ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка системи управління даними для інтернет-магазину»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

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

_____ *Максим РЕМЕЗОВСЬКИЙ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. КІД-42
Максим *РЕМЕЗОВСЬКИЙ*
Ім'я, ПРІЗВИЩЕ
Керівник: Владислав СОСНОВИЙ
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент:
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ремезовський Максим Олександрович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка системи управління даними для інтернет-магазину
керівник кваліфікаційної роботи Владислав СОСНОВИЙ,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024р. № 36
2. Строк подання кваліфікаційної роботи «3» червня 2024р.
3. Вихідні дані до кваліфікаційної роботи: _____
_____
_____
_____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Питання організації, збереження та управління даними.
2. Питання проектування системи управління даними для інтернет-магазина.
3. Питання реалізації системи управління даними для інтернет-магазина.
5. Перелік ілюстративного матеріалу: *презентація*
6. Дата видачі завдання «27» лютого 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	28.02-22.03.2024	Виконано
2	Вступ	23.03-25.03.2024	Виконано
3	Розділ 1. Аналіз предметної області	26.03-06.04.2024	Виконано
4	Розділ 2. Проектування системи управління даними для інтернет магазина	07.04-17.04.2024	Виконано
5	Розділ 3. Реалізація системи управління даними для інтернет магазина	18.04-03.05.2024	Виконано
6	Висновки за результатами аналізу	04.05-08.05.2024	Виконано
7	Розробка презентації	09.05-13.05.2024	Виконано
8	Передзахист	14.05-17.05.2024	Виконано
9	Здача в деканат	25.05-3.06.2024	Виконано

Здобувач вищої освіти _____ Максим РЕМЕЗОВСЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Владислав СОСНОВИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 40 стор., 18 рис., 7 табл., 22 джерела.

Мета роботи – розробка і впровадження бази даних та програмного забезпечення для управління даними інтернет-магазина.

Об'єкт дослідження – процес створення та впровадження бази даних та програмного забезпечення для управління даними інтернет-магазина.

Предмет дослідження – методи та засоби розробки і впровадження бази даних та програмного забезпечення для управління базою даних.

Короткий зміст роботи: У роботі було досліджено різні підходи до організації, збереження та управління даними. Розглянуто існуючі програми для управління базами даних і виявлені недоліки. Описані компоненти системи управління даними інтернет-магазина Оглянуто інструменти для розробки програмного забезпечення та розгортання бази даних. Спроектовано і розгорнуто базу даних та розроблено програмне забезпечення для її управління.

КЛЮЧОВІ СЛОВА: ДАНІ, УПРАВЛІННЯ ДАНИМИ, БАЗА ДАНИХ, СИСТЕМА УРПАВЛІННЯ БАЗАМИ ДАНИХ, SQL, MySQL.

ЗМІСТ

ВСТУП

Електронна комерція в наш час є важливим інструментом для бізнесу. Вона надає компаніям глобальний доступ до ринків, зменшує витрати на інфраструктуру та оплату персоналу, а також сприяє зручності покупок для споживачів, які можуть здійснювати їх у будь-який час та з будь-якого місця.

Електронна комерція також відкриває можливості для збору та аналізу даних про клієнтів, що дозволяє підприємствам розробляти персоналізовані пропозиції та оптимізувати стратегії маркетингу та продажів. При цьому, ефективне та безпечне управління даними є ключовим аспектом для забезпечення успіху та довіри клієнтів.

Сьогодні електронна комерція стрімко розвивається, вимагаючи від компаній постійного пошуку нових шляхів розвитку для закріплення позицій на ринку та збільшення прибутку. Зокрема, важливо постійно вдосконалювати процес обслуговування користувачів, оскільки якість обслуговування є невід'ємною складовою ефективності роботи магазинів та задоволення потреб покупців.

Наразі спостерігається швидкий розвиток інформаційних технологій і програмних рішень у сфері електронної комерції, що пояснюється розширенням можливостей компаній через підключення додаткових пристроїв, електронних ресурсів та програмного забезпечення. Сучасні інформаційні технології дозволяють максимально ефективно організовувати роботу підприємства.

В торговельному бізнесі зараз велика конкуренція, що стимулює лідерів галузі впроваджувати передові технології, зокрема системи управління базами даних та хмарні сервіси, що у поєднанні з програмним забезпечення для роботи з цією системою дозволить підвищити якість надання послуг споживачам, що забезпечить збільшення прибутку. Саме тому тема дипломної роботи допомагає вирішити проблему вдосконалення процесу управління даними для інтерне-магазину.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження і аналіз об’єкту розробки

Основною діяльністю інтернет-магазинів є продаж товару. Для якісного виконання цієї задачі, необхідно зберігати та оброблювати різноманітні дані. Наприклад:

– товарна інформація. Опис товарів, їхні характеристики (розміри, колір, матеріал тощо), фотографії та відео товарів;

– цінова інформація. Ціни на товари, знижки, пропозиції та акції;

– інформація про клієнтів. Контактні дані клієнтів (ім'я, адреса, телефон, електронна пошта), історія покупок;

– інформація про замовлення. Деталі замовлення, статуси замовлень, історія оплати та доставки;

– дані про веб-аналітику. Інформація про трафік на сайті, відвідуваність сторінок, поведінку користувачів;

– інвентаризаційна інформація. Кількість товарів на складі;

– дані про доставку. Інформація про адресу доставки, тарифи, терміни доставки, служби доставки.

Два найпоширеніших підходи до зберігання та управління даними - це використання файлової системи і систем управління базами даних (СУБД).

Файлова система - це метод збереження даних у вигляді файлів на диску. Кожен файл може містити певний обсяг інформації в будь-якому форматі, такому як текст, зображення, відео тощо. Дані зберігаються в окремих файлових структурах, і доступ до них зазвичай здійснюється через шляхи файлової системи. Переваги файлової системи:

– простота реалізації і використання;

– відповідає потребам невеликих проектів або тих, що не вимагають складного управління даними;

– можливість роботи без використання спеціалізованого програмного забезпечення;

Недоліки файлової системи:

– важкість організації та керування даними великих обсягів;

– низький рівень безпеки та ефективності порівняно з СУБД;

– обмежена можливість одночасного доступу до даних більшою кількістю користувачів.

СУБД – це набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних. Дані організовані у вигляді таблиць з рядками та стовпцями, які легко зв'язуються та опрацьовуються за допомогою SQL-запитів.

Переваги СУБД:

– високий рівень ефективності та швидкості доступу до даних;

– вбудовані механізми безпеки, резервного копіювання та відновлення даних;

– зручний інтерфейс для роботи з великими обсягами даних та складних операцій;

– широкий набір функцій для аналізу, пошуку та обробки даних;

– підтримка одночасного доступу до даних багатьма користувачами.

Недоліки СУБД:

– вимагає додаткових знань для управління даними;

– підвищена складність розгортання та налаштування.

У випадку з інтернет-магазинами використання систем управління базами даних (СУБД) є кращим вибором з кількох причин.

По-перше, інтернет-магазини зазвичай мають великий обсяг даних, таких як інформація про товари, клієнтів, замовлення тощо. СУБД забезпечують ефективне

зберігання та швидкий доступ до цих даних.

По-друге, СУБД мають вбудовані механізми безпеки, які дозволяють захистити конфіденційні дані клієнтів та іншу важливу інформацію від несанкціонованого доступу.

Крім того, СУБД забезпечують можливість використання складних запитів для аналізу даних, що дозволяє покращити процеси управління та маркетингу.

Однак варто зазначити, що взаємодія з даними в СУБД відбувається за допомогою мови запитів SQL, що може бути складним і незручним для звичайного користувача. Тому для ефективного управління базою даних інтернет-магазину необхідно мати спеціалізоване ПЗ, яке спрощує процес роботи з даними і відповідає конкретним потребам бізнесу.

1.2 Огляд існуючого ПЗ для управління базами даних

1.2.1 phpMyAdmin

phpMyAdmin — це безкоштовний програмний інструмент, написаний на PHP, призначений для адміністрування MySQL через Інтернет. phpMyAdmin підтримує широкий спектр операцій з MySQL і MariaDB. Операції, які часто використовуються (керування базами даних, таблицями, стовпцями, відношеннями, індексами, користувачами, дозволами тощо) можна виконувати через інтерфейс користувача, у той час як у вас залишається можливість безпосередньо виконувати будь-який SQL запит.

Функціонал phpMyAdmin:

- інтуїтивно зрозумілий веб-інтерфейс;
- підтримка більшості функцій MySQL:
  1) перегляд і видалення баз даних, таблиць, представлень, полів та індексів;
  2) створення, копіювання, видалення, перейменування та зміна баз даних, таблиць, полів та індексів;
  3) обслуговування сервера, бази даних і таблиць;

4) виконання і редагування будь-яких SQL-запитів;
5) керування обліковими записами та привілеями користувачів MySQL;
6) керування збереженими процедурами та тригерами;

– імпорт даних із CSV та SQL;
– експорт даних у різні формати: CSV, SQL, XML, PDF;
– адміністрування кількох серверів;
– створення графічного макета вашої бази даних у різних форматах;
– глобальний пошук у базі даних або її підмножини.

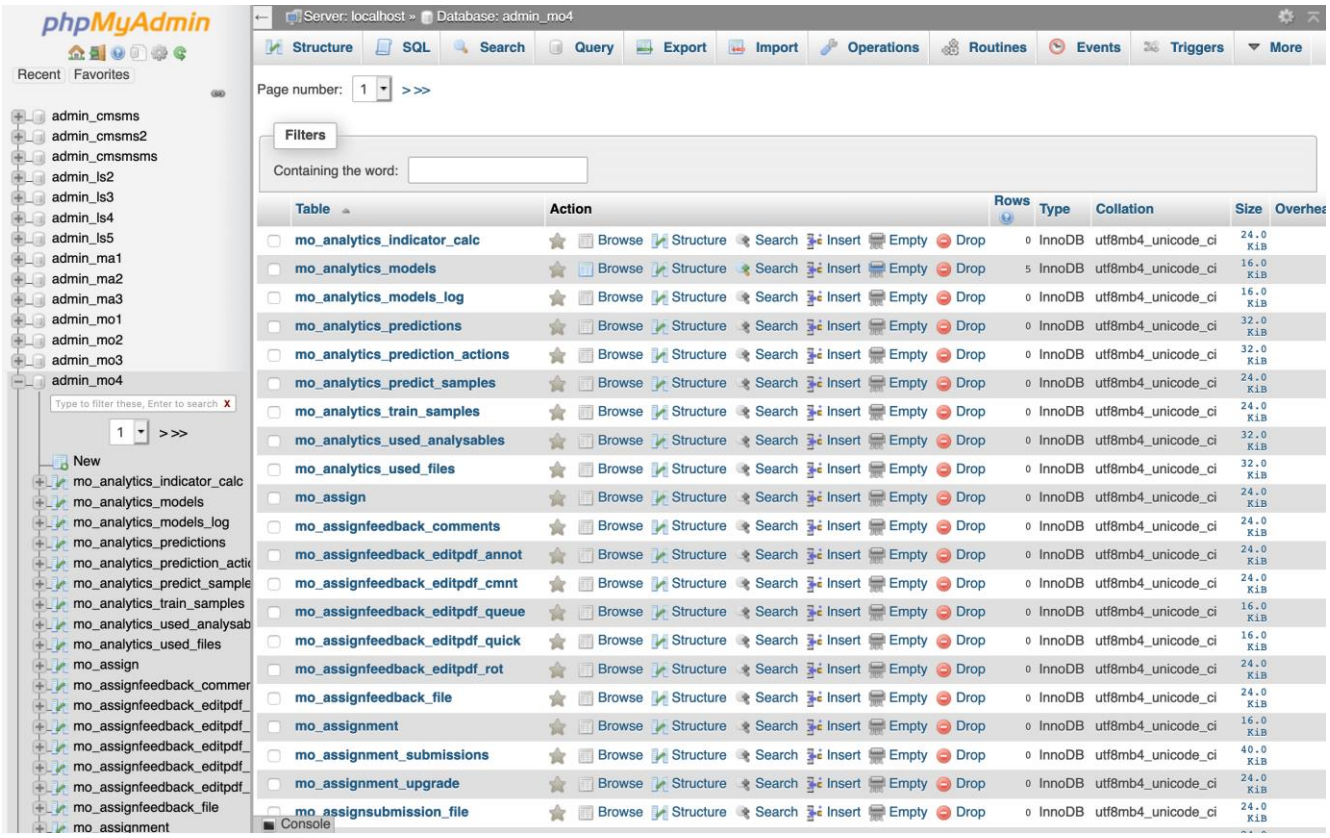


Рисунок 1.1 – Графічний інтерфейс phpMyAdmin

1.2.2 MySQL Workbench

MySQL Workbench — це візуальний інструмент для архітекторів баз даних, розробників і адміністраторів баз даних. MySQL Workbench забезпечує моделювання даних, виконання запитів SQL і комплексні засоби адміністрування для налаштування сервера, адміністрування користувачів, резервного копіювання та багато іншого. MySQL Workbench доступний у Windows, Linux і Mac OS X.

Можливості прогарми:

– дозволяє наочно представити модель бази даних в графічному вигляді;

– наочний і функціональний механізм установки зв'язків між таблицями, в тому числі "багато до багатьох" із створенням таблиці зв'язків;

– зручний редактор SQL запитів, що дозволяє відразу ж відправляти їх серверові і отримати відповідь у вигляді таблиці;

– можливість редагування даних у таблиці в візуальному режимі.

