МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
Кафедра комп'ютерних наук

ДИПЛОМНА РОБОТА

освітній ступінь – «Бакалавр»

на тему: **СИСТЕМА ТОВАРООБІГУ МАГАЗИНУ ОДЯГУ**

Виконала: студентка 4 курсу,групи БІТск-21
спеціальності 122 Комп'ютерні науки
Аріана СЛАВІЦЬКА
Керівник: **Тетяна ДЕМКІВСЬКА**
Рецензент: **Віктор ЧУПРИНКА**

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет	мехатроніки та комп’ютерних технологій
Кафедра	комп’ютерних наук та технологій
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп’ютерних наук та технологій
_____ Володимир ЩЕРБАНЬ
(підпис)
« » 20 р.

ЗАВДАННЯ

НА ДИПЛОМНУ БАКАЛАВРСЬКУ РОБОТУ

студентки
Славіцької Аріани Анатоліївни

1. Тема роботи**:** Система товарообігу магазину одягу

Науковий керівник роботи: Демківська Тетяна Іванівна, доцент кафедри комп'ютерних наук, затверджена наказом КНУТД від “08” листопада 2022 року № 224-уч.

2. Строк подання студентом дипломної роботи 25.05.2023 р.

3. Вихідні дані до дипломної роботи: Розробки кафедри комп’ютерних наук та технологій, рекомендована література, додатки

4. Зміст дипломної бакалаврської роботи: Розділ 1. Аналіз предметної області, Розділ 2. Проєктування реалізації системи MerchFlow, Розділ 3. Програмна реалізація системи MerchFlow.

5. Дата видачі завдання 15.03.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Терміни виконання етапів	Примітка
1	Вступ	20.04.2023	
2	Розділ 1 Аналіз предметної області	25.04.2023	
3	Розділ 2 Проектування програмної реалізації	06.05.2023	
4	Розділ 3 Розробка програмної реалізації	09.05.2023	
5	Висновки	10.05.2023	
6	Оформлення дипломної бакалаврської роботи	15.05.2023	
7	Здача дипломної бакалаврської роботи на кафедру для рецензування (за 14 днів до захисту)	23.05.2023	
8	Перевірка дипломної бакалаврської роботи на наявність ознак плагіату (за 10 днів до захисту)		
9	Подання дипломної бакалаврської роботи на затвердження завідувачу кафедри (за 7 днів до захисту)		

Студентка _____ Аріана СЛАВІЦЬКА
(підпис)

Науковий керівник роботи _____ Тетяна ДЕМКІВСЬКА
(підпис)

Рецензент _____ Віктор ЧУПРИНКА
(підпис)

АНОТАЦІЯ

Славіцька Аріана Анатоліївна. Система товарообігу магазину одягу.

Дипломна бакалаврська робота за спеціальністю 122 — “Комп’ютерні науки” — Київський національний університет технологій та дизайну, Київ, 2023 рік.

В роботі вирішено завдання автоматизації товарообігу в магазині одягу. Проєктування та налаштування системи виконано згідно з вимогами. Для виконання якісного та зручного обліку товарів створено систему, що надає змогу обробляти інформації про рух товарів.

Розроблена система призначена для відстеження та редагування даних про товарообіг підприємства.

Ключові слова: система, база даних, автоматизація, застосунок, товарообіг, облік товару.

ABSTRACT

Slavitska Ariana Anatoliivna. The turnover system of a clothing store.

Bachelor's thesis in speciality 122 - "Computer Science" - Kyiv National University of Technology and Design, Kyiv, 2023.

The work solves the problem of automating the turnover of goods in a clothing store. The system was designed and configured in accordance with the requirements. To perform high-quality and convenient accounting of goods, a system has been created that allows processing information on the movement of goods.

The developed system is designed to track and edit data on the turnover of the enterprise.

Keywords: system, database, automation, application, goods turnover, goods accounting. Keywords: system, database, automation, application, goods turnover, goods accounting.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	База даних
СУБД	Система управління базами даних
ПК	Персональний комп’ютер
UI *(User interface)*	Інтерфейс користувача
BAS *(Business Automation Software)*	Програма ведення бухгалтеського обліку
URL *(Uniform Resource Locator)*	Визначник місцезнаходження сайту в мережі Інтернет
IDE *(Integrated development environment)*	Інтегроване середовище розробки
OLE *(Object Linking and Embedding)*	Технологія зв'язування та впровадження об'єктів в інші документи та об'єкти

ЗМІСТ

ВСТУП

Актуальність теми дослідження

Сьогодні сфера електронної комерції стрімко розвивається. Щоб закріпитись на ринку та втримати позиції, потрібно знаходити нові шляхи розвитку. Постійно виникає необхідність шукати сучасні варіанти вдосконалення процесу обслуговування користувачів і надання послуг для успішного існування на торгівельному ринку та збільшення прибутку. Якість обслуговування – невід'ємна складова ефективності роботи магазинів, завдяки якій можливо задовольнити потреби покупців. «Обслуговування на вищому рівні» стало одним з найважлиіших елементів існування закладів торгівлі, оскільки клієнти вимагають не лише високої якості товарів, а й відмінного обслуговування та ефективної системи управління товарами.

Для забезпечення дієвого функціонування магазинів виникає необхідність дослідження проблем, їх вирішення з метою розвитку систем управління товарообігом.

В даний час спостерігається стрімкий розвиток інформаційних технологій і програмних засобів у сфері електронної комерції. Це обумовлено розширенням можливостей компаній за допомогою підключення додаткових пристроїв, електронних ресурсів, програмних забезпечень. Сучасні інформаційні технології дозволяють організувати роботу підприємства з максимальною ефективністю.

В наш час склалося так, що програмне забезпечення стало невід'ємною частиною сучасного повсякдення. Тепер воно знаходить своє застосування майже у всіх сферах людської діяльності. Звичайно, все більше і більше у різних галузях віддається перевага комп`ютеризованним системам керування, а не людям. Це лякає з одного боку, але добре спроектована система керування надає набагато більше впевненості та зводить до мінімуму критичні ситуації, які раніше були пов`язані з так званим «людським фактором».

З розвитком торгівлі, обслуговування стало однією з найважливіших галузей, які потребують надання послуг. Зараз у торгівельному бізнесі дуже

багато конкуренції, що стимулює багатьох лідерів цієї галузі впроваджувати найсучасніші та найкомфортніші технології, в тому числі і системи керування товарообігу. Більшість програм для ведення обліку не є універсальними, а заточені під особливості роботи компанії, яка використовує програму. Системи керування є дуже складними, адже складаються з багатьох взаємопов`язаних служб та підсистем. Наприклад, служба управління замовленнями, ведення постачальників, продажів, адміністрування товару тощо. Саме автоматизації торгівельного бізнесу присвячена дана дипломна робота.

Запровадження автоматизованої системи в магазинах одягу дозволить підвищити якість надання послуг споживачам, що забезпечить збільшення прибутку. Саме тому тема дипломної роботи допомагає вирішити проблему вдосконалення автоматизації процесу товарообігу в магазинах одягу.

Предмет дослідження

Предметом даного дослідження є вивчення автоматизації обліку товарів в магазинах одягу, а також впровадження системи товарообігу в підприємство. Інформаційною базою дипломної роботи є літературні джерела та інформація з глобальної мережі Інтернет.

Об’єкт дослідження

Об’єктом дослідження є створена система для обліку товарів магазину одягу.

Методи розробки

Для вирішення практичного завдання було використано мову програмування C++ та середовище розробки RAD Studio.

Структура, зміст та обсяг роботи

Дипломна робота складається з трьох розділів та містить 3 додатки, 48 рисунків та 24 джерела. У першому розділі був проведений аналіз предметної

області, огляд існуючих аналогів. У другому розділі визначено призначення та функціонал програмної реалізації; дослідження технологій та засобів розробки. У третьому розділі описана розробка програмної реалізації та її тестування. У додатку наведено лістинг коду програм, верстки та презентація.

РОЗДІЛ 1. Аналіз предметної області

1.1 Постановка задачі

Метою дипломної роботи є аналіз проблем автоматизації процесу обробки інформації, яка стосується обліку товарів , а також розробка пропозицій по вдосконаленню її діяльності.

Здійснення поставленої мети відбувалось шляхом вирішення наступних задач:

- вивчення основних напрямів впровадження нових технологій в торгівелькому бізнесі;
- дослідити основи організації ведення обліку товару і надання послуг споживачам в магазинах;
- визначити сутність автоматизації та її роль у підприємстві електронної комерції;
- розглянути види програмного забезпечення на ринку торгівлі;
- розглянути процес автоматизації процесу автоматизації введеня та обробки інформаціх, про товар;
- провести дослідження існуючих проблем в галузі комерції;
- провести критичний аналіз програмних продуктів магазинів;
- розглянути напрями вирішення питань щодо вдосконалення процесу автоматизації роботи користувачів.

Поставлені такі завдання:

- сформувати бази даних постачальників, їх адміністрування, а також мінімізувати людський фактор у функціонуванні системи обліку;
- підвищити продуктивність праці і понизити ймовірності помилок персоналу за рахунок заздалегідь продуманної і добре налагодженної системи керування;
- скоротити час та зекономити людські ресурси під час оформлення звітів, чеків, ведення обліку вручну;

- налагодити співпрацю усіх служб, аби заклад працював єдиною системою;
- автоматизувати та спростити роботу з веденням товарообігу.

1.2 Дослідження і аналіз об’єкту програмування

Основною діяльністю магазинів є продаж товару. Для якісного виконання цієї задачі, необхідно розуміти скільки товару і за якою ціною знаходиться на вашому складі. Товарний облік існує для того, щоб управляти переміщеннями товару у реалізації, тобто допомогає розуміти, що відбувається з товаром у магазиі, оптимізувати процеси, сформувати необхідний асортимент, який буде привабливим для клієнтів та уникнути крадіжок.

Невеликі магазини ведуть облік «вручну», тобто без допомоги додаткових пристроїв. "Ручний режим" в бухгалтерії допускається чинним законодавством і не тягне санкцій з боку податкової служби або правоохоронних органів. Однак на практиці цей спосіб не просто незручний і дорогий, але і тягне за собою не мало негативних наслідків, таких як:

- погіршення якості розрахунків;
- низький товарообіг;
- складність ведення документації;
- відсутність систематизації;
- неможливість сортування, або відбору за заданими критеріями;
- не ефективна робота постачальниками;
- помилки, викликані людським фактором;
- нестачі.

Це не всі негативні наслідки відсутності обліку, але їх достатньо, щоб віддати перевагу якісному автоматизованому обліку товарів.Дуже важливо, щоб обрані методи товарообліку у магазині роздрібної торгівлі забезпечували власника магазину достовірною та актуальною інформацією про товарообіг, були сучасними і технологічними.

Варто завжди думати в сторону удосконалення, спрощення та автоматизації. Сьогодні програмні рішення є помічниками для розв'язання рутинних питань та контролю бізнес-процесів.

Автоматизована система ведення товарообігу є зручним та дієвим за-собом для ведення товарообліку так, як програми виключає помилки такого типу:

Списання товару, якого немає на залишках. Тобто програма веде об-лік по кожній одиниці товару,який був оприбуткований документом надхо-дження товару та списаний документом продаж.

Ведення обліку коштів. На кожний товар назначена своя ціна та наці-нка. Програма автоматично розраховує суми продажу товару.В застосунку також зберігаються вся історія операцій та документів, що запобігає кращому контролю обліку товару та зменшує ризики зникнен-ня товару або коштів.

Автоматизована система ведення товарообігу це застосунок, який по-легує ведення бізнесу та дозволяє зосередитись на більш важливих та тяжких процесах.

Програми товарообліку спеціалізуються на різних напрямках виробництва. Кожна компанія обирає таку систему обліку товару, яка підходить їм та виконує всі необхідні функції ведення обліку для підриємства. Перед розробкою програмної реалізації системи, необхідно ознайомитись із існучими аналогами та проаналізувати їх сильні та слабкі сторони, щоб уникнти повторення слабкостей та створити унікальний функціонал системи.

Під час пошуту програм-аналогів не було знадено ідентичних програм, але більшість схожа по функціоналу. Розглянемо програми для складського обліку товару та програми для продаж товару. Щоб поєднати функції обидвох видів систем в одну.

1.3 Огляд існуючих аналогів

Програми товарообліку спеціалізуються на різних напрямках виробництва. Кожна компанія обирає таку систему обліку товару, яка підходить їм та виконує

всі необхідні функції ведення обліку для підриємства. Перед розробкою програмної реалізації системи, необхідно ознайомитись із існучими аналогами та проаналізувати їх сильні та слабкі сторони, щоб уникнти повторення слабкостей та створити унікальний функціонал системи.

Під час пошуку програм аналогів не було знадено ідентичних програм, але більшість схожа по функціоналу. Розглянемо програми для складського обліку товару та програми для продаж товару. Щоб поєднати функції обидвох видів систем в одну.

1.3.1 Poster POS

Одним із аналогів програми для ведення бізнесу є Poster POS - це хмарна каса для кафе, ресторанів, магазинів. POS-система Poster допомагає впорядкувати бізнес-процеси, автоматизувати заклад, оптимізувати витрати, мінімізувати крадіжки та збільшити продажі. Вона дає можливість робити продажі, вести фінансовий, бухгалтерський, складський облік, управляти персоналом та працювати з клієнтською базою.

Програма призначена для підприємств, що займаються виробництвом, реалізацією та організацією споживання кулінарної продукції. Приклади підприємств зображено на рисунку 1.1.

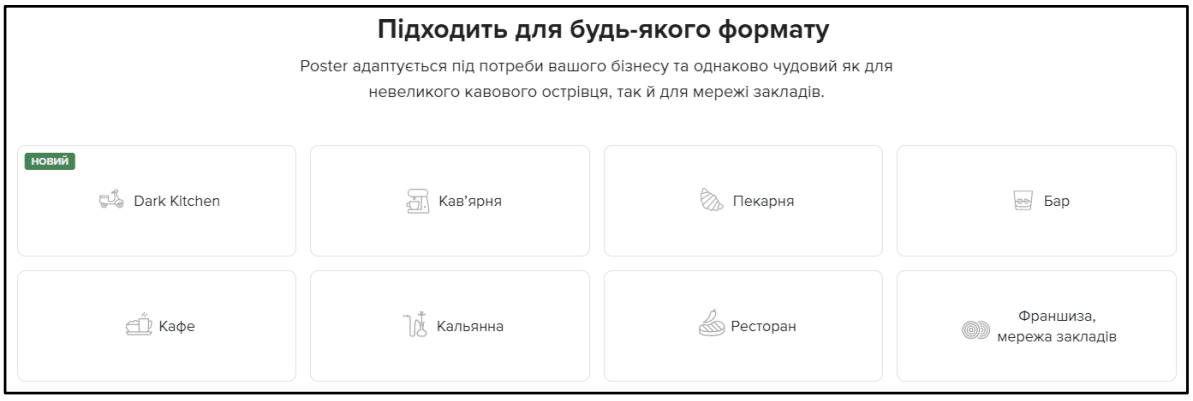
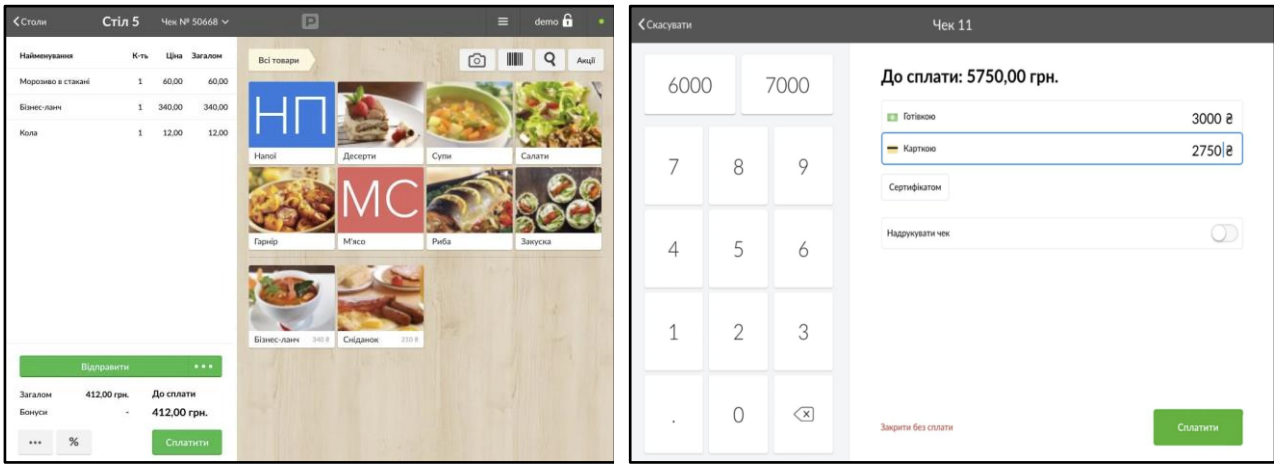


Рисунок 1.1 – перелік підприємст для застосування програми Poster POS

Poster POS можна назвати портативною касою. За допомогою додатку можна оформляти замовлення, переглядати меню, розраховувати клієнтів тощо. Приклад фунцій програми зображено на рисунку 1.2.



а) б)

Рисунок 1.2 – Функції Poster POS: а) оформлення замовлень; б) розрахунок

1.3.2 BAS Бухгалтерія

BAS Бухгалтерія - інструмент, який допомагає вирішити будь-які бухгалтерські завдання. Використовується для автоматизації бухгалтерського і податкового обліку, а також для регламентованої звітності. На рисунку 1.3 зображено приклад застосування BAS Бухгалтерії.

