізаціях на стороні клієнта. Це забезпечує можливість

побудови графіків і діаграм безпосередньо у вебінтерфейсі з використанням сучасних засобів відображення даних.

На завершальному етапі формується контекст, що містить усі необхідні дані для відображення сторінки, включаючи результати обчислень, параметри форм і службову інформацію, після чого виконується рендеринг HTML-шаблону.

Рис. 3.9 відображає метод, який відповідає за збереження параметрів аналітичного звіту, сформованого користувачем у системі.

```
@login_required
def save_report_view(request):
    if request.method != "POST":
        return redirect("dashboard_home")
    form = SaveReportForm(request.POST)
    if not form.is_valid():
        messages.error(request, "Некоректно заповнено назву звіту.")
        return redirect("dashboard_home")
    filter_json = {
        "date_from": request.POST.get("date_from", ""),
        "date_to": request.POST.get("date_to", ""),
    }
    SavedReport.objects.create(
        report_name=form.cleaned_data["report_name"],
        report_type=ReportType.DASHBOARD,
        filter_json=filter_json,
        created_by=request.user.username,
        is_shared=False,
        user=request.user,
    )
```

Рисунок 3.9 – Збереження параметрів аналітичного звіту користувача

Виконання цієї функції можливе лише для авторизованих користувачів і передбачає обробку POST-запиту, що містить введені дані форми. У разі невідповідності методу запиту або некоректного заповнення форми користувач отримує повідомлення про помилку, після чого здійснюється перенаправлення на головну сторінку дашборду.

Після успішної валідації введених даних відбувається формування структури параметрів фільтрації, зокрема значень початкової та кінцевої дати, які

використовуються для подальшого відтворення звіту. Ці параметри зберігаються у форматі JSON, що забезпечує гнучкість у подальшому використанні та дозволяє відновлювати налаштування аналітики без повторного введення даних. Далі виконується створення нового запису у відповідній таблиці бази даних, де фіксується назва звіту, його тип, параметри фільтрації, інформація про користувача та ознака доступності. Завершальним етапом є формування повідомлення про успішне збереження параметрів та перенаправлення користувача до сторінки зі списком збережених звітів.

Наведений метод на рис. 3.10 реалізує оброблення запиту, пов'язаного з відображенням списку збережених звітів користувача, і доступний лише для авторизованих користувачів системи. У процесі виконання формується вибірка записів із бази даних, яка містить лише ті звіти, що належать поточному користувачеві, із впорядкуванням за датою створення у спадному порядку, що дозволяє відображати найновіші записи першими.

```
@login_required
def saved_reports_list_view(request):
    reports =
    SavedReport.objects.filter(user=request.user).order_by("-created_at")
    context = {
        "page_title": "Збережені звіти",
        "reports": reports,
    }
    return render(request,
        "analytics_dashboard/saved_reports_list.html", context)
```

Рисунок 3.10 – Відображення списку збережених звітів користувача

Після отримання відповідних даних формується контекст, який містить службову інформацію для інтерфейсу та список звітів, що передається до шаблону для подальшого відображення. На завершальному етапі виконується генерація HTML-сторінки, яка представляє користувачеві перелік збережених звітів у зручному вигляді.

Наведений метод на рис. 3.11 описує обробник запиту, який забезпечує повторне застосування параметрів раніше збереженого звіту для відображення аналітичних даних. Виконання цього методу доступне лише для авторизованих користувачів і передбачає отримання відповідного звіту з бази даних із перевіркою його належності поточному користувачеві, що гарантує безпечний доступ до даних. У разі відсутності такого запису система автоматично генерує помилку, що запобігає несанкціонованому доступу.

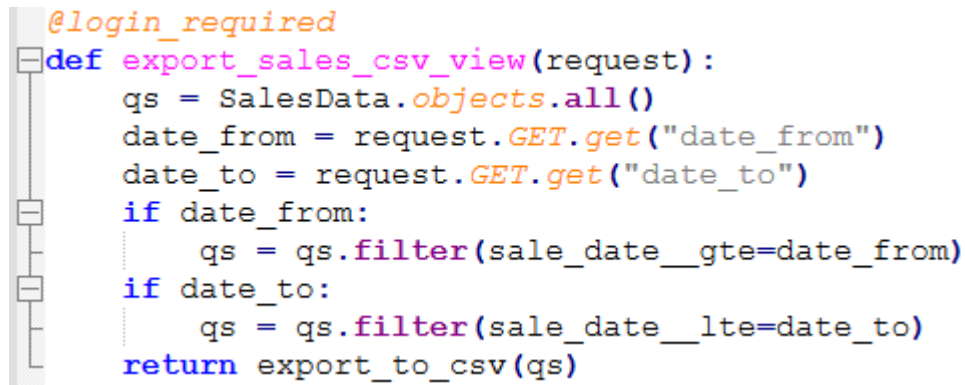
```
@login_required
def apply_saved_report_view(request, report_id):
    report = get_object_or_404(SavedReport, id=report_id, user=request.user)
    filter_json = report.filter_json or {}
    query_params = []
    if filter_json.get("date_from"):
        query_params.append(f"date_from={filter_json['date_from']}")
    if filter_json.get("date_to"):
        query_params.append(f"date_to={filter_json['date_to']}")
    query_string = "&".join(query_params)
    if query_string:
        return redirect(f"?{query_string}")
    return redirect("dashboard_home")
```

Рисунок 3.11 – Застосування збережених параметрів звіту для формування аналітичного відображення

Після отримання звіту виконується аналіз збережених параметрів фільтрації, які представлені у структурованому форматі, та формується набір параметрів запиту, що відповідають умовам фільтрації, зокрема часовому інтервалу. Ці параметри перетворюються у рядок запиту, який додається до адреси головної сторінки системи, що дозволяє автоматично відобразити аналітичні дані відповідно до заданих умов. У випадку відсутності параметрів фільтрації виконується стандартне перенаправлення на головну сторінку дашборду.

Метод на рис. 3.12 описує обробник запиту, який забезпечує експорт даних про продажі у формат CSV з урахуванням заданих параметрів фільтрації. Виконання цього методу доступне лише для авторизованих користувачів і

передбачає формування вибірки даних із бази, яка спочатку містить усі записи, а далі уточнюється відповідно до вхідних параметрів. Зокрема, здійснюється перевірка наявності значень початкової та кінцевої дати, що дозволяє обмежити вибірку даних заданим часовим інтервалом.



```
@login_required
def export_sales_csv_view(request):
    qs = SalesData.objects.all()
    date_from = request.GET.get("date_from")
    date_to = request.GET.get("date_to")
    if date_from:
        qs = qs.filter(sale_date__gte=date_from)
    if date_to:
        qs = qs.filter(sale_date__lte=date_to)
    return export_to_csv(qs)
```

Рисунок 3.12 – Експорт відфільтрованих даних продажів у формат CSV

У процесі виконання відбувається послідовне застосування умов фільтрації, які забезпечують відбір лише тих записів, що відповідають визначеним критеріям, зокрема належать до заданого періоду. Це дозволяє формувати результати, релевантні до потреб користувача, та уникати надлишкового обсягу інформації у вихідному файлі. Після формування остаточної вибірки виконується передача даних до відповідної функції, яка здійснює їх перетворення у формат CSV і ініціює процес завантаження файлу.

Наведений фрагмент програмної реалізації на рис. 3.13 описує модель даних, яка призначена для збереження аналітичних звітів користувачів разом із параметрами їх формування. У межах цієї моделі визначаються основні атрибути, що характеризують звіт, зокрема його назва, тип і набір параметрів фільтрації, які зберігаються у структурованому форматі. Це дозволяє відтворювати звіти у майбутньому без повторного введення умов, що підвищує зручність роботи із системою.

```

class SavedReport(models.Model):
    report_name = models.CharField(max_length=255, verbose_name="Назва звіту")
    report_type = models.CharField(
        max_length=50,
        choices=ReportType.choices,
        verbose_name="Тип звіту"
    )
    filter_json = models.JSONField(verbose_name="Параметри фільтрації")
    created_by = models.CharField(max_length=150, verbose_name="Створив")
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата створення")
    is_shared = models.BooleanField(default=False, verbose_name="Спільний доступ")
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name="saved_reports",
        verbose_name="Користувач"
    )

```

Рисунок 3.13 – Реалізація моделі збереження параметрів аналітичних звітів користувача

Додатково фіксується інформація про користувача, який створив звіт, а також дата його створення, що забезпечує можливість ведення історії аналітичної діяльності. Передбачено також атрибут, який визначає доступність звіту для інших користувачів, що дозволяє організувати спільне використання аналітичних результатів у межах системи. Зв'язок із користувачем реалізовано через зовнішній ключ, що забезпечує цілісність даних і дозволяє пов'язати кожен звіт із конкретним власником.

Метод на рис. 3.14 реалізує функціонал експорту даних про продажі у формат CSV, забезпечуючи формування файлу для подальшого завантаження користувачем. У процесі виконання створюється відповідь вебсервера зі встановленим типом вмісту та параметрами, які ініціюють автоматичне завантаження файлу з визначеною назвою. Додатково здійснюється запис спеціального символу кодування, що гарантує коректне відображення текстових даних при відкритті файлу у середовищах, таких як Microsoft Excel.

```

def export_to_csv(queryset):
    response = HttpResponse(content_type="text/csv; charset=utf-8")
    response["Content-Disposition"] = 'attachment; filename="sales_export.csv"'
    # Додаємо BOM для коректного відкриття UTF-8 у Excel/Windows
    response.write("\u00ef\u00bb\u00bf")
    writer = csv.writer(response)
    writer.writerow([
        "Дата", "Товар", "Категорія", "Клієнт",
        "Region", "Канал", "Оплата",
        "Кількість", "Ціна", "Знижка", "Сума"
    ])
    for obj in queryset:
        writer.writerow([
            obj.sale_date,
            obj.product_name,
            obj.category_name,
            obj.customer_name,
            obj.region_name,
            obj.sales_channel,
            obj.payment_method,
            obj.quantity,
            obj.unit_price,
            obj.discount_percent,
            obj.total_amount,
        ])
    return response

```

Рисунок 3.14 – Формування CSV-файлу для експорту даних продажів

Після ініціалізації об'єкта для запису даних формується заголовок таблиці, який визначає структуру вихідного файлу та відповідає основним атрибутам даних продажів. Далі виконується послідовний обхід переданої вибірки об'єктів, у межах якого кожен запис перетворюється у рядок CSV-файлу з відповідними значеннями полів.

Розроблена система має модульну структуру, у якій кожен функціональний компонент відповідає за окремий етап оброблення даних – від їх імпорту та валідації до аналітичного оброблення, візуалізації та формування звітів, що забезпечує зручність супроводу та розширення програмного забезпечення.

3.3 Висновки за розділом 3

У рамках даного розділу виконано розробку програмного забезпечення бізнес-аналітики продажів, що дозволило реалізувати визначені на попередніх

етапах архітектурні та функціональні рішення. Побудовано алгоритм функціонування системи, який відображає повний цикл оброблення даних — від автентифікації користувача та імпорту інформації до формування аналітичних звітів і візуалізацій. Це забезпечило логічну узгодженість усіх процесів та визначило послідовність взаємодії компонентів системи.

Детально опрацьовано основні програмні модулі, зокрема механізми автентифікації та управління користувачами, оброблення імпорту даних, ведення журналу імпортів, формування та фільтрацію даних продажів, побудову KPI-дашборду, а також створення, збереження і повторне використання аналітичних звітів. Окрему увагу приділено реалізації функцій експорту даних, що дозволяє використовувати результати аналізу поза межами системи. Усі модулі інтегровані між собою та забезпечують цілісність, узгодженість і ефективність оброблення інформації. Отримані результати дозволяють перейти до наступного розділу, у якому буде здійснено експлуатацію програмного забезпечення, проведено аналіз отриманих результатів та виконано тестування системи з метою оцінки її працездатності, надійності та відповідності поставленим вимогам.

4 ЕКСПЛУАТАЦІЯ, АНАЛІЗ РЕЗУЛЬТАТІВ ТА ТЕСТУВАННЯ СИСТЕМИ

4.1 Призначення та функціональні можливості програми

Розроблене програмне забезпечення призначене для автоматизації процесів збору, перевірки, зберігання та аналітичного опрацювання даних про продажі товарів з метою підтримки прийняття управлінських рішень. У ході реалізації створено веборієнтовану систему, яка забезпечує централізоване накопичення інформації з файлів, її формалізовану обробку та подальше представлення у вигляді аналітичних звітів, графіків і KPI-дашбордів. Реалізоване програмне рішення дозволяє користувачам оперативно отримувати узагальнену інформацію про динаміку продажів, структуру виручки та ключові показники діяльності підприємства .