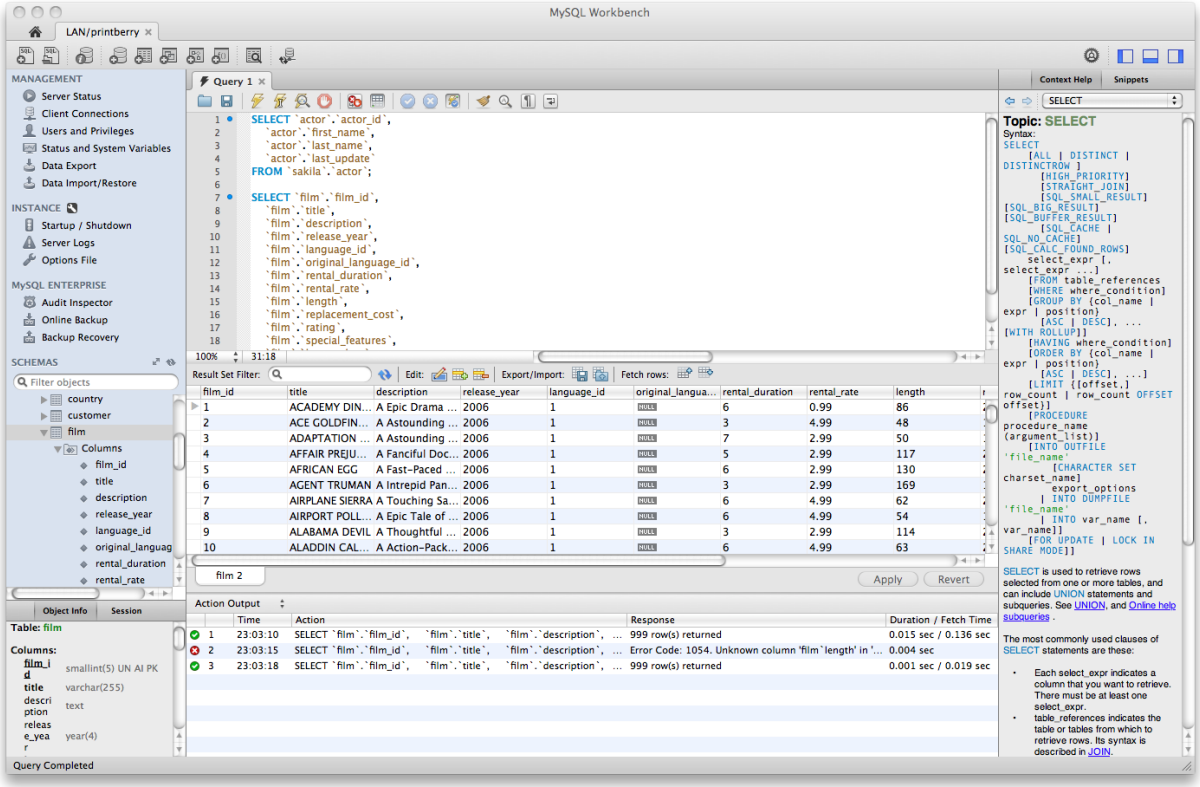


Рисунок 1.2 – Графічний інтерфейс MySQL Workbench

1.2.3 Navicat

Navicat — серія програм керування базами даних для MySQL, Oracle, SQLite, PostgreSQL, MariaDB і Microsoft SQL Server. Він підтримує кілька з'єднань з базою даних для локальних і віддалених баз. Дизайн програми виконаний з урахуванням потреб найрізноманітніших аудиторій: від адміністраторів баз даних і програмістів до різноманітних підприємств/компаній, які обслуговують клієнтів і обмінюються інформацією з партнерами.

Окрім звичайних функцій Navicat пропонує наступні можливості:

– навігація хмарною базою даних;
– редактор SQL запитів з автодоповненням і фрагментами коду;
– просунуті інструменти моделювання даних;
– графічне представлення даних.

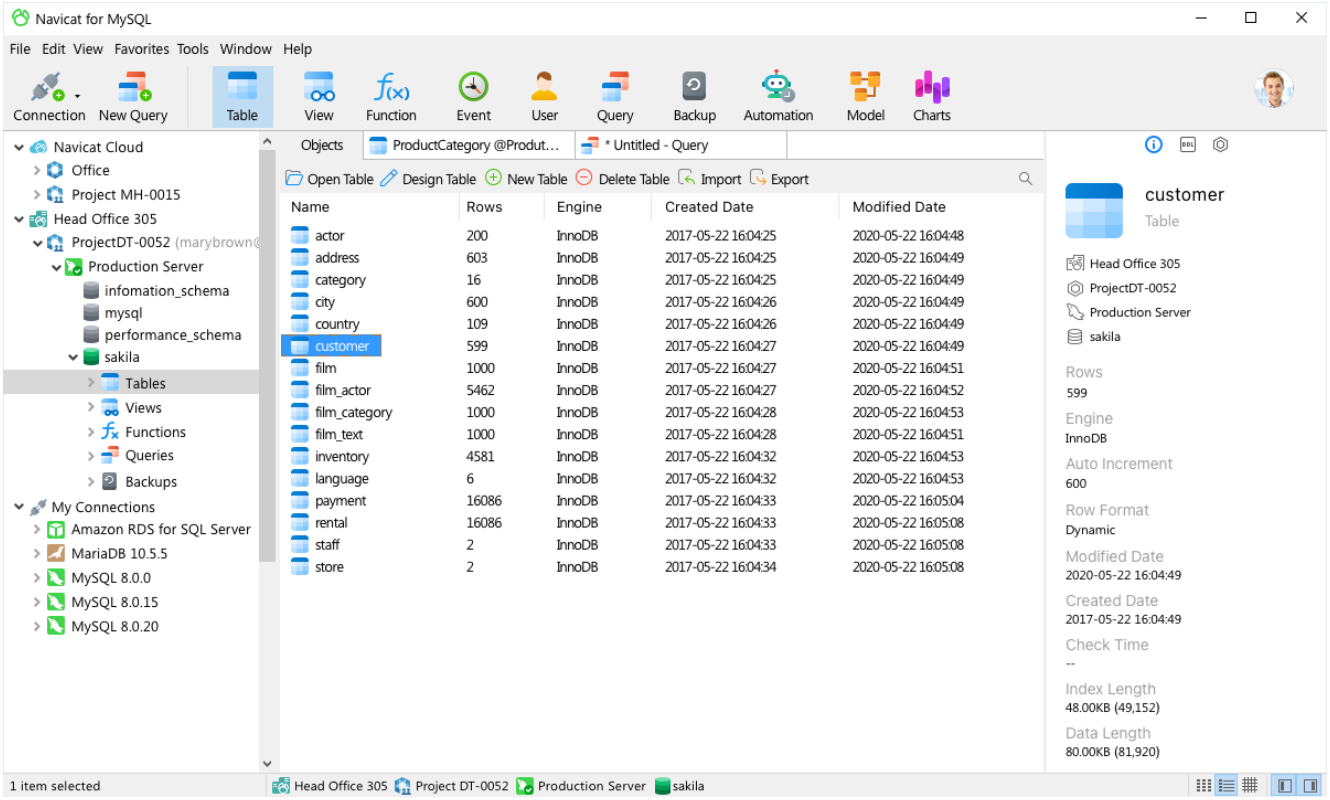


Рисунок 1.3 – Графічний інтерфейс Navicat for MySQL

1.3 Недоліки існуючого ПЗ для управління базами даних

Усі розглянуті програми націлені на вирішення завдань, пов'язаних з адмініструванням баз даних та розробкою програмного забезпечення, що робить їх дещо складними у використанні для непрофесійних користувачів, таких як адміністратор інтернет-магазину або працівник служби підтримки. Ці програми володіють великою кількістю функцій, які можуть бути надлишковими для людей, що працюють безпосередньо з даними. Більшість з них також потребує певного

рівня знань SQL, що може бути перешкодою для тих, хто не має технічної освіти або попереднього досвіду у сфері баз даних.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ УРПАВЛІННЯ ДАНИМИ ДЛЯ ІНТЕРНЕТ-МАГАЗИНА

2.1 Опис системи

Система управління даними для інтернет-магазину - це комплексне рішення, що складається з наступних компонентів:

– база даних. Сховище даних потрібне для зберігання всієї інформації про товари, замовлення, клієнтів та інші важливі дані, необхідні для функціонування інтернет-магазину;

– система управління базами даних. Набір програм який відповідає за ефективне управління цією базою даних, забезпечуючи швидкий доступ до інформації, безпеку та надійність її зберігання;

– середовище розгортання бази даних. У розгортанні програмного забезпечення середовище - це комп’ютерна система або набір систем, у яких комп’ютерна програма або компонент програмного забезпечення розгортається та виконується;

– програмне забезпечення для взаємодії з базою даних. Таке ПЗ дозволяє здійснювати наступні операції:

1) перегляд інформації про товар. Можливість швидко переглядати детальну інформацію про кожен товар, включаючи зображення, опис, характеристики, наявність на складі та інші важливі дані;
2) додавання товарів. Можливість додавати нові товари в базу даних, включаючи інформацію про назву, опис, ціну, категорію та інші характеристики;
3) редагування товарів. Можливість змінювати дані про існуючі товари;
4) видалення товарів. Можливість видаляти товари з бази даних, які більше не продаються або неактуальні;

5) керування замовленнями. Здатність переглядати, редагувати та видаляти замовлення, що надійшли від клієнтів, а також створювати нові замовлення вручну;
6) управління клієнтами. Можливість переглядати та редагувати інформацію про клієнтів, їх замовлення та історію покупок;
7) управління постачальниками. Здатність додавати, редагувати, видаляти постачальників, переглядати список, шукати та фільтрувати їх;
8) імпорт та експорт даних. Можливість завантажувати дані зовнішніми файлами (наприклад, CSV) або експортувати дані для зручного обміну інформацією з іншими системами.

2.2 Вибір засобів розробки

2.2.1 Система управління базами даних

База даних — це структурований набір даних. Це може бути що завгодно: від простого списку покупок до галереї зображень або величезних обсягів інформації в корпоративній мережі. Щоб додавати, отримувати доступ і обробляти дані, що зберігаються в комп’ютерній базі даних, потрібна система управління базами даних.

Система управління базами даних (СУБД) - це програмне забезпечення, яке дозволяє зберігати та керувати великими обсягами даних. Вона надає можливість створювати, видаляти, оновлювати та отримувати дані з бази даних.

Існують різні типи СУБД, серед яких найбільш поширені - реляційно-орієнтовані та нереляційно-орієнтовані системи. Реляційно-орієнтовані СУБД використовують реляційну модель даних для організації даних у вигляді таблиць, які мають зв'язки між собою. Доступ до даних забезпечується через запити SQL.

SQL - декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування

реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних.

Нереляційно-орієнтовані СУБД призначені для роботи з нереляційними даними, які можуть мати складну структуру або не відповідати традиційній табличній формі. Вони можуть використовувати різні моделі даних, такі як документи, графи або ключ-значення, щоб зберігати дані.

Використання реляційної СУБД для збереження і управління даними інтернет-магазину є найкращим вибором з кількох причин:

– структурованість даних. У реляційних СУБД дані структуровані у вигляді таблиць, що спрощує організацію інформації про товари, замовлення, клієнтів тощо;

– зв'язаність даних. Різні сутності пов’язані між собою, що дозволяє ефективно організовувати та обробляти дані. Наприклад, зв'язок між замовленнями і клієнтами або між товарами і категоріями;

– забезпечення цілісності даних. Реляційні СУБД можуть застосовувати обмеження цілісності, які гарантують правильність та послідовність даних. Наприклад, унікальність ідентифікаторів товарів або заборона на видалення клієнтів з активними замовленнями;

– система запитів. Реляційні бази даних підтримують мову структурованих запитів SQL, яка дозволяє виконувати складні запити до БД, використовуючи широкий спектр функцій та операцій;

Для цього проекту я обрав реляційну СУБД MySQL. MySQL є популярним вибором для використання у веб-застосунках з наступних причин.

По-перше, це система керування реляційною базою даних з відкритим вихідним кодом, що означає, що вона вільно доступна для використання та може бути налаштована відповідно до конкретних потреб.

Також MySQL відомий своєю надійністю, масштабованістю та продуктивністю, що робить його добре придатним для зберігання та пошуку даних у веб-додатках.

MySQL добре інтегрується з багатьма мовами програмування та фреймворками, які зазвичай використовуються у веб-розробці, такими як PHP, Python і JavaScript.

Крім того, його широке впровадження також означає, що розробникам, які працюють з MySQL, доступна велика кількість ресурсів і підтримки спільноти.

До основних можливостей MySQL можна віднести:

– створення та управління базами даних. MySQL дозволяє створювати нові бази даних та керувати їх структурою, включаючи таблиці, індекси, ключі, тригери та процедури;

– мова запитів SQL. MySQL підтримує стандартну мову запитів SQL для взаємодії з базою даних. Це дозволяє виконувати різноманітні операції, такі як вибірка, оновлення, видалення та вставка даних;

– безпека даних. MySQL надає механізми для забезпечення безпеки даних, включаючи аутентифікацію користувачів, авторизацію доступу до бази даних, шифрування даних та обмеження прав доступу;

– оптимізація запитів. MySQL має вбудовані засоби оптимізації запитів, які дозволяють підвищити продуктивність бази даних шляхом швидкого виконання запитів та оптимізації роботи з індексами;

– реплікація та резервне копіювання. MySQL підтримує реплікацію, що дозволяє створювати копії бази даних для забезпечення високої доступності та резервне копіювання для збереження даних в разі аварій;

– підтримка різноманітних типів даних. MySQL підтримує різноманітні типи даних, включаючи числа, рядки, дати, часи, географічні дані та бінарні об'єкти;

– клієнтські інтерфейси. MySQL надає різні інтерфейси для взаємодії з базою даних, включаючи командний рядок, графічний інтерфейс користувача та різні клієнтські бібліотеки для програмістів;

– підтримка для різних операційних систем. MySQL підтримується на багатьох операційних системах, включаючи Windows, Linux та macOS, що робить його універсальним рішенням для різних середовищ виконання.

2.2.2 Мова програмування

Для розробки ПЗ було обрано мову програмування Python. Python — це інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Високорівневі вбудовані структури даних у поєднанні з динамічною типізацією та динамічним зв’язуванням роблять її дуже привабливим для швидкої розробки додатків, а також для використання як мови сценаріїв або з’єднувальної мови для з’єднання існуючих компонентів. Простий, легкий для вивчення синтаксис Python підкреслює читабельність і, отже, знижує вартість обслуговування програми.

Python підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду. Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді безкоштовно для всіх основних платформ і можуть вільно поширюватися.