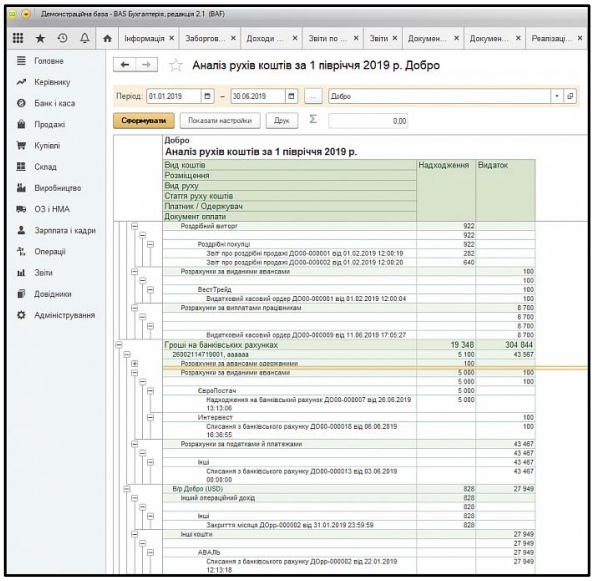


Рисунок 1.3 – Аналіз руху коштів в BAS Бухгалтерія

BAS Бухгалтерія включає такі підсистеми для вирішення задач:

- складський облік;
- облік торговельних операцій;

- облік розрахунків з контрагентами;
- ведення обліку діяльності кількох організацій;
- облік комісійної торгівлі;
- завершальні операції періоду;
- експрес-перевірка обліку;
- регламентована звітність.

BAS Бухгатерія якісна альтернатива на 1С Підприємству, що є російським продуктом.

1.3.4 RemOnline

RemOnline – програма для обліку товару. Застосунок підходить для багатьох типів бізнесу. Управління продажами в RemOnline дозволяє:

- контроль закупівель та оплати постачальників;
- відстеження стану та руху товарів;
- контроль роботи персоналу та складський облік;
- підключення онлайн-каси та інших інструментів продажу.

Деякі можливості програми зображено на рисунку 1.4.

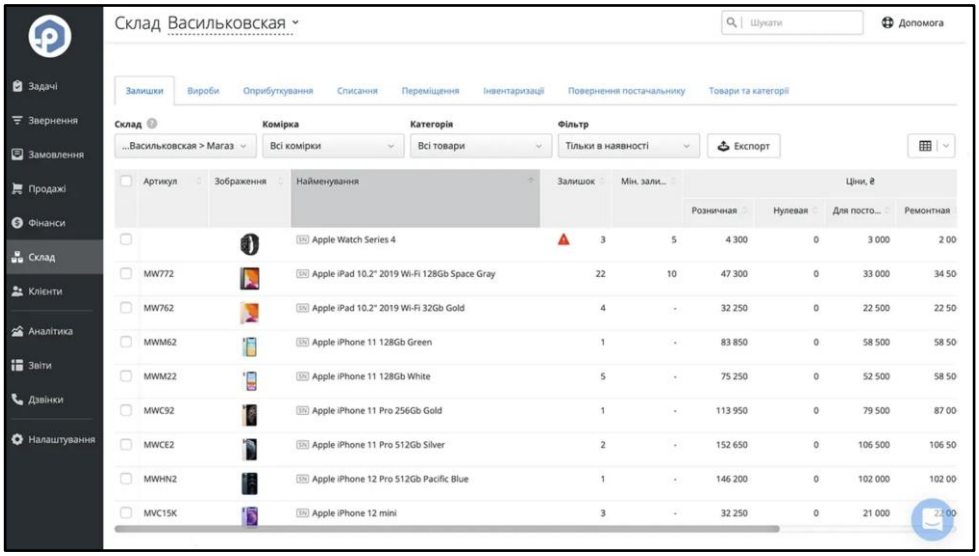


Рисунок 1.4 – Функціонал RemOnline

РОЗДІЛ 2. Проєктування реалізації системи Merchflow

2.1 Особливості програми

Умовним замовником програми буде компанія, яка займається закупівлею та реаізацією товарів. В даній компанії є співробітник, який відповідає за товарообіг та веде його вручну. Компанії потрібна програма, яка автоматизує облік товару, полегшить роботу та зменшить кількість помилок під час руху товарів.

Тому, під час розробки необхідно, щоб майбутня програма вирішила наступні задачі:

- створення картки товару із характеристиками самого товару;
- створення картки постачальників та редагування їх;
- створення замовлень товару постачальнику;
- створення документу надходження із документу замовлення;
- створення документів продаж за день та контроль залишків по товару;
- створення документу бронювання товару;
- створення документу реалізації бронювання;

2.2 Опис вимог до системи

Перш за все, програма системи товарообліку має контролювати залишки кожного товару. У разі можливих розходжень, програма має зстерігати користувача від помилок. Кожний рах товару записується в документ і контролюється системою. Система повинна забезпечувати можливість реєстрації, зберігання та оновлення інформації про товари на складі. Це включає атрибути товарів, такі як код, кольори, ціни, постачальники тощо. Кожен товар має бути унікально ідентифікований, щоб уникнути плутанини та помилок у процесі замовлення та реалізації. Система повинна також забезпечувати можливість швидкого пошуку та перегляду інформації про конкретний товар.

При допущенній помилці завжди є можливість виправити її. Розорблена програма повинна працювати у зв’зці із базоб даних, яка дозволяє швидко та безперебійно опрацьовувати великі об’єми інформації. Система товарообліку повинна бути здатна інтегруватися з касовою системою магазину. Це дозволяє автоматично оновлювати інформацію про продажі і залишки товарів. Кожен продаж фіксується в системі, що дозволяє вести точний облік проданих одиниць товару та підраховувати суми грошей, отриманих від продажу.

2.3 Проектування функціональних можливостей застосунку

Розроблюваний в даній бакалаврській дипломній роботі застосунок системи товарообліку, має наступні функціональні можливості:

- взаємодія з користуваче за допомогою кнопок та навігація за допомогою головного меню;
- відображення інформації необхідної для користувача при виконанні запитів;
- зрозумілий та простий інтерфейс функцій видатку, замовлення, внесення та створення товару.
- інтуїтивно зрозумілий та приємний інтерфейс;
- робота в офлайн режимі та застосунок, який працює на всіх версіях операційної системи Windows, починаючи із 7;
- детальне оповіщення корристувача про зроблені помилка та інструкції із вирішення їх.

Робота в програмі починається із головного меню, де розташовано логотип та перелік кнопок.

Кожний розділ з меню відповідає за різні функції, розглянемо їх:

1. «Каталог». При настиканні ви перейдете до катологу всього товару, який є в компанії. Адміністратор можу коригувати товар, створювати новий товар, вказуючи всі пааметри цього товару та видаляти товар, який більше не буде обліковуватись. Також користувач може робити пошук товару по найменуванню

товару та знаючи код постачальника, можливо знайти всі товари певного посатчальника.

2. «Постачальники». В цьому пункті ведеться облік всіх постачальників в базі даних. Адміністратор має можливість створити нових постачальників та видалити вже існуючого постачальника, маючи його код. За необхідності можливо знайти постачальника за його прізвищем. Також при створенні є перевірка на унікльність постачальника. Якщо в базі вже є постачальник, то іншого такого створити не можливо.

3. «Замовлення». Включає в себе перелік замовлень товару які вже створенні на постачальників. Замовлення можливо шушкати по коду постачальника, по коду самого замовлення та зробити відбір замовлень за дату, яку необхідно. На даній формі адміністратор має можливість створити нове замовлення, переглянути інформацію про товар, та створити надходження, вибравши замовлення, яке необхідно.

4. «Надходження товарів». В цьому пункті відображені всі документи надходжень товару від постачальника. Користувач може здійснювати пошук документів надходжень за номером, за кодом замовлення, за кодом товару та за кодом постачальника. Також користувач оже зробити відбір документів на дту, яку необхідно. Адміністратор має можливість створити новий документ надходжень, натиснувши кнопку «Створити новий документ надходження», де спочатку з’явиться форма вибору документа замовлення. А після вибору замовлення, з’явиться форма створення документу надходження із заповненими параметрами обраного документа замовленя. Після створення документу надходження, по товар буде зроблений рух прибутку на кількість, яка вказана в документі надходження.

5. «Продажі». За допомогою цього пункту адміністратор має можливість переглянути всі докумемнти продажів та застосувавувати відбор по даті та робити пошуки документів продажу. Пошук реалізовано до кодом продажі. Також адміністратор можле створити документ поверення продажу, вказавши код самого документу продажу. В такому випадку буде створено рух

оприбуткування товару на склад. Тако користувач може вносити продажі, де необхідно вказати код товару та найменування його, що реалізується вибором зі переліку товару. Також користува має внести кількість товару та ціну, по якій продано даний товар. При внесені кількісті, програма перевіряє залишок по товару і не дасть користувачеві створити документ продажу із кількістю більше ніж є на залишках. Сума розраховується автоматично. Тому помилок при розрахунку ціни не має бути. При додаванні документа продажі по відповідному товару буде створено рух видатку із залишків цього товару. Також із вікна продажів, користувач може реалізувати бронювання товару, натиснувши кнопку «Реалізувати бронь». Після натискання на дану кнопку з'являється вікно вибору документу бронювання, користувач має обрати документ, який не є реалізованим, а в статусі забронюванo, в іншому випадку, програма попередить користувача про некоректний вибір та закриє форму вибору документа бронювання. У разі коректного вибору документа, закриється вікно вибору бронювання та відккриється вікно створенн документу продажу зі всіма заповненими параметрами. При створенні документа продажу, буде виконано рух видатку товару із залишків.

6. «Бронювання». В даному пункту меню користувач може забронювати товар для клієнта, який зв'язався із оператором в телефонному режимі або через сайт та забронював товар. Користуваач створює документ бронювання товару. В певній кількості та вводить ціну, по якій буде продаватись товар. Обирати товар користувач буде через вікно вибору товару, а не вводити вручну, що є перевагою та виключає можливість вибору іншого товару. При створенні документу бронювання, буде створено рух бронювання товару із залишків. Та товар буде списано при реалізації документа бронювання вікна Продаж.

На рисунку 2.1 зображена діаграма варіантів використання, на якій можна побачити відносини між користувачем, застосунком та БД та варіанти взаємодії між ними.

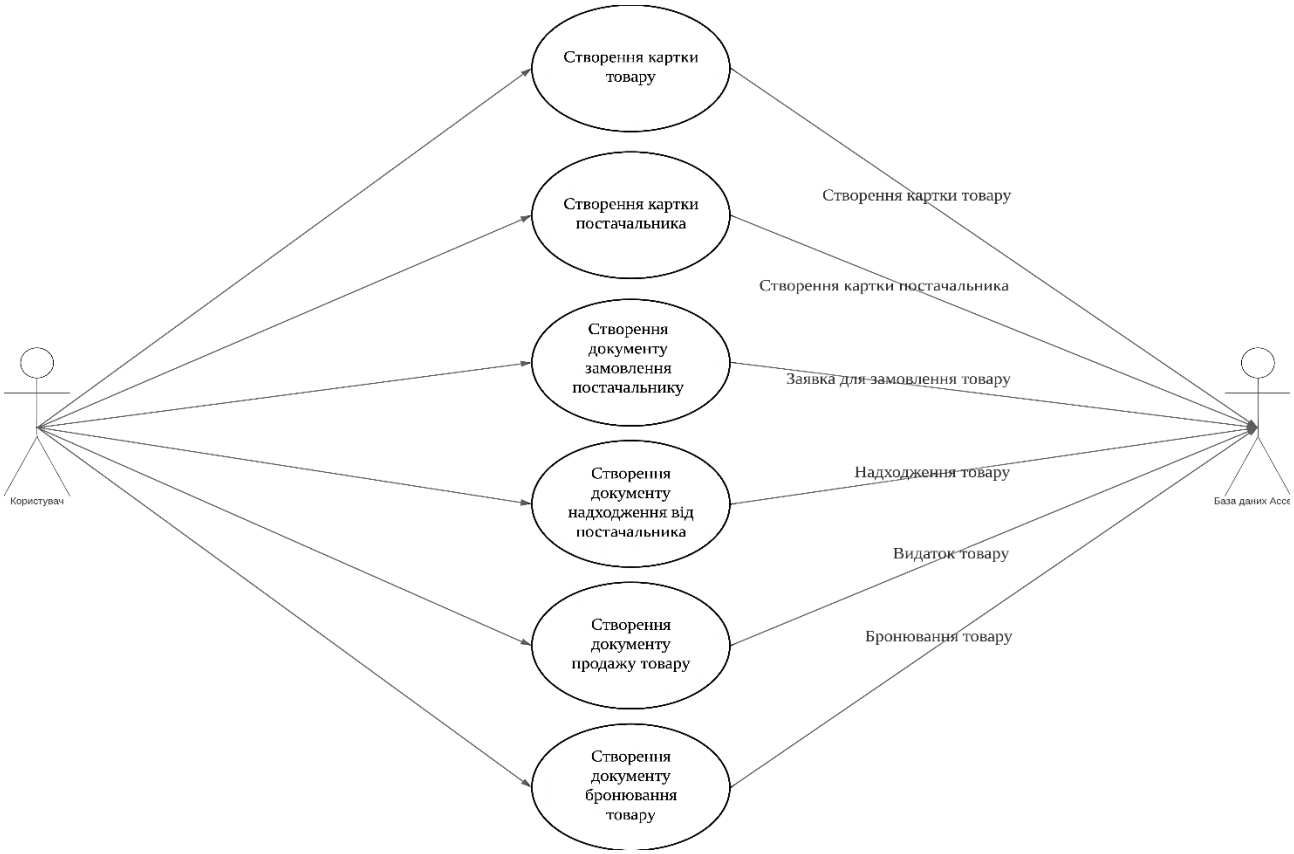


Рисунок 2.1 - Діаграма варіантів використання

Алгоритм роботи застосунку зображно на рисунку 2.2.

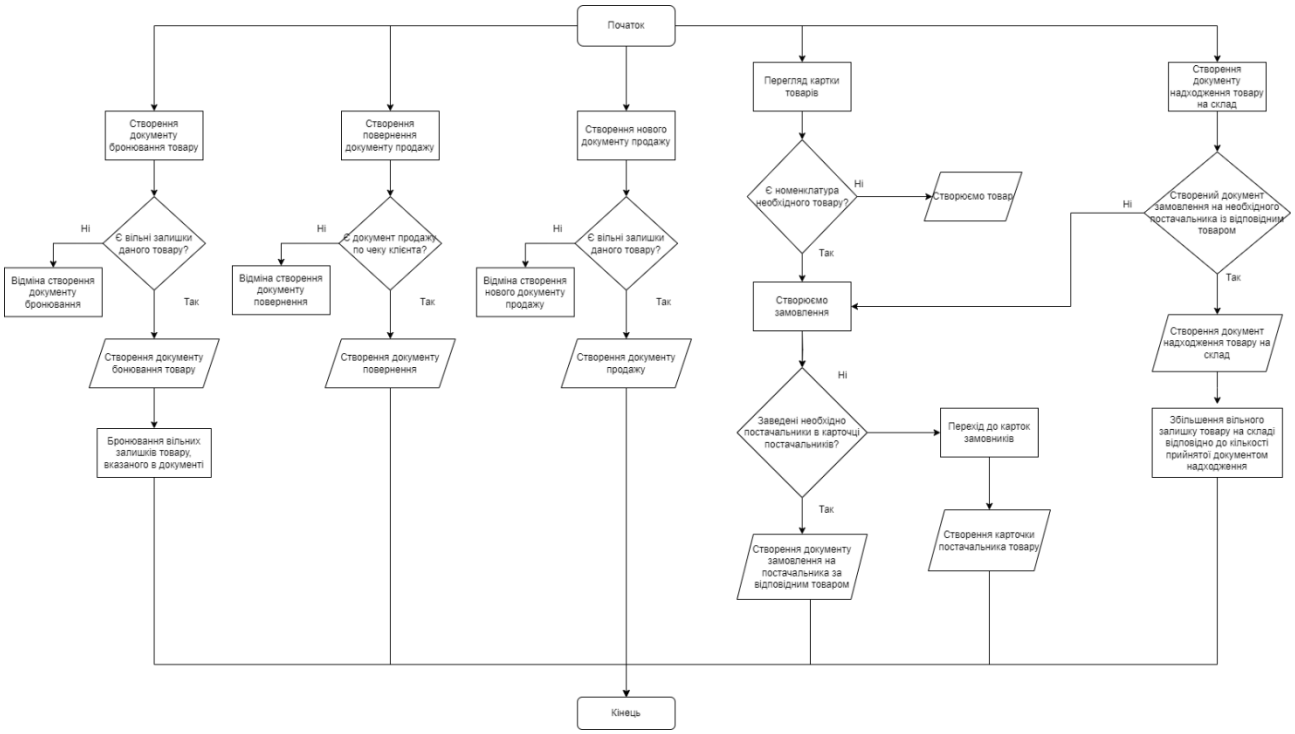


Рисунок 2.2 – Блок-схема застосунку

2.4 Вибір засобів розробки

2.4.1 Опис мови програмуваня

Одним з найважливіших завдань під час створення дипломного проекту є правильний вибір засобів розробки. В результаті аналізу мов програмування, які зазвичай використовуються для створення застосунків, визначено, що C++ є найбільш вдалим варіантом для даного проекту.

C++ (С плюс плюс) є скомпільованою, статично типізованою мовою програмування загального призначення. Вона була розроблена як розширення мови С і забезпечує вищий рівень абстракції, підтримуючи процедурне, об'єктно-орієнтоване та загальне програмування одночасно.

Зараз мова C++ широко використовується для розробки програмного забезпечення, будучи однією з найпопулярніших мов програмування. З її допомогою створюють операційні системи, різноманітні прикладні програми, драйвери пристроїв, ігри тощо. Мова C++ є мовою високого рівня і основою багатьох систем програмування.

C++ - мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної.

При створенні C++ прагнули зберегти сумісність з мовою С. Більшість програм на С справно працюватимуть і з компілятором C++.

C++ має синтаксис, заснований на синтаксисі С .

Нововведеннями C++ порівняно з С є:

- підтримка об'єктно-орієнтованого програмування через класи;
- підтримка узагальненого програмування через шаблони;
- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- простори імен;
- вбудовані функції;
- перевантаження операторів;

- перевантаження імен функцій;
- посилання і оператори управління вільно розподіленою пам'яттю.