Функціональні можливості реалізованої програми включають:

- автентифікація та розмежування доступу користувачів. Реалізовано механізм входу до системи із перевіркою облікових даних та розподілом прав доступу відповідно до ролей «адміністратор» і «аналітик», що забезпечує безпечну роботу та контроль доступу до функціоналу системи;
- імпорт даних продажів із файлів. Реалізовано можливість завантаження файлів у форматах CSV та XLSX, після чого система автоматично зчитує дані та ініціює процес їх оброблення із використанням бібліотеки обробки табличних даних;
- перевірка коректності структури та даних. У процесі імпорту здійснюється контроль наявності обов'язкових полів, відповідності типів даних, логічної узгодженості значень та можливості коректного зчитування файлу, що дозволяє виключити помилкові записи;
- виявлення та недопущення дублювання записів. Реалізовано механізм формування унікального контрольного хешу для кожного запису та перевірки його на наявність у базі даних, що забезпечує уникнення повторного збереження однакових даних;

- збереження даних у базі PostgreSQL. Усі перевірені записи зберігаються у централізованій базі даних, структура якої оптимізована для подальшого аналітичного опрацювання та швидкого виконання запитів;
- ведення журналу імпорту. Реалізовано збереження інформації про кожен процес імпорту, включаючи кількість оброблених, помилкових і дубльованих записів, що дозволяє контролювати якість даних та історію завантажень;
- аналітична обробка даних. Система виконує розрахунок ключових показників, таких як загальна виручка, кількість продажів, середній чек, обсяги реалізації за категоріями, регіонами та каналами збуту;
- формування KPI-дашборду. Реалізовано інтерактивну головну сторінку, на якій відображаються основні аналітичні показники у вигляді числових індикаторів, графіків і діаграм для швидкого аналізу стану продажів;
- фільтрація аналітичних даних. Користувач має можливість застосовувати фільтри за датою, категорією, товаром, клієнтом, регіоном, каналом продажу та способом оплати, що дозволяє формувати цільові вибірки даних;
- формування та збереження звітів. Реалізовано механізм створення аналітичних звітів із можливістю збереження параметрів їх побудови для подальшого повторного використання;
- перегляд збережених звітів та їх повторне застосування. Користувач може переглядати список раніше створених звітів та швидко відтворювати їх із використанням збережених параметрів фільтрації;
- експорт результатів аналізу. Передбачено можливість експорту даних у формат CSV, що дозволяє використовувати результати аналізу у зовнішніх системах або для подальшої обробки;
- перегляд та фільтрація даних продажів. Реалізовано окремий інтерфейс для роботи з первинними даними, що дозволяє здійснювати їх перегляд, пошук і детальний аналіз;

– інформування користувача про результати операцій. Система відображає повідомлення про успішність імпорту, наявність помилок або дублікатів, що забезпечує зворотний зв'язок та підвищує зручність використання.

Реалізований функціонал у сукупності забезпечує повний цикл роботи з даними продажів – від їх завантаження до отримання аналітичних результатів, що відповідає поставленим вимогам і дозволяє ефективно використовувати систему для бізнес-аналітик

4.2 Умови виконання програмного забезпечення

Функціонування розробленого програмного забезпечення бізнес-аналітики продажів забезпечується за умови дотримання визначених вимог до апаратного, програмного та мережевого середовища, а також коректного налаштування компонентів системи. Застосунок реалізовано у вигляді веборієнтованої клієнт-серверної системи, де клієнтська частина виконується у браузері користувача, а серверна – на базі фреймворку Django із використанням СУБД PostgreSQL, розгорнутої в контейнерному середовищі Docker.

Апаратні вимоги до серверної частини є помірними та відповідають типовим умовам розгортання навчальних або малих корпоративних інформаційних систем. Для стабільної роботи достатньо персонального комп'ютера або сервера з процесором рівня Intel Core i3 / AMD Ryzen 3 або вище, обсягом оперативної пам'яті не менше 4 ГБ (рекомендовано 8 ГБ), а також наявністю вільного дискового простору від 2 ГБ для розміщення програмного коду, контейнерів Docker та бази даних. Для клієнтської частини достатньо будь-якого сучасного пристрою (ПК або ноутбук), здатного запускати актуальний веббраузер.

Програмне середовище передбачає використання операційної системи сімейства Windows, Linux або macOS. На робочій станції має бути встановлено інтерпретатор мови програмування Python 3.10.1 або сумісної версії, що

забезпечує виконання серверної логіки застосунку. Для ізоляції залежностей використовується віртуальне середовище Python, у якому встановлюються необхідні бібліотеки, зокрема Django, Pandas, OpenPyXL, psycopg2-binary та python-dotenv. Важливою умовою є наявність встановленого та запущеного Docker, який забезпечує контейнеризацію бази даних PostgreSQL і дозволяє уникнути складної локальної інсталяції СУБД.

Мережева взаємодія в системі реалізується через протокол HTTP, що забезпечує обмін даними між клієнтською та серверною частинами. Для локального розгортання достатньо доступу до адреси <http://127.0.0.1:8000/>, тоді як у випадку розгортання в мережевому середовищі необхідно забезпечити відкриття відповідного порту та налаштування мережевого доступу до сервера. Для роботи з контейнеризованою базою даних використовується TCP-з'єднання через порт 5433, що проброшений із контейнера Docker на хост-систему.

Перед запуском програмного забезпечення необхідно виконати ініціалізацію середовища, що включає створення та активацію віртуального середовища Python, встановлення залежностей, запуск контейнера PostgreSQL та застосування міграцій бази даних. Міграції виконують створення структури таблиць відповідно до описаних моделей даних, що забезпечує коректну роботу системи без ручного втручання в структуру бази даних. За потреби додатково виконується відновлення даних із резервної копії у форматі JSON, що дозволяє швидко відтворити робочий стан системи на іншому комп'ютері.

Особливістю виконання програмного забезпечення є використання механізмів ORM, що абстрагують доступ до бази даних і забезпечують незалежність від конкретної СУБД. Це дозволяє при необхідності змінювати конфігурацію підключення до бази даних без модифікації бізнес-логіки системи. У процесі роботи система виконує оброблення табличних даних із використанням бібліотеки Pandas, що забезпечує ефективну валідацію, очищення та агрегацію даних перед їх збереженням у базі даних.

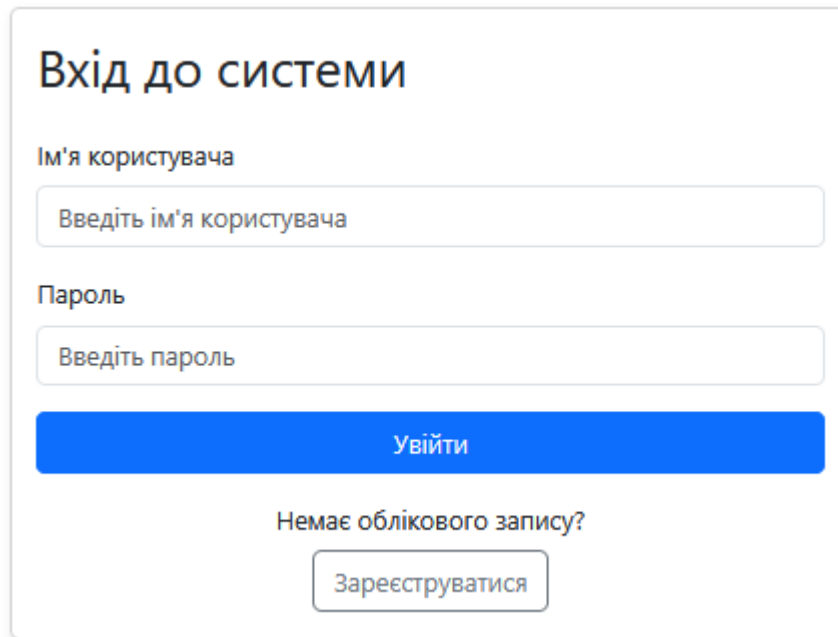
Для коректного функціонування інтерфейсу необхідна підтримка сучасних вебтехнологій, включаючи HTML5, CSS3 та JavaScript. Візуалізація аналітичних даних здійснюється за допомогою бібліотеки Chart.js, що дозволяє формувати інтерактивні графіки без додаткового встановлення клієнтського програмного забезпечення. Експорт результатів аналізу у формати CSV та Excel виконується на сервері з використанням стандартних бібліотек Python, що гарантує сумісність із офісними застосунками.

Таким чином, умови виконання програмного забезпечення характеризуються відсутністю жорстких апаратних вимог, використанням стандартного програмного стеку та можливістю швидкого розгортання системи на будь-якому сучасному комп'ютері. Використання Docker та PostgreSQL забезпечує стабільність, масштабованість і незалежність середовища виконання, що є важливою характеристикою для практичного застосування розробленої системи.

4.3 Тестування програмного забезпечення

На завершальному етапі розроблення системи бізнес-аналітики продажів виконано комплексне тестування, спрямоване на перевірку коректності реалізації функціональних можливостей, стабільності роботи та відповідності програмного забезпечення поставленим вимогам. У процесі тестування особлива увага приділяється перевірці механізмів автентифікації користувачів, оброблення даних, формування аналітичних звітів і взаємодії між основними компонентами системи. Це дозволяє виявити можливі помилки, оцінити зручність використання інтерфейсу та забезпечити надійність роботи програмного продукту в умовах експлуатації.

На початку проведено тестування форми входу до системи, яка забезпечує контроль доступу користувачів до функціоналу програми. На рис. 4.1 наведено інтерфейс форми автентифікації користувача.



Вхід до системи

Ім'я користувача

Введіть ім'я користувача

Пароль

Введіть пароль

Увійти

Немає облікового запису?

Зареєструватися

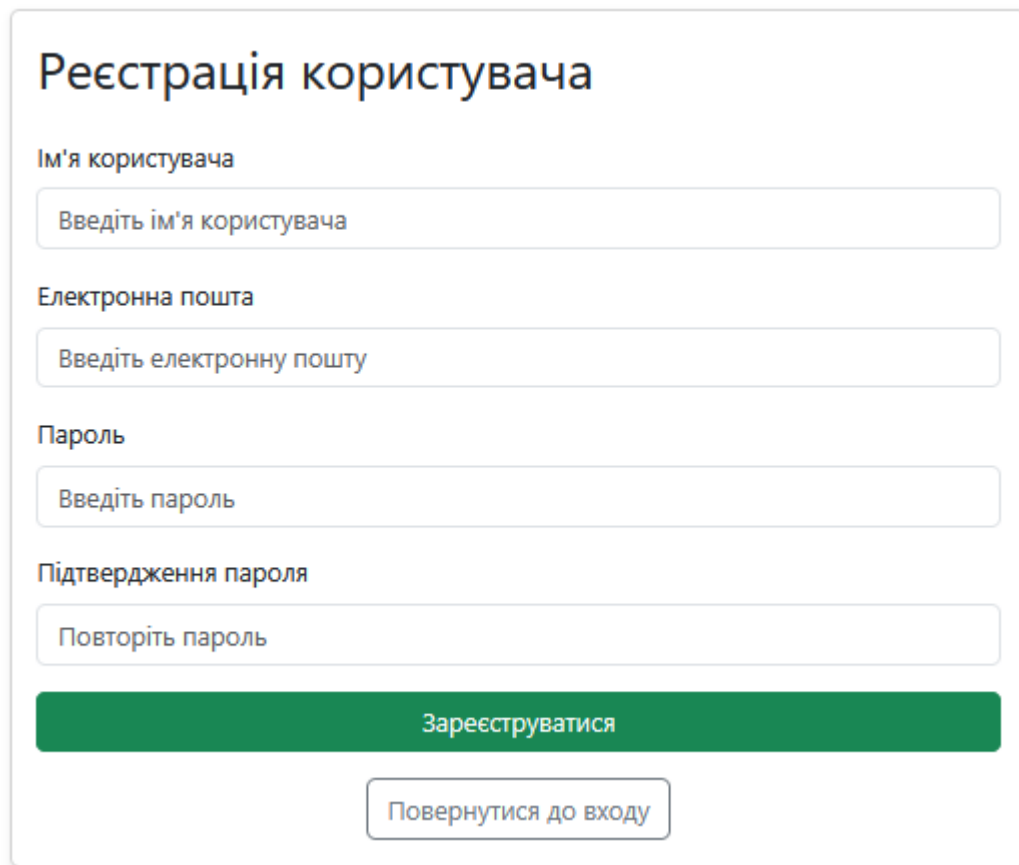
Рисунок 4.1 – Форма входу у систему

У процесі тестування встановлено, що форма входу коректно відображає поля для введення імені користувача та пароля, а також забезпечує зрозумілий інтерфейс взаємодії. Після введення валідних облікових даних система успішно виконує автентифікацію та перенаправляє користувача до головної сторінки. У випадку введення некоректних даних відображається відповідне повідомлення про помилку, що дозволяє користувачу своєчасно виправити введену інформацію.

Рис. 4.2 відображає форму реєстрації користувача, яка використовується для створення нового облікового запису в системі бізнес-аналітики продажів.

У процесі тестування встановлено, що форма коректно відображає всі необхідні поля для введення реєстраційних даних, зокрема ім'я користувача, електронну пошту, пароль та підтвердження пароля. При введенні валідної інформації система успішно виконує перевірку даних, зокрема відповідність паролів та унікальність облікових реквізитів, після чого створює новий обліковий запис і надає можливість подальшої автентифікації. У випадку некоректного введення, зокрема невідповідності паролів або відсутності обов'язкових даних,

система відображає відповідні повідомлення, що дозволяє користувачу виправити помилки.



The image shows a registration form titled "Реєстрація користувача" (User Registration). It contains four input fields: "Ім'я користувача" (Username) with placeholder "Введіть ім'я користувача", "Електронна пошта" (Email) with placeholder "Введіть електронну пошту", "Пароль" (Password) with placeholder "Введіть пароль", and "Підтвердження пароля" (Confirm password) with placeholder "Повторіть пароль". Below the fields is a large green button labeled "Зареєструватися" (Register) and a smaller button labeled "Повернутися до входу" (Return to login).

Рисунок 4.2 – Форма реєстрації користувача

На рис. 4.3 наведено сторінку перегляду профілю користувача, яка надає інформацію про обліковий запис у системі. У процесі тестування встановлено, що сторінка коректно відображає основні атрибути користувача, зокрема ім'я, електронну адресу, роль, статус активності та дату створення облікового запису. Дані завантажуються з бази даних та відображаються у зручному для сприйняття форматі без спотворень або втрати інформації. При цьому забезпечується відповідність відображених даних фактичному стану запису у системі, що підтверджує коректність інтеграції між рівнем представлення та базою даних.