Основні характеристики та особливості мови Python:

– зрозумілий і простий синтаксис. Синтаксис Python нагадує природну мову, що робить його легким для вивчення та розуміння. Така читабельність покращує зручність обслуговування коду та співпрацю між розробниками;

– мова високого рівня. Python абстрагує складні завдання, дозволяючи розробникам зосередитися на вирішенні проблем, а не на керуванні деталями системи. Цей високорівневий характер спрощує розробку коду та підвищує продуктивність;

– динамічна типізація. Python – це мова з динамічною типізнацією, що дозволяє призначати змінні без явного визначення типів даних. Ця гнучкість спрощує написання коду та сприяє швидкій розробці;

– інтерпретована та інтерактивна мова. Python є інтерпретованою мовою, тобто код виконується рядок за рядком, що полегшує налагодження та тестування в реальному часі. Його інтерактивний характер дозволяє отримати негайний зворотний зв’язок, покращуючи процес розробки;

– об`ємна стандартна бібліотека. Python містить комплексну стандартну бібліотеку, яка надає модулі та пакети для різних завдань, від обробки файлів і роботи в мережі до обробки даних і веб-розробки. Ця багата бібліотечна екосистема прискорює розробку, усуваючи необхідність писати код з нуля;

– незалежність від платформи. Python не залежить від платформи, тобто код, написаний на Python, може легко працювати в різних операційних системах без змін. Ця портативність покращує розгортання коду та масштабованість у різних середовищах;

– підтримка кількох парадигм. Python підтримує декілька парадигм програмування, включаючи процедурне, об’єктно-орієнтоване та функціональне програмування. Розробники можуть вибрати парадигму, яка найкраще відповідає вимогам їхніх проектів, сприяючи організації коду та повторному його використанню;

– застосування у різних сферах. Python використовують у різних сферах, включаючи веб-розробку, аналіз даних, машинне навчання, штучний інтелект, наукові обчислення та автоматизацію. Його універсальність дозволяє розробникам вирішувати різноманітні завдання та створювати інноваційні рішення;

Крім того, Python має потужні бібліотеки та фреймворки для створення додатків з графічним інтерфейсом. Також, що важливо саме для цього проекту, існують високоякісні бібліотеки для роботи з MySQL, які роблять взаємодію з базою даних простою та ефективною.

2.2.3 Середовище розробки

Інтегроване середовище розробки - комплексне програмне рішення для розробки програмного забезпечення. Зазвичай, складається з редактора початкового коду, інструментів для автоматизації складання та відлагодження програм. Більшість сучасних середовищ розробки мають можливість автодоповнення коду.

Інтегровані середовища розробки створені для того, щоб максимізувати продуктивність програміста, надавши йому пов'язані інструменти розробки зі

схожими інтерфейсами як одну програму, в якій відбуватиметься весь процес розробки й яка надає необхідні функції для модифікації, компілювання, розгортання та налагодження програмного забезпечення.

Одним із завдань ICP є зменшення часу, необхідного на конфігурацію різноманітних інструментів розробки, натомість пропонуючи той самий набір, як єдине ціле. Це може збільшити продуктивність розробника, у випадку, коли навчання тому, як працює інтегроване середовище розробки є швидшим, ніж освоєння всіх інструментів зокрема. Крім того, більша інтеграція між вбудованими інструментами потенційно може сприяти додатковому збільшенню продуктивності. Наприклад, синтаксичний аналіз коду може відбуватися безпосередньо під час його редагування, тим самим виявляючи помилки ще до трансляції коду.

У зв'язку з використанням мови програмування Python для цього проекту, було обрано PyCharm як середовище розробки.

PyCharm — інтегроване середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний зневаджувач, інструмент для запуску юніт-тестів та забезпечує інтеграцію з системами контролю версій.

PyCharm працює під операційними системами Windows, Mac OS X і Linux. PyCharm має кілька версій які відрізняються функціональністю та вартістю. PyCharm Community Edition, безкоштовна версія з відкритим початковим кодом та PyCharm Professional Edition, випущена за пропрієтарною ліцензією, яка має більш обширний функіонал.

Основні можливості PyCharm:

– статичний аналіз коду, підсвічування синтаксису і помилок. Навігація серед проектів і початкового коду: відображення файлової структури проекту, швидкий перехід між файлами, класами і методами;

– рефакторинг. Перейменування, витяг методу, введення змінної, введення константи, підняття і опускання методу тощо;

– вбудований налагоджувач Python;

– підтримка систем контролю версій: загальний користувацький інтерфейс для Mercurial, Git, Subversion, Perforce і CVS з підтримкою списків змін та злиття;

– PyCharm пропонує вбудовану підтримку провідних форматів баз даних, включаючи PostgreSQL, Oracle, MongoDB і Redis, MySQL;

– підтримка не лише Python, але й JavaScript, TypeScript, HTML, CSS, SQL тощо.

2.2.4 Середовище розгортання бази даних

Для того щоб користувачі і веб-сайт мав доступ до бази даних її потрібно розгорнути. Розгортання бази даних може включати такі завдання:

– створення необхідної інфраструктури;

– конфігурація програмного забезпечення;

– забезпечення належної інтеграції бази даних в систему;

– встановлення системи управління базами даних;

– створення баз даних і таблиць;

– налаштування параметрів безпеки.

2.2.4.1 Порівняння локальних і хмарних баз даних

Існує два способи розгорнути базу даних: локально або у хмарі.

Локальні бази даних – це фізичні сервери, які обслуговуються та керуються на території підприємства.

Переваги локальних баз даних:

– продуктивність. Локальні бази даних забезпечують негайний доступ до даних, який часто може бути швидшим, у порівнянні з хмарними системами, оскільки на них не впливає затримка мережі або її перевантаження;

– контроль. Локальні бази даних також надають більше можливостей для керування даними. Хоча вони потребують більшого обслуговування, локальні бази даних забезпечують більший контроль над усією інфраструктурою;

– безпека. Локальні бази даних надають підприємствам повний доступ і контроль над своїми даними без стороннього втручання. Це також означає, що підприємства можуть захищати свої дані будь-яким способом;

– автономність. Функціонування локальних баз даних не залежить від підключення до Інтернету або постачальника хмарних послуг. Вони можуть працювати автономно на локальних серверах або комп'ютерах, незалежно від зовнішніх обставин, таких як збої мережі.

Недоліки локальних баз даних:

– ризики при відновленні даних. Коли справа доходить до аварійного відновлення та резервного копіювання даних, існує кілька ризиків для безпеки – збій обладнання, пошкодження, недбалість, тощо;

– високі початкові та поточні витрати. Початкові витрати на створення внутрішньої бази даних можуть бути дуже високими, оскільки необхідно залучити спеціалісти для налаштування сервера бази даних. Це також вимагає більше фізичного простору, що може мати наслідки для збільшення орендної плати офісу. Крім того, дорого обслуговувати локальні сервери та керувати ними, так як для цього завжди потрібні IT-фахівці. Крім того, масштабувати традиційну базу даних дорого, оскільки це вимагає від IT-фахівців додавати програмне та апаратне забезпеченн. Ця вимога буде необхідна на кожному етапі масштабування, що робить її дорогою періодичною витратою;

– складнощі масштабування. Локальні бази даних мають обмежену масштабованість і гнучкість, що ускладнює обробку величезних обсягів даних яка може бути викликана різким зростання попиту. Збільшення фізичних ресурсів або розподілення даних між кількома серверами часто є необхідним, і водночас може бути дорогим і складним.

Хмарні бази даних працюють у хмарному середовищі – інформація зберігається на різних приватних, загальнодоступних або гібридних хмарних зовнішніх серверах, доступ до яких здійснюється через Інтернет. Їх легко встановлювати та переналаштовувати, що дозволяє швидко запускати у роботу нові бази даних або підлаштовувати існуючі під поточні потреби.

Переваги хмарних баз даних:

– гнучкість у маштабуванні. Основною перевагою хмарних баз даних є їхня здатність масштабуватися відповідно до мінливих вимог бізнесу;

– надійне відновлення даних. Оскільки дані зберігаються на кількох сторонніх серверах, якими керують хмарні постачальники, безпека даних і резервне копіювання майже завжди позбавлені ризику пошкодження, або збою;

– відсутність початкових витрат. Хмарні бази даних не вимагають початкових витрат при оренді в постачальника бази даних як послуги. Оскільки вони розміщені в центрах обробки даних провайдера, витрат на апаратне забезпечення (сервери, мережеві пристрої тощо) та оренду офісних приміщень немає.

Недоліки хмарних баз даних:

– відсутність контролю. Хмарні бази даних не надають необмежений контроль над базою даних або сервером, як у випадку з локальними базами даних. Постачальники мають повний контроль і доступ до систем і серверів. Недоліком також є те, що клієнти не впливають на те, які програми чи системи використовуються, оскільки вони заздалегідь визначені та вибрані постачальниками хмарних послуг;

– ризик вразливості чутливих даних. Це пов’язано з тим, що безпека та збереження даних не в руках компаній, а в руках постачальників. Цей централізований контроль може спричинити потенційні проблеми та вразливості, оскільки організації не мають контролю над своїми цінними та конфіденційними даними;

– повільніший доступ і доставка даних через затримку або перевантаження мережі. Робота хмарних баз даних залежить від підключення до Інтернету та постачальника хмарних послуг. Це може призвести до затримки доступу до даних і доставки, особливо для програм, які вимагають даних у реальному або майже реальному часі. Користувачі потенційно можуть зіткнутися з перебоями в роботі мережі, порушеннями безпеки або збоями в роботі сервісу, що вплине на доступність їхніх даних.

2.2.4.2 Огляд Amazon RDS

Враховуючи усі переваги і недоліки для реалізації мого проекту я буду використовувати хмарну базу даних, а саме сервіс реляційних баз даних Amazon RDS від Amazon Web Services.

Amazon Web Services – це найпоширеніша у світі хмара з найширшими можливостями, що надає понад 200 повнофункціональних сервісів для центрів обробки даних у всьому світі. AWS надає більш ніж 70 сервісів, що охоплюють широкий спектр, включаючи обчислення та зберігання даних, їхню передачу по мережі, аналітику, мобільні застосунки, інструменти для розробників і т. д. Amazon рекламує AWS як спосіб отримання обчислювальної потужності що масштабується швидше та дешевше, ніж побудова власного фізичного серверного кластеру. Усі послуги оплачуються залежно від використання, однак кожна служба вимірює використання своїм методом.

Amazon Relational Database Service – це сервіс реляційних баз даних від Amazon Web Services. Це серверне програмне забезпечення, яке виконується у «хмарі», клієнту надається доступ до всіх можливостей використання бази даних як створення баз, таблиць та роботи з ними. Провайдер забезпечує послуги системного адміністрування як розгортання, налаштування, масштабування, резервного копіювання. Управління такими адміністративними діями як зміна дискового простору, ресурсів для обчислення, запити на налаштування резервного копіювання та відновлення резервної копії виконується через вебінтерфейс або через API AWS.

Особливості Amazon RDS:

– управління завданнями з адміністрування бази даних. Одна з ключових переваг сервісу Amazon RDS полягає в тому, що він виконує завдання управління базами даних, такі як виділення ресурсів, резервне копіювання, відновлення, виявлення збоїв та виправлення.

– Amazon RDS дозволяє працювати з більшістю систем управління базами даних: PostgreSQL, MySQL, MariaDB, SQL Server, Oracle. Db2;

– моніторинг та перегляд метрик бази даних. У сервісі Amazon RDS можна отримати доступ до метриків бази даних без додаткової плати. За допомогою консолі Amazon RDS можна переглядати основні робочі метрики, включаючи використання обчислювальних ресурсів, пам'яті та сховища, інтенсивність операцій введення-виведення та звернення до бази даних. Крім цього, Amazon RDS пропонує покращений моніторинг, який забезпечує доступ більш ніж до 50 метриків використання процесора, пам'яті, файлової системи та жорсткого диска;

– відновлення на момент часу за допомогою автоматичного резервного копіювання. Функція автоматичного резервного копіювання Amazon RDS дозволяє відновлювати базу даних на певний час. Amazon RDS виконує резервне копіювання бази даних та журналів транзакцій та зберігає їх протягом зазначеного користувачем періоду. Це дозволяє відновити БД на будь-який момент терміну зберігання (з точністю до секунди). Період зберігання автоматично створених резервних копій може становити до 35 днів;

– створення резервних копій знімків стану бази даних з власної ініціативи. Знімки стану бази даних - це ініційовані користувачем резервні копії БД, що зберігаються до моменту їхнього навмисного видалення користувачем. Зі знімку стану бази даних можна у будь-який момент створити нову БД;

– розширення можливості своєї бази даних за рахунок простого масштабування обчислювальних функцій. Amazon RDS дозволяє масштабувати обчислювальні ресурси та ресурси пам'яті, що забезпечують роботу системи, зменшуючи або збільшуючи їх обсяг до максимально можливих 128 віртуальних ЦПУ та 4096 ГіБ оперативної пам'яті. Масштабування обчислювальних ресурсів, зазвичай, займає лише кілька хвилин.

Для розгортання бази даних за допомогою Amazon RDS потрібно створити екземпляр бази даних Amazon RDS. Екземпляр БД - це ізольоване середовище бази даних, що працює в хмарі. Це основний будівельний блок Amazon RDS. Екземпляр БД може містити кілька баз даних, створених користувачами, і доступ до нього можна отримати за допомогою клієнтських інструментів і програм.