Основні характеристики та особливості мови C++:

- Синтаксис: C++ використовує синтаксис, подібний до мови C, що дозволяє програмістам, які вже знайомі з C, легко освоїти нову мову. Однак C++ також додає нові конструкції та можливості, такі як класи та об'єкти.
- Об'єктно-орієнтоване програмування (ООП): C++ підтримує основні принципи ООП, включаючи інкапсуляцію, успадкування та поліморфізм. Класи дозволяють визначати користувацькі типи даних, які можуть містити змінні (члени даних) та функції (члени класу).
- Загальне програмування (шаблони): C++ надає можливість використовувати шаблони для створення загальних алгоритмів та контейнерів. Шаблони дозволяють створювати параметризовані типи даних та функції, які можуть працювати з різними типами даних без необхідності дублювання коду.
- Керування пам'яттю: У C++ програмісту надається контроль над керуванням пам'яттю. Він може явно виділяти та звільняти пам'ять за допомогою операторів new та delete, що дозволяє ефективно керувати ресурсами та уникати витоків пам'яті.
- Багатогілковість: C++ має вбудовану підтримку багатопотоковості за допомогою стандартної бібліотеки потоків (std::thread) та механізму синхронізації, таких як м'ютекси та умовні змінні. Це дозволяє створювати паралельні та конкурентні програми.
- Багата стандартна бібліотека: C++ має обширну стандартну бібліотеку, яка включає контейнери (наприклад, вектори, списки, множини), алгоритми сортування та пошуку, введення/виведення, роботу з файлами, рядками та багато іншого.

- Низькорівневий доступ: C++ дозволяє програмістам здійснювати низькорівневий доступ до пам'яті та працювати з покажчиками, що дає більшу гнучкість та контроль над виконанням програми.
- Переносимість: Мова C++ є переносимою між різними платформами, що дозволяє розробляти програми, які працюють на різних операційних системах та апаратних платформах.
- Широке використання: C++ є однією з найпоширеніших мов програмування та використовується в багатьох галузях, включаючи розробку ігор, вбудовані системи, наукові дослідження, фінансові програми та багато іншого.

Це лише короткий огляд основних характеристик мови C++. Вона має багато інших функцій та можливостей, які роблять її потужним інструментом для розробки різноманітних програмних проектів.

2.4.2 Середовище розробки

Для дипломної роботи було обрано середовище RAD Studio.

RAD Studio - це інтегроване середовище розробки програмного забезпечення (IDE), яке дозволяє розробникам створювати програми для різних платформ, включаючи Windows, macOS, iOS та Android. RAD Studio містить в собі широкий набір інструментів для розробки програм, які включають в себе текстовий редактор, візуальний редактор інтерфейсу користувача, інструменти для налагодження та тестування програм, а також компілятори для різних мов програмування, таких як Delphi, C++ і Object Pascal.

RAD Studio надає розробникам можливість працювати з різними платформами і мовами програмування, що дозволяє створювати багатоплатформні програми, зменшуючи витрати на розробку та підтримку. RAD Studio також має вбудовані інструменти для розробки мобільних додатків, які дозволяють розробникам створювати додатки для iOS та Android без необхідності використовувати окремі інструменти для кожної платформи.

Однією з ключових переваг RAD Studio є його візуальний редактор інтерфейсу користувача (UI), який дозволяє розробникам створювати відповідний UI для своїх додатків. RAD Studio має широкий набір компонентів UI, включаючи кнопки, поля введення, списки, таблиці та багато іншого, що дозволяє розробникам швидко створювати відповідний UI для своїх додатків.

RAD Studio також надає можливість розробникам використовувати різні бази даних, такі як MySQL, Oracle та Access, а також можливість підключення до веб-сервісів та інших джерел даних. Це дозволяє розробникам створювати програми, які можуть працювати з різними джерелами даних та виконувати різні завдання.

Окрім цього, RAD Studio має вбудований механізм для налагодження програм, який дозволяє розробникам швидко виявляти та виправляти помилки в своїх програмах. Середовище містить вбудовані інструменти для профілювання програм, що дозволяє розробникам виявляти та оптимізувати роботу своїх програм.

Крім того, RAD Studio надає розробникам можливість використовувати різні фреймворки та бібліотеки, , що дозволяє розробникам швидко створювати програми з відповідним UI та функціональністю.

RAD Studio також підтримує різні системи контролю версій, такі як Git та Subversion, що дозволяє розробникам ефективно працювати в команді та керувати версіями своїх програм.

Основна мета RAD Studio полягає в полегшенні розробки програмного забезпечення шляхом надання зручного та ефективного середовища, а також набору інструментів і компонентів, які допомагають зменшити час, затрачений на розробку і підтримку проектів.

Основні компоненти RAD Studio включають наступні елементи:

1. Delphi: RAD Studio є одним з основних компонентів RAD Studio і є потужним середовищем розробки, що базується на мові програмування Object Pascal. Delphi надає розробникам інструменти для швидкої розробки десктопних, мобільних та веб-додатків. Він пропонує зручний інтерфейс

користувача та багатофункціональну інтегровану розробку, включаючи редактор коду, візуальну розробку і налагоджувальний засіб. Delphi також підтримує велику кількість бібліотек та компонентів для швидкого створення функціональних додатків.

2. C++Builder: C++Builder використовує мову програмування C++, яка відома своєю потужністю та гнучкістю. C++ дозволяє розробникам створювати швидкі та ефективні додатки, а також працювати з низькорівневими операціями та прямою маніпуляцією пам'яттю. C++Builder надає візуальний редактор, що дозволяє створювати користувацький інтерфейс за допомогою методу перетягування та розміщення компонентів на формі додатку. Це спрощує процес створення графічного інтерфейсу та розміщення елементів керування. C++Builder дозволяє розробляти кросплатформенні додатки, що можуть працювати на різних операційних системах, таких як Windows, macOS, iOS та Android. Це досягається завдяки використанню фреймворку FireMonkey, який надає кросплатформенну підтримку та можливості.
3. FireMonkey: Це платформа для розробки кросплатформенних додатків, що дозволяє розробникам створювати один код та використовувати його для побудови додатків на різних платформах, таких як Windows, macOS, iOS та Android. FireMonkey дозволяє розробникам створювати додатки один раз і компілювати їх для різних платформ. Це забезпечує ефективну використання загального коду та ресурсів, що дозволяє зберегти час і зусилля, які зазвичай витрачаються на окрему розробку для кожної платформи. FireMonkey надає візуальний редактор для швидкої та зручної розробки користувацького інтерфейсу. Розробники можуть перетягувати та розміщувати компоненти на формі додатку, встановлювати їх властивості та взаємодіяти з ними. Це дозволяє швидко створювати привабливі та функціональні інтерфейси користувача. FireMonkey підтримує розробку як 2D, так і 3D графічних додатків. Він надає набір компонентів і інструментів для створення графічних об'єктів, анімації,

ефектів, переходів та інших візуальних елементів. Дозволяє легко працювати з мультимедійними ресурсами, такими як аудіо та відео. Він підтримує відтворення мультимедійних файлів, запис аудіо, відеозахоп, потокову передачу даних та багато іншого.

4. VCL (Visual Component Library): VCL є набором компонентів та класів, які дозволяють розробникам швидко побудувати десктопні додатки для платформи Windows. VCL надає широкий набір готових компонентів, які можна використовувати при розробці додатків. Ці компоненти включають кнопки, текстові поля, списки, таблиці, вкладки, меню, діалогові вікна, зображення та багато інших. Компоненти VCL можуть бути легко перетягнуті на форму додатку та налаштовані через інтерфейс користувача. VCL надає розробникам широкі можливості для налаштування та розширення функціональності компонентів. Розробники можуть змінювати зовнішній вигляд компонентів, встановлювати властивості, використовувати події та методи для реагування на взаємодію з користувачем. Компоненти VCL підтримують події, що дозволяють реагувати на дії користувача або зміни в програмі. Розробники можуть зв'язувати функції або процедури з подіями компонентів, що дозволяє виконувати певний код при виникненні певної події.

У загальному, RAD Studio - це потужне інтегроване середовище розробки програмного забезпечення, яке дозволяє розробникам створювати програми для різних платформ з відповідним UI та функціональністю, швидко виявляти та виправляти помилки та ефективно працювати в команді.

Деякі з основних можливостей RAD Studio включають:

- Візуальний редактор форм для швидкої розробки інтерфейсів користувача.
- Мови програмування, такі як Delphi, C++, і C#.
- Вбудована підтримка баз даних, включаючи SQLite, MySQL та Oracle.
- Можливість створювати мобільні додатки для Android і iOS.

- Інструменти для тестування та налагодження програмного забезпечення.
- Веб-розробка з підтримкою PHP, HTML, CSS та JavaScript.

2.4.3 База даних Access

Для виконання всіх поставлених задач, чат-бот повинен не тільки володіти інформацією та відображати її для користувача, а й зберігати введену інформацію користувачем. У даній дипломній бакалаврській роботі необхідно зберігати інформацію про людей, які взяли участь в заході та інформацію про тих хто взяв участь в опитуванні та їхній вибір. Тут і виникає потреба у тому, щоб вибрати найбільш зручний та сучасний спосіб для зберігання великого обсягу особистої інформації. В результаті аналізу цього питання, було обрано Access, компактна вбудована база даних, вихідний код якої був переданий в суспільне надбання. Але перед початком роботи з нею, необхідно ознайомитись з основними поняттями.

База даних (БД) - це централізоване сховище даних для зберігання, доступу, первинної обробки та отримання інформації. Це набір впорядкованих логічно взаємопов’язаних даних, якими можна спільно користуватися та призначених для задоволення інформаційних потреб користувачів.

База даних повинна містити:

- незалежність даних;
- відкритий доступ до даних;
- підтримка операцій, забезпечених відповідними властивостями;
- гарантування, що немає несправностей;
- робота з багатьма користувачами одночасно.

Microsoft Access – це система управління реляційними базами даних (СУБД), призначена для роботи на автономному персональному комп’ютері (ПК) або в локальній обчислювальній мережі під управлінням сімейства операційних систем Microsoft Windows.

СУБД- це сукупність мовних та програмних засобів, які призначено для створення, ведення, спільного використання баз даних багатьма користувачами. Популярність СУБД Microsoft Access обумовлена наступними причинами:

- доступність у вивченні й зрозумілість дозволяють Access бути однієї із кращих систем швидкого створення додатків керування базами даних;
- можливість використання OLE технології;
- інтегрованість із пакетом Microsoft Office;
- повна підтримка Web-Технологій;
- візуальна технологія дозволяє постійно бачити результати своїх дій і коректувати їх;
- наявність великого набору «майстрів» по розробці об’єктів.

Microsoft Access є реляційною СУБД із табличною структурою даних. Це означає, що таблиці являють собою основний об’єкт збереження даних.

Наприклад, одна таблиця може містити дані про товари, інша – про виробників, третя – про постачальників цих товарів і т.д.

Ці окремі таблиці зв’язуються воєдино. Комбінація всіх таблиць і їхніх взаємних зв’язків складає «фундамент» бази даних.

СУБД Microsoft Access має потужні, зручні засоби візуального проектування об’єктів за допомогою Майстрів, що дозволяє користувачеві при мінімальній попередній підготовці досить швидко створити повноцінну інформаційну систему на рівні таблиць, запитів, форм і звітів.

До основних можливостям СУБД Microsoft Access можна віднести наступні:

- проектування базових об’єктів – двовимірні таблиці з полями різних типів даних;
- створення зв’язків між таблицями, з підтримкою цілісності даних, каскадного відновлення полів і каскадного видалення записів;

- введення, зберігання, перегляд, сортування, зміна й вибірка даних з таблиць із використанням різних засобів контролю інформації, індексування таблиць і апарата алгебри логіки;

- створення, модифікація й використання похідних об’єктів (запитів, форм і звітів).

РОЗДІЛ 3. Програмна реалізація системи Merchflow

3.1 Створення бази даних Access

Перший крок в розробці системи товарообігу MerchFlow – створення бази даних Access, де будуть зберігатися та опрацьовуватись всі дані. Створимо необхідні таблиці та відповідні поля в таблицях (рисунок 3.1.)

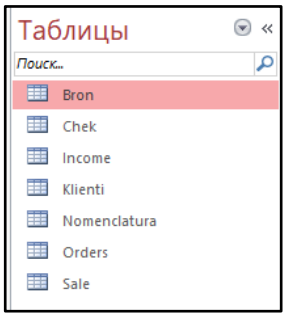


Рисунок 3.1 Створення таблиць та полів із данними в них

Після створення таблиць та реквізитів, необхідно створити схему компоновки даних (рисунок 3.2.). Додамо всі таблиці до схеми та налаштуємо зв’язки між ними для цілісності данних так щоб обирати елементи одних таблиць в інших.

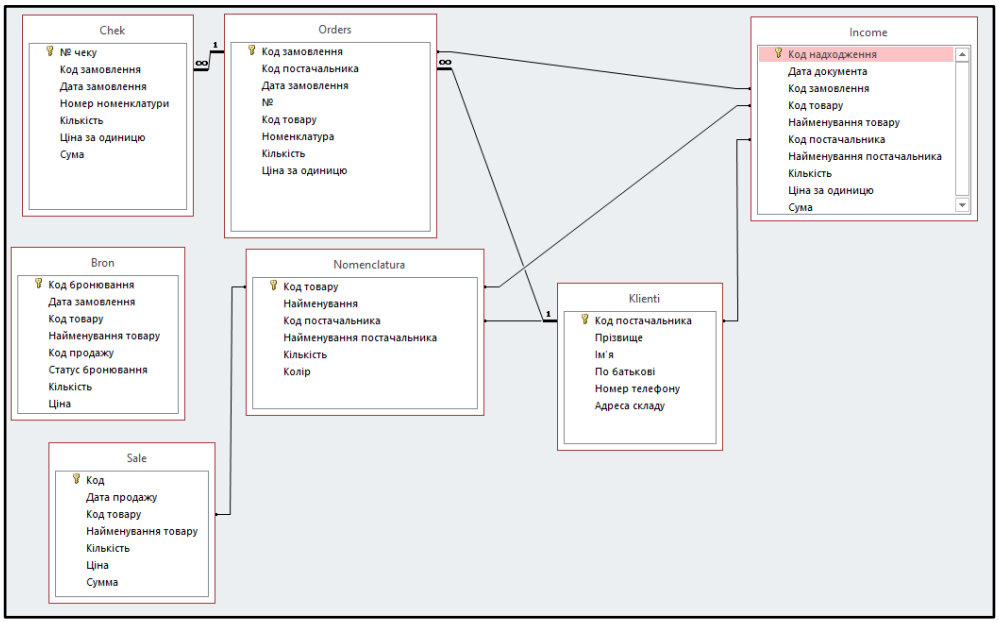


Рисунок 3.2. Схема компоновки даних

Після створення бази даних, переходимо до створення проєкту в застосунку RAD Studio C++ Builder.

3.2 Реалізація програмної частини

Проект складається з файлів, які зображені на рисунку 3.3, що утворюють програмну реалізацію застосунку системи товароболіку магазину одягу.

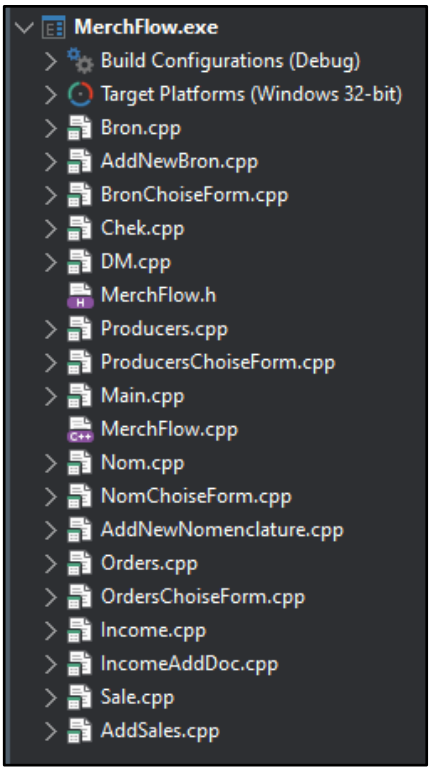


Рисунок 3.3. Файлова структура застосунку системи товароболіку магазину одягу

Спочатку треба підключити до проєкту базу даних. Використовується для цього компонент ADOConnection із типом підключення OLE. Також за допомогою компоненти ADOTable визначаємо таблиці бази данних, для того, щоб звертатись до них та обробляти інформацію з бази даних. Компонент DataSource необхідний для представлення даних в таблицях застосунку і він має відповідати одній тваблиці бази данних (рисунок 3.4)

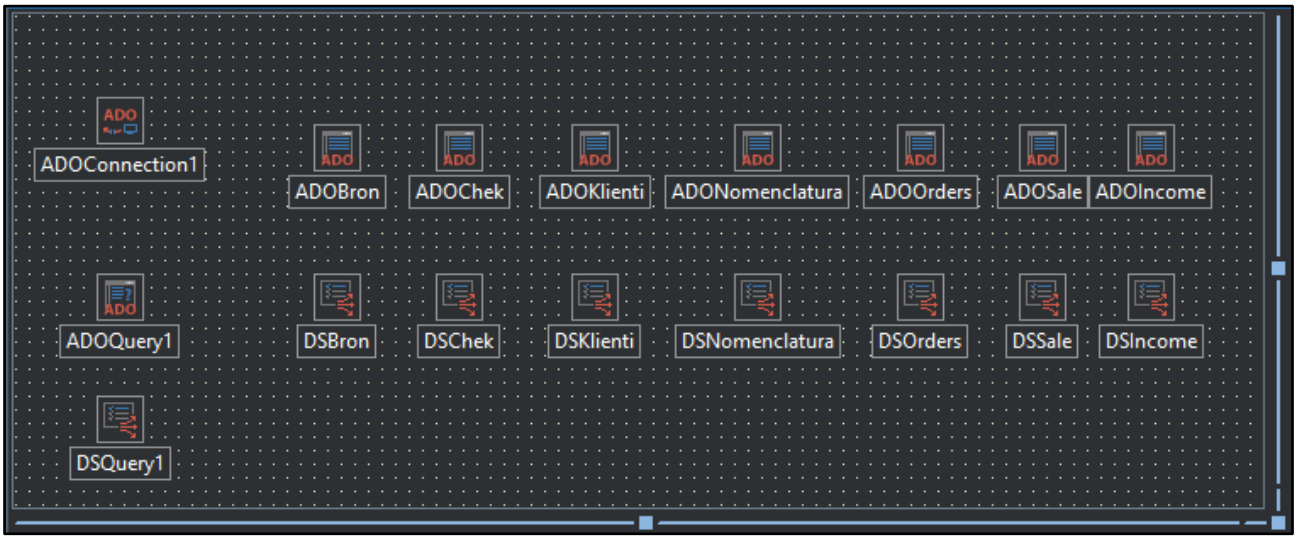


Рисунок 3.4. Підключення бази даних та визначення її таблиць.