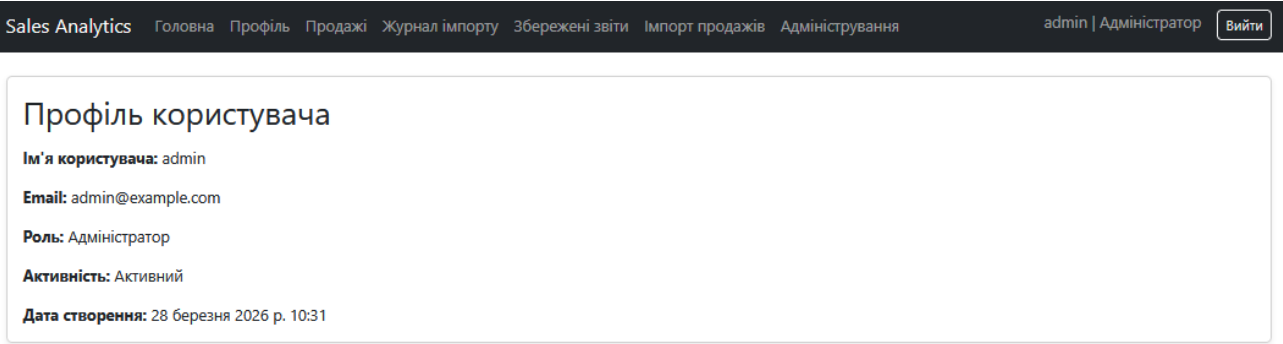


Рисунок 4.3 – Сторінка перегляду профілю користувача

Рис. 4.4 відображає сторінку імпорту файлу продажів, яка призначена для завантаження та первинного оброблення даних у системі.

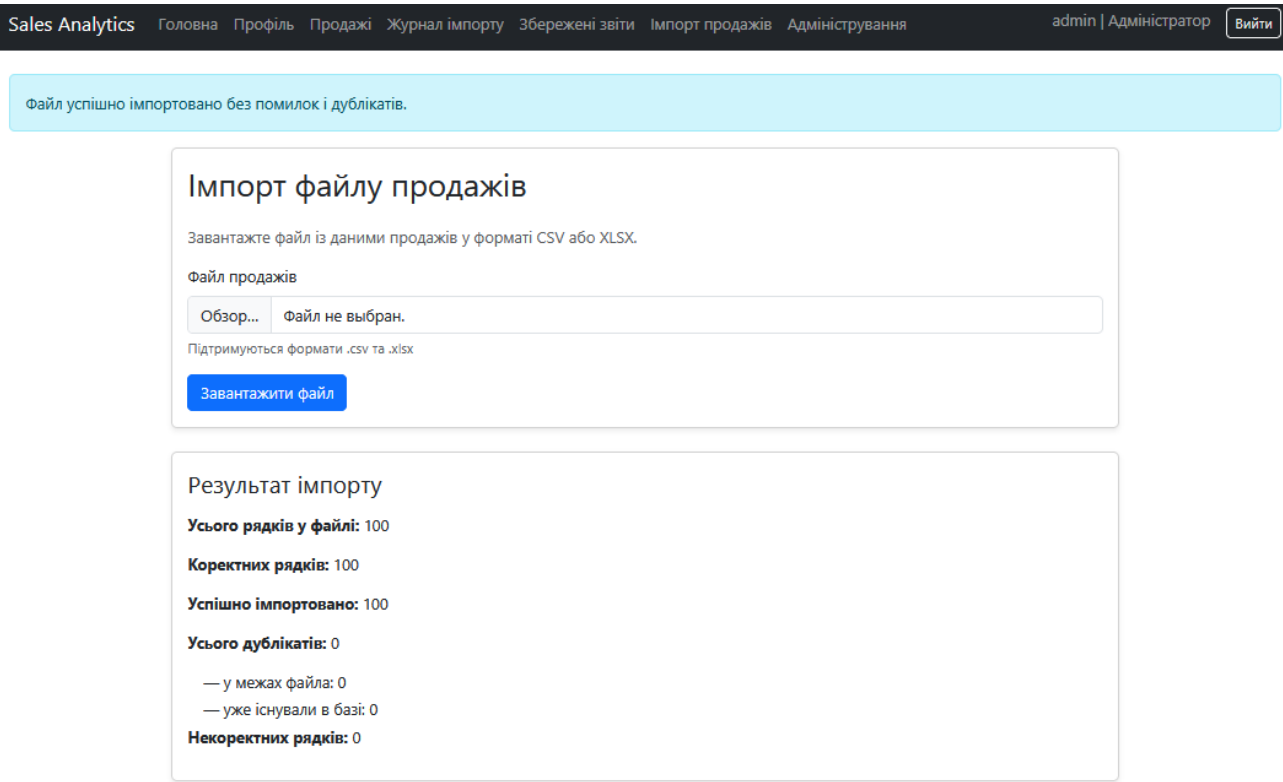


Рисунок 4.4 – Сторінка імпорту файлу продажів

Під час тестування встановлено, що інтерфейс сторінки забезпечує коректний вибір файлу у підтримуваних форматах та ініціює процес його завантаження без технічних збоїв. Після виконання імпорту система відображає інформаційне повідомлення про результат операції, а також детальну статистику

оброблення, включаючи загальну кількість рядків, число успішно імпортованих записів, кількість виявлених дублікатів та некоректних даних. Такий підхід дозволяє користувачу оперативно оцінити якість завантаженої інформації та переконатися у правильності її оброблення. Окремо перевірено, що при коректному файлі всі записи успішно імпортуються без втрат, а у випадку наявності помилок система фіксує їх у відповідних показниках і не допускає некоректні дані до збереження.

На рис. 4.5 наведено сторінку перегляду журналу імпорту даних, яка відображає історію виконаних операцій завантаження файлів у систему.

Sales Analytics Головна Профіль Продажі Журнал імпорту Збережені звіти Імпорт продажів Адміністрування admin Адміністратор Вийти									
Журнал імпорту									
ID	Файл	Користувач	Дата завантаження	Усього рядків	Імпортовано	Дублікатів	Некоректних	Статус	Коментар
7	sales_data.csv	admin	31.03.2026 17:46	100	100	0	0	Успішно	Усі рядки успішно імпортовано.
6	sales_data.csv	admin	28.03.2026 11:23	100	100	0	0	Успішно	Усі рядки успішно імпортовано.
5	test_corect.csv	admin	28.03.2026 11:07	2	0	2	0	Помилка	Не імпортовано жодного нового рядка.
4	test_dobl.csv	admin	28.03.2026 11:07	3	2	1	0	Частково успішно	Імпорт завершено частково: частина рядків імпортована, частина відхилена як помилкова або дублікати.
3	test_errors.csv	admin	28.03.2026 11:01	2	1	0	1	Частково успішно	Частина рядків містить помилки та не може бути імпортована.
2	sales_data.csv	admin	28.03.2026 11:00	100	100	0	0	Успішно	Усі рядки пройшли перевірку успішно.
1	sales_data.csv	admin	28.03.2026 10:52	0	0	0	0	Успішно	Файл успішно завантажено. Оброблення буде виконано на наступному етапі.

Рисунок 4.5 – Сторінка перегляду журналу імпорту даних

Сторінка коректно формує табличне представлення записів, яке містить ключові параметри кожного імпорту, зокрема назву файлу, користувача, дату та час завантаження, загальну кількість рядків, результати оброблення, а також статус виконання операції. Важливим аспектом є відображення детальної інформації щодо кількості імпортованих, дубльованих і некоректних записів, що дозволяє оцінити якість кожного завантаження без необхідності додаткового аналізу. Перевірка також показала, що статуси імпорту відображаються відповідно до фактичного результату оброблення, включаючи успішне

завершення, часткову обробку або наявність помилок. Коментарі, що супроводжують записи, надають додаткові пояснення щодо виконаних дій, що підвищує інформативність інтерфейсу.

Рис. 4.6 відображає фрагмент аналітичної сторінки, який відображає результати оброблення даних продажів у вигляді ключових показників та узагальнених характеристик.

Аналітична сторінка

Дата від: 20.05.2025 Дата до: 31.03.2026 Застосувати фільтр Скинути

Назва звіту: Продажі за останній рік Зберегти звіт

Загальна виручка 5425281,44	Кількість продажів 145	Продано одиниць 449	Середній чек 37415,73
---------------------------------------	----------------------------------	-------------------------------	---------------------------------

Найприбутковіша категорія Категорія: Smartphones Виручка: 1877718,54	Найактивніший регіон Регіон: Дніпро Виручка: 815400,17	Найпоширеніший канал продажу Канал: Offline Кількість продажів: 81
---	---	---

Експорт CSV Експорт Excel

Рисунок 4.6 – Фрагмент аналітичної сторінки

У процесі тестування встановлено, що сторінка коректно реагує на задані параметри фільтрації, зокрема часовий інтервал, та оновлює аналітичні показники відповідно до вибраних умов. Відображення KPI здійснюється у структурованому вигляді, що включає загальну виручку, кількість продажів, обсяг реалізованих одиниць та середній чек, при цьому всі значення узгоджуються з даними, що зберігаються у базі. Додатково перевірено коректність визначення узагальнених характеристик, таких як найприбутковіша категорія, найактивніший регіон та найпоширеніший канал продажу, що свідчить про правильність реалізації алгоритмів агрегації.

Окрему увагу приділено перевірці функцій збереження звітів та експорту даних, які доступні безпосередньо з аналітичної сторінки. Встановлено, що система коректно зберігає параметри аналізу та забезпечує формування файлів у відповідних форматах без втрати інформації.

На рис. 4.7 наведено фрагмент аналітичної сторінки, що містить графічне представлення результатів оброблення даних продажів у вигляді діаграм і графіків.

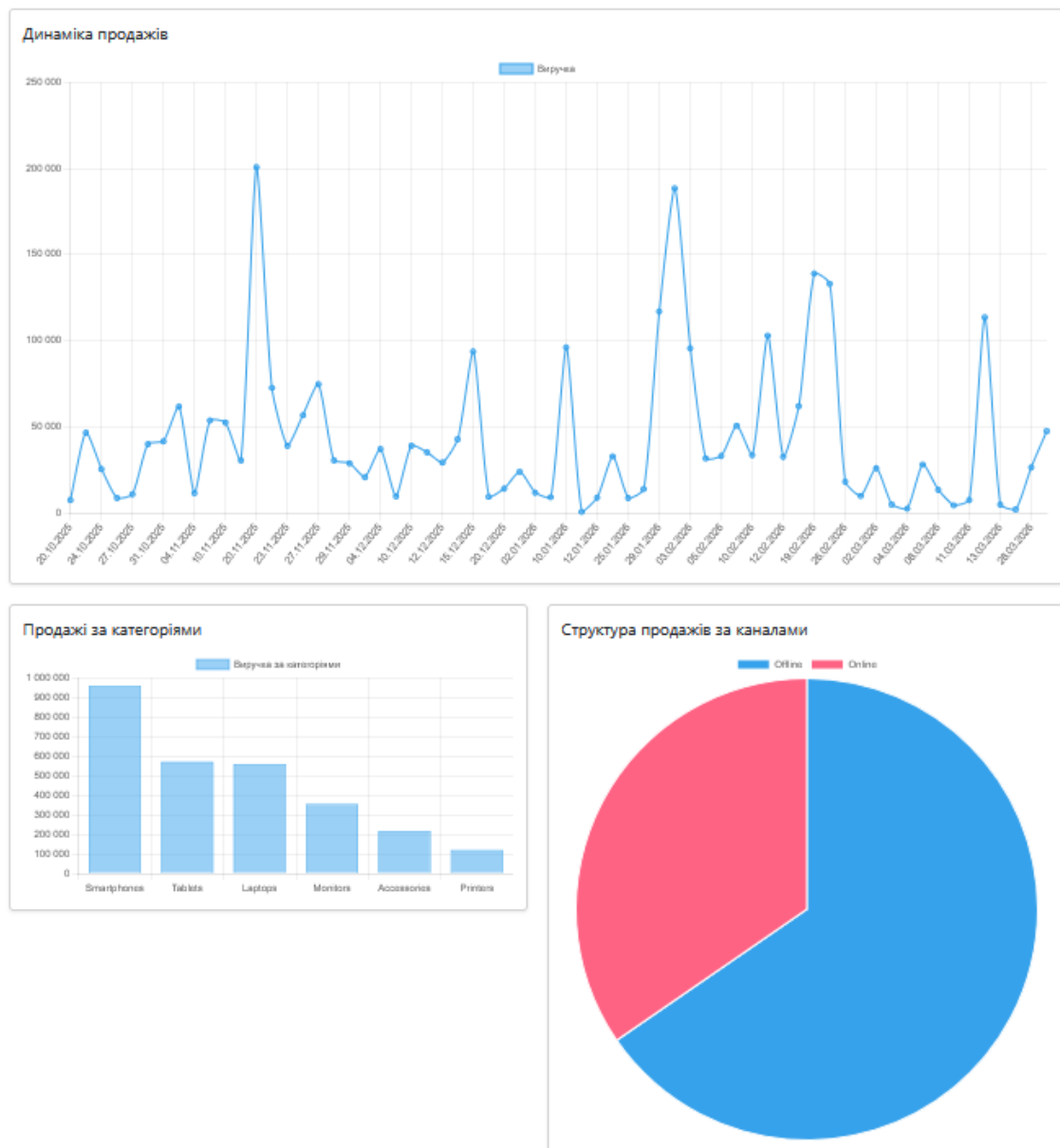


Рисунок 4.7 – Фрагмент аналітичної сторінки

У ході тестування підтверджено, що лінійний графік динаміки продажів коректно відображає зміну виручки у часі відповідно до обраного періоду, при цьому всі значення узгоджуються з агрегованими даними бази. Перевірка також показала, що стовпчикова діаграма продажів за категоріями правильно відображає співвідношення між різними групами товарів, що дозволяє виявляти найбільш прибуткові напрями. Кругова діаграма структури продажів за каналами демонструє розподіл між різними способами реалізації продукції, забезпечуючи наочне уявлення про їх частки.