При створенні нового екземпляру бази даних Amazon RDS потрібно визначити наступні параметри:

– тип двигуна. Amazon RDS пропонує на вибір вісім популярних ядер баз даних: версію Amazon Aurora, сумісну з PostgreSQL, версію Amazon Aurora, сумісну з MySQL, RDS для PostgreSQL, RDS для MySQL , RDS для MariaDB, RDS для SQL Server, RDS для Oracle та RDS для Db2;

– версія двигуна. Для кожного двигуна існує декілька версій, тому важливо обрати саме ту, яка є відповідає вимогам системи з якою СУБД буде взаємодіяти;

– варіант розгортання. Amazon RDS надає три варіанта розгортання:

1) кластер БД з декількома зонами доступності. Створює кластер БД із трьома екземплярами БД. Кожен екземпляр БД знаходиться в іншій зоні доступності. Кластер БД Multi-AZ має один основний екземпляр БД і два доступні для читання резервні екземпляри БД. Використання кластера БД Multi-AZ забезпечує високу доступність, підвищену ємність для робочих навантажень читання та меншу затримку;
2) екземпляр БД з декількома зонами доступності. Створює основний екземпляр БД з одним резервним екземпляром БД в іншій зоні доступності. Використання екземпляра БД з декількома зонами доступності забезпечує високу доступність, але екземпляр резервної БД не підтримує з’єднання для читання;
3) один екземпляр БД. Створює один екземпляр БД без резервних екземплярів;

– ідентифікатор екземпляра/кластера БД. Унікальне ім’я екземпляра/кластера;

– головне ім'я користувача. Ідентифікатор входу для головного користувача. Ім’я головного користувача використовується щоб почати визначення всіх користувачів, об’єктів і дозволів у базах даних екземпляра БД;

– керування обліковими даними. Можна обрати з двох варіантів:

1) керування за допомогою AWS Secrets Manager. RDS генерує для вас пароль і керує ним протягом життєвого циклу за допомогою AWS Secrets Manager;
2) самокерування. Створення власного паролю;

– головний пароль. Пароль для головного користувача;

– підтвердження головного паролю. Повторення значення, яке вказано для головного пароля;

– клас екземпляра БД. Клас екземпляра БД визначає обчислення та обсяг пам’яті екземпляра БД Amazon RDS. Можна обрати серед трьох типів класів:

1) стандартний. Стандартні екземпляри забезпечують баланс обчислювальних ресурсів, пам’яті та мережевих ресурсів. Вони є хорошим вибором для більшості робочих навантажень бази даних;
2) класи з оптимізацією пам'яті. Оптимізовані для пам’яті екземпляри підвищують продуктивність для робочих навантажень, які обробляють великі набори даних у пам’яті;
3) класи з оптимізацією обчислень. Оптимізовані для обчислень екземпляри дозволяють зберігати тимчасові табличні простори у сховищі екземпляра а також ви підвищують продуктивність і зменшують навантаження на БД;

– тип сховища. Amazon RDS пропонує три типи сховища:

1) SSD загального призначення. SSD загального призначення пропонує економічне сховище, яке ідеально підходить для широкого діапазону робочих навантажень, що виконуються на екземплярах БД середнього розміру. Сховище загального призначення найкраще підходить для середовищ розробки та тестування;
2) SSD забезпечене IOPS (англ. Provisioned IOPS SSD) – сховище забезпечене IOPS розроблено для задоволення потреб інтенсивних навантажень вводу-виводу, зокрема робочих навантажень баз даних, які потребують низької затримки вводу-виводу та постійної пропускної

здатності вводу-виводу. Надане сховище IOPS найкраще підходить для виробничих середовищ;

3) магнітне. Amazon RDS також підтримує магнітне сховище для зворотної сумісності. Рекомендується використовувати SSD загального призначення або SSD забезпечене IOPS для будь-яких нових потреб у сховищі. Максимальний обсяг пам’яті, дозволений для примірників БД на магнітній пам’яті, менший, ніж для інших типів пам’яті;

– виділена пам'ять. Розмір виділеного сховища Кількість операцій виводу-виводу в секунду;

– кількість операцій введення-виведення за секунду, яку може підтримувати екземпляр БД;

– пропускна здатність сховища. Швидкість, з якою виконується читання та запис;

– автоматичне масштабування сховища. Ця функція дозволяє збільшити обсяг пам’яті після перевищення вказаного порогу;

– максимальний поріг зберігання. Об’єм сховище після якого автоматичне масштабування сховища припиниться;

– віртуальна приватна хмара (англ. VPC). За допомогою віртуальної приватної хмари можна запускати ресурси AWS у логічно ізольованій віртуальній мережі, яку ви визначили. Ця віртуальна мережа дуже нагадує традиційну мережу але з перевагами використання масштабованої інфраструктури AWS. Група підмережі БД;

– група підмереж БД визначає, які підмережі у віртуальній приватній хмарі призначено для бази даних;

– загальний доступ. Якщо обрати «Так», то інші ресурси за межами VPC, на якому розміщена база даних, будуть мати можливість підключатися до БД. Якщо вибрати «Ні», Amazon RDS не призначатиме загальнодоступну IP-адресу базі даних. У цьому випадку жодні ресурси за межами VPC не зможуть підключитися до нього без додаткової конфігурації;

– група безпеки VPC (брандмауер). Група безпеки діє як віртуальний брандмауер, контролюючи трафік, якому дозволено досягати та залишати ресурси, з якими пов’язана група безпеки. Група безпеки, пов’язана з базою даних, контролює вхідний і вихідний трафік для бази даних;

– зона доступності. Зони доступності покращують високу доступність, ізолюючи збої від інших зон доступності, підтримуючи підключення з низькою затримкою в регіоні;

– центр сертифікації. Використання сертифіката сервера забезпечує додатковий рівень безпеки, підтверджуючи, що підключення здійснюється до бази даних Amazon. Це робиться шляхом перевірки сертифіката сервера, який автоматично встановлюється на всіх базах даних. Порт бази даних;

– порт TCP/IP, який екземпляр БД використовуватиме для підключення програми. Рядок підключення будь-якої програми, що підключається до екземпляра БД, повинен вказувати номер порту екземпляра БД. Група безпеки, застосована до екземпляра БД повинна дозволяти підключення до порту;

– статистика ефективності (англ. Performance Insights). Performance Insights — це вдосконалена функція моніторингу продуктивності бази даних, яка спрощує діагностику та вирішення проблем продуктивності баз даних Amazon RDS;

– термін зберігання моніторингових даних. Визначає розмір вікна історії даних які зберігаються;

– автоматичне резервне копіювання. Створює знімок бази даних на певний момент часу;

– термін зберігання резервної копії. Кількість днів, протягом яких Amazon RDS має зберігати автоматичні резервні копії цього екземпляра БД. Термін зберігання резервної копії визначає період, протягом якого ви можете виконати відновлення на певний момент часу;

– вікно резервного копіювання. Щоденний часовий діапазон, протягом якого створюються автоматичні резервні копії;

– автоматичне оновлення проміжної версії. Автоматичне оновлення до нових проміжних версій у міру їх випуску. Автоматичні оновлення відбуваються під час періоду обслуговування екземпляра БД;

– вікно обслуговування. Період, протягом якого потрібно внести незавершені модифікації (наприклад, змінити клас екземпляра БД) або виправлення, застосовані до екземпляра БД за допомогою Amazon RDS. Будь-яке таке обслуговування має бути розпочато та завершено протягом вибраного періоду;

– захист від видалення. Захищає базу даних від випадкового видалення. Поки цей параметр увімкнено, ви не можете видалити базу даних.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ДАНИМИ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ

3.1 Проектування бази даних

Першим кроком для розробки бази даних є створення таблиць. Кожна таблиця повинна уособлювати сутність дані якої вона зберігає. База даних для інтернет-магазину буде містити наступні таблиці:

- товари (англ. products);
- постачальники (англ. suppliers);
- клієнти (англ. clients);
- замовлення (англ. orders).

Далі необхідно визначити поля таблиць. Кожне поле має свою назву та тип даних, який визначає формат даних, що можуть бути збережені в цьому полі. Крім того, поля можуть мати додаткові властивості, такі як унікальність, обов'язковість, індексація та інші, які дозволяють точніше визначити специфікації та обмеження для даних, які зберігаються в цьому полі. Також потрібно призначити полям таблиць первинні і зовнішні ключі. Первинний ключ – це поле, яке однозначно ідентифікує кожен рядок у таблиці. Зовнішній ключ - це поле у а таблиці, яке посилається на первинний ключ в іншій таблиці.

Таблиця 3.1 – Специфікація таблиці products

Назва	Тип даних	Опис
product_id	int	Ідентифікатор товару (ПК)
name	varchar	Назва товару
description	varchar	Опис товару
price	decimal	Ціна товару

Продовження таблиці 3.1

category	Varchar	Категорія товару
supplier_id	int	Ідентифікатор постачальника (ЗК)
discount	int	Знижка на товар
photo	varchar	Фото товару
quantity	int	Кількість товару на складі

Таблиця 3.2 – Специфікація таблиці clients

Назва	Тип даних	Опис
client_id	int	Ідентифікатор клієнта (ПК)
first_name	varchar	Ім’я клієнта
middle_name	varchar	По батькові клієнта
last_name	varchar	Прізвище клієнта
email	varchar	Електронна пошта клієнта
password	varchar	Пароль клієнта
phone	int	Номер телефону клієнта

Таблиця 3.3 – Специфікація таблиці orders

Назва	Тип даних	Опис
order_id	int	Ідентифікатор замовлення
client_id	int	Ідентифікатор клієнта (ЗК)
status	varchar	Статус замовлення
order_date	date	Дата замовлення

Продовження таблиці 3.3

shipped_date	date	Дата відправки замовлення
product_id	int	Ідентифікатор товару (ЗК)
quantity	int	Кількість замовленого товару
address	varchar	Адреса для доставки замовлення
city	varchar	Місто для доставки замовлення
country	varchar	Країна для доставки замовлення

Таблиця 3.4 – Специфікація таблиці suppliers

Назва	Тип даних	Опис
supplier_id	int	Ідентифікатор постачальника (ПК)
name	int	Ім’я постачальника
phone	int	Номер телефону постачальника
address	varchar	Адреса постачальника
city	varchar	Місто постачальника
country	varchar	Країна постачальника

Наступним кроком є нормалізація бази даних. Нормалізація — це формальний процес, який можна використовувати для розділення даних у структурі даних на пов’язані таблиці. Нормалізація зменшує надмірність даних, що може спричинити проблеми зберігання та обслуговування. У ненормалізованій структурі даних таблиця може містити інформацію про дві або більше сутностей.

Також може містити повторювані стовпці, стовпці, які містять повторювані значення та дані, які повторюються в двох або більше рядках.

У нашій базі даних лише таблиця «Замовлення» може містити повторювані значення, оскільки один клієнт може замовити різні товари і отже для одного замовлення буде декілька записів, які будуть відрізнятися лише значеннями product_id і quantity. Для того щоб вирішити цю проблему потрібно розбити таблицю «Замовлення» на дві таблиці – «Замовлення» і «Деталі Замовлення» (англ. order details).

Таблиця 3.5 – Специфікація нормалізованої таблиці orders

Назва	Тип даних	Опис
order_id	int	Ідентифікатор замовлення (ПК)
client_id	int	Ідентифікатор клієнта (ЗК)
status	varchar	Статус замовлення
order_date	date	Дата замовлення
shipped_date	date	Дата відправки замовлення
address	varchar	Адреса для доставки замовлення
city	varchar	Місто для доставки замовлення
country	varchar	Країна для доставки замовлення

Таблиця 3.6 – Специфікація таблиці order details

Назва	Тип даних	Опис
order_id	int	Ідентифікатор товару (ЗК)
product_id	int	Ідентифікатор товару (ЗК)
quantity	int	Кількість замовленого товару

Останнє, що потрібно зробити це визначити зв’язки між таблицями. Зв'язки визначаються за допомогою зовнішніх ключів, які вказують на ключ або ключі в іншій таблиці, з якою вони пов'язані.

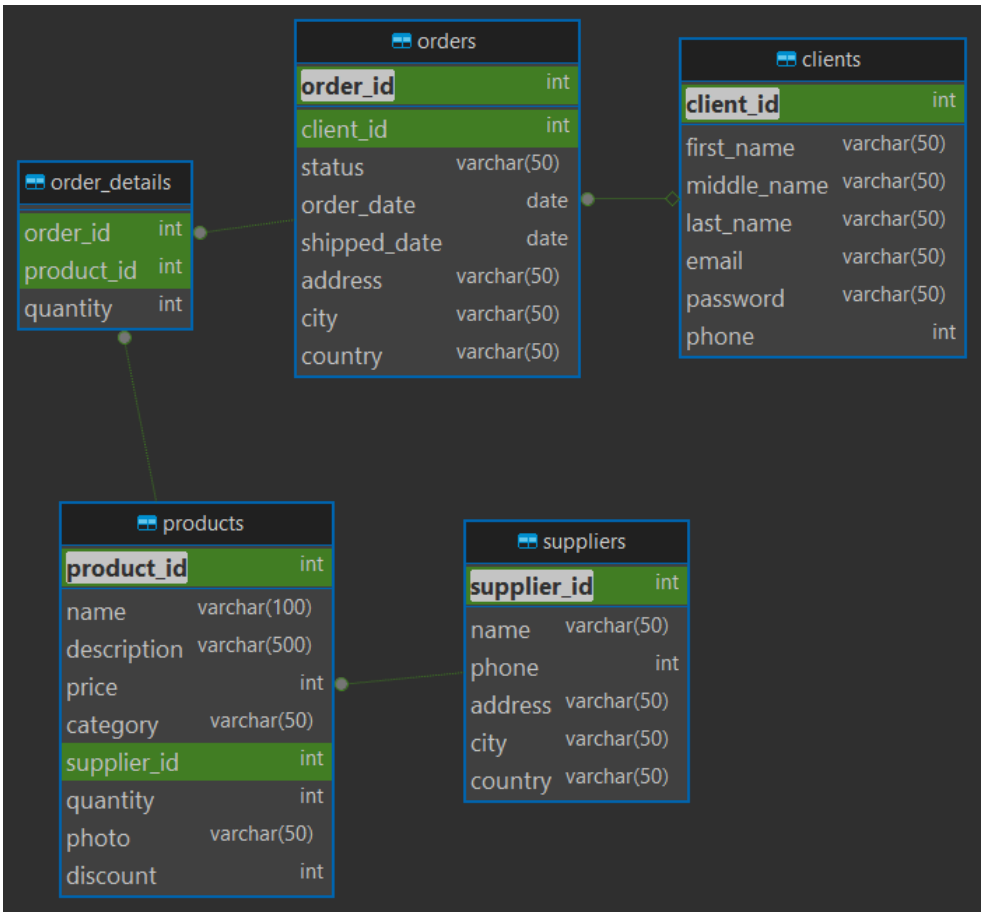


Рисунок 3.1 – Схема бази даних

3.2 Розгортання бази даних

Спочатку потрібно створити екземпляр бази даних Amazon RDS, визначивши потрібну конфігурацію.

Таблиця 3.7 – Конфігурація екземпляра БД Amazon RDS

Параметр	Значення
Тип двигуна	MySQL
Видання	MySQL Community
Версія двигуна	MySQL 8.0.35
Варіант розгортання	Один екземпляр БД
Ідентифікатор екземпляра БД	online-store-db
Ім'я головного користувача	admin
Керування обліковими даними	Самокерування
Клас екземпляра БД	db.t3.micro
Тип сховища	SSD загального призначення (gp2)
Розмір виділеного сховища	20 ГіБ
Увімкнути автомасштабування сховища	Так
Максимальний поріг зберігання	30 ГіБ
Обчислювальний ресурс	Не підключайтеся до обчислювального ресурсу EC2
Віртуальна приватна хмара (VPC)	Default VPC
Група підмережі БД	Default
Загальний доступ	Так
Група безпеки VPC (брандмауер)	default
Зона доступності	Без уподобань
Створити проксі RDS	Ні

Продовження таблиці 3.7

Центр сертифікації	default
Порт БД	3306
Спосіб автентифікації	Автентифікація за допомогою паролів бази даних
Увімкнути розширений моніторинг	Так
Увімкнути автоматичне резервне копіювання	Так
Термін зберігання резервної копії	7 днів
Вікно резервного копіювання	Время початку: 03:00 Тривалість: 3 години
Увімкнути шифрування	Так
Увімкнути автоматичне оновлення проміжної версії	Ні
Вікно обслуговування	День початку: понеділок Время початку: 01:00 Тривалість: 1.5 години
Увімкнути захист від видалення	Так

Далі потрібно налаштувати групу безпеки VPC. За замовчуванням існує тільки одне вхідне правило, яке дозволяє підключитися до сервера MySQL тільки з екземплярів які асоційовані з тією ж групую безпеки що і екземпляр БД. Для того щоб дозволити доступ до сервера з ком'ютера, потрібно створити нове вхідне правило, вказавши тип правила, протокол, порт і IP адресу комп'ютера.