3.2.1 Створення з’єднання із базою данних.

Для початку необхідно створити ConnectionString в компоненті ADOConnection, де обираємо Microsoft Jet 4.0 OLE DB Provider та обираємо базу данних (рисунок 3.4.)

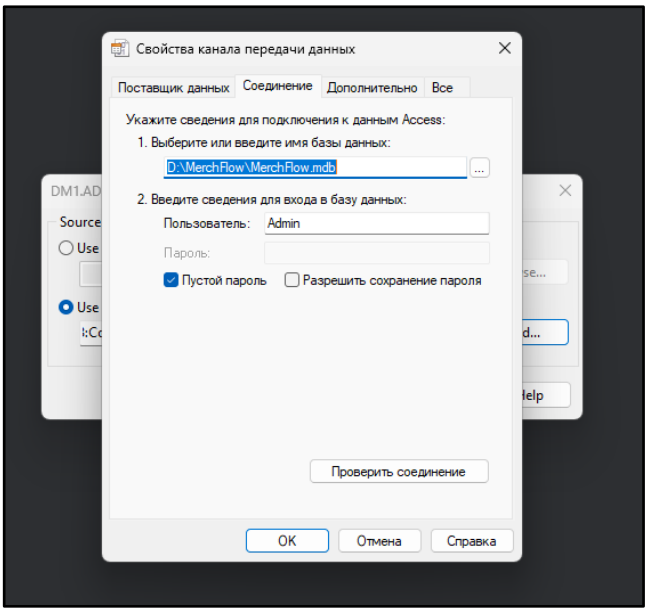


Рисунок 3.5. Вибір розташування бази даних для підключення її до проєкту

3.2.2 Розробка головного меню та його складових

При запуску застосунку користувачу відкриється вікно із наступними кнопками:

- «Каталог»;
- «Постачальники»;
- «Замовлення»;
- «Надходження товарів».
- «Продажі»
- «Бронювання»

Для того, щоб перейти на форму обраного користувачем розділу, необхідно використати функцію ShowModal(). Завдяки цій функції. Користувач поки не закінчить роботу у відкритому вікні, не зможе повернутись до початкового меню.

Кожен розділ має власні функції. Розділ «Каталог» має функції пошуку, додавання та видалення інформації.

На формі розділу є таблиця, в яку завантажуються данні з відповідної таблиці бази даних. Програмно це виглядає на рисунку 3.6

```
DBGrid1->DataSource=DM1->DSNomenclatura;
```

Рисунок 3.6. Заповлення табличного поля DBGrid даними із відповідної таблиці

Для виконання SQL запиту необхідно використовувати компоненту ADOQuery.

ADOQuery є компонентом в Rad Studio, який використовується для виконання SQL запитів до бази даних з використанням технології ADO (ActiveX Data Objects). Він дозволяє взаємодіяти з базою даних, виконувати запити, отримувати результати і обробляти дані.

ADOQuery має широкий набір методів, які дозволяють працювати з SQL запитами та отримувати результати. Ось декілька основних методів ADOQuery:

1. **Open**: Виконує SQL запит та відкриває набір результатів. Цей метод виконує запит до бази даних та отримує результати, які потім можна обробити.

2. **Close**: Закриває набір результатів. Цей метод закриває відкритий набір результатів та звільняє ресурси, пов'язані з ним.
3. **ExecSQL**: Виконує SQL запит, який не повертає результатів. Цей метод використовується для виконання SQL запитів, які не повертають набір результатів, наприклад, для оновлення даних у базі даних.
4. **First**: Переміщає курсор на перший запис у наборі результатів.
5. **Next**: Переміщає курсор на наступний запис у наборі результатів.
6. **Eof**: Перевіряє, чи досягнуто кінця набору результатів (End of File).
7. **FieldByName**: Отримує значення поля за його назвою. Цей метод дозволяє отримати значення поля з поточного запису набору результатів за назвою поля.
8. **ParamByName**: Отримує доступ до параметра запиту за його назвою. Цей метод дозволяє отримувати доступ до параметрів SQL запиту та задавати їх значення.
9. **CreateParam**: Створює новий параметр для SQL запиту. Цей метод дозволяє створити новий параметр та налаштувати його властивості.

ADOQuery є одним з компонентів, доступних у Rad Studio, і він використовується для взаємодії з базами даних за допомогою технології ADO (ActiveX Data Objects). ADOQuery дозволяє виконувати SQL-запити до бази даних, отримувати результати і працювати з ними.

ADOQuery є представником набору результатів (Recordset) у базі даних. Він містить в собі відкрите з'єднання з базою даних і може виконувати SQL-запити для отримання даних з таблиць або виконання інших дій, таких як вставка, оновлення або видалення даних.

ADOQuery має ряд властивостей і методів, які дозволяють налаштовувати його поведінку і взаємодіяти з базою даних. Основні властивості ADOQuery включають:

1. **Connection**: Ця властивість визначає з'єднання з базою даних, з якою ви хочете взаємодіяти. Ви можете встановити цю властивість на об'єкт

TADOConnection або використовувати існуюче з'єднання, встановлене в дизайнері.

2. **SQL**: Ця властивість дозволяє визначити SQL-запит, який буде виконуватися. Ви можете ввести SQL-запит безпосередньо у властивість або використовувати візуальний редактор запитів для його створення.
3. **Parameters**: Ця властивість дозволяє визначити параметри для SQL-запиту. Ви можете додати параметри до запиту і задати їх значення під час виконання.
4. **Prepared**: Ця властивість вказує, чи буде SQL-запит підготовлений перед виконанням. Якщо ви встановите її в значення True, ADOQuery попередньо підготує запит, що може збільшити продуктивність при виконанні багатьох запитів.
5. **Active**: Ця властивість вказує, чи активний набір результатів ADOQuery. Якщо ви встановите її в значення True, ADOQuery виконає SQL-запит і отримає результати в набірі результатів. Якщо ви встановите її в значення False, набір результатів буде закритий і дані будуть недоступні.
6. **RecordCount**: Ця властивість повертає кількість записів у наборі результатів ADOQuery. Ви можете використовувати цю властивість для отримання загальної кількості записів, що відповідають SQL-запиту.
7. **Eof**: Ця властивість вказує, чи поточний запис ADOQuery є останнім записом в наборі результатів. Вона повертає значення True, якщо поточний запис є останнім, і False - в іншому випадку.
8. **Bof**: Ця властивість вказує, чи поточний запис ADOQuery є першим записом в наборі результатів. Вона повертає значення True, якщо поточний запис є першим, і False - в іншому випадку.
9. **First()**: Цей метод переміщує поточний запис до першого запису в наборі результатів.
10. **Next()**: Цей метод переміщує поточний запис до наступного запису в наборі результатів

Пошук товару виконується за допомогою запитів мовою SQL. SQL SELECT - це запит, що використовується для вибірки даних з таблиці або набору таблиць у базі даних. У Microsoft Access, який є системою керування базами даних (СУБД), SQL SELECT використовується для отримання даних з таблиць:

SELECT колонка1, колонка2, ... FROM назва таблиці

WHERE умова;

1. SELECT: Вказуємо стовпці, які ми хочемо вибрати з таблиці. Ви можете вказати конкретні стовпці, розділені комами, або використовувати * для вибору всіх стовпців.
2. FROM: Вказуємо таблицю або набір таблиць, з яких ми хочемо отримати дані. Ви можете вказати одну або кілька таблиць.
3. WHERE: Необов'язкова частина запиту, яка використовується для встановлення умов відбору рядків. Ми можемо використовувати оператори порівняння (=, <>, >, <, >=, <=) та логічні оператори (AND, OR) для вказівки критеріїв відбору. Приклад функції пошуку зображено на рисунку 3.7.

```
if(Edit9->Text.IsEmpty())
{
    String message = "Введіть код замовлення.";
    ShowMessage(message);
    return;
}
DM1->ADOQuery1->Close();
DM1-> ADOQuery1->SQL->Clear();
DM1 ->ADOQuery1->SQL->Add( "select * FROM [Orders]");
DM1->ADOQuery1->SQL->Add("where [Orders].[Код замовлення] =code2");
DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(Edit9->Text);
DM1->ADOQuery1->Prepared=True;
DM1->ADOQuery1->Open();

if(DM1->ADOQuery1->IsEmpty())
{
    String message = "За даним кодом нічого не знайдено";
    ShowMessage(message);
    return;
}
else
{
    DBGrid1->DataSource = DM1->DSQuery1;
    DBNavigator1->DataSource = DM1->DSQuery1;
}
```

Рисунок 3.7. Пошук за значенням в таблиці бази даних та заповненя даними табличне поле

Для того, щоб додати новий запис до бази даних використовується метод Insert() та Post().

У Rad Studio, методи **Insert()** та **Post()** використовуються для роботи з наборами даних (dataset), зокрема з компонентами TDataSet та його похідними, такими як TTable, TQuery або TClientDataSet. Ці методи дозволяють додавати нові записи до набору даних та зберігати зміни, внесені до існуючих записів.

1. Метод **Insert()**: Цей метод використовується для додавання нового запису до набору даних. Коли ви викликаєте метод **Insert()**, новий порожній запис додається до набору даних, і ви можете заповнити його значеннями полів. Цей запис залишається у "режимі редагування", тобто ви можете змінювати значення полів, поки не викличете метод **Post()** для збереження змін.

2. Метод **Post()**: Цей метод використовується для збереження змін, внесених до існуючого запису набору даних. Коли ви викликаєте метод **Post()**, зміни, внесені до активного запису, фіксуються і зберігаються у базі даних або в пам'яті, залежно від типу набору даних. Після виклику **Post()** запис переходить у "режим перегляду", і його дані залишаються незмінними, поки ви не розпочнете нову операцію редагування. Приклад додавання запису до бази даних зображено на рисунку 3.8.

```
DM1->ADONomenclatura->Insert();
DM1->ADONomenclatura->FieldByName("Код товару")->AsInteger = StrToInt(Edit1->Text);
DM1->ADONomenclatura->FieldByName("Найменування")->AsString = Edit2->Text;
DM1->ADONomenclatura->FieldByName("Колір")->AsString = Edit3->Text;
DM1->ADONomenclatura->FieldByName("Код постачальника")->AsInteger = StrToInt(Edit4->Text);
DM1->ADONomenclatura->FieldByName("Найменування постачальника")->AsString = Edit5->Text;
DM1->ADONomenclatura->Post();
```

Рисунок 3.8. -Додавання запису до таблиці бази даних

Видалення запису із бази даних виконується SQL запитом до бази даних. Видалення відбувається за кодом запису, в данному випадку, за кодом товару. SQL DELETE використовується для видалення записів з таблиці бази даних. В Microsoft Access ви можете використовувати SQL DELETE запит для видалення даних з таблиць.

DELETE FROM table_name

WHERE condition;

1. DELETE FROM: Ця частина вказує, що ми хочемо видалити записи з таблиці.

2. table_name: Вкажіть назву таблиці, з якої ви хочете видалити записи.

3. WHERE: Ця частина є необов'язковою і використовується для встановлення умов видалення. Ви можете вказати умови, за якими будуть видалені певні записи. Наприклад, ви можете видалити записи, де значення певного стовпця відповідає певному критерію. Приклад видалення запису із бази даних зображено на рисунку 3.9.

```
if (MessageBox(NULL, L"Ви дійсно хочете виконати цю дію?", L"Підтвердження", MB_YESNO) == IDYES)
{
    DM1->ADOQuery1->SQL->Clear();
    DM1->ADOQuery1->SQL->Add("delete from [Nomenclatura]");
    DM1->ADOQuery1->SQL->Add("where [Nomenclatura].[Код товару] = "+DBEdit6->Text+";");
    DM1->ADOQuery1->ExecSQL();

    DM1->ADONomenclatura->Close();
    DM1->ADONomenclatura->Open();
}
else
{
    return;
}
```

Рисунок 3.9. Видалення запису з бази даних за кодом.

Для того, щоб оновити реквізит в таблиці використовуються SQL запит до бази даних. Пошук необхідноо елемента відбувається за унікальним кодом елемента та вказується нове значення або нове значення визначається за допомогою виразів. SQL UPDATE використовується для оновлення записів в таблиці бази даних. В Microsoft Access ви можете використовувати SQL UPDATE запит для зміни даних в таблиці.

UPDATE table_name SET column1 = value1, column2 = value2,

WHERE condition;

1. UPDATE: Ця частина вказує, що ми хочемо оновити записи в таблиці.

2. SET: Вказуємо стовпці, які потрібно змінити, та їх нові значення. Ви можете вказати багато стовпців, розділених комами.

3. WHERE: Ця частина є необов'язковою і використовується для встановлення умов оновлення. Ви можете вказати умови, за яких будуть оновлені певні записи. Наприклад, ви можете оновити записи, де значення певного стовпця відповідає певному критерію.

SQL UPDATE може оновлювати лише один запис, якщо використовується унікальна умова (наприклад, ID запису). У цьому випадку будуть змінені значення лише в одному записі. SQL UPDATE також може оновлювати кілька записів одночасно, якщо вказана умова вибирає більше одного запису. У цьому випадку всі відповідні записи будуть оновлені зазначеними значеннями.

SQL UPDATE може використовувати підзапити для вибору значень, які потрібно оновити. Це дозволяє виконувати складніші логічні перевірки або обчислення перед оновленням даних. В багатьох СУБД, включаючи Access, SQL UPDATE може бути використаний для пакетного оновлення даних з однієї таблиці в іншу. Наприклад, ви можете оновити значення в полі однієї таблиці на основі значень з іншої таблиці, використовуючи зв'язок між ними.

При використанні SQL UPDATE ви можете також використовувати функції і вирази для обчислення нових значень для оновлення. Наприклад, ви можете використовувати математичні операції, рядкові функції, функції дати та часу тощо, щоб змінити значення полів. SQL UPDATE може бути включений у транзакцію, що дозволяє вам забезпечити цілісність даних і забезпечити, що зміни будуть збережені або відкочені як єдиний блок операцій. При виконанні SQL UPDATE СУБД може перевіряти обмеження, такі як обмеження цілісності, унікальність, зовнішні ключі тощо. Це дозволяє гарантувати, що дані відповідають встановленим правилам і обмеженням.

SQL UPDATE є потужним інструментом для модифікації даних у базі даних. Він дозволяє оновлювати один або кілька записів з використанням визначених умов і нових значень. Ця операція є важливою для забезпечення

актуальності даних і виконання змін в базі даних залежно від потреб вашого додатку чи системи.

Приклад коду, де збільшується кількість товару на залишках на рисунку 3.10.

```
if(Label10->Caption == "0")
{
    DM1->ADOQuery1->Close();
    DM1->ADOQuery1->SQL->Clear();
    DM1->ADOQuery1->SQL->Add("update [Nomenclatura] set [Кількість] = [Кількість] - quantity");
    DM1->ADOQuery1->SQL->Add("Where [Nomenclatura].[Код товару] = code2");
    DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(Edit2->Text);
    DM1->ADOQuery1->Parameters->ParamByName("quantity")->Value=StrToInt(Edit4->Text);
    DM1->ADOQuery1->ExecSQL();

    DM1->ADONomenclatura->Close();
    DM1->ADONomenclatura->Open();
}
```

Рисунок 3.10. Зміна кількості товару в елементі за кодом елемента.

Розділ «Каталог» є інформаційним, який містить в собі довідникову інформацію про весь асортимент товару, який є на залишках в компанії та відображає залишок кількості товару. В цьому розділі адмініструється новий торвар або видалення вже не потрібного товару.

Розділ «Постачальники» також є інофрмаційним, який містить в собі інформацію про всіх постачальників та містить можливості адміністрування записів, тобто додавання нових, редагування та видалення вже існуючих елментів. Код прогами пошуку,додавання та видалення наведено на рисунку 3.11 та рисунку 3.12.

Метод **ShowMessage** є функцією в Rad Studio (Delphi або C++Builder), яка використовується для відображення повідомлення користувачу у вигляді модального діалогового вікна. Воно зазвичай використовується для відображення повідомлень про помилки, попереджень або інформаційних повідомлень, які повинні привернути увагу користувача.

Метод **ShowMessage** приймає один параметр **Msg**, який вказує текст повідомлення, яке має бути відображено.

Основні особливості методу **ShowMessage**:

1. **Модальне вікно**: **ShowMessage** відображає повідомлення у вигляді модального діалогового вікна, що означає, що користувач повинен

натиснути кнопку "OK" або закрити вікно, щоб продовжити виконання програми.