Додатково встановлено, що всі візуальні елементи динамічно оновлюються при зміні параметрів фільтрації, а також коректно реагують на зміну вхідних даних без виникнення помилок відображення.

Рис. 4.8 висвітлює сторінку перегляду даних продажів, яка забезпечує доступ до детальної інформації про всі зафіксовані операції у системі.

Перегляд даних продажів

Дата від

Дата до

Категорія

Регіон

Канал продажу

Спосіб оплати

20.02.2025

20.03.2026

Наприклад, Laptops

Наприклад, Київ

Наприклад, Online

Наприклад, Card

Застосувати фільтри

Скинути

ID	Дата	Товар	Категорія	Клієнт	Регіон	Канал	Оплата	Кількість	Ціна	Знижка %	Сума
198	13.03.2026	Миша Logitech	Accessories	ФОП Іваненко	Харків	Offline	Cash	5	1020,83	0,00	5104,15
102	12.03.2026	Смартфон Xiaomi	Smartphones	ТОВ Альфа	Київ	Offline	Card	5	23394,57	3,00	113463,66
101	11.03.2026	Монітор Dell	Monitors	ТОВ Техно-Плюс	Одеса	Offline	Card	1	8464,50	10,00	7618,05
100	10.03.2026	Клавіатура Redragon	Accessories	ФОП Петренко	Ужгород	Offline	Card	5	986,86	5,00	4687,58
197	08.03.2026	Навушники HyperX	Accessories	ФОП Мельник	Харків	Offline	Card	8	1981,68	15,00	13475,42
99	07.03.2026	Монітор Samsung	Monitors	ТОВ Бета	Київ	Offline	Card	3	10464,75	10,00	28254,82
98	04.03.2026	Миша Logitech	Accessories	ФОП Петренко	Тернопіль	Offline	Card	5	584,56	10,00	2630,52
97	03.03.2026	Миша Logitech	Accessories	ТОВ Техно-Плюс	Вінниця	Online	Bank Transfer	8	717,55	10,00	5166,36
96	02.03.2026	Ноутбук HP	Laptops	ТОВ Гамма	Львів	Offline	Cash	1	26208,78	0,00	26208,78
196	27.02.2026	Клавіатура Redragon	Accessories	ТОВ Гамма	Дніпро	Online	Card	7	1683,57	15,00	10017,24
95	26.02.2026	Web-камера Logitech	Accessories	ТОВ Смарт Сервіс	Харків	Offline	Card	9	2028,34	0,00	18255,06
195	21.02.2026	Планшет Apple iPad	Tablets	ФОП Мельник	Дніпро	Offline	Cash	3	40519,83	10,00	109403,54
194	21.02.2026	МФУ Epson	Printers	ТОВ Альфа	Тернопіль	Offline	Card	2	11771,03	0,00	23542,06
94	19.02.2026	Смартфон Samsung Galaxy	Smartphones	ФОП Петренко	Тернопіль	Offline	Cash	5	29847,82	7,00	138792,36
93	14.02.2026	Планшет Apple iPad	Tablets	ТОВ Техно-Плюс	Одеса	Online	Bank Transfer	2	34466,31	10,00	62039,36

Рисунок 4.8 – Сторінка перегляду даних продажів

Інтерфейс сторінки коректно відображає табличні дані, включаючи такі атрибути, як дата продажу, товар, категорія, клієнт, регіон, канал реалізації, спосіб оплати, кількість, ціна, знижка та підсумкова сума. Особливу увагу приділено перевірці механізму фільтрації, який дозволяє задавати параметри відбору за часовим інтервалом, категорією, регіоном, каналом продажу та способом оплати, при цьому результати вибірки відповідають заданим умовам без спотворення даних. Оновлення таблиці відбувається динамічно після застосування фільтрів, що свідчить про коректну реалізацію оброблення запитів.

Додатково перевірено правильність відображення числових значень та їх узгодженість із даними, що зберігаються у базі, а також відсутність помилок при роботі з великими обсягами інформації. Таким чином, тестування підтверджує надійність реалізації функціоналу перегляду та фільтрації даних продажів, а також зручність його використання для аналітичних задач.

На рис. 4.9 наведено сторінку збережених звітів, яка призначена для перегляду та повторного використання раніше сформованих аналітичних результатів.

Sales Analytics
Головна
Профіль
Продажі
Журнал імпорту
Збережені звіти
Імпорт продажів
Адміністрування
admin | Адміністратор
Вийти

Параметри звіту успішно збережено.

Збережені звіти

ID	Назва звіту	Тип	Дата створення	Параметри	Дія
2	Продажі за останній рік	Дашборд	31.03.2026 17:51	{'date_to': '2026-03-31', 'date_from': '2025-05-20'}	Застосувати
1	Продажі	Дашборд	28.03.2026 11:46	{'date_to': '2026-11-20', 'date_from': '2025-10-20'}	Застосувати

Рисунок 4.9 – Сторінка збережених звітів

Сторінка коректно відображає перелік збережених звітів у табличному вигляді із зазначенням їх назви, типу, дати створення та параметрів фільтрації. Відображені параметри відповідають фактичним значенням, що були використані

при формуванні звітів, що забезпечує можливість їх точного відтворення. Перевірено, що при виборі відповідної дії система коректно застосовує збережені параметри та перенаправляє користувача до аналітичної сторінки з уже налаштованими умовами відбору даних. Додатково встановлено, що повідомлення про успішне збереження параметрів відображається без помилок, а всі записи у списку актуалізуються відповідно до останніх змін.

Рис. 4.10 відображає панель адміністратора, яка призначена для централізованого керування основними сутностями системи та виконання адміністративних операцій.

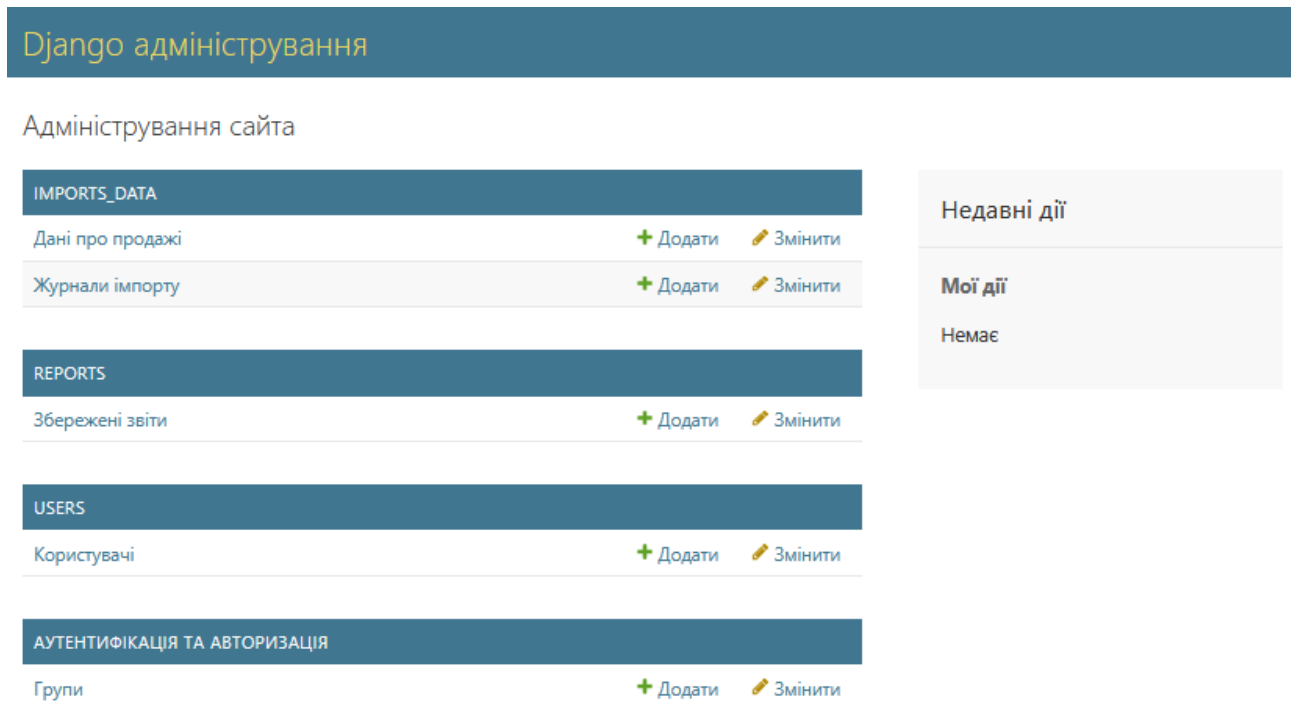


Рисунок 4.10 – Панель адміністратора

Інтерфейс панелі забезпечує зручний доступ до ключових розділів, зокрема управління даними продажів, журналами імпорту, збереженими звітами та обліковими записами користувачів. Адміністративні дії, такі як додавання нових записів або їх редагування, виконуються коректно, при цьому всі зміни відображаються у системі без затримок і втрати даних. Перевірка також показала, що доступ до панелі обмежений відповідною роллю, що підтверджує реалізацію

механізму розмежування прав доступу. Окремо перевірено відображення допоміжної інформації, зокрема блоку останніх дій, який дозволяє відстежувати виконані операції, що підвищує контроль за роботою системи.

На рис. 4.11 наведено сторінку перегляду даних про продажі в адміністративній панелі, яка використовується для детального контролю та управління записами у базі даних.

Вибір даних про продажі

ID	Дата продажу	Назва товару	Категорія	Клієнт	Регіон	Канал продажу	Спосіб оплати	Кількість	Ціна за одиницю	Знижка, %	Підсумкова сума
202	31 березня 2026 р.	Наушники HyperX	Accessories	ТОВ Бета	Київ	Online	Card	10	3055,48	10,00	27499,32
201	31 березня 2026 р.	Наушники HyperX	Accessories	ФОП Мельник	Івано-Франківськ	Online	Bank Transfer	7	2869,38	0,00	20085,66
200	28 березня 2026 р.	Монитор Samsung	Monitors	ТОВ Техно-Плюс	Вінниця	Online	Card	2	14025,81	5,00	26649,04
199	27 березня 2026 р.	Миша Logitech	Accessories	ФОП Петренко	Одеса	Online	Bank Transfer	2	1290,66	10,00	2323,19
198	13 березня 2026 р.	Миша Logitech	Accessories	ФОП Іваненко	Харків	Offline	Cash	5	1020,83	0,00	5104,15
102	12 березня 2026 р.	Смартфон Xiaomi	Smartphones	ТОВ Альфа	Київ	Offline	Card	5	23394,57	3,00	113463,66
101	11 березня 2026 р.	Монитор Dell	Monitors	ТОВ Техно-Плюс	Одеса	Offline	Card	1	8464,50	10,00	7618,05
100	10 березня 2026 р.	Клавіатура Redragon	Accessories	ФОП Петренко	Ужгород	Offline	Card	5	986,86	5,00	4687,58
197	08 березня 2026 р.	Наушники HyperX	Accessories	ФОП Мельник	Харків	Offline	Card	8	1981,68	15,00	13475,42
99	07 березня 2026 р.	Монитор Samsung	Monitors	ТОВ Бета	Київ	Offline	Card	3	10464,75	10,00	28254,82
98	04 березня 2026 р.	Миша Logitech	Accessories	ФОП Петренко	Тернопіль	Offline	Card	5	584,56	10,00	2630,52
97	03 березня 2026 р.	Миша Logitech	Accessories	ТОВ Техно-Плюс	Вінниця	Online	Bank Transfer	8	717,55	10,00	5166,36
96	02 березня 2026 р.	Ноутбук HP	Laptops	ТОВ Гамма	Львів	Offline	Cash	1	26308,78	0,00	26308,78

Рисунок 4.11 – Сторінка перегляду даних про продажі

Інтерфейс забезпечує повноцінне відображення всіх атрибутів записів, включаючи дату продажу, назву товару, категорію, клієнта, регіон, канал збуту, спосіб оплати, кількість, ціну, знижку та підсумкову суму. Дані відображаються у вигляді таблиці з можливістю сортування та фільтрації, що дозволяє швидко знаходити необхідну інформацію навіть при значних обсягах записів. Перевірка також підтвердила коректність роботи пошуку та фільтрів за різними параметрами, що забезпечує гнучкість при аналізі інформації.

Рис. 4.12 відображає сторінку перегляду журналу імпорту в адміністративному інтерфейсі, яка забезпечує детальний контроль за всіма операціями завантаження даних у систему.