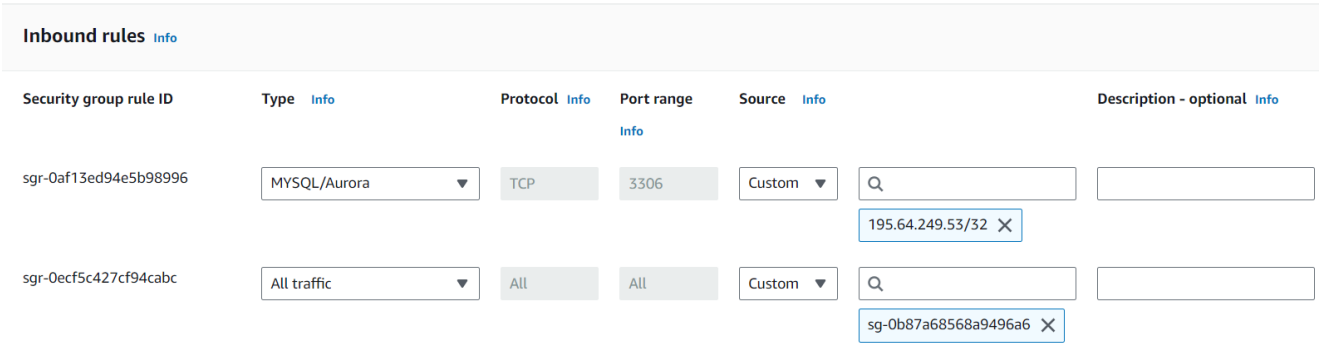


Рисунок 3.2 – Вхідні правила для групи безпеки асоційованою з екземпляром БД

Останнім кроком буде створення бази даних і таблиць. Для цього потрібно підключитися до сервера MySQL за допомогою клієнтського ПЗ, використавши назву кінцевої точки і облікові дані головного користувача.

```
C:\Users\maksr>mysql -u admin -p -h online-store-db.chouow8247vw.eu-north-1.rds.amazonaws.com
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 77
Server version: 8.0.35 Source distribution

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Рисунок 3.3 – Підключення до бази даних за допомогою інструмента командного рядка mysql

Після підключення потрібно виконати запити на мові SQL для створення бази даних і таблиць. Усі запити для створення бази даних і таблиць наведені у додатку А.

3.3 Опис інтерфейсу та функціоналу програми

При запуску програми користувачу необхідно підключитися до бази даних, ввівши назву хоста, ім’я користувача і пароль.

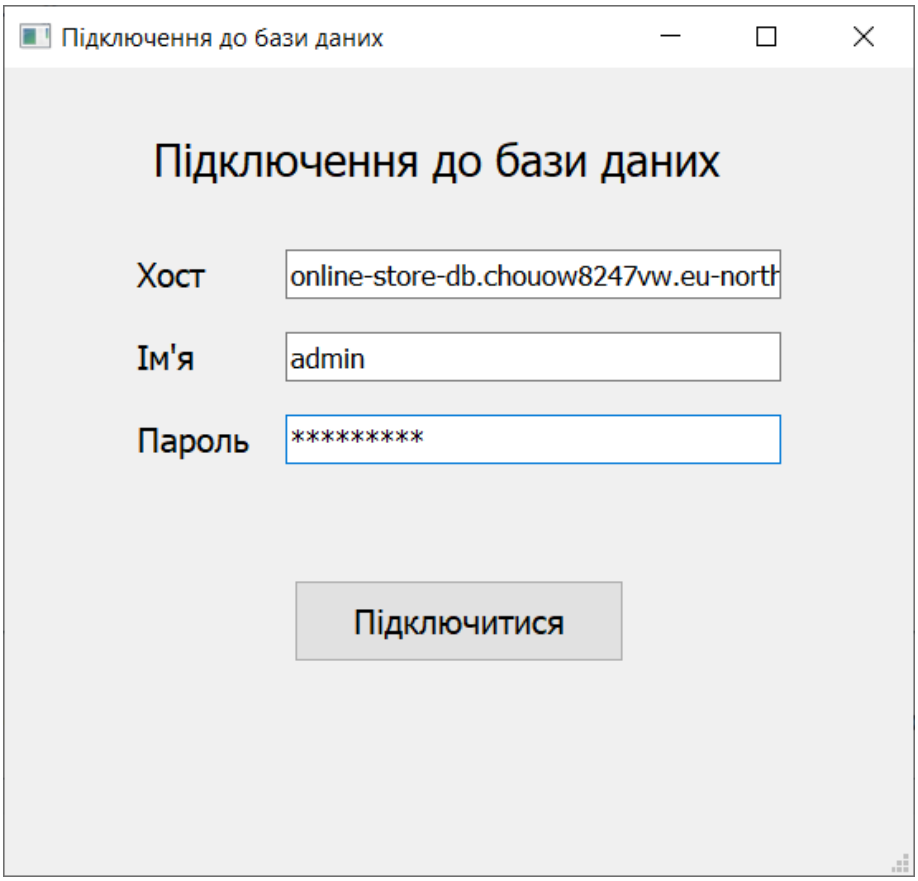


Рисунок 3.4 – Вікно «Підключення до бази даних»

Після введення необхідних даних і натискання на кнопку «Підключитися», програма намагається підключитися до бази даних. Якщо якісь з полів були заповнені неправильно, з’явиться вікно яке повідомить про помилку.

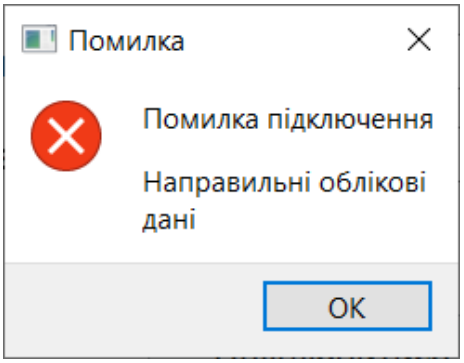


Рисунок 3.5 – Вікно «Помилка»

У разі вдалого підключення відкриється головне вікно програми.

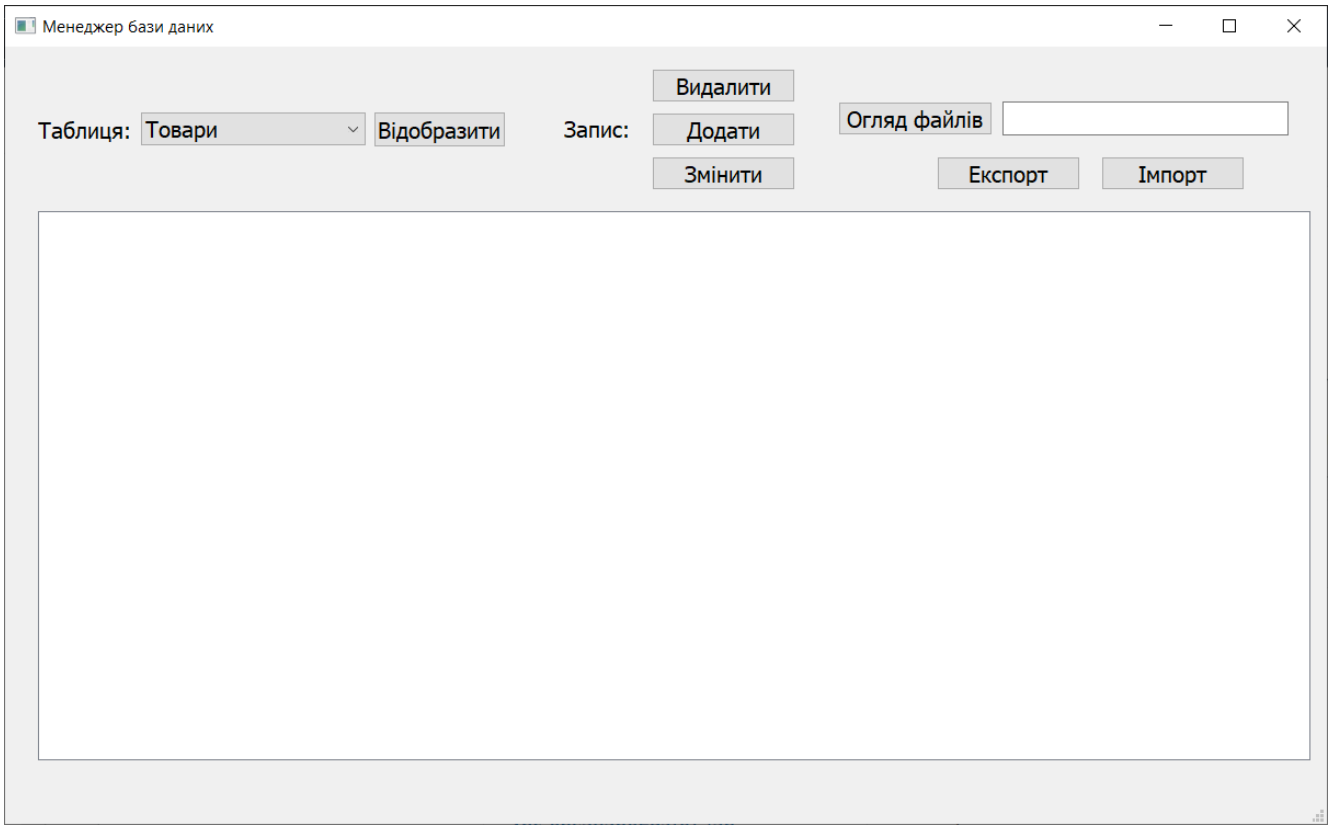


Рисунок 3.6 – Головне вікно програми

На головному вікні у верхній частині розташовані елементи інтерфейсу необхідні для взаємодії з базою даних. Обравши таблицю зі списку і натиснувши на кнопку «Відобразити» з’явиться відповідна таблиця.

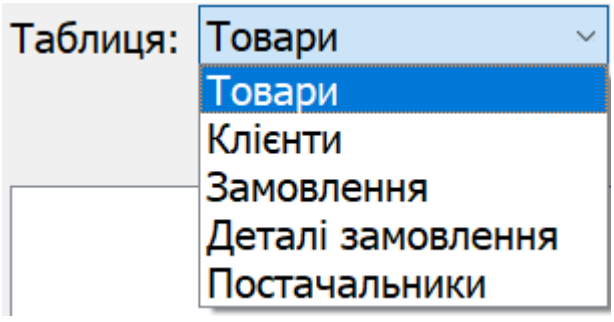
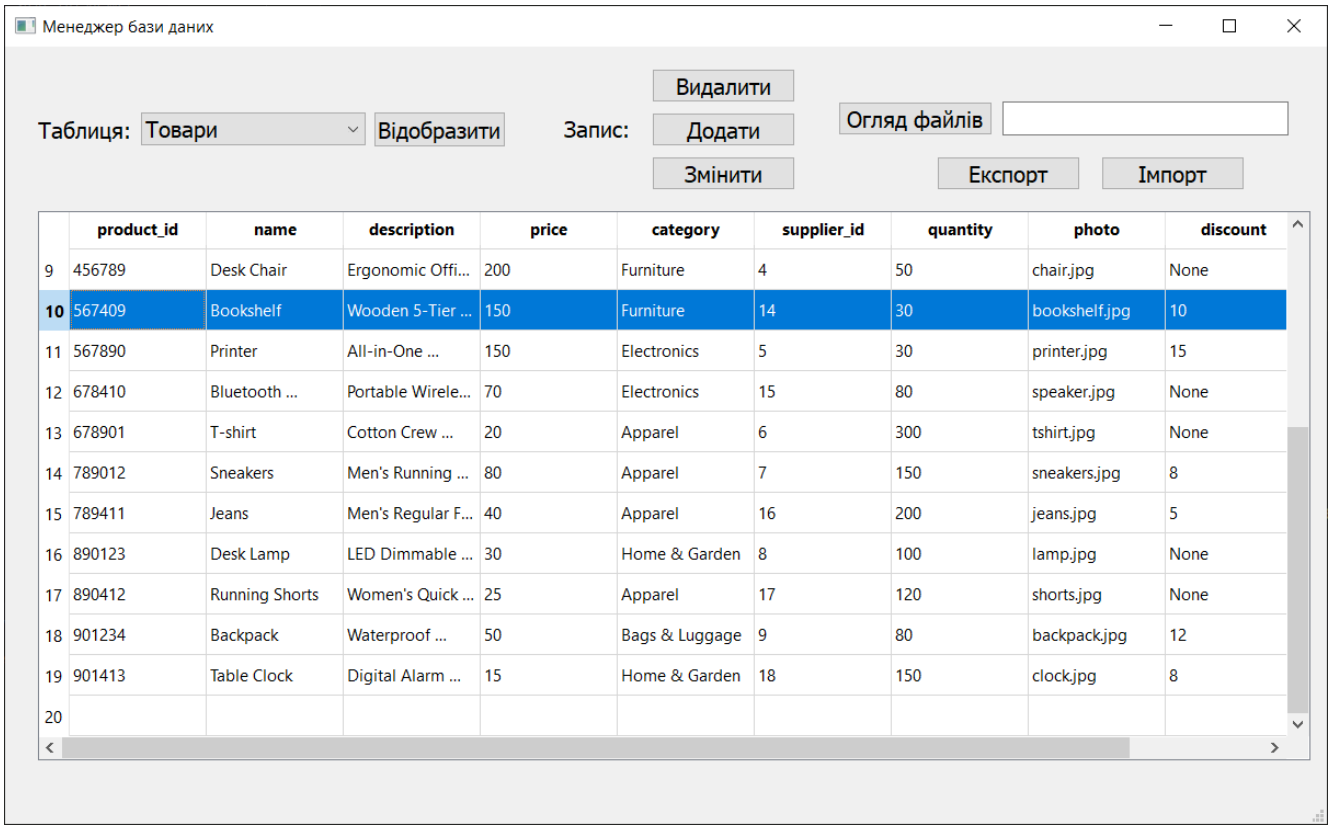


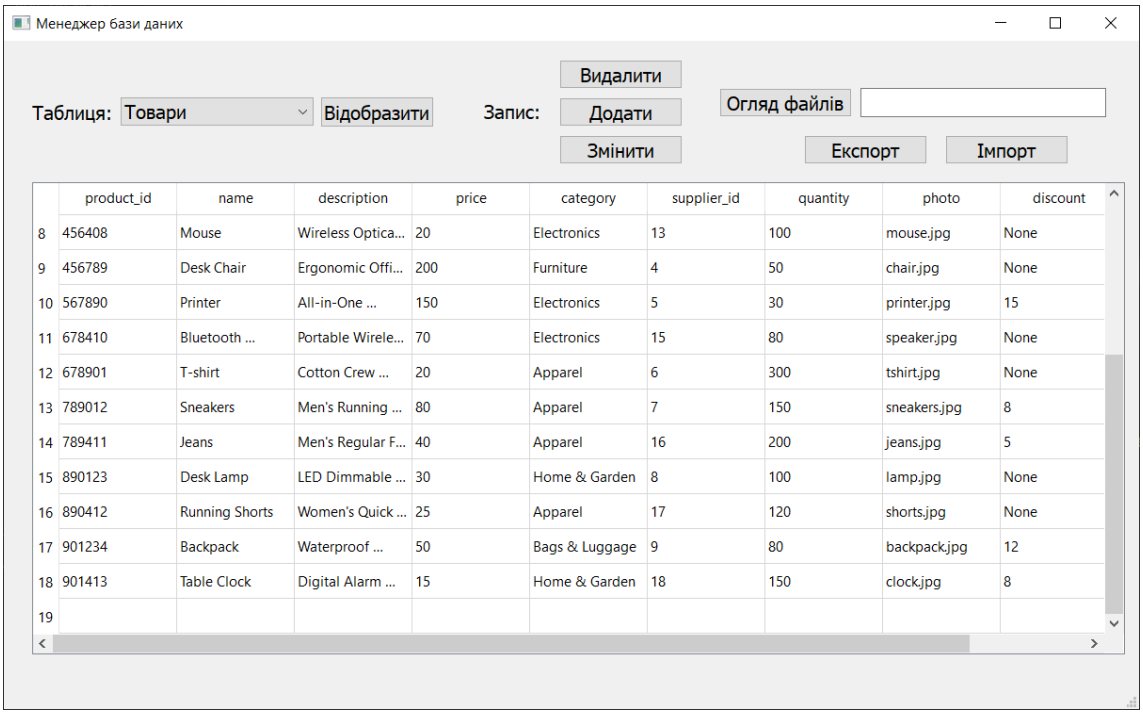
Рисунок 3.7 – Список таблиць



	product_id	name	description	price	category	supplier_id	quantity	photo	discount
9	456789	Desk Chair	Ergonomic Offi...	200	Furniture	4	50	chair.jpg	None
10	567409	Bookshelf	Wooden 5-Tier ...	150	Furniture	14	30	bookshelf.jpg	10
11	567890	Printer	All-in-One ...	150	Electronics	5	30	printer.jpg	15
12	678410	Bluetooth ...	Portable Wirele...	70	Electronics	15	80	speaker.jpg	None
13	678901	T-shirt	Cotton Crew ...	20	Apparel	6	300	tshirt.jpg	None
14	789012	Sneakers	Men's Running ...	80	Apparel	7	150	sneakers.jpg	8
15	789411	Jeans	Men's Regular F...	40	Apparel	16	200	jeans.jpg	5
16	890123	Desk Lamp	LED Dimmable ...	30	Home & Garden	8	100	lamp.jpg	None
17	890412	Running Shorts	Women's Quick ...	25	Apparel	17	120	shorts.jpg	None
18	901234	Backpack	Waterproof ...	50	Bags & Luggage	9	80	backpack.jpg	12
19	901413	Table Clock	Digital Alarm ...	15	Home & Garden	18	150	clock.jpg	8
20									

Рисунок 3.8 – Відображення даних таблиці «Товари»

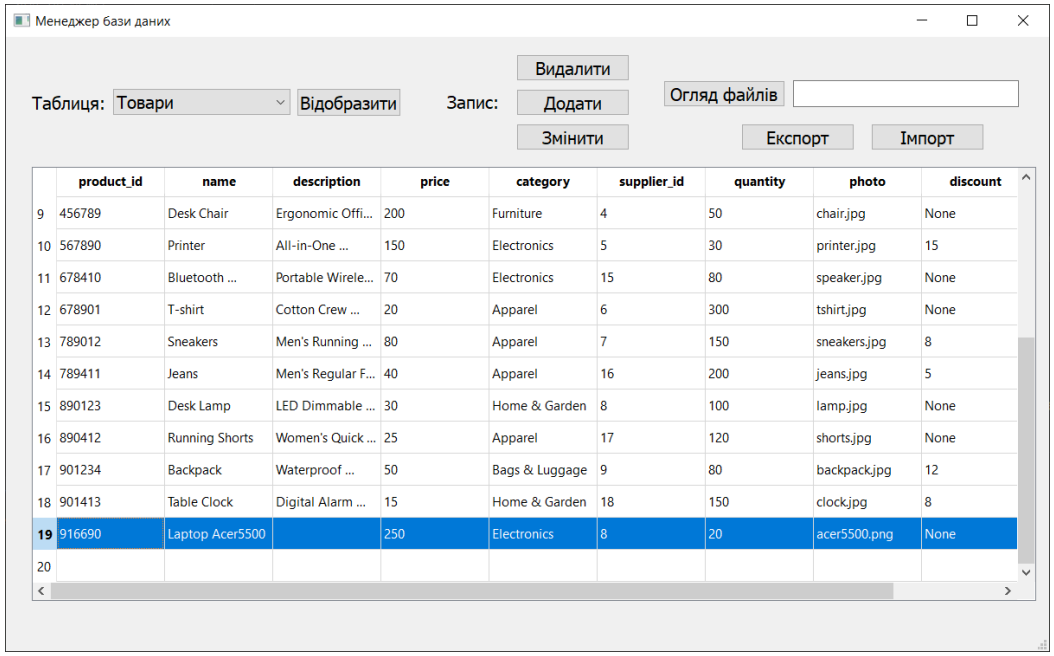
Для того щоб, видалити запис у таблиці потрібно виділити увесь рядок або тільки одну клітинку і натиснути кнопку «Видалити».



	product_id	name	description	price	category	supplier_id	quantity	photo	discount
8	456408	Mouse	Wireless Optica...	20	Electronics	13	100	mouse.jpg	None
9	456789	Desk Chair	Ergonomic Offi...	200	Furniture	4	50	chair.jpg	None
10	567890	Printer	All-in-One ...	150	Electronics	5	30	printer.jpg	15
11	678410	Bluetooth ...	Portable Wirele...	70	Electronics	15	80	speaker.jpg	None
12	678901	T-shirt	Cotton Crew ...	20	Apparel	6	300	tshirt.jpg	None
13	789012	Sneakers	Men's Running ...	80	Apparel	7	150	sneakers.jpg	8
14	789411	Jeans	Men's Regular F...	40	Apparel	16	200	jeans.jpg	5
15	890123	Desk Lamp	LED Dimmable ...	30	Home & Garden	8	100	lamp.jpg	None
16	890412	Running Shorts	Women's Quick ...	25	Apparel	17	120	shorts.jpg	None
17	901234	Backpack	Waterproof ...	50	Bags & Luggage	9	80	backpack.jpg	12
18	901413	Table Clock	Digital Alarm ...	15	Home & Garden	18	150	clock.jpg	8
19									

Рисунок 3.9 – Таблиця після видалення запису

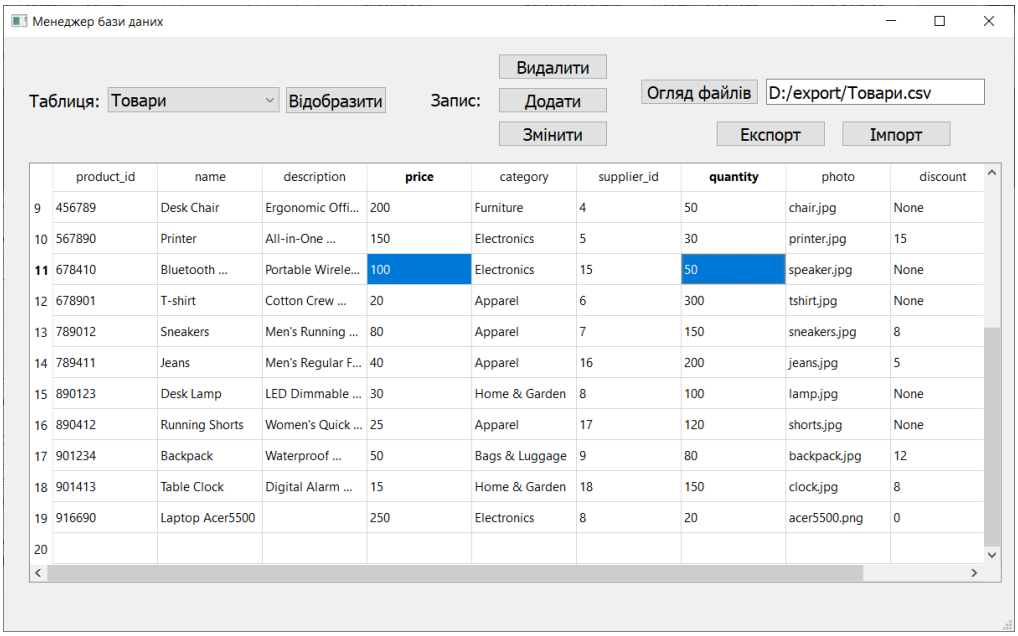
Для додавання запису необхідно прогортати в кінець таблиці і у порожньому рядку вписати потрібні дані, а потім натиснути кнопку «Додати».