2. **Текст повідомлення**: Можливо передавати різні текстові повідомлення до методу **ShowMessage**. Вони можуть містити інформацію про помилки, попередження або просто повідомлення для користувача.
3. **Використання змінних**: Якщо потрібно вставити значення змінних у текст повідомлення, можливо використовувати форматування рядків.
4. **Використання повідомлень з вибором**: В Rad Studio також доступні розширені варіанти **ShowMessage**, які дозволяють використовувати додаткові параметри для створення вибіркових повідомлень з кнопками "OK", "Cancel" або іншими. Наприклад, **MessageDlg** або **MessageDlgPos** функції.

```
//Перевірки заповнення та коректного введення
//---------------------------------------------------------------------------------------------------

DM1->ADOQuery1->Close();
DM1->ADOQuery1->SQL->Clear();
DM1->ADOQuery1->SQL->Add("select [Код] From [Sale]");
DM1->ADOQuery1->SQL->Add("Where [Sale].[Код] = code1");
DM1->ADOQuery1->Parameters->ParamByName("code1")->Value=StrToInt(Edit1->Text);
DM1->ADOQuery1->Prepared=True;
DM1->ADOQuery1->Open();
if(!DM1->ADOQuery1->IsEmpty())
{
    String message = "Знайдено запис із даним кодом продажі, будь ласка введіть наступний";
    ShowMessage(message);
    return;
}
DM1->ADOQuery1->Close();
if(Label10->Caption == "0")
{
    if(Edit4->Text.IsEmpty() || StrToInt(Edit4->Text) > StrToInt(Label9->Caption))
    {
        String message = "Ви не ввели кількість, або ввели кількість більшу за доступну.";
        ShowMessage(message);
        return;
    }
}
if(Edit1->Text.IsEmpty() || Edit3->Text.IsEmpty() || Edit4->Text.IsEmpty() || Edit5->Text.IsEmpty()
    || Edit6->Text.IsEmpty() || MaskEdit1->Text.IsEmpty() || Edit2->Text.IsEmpty())
{
    String message = "Не всі параметри заповнено, прохання заповнити всі параметри.";
    ShowMessage(message);
    return;
}

//---------------------------------------------------------------------------------------------------

//Проведення продажі. Додавання запису в таблицю продажів
DM1->ADOSale->Insert();
DM1->ADOSale->FieldByName("Код")->AsInteger = StrToInt(Edit1->Text);
DM1->ADOSale->FieldByName("Дата продажу")->AsDateTime = StrToDate(MaskEdit1->Text);
DM1->ADOSale->FieldByName("Код товару")->AsInteger = StrToInt(Edit2->Text);
DM1->ADOSale->FieldByName("Найменування товару")->AsString = Edit3->Text;
DM1->ADOSale->FieldByName("Кількість")->AsInteger = StrToInt(Edit4->Text);
DM1->ADOSale->FieldByName("Ціна")->AsInteger = StrToInt(Edit5->Text);
DM1->ADOSale->FieldByName("Сумма")->AsInteger = StrToInt(Edit6->Text);
DM1->ADOSale->Post();

//Пошук та редагування кількосмі товару.
// 0 - Створюємо звичайний продаж, 1 - реалізація бронювання(списується товар при створені броні).
if(Label10->Caption == "0")
{
    DM1->ADOQuery1->Close();
    DM1->ADOQuery1->SQL->Clear();
    DM1->ADOQuery1->SQL->Add("update [Nomenclatura] set [Кількість] = [Кількість] - quantity");
    DM1->ADOQuery1->SQL->Add("Where [Nomenclatura].[Код товару] = code2");
    DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(Edit2->Text);
    DM1->ADOQuery1->Parameters->ParamByName("quantity")->Value=StrToInt(Edit4->Text);
    DM1->ADOQuery1->ExecSQL();

    DM1->ADONomenclatura->Close();
    DM1->ADONomenclatura->Open();
```

Рисунок 3.11. Додавання запису до бази даних та редагування вже існуючого запису

```
DM1->ADOQuery1->Close();
DM1->ADOQuery1->SQL->Clear();
DM1->ADOQuery1->SQL->Add("select * From [Sale]");
DM1->ADOQuery1->SQL->Add("Where [Sale].[Дата продажу] = date");
DM1->ADOQuery1->Parameters->ParamByName("date")->Value=StrToDate(MaskEdit1->Text);
DM1->ADOQuery1->Prepared=True;
DM1->ADOQuery1->Open();
DBGrid1->DataSource = DM1->DSQuery1;

//---------------------------------------------------------------------------
d __fastcall TForm8::Button3Click(TObject *Sender)

MaskEdit1->Text = "";
DBGrid1->DataSource=DM1->DSSale;

//---------------------------------------------------------------------------
d __fastcall TForm8::Button4Click(TObject *Sender)

DM1->ADOQuery1->Close();
DM1->ADOQuery1->SQL->Clear();
DM1->ADOQuery1->SQL->Add("select * From [Sale]");
DM1->ADOQuery1->SQL->Add("Where [Sale].[Код] = code");
DM1->ADOQuery1->Parameters->ParamByName("code")->Value=StrToInt(Edit1->Text);
DM1->ADOQuery1->Prepared=True;
DM1->ADOQuery1->Open();

if(!DM1->ADOQuery1->IsEmpty())
{
    DBGrid1->DataSource = DM1->DSQuery1;
}else
{
    String message = "По данному коду нічого не знайдено.";
    ShowMessage(message);
    return;
}
```

Рисунок 3.12. Пошук за кодом елемента та відбір по даті створення документа

В розділі «Продажі» при створенні нового запису до бази даних, в карточці товару таблиці Каталог списується кількість проданої номенклатури. А при поверненні продажу в карточці товару кількість плюсується. Приклад програмного коду відображено на рисунку 3.13, 3.14 та 3.15.

```
//Перевірки заповнення та коректного введення
//---------------------------------------------------------------------------

DM1->ADOQuery1->Close();
DM1->ADOQuery1->SQL->Clear();
DM1->ADOQuery1->SQL->Add("select [Kod] From [Sale]");
DM1->ADOQuery1->SQL->Add("Where [Sale].[Kod] = code1");
DM1->ADOQuery1->Parameters->ParamByName("code1")->Value=StrToInt(Edit1->Text);
DM1->ADOQuery1->Prepared=True;
DM1->ADOQuery1->Open();
if(!DM1->ADOQuery1->IsEmpty())
{
    String message = "Знайдено запис із даним кодом продажі, будь ласка введіть наступний";
    ShowMessage(message);
    return;
}
DM1->ADOQuery1->Close();
if(Label10->Caption == "0")
{
    if(Edit4->Text.IsEmpty() || StrToInt(Edit4->Text) > StrToInt(Label9->Caption))
    {
        String message = "Ви не ввели кількість, або ввели кількість більшу за доступну.";
        ShowMessage(message);
        return;
    }
}
if(Edit1->Text.IsEmpty() || Edit3->Text.IsEmpty() || Edit4->Text.IsEmpty() || Edit5->Text.IsEmpty()
    || Edit6->Text.IsEmpty() || MaskEdit1->Text.IsEmpty() || Edit2->Text.IsEmpty())
{
    String message = "Не всі параметри заповнено, прохання заповнити всі параметри.";
    ShowMessage(message);
    return;
}

//---------------------------------------------------------------------------
```

Рисунок 3.13. Перевірка корректності вводу даних та створення нового запису до таблиці продажі

```
//Проведення продажі. Додавання запису в таблицю продажів
DM1->ADOSale->Insert();
DM1->ADOSale->FieldByName("Код")->AsInteger = StrToInt(Edit1->Text);
DM1->ADOSale->FieldByName("Дата продажу")->AsDateTime = StrToDate(MaskEdit1->Text);
DM1->ADOSale->FieldByName("Код товару")->AsInteger = StrToInt(Edit2->Text);
DM1->ADOSale->FieldByName("Найменування товару")->AsString = Edit3->Text;
DM1->ADOSale->FieldByName("Кількість")->AsInteger = StrToInt(Edit4->Text);
DM1->ADOSale->FieldByName("Ціна")->AsInteger = StrToInt(Edit5->Text);
DM1->ADOSale->FieldByName("Сумма")->AsInteger = StrToInt(Edit6->Text);
DM1->ADOSale->Post();

//Пошук та редагування кількості товару.
// 0 - Створюємо звичайний продаж, 1 - реалізація Бронювання(списується товар при створені Броні).
if(Label10->Caption == "0")
{
    DM1->ADOQuery1->Close();
    DM1->ADOQuery1->SQL->Clear();
    DM1->ADOQuery1->SQL->Add("update [Nomenclatura] set [Кількість] = [Кількість] - quantity");
    DM1->ADOQuery1->SQL->Add("Where [Nomenclatura].[Код товару] = code2");
    DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(Edit2->Text);
    DM1->ADOQuery1->Parameters->ParamByName("quantity")->Value=StrToInt(Edit4->Text);
    DM1->ADOQuery1->ExecSQL();

    DM1->ADONomenclatura->Close();
    DM1->ADONomenclatura->Open();
}
```

Рисунок 3.14. Редагування кількості в карточці товару

```
if(Edit2->Text.IsEmpty())
{
    String message = "Поле коду продажі для повернення не заповнене.";
    ShowMessage(message);
    return;
}

DM1->ADOQuery1->Close();
DM1->ADOQuery1->SQL->Clear();
DM1->ADOQuery1->SQL->Add("select * From [Sale]");
DM1->ADOQuery1->SQL->Add("Where [Sale].[Код] = code");
DM1->ADOQuery1->Parameters->ParamByName("code")->Value=StrToInt(Edit2->Text);
DM1->ADOQuery1->Prepared=True;
DM1->ADOQuery1->Open();

int quantity = DM1->ADOQuery1->FieldByName("Кількість")->AsInteger;

if(!DM1->ADOQuery1->IsEmpty())
{
    DM1->ADOQuery1->Close();
    DM1->ADOQuery1->SQL->Clear();
    DM1->ADOQuery1->SQL->Add("update [Nomenclatura] set [Кількість] = [Кількість] + quantity");
    DM1->ADOQuery1->SQL->Add("Where [Nomenclatura].[Код товару] = code2");
    DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(Edit2->Text);
    DM1->ADOQuery1->Parameters->ParamByName("quantity")->Value=quantity;
    DM1->ADOQuery1->ExecSQL();

    DM1->ADOQuery1->SQL->Clear();
    DM1->ADOQuery1->SQL->Add("delete from [Sale]");
    DM1->ADOQuery1->SQL->Add("where [Sale].[Код] = CodeForDelete");
    DM1->ADOQuery1->Parameters->ParamByName("CodeForDelete")->Value = StrToInt(Edit2->Text);
    DM1->ADOQuery1->ExecSQL();

    DM1->ADONomenclatura->Close();
    DM1->ADONomenclatura->Open();
    DM1->ADOSale->Close();
    DM1->ADOSale->Open();

}else
{
    String message = "По данному коду нічого не знайдено.";
    ShowMessage(message);
    return;
}
```

Рисунок 3.15. Перевірка на наявність запису продажу та видалення запису із бази даних

В розділі «Замовлення» створюється запис замовлення до постачальника на вказаний товар та вказані кількість. Перед додаванням запису відбувається перевірка на унікальність, яка не довзоляє ствіорювати замовлення із однаковим

кодом. Приклад програмного коду створення замовлення та перевірки відображено на рисунку 3.16.

```
DM1->ADOQuery1->Close();
DM1->ADOQuery1->SQL->Clear();
DM1->ADOQuery1->SQL->Add("select [Код замовлення] From [Orders]");
DM1->ADOQuery1->SQL->Add("Where [Orders].[Код замовлення] = code2");
DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(Edit1->Text);
DM1->ADOQuery1->Prepared=True;
DM1->ADOQuery1->Open();
if(!DM1->ADOQuery1->IsEmpty())
{
    String message = "Знайдено запис із даним Кодом Замовника, будь ласка оберіть інший";
    ShowMessage(message);
    return;
}
DM1->ADOQuery1->Close();


DM1->ADOOrders->Insert();
DM1->ADOOrders->FieldByName("Код замовлення")->AsInteger=StrToInt(Edit1->Text);
DM1->ADOOrders->FieldByName ("Код посмачальника")->AsInteger=StrToInt(Edit2->Text);
DM1->ADOOrders->FieldByName("Дата замовлення")->AsDateTime=StrToDate(Date().FormatString("dd/mm/yyyy"));
DM1->ADOOrders->FieldByName("Кількість")->AsInteger=StrToInt(Edit5->Text);
DM1->ADOOrders->FieldByName("Ціна за одиницю")->AsInteger=StrToInt(Edit7->Text);
DM1->ADOOrders->FieldByName("№")->AsInteger=StrToInt(Edit6->Text);
DM1->ADOOrders->FieldByName("Код товару")->AsInteger=StrToInt(Edit3->Text);
DM1->ADOOrders->FieldByName("Номенклатура")->AsString=Edit4->Text;
DM1->ADOOrders->Post();
```

Рисунок 3.16. Код створення замовлення та перевірки унікальності коду замовлення

В розіділ «Надходження» для створення запису надходження необхідно спочтаку обрати замовлення, як документ-основу, а потім на його оснуві створити документ надходження, який збільшує кількість залишку товару. Вибір документа замовлення реалізовано за допомогою додаткової форми та компонентів DBGrid, які відображають інформацію реквізиту, який вкзааний програмним кодом. DBEdit є компонентом в Rad Studio, який використовується для відображення та редагування даних зі зв'язаних полів у наборі даних (dataset). Він є частиною бібліотеки компонентів Visual Component Library (VCL) та FireMonkey (FMX) і призначений для використання в програмах, які працюють з базами даних. DBEdit дозволяє зв'язати поле компонента з певним полем у наборі даних. Коли значення поля у наборі даних змінюється, відповідне значення в DBEdit також оновлюється, і навпаки, коли користувач вводить нове значення у DBEdit, відповідне поле у наборі даних оновлюється.

Основні властивості DBEdit включають:

- **DataSource**: Вказує на джерело даних, до якого пов'язаний DBEdit. Вибравши відповідний набір даних (TDataSet) у DataSource, DBEdit

автоматично відображатиме та оновлюватиме значення пов'язаного поля.

- **DataField**: Вказує на поле в наборі даних, з яким пов'язаний DBEdit. Зазвичай вибирається поле, яке ви хочете відобразити та редагувати у DBEdit.
- **ReadOnly**: Визначає, чи можна редагувати значення у DBEdit. Якщо властивість встановлена в True, значення буде тільки для читання.
- **Alignment**: Визначає вирівнювання тексту у DBEdit (зліва, посередині, справа).

Крім DBEdit, в Rad Studio є й інші схожі компоненти для роботи з даними:

1. **DBMemo**: Використовується для відображення та редагування великого обсягу текстових даних.
2. **DBComboBox**: Дозволяє вибирати значення зі списку, який можна зв'язати з набором даних.
3. **DBListBox**: Дозволяє вибирати одне або кілька значень зі списку, який можна зв'язати з набором даних.
4. **DBImage**: Використовується для відображення та редагування графічних зображень, що зберігаються у базі даних

Приклади виконання програмного коду зображені на рисунках 3.17, 3.18 та 3.19.

```
DBEditOrderCode->DataSource=DM1->DSOrders;
DBEditOrderCode->DataField = "Код замовлення";

DBEditProducerCode->DataSource=DM1->DSOrders;
DBEditProducerCode->DataField = "Код постачальника";

DBEditNomenCode->DataSource=DM1->DSOrders;
DBEditNomenCode->DataField = "Код товару";

DBEditNomenName->DataSource=DM1->DSOrders;
DBEditNomenName->DataField = "Номенкламура";

DBEditCount->DataSource=DM1->DSOrders;
DBEditCount->DataField = "Кількість";

DBEditPrice->DataSource=DM1->DSOrders;
DBEditPrice->DataField = "Ціна за одиницю";
```

Рисунок 3.17. Вказання реквізита для компонента DBEit

```
if(Form12->ShowModal() == mrOk)
{

    int OrderCode = Form12->DBEditOrderCode->Text.ToInt();
    int NomCode = Form12->DBEditNomenCode->Text.ToInt();
    String NomName = Form12->DBEditNomenName->Text;
    int ProducerCode = Form12->DBEditProducerCode->Text.ToInt();
    int Count = Form12->DBEditCount->Text.ToInt();
    int Price = Form12->DBEditPrice->Text.ToInt();

    //Знайдемо найменування постачальника

    DM1->ADOQuery1->Close();
    DM1->ADOQuery1->SQL->Clear();
    DM1->ADOQuery1->SQL->Add("select [Ім'я] FROM [Klienti]");
    DM1->ADOQuery1->SQL->Add("where [Klienti].[Код постачальника] = code2");
    DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(ProducerCode);
    DM1->ADOQuery1->Prepared=True;
    DM1->ADOQuery1->Open();

    String ProducerName = DM1->ADOQuery1->FieldByName("Ім'я")->AsString;

    //Відкрити форму створення надходження і передати параметри.

    DM1->ADOQuery1->Close();
    DM1->ADOQuery1->SQL->Clear();
    DM1->ADOQuery1->SQL->Add("select top 1 [Код надходження] From [Income]");
    DM1->ADOQuery1->SQL->Add("order by [Код надходження] desc");
    DM1->ADOQuery1->Prepared=True;
    DM1->ADOQuery1->Open();

    Form13->Edit1->Text = (DM1->ADOQuery1->FieldByName("Код надходження")->AsInteger) + 1;
    DM1->ADOQuery1->Close();

    Form13->Edit2->Text = OrderCode;
    Form13->Edit3->Text = NomCode;
    Form13->Edit4->Text = NomName;
    Form13->Edit5->Text = ProducerCode;
    Form13->Edit6->Text = ProducerName;
    Form13->Quantity = Count;
    Form13->Edit7->Text = Count;
    Form13->Edit8->Text = Price;
    Form13->ShowModal();
}
```

Рисунок 3.18. Вибір документу-основи

```
//-------------------------------------
//Додаємо запис до таблиці Надходження
DM1->ADOIncome->Insert();
DM1->ADOIncome->FieldByName("Код надходження")->AsInteger = StrToInt(Edit1->Text);
DM1->ADOIncome->FieldByName("Дата документа")->AsDateTime = StrToDate(MaskEdit1->Text);
DM1->ADOIncome->FieldByName("Код замовлення")->AsInteger = StrToInt(Edit2->Text);
DM1->ADOIncome->FieldByName("Код товару")->AsInteger = StrToInt(Edit3->Text);
DM1->ADOIncome->FieldByName("Найменування товару")->AsString = Edit4->Text;
DM1->ADOIncome->FieldByName("Код постачальника")->AsInteger = StrToInt(Edit5->Text);
DM1->ADOIncome->FieldByName("Найменування постачальника")->AsString = Edit6->Text;
DM1->ADOIncome->FieldByName("Кількість")->AsInteger = StrToInt(Edit7->Text);
DM1->ADOIncome->FieldByName("Ціна за одиницю")->AsInteger = StrToInt(Edit8->Text);
DM1->ADOIncome->FieldByName("Сума")->AsInteger = StrToInt(Edit9->Text);

DM1->ADOIncome->Post();
//-------------------------------------

//-------------------------------------
//Редагуємо кількість товару по коду найменування.
//-------------------------------------
DM1->ADOQuery1->Close();
DM1->ADOQuery1->SQL->Clear();
DM1->ADOQuery1->SQL->Add("update [Nomenclatura] set [Кількість] = [Кількість] + quantity");
DM1->ADOQuery1->SQL->Add("Where [Nomenclatura].[Код товару] = code2");
DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(Edit3->Text);
DM1->ADOQuery1->Parameters->ParamByName("quantity")->Value=StrToInt(Edit7->Text);
DM1->ADOQuery1->ExecSQL();

DM1->ADONomenclatura->Close();
DM1->ADONomenclatura->Open();
```

Рисунок 3.19. Код створення запису до бази даних та редагування кількості товару

Записані дані в базу даних мають наступний вигляд (рисунок 3.20).