Виберіть Журнал імпорту щоб змінити

Діа: ----- Вперед Обрано 0 з 7

ID	НАЗВА файлу	КОРИСТУВАЧ, РОЙІ ЗАВАНТАЖИВ	ДАТА І ЧАС ЗАВАНТАЖЕННЯ	УСЬОГО РЯДКІВ	ІМПОРТОВАНО РЯДКІВ	ДУБЛІКАТІВ	НЕКОРЕКТНИХ РЯДКІВ	СТАТУС ІМПОРТУ
7	sales_data.csv	admin	31 березня 2026 р. 17:46	100	100	0	0	Успішно
6	sales_data.csv	admin	28 березня 2026 р. 11:23	100	100	0	0	Успішно
5	test_correct.csv	admin	28 березня 2026 р. 11:07	2	0	2	0	Помилка
4	test_dobl.csv	admin	28 березня 2026 р. 11:07	3	2	1	0	Частково успішно
3	test_errors.csv	admin	28 березня 2026 р. 11:01	2	1	0	1	Частково успішно
2	sales_data.csv	admin	28 березня 2026 р. 11:00	100	100	0	0	Успішно
1	sales_data.csv	admin	28 березня 2026 р. 10:52	0	0	0	0	Успішно

7 Журнали імпорту

ВІДФІЛЬТРОВАТИ

Показати кількість

За Статус імпорту

Всі

Успішно

Частково успішно

Помилка

За Дата і час завантаження

Будь-яка дата

Сьогодні

Останні 7 днів

Цього місяця

Цього року

Рисунок 4.12 – Сторінка перегляду журналу імпорту

У процесі тестування встановлено, що сторінка коректно відображає записи журналу із зазначенням ключових характеристик кожного імпорту, включаючи назву файлу, користувача, дату виконання, кількість оброблених рядків, а також результати перевірки даних. Інформація подається у впорядкованому вигляді, що дозволяє швидко оцінити успішність виконаних операцій та виявити можливі проблеми. Додатково перевірено роботу вбудованих засобів фільтрації, які дають змогу відбирати записи за статусом імпорту або часовими параметрами, що підвищує зручність аналізу історії завантажень.

На рис. 4.13 наведено сторінку перегляду збережених звітів в адміністративному інтерфейсі, яка забезпечує централізоване управління аналітичними звітами системи. У процесі тестування підтверджено, що сторінка коректно відображає список звітів із зазначенням їх основних характеристик, зокрема назви, типу, користувача, який створив звіт, дати створення та ознаки спільного доступу. Дані подаються у структурованому вигляді, що дозволяє швидко оцінити наявні звіти та їх властивості. Перевірка також показала, що функціонал пошуку та фільтрації працює відповідно до заданих критеріїв, забезпечуючи зручний доступ до потрібних записів навіть при збільшенні обсягу даних.

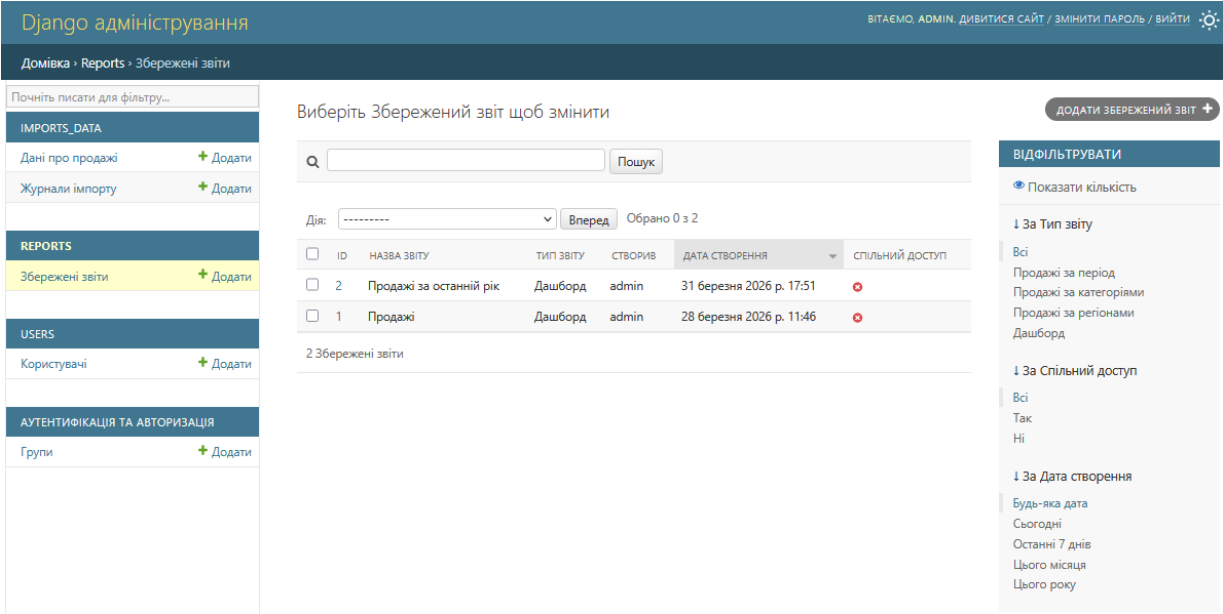


Рисунок 4.13 – Сторінка перегляду збережених звітів

На рис. 4.14 наведено сторінку перегляду користувачів в адміністративному інтерфейсі, яка призначена для контролю та управління обліковими записами системи.

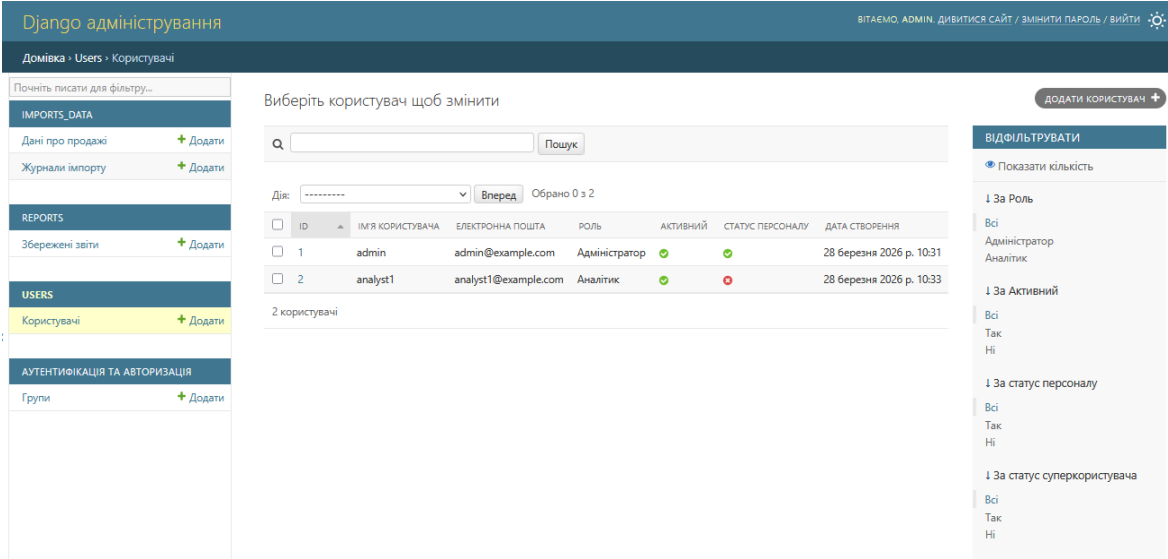


Рисунок 4.14 – Сторінка перегляду користувачів

У ході тестування встановлено, що сторінка коректно відображає список користувачів із зазначенням основних характеристик, зокрема імені, електронної

пошти, ролі, статусу активності та дати створення облікового запису. Дані відображаються у впорядкованому табличному вигляді, що дозволяє швидко ідентифікувати користувачів та оцінити їх поточний стан. Перевірка функціоналу фільтрації підтвердила можливість відбору записів за ролями, активністю та іншими параметрами, що спрощує адміністрування системи.

4.4 Висновки за розділом 4

У рамках даного розділу виконано опис призначення розробленого програмного забезпечення та розкрито його функціональні можливості, що підтверджують здатність системи забезпечувати повний цикл роботи з даними продажів – від їх імпорту та оброблення до формування аналітичних показників і звітів. Окрему увагу приділено визначенню умов виконання програмного забезпечення, що включають вимоги до програмного середовища, використовуваних технологій та особливостей розгортання системи, що забезпечує її стабільну роботу та можливість практичного використання.

Перевірено основні компоненти системи, зокрема механізми автентифікації та реєстрації користувачів, функціонал імпорту даних, ведення журналу імпортів, перегляд і фільтрацію даних продажів, формування КРІ-дашбордів, збереження та повторне застосування звітів, а також експорт результатів аналізу. Отримані результати підтвердили коректність роботи всіх реалізованих модулів, узгодженість оброблення даних та відповідність інтерфейсу вимогам зручності використання.

Проведене тестування дозволило встановити, що програмне забезпечення функціонує стабільно, забезпечує достовірність оброблення даних і може бути використане для підтримки прийняття управлінських рішень, що підтверджує досягнення поставлених цілей та готовність системи до практичного застосування.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процес розроблення програмного забезпечення для бізнес-аналітики продажів товарів.

Визначено, що існуючі підходи до аналізу продажів характеризуються фрагментованістю джерел даних, наявністю помилок і дублювання записів, а також складністю формування аналітичних звітів, що ускладнює прийняття обґрунтованих управлінських рішень та знижує оперативність отримання результатів.

Розроблено програмне забезпечення, яке забезпечує автоматизований імпорт даних продажів, їх перевірку на коректність, усунення дублікатів, збереження у базі даних та подальше аналітичне опрацювання. Реалізовано механізми автентифікації та розмежування доступу користувачів, формування KPI-дашбордів, фільтрації даних, створення та збереження аналітичних звітів, а також експорт результатів у формат CSV. Забезпечено узгоджену взаємодію між компонентами системи, що дозволяє виконувати повний цикл оброблення даних у межах єдиного програмного середовища.

Виконано тестування програмного забезпечення, яке охоплює перевірку роботи інтерфейсів користувача, механізмів автентифікації, імпорту даних, ведення журналу імпортів, формування аналітичних показників, візуалізації результатів та експорту даних.

Результати тестування програмного забезпечення показали, що система функціонує стабільно, коректно обробляє вхідні дані, забезпечує достовірність аналітичних результатів та відповідає поставленим функціональним і нефункціональним вимогам, що підтверджує можливість її ефективного практичного використання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Big data statistics. URL: <https://riverty.io/blog/big-data-statistics-how-much-data-is-there-in-the-world/> (date of access 27.03.2026).
2. Big Data Statistics 2026 (Growth, Trends & Market Size). URL: <https://www.demandsage.com/big-data-statistics/> (date of access 27.03.2026).
3. 85+ Big Data Statistics To Map Growth in 2025. URL: <https://www.g2.com/articles/big-data-statistics> (date of access 27.03.2026).
4. Sivarajah U., Kumar S., Kumar V., Chatterjee S., Li J. A study on big data analytics and innovation: From technological and business cycle perspectives. Technological Forecasting and Social Change. 2024. Vol. 202, 328 p.
5. Big Data Analytics Market Size. URL: <https://www.fortunebusinessinsights.com/big-data-analytics-market-106179> (date of access 27.03.2026).
6. Carlisle S. Software: Tableau and microsoft power bi. Technology| Architecture+ Design. 2018. Vol. 2, No 2, pp. 256-259.
7. Tableau Review: Features, Pricing, Pros and Cons. URL: <https://www.techrepublic.com/article/tableau-review/> (date of access 27.03.2026).
8. Krishnan A. R. Research trends in criteria importance through intercriteria correlation (CRITIC) method: a visual analysis of bibliographic data using the Tableau software. Information discovery and delivery. 2025. Vol. 53, No 2, pp. 233-247.
9. Product Information on Metabase. URL: <https://www.gartner.com/reviews/product/metabase> (date of access 27.03.2026).
10. Metabase | Reviews, Pricing & Demos - SoftwareAdvice AU. URL: <https://www.softwareadvice.com.au/software/312574/metabase> (date of access 27.03.2026).
11. Metabase Reviews & Product Details. URL: <https://www.g2.com/products/metabase/reviews> (date of access 27.03.2026).

12. Zoho Analytics. URL: <https://www.gartner.com/reviews/product/zoho-analytics> (date of access 27.03.2026).

13. Zoho Analytics Review | PCMag. URL: <https://www.pcmag.com/reviews/zoho-reports> (date of access 27.03.2026).

14. Srivastava G., Venkataraman R., A review of the state of the art in business intelligence software. *Enterprise Information Systems*. 2022. Vol. 16, No 1, pp. 1-28.

15. Бондаренко В.В., Коваленко В.В. Розробка програмного забезпечення з використанням Python, Java та C#: Збірник наукових праць. Одеса: Видавництво "Сонячна Україна", 2018. 250 с.

16. Opik O., Sunendar D., Damayanti V., Maftuh B., Rifanto D. Exploring multicultural narratives in 1000 library grant book: A focus on West Java context. *DWIJA CENDEKIA: Jurnal Riset Pedagogik*. 2025. Vol. 9, No 2, pp. 186-198.

17. Klochkov D., Mulawka J. Improving ruby on rails-based web application performance. *Information*. 2021. Vol. 12, No 8, 319 p.

18. Solarz A., Szymczyk T. Oracle 19c, SQL Server 2019, Postgresql 12 and MySQL 8 database systems comparison. *Journal of Computer Sciences Institute*. 2020. pp. 373-378.

19. Pendse S., Krishnaswamy V., Kulkarni K., Li Y., Lahiri T., Raja V., Kulkarni A. Oracle database in-memory on active data guard: Real-time analytics on a standby database. In *2020 IEEE 36th International Conference on Data Engineering* 2020. pp. 570-578.