	product_id	name	description	price	category	supplier_id	quantity	photo	discount
9	456789	Desk Chair	Ergonomic Offi...	200	Furniture	4	50	chair.jpg	None
10	567890	Printer	All-in-One ...	150	Electronics	5	30	printer.jpg	15
11	678410	Bluetooth ...	Portable Wirele...	70	Electronics	15	80	speaker.jpg	None
12	678901	T-shirt	Cotton Crew ...	20	Apparel	6	300	tshirt.jpg	None
13	789012	Sneakers	Men's Running ...	80	Apparel	7	150	sneakers.jpg	8
14	789411	Jeans	Men's Regular F...	40	Apparel	16	200	jeans.jpg	5
15	890123	Desk Lamp	LED Dimmable ...	30	Home & Garden	8	100	lamp.jpg	None
16	890412	Running Shorts	Women's Quick ...	25	Apparel	17	120	shorts.jpg	None
17	901234	Backpack	Waterproof ...	50	Bags & Luggage	9	80	backpack.jpg	12
18	901413	Table Clock	Digital Alarm ...	15	Home & Garden	18	150	clock.jpg	8
19	916690	Laptop Acer5500		250	Electronics	8	20	acer5500.png	None
20									

Рисунок 3.10 – Таблиця після видалення запису

Для того аби змінити запис треба натиснути на бажану клітинку, відредагувати її і натиснути на кнопку «Змінити».



	product_id	name	description	price	category	supplier_id	quantity	photo	discount
9	456789	Desk Chair	Ergonomic Offi...	200	Furniture	4	50	chair.jpg	None
10	567890	Printer	All-in-One ...	150	Electronics	5	30	printer.jpg	15
11	678410	Bluetooth ...	Portable Wirele...	100	Electronics	15	50	speaker.jpg	None
12	678901	T-shirt	Cotton Crew ...	20	Apparel	6	300	tshirt.jpg	None
13	789012	Sneakers	Men's Running ...	80	Apparel	7	150	sneakers.jpg	8
14	789411	Jeans	Men's Regular F...	40	Apparel	16	200	jeans.jpg	5
15	890123	Desk Lamp	LED Dimmable ...	30	Home & Garden	8	100	lamp.jpg	None
16	890412	Running Shorts	Women's Quick ...	25	Apparel	17	120	shorts.jpg	None
17	901234	Backpack	Waterproof ...	50	Bags & Luggage	9	80	backpack.jpg	12
18	901413	Table Clock	Digital Alarm ...	15	Home & Garden	18	150	clock.jpg	8
19	916690	Laptop Acer5500		250	Electronics	8	20	acer5500.png	0
20									

Рисунок 3.11 – Таблиця після зміни запису

Також програма дозволяє експортувати дані обраної таблиці. Для цього потрібно вказати файл для експорту і натиснути кнопку «Експорт». Обрати файл можна вписавши назву файлу або за допомогою файлового провідники натиснувши на кнопку «Огляд файлів».

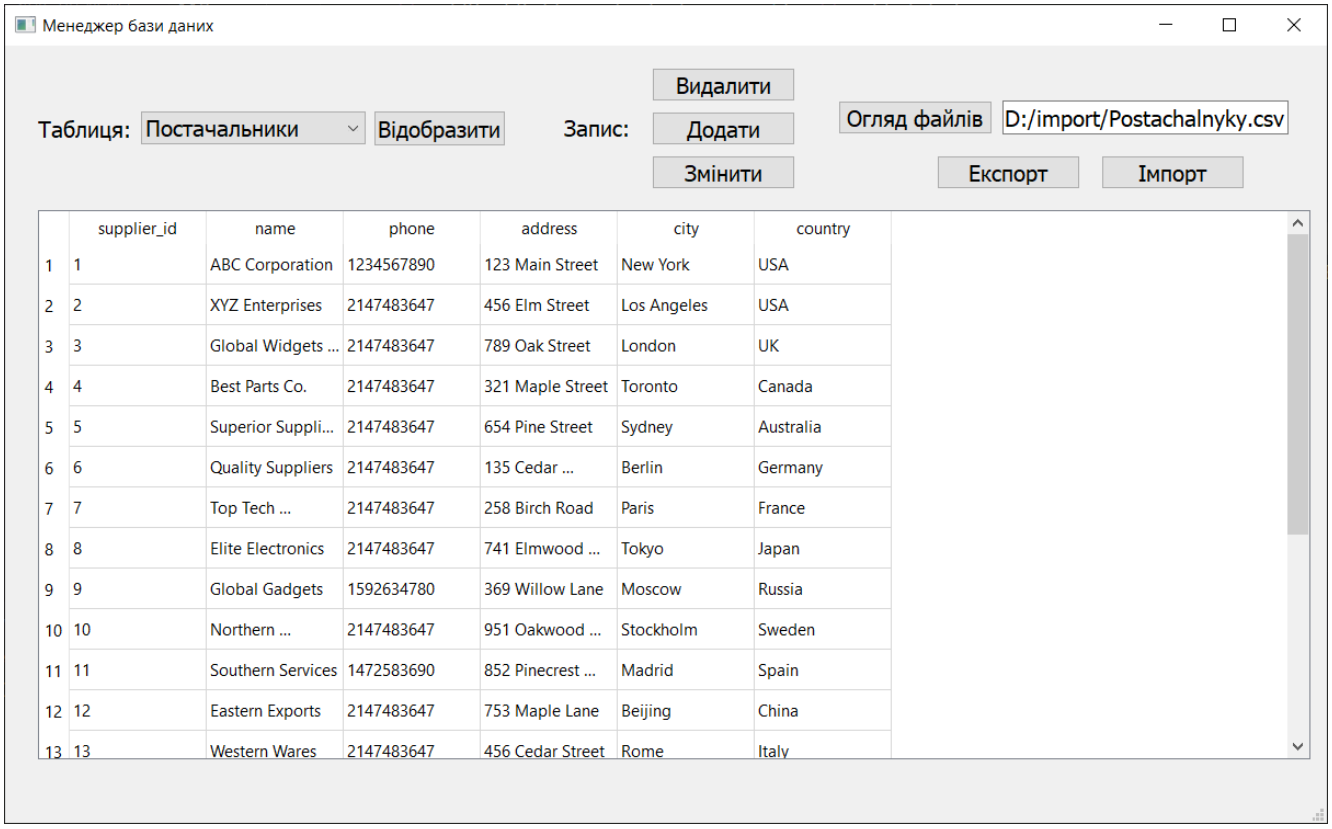
	A	B	C	D	E	F	G	H	I
1	product_id	name	description	price	category	supplier_id	quantity	photo	discount
2	123405	Coffee Maker	Programmable Coffee Machine	50	Appliances	10	20	coffee_maker.jpg	NULL
3	123414	Travel Bag	Durable Carry-On Travel Duffel Bag	60	Bags & Luggage	19	50	bag.jpg	NULL
4	123456	Laptop	15.6" Intel Core i5 8GB RAM 256GB SSD	800	Electronics	1	50	laptop.jpg	10
5	234406	Monitor	27" Full HD IPS Monitor	250	Electronics	11	40	monitor.jpg	20
6	234567	Smartphone	6.1" Android 12 128GB Storage	600	Electronics	2	100	phone.jpg	NULL
7	345407	Keyboard	Mechanical Gaming Keyboard	100	Electronics	12	60	keyboard.jpg	NULL
8	345678	Headphones	Wireless Bluetooth Over-Ear Headphones	100	Electronics	3	200	headphones.jpg	5
9	456408	Mouse	Wireless Optical Mouse	20	Electronics	13	100	mouse.jpg	NULL
10	456789	Desk Chair	Ergonomic Office Chair with Lumbar Support	200	Furniture	4	50	chair.jpg	NULL
11	567890	Printer	All-in-One Wireless Color Inkjet Printer	150	Electronics	5	30	printer.jpg	15
12	678410	Bluetooth Speaker	Portable Wireless Speaker	70	Electronics	15	80	speaker.jpg	NULL
13	678901	T-shirt	Cotton Crew Neck T-shirt	20	Apparel	6	300	tshirt.jpg	NULL
14	789012	Sneakers	Men's Running Shoes	80	Apparel	7	150	sneakers.jpg	8
15	789411	Jeans	Men's Regular Fit Jeans	40	Apparel	16	200	jeans.jpg	5
16	890123	Desk Lamp	LED Dimmable Table Lamp	30	Home & Garden	8	100	lamp.jpg	NULL
17	890412	Running Shorts	Women's Quick Dry Running Shorts	25	Apparel	17	120	shorts.jpg	NULL
18	901234	Backpack	Waterproof Laptop Backpack	50	Bags & Luggage	9	80	backpack.jpg	12
19	901413	Table Clock	Digital Alarm Clock with Snooze	15	Home & Garden	18	150	clock.jpg	8
20	916690	Laptop Acer5500		250	Electronics	8	20	acer5500.png	0

Рисунок 3.12 – Експортовані дані з таблиці «Товари»

Так само можна імпортувати дані. Файл для імпорту повинен містити заголовки, записи відокремлені переносом рядка і значення записів відокремлені табуляцією.