Код	Дата продажу	Код товару	Найменуваі	Кількість	Ціна	Сумма	Щелкните для добавления
1	15.05.2023	4	Джинси	12	333	3996	
2	20.05.2023	6	Поло	5	750	3750	
3	20.05.2023	10	Шкарпетки	10	1000	10000	
0		0		0	0	0	

Рисунок 3.20 - Вигляд даних записаних у базу даних

Видалення та редагування даних в базі даних безпосередньо, а не через створену програму, може мати важливі наслідки, особливо якщо виконується недбало або неконтрольовано.

Неправильне видалення або редагування даних може призвести до втрати важливої інформації. Якщо випадково видалити дані без належного резервного копіювання, можливо втратити їх назавжди.

Видалення або редагування даних без належного контролю може призвести до порушення прав доступу. Якщо користувач не має необхідних дозволів для видалення або редагування даних, спроби виконати ці операції можуть бути заборонені системою або СУБД.

Якщо видалити або редагувати дані без контролю, може бути важко прослідкувати або відновити зміни. Аудит даних і належний контроль дозволяють встановити, хто і коли здійснював зміни, що може бути важливим для відновлення даних або виявлення проблем.

Якщо необхідно змінити реквізити таблиці або додати новий реквіт, то при оновленні таблиці в базі даних необхідно також вказати ці зміни і в програмі. При видаленні або редагування полів таблиці, без відповідного змінення коду в програмі, можливі помилки пошкодження даних, пошкодження структури СУБД та можливі летальні наслідки, коли інформацію у вже довго існуючій базі данних, не відновити.

Для застереження себе від таких наслідків більшість підприємств роблять резервні копії своїх баз даних.

Для створення резервної копії даних в базі Access можливо використати декілька методів:

- За допомогою функції «Резервне копіювання бази даних»: Access надає вбудовану функцію "Резервне копіювання бази даних", яка дозволяє зробити повну резервну копію бази даних з вмістом таблиць, запитів, форм, звітів і модулів. Щоб скористатися цією функцією, слід відкрити базу даних, перейти до вкладки "Файл" в головному меню, обрати "Зберегти та розмістити" і далі "Резервне копіювання бази даних".
- За допомогою копіювання файлу бази даних: для створення резервної копії можливо просто скопіювати файл бази даних (зазвичай має розширення .mdb або .accdb). Його можна скопіювати та зберігати в іншому місці як резевну копію. Цй метод дозволяє точно відновити базу данних, так як всі дані та структура зберігаються в файлі.
- За допомогою стороннього ПЗ: окрім вбудованих функцій Access, існують різні інструменти сторонніх розробників, які надають розширені можливості для резервного копіювання бази даних Access. Ці інструменти можуть дозволяти автоматичне регулярне створення резервних копій, створення інкрементних або диференційних копій.

ВИСНОВКИ

В результаті проведення дослідження розроблено Систему товарообігу магазину одягу – «MerchFlow», яка вирішує задачі обліку товарів в підприємствах.

В першому розділі був проведений аналіз предметної області, поставлена задача розробки, була досліджена предметна область, загальні принципи застосунків для ведення обліку та принцип їх роботи. Розглянуті існуючі програми для порівняння та встановлення функціональності власної розроблювальної системи.

У другому розділі описано особливості створюваної програми, опис вимог, проектування її функціональності та задач, які повина виконувати. Також важливими пунктами є вибір мови програмування та середовища розробки, бази даних та огляд інтерфейсу додатку. Вибір цих всіх складових впливає на роботу проекту і кінцевий результат.

У третьому розділі описані етапи розробки MerchFlow, починаючи від її створення і закінчуючи підключенням і взаємодією з нею бази даних.

Завдяки правильному відбору всіх аспектів, необхідних для отримання високоякісних та сучасних програмних продуктів, було розроблено систему, яка повністю відповідає описаним вимогам та особливостям, виконує всі поставлені завдання та автоматизує роботу.

Вона є унікальним помічником під час веденя бізнесу і без сумніву полегшує обробку інформації, швидкість її обробки, а також надає функції для зберігання введеної інформації для її подальшої обробки.

Даний програмний продукт має перспективу для розвитку і розширення свого функціоналу, що дозволить оптимізувати і полегшити вирішення ще великої кількості задач обробки інформації у веденні обліку руху товарів в галузі електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кетков Ю., Кетков О. Практика програмування visual basic, C++ builder, delph. 2016.
2. ACode. Уроки програмування на C++. URL: https://acode.com.ua/uroki-po-cpp/
3. Besedin A. Secrets of access database development and programming. 2017.
4. C++ програмування. URL: http://cpp.dp.ua/
5. C++ builder pro named term license (embarcadero). США : Embarcadero.
6. Conrad J., Viescas J. Microsoft access inside out. 2010.
7. Ravesli. Уроки C++. URL: https://ravesli.com/
8. Організаціяя товарообігу. URL: https://library.if.ua/book/43/2972.html
9. Механізм управління товарооборотом. Ефективна економіка. 2019. 5. URL: http://www.economy.nayka.com.ua/?op=1&z=555
10. Про що розповідає роздрібний товарообіг. URL: http://edclub.com.ua/analityka/pro-shcho-rozpovidaye-rozdribnyy-tovaroobig
11. Head First C: A Brain-Friendly Guide. Beijing: O'Reilly, 2012. 591 с.
12. C++ Tutorial. URL: https://www.w3schools.com/cpp/default.asp
13. C++ Programing Language. URL: https://www.geeksforgeeks.org/c-plus-plus/
14. SQL Tutorial. URL: https://www.w3schools.com/sql/default.asp
15. What is Structured Query Language (SQL). URL: https://www.techtarget.com/searchdatamanagement/definition/SQL
16. RAD Studio wiki. URL: https://docwiki.embarcadero.com/RADStudio/Alexandria/en/Main_Page
17. C++ builder. URL: https://www.bestprog.net/en/2016/02/25/008-c-builder-an-example-of-creating-and-calling-a-new-form-from-main-form-of-application/
18. Green D., Guntheroth K., Ross Mitchell S. The C++ Workshop: Learn to write clean, maintainable code in C++ and advance your career in software engineering : підручник. Packt Publishing, 2020. 606 p.

19. Bancila M. Modern C++ Programming Cookbook: Master C++ core language and standard library features, with over 100 recipes, updated to C++20, 2nd Edition : підручник. 2nd ed. Packt Publishing, 2020. 1141 p.
20. Roth S. Clean C++20: sustainable software development patterns and best practices : підручник. 2nd ed. Apress, 2021. 657 p.
21. Shields W. SQL quickstart guide: the simplified beginner's guide to managing, analyzing, and manipulating data with SQL (quickstart guides™ - technology). 2019. 242 p.
22. Moestl Vasilik S. SQL Practice Problems: 57 beginning, intermediate, and advanced challenges for you to solve using a “learn-by-doing” approach. 2017. 125 p.
23. Beighley L. Head first SQL: your brain on SQL : підручник. O'Reilly Media, 2007. 608 p.
24. SQL Querry Examples. URL: https://www.datacamp.com/tutorial/sql-query-examples-and-tutorial

ДОДАТОК А. Текст програми

Файл Main.h

```
#pragma hdrstop
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "Sale.h"
#include "Income.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Form2->ShowModal();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N6Click(TObject *Sender)
{
Form5->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N8Click(TObject *Sender)
{
Form3->ShowModal();
}
```

```
//---------------------------------------------------------------------------


void __fastcall TForm1::N2Click(TObject *Sender)
{
Form2->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N7Click(TObject *Sender)
{
Form6->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N9Click(TObject *Sender)
{
Form4->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
Form5->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
Form4->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button4Click(TObject *Sender)
{
Form3->ShowModal();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Button5Click(TObject *Sender)
{
Form8->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button6Click(TObject *Sender)
{
Form7->ShowModal();
}
```

Файл Nom.h

```
#include <vcl.h>
#pragma hdrstop
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "Income.h"
#include "AddNewNomenclature.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
//---------------------------------------------------------------------------
__fastcall TForm5::TForm5(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm5::Button2Click(TObject *Sender)
{
        Form4->ShowModal();
}
void __fastcall TForm5::Button1Click(TObject *Sender)
```

```
{
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Nomenclatura]");
        DM1->ADOQuery1->SQL->Add("where      [Nomenclatura].[Найменування]
=\""+Edit4->Text+"\"");
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        DBGrid1->DataSource=DM1->DSQuery1;
}
void __fastcall TForm5::FormShow(TObject *Sender)
{
        DBGrid1->DataSource=DM1->DSNomenclatura;

        DBEdit1->DataSource=DM1->DSNomenclatura;
        DBEdit1->DataField = "Код товару";
        DBEdit2->DataSource=DM1->DSNomenclatura;
        DBEdit2->DataField = "Найменування";
        DBEdit3->DataSource=DM1->DSNomenclatura;
        DBEdit3->DataField = "Код постачальника";
        DBEdit4->DataSource=DM1->DSNomenclatura;
        DBEdit4->DataField = "Кількість";
        DBEdit5->DataSource=DM1->DSNomenclatura;
        DBEdit5->DataField = "Колір";
        DBEdit6->DataSource=DM1->DSNomenclatura;
        DBEdit6->DataField = "Код товару";
        DBEdit7->DataSource=DM1->DSNomenclatura;
        DBEdit7->DataField = "Найменування постачальника";
}
void __fastcall TForm5::dClick(TObject *Sender)
{
  DBGrid1->DataSource=DM1->DSNomenclatura;
}
void __fastcall TForm5::Button4Click(TObject *Sender)
{
        if(Edit1->Text.IsEmpty())
        {
```

```
            String message = "Код постачальника не введено, пошук не виконано";
            ShowMessage(message);
            return;
        }
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select * from [Nomenclatura]");
        DM1->ADOQuery1->SQL->Add("where [Nomenclatura].[Код постачальника] = code");
        DM1->ADOQuery1->Parameters->ParamByName("code")->Value = StrToInt(Edit1->Text);
        DM1->ADOQuery1->Open();

   DBGrid1->DataSource = DM1->DSQuery1;
}
void __fastcall TForm5::Button5Click(TObject *Sender)
{
   if (MessageBox(NULL, L"Ви дійсно хочете виконати цю дію?", L"Підтвердження", MB_YESNO) == IDYES)
        {
            DM1->ADOQuery1->SQL->Clear();
            DM1->ADOQuery1->SQL->Add("delete from [Nomenclatura]");
            DM1->ADOQuery1->SQL->Add("where [Nomenclatura].[Код товару] = "+DBEdit6->Text+";");
            DM1->ADOQuery1->ExecSQL();
            DM1->ADONomenclatura->Close();
            DM1->ADONomenclatura->Open();
        }
        else
        {
      return;
        }
}
void __fastcall TForm5::Button6Click(TObject *Sender)
{
        Edit1->Text = "";
        DBGrid1->DataSource=DM1->DSNomenclatura;
```

```
}
void __fastcall TForm5::Button3Click(TObject *Sender)
{
        Edit4->Text = "";
        DBGrid1->DataSource=DM1->DSNomenclatura;
}
void __fastcall TForm5::Button10Click(TObject *Sender)
{
   Form14->ShowModal();
}
```

Файл NomChoiseForm.h

```
#include <vcl.h>
#pragma hdrstop
#include "NomChoiseForm.h"
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "Income.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm10 *Form10;
//---------------------------------------------------------------------------
__fastcall TForm10::TForm10(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm10::FormShow(TObject *Sender)
{
        DBGrid1->DataSource=DM1->DSNomenclatura;
        DBEditCodeTovaru->DataSource=DM1->DSNomenclatura;
        DBEditCodeTovaru->DataField = "Код товару";
```

```
        DBEditName->DataSource=DM1->DSNomenclatura;
        DBEditName->DataField = "Найменування";
        DBEditCount->DataSource=DM1->DSNomenclatura;
        DBEditCount->DataField = "Кількість";
}
```

Файл AddNewNomenclature.h

```
#include <vcl.h>
#pragma hdrstop
#include "AddNewNomenclature.h"
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "Income.h"
#include "ProducersChoiseForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm14 *Form14;
//---------------------------------------------------------------------------
__fastcall TForm14::TForm14(TComponent* Owner)
        : TForm(Owner)
{
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm14::Button1Click(TObject *Sender)
{
   if(Form11->ShowModal() == mrOk)
        {
                int ProdCode = Form11->DBEditCode->Text.ToInt();
                String ProdName = Form11->DBEditName->Text;
                this->Edit4->Text = ProdCode;
                this->Edit5->Text = ProdName;
        }
}
//---------------------------------------------------------------------------
void __fastcall TForm14::FormShow(TObject *Sender)
{
   //Шукаємо та заповнюємо останній індекс + 1
        //--------------------------------------
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select top 1 [Код товару] From
[Nomenclatura]");
        DM1->ADOQuery1->SQL->Add("order by [Код товару] desc");
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        Edit1->Text = (DM1->ADOQuery1->FieldByName("Код товару")-
>AsInteger) + 1;
        DM1->ADOQuery1->Close();
}
void __fastcall TForm14::Button2Click(TObject *Sender)
```

```
{
        DM1->ADONomenclatura->Insert();
        DM1->ADONomenclatura->FieldByName("Код товару")->AsInteger =
StrToInt(Edit1->Text);
        DM1->ADONomenclatura->FieldByName("Найменування")->AsString =
Edit2->Text;
        DM1->ADONomenclatura->FieldByName("Колір")->AsString = Edit3-
>Text;
        DM1->ADONomenclatura->FieldByName("Код постачальника")-
>AsInteger = StrToInt(Edit4->Text);
        DM1->ADONomenclatura->FieldByName("Найменування
постачальника")->AsString = Edit5->Text;
        DM1->ADONomenclatura->Post();
}
```

Файл Producers.h

```
#include <vcl.h>
#pragma hdrstop
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
```

```
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm2::Button2Click(TObject *Sender)
{
        Form3->ShowModal();
}
void __fastcall TForm2::FormShow(TObject *Sender)
{
        DBGrid1->DataSource=DM1->DSKlienti;
}
void __fastcall TForm2::Button1Click(TObject *Sender)
{
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select [Код постачальника] From [Klienti]");
        DM1->ADOQuery1->SQL->Add("Where [Klienti].[Код постачальника] = code1");
        DM1->ADOQuery1->Parameters->ParamByName("code1")->Value=Code1;
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(!DM1->ADOQuery1->IsEmpty())
        {
                String message = "Знайдено запис із даним Кодом постачальника, будь ласка введіть інший";
                ShowMessage(message);
```

```
            return;
        }
        DM1->ADOQuery1->Close();
    DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select [Номер телефону] From [Klienti]");
        DM1->ADOQuery1->SQL->Add("Where [Klienti].[Номер телефону] =
number");
        DM1->ADOQuery1->Parameters->ParamByName("number")->Value =
MaskEdit1->Text;
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(!DM1->ADOQuery1->IsEmpty())
        {
            String message = "Знайдено запис із даним номером телефону";
            ShowMessage(message);
            return;
        }
        DM1->ADOQuery1->Close();
        DM1->ADOKlienti->Insert();
        DM1->ADOKlienti->FieldByName("Код постачальника")-
>AsInteger=StrToInt(Edit1->Text);
        DM1->ADOKlienti->FieldByName ("Прізвище")->AsString=Edit4->Text;
        DM1->ADOKlienti->FieldByName("Ім'я")->AsString=Edit2->Text;
        DM1->ADOKlienti->FieldByName("Номер телефону")-
>AsString=MaskEdit1->Text;
        DM1->ADOKlienti->FieldByName("По батькові")->AsString=Edit5->Text;
        DM1->ADOKlienti->FieldByName("Адреса складу")->AsString=Edit6-
>Text;
```

```
        DM1->ADOKlienti->Post();
        DBGrid1->DataSource=DM1->DSKlienti;
}
void __fastcall TForm2::Button3Click(TObject *Sender)
{
        DM1->ADOQuery1->Close();
        DM1-> ADOQuery1->SQL->Clear();
        DM1 ->ADOQuery1->SQL->Add( "select * FROM [Klienti]");
        DM1->ADOQuery1->SQL->Add("where [Klienti].[Прізвище] =\""+Edit7-
>Text+"\"");
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        DBGrid1->DataSource=DM1->DSQuery1;
}
void __fastcall TForm2::Button4Click(TObject *Sender)
{
        Edit7->Text = "";
        DBGrid1->DataSource=DM1->DSKlienti;
}
void __fastcall TForm2::Image1Click(TObject *Sender)
{
        DBGrid1->DataSource=DM1->DSKlienti;
}
void __fastcall TForm2::Button5Click(TObject *Sender)
{
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("delete from [Klienti]");
```

```
        DM1->ADOQuery1->SQL->Add("where [Klienti].[Код постачальника] =
"+Edit3->Text+";");
        DM1->ADOQuery1->ExecSQL();
        DM1->ADOKlienti->Close();
        DM1->ADOKlienti->Open();
        DBGrid1->DataSource=DM1->DSKlienti;
}
```

Файл ProducersChioseForm.h

```
#include <vcl.h>
#pragma hdrstop
#include "ProducersChoiseForm.h"
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "Income.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm11 *Form11;
//---------------------------------------------------------------------------
__fastcall TForm11::TForm11(TComponent* Owner)
```

```
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm11::FormShow(TObject *Sender)
{
        DBGrid1->DataSource=DM1->DSKlienti;
        DBEditCode->DataSource=DM1->DSKlienti;
        DBEditCode->DataField = "Код постачальника";
        DBEditName->DataSource=DM1->DSKlienti;
        DBEditName->DataField = "Ім'я";
}
void __fastcall TForm11::Button2Click(TObject *Sender)
{
        if(Edit8->Text.IsEmpty())
        {
                String message = "Введіть код постачальника.";
                ShowMessage(message);
                return;
        }
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Klienti]");
        DM1->ADOQuery1->SQL->Add("where [Klienti].[Код постачальника]
=code1");
        DM1->ADOQuery1->Parameters->ParamByName("code1")-
>Value=StrToInt(Edit8->Text);
        DM1->ADOQuery1->Prepared=True;
```

```
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
                DBEditCode->Text = DM1->ADOQuery1->FieldByName("Код постачальника")->AsInteger;
        }
}
void __fastcall TForm11::Button7Click(TObject *Sender)
{
        DBGrid1->DataSource = DM1->DSKlienti;
   Edit8->Text = "";
}
```