20. Петров І. Основи Firebird SQL. – Харків, видавництво “Фоліо” 2020. – 350 с. Транслітерація: Ivan Petrov. *Fundamentals of Firebird SQL*. / Packt Publishing – Birmingham, 2018. 350 p.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для автоматизації процесів бізнес-аналітики продажів товарів у межах інформаційних систем підприємств.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Розроблення програмного забезпечення для бізнес-аналітики продажів товарів», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

A.2 Призначення розробки

Програмний продукт призначений для збору, оброблення, аналізу та візуалізації даних продажів товарів з метою підтримки прийняття управлінських рішень.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення бізнес-аналітики продажів забезпечує виконання таких функцій:

- підтримка можливості імпорту даних продажів із зовнішніх файлів (CSV, XLSX) у базу даних;
- підтримка можливості перевірки коректності структури та вмісту даних під час імпорту;
- можливість виявлення та недопущення дублювання записів у базі даних;

- підтримка можливості збереження та оброблення даних продажів у централізованій базі даних;
- можливість реєстрації та авторизації користувачів системи з розмежуванням прав доступу;
- підтримка можливості формування аналітичних показників (KPI) на основі даних продажів;
- можливість фільтрації та пошуку даних за різними параметрами (дата, категорія, регіон, канал продажу тощо);
- підтримка можливості формування та збереження аналітичних звітів;
- можливість повторного використання збережених звітів;
- підтримка можливості візуалізації даних у вигляді графіків і діаграм;
- можливість експорту результатів аналізу у формат CSV;
- підтримка ведення журналу імпорту даних для контролю якості оброблення.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення бізнес-аналітики продажів повинен бути інтуїтивно зрозумілим та зручним для користувачів різних рівнів підготовки. Повинна бути забезпечена можливість роботи у візуальному режимі з використанням вебінтерфейсу, що включає форми введення даних, таблиці, графіки та діаграми для відображення результатів аналізу.

A.3.3 Вимоги до надійності

Програмне забезпечення бізнес-аналітики продажів повинно забезпечувати стабільне та безперервне функціонування в умовах багатокористувацької роботи. Система повинна коректно обробляти помилки введення даних, забезпечувати їх

цілісність, а також гарантувати збереження інформації під час виконання операцій імпорту, оброблення та аналізу.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення бізнес-аналітики продажів необхідна наявність персонального комп'ютера або іншого пристрою з доступом до мережі Інтернет. Робота із системою здійснюється через веббраузер без необхідності встановлення додаткового програмного забезпечення на клієнтському пристрої.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи програмного забезпечення бізнес-аналітики продажів необхідні такі технічні та програмні ресурси:

- оперативна пам'ять об'ємом 1 ГБ і більше;
- щонайменше 1 ГБ вільного місця на пристрої зберігання даних;
- процесор із тактовою частотою не нижче 2 ГГц;
- встановлене серверне середовище для запуску вебзастосунку (наприклад, Python-сервер);
- система керування базами даних PostgreSQL;
- операційна система та сучасний веббраузер на клієнтському пристрої.

A.3.6 Вимоги до маркування і пакування

Програмне забезпечення бізнес-аналітики продажів може бути розповсюджене у вигляді архіву з програмним кодом або розгорнутого вебзастосунку на сервері.

На пакуванні або у супровідній документації повинна бути зазначена назва програмного продукту – «Програмне забезпечення для бізнес-аналітики продажів товарів».

A.4 Вхідні дані до роботи

Вхідними даними для програмного забезпечення бізнес-аналітики продажів є файли з даними продажів у форматах CSV або XLSX, що містять інформацію про дату продажу, товар, категорію, клієнта, регіон, канал збуту, спосіб оплати, кількість, ціну, знижку та суму операції.

Вихідними даними програмного забезпечення є структуровані записи у базі даних, аналітичні показники, сформовані звіти, графічні візуалізації (діаграми, графіки), а також експортовані файли з результатами аналізу у форматі CSV.

ДОДАТОК Б
Текст програми

Б.1 Текст файлу config/setting.py

```

from pathlib import Path
import os
from dotenv import load_dotenv
# Базова директорія проєкту
BASE_DIR = Path(__file__).resolve().parent.parent
# Завантаження змінних з .env
load_dotenv(BASE_DIR / ".env")
# Базові налаштування безпеки
SECRET_KEY = os.getenv("SECRET_KEY", "django-insecure-fallback-key")
DEBUG = os.getenv("DEBUG", "False") == "True"
ALLOWED_HOSTS = ["127.0.0.1", "localhost"]
# Встановлені застосунки
INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "django_filters",
    "users",
    "imports_data",
    "analytics_dashboard",
    "reports",
    "core",
]
# Проміжне ПЗ
MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
]
ROOT_URLCONF = "config.urls"
TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",

```

```

"DIRS": [BASE_DIR / "templates"],
"APP_DIRS": True,
"OPTIONS": {
    "context_processors": [
        "django.template.context_processors.debug",
        "django.template.context_processors.request",
        "django.contrib.auth.context_processors.auth",
        "django.contrib.messages.context_processors.messages",
    ],
},
]
WSGI_APPLICATION = "config.wsgi.application"
# Підключення до PostgreSQL
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.postgresql",
        "NAME": os.getenv("DB_NAME"),
        "USER": os.getenv("DB_USER"),
        "PASSWORD": os.getenv("DB_PASSWORD"),
        "HOST": os.getenv("DB_HOST"),
        "PORT": os.getenv("DB_PORT"),
    }
}
# Валідатори паролів
AUTH_PASSWORD_VALIDATORS = [
    {
        "NAME":
"django.contrib.auth.password_validation.UserAttributeSimilarityValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.MinimumLengthValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.CommonPasswordValidator",
    },
    {
        "NAME": "django.contrib.auth.password_validation.NumericPasswordValidator",
    },
]
# Локалізація
LANGUAGE_CODE = "uk"
TIME_ZONE = "Europe/Kyiv"

```

```

USE_I18N = True
USE_TZ = True
# Статичні файли
STATIC_URL = "static/"
STATICFILES_DIRS = [BASE_DIR / "static"]
# Медіафайли
MEDIA_URL = "/media/"
MEDIA_ROOT = BASE_DIR / "media"
# Тип первинного ключа
DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"
# Маршрути авторизації
LOGIN_URL = "login"
LOGIN_REDIRECT_URL = "dashboard_home"
LOGOUT_REDIRECT_URL = "login"
AUTH_USER_MODEL = "users.CustomUser"

```

Б.2 Текст файлу config/urls.py

```

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path("admin/", admin.site.urls),
    path("users/", include("users.urls")),
    path("imports/", include("imports_data.urls")),
    path("", include("analytics_dashboard.urls")),
]
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)

```

Б.3 Текст файлу docker-compose.yml

```

services:
  postgres:
    image: postgres:16
    container_name: sales_analytics_postgres
    restart: always

```

```

environment:
  POSTGRES_DB: sales_analytics_db
  POSTGRES_USER: sales_user
  POSTGRES_PASSWORD: sales_password
ports:
  - "5433:5432"
volumes:
  - postgres_data:/var/lib/postgresql/data
volumes:
  postgres_data:

```

Б.4 Текст файлу imports_data/models.py

```

from django.conf import settings
from django.db import models

class ImportStatus(models.TextChoices):
    SUCCESS = "SUCCESS", "Успішно"
    PARTIAL = "PARTIAL", "Частково успішно"
    FAILED = "FAILED", "Помилка"

class ImportLog(models.Model):
    file_name = models.CharField(max_length=255, verbose_name="Назва файлу")
    uploaded_by = models.CharField(max_length=150, verbose_name="Користувач, який завантажив")
    uploaded_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата і час завантаження")
    total_rows = models.PositiveIntegerField(default=0, verbose_name="Усього рядків")
    imported_rows = models.PositiveIntegerField(default=0, verbose_name="Імпортовано рядків")
    duplicate_rows = models.PositiveIntegerField(default=0, verbose_name="Дублікатів")
    invalid_rows = models.PositiveIntegerField(default=0, verbose_name="Некоректних рядків")
    status_name = models.CharField(
        max_length=20,
        choices=ImportStatus.choices,
        default=ImportStatus.SUCCESS,
        verbose_name="Статус імпорту"
    )

```

```

comment_text = models.TextField(blank=True, null=True,
verbose_name="Коментар")
user = models.ForeignKey(
    settings.AUTH_USER_MODEL,
    on_delete=models.CASCADE,
    related_name="import_logs",
    verbose_name="Користувач"
)

class Meta:
    db_table = "import_logs"
    verbose_name = "Журнал імпорту"
    verbose_name_plural = "Журнали імпорту"
    ordering = ["-uploaded_at"]

def __str__(self):
    return f"{self.file_name} - {self.get_status_name_display()}"

class SalesData(models.Model):
    sale_date = models.DateField(verbose_name="Дата продажу")
    product_name = models.CharField(max_length=255, verbose_name="Назва
товару")
    category_name = models.CharField(max_length=150, verbose_name="Категорія")
    customer_name = models.CharField(max_length=255, verbose_name="Клієнт")
    region_name = models.CharField(max_length=150, verbose_name="Регіон")
    sales_channel = models.CharField(max_length=100, verbose_name="Канал
продажу")
    payment_method = models.CharField(max_length=100, verbose_name="Спосіб
оплати")
    quantity = models.PositiveIntegerField(verbose_name="Кількість")
    unit_price = models.DecimalField(max_digits=12, decimal_places=2,
verbose_name="Ціна за одиницю")
    discount_percent = models.DecimalField(max_digits=5, decimal_places=2,
verbose_name="Знижка, %")
    total_amount = models.DecimalField(max_digits=14, decimal_places=2,
verbose_name="Підсумкова сума")
    import_batch_id = models.CharField(max_length=100,
verbose_name="Ідентифікатор завантаження")
    row_hash = models.CharField(max_length=64, unique=True, verbose_name="Хеш
рядка")
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата
імпорту")
    user = models.ForeignKey(

```

```

settings.AUTH_USER_MODEL,
on_delete=models.CASCADE,
related_name="sales_records",
verbose_name="Користувач"
)
import_log = models.ForeignKey(
    ImportLog,
    on_delete=models.SET_NULL,
    null=True,
    blank=True,
    related_name="sales_rows",
    verbose_name="Журнал імпорту"
)

class Meta:
    db_table = "sales_data"
    verbose_name = "Дані про продаж"
    verbose_name_plural = "Дані про продажі"
    ordering = ["-sale_date", "-id"]
    indexes = [
        models.Index(fields=["sale_date"]),
        models.Index(fields=["category_name"]),
        models.Index(fields=["region_name"]),
        models.Index(fields=["sales_channel"]),
        models.Index(fields=["payment_method"]),
        models.Index(fields=["product_name"]),
        models.Index(fields=["customer_name"]),
    ]

    def __str__(self):
        return f'{self.sale_date} - {self.product_name} - {self.total_amount}'

```

Б.5 Текст файлу imports_data/services.py

```

import os
import uuid
import pandas as pd
from django.conf import settings
from django.core.files.storage import FileSystemStorage

def save_uploaded_file(uploaded_file):
    uploads_dir = os.path.join(settings.MEDIA_ROOT, "uploads")

```

```

if not os.path.exists(uploads_dir):
    os.makedirs(uploads_dir)

file_extension = os.path.splitext(uploaded_file.name)[1]
unique_file_name = f"{uuid.uuid4().hex}{file_extension}"

storage = FileSystemStorage(location=uploads_dir)
saved_name = storage.save(unique_file_name, uploaded_file)

saved_path = os.path.join(uploads_dir, saved_name)
relative_path = os.path.join("uploads", saved_name)

return {
    "saved_name": saved_name,
    "saved_path": saved_path,
    "relative_path": relative_path,
    "original_name": uploaded_file.name,
}

def read_sales_file(file_path: str):
    lower_path = file_path.lower()

    if lower_path.endswith(".csv"):
        return pd.read_csv(file_path)

    if lower_path.endswith(".xlsx"):
        return pd.read_excel(file_path)

    raise ValueError("Непідтримуваний формат файлу.")

```

Б.6 Текст файлу imports_data/validators.py

```

from decimal import Decimal, InvalidOperation
import pandas as pd

REQUIRED_COLUMNS = [
    "sale_date",
    "product_name",
    "category_name",
    "customer_name",
    "region_name",

```

```

    "sales_channel",
    "payment_method",
    "quantity",
    "unit_price",
    "discount_percent",
]

def normalize_columns(df: pd.DataFrame) -> pd.DataFrame:
    df.columns = [str(column).strip().lower() for column in df.columns]
    return df

def validate_required_columns(df: pd.DataFrame):
    missing_columns = [col for col in REQUIRED_COLUMNS if col not in df.columns]
    duplicate_columns = df.columns[df.columns.duplicated()].tolist()

    return {
        "missing_columns": missing_columns,
        "duplicate_columns": duplicate_columns,
        "is_valid": len(missing_columns) == 0 and len(duplicate_columns) == 0,
    }

def parse_decimal(value, default=None):
    if pd.isna(value):
        return default

    try:
        normalized = str(value).strip().replace(",", ".")
        return Decimal(normalized)
    except (InvalidOperation, ValueError, TypeError):
        return default

def validate_single_row(row: pd.Series, row_index: int):
    errors = []

    sale_date = pd.to_datetime(row.get("sale_date"), errors="coerce")
    if pd.isna(sale_date):
        errors.append("Некоректна або порожня дата продажу.")

    product_name = str(row.get("product_name", "")).strip()
    if not product_name:
        errors.append("Не заповнено назву товару.")

    category_name = str(row.get("category_name", "")).strip()

```

```

if not category_name:
    errors.append("Не заповнено назву категорії.")

customer_name = str(row.get("customer_name", "")).strip()
if not customer_name:
    errors.append("Не заповнено назву клієнта.")

region_name = str(row.get("region_name", "")).strip()
if not region_name:
    errors.append("Не заповнено регіон.")

sales_channel = str(row.get("sales_channel", "")).strip()
if not sales_channel:
    errors.append("Не заповнено канал продажу.")

payment_method = str(row.get("payment_method", "")).strip()
if not payment_method:
    errors.append("Не заповнено спосіб оплати.")

quantity = parse_decimal(row.get("quantity"))
if quantity is None:
    errors.append("Поле quantity не є числом.")
elif quantity <= 0:
    errors.append("Поле quantity повинно бути більше нуля.")

unit_price = parse_decimal(row.get("unit_price"))
if unit_price is None:
    errors.append("Поле unit_price не є числом.")
elif unit_price < 0:
    errors.append("Поле unit_price не може бути від'ємним.")

discount_percent = parse_decimal(row.get("discount_percent"))
if discount_percent is None:
    errors.append("Поле discount_percent не є числом.")
elif discount_percent < 0 or discount_percent > 100:
    errors.append("Поле discount_percent повинно бути в межах від 0 до 100.")

total_amount = None
if quantity is not None and unit_price is not None and discount_percent is not None:
    total_amount = quantity * unit_price * (Decimal("1") - discount_percent /
Decimal("100"))
    total_amount = total_amount.quantize(Decimal("0.01"))

```

```

if total_amount < 0:
    errors.append("Підсумкова сума не може бути від'ємною.")