```
supplier_id name    phone   address city    country
1   ABC Corporation 1234567890  123 Main Street New York    USA
2   XYZ Enterprises 2147483647  456 Elm Street  Los Angeles USA
3   Global Widgets Ltd. 2147483647  789 Oak Street  London  UK
4   Best Parts Co.  2147483647  321 Maple Street    Toronto Canada
5   Superior Supplies Inc.  2147483647  654 Pine Street Sydney  Australia
6   Quality Suppliers   2147483647  135 Cedar Avenue    Berlin  Germany
7   Top Tech Solutions  2147483647  258 Birch Road  Paris   France
8   Elite Electronics   2147483647  741 Elmwood Drive   Tokyo   Japan
9   Global Gadgets  1592634780  369 Willow Lane Moscow  Russia
10  Northern Suppliers  2147483647  951 Oakwood Court   Stockholm   Sweden
11  Southern Services   1472583690  852 Pinecrest Avenue    Madrid  Spain
12  Eastern Exports 2147483647  753 Maple Lane  Beijing China
13  Western Wares   2147483647  456 Cedar Street    Rome    Italy
14  Sunrise Supplies    2147483647  987 Birchwood Drive Cairo   Egypt
15  Moonlight Merchants 2147483647  147 Elmwood Drive   Mexico City Mexico
16  Stellar Systems 2147483647  369 Oakwood Drive   Mumbai  India
17  Galaxy Goods    2147483647  123 Willow Street   Sydney  Australia
18  Planetary Parts 2147483647  258 Cedar Avenue    Rio de Janeiro  Brazil
19  Celestial Components    2147483647  369 Pine Lane   Athens  Greece
```

Рисунок 3.13 – Дані для імпорту у таблицю «Постачальники»



	supplier_id	name	phone	address	city	country
1	1	ABC Corporation	1234567890	123 Main Street	New York	USA
2	2	XYZ Enterprises	2147483647	456 Elm Street	Los Angeles	USA
3	3	Global Widgets ...	2147483647	789 Oak Street	London	UK
4	4	Best Parts Co.	2147483647	321 Maple Street	Toronto	Canada
5	5	Superior Suppli...	2147483647	654 Pine Street	Sydney	Australia
6	6	Quality Suppliers	2147483647	135 Cedar ...	Berlin	Germany
7	7	Top Tech ...	2147483647	258 Birch Road	Paris	France
8	8	Elite Electronics	2147483647	741 Elmwood ...	Tokyo	Japan
9	9	Global Gadgets	1592634780	369 Willow Lane	Moscow	Russia
10	10	Northern ...	2147483647	951 Oakwood ...	Stockholm	Sweden
11	11	Southern Services	1472583690	852 Pinecrest ...	Madrid	Spain
12	12	Eastern Exports	2147483647	753 Maple Lane	Beijing	China
13	13	Western Wares	2147483647	456 Cedar Street	Rome	Italy

Рисунок 3.14 – Таблиця «Постачальники» після імпорту

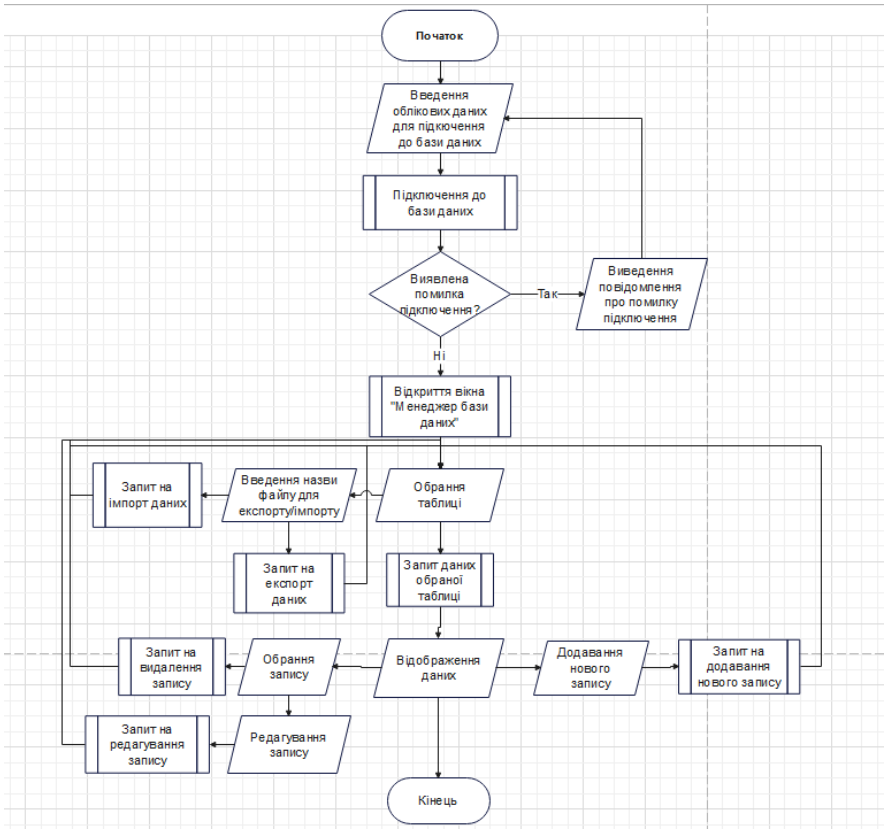


Рисунок 3.15 – Блок-схема алгоритму роботи застосунку

ВИСНОВКИ

В результаті виконання дипломної роботи було розроблено систему управління даними для інтернет-магазину, яка вирішує задачі організації, збереження та управління даними.

В першому розділі був проведений аналіз предметної області, було досліджено методи зберігання та обробки інформації, розглянуті існуючі програми для порівняння та визначення їх недоліків.

У другому розділі описано складові системи та задачі які вона повинна виконувати. Також обрано та описано засоби розробки, такі як: СУБД, мова програмування, середовище розробки і хмарний сервіс.

У третьому розділі було розроблено базу даних для інтернет-магазину а також описано основні етапи її проектування та розгортання за допомогою хмарного сервісу Amazon RDS. Крім того, описано інтерфейс і функціонал програми для взаємодії з базою даних.

Дослідження показало, що хмарні бази даних є ефективним та зручним рішенням для забезпечення доступу, безпеки та масштабованості даних для інтернет-магазинів. Також проект демонструє важливість інтеграції систем управління базами даних і програмного забезпечення для удосконалення процесів пов’язаних з організацією, збереженням та управлінням даними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Учасники проектів Вікімедіа. Інтегроване середовище розробки. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Інтегроване_середовище_розробки.
2. About. phpMyAdmin. URL: https://www.phpmyadmin.net/
3. Учасники проектів Вікімедіа. MySQL Workbench. Вікіпедія. URL: https://uk.wikipedia.org/wiki/MySQL_Workbench.
4. MySQL Enterprise Edition. MySQL. URL: https://www.mysql.com/products/workbench/.
5. The Python Tutorial. Python documentation. URL: https://docs.python.org/3/tutorial/index.html.
6. What is Python? Executive Summary. Python.org. URL: https://www.python.org/doc/essays/blurb/.
7. eComStreet. What are the main features of Python?. LinkedIn. URL: https://www.linkedin.com/pulse/what-main-features-python-ecomstreet-5yawc.
8. Учасники проектів Вікімедіа. Система керування базами даних. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Система_керування_базами_даних.
9. PyCharm Features. *JetBrains*. URL: https://www.jetbrains.com/pycharm/features/.
10. PyCharm: The Python IDE for Web Development. JetBrains. URL: https://www.jetbrains.com/pycharm/web-development/.
11. Cloud-based vs Traditional databases: which one should you choose?. WeAreBrain. URL: https://wearebrain.com/blog/cloud-based-vs-traditional-databases-comparison
12. Choosing the Right Database: Cloud-Based vs. Traditional Options. Flat Rock Technology. URL: https://flatrocktech.com/blog/cloud-based-vs-traditional-databases
13. Учасники проектів Вікімедіа. Amazon Web Services. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Amazon_Web_Services.
14. Managed SQL Database - Amazon Relational Database Service (RDS) -

AWS. Amazon Web Services, Inc. URL: https://aws.amazon.com/rds/?nc1=h_ls.

15. Учасники проектів Вікімедіа. Amazon Relational Database Service. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Amazon_Relational_Database_Service.

16. Amazon RDS Features | Cloud Relational Database | Amazon Web Services. Amazon Web Services, Inc. URL: https://aws.amazon.com/rds/features/?nc1=h_ls.

17. Amazon RDS DB instances - Amazon Relational Database Service. AWS Documentation. URL: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.DBInstance.html.

18. Configuring and managing a Multi-AZ deployment - Amazon Relational Database Service. AWS Documentation. URL: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html

19. DB instance classes - Amazon Relational Database Service. AWS Documentation. URL: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html.

20. Amazon RDS DB instance storage - Amazon Relational Database Service. AWS Documentation. URL: https://docs.aws.amazon.com//AmazonRDS/latest/UserGuide/CHAP_Storage.html.

21. What is Amazon VPC? - Amazon Virtual Private Cloud. AWS Documentation. URL: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html.

22. Murach J. Murach's MySQL. Mike Murach & Associates, 2012 p. C. 306–322.

Додаток А

SQL ЗАПИТИ ДЛЯ СТВОРЕННЯ БАЗИ ДАНИХ

```
create database online_store;

create table products (
	product_id int primary key,
	name varchar(100) not null,
	description varchar(500),
	price int not null,
	category varchar (50) not null,
	supplier_id int not null,
	quantity int not null,
	photo varchar(50),
	discount int,
	constraint fk_supplier_id foreign key(supplier_id)
		references suppliers(supplier_id)
		on update cascade
		on delete cascade
);

create table clients (
	client_id int auto_increment primary key,
	first_name varchar(50) not null,
	middle_name varchar(50),
	last_name varchar(50) not null,
	email varchar(50) not null,
	password varchar(50) not null,
	phone int not null
```

```
);

create table orders (
        order_id int auto_increment primary key,
        client_id int,
        status varchar(50) not null,
        order_date date not null,
        shipped_date date,
        address varchar(50) not null,
        city varchar(50) not null,
        country varchar(50) not null,
        constraint fk_client_id foreign key(client_id)
                references clients(client_id)
                on update cascade
                on delete set null
);

create table order_details (
        order_id int not null,
        product_id int not null,
        quantity int not null,
        constraint fk_order_id foreign key(order_id)
                references orders(order_id)
                on update cascade
                on delete cascade,
        constraint fk_product_id foreign key(product_id)
                references products(product_id)
                on update cascade
                on delete cascade
);
```