Файл OrdersChoiseForm.h

```
#include <vcl.h>
#pragma hdrstop
#include "OrdersChoiseForm.h"
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
```

```
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "Income.h"
#include "NomChoiseForm.h"
#include "ProducersChoiseForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm12 *Form12;
//---------------------------------------------------------------------------
__fastcall TForm12::TForm12(TComponent* Owner)
        : TForm(Owner)
{
   DBGrid1->DataSource=DM1->DSOrders;
}
//---------------------------------------------------------------------------
void __fastcall TForm12::FormShow(TObject *Sender)
{
        DBEditOrderCode->DataSource=DM1->DSOrders;
        DBEditOrderCode->DataField = "Код замовлення";
        DBEditProducerCode->DataSource=DM1->DSOrders;
        DBEditProducerCode->DataField = "Код постачальника";
        DBEditNomenCode->DataSource=DM1->DSOrders;
        DBEditNomenCode->DataField = "Код товару";
        DBEditNomenName->DataSource=DM1->DSOrders;
        DBEditNomenName->DataField = "Номенклатура";
```

```
        DBEditCount->DataSource=DM1->DSOrders;
        DBEditCount->DataField = "Кількість";
        DBEditPrice->DataSource=DM1->DSOrders;
        DBEditPrice->DataField = "Ціна за одиницю";
}
```

Файл Orders.h

```
 #include <vcl.h>
#pragma hdrstop
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "Income.h"
#include "NomChoiseForm.h"
#include "ProducersChoiseForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
        DBGrid1->DataSource=DM1->DSOrders;
```

```
        DBNavigator1->DataSource=DM1->DSOrders;
}
void __fastcall TForm4::Button1Click(TObject *Sender)
{
        Edit8->Text = "";
        DBGrid1->DataSource=DM1->DSOrders;
        DBNavigator1->DataSource=DM1->DSOrders;
 }
void __fastcall TForm4::Button3Click(TObject *Sender)
{
        if(MaskEdit1->Text.IsEmpty())
        {
       String message = "Введіть дату для пошуку.";
                ShowMessage(message);
                return;
        }
   DM1->ADOQuery1->Close();
        DM1-> ADOQuery1->SQL->Clear();
        DM1 ->ADOQuery1->SQL->Add( "select * FROM [Orders]");
        DM1->ADOQuery1->SQL->Add("where [Orders].[Дата замовлення]
=date");
        DM1->ADOQuery1->Parameters->ParamByName("date")-
>Value=StrToDate(MaskEdit1->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
```

```
            ShowMessage(message);
            return;
        }
        else
        {
            DBGrid1->DataSource = DM1->DSQuery1;
            DBNavigator1->DataSource = DM1->DSQuery1;
        }

}
void __fastcall TForm4::Button4Click(TObject *Sender)
{
        Form6->ShowModal();
}
void __fastcall TForm4::Button2Click(TObject *Sender)
{
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select [Код замовлення] From [Orders]");
        DM1->ADOQuery1->SQL->Add("Where [Orders].[Код замовлення] =
code2");
        DM1->ADOQuery1->Parameters->ParamByName("code2")-
>Value=StrToInt(Edit1->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(!DM1->ADOQuery1->IsEmpty())
        {
```

```
            String message = "Знайдено запис із даним Кодом Замовника, будь ласка введіть інший";
            ShowMessage(message);
            return;
        }
        DM1->ADOQuery1->Close();
        DM1->ADOOrders->Insert();
        DM1->ADOOrders->FieldByName("Код замовлення")->AsInteger=StrToInt(Edit1->Text);
        DM1->ADOOrders->FieldByName ("Код постачальника")->AsInteger=StrToInt(Edit2->Text);
        DM1->ADOOrders->FieldByName("Дата замовлення")->AsDateTime=StrToDate(Date().FormatString("dd/mm/yyyy"));
        DM1->ADOOrders->FieldByName("Кількість")->AsInteger=StrToInt(Edit5->Text);
        DM1->ADOOrders->FieldByName("Ціна за одиницю")->AsInteger=StrToInt(Edit7->Text);
        DM1->ADOOrders->FieldByName("№")->AsInteger=StrToInt(Edit6->Text);
        DM1->ADOOrders->FieldByName("Код товару")->AsInteger=StrToInt(Edit3->Text);
        DM1->ADOOrders->FieldByName("Номенклатура")->AsString=Edit4->Text;
        DM1->ADOOrders->Post();
}
void __fastcall TForm4::Button6Click(TObject *Sender)
{
        Form5->ShowModal();
}
void __fastcall TForm4::Button5Click(TObject *Sender)
```

```
{
        if(Edit8->Text.IsEmpty())
        {
                String message = "Введіть код постачальника.";
                ShowMessage(message);
                return;
        }
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Orders]");
        DM1->ADOQuery1->SQL->Add("where [Orders].[Код постачальника] =code1");
        DM1->ADOQuery1->Parameters->ParamByName("code1")->Value=StrToInt(Edit8->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
                DBNavigator1->DataSource = DM1->DSQuery1;
        }
}
```

```
void __fastcall TForm4::Image1Click(TObject *Sender)
{
        DBGrid1->DataSource=DM1->DSOrders;
        DBNavigator1->DataSource=DM1->DSOrders;
}
void __fastcall TForm4::Button10Click(TObject *Sender)
{
        if(Edit9->Text.IsEmpty())
        {
                String message = "Введіть код замовлення.";
                ShowMessage(message);
                return;
        }
        DM1->ADOQuery1->Close();
        DM1-> ADOQuery1->SQL->Clear();
        DM1 ->ADOQuery1->SQL->Add( "select * FROM [Orders]");
        DM1->ADOQuery1->SQL->Add("where [Orders].[Код замовлення]
=code2");
        DM1->ADOQuery1->Parameters->ParamByName("code2")-
>Value=StrToInt(Edit9->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
                ShowMessage(message);
                return;
        }
```

```
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
                DBNavigator1->DataSource = DM1->DSQuery1;
        }
}
void __fastcall TForm4::Button11Click(TObject *Sender)
{
        Edit9->Text = "";
        DBGrid1->DataSource=DM1->DSOrders;
        DBNavigator1->DataSource=DM1->DSOrders;
}
void __fastcall TForm4::Button9Click(TObject *Sender)
{
   MaskEdit1->Text = "";
        DBGrid1->DataSource=DM1->DSOrders;
        DBNavigator1->DataSource=DM1->DSOrders;
}
void __fastcall TForm4::Button8Click(TObject *Sender)
{
        if(Form10->ShowModal() == mrOk)
        {
                int CodeTovaru = Form10->DBEditCodeTovaru->Text.ToInt();
                String name = Form10->DBEditName->Text;
                this->Edit3->Text = CodeTovaru;
                this->Edit4->Text = name;
        }
```

```
}
void __fastcall TForm4::Button12Click(TObject *Sender)
{
        if(Form11->ShowModal() == mrOk)
        {
                int CodeProducer = Form11->DBEditCode->Text.ToInt();
                this->Edit2->Text = CodeProducer;
        }
}
void __fastcall TForm4::Button13Click(TObject *Sender)
{
   if (DBGrid1->SelectedRows->Count > 0)
  {
        // Отримуємо значення необхідних полів поточного рядка
    TBookmark bookmark = DBGrid1->SelectedRows->Items[0];
    DBGrid1->DataSource->DataSet->GotoBookmark(bookmark);
        int OrderId = DBGrid1->DataSource->DataSet->FieldByName("Код замовлення")->AsInteger;
        int ProdId = DBGrid1->DataSource->DataSet->FieldByName("Код постачальника")->AsInteger;
        int GoodId = DBGrid1->DataSource->DataSet->FieldByName("Код постачальника")->AsInteger;
    // Відображаємо модальну форму
        Form7->ShowModal();
  }
  else
  {
    ShowMessage("Будь ласка, виберіть рядок в DBGrid1.");
```

```
  }
}
```

Файл IncomeAddDoc .h

```
#include <vcl.h>
#pragma hdrstop
#include "IncomeAddDoc.h"
#include "Income.h"
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "OrdersChoiseForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm13 *Form13;
//---------------------------------------------------------------------------
__fastcall TForm13::TForm13(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm13::FormShow(TObject *Sender)
{
```

```
        //Заповнюємо сьогодняшню дату
        //---------------------------------------
        MaskEdit1->Text = Date().FormatString("dd/mm/yyyy");
}
void __fastcall TForm13::FormClose(TObject *Sender, TCloseAction &Action)
{
        Edit1->Text = "";
        Edit2->Text = "";
        Edit3->Text = "";
        Edit4->Text = "";
        Edit5->Text = "";
        Edit6->Text = "";
        //Edit7->Text = "";
        Edit8->Text = "";
        Edit9->Text = "";
        MaskEdit1->Text = "";
}
void __fastcall TForm13::Button1Click(TObject *Sender)
{
   //Додаємо запис до таблиці Надходження
        DM1->ADOIncome->Insert();
        DM1->ADOIncome->FieldByName("Код надходження")->AsInteger =
StrToInt(Edit1->Text);
        DM1->ADOIncome->FieldByName("Дата документа")->AsDateTime =
StrToDate(MaskEdit1->Text);
   DM1->ADOIncome->FieldByName("Код замовлення")->AsInteger =
StrToInt(Edit2->Text);
```

```
DM1->ADOIncome->FieldByName("Код товару")->AsInteger = StrToInt(Edit3->Text);

DM1->ADOIncome->FieldByName("Найменування товару")->AsString = Edit4->Text;

DM1->ADOIncome->FieldByName("Код постачальника")->AsInteger = StrToInt(Edit5->Text);

DM1->ADOIncome->FieldByName("Найменування постачальника")->AsString = Edit6->Text;

DM1->ADOIncome->FieldByName("Кількість")->AsInteger = StrToInt(Edit7->Text);

DM1->ADOIncome->FieldByName("Ціна за одиницю")->AsInteger = StrToInt(Edit8->Text);

DM1->ADOIncome->FieldByName("Сума")->AsInteger = StrToInt(Edit9->Text);


DM1->ADOIncome->Post();

//-------------------------------------

//Редагуємо кількість товару по коду найменування.

//-------------------------------------

DM1->ADOQuery1->Close();

DM1->ADOQuery1->SQL->Clear();

DM1->ADOQuery1->SQL->Add("update [Nomenclatura] set [Кількість] = [Кількість] + quantity");

DM1->ADOQuery1->SQL->Add("Where [Nomenclatura].[Код товару] = code2");

DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(Edit3->Text);

DM1->ADOQuery1->Parameters->ParamByName("quantity")->Value=StrToInt(Edit7->Text);

DM1->ADOQuery1->ExecSQL();
```

```
        DM1->ADONomenclatura->Close();
        DM1->ADONomenclatura->Open();
}
void __fastcall TForm13::Edit7Change(TObject *Sender)
{
        if(!Edit7->Text.IsEmpty() && Edit7->Text.ToInt() > Quantity)
        {
                String message = "Кількість у надходженнні не може бути більше ніж у замовленні.";
                ShowMessage(message);
                return;
        }else if(Edit7->Text.ToInt() < 0)
        {
      String message = "Кількість у надходженнні не може бути менше 0.";
                ShowMessage(message);
                return;
        }
}
void __fastcall TForm13::Edit8Change(TObject *Sender)
{
   if(!Edit7->Text.IsEmpty() && !Edit8->Text.IsEmpty())
        {
                Edit9->Text = Edit7->Text.ToInt() * Edit8->Text.ToInt();
        }
}
```

Файл Income.h

```
#include <vcl.h>
```

```
#pragma hdrstop
#include "Income.h"
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "OrdersChoiseForm.h"
#include "IncomeAddDoc.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm7 *Form7;
//---------------------------------------------------------------------------
__fastcall TForm7::TForm7(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm7::Button7Click(TObject *Sender)
{
        if(MaskEdit1->Text.IsEmpty())
        {
                String message = "Введіть дату для пошуку.";
                ShowMessage(message);
                return;
```

```
        }
    DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Income]");
        DM1->ADOQuery1->SQL->Add("where [Income].[Дата документу] =date");
        DM1->ADOQuery1->Parameters->ParamByName("date")-
>Value=StrToDate(MaskEdit1->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
        }
}
void __fastcall TForm7::Button3Click(TObject *Sender)
{
        if(Edit2->Text.IsEmpty())
        {
                String message = "Введіть код для пошуку.";
                ShowMessage(message);
                return;
```

```
        }
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Income]");
        DM1->ADOQuery1->SQL->Add("where [Income].[Код надходження] =
code1");
        DM1->ADOQuery1->Parameters->ParamByName("code1")-
>Value=StrToInt(Edit2->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
        }
}
void __fastcall TForm7::Button4Click(TObject *Sender)
{
        if(Edit3->Text.IsEmpty())
        {
                String message = "Введіть код для пошуку.";
                ShowMessage(message);
                return;
```

```
        }
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Income]");
        DM1->ADOQuery1->SQL->Add("where [Income].[Код замовлення] =
code1");
        DM1->ADOQuery1->Parameters->ParamByName("code1")-
>Value=StrToInt(Edit3->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
        }
}
void __fastcall TForm7::Button5Click(TObject *Sender)
{
        if(Edit4->Text.IsEmpty())
        {
                String message = "Введіть код для пошуку.";
                ShowMessage(message);
                return;
```

```
        }
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Income]");
        DM1->ADOQuery1->SQL->Add("where [Income].[Код товару] = code1");
        DM1->ADOQuery1->Parameters->ParamByName("code1")-
>Value=StrToInt(Edit4->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
        }
}
void __fastcall TForm7::Button6Click(TObject *Sender)
{
        if(Edit5->Text.IsEmpty())
        {
                String message = "Введіть код для пошуку.";
                ShowMessage(message);
                return;
```

```
        }
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Income]");
        DM1->ADOQuery1->SQL->Add("where [Income].[Код постачальника] =
code1");
        DM1->ADOQuery1->Parameters->ParamByName("code1")-
>Value=StrToInt(Edit5->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
        }
}
void __fastcall TForm7::Button1Click(TObject *Sender)
{
        if(Form12->ShowModal() == mrOk)
        {
                int OrderCode = Form12->DBEditOrderCode->Text.ToInt();
                int NomCode = Form12->DBEditNomenCode->Text.ToInt();
                String NomName = Form12->DBEditNomenName->Text;
```

```
int ProducerCode = Form12->DBEditProducerCode->Text.ToInt();
int Count = Form12->DBEditCount->Text.ToInt();
int Price = Form12->DBEditPrice->Text.ToInt();
//Знайдемо найменування постачальника
DM1->ADOQuery1->Close();
DM1->ADOQuery1->SQL->Clear();
DM1->ADOQuery1->SQL->Add("select [Ім'я] FROM [Klienti]");
DM1->ADOQuery1->SQL->Add("where [Klienti].[Код постачальника] = code2");
DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(ProducerCode);
DM1->ADOQuery1->Prepared=True;
DM1->ADOQuery1->Open();
String ProducerName = DM1->ADOQuery1->FieldByName("Ім'я")->AsString;
//Відкрити форму створення надходження і передати параметри.
DM1->ADOQuery1->Close();
DM1->ADOQuery1->SQL->Clear();
DM1->ADOQuery1->SQL->Add("select top 1 [Код надходження] From [Income]");
DM1->ADOQuery1->SQL->Add("order by [Код надходження] desc");
DM1->ADOQuery1->Prepared=True;
DM1->ADOQuery1->Open();
Form13->Edit1->Text = (DM1->ADOQuery1->FieldByName("Код надходження")->AsInteger) + 1;
DM1->ADOQuery1->Close();
Form13->Edit2->Text = OrderCode;
Form13->Edit3->Text = NomCode;
```

```
                Form13->Edit4->Text = NomName;
                Form13->Edit5->Text = ProducerCode;
                Form13->Edit6->Text = ProducerName;
                Form13->Quantity = Count;
                Form13->Edit7->Text = Count;
                Form13->Edit8->Text = Price;
                Form13->ShowModal();
        }
}
void __fastcall TForm7::Button2Click(TObject *Sender)
{
   DBGrid1->DataSource = DM1->DSIncome;
}
```