```

```

return {
    "row_index": row_index,
    "is_valid": len(errors) == 0,
    "errors": errors,
    "cleaned_data": {
        "sale_date": sale_date.date() if not pd.isna(sale_date) else None,
        "product_name": product_name,
        "category_name": category_name,
        "customer_name": customer_name,
        "region_name": region_name,
        "sales_channel": sales_channel,
        "payment_method": payment_method,
        "quantity": int(quantity) if quantity is not None else None,
        "unit_price": unit_price,
        "discount_percent": discount_percent,
        "total_amount": total_amount,
    }
}

```

```

def validate_dataframe_rows(df: pd.DataFrame):
    valid_rows = []
    invalid_rows = []

    for index, row in df.iterrows():
        result = validate_single_row(row, index + 2)

        if result["is_valid"]:
            valid_rows.append(result["cleaned_data"])
        else:
            invalid_rows.append({
                "row_index": result["row_index"],
                "errors": result["errors"],
            })

    return {
        "valid_rows": valid_rows,
        "invalid_rows": invalid_rows,
    }

```

Б.7 Текст файла `imports_data/hashing.py`

```
import hashlib

def normalize_text(value):
    if value is None:
        return ""
    return str(value).strip().lower()

def build_row_signature(cleaned_data: dict) -> str:
    parts = [
        normalize_text(cleaned_data.get("sale_date")),
        normalize_text(cleaned_data.get("product_name")),
        normalize_text(cleaned_data.get("category_name")),
        normalize_text(cleaned_data.get("customer_name")),
        normalize_text(cleaned_data.get("region_name")),
        normalize_text(cleaned_data.get("sales_channel")),
        normalize_text(cleaned_data.get("payment_method")),
        normalize_text(cleaned_data.get("quantity")),
        normalize_text(cleaned_data.get("unit_price")),
        normalize_text(cleaned_data.get("discount_percent")),
    ]

    normalized_row = "|".join(parts)
    return normalized_row

def calculate_row_hash(cleaned_data: dict) -> str:
    signature = build_row_signature(cleaned_data)
    return hashlib.sha256(signature.encode("utf-8")).hexdigest()
```

Б.8 Текст файла `analytics_dashboard/views.py`

```
import json
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.shortcuts import render, redirect, get_object_or_404
from .export_services import export_to_csv, export_to_excel
from imports_data.models import SalesData

from .forms import DashboardFilterForm, SaveReportForm
from .services import (
```

```

    get_dashboard_kpis,
    get_sales_trend_data,
    get_sales_by_category_data,
    get_sales_by_channel_data,
)
from reports.models import SavedReport, ReportType

@login_required
def dashboard_home_view(request):
    form = DashboardFilterForm(request.GET or None)
    save_form = SaveReportForm()

    date_from = None
    date_to = None

    if form.is_valid():
        date_from = form.cleaned_data.get("date_from")
        date_to = form.cleaned_data.get("date_to")

    kpis = get_dashboard_kpis(date_from=date_from, date_to=date_to)
    trend_data = get_sales_trend_data(date_from=date_from, date_to=date_to)
    category_data = get_sales_by_category_data(date_from=date_from, date_to=date_to)
    channel_data = get_sales_by_channel_data(date_from=date_from, date_to=date_to)

    context = {
        "page_title": "KPI-дашборд",
        "form": form,
        "save_form": save_form,
        "kpis": kpis,
        "trend_labels_json": json.dumps(trend_data["labels"], ensure_ascii=False),
        "trend_values_json": json.dumps(trend_data["values"]),
        "category_labels_json": json.dumps(category_data["labels"], ensure_ascii=False),
        "category_values_json": json.dumps(category_data["values"]),
        "channel_labels_json": json.dumps(channel_data["labels"], ensure_ascii=False),
        "channel_values_json": json.dumps(channel_data["values"]),
    }
    return render(request, "analytics_dashboard/dashboard_home.html", context)

@login_required
def save_report_view(request):
    if request.method != "POST":
        return redirect("dashboard_home")

```

```

form = SaveReportForm(request.POST)

if not form.is_valid():
    messages.error(request, "Некоректно заповнено назву звіту.")
    return redirect("dashboard_home")

filter_json = {
    "date_from": request.POST.get("date_from", ""),
    "date_to": request.POST.get("date_to", ""),
}

SavedReport.objects.create(
    report_name=form.cleaned_data["report_name"],
    report_type=ReportType.DASHBOARD,
    filter_json=filter_json,
    created_by=request.user.username,
    is_shared=False,
    user=request.user,
)

messages.success(request, "Параметри звіту успішно збережено.")
return redirect("saved_reports_list")

```

```

@login_required
def saved_reports_list_view(request):
    reports = SavedReport.objects.filter(user=request.user).order_by("-created_at")

    context = {
        "page_title": "Збережені звіти",
        "reports": reports,
    }
    return render(request, "analytics_dashboard/saved_reports_list.html", context)

```

```

@login_required
def apply_saved_report_view(request, report_id):
    report = get_object_or_404(SavedReport, id=report_id, user=request.user)
    filter_json = report.filter_json or {}
    query_params = []
    if filter_json.get("date_from"):
        query_params.append(f'date_from={filter_json["date_from"]}')

    if filter_json.get("date_to"):
        query_params.append(f'date_to={filter_json["date_to"]}')

```

```
query_string = "&".join(query_params)
```

```
if query_string:
    return redirect(f'/?{query_string}')
```

```
return redirect("dashboard_home")
```

```
@login_required
```

```
def export_sales_csv_view(request):
```

```
    qs = SalesData.objects.all()
```

```
    date_from = request.GET.get("date_from")
```

```
    date_to = request.GET.get("date_to")
```

```
    if date_from:
```

```
        qs = qs.filter(sale_date__gte=date_from)
```

```
    if date_to:
```

```
        qs = qs.filter(sale_date__lte=date_to)
```

```
    return export_to_csv(qs)
```

```
@login_required
```

```
def export_sales_excel_view(request):
```

```
    qs = SalesData.objects.all()
```

```
    date_from = request.GET.get("date_from")
```

```
    date_to = request.GET.get("date_to")
```

```
    if date_from:
```

```
        qs = qs.filter(sale_date__gte=date_from)
```

```
    if date_to:
```

```
        qs = qs.filter(sale_date__lte=date_to)
```

```
    return export_to_excel(qs)
```

Б.9 Текст файла analytics_dashboard/services.py

```
from decimal import Decimal
```

```
from django.db.models import Sum, Count, Avg
```

```

from imports_data.models import SalesData

def get_dashboard_queryset(date_from=None, date_to=None):
    qs = SalesData.objects.all()

    if date_from:
        qs = qs.filter(sale_date__gte=date_from)

    if date_to:
        qs = qs.filter(sale_date__lte=date_to)

    return qs

def get_dashboard_kpis(date_from=None, date_to=None):
    qs = get_dashboard_queryset(date_from, date_to)

    aggregates = qs.aggregate(
        total_revenue=Sum("total_amount"),
        total_sales=Count("id"),
        total_quantity=Sum("quantity"),
        average_check=Avg("total_amount"),
    )

    avg = aggregates["average_check"]

    if avg is not None:
        avg = avg.quantize(Decimal("0.01"))
    else:
        avg = Decimal("0.00")

    top_category = (
        qs.values("category_name")
        .annotate(total=Sum("total_amount"))
        .order_by("-total")
        .first()
    )

    top_region = (
        qs.values("region_name")
        .annotate(total=Sum("total_amount"))
        .order_by("-total")
        .first()
    )

```

```

top_channel = (
    qs.values("sales_channel")
    .annotate(total=Count("id"))
    .order_by("-total")
    .first()
)

result = {
    "total_revenue": aggregates["total_revenue"] or Decimal("0.00"),
    "total_sales": aggregates["total_sales"] or 0,
    "total_quantity": aggregates["total_quantity"] or 0,
    "average_check": avg,
    "top_category_name": top_category["category_name"] if top_category else
    "Немає даних",
    "top_category_value": top_category["total"] if top_category else Decimal("0.00"),
    "top_region_name": top_region["region_name"] if top_region else "Немає даних",
    "top_region_value": top_region["total"] if top_region else Decimal("0.00"),
    "top_channel_name": top_channel["sales_channel"] if top_channel else "Немає
даних",
    "top_channel_value": top_channel["total"] if top_channel else 0,
}

return result

def get_sales_trend_data(date_from=None, date_to=None):
    qs = get_dashboard_queryset(date_from, date_to)

    trend_data = (
        qs.values("sale_date")
        .annotate(total=Sum("total_amount"))
        .order_by("sale_date")
    )

    labels = [item["sale_date"].strftime("%d.%m.%Y") for item in trend_data]
    values = [float(item["total"]) for item in trend_data]

    return {
        "labels": labels,
        "values": values,
    }

def get_sales_by_category_data(date_from=None, date_to=None):
    qs = get_dashboard_queryset(date_from, date_to)

```

```

category_data = (
    qs.values("category_name")
    .annotate(total=Sum("total_amount"))
    .order_by("-total")[:10]
)

labels = [item["category_name"] for item in category_data]
values = [float(item["total"]) for item in category_data]

return {
    "labels": labels,
    "values": values,
}

def get_sales_by_channel_data(date_from=None, date_to=None):
    qs = get_dashboard_queryset(date_from, date_to)

    channel_data = (
        qs.values("sales_channel")
        .annotate(total=Sum("total_amount"))
        .order_by("-total")
    )

    labels = [item["sales_channel"] for item in channel_data]
    values = [float(item["total"]) for item in channel_data]

    return {
        "labels": labels,
        "values": values,
    }

```

Б.10 Текст файла `analytics_services/export-services.py`

```

import csv
from django.http import HttpResponse
from openpyxl import Workbook

def export_to_csv(queryset):
    response = HttpResponse(content_type="text/csv; charset=utf-8")
    response["Content-Disposition"] = 'attachment; filename="sales_export.csv"'

```

```
# Додаємо BOM для коректного відкриття UTF-8 у Excel/Windows
response.write("\u0000\u0000\u0000")
```

```
writer = csv.writer(response)
writer.writerow([
    "Дата", "Товар", "Категорія", "Клієнт",
    "Регіон", "Канал", "Оплата",
    "Кількість", "Ціна", "Знижка", "Сума"
])
```

```
for obj in queryset:
    writer.writerow([
        obj.sale_date,
        obj.product_name,
        obj.category_name,
        obj.customer_name,
        obj.region_name,
        obj.sales_channel,
        obj.payment_method,
        obj.quantity,
        obj.unit_price,
        obj.discount_percent,
        obj.total_amount,
    ])

```

```
return response
```

```
def export_to_excel(queryset):
    wb = Workbook()
    ws = wb.active
    ws.title = "Sales Data"

    headers = [
        "Дата", "Товар", "Категорія", "Клієнт",
        "Регіон", "Канал", "Оплата",
        "Кількість", "Ціна", "Знижка", "Сума"
    ]

    ws.append(headers)

    for obj in queryset:
        ws.append([
```

```

        str(obj.sale_date),
        obj.product_name,
        obj.category_name,
        obj.customer_name,
        obj.region_name,
        obj.sales_channel,
        obj.payment_method,
        obj.quantity,
        float(obj.unit_price),
        float(obj.discount_percent),
        float(obj.total_amount),
    ])

```

```

response = HttpResponse(
    content_type="application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"
)
response["Content-Disposition"] = 'attachment; filename="sales_export.xlsx"'

wb.save(response)
return response

```

Б.11 Текст файлу users/models.py

```

from django.contrib.auth.models import AbstractUser
from django.db import models

class UserRole(models.TextChoices):
    ADMIN = "ADMIN", "Адміністратор"
    ANALYST = "ANALYST", "Аналітик"

class CustomUser(AbstractUser):
    email = models.EmailField(unique=True, verbose_name="Електронна пошта")
    role_name = models.CharField(
        max_length=20,
        choices=UserRole.choices,
        default=UserRole.ANALYST,
        verbose_name="Роль"
    )
    is_active = models.BooleanField(default=True, verbose_name="Активний")
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Дата
створення")

```

```
def __str__(self):
    return f'{self.username} ({self.get_role_name_display()})'
```

Б.12 Текст файлу users/views.py

```
from django.contrib import messages
from django.contrib.auth.views import LoginView, LogoutView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.shortcuts import render, redirect
from django.views.generic import TemplateView

from .forms import LoginForm, RegisterForm

class UserLoginView(LoginView):
    template_name = "users/login.html"
    authentication_form = LoginForm
    redirect_authenticated_user = True

class UserLogoutView(LogoutView):
    next_page = "login"

class ProfileView(LoginRequiredMixin, TemplateView):
    template_name = "users/profile.html"

def register_view(request):
    if request.user.is_authenticated:
        return redirect("dashboard_home")
    if request.method == "POST":
        form = RegisterForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, "Реєстрацію успішно завершено. Тепер увійдіть до системи.")
            return redirect("login")
    else:
        form = RegisterForm()
    context = {
        "form": form
    }
    return render(request, "users/register.html", context)
```

ДОДАТОК В
Слайди презентації

РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ БІЗНЕС-АНАЛІТИКИ ПРОДАЖІВ ТОВАРІВ

Виконав:

Студент гр. КНТ-122

Мандик Олександр

Керівник:

д.т.н., професор

Олійник Андрій

1

Рисунок В.1 – Слайд 1

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

©МЕТА

Створення автоматизованої системи збору та візуалізації даних продажів для обґрунтування управлінських рішень.