```
create table suppliers (
        supplier_id int auto_increment primary key,
        name varchar(50) not null,
        phone int not null,
        address varchar(50) not null,
        city varchar(50) not null,
        country varchar(50) not null
);
```

Додаток Б

ЛІСТИНГ ПРОГРАМИ

Файл login_window.py

```
from PyQt5.QtWidgets import QMessageBox
from main_window import Ui_MainWindow
from PyQt5 import QtCore, QtGui, QtWidgets
import mysql.connector

class Ui_LoginWindow(object):
    def setupUi(self, LoginWindow):
        LoginWindow.setObjectName("LoginWindow")
        LoginWindow.resize(550, 489)
        self.centralwidget = QtWidgets.QWidget(LoginWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(90, 20, 360, 70))
        font = QtGui.QFont()
        font.setPointSize(16)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(80, 110, 140, 30))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.label_2.setFont(font)
        self.label_2.setObjectName("label_2")
        self.hostLineEdit = QtWidgets.QLineEdit(self.centralwidget)
```

```
self.hostLineEdit.setGeometry(QtCore.QRect(170, 110, 300, 30))
font = QtGui.QFont()
font.setPointSize(10)
self.hostLineEdit.setFont(font)
self.hostLineEdit.setText("")
self.hostLineEdit.setObjectName("hostLineEdit")
self.userLineEdit = QtWidgets.QLineEdit(self.centralwidget)
self.userLineEdit.setGeometry(QtCore.QRect(170, 160, 300, 30))
font = QtGui.QFont()
font.setPointSize(10)
self.userLineEdit.setFont(font)
self.userLineEdit.setObjectName("userLineEdit")
self.label_3 = QtWidgets.QLabel(self.centralwidget)
self.label_3.setGeometry(QtCore.QRect(80, 160, 140, 30))
font = QtGui.QFont()
font.setPointSize(12)
self.label_3.setFont(font)
self.label_3.setObjectName("label_3")
self.passwrodLineEdit = QtWidgets.QLineEdit(self.centralwidget)
self.passwrodLineEdit.setGeometry(QtCore.QRect(170, 210, 300, 30))
font = QtGui.QFont()
font.setPointSize(10)
self.passwrodLineEdit.setFont(font)
self.passwrodLineEdit.setObjectName("passwrodLineEdit")
self.label_4 = QtWidgets.QLabel(self.centralwidget)
self.label_4.setGeometry(QtCore.QRect(80, 210, 140, 30))
font = QtGui.QFont()
font.setPointSize(12)
self.label_4.setFont(font)
self.label_4.setObjectName("label_4")
```

```
        self.connectPushButton = QtWidgets.QPushButton(self.centralwidget)
        self.connectPushButton.setGeometry(QtCore.QRect(175, 310, 200, 50))
        font = QtGui.QFont()
        font.setPointSize(12)
        self.connectPushButton.setFont(font)
        self.connectPushButton.setObjectName("connectPushButton")
        LoginWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(LoginWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 550, 26))
        self.menubar.setObjectName("menubar")
        LoginWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(LoginWindow)
        self.statusbar.setObjectName("statusbar")
        LoginWindow.setStatusBar(self.statusbar)

        self.retranslateUi(LoginWindow)
        QtCore.QMetaObject.connectSlotsByName(LoginWindow)

        self.connectPushButton.clicked.connect(self.connect_to_db)

    def connect_to_db(self):
        self.host = self.hostLineEdit.text()
        self.user = self.userLineEdit.text()
        self.password = self.passwrodLineEdit.text()

        try:
            self.cnx = mysql.connector.connect(user=self.user, password=self.password,
host=self.host, database='online_store',
                                               allow_local_infile=True)
            self.show_main_window()
```

```
        except mysql.connector.Error as err:
            msg = QMessageBox()
            msg.setIcon(QMessageBox.Critical)
            msg.setText("Помилка підключення")
            msg.setInformativeText('Направильні облікові дані')
            msg.setWindowTitle("Помилка")
            msg.exec_()

    def show_main_window(self):
        self.main_window = QtWidgets.QMainWindow()
        self.ui_mw = Ui_MainWindow(cnx=self.cnx, user=self.user,
password=self.password, host=self.host)
        self.ui_mw.setupUi(self.main_window)
        self.main_window.show()
        LoginWindow.close()

    def retranslateUi(self, LoginWindow):
        _translate = QtCore.QCoreApplication.translate
        LoginWindow.setWindowTitle(_translate("LoginWindow", "Підключення до
бази даних"))
        self.label.setText(_translate("LoginWindow", "Підключення до бази даних"))
        self.label_2.setText(_translate("LoginWindow", "Хост"))
        self.label_3.setText(_translate("LoginWindow", "Ім\'я"))
        self.label_4.setText(_translate("LoginWindow", "Пароль"))
        self.connectPushButton.setText(_translate("LoginWindow", "Підключитися"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
```

```
    LoginWindow = QtWidgets.QMainWindow()
    ui = Ui_LoginWindow()
    ui.setupUi(LoginWindow)
    LoginWindow.show()
    sys.exit(app.exec_())
```

Файл main_window.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QTableWidgetItem
import os

class Ui_MainWindow(object):
    def __init__(self, cnx, user, password, host):
        self.cnx = cnx
        self.user = user
        self.password = password
        self.host = host

    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1206, 709)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.horizontalLayoutWidget = QtWidgets.QWidget(self.centralwidget)
        self.horizontalLayoutWidget.setGeometry(QtCore.QRect(30, 40, 427, 71))
        self.horizontalLayoutWidget.setObjectName("horizontalLayoutWidget")
        self.horizontalLayout = QtWidgets.QHBoxLayout(self.horizontalLayoutWidget)
        self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
        self.horizontalLayout.setObjectName("horizontalLayout")
```

```
self.label = QtWidgets.QLabel(self.horizontalLayoutWidget)
font = QtGui.QFont()
font.setPointSize(12)
self.label.setFont(font)
self.label.setObjectName("label")
self.horizontalLayout.addWidget(self.label)
self.comboBox = QtWidgets.QComboBox(self.horizontalLayoutWidget)
font = QtGui.QFont()
font.setPointSize(12)
self.comboBox.setFont(font)
self.comboBox.setObjectName("comboBox")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.horizontalLayout.addWidget(self.comboBox)
self.showPushButton = QtWidgets.QPushButton(self.horizontalLayoutWidget)
font = QtGui.QFont()
font.setPointSize(12)
self.showPushButton.setFont(font)
self.showPushButton.setObjectName("showPushButton")
self.horizontalLayout.addWidget(self.showPushButton)
self.tableWidget = QtWidgets.QTableWidget(self.centralwidget)
self.tableWidget.setGeometry(QtCore.QRect(30, 150, 1161, 501))
self.tableWidget.setObjectName("tableWidget")
self.tableWidget.setColumnCount(0)
self.tableWidget.setRowCount(0)
self.deletePushButton = QtWidgets.QPushButton(self.centralwidget)
self.deletePushButton.setGeometry(QtCore.QRect(590, 20, 131, 31))
```

```
font = QtGui.QFont()
font.setPointSize(12)
self.deletePushButton.setFont(font)
self.deletePushButton.setObjectName("deletePushButton")
self.addPushButton = QtWidgets.QPushButton(self.centralwidget)
self.addPushButton.setGeometry(QtCore.QRect(590, 60, 131, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.addPushButton.setFont(font)
self.addPushButton.setObjectName("addPushButton")
self.updatePushButton = QtWidgets.QPushButton(self.centralwidget)
self.updatePushButton.setGeometry(QtCore.QRect(590, 100, 131, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.updatePushButton.setFont(font)
self.updatePushButton.setObjectName("updatePushButton")
self.label_2 = QtWidgets.QLabel(self.centralwidget)
self.label_2.setGeometry(QtCore.QRect(510, 40, 105, 69))
font = QtGui.QFont()
font.setPointSize(12)
self.label_2.setFont(font)
self.label_2.setObjectName("label_2")
self.exportPushButton = QtWidgets.QPushButton(self.centralwidget)
self.exportPushButton.setGeometry(QtCore.QRect(850, 100, 131, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.exportPushButton.setFont(font)
self.exportPushButton.setObjectName("exportPushButton")
self.file_browser_PushButton = QtWidgets.QPushButton(self.centralwidget)
self.file_browser_PushButton.setGeometry(QtCore.QRect(760, 50, 141, 31))
```

```
font = QtGui.QFont()
font.setPointSize(12)
self.file_browser_PushButton.setFont(font)
self.file_browser_PushButton.setObjectName("file_browser_PushButton")
self.importPushButton = QtWidgets.QPushButton(self.centralwidget)
self.importPushButton.setGeometry(QtCore.QRect(1000, 100, 131, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.importPushButton.setFont(font)
self.importPushButton.setObjectName("importPushButton")
self.file_browserLineEdit = QtWidgets.QLineEdit(self.centralwidget)
self.file_browserLineEdit.setGeometry(QtCore.QRect(910, 50, 261, 31))
font = QtGui.QFont()
font.setPointSize(12)
self.file_browserLineEdit.setFont(font)
self.file_browserLineEdit.setObjectName("file_browserLineEdit")
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 1206, 26))
self.menubar.setObjectName("menubar")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)

# ----------------------------------------------------
```

```
        self.cursor = self.cnx.cursor()
        self.showPushButton.clicked.connect(self.show_data)
        self.deletePushButton.clicked.connect(self.delete_data)
        self.addPushButton.clicked.connect(self.add_data)
        self.updatePushButton.clicked.connect(self.update_data)
        self.file_browser_PushButton.clicked.connect(self.show_file_browser_dialog)
        self.exportPushButton.clicked.connect(self.export_data)
        self.importPushButton.clicked.connect(self.import_data)

    def import_data(self):
        selected_table = self.get_selected_table()
        command = f"LOAD DATA LOCAL INFILE '{self.file_browserLineEdit.text()}'
INTO TABLE {selected_table} IGNORE 1 LINES;"
        self.cursor.execute(command)
        self.cnx.commit()

    def export_data(self):
        selected_table = self.get_selected_table()
        command = f'mysql -u {self.user} -p{self.password} --database=online_store --
host={self.host} --port=3306 --batch -e "select * from {selected_table}" >
{self.file_browserLineEdit.text()}'
        os.system(command)

    def show_file_browser_dialog(self):
        fname = QtWidgets.QFileDialog.getOpenFileName(QtWidgets.QFileDialog(),
'Оберіть файл')
        self.file_browserLineEdit.setText(fname[0])

    def update_data(self):
        index = self.tableWidget.currentIndex()
```

```
        selected_table = self.get_selected_table()
        columns_describe = self.get_columns_describe()
        column_count = len(columns_describe)
        column_names = [i[0] for i in columns_describe]
        id_column_name = columns_describe[0][0]
        row_data = []
        for i in range(column_count):
            cell_data = self.tableWidget.model().index(index.row(), i).data()
            row_data.append(cell_data)

        set_args = ', '.join([f"{column_name} = '{data}'" for column_name, data in
zip(column_names, row_data)])
        self.cursor.execute(f'update {selected_table} set {set_args} where
{id_column_name} = {row_data[0]};')
        self.cnx.commit()

    def add_data(self):
        index = self.tableWidget.currentIndex()
        selected_table = self.get_selected_table()
        columns_describe = self.get_columns_describe()
        columns_count = len(columns_describe)
        row_data = []
        for i in range(columns_count):
            cell_data = self.tableWidget.model().index(index.row(), i).data()
            row_data.append(cell_data)

        self.cursor.execute(f'insert into {selected_table} values {tuple(row_data)}')
        self.cnx.commit()

        self.tableWidget.insertRow(self.tableWidget.rowCount())
```

```
    def delete_data(self):
        index = self.tableWidget.currentIndex()
        id = self.tableWidget.model().index(index.row(), 0).data()
        self.tableWidget.removeRow(index.row())

        selected_table = self.get_selected_table()
        columns_describe = self.get_columns_describe()
        id_column_name = columns_describe[0][0]
        self.cursor.execute(f'delete from {selected_table} where {id_column_name} =
{id};')
        self.cnx.commit()

    def show_data(self):
        while self.tableWidget.rowCount() > 0:
            self.tableWidget.removeRow(0)

        columns_describe = self.get_columns_describe()
        column_names = [i[0] for i in columns_describe]

        selected_table = self.get_selected_table()
        self.cursor.execute(f'select * from {selected_table}')
        result = self.cursor.fetchall()

        if bool(result) is False:
            self.tableWidget.insertRow(0)
            for i in range(len(column_names)):
                self.tableWidget.insertColumn(i)
            self.tableWidget.setHorizontalHeaderLabels(column_names)
            return 0
```

```
        self.tableWidget.setColumnCount(len(result[0]))
        for row_number, row_data in enumerate(result):
            self.tableWidget.insertRow(row_number)
            for column_number, data in enumerate(row_data):
                self.tableWidget.setItem(row_number, column_number,
QTableWidgetItem(str(data)))

        self.tableWidget.setHorizontalHeaderLabels(column_names)
        self.tableWidget.insertRow(len(result))

    def get_selected_table(self):
        tables = {
            'Товари': 'products',
            'Клієнти': 'clients',
            'Замовлення': 'orders',
            'Деталі замовлення': 'order_details',
            'Постачальники': 'suppliers'
        }

        selected_table = tables[self.comboBox.currentText()]

        return selected_table

    def get_columns_describe(self):
        selected_table = self.get_selected_table()
        self.cursor.execute(f'show columns from {selected_table}')
        columns_describe = self.cursor.fetchall()
        return columns_describe
```

```
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Менеджер бази даних"))
        self.label.setText(_translate("MainWindow", "Таблиця:"))
        self.comboBox.setItemText(0, _translate("MainWindow", "Товари"))
        self.comboBox.setItemText(1, _translate("MainWindow", "Клієнти"))
        self.comboBox.setItemText(2, _translate("MainWindow", "Замовлення"))
        self.comboBox.setItemText(3, _translate("MainWindow", "Деталі замовлення"))
        self.comboBox.setItemText(4, _translate("MainWindow", "Постачальники"))
        self.showPushButton.setText(_translate("MainWindow", "Відобразити"))
        self.deletePushButton.setText(_translate("MainWindow", "Видалити"))
        self.addPushButton.setText(_translate("MainWindow", "Додати"))
        self.updatePushButton.setText(_translate("MainWindow", "Змінити"))
        self.label_2.setText(_translate("MainWindow", "Запис:"))
        self.exportPushButton.setText(_translate("MainWindow", "Експорт"))
        self.file_browser_PushButton.setText(_translate("MainWindow", "Огляд файлів"))
        self.importPushButton.setText(_translate("MainWindow", "Імпорт"))
```