Файл Sale.h

```
#include <vcl.h>
#pragma hdrstop
#include "Sale.h"
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "AddSales.h"
#include "BronChoiseForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
```

```
#pragma resource "*.dfm"
TForm8 *Form8;
//---------------------------------------------------------------------------
__fastcall TForm8::TForm8(TComponent* Owner)
        : TForm(Owner)
{
        DBGrid1->DataSource=DM1->DSSale;
}
void __fastcall TForm8::Button1Click(TObject *Sender)
{
        Form9->ShowModal();
}
void __fastcall TForm8::Button2Click(TObject *Sender)
{
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select * From [Sale]");
        DM1->ADOQuery1->SQL->Add("Where [Sale].[Дата продажу] = date");
        DM1->ADOQuery1->Parameters->ParamByName("date")-
>Value=StrToDate(MaskEdit1->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        DBGrid1->DataSource = DM1->DSQuery1;
}
void __fastcall TForm8::Button3Click(TObject *Sender)
{
        MaskEdit1->Text = "";
```

```
        DBGrid1->DataSource=DM1->DSSale;
}
void __fastcall TForm8::Button4Click(TObject *Sender)
{
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select * From [Sale]");
        DM1->ADOQuery1->SQL->Add("Where [Sale].[Код] = code");
        DM1->ADOQuery1->Parameters->ParamByName("code")-
>Value=StrToInt(Edit1->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
   if(!DM1->ADOQuery1->IsEmpty())
        {
      DBGrid1->DataSource = DM1->DSQuery1;
        }else
        {
                String message = "По данному коду нічого не знайдено.";
                ShowMessage(message);
                return;
        }
}
void __fastcall TForm8::Button5Click(TObject *Sender)
{
   Edit1->Text = "";
        DBGrid1->DataSource=DM1->DSSale;
}
```

```
void __fastcall TForm8::Button6Click(TObject *Sender)
{
        if(Edit2->Text.IsEmpty())
        {
      String message = "Поле коду продажі для повернення не заповнене.";
                ShowMessage(message);
                return;
        }
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select * From [Sale]");
        DM1->ADOQuery1->SQL->Add("Where [Sale].[Код] = code");
        DM1->ADOQuery1->Parameters->ParamByName("code")-
>Value=StrToInt(Edit2->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        int quantity = DM1->ADOQuery1->FieldByName("Кількість")->AsInteger;
        if(!DM1->ADOQuery1->IsEmpty())
        {
                DM1->ADOQuery1->Close();
                DM1->ADOQuery1->SQL->Clear();
                DM1->ADOQuery1->SQL->Add("update [Nomenclatura] set
[Кількість] = [Кількість] + quantity");
                DM1->ADOQuery1->SQL->Add("Where [Nomenclatura].[Код товару]
= code2");
                DM1->ADOQuery1->Parameters->ParamByName("code2")-
>Value=StrToInt(Edit2->Text);
```

```
            DM1->ADOQuery1->Parameters->ParamByName("quantity")->Value=quantity;
            DM1->ADOQuery1->ExecSQL();
            DM1->ADOQuery1->SQL->Clear();
            DM1->ADOQuery1->SQL->Add("delete from [Sale]");
            DM1->ADOQuery1->SQL->Add("where [Sale].[Код] = CodeForDelete");
            DM1->ADOQuery1->Parameters->ParamByName("CodeForDelete")->Value = StrToInt(Edit2->Text);
            DM1->ADOQuery1->ExecSQL();
            DM1->ADONomenclatura->Close();
            DM1->ADONomenclatura->Open();
            DM1->ADOSale->Close();
      DM1->ADOSale->Open();
        }else
        {
            String message = "По данному коду нічого не знайдено.";
            ShowMessage(message);
            return;
        }
}
void __fastcall TForm8::Button7Click(TObject *Sender)
{
   Form5->ShowModal();
}
void __fastcall TForm8::Button8Click(TObject *Sender)
{
        if(Form16->ShowModal() == mrOk)
```

```
        {
                int BronCode = Form16->DBEditCodeBron->Text.ToInt();
                int NomCode = Form16->DBEditNomCode->Text.ToInt();
                String NomName = Form16->DBEditNomName->Text;
                int Count = Form16->DBEditCount->Text.ToInt();
      int Price = Form16->DBEditPrice->Text.ToInt();
                DM1->ADOQuery1->Close();
                DM1->ADOQuery1->SQL->Clear();
                DM1->ADOQuery1->SQL->Add( "select * FROM [Nomenclatura]");
                DM1->ADOQuery1->SQL->Add("where [Nomenclatura].[Код товару]
=code");
                DM1->ADOQuery1->Parameters->ParamByName("code")->Value =
NomCode;
                DM1->ADOQuery1->Prepared = True;
                DM1->ADOQuery1->Open();
                if(!DM1->ADOQuery1->IsEmpty())
                {
                        CountOfNom = DM1->ADOQuery1-
>FieldByName("Кількість")->AsInteger;
                }
                Form9->Edit2->Text = NomCode;
                Form9->Edit3->Text = NomName;
                Form9->Label9->Caption = CountOfNom;
      Form9->Label10->Caption = "1";
                Form9->Edit4->Text = Count;
                Form9->Edit5->Text = Price;
                if(Form9->ShowModal() == mrOk)
                {
```

```
                int SaleCode = Form9->Edit1->Text.ToInt();
                DM1->ADOQuery1->Close();
                DM1->ADOQuery1->SQL->Clear();
                DM1->ADOQuery1->SQL->Add("update [Bron] set [Статус бронювання] = status, [Код продажу] = saleCode");
                DM1->ADOQuery1->SQL->Add("Where [Bron].[Код бронювання] = code1");
                DM1->ADOQuery1->Parameters->ParamByName("code1")->Value= BronCode;
                DM1->ADOQuery1->Parameters->ParamByName("saleCode")->Value= SaleCode;
                DM1->ADOQuery1->Parameters->ParamByName("status")->Value= "Реалізовано";
                DM1->ADOQuery1->ExecSQL();
                DM1->ADOBron->Close();
                DM1->ADOBron->Open();
            }
        }
}
```

Файл AddSales.h

```
#include <vcl.h>
#pragma hdrstop
#include "AddSales.h"
#include "Sale.h"
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
```

```
#include "Main.h"
#include "DM.h"
#include "NomChoiseForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm9 *Form9;
//---------------------------------------------------------------------------
__fastcall TForm9::TForm9(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm9::Edit4Change(TObject *Sender)
{
        if(!Edit4->Text.IsEmpty() && !Edit5->Text.IsEmpty())
        {
                Edit6->Text = IntToStr(StrToInt(Edit4->Text) * StrToInt(Edit5->Text));
        }
}
void __fastcall TForm9::Button1Click(TObject *Sender)
{
        //Перевірки заповнення та коректного введення
        //-----------------------------------------------------------------------------------
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select [Код] From [Sale]");
        DM1->ADOQuery1->SQL->Add("Where [Sale].[Код] = code1");
```

```
DM1->ADOQuery1->Parameters->ParamByName("code1")->Value=StrToInt(Edit1->Text);
DM1->ADOQuery1->Prepared=True;
DM1->ADOQuery1->Open();
if(!DM1->ADOQuery1->IsEmpty())
{
    String message = "Знайдено запис із даним кодом продажі, будь ласка введіть наступний";
    ShowMessage(message);
    return;
}
DM1->ADOQuery1->Close();
if(Label10->Caption == "0")
{
    if(Edit4->Text.IsEmpty() || StrToInt(Edit4->Text) > StrToInt(Label9->Caption))
    {
        String message = "Ви не ввели кількість, або ввели кількість більшу за доступну.";
        ShowMessage(message);
        return;
    }
}
if(Edit1->Text.IsEmpty() || Edit3->Text.IsEmpty() || Edit4->Text.IsEmpty() || Edit5->Text.IsEmpty()
    || Edit6->Text.IsEmpty() || MaskEdit1->Text.IsEmpty() || Edit2->Text.IsEmpty())
{
```

```
String message = "Не всі параметри заповнено, прохання заповнити всі параметри.";
        ShowMessage(message);
        return;
    }
    //Проведення продажі. Додавання запису в таблицю продажів
    DM1->ADOSale->Insert();
    DM1->ADOSale->FieldByName("Код")->AsInteger = StrToInt(Edit1->Text);
    DM1->ADOSale->FieldByName("Дата продажу")->AsDateTime = StrToDate(MaskEdit1->Text);
    DM1->ADOSale->FieldByName("Код товару")->AsInteger = StrToInt(Edit2->Text);
    DM1->ADOSale->FieldByName("Найменування товару")->AsString = Edit3->Text;
    DM1->ADOSale->FieldByName("Кількість")->AsInteger = StrToInt(Edit4->Text);
    DM1->ADOSale->FieldByName("Ціна")->AsInteger = StrToInt(Edit5->Text);
    DM1->ADOSale->FieldByName("Сумма")->AsInteger = StrToInt(Edit6->Text);
    DM1->ADOSale->Post();
  //Пошук та редагування кількості товару.
  // 0 - Створюємо звичайний продаж, 1 - реалізація бронювання(списується товар при створені броні).
    if(Label10->Caption == "0")
    {
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("update [Nomenclatura] set [Кількість] = [Кількість] - quantity");
```

```
            DM1->ADOQuery1->SQL->Add("Where [Nomenclatura].[Код товару]
= code2");
            DM1->ADOQuery1->Parameters->ParamByName("code2")-
>Value=StrToInt(Edit2->Text);
            DM1->ADOQuery1->Parameters->ParamByName("quantity")-
>Value=StrToInt(Edit4->Text);
            DM1->ADOQuery1->ExecSQL();
            DM1->ADONomenclatura->Close();
            DM1->ADONomenclatura->Open();
        }
}
void __fastcall TForm9::FormShow(TObject *Sender)
{
        //Очищаємо поля вводу, якщо форма перевідкривається
   if(Label10->Caption == "0")
        {
            Edit1->Text = "";
            Edit2->Text = "";
            Edit3->Text = "";
            Edit4->Text = "";
            Edit5->Text = "";
            Edit6->Text = "";
            MaskEdit1->Text = "";
            Label9->Caption = "";
        }
        //Шукаємо та заповнюємо останній індекс + 1
        //--------------------------------------
        DM1->ADOQuery1->Close();
```

```
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select top 1 [Код] From [Sale]");
        DM1->ADOQuery1->SQL->Add("order by [Код] desc");
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
   Edit1->Text = (DM1->ADOQuery1->FieldByName("Код")->AsInteger) + 1;
   DM1->ADOQuery1->Close();
        //---------------------------------------
        //Заповнюємо сьогодняшню дату
        //---------------------------------------
        MaskEdit1->Text = Date().FormatString("dd/mm/yyyy");
}
void __fastcall TForm9::Button3Click(TObject *Sender)
{
        if(Form10->ShowModal() == mrOk)
                {
                        int CodeTovaru = Form10->DBEditCodeTovaru->Text.ToInt();
                        String Name = Form10->DBEditName->Text;
                        this->Edit2->Text = CodeTovaru;
                        this->Edit3->Text = Name;
                }
}
```

Файл Bron.h

```
#include <vcl.h>
#pragma hdrstop
#include "Nom.h"
#include "Bron.h"
```

```
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "Income.h"
#include "AddNewBron.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm3::Button1Click(TObject *Sender)
{
DBGrid1->DataSource=DM1->DSBron;
DBNavigator1->DataSource=DM1->DSBron;
}
void __fastcall TForm3::Button2Click(TObject *Sender)
{
Form7->ShowModal();
}
void __fastcall TForm3::Image1Click(TObject *Sender)
{
```

```
DBGrid1->DataSource=DM1->DSBron;
DBNavigator1->DataSource=DM1->DSBron;
}
void __fastcall TForm3::Button3Click(TObject *Sender)
{
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Income]");
        DM1->ADOQuery1->SQL->Add("where [Income].[Дата документу] =date");
        DM1->ADOQuery1->Parameters->ParamByName("date")->Value =
DateTimePicker1->Date;
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "На вказану дату бронювань не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
        }
}
void __fastcall TForm3::Button5Click(TObject *Sender)
{
        if(Edit1->Text.IsEmpty())
```

```
	{
		String message = "Введіть код для пошуку.";
		ShowMessage(message);
		return;
	}
	DM1->ADOQuery1->Close();
	DM1->ADOQuery1->SQL->Clear();
	DM1->ADOQuery1->SQL->Add( "select * FROM [Bron]");
	DM1->ADOQuery1->SQL->Add("where [Bron].[Код бронювання] =
code1");
	DM1->ADOQuery1->Parameters->ParamByName("code1")-
>Value=StrToInt(Edit1->Text);
	DM1->ADOQuery1->Prepared=True;
	DM1->ADOQuery1->Open();
	if(DM1->ADOQuery1->IsEmpty())
	{
		String message = "За даним кодом нічого не знайдено";
		ShowMessage(message);
		return;
	}
	else
	{
		DBGrid1->DataSource = DM1->DSQuery1;
	}
}
void __fastcall TForm3::Button6Click(TObject *Sender)
{
	if(Edit2->Text.IsEmpty())
```

```
        {
                String message = "Введіть код для пошуку.";
                ShowMessage(message);
                return;
        }
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Bron]");
        DM1->ADOQuery1->SQL->Add("where [Bron].[Код товару] = code1");
        DM1->ADOQuery1->Parameters->ParamByName("code1")-
>Value=StrToInt(Edit2->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
        }
}
void __fastcall TForm3::Button4Click(TObject *Sender)
{
   Form15->ShowModal();
```

```
}
void __fastcall TForm3::FormShow(TObject *Sender)
{
   DBGrid1->DataSource = DM1->DSBron;
}
```

Файл AddNewBron.h

```
#include <vcl.h>
#pragma hdrstop
#include "AddNewBron.h"
#include "NomChoiseForm.h"
#include "DM.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm15 *Form15;
//---------------------------------------------------------------------------
__fastcall TForm15::TForm15(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm15::Button3Click(TObject *Sender)
{
        if(Form10->ShowModal() == mrOk)
        {
                int CodeTovaru = Form10->DBEditCodeTovaru->Text.ToInt();
                String name = Form10->DBEditName->Text;
      int Count = Form10->DBEditCount->Text.ToInt();
```

```
            this->Edit3->Text = CodeTovaru;
            this->Edit4->Text = name;
      this->Label10->Caption = Count;
        }
}
void __fastcall TForm15::FormShow(TObject *Sender)
{
        //Шукаємо та заповнюємо останній індекс + 1
        //---------------------------------------
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add("select top 1 [Код бронювання] From [Bron]");
        DM1->ADOQuery1->SQL->Add("order by [Код бронювання] desc");
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
   Edit1->Text = (DM1->ADOQuery1->FieldByName("Код бронювання")->AsInteger) + 1;
   DM1->ADOQuery1->Close();
        //---------------------------------------
        //Заповнюємо сьогодняшню дату
        //---------------------------------------
        MaskEdit1->Text = Date().FormatString("dd/mm/yyyy");
}
void __fastcall TForm15::Button1Click(TObject *Sender)
{
        if(Edit1->Text.IsEmpty() || Edit3->Text.IsEmpty() || Edit4->Text.IsEmpty() || Edit6->Text.IsEmpty()
```

```
    || MaskEdit1->Text.IsEmpty() || Edit7->Text.IsEmpty() || Edit8->Text.IsEmpty())
    {
  String message = "Не всі параметри заповнено, прохання заповнити всі параметри.";
        ShowMessage(message);
        return;
    }
    DM1->ADOBron->Insert();
    DM1->ADOBron->FieldByName("Код бронювання")->AsInteger = StrToInt(Edit1->Text);
    DM1->ADOBron->FieldByName("Дата замовлення")->AsDateTime = StrToDate(MaskEdit1->Text);
    DM1->ADOBron->FieldByName("Код товару")->AsInteger = StrToInt(Edit3->Text);
    DM1->ADOBron->FieldByName("Найменування товару")->AsString= Edit4->Text;
    DM1->ADOBron->FieldByName("Статус бронювання")->AsString = Edit6->Text;
    DM1->ADOBron->FieldByName("Кількість")->AsInteger = StrToInt(Edit7->Text);
    DM1->ADOBron->FieldByName("Ціна")->AsInteger = StrToInt(Edit8->Text);
    DM1->ADOBron->Post();
  //Пошук та редагування кількості товару.
  DM1->ADOQuery1->Close();
    DM1->ADOQuery1->SQL->Clear();
    DM1->ADOQuery1->SQL->Add("update [Nomenclatura] set [Кількість] = [Кількість] - quantity");
```

```
        DM1->ADOQuery1->SQL->Add("Where [Nomenclatura].[Код товару] = code2");
        DM1->ADOQuery1->Parameters->ParamByName("code2")->Value=StrToInt(Edit3->Text);
        DM1->ADOQuery1->Parameters->ParamByName("quantity")->Value=StrToInt(Edit7->Text);
        DM1->ADOQuery1->ExecSQL();
        DM1->ADONomenclatura->Close();
        DM1->ADONomenclatura->Open();
}
void __fastcall TForm15::Edit7Exit(TObject *Sender)
{
        if(!Edit7->Text.IsEmpty() && !Label10->Caption.IsEmpty() && Edit7->Text.ToInt() > Label10->Caption.ToInt())
        {
                String message = "Не можливо забронювати кількість більшу, ніж є в наявності.";
                ShowMessage(message);
      Edit7->Text = "";
      return;
        }
}
```

Файл BronChoiseForm.h

```
#include <vcl.h>
#pragma hdrstop
#include "BronChoiseForm.h"
```

```
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Main.h"
#include "DM.h"
#include "Chek.h"
#include "Income.h"
#include "AddNewBron.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm16 *Form16;
//---------------------------------------------------------------------------
__fastcall TForm16::TForm16(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm16::Button5Click(TObject *Sender)
{
        if(Edit1->Text.IsEmpty())
        {
                String message = "Введіть код для пошуку.";
                ShowMessage(message);
                return;
        }
```

```
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Bron]");
        DM1->ADOQuery1->SQL->Add("where [Bron].[Код бронювання] =
code1");
        DM1->ADOQuery1->Parameters->ParamByName("code1")-
>Value=StrToInt(Edit1->Text);
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "За даним кодом нічого не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
        }
}
void __fastcall TForm16::Button3Click(TObject *Sender)
{
        DM1->ADOQuery1->Close();
        DM1->ADOQuery1->SQL->Clear();
        DM1->ADOQuery1->SQL->Add( "select * FROM [Income]");
        DM1->ADOQuery1->SQL->Add("where [Income].[Дата документу] =date");
        DM1->ADOQuery1->Parameters->ParamByName("date")->Value =
DateTimePicker1->Date;
```

```
        DM1->ADOQuery1->Prepared=True;
        DM1->ADOQuery1->Open();
        if(DM1->ADOQuery1->IsEmpty())
        {
                String message = "На вказану дату бронювань не знайдено";
                ShowMessage(message);
                return;
        }
        else
        {
                DBGrid1->DataSource = DM1->DSQuery1;
        }
}
void __fastcall TForm16::Button1Click(TObject *Sender)
{
        DBGrid1->DataSource=DM1->DSBron;
}
void __fastcall TForm16::FormShow(TObject *Sender)
{
        DBGrid1->DataSource=DM1->DSBron;
        DBEditCodeBron->DataSource=DM1->DSBron;
        DBEditCodeBron->DataField = "Код бронювання";
        DBEditNomCode->DataSource=DM1->DSBron;
        DBEditNomCode->DataField = "Код товару";
        DBEditNomName->DataSource=DM1->DSBron;
        DBEditNomName->DataField = "Найменування товару";
        DBEditStatus->DataSource=DM1->DSBron;
```

```
        DBEditStatus->DataField = "Статус бронювання";
        DBEditCount->DataSource=DM1->DSBron;
        DBEditCount->DataField = "Кількість";
        DBEditPrice