Об'єкт: Процес розроблення ПЗ для аналізу бізнес-показників.

Предмет: Методи обробки, валідації та KPI-візуалізації даних.

≡Ключові задачі:

1. Аналіз джерел даних та існуючих BI-систем.

2. Проектування архітектури та бази даних.

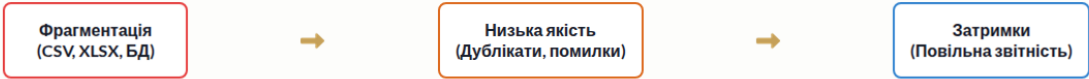
3. Реалізація модулів імпорту та візуалізації.

2

Рисунок В.2 – Слайд 2

ПРОБЛЕМАТИКА ТА АНАЛІЗ ДЖЕРЕЛ

Аналіз виявив критичні проблеми при ручній обробці даних продажів (на основі рис. 1.1):



Висновок: Необхідне впровадження ETL-процесу (Extract, Transform, Load) всередині застосунку для очищення вхідного потоку даних.

3

Рисунок В.3 – Слайд 3

ОБҐРУНТУВАННЯ РОЗРОБКИ (ПОРІВНЯЛЬНИЙ АНАЛІЗ)

Критерій	Tableau / PowerBI	Metabase (Open)	Проектне рішення
Вартість	Висока (ліцензія)	Безкоштовно	Мінімальна (Open Source)
Кастомізація	Обмежена платформою	Середня	Повна (під бізнес-процес)
Інтеграція ETL	Зовнішні модулі	Базова	Вбудована (Python/Pandas)

** Власне рішення дозволяє автоматизувати специфічні правила перевірки дублікатів через хешування рядків.*

4

Рисунок В.4 – Слайд 4

ПОРІВНЯННЯ БЕКЕНД-ФРЕЙМВОРКІВ

Критерій порівняння	Django (Вибрано)	Flask	Node.js (Express)
Швидкість розробки	Висока (Batteries included)	Середня (потребує підбору модулів)	Висока
Вбудована безпека	Максимальна (CSRF, XSS, SQLi protection)	Мінімальна (потребує налаштування)	Залежить від сторонніх пакетів
Робота з БД (ORM)	Потужний вбудований ORM	SQLAlchemy (окремо)	TypeORM / Sequelize
Адмін-панель	Вбудована	Відсутня	Відсутня

Вибір Django обумовлений необхідністю обробки критичних бізнес-даних. На відміну від Flask або Node.js, Django має вбудовані механізми захисту від SQL-ін'єкцій та CSRF-атак «з коробки». Крім того, вбудована система міграцій гарантує цілісність структури бази даних при масштабуванні проекту.

5

Рисунок В.5 – Слайд 5

ПОРІВНЯННЯ БІБЛІОТЕК ВІЗУАЛІЗАЦІЇ

Бібліотека	Chart.js (Вибрано)	D3.js	Google Charts
Складність	Низька (просте API)	Дуже висока	Середня
Продуктивність	Висока (Canvas rendering)	Екстремальна (SVG/Canvas)	Середня
Інтерактивність	Достатня для BI	Повна кастомізація	Базова

*Для візуалізації обрано Chart.js, оскільки вона використовує Canvas для рендерингу, що забезпечує високу швидкість роботи при відображенні великих масивів даних продажів (понад 10 000 записів), на відміну від SVG-рішень, які можуть сповільнювати інтерфейс.

6

Рисунок В.6 – Слайд 6

ОБҐРУНТУВАННЯ ТЕХНОЛОГІЧНОГО СТЕКУ

Python / Django

Найвища швидкість розробки та наявність бібліотек Pandas, NumPy для математичних обчислень.

PostgreSQL

Підтримка складних SQL-запитів, JSONB для зберігання гнучких налаштувань звітів та висока надійність (ACID).

Chart.js

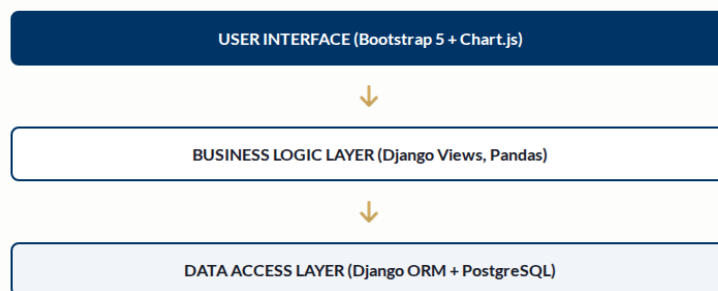
Легка інтеграція з вебінтерфейсом та підтримка інтерактивності без перевантаження сторінок.

7

Рисунок В.7 – Слайд 7

АРХІТЕКТУРА ПРОГРАМНОГО РІШЕННЯ

Використано багаторівневу архітектуру Model-View-Template (MVT):



8

Рисунок В.8 – Слайд 8

ПРОЄКТУВАННЯ СТРУКТУРИ БАЗИ ДАНИХ

ER-діаграма сутностей:

- Users: Керування доступом та ролями.
- SalesData: Центральна денормалізована таблиця (для швидкості OLAP-запитів).
- ImportLogs: Журнал аудиту завантажень.
- SavedReports: Конфігурації фільтрів користувача.

Формат зовнішніх даних (CSV/XLSX):

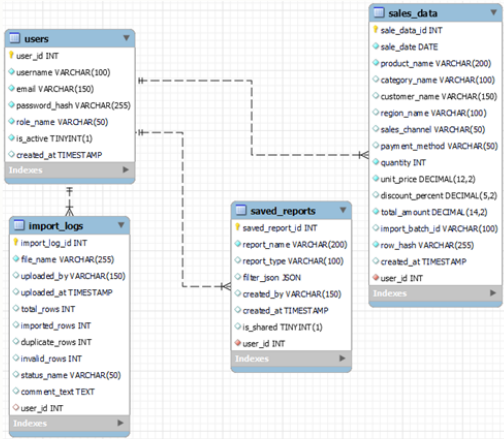
Поле	Тип
sale_date	Date
product_name	String
total_amount	Decimal

9

Рисунок В.9 – Слайд 9

Архітектура бази даних та забезпечення цілісності даних

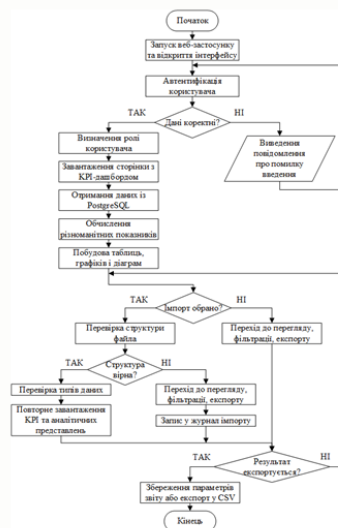
- Тип архітектури: Схема типу «Зірка» (Star Schema), оптимізована для OLAP-запитів та швидкої генерації звітів.
- Центральна сутність: Таблиця SalesData, яка містить денормалізовані дані для мінімізації операцій JOIN при великих обсягах.
- Забезпечення цілісності: Використання зовнішніх ключів для зв'язку транзакцій з журналом імпорту та користувачами.
- Оптимізація продуктивності: Індексація полів sale_date та category_name (це прискорює фільтрацію на дашборді у 5-10 разів).



10

Рисунок В.10 – Слайд 10

Алгоритм функціонування програмного забезпечення



11

Рисунок В.11 – Слайд 11

МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ АНАЛІТИКИ

Розрахунок ключових показників ефективності (KPI) реалізовано за формулами:

$$R_{\text{total}} = \sum_{i=1}^n (Q_i \cdot P_i \cdot (1 - d_i))$$

Де: Q - кількість, P - ціна одиниці, d - відсоток знижки.

$$AOV = \frac{R_{\text{total}}}{N}$$

AOV (Average Order Value) - середній чек.

12

Рисунок В.12 – Слайд 12

РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

Вхід до системи

Ім'я користувача

Введіть ім'я користувача

Пароль

Введіть пароль

Увійти

Немає облікового запису?

Зареєструватися

Реєстрація користувача

Ім'я користувача

Введіть ім'я користувача

Електронна пошта

Введіть електронну пошту

Пароль

Введіть пароль

Підтвердження пароля

Повторіть пароль

Зареєструватися

Повернутися до входу

**Форма входу та реєстрації користувача.*

13

Рисунок В.13 – Слайд 13

РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

Аналітична сторінка

Дата від

20.05.2025

Дата до

31.03.2026

Застосувати фільтр

Скинути

Назва звіту

Продажі за останній рік

Зберегти звіт

Загальна виручка

5425281,44

Кількість продажів

145

Продано одиниць

449

Середній чек

37415,73

Найприбутковіша категорія

Категорія: Smartphones

Виручка: 1877718,54

Найактивніший регіон

Регіон: Київ

Виручка: 815406,17

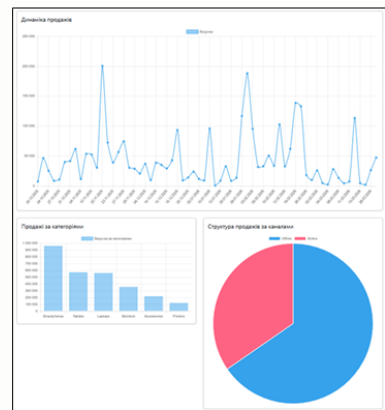
Найпоширеніший канал продажу

Канал: Office

Кількість продажів: 61

Експорт CSV

Експорт Excel



**Аналітична сторінка (Дашборд).*

14

Рисунок В.14 – Слайд 14

РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

ІМПОРТ,
ФІЛЬТРАЦІЯ,
МОНІТОРИНГ
ТА АУДИТ
ПРОЦЕСІВ

15

Рисунок В.15 – Слайд 15

РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ АДМІНІСТРАТОРА

*Панель адміністратора

Адміністрування та розмежування прав доступу

- ✓ Зручний інтерфейс для доступу до всіх модулів
- ✓ Наявність «Останні дії» для відстеження змін у системі
- ✓ Підтверджено роботу механізму обмеження доступу – адміністративні функції доступні лише авторизованим користувачам із відповідними правами.

16

Рисунок В.16 – Слайд 16

ПОРІВНЯЛЬНА ТАБЛИЦЯ ПРОДУКТИВНОСТІ

Для підтвердження ефективності архітектурних рішень було проведено навантажувальне тестування

Обсяг даних (рядків CSV)	Час імпорту та дедуплікації (сек)	Час генерації аналітики (сек)	Використання пам'яті (МБ)
1 000	0.82	0.14	42
10 000	4.15	0.45	58
50 000	14.80	1.12	115

Результати показують лінійну залежність часу обробки від обсягу даних, що підтверджує масштабованість системи завдяки оптимізованим запитам Django ORM та індексації в PostgreSQL.

17

Рисунок В.17 – Слайд 17

ТЕХНІКО-ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ

- Для підтвердження ефективності архітектурних рішень було проведено навантажувальне тестування
- Результати показують лінійну залежність часу обробки від обсягу даних, що підтверджує масштабованість системи завдяки оптимізованим запитам Django ORM та індексації в PostgreSQL.

Обсяг даних (рядків CSV)	Час імпорту та дедуплікації (сек)	Час генерації аналітики (сек)	Використання пам'яті (МБ)
1 000	0.82	0.14	42
10 000	4.15	0.45	58
50 000	14.80	1.12	115

Показник	Традиційний підхід	З використанням ПЗ
Час підготовки звіту	2-4 години	< 5 хвилин
Ймовірність помилки	Висока (людський фактор)	Мінімальна (алгоритмічна)
Оперативність рішень	Раз на тиждень	В режимі реального часу

Практична цінність: Зниження витрат на адміністрування даних на 70%.

18

Рисунок В.18 – Слайд 18

ВИСНОВКИ

1. Розроблено та впроваджено ПЗ для автоматизації бізнес-аналітики продажів.
2. Реалізовано унікальний алгоритм очищення даних, що виключає дублювання транзакцій.
3. Забезпечено інтерактивну візуалізацію KPI через веб-інтерфейс.

Результат: Мету роботи досягнуто. Система готова до експлуатації в реальних бізнес-умовах.

Рисунок В.19 – Слайд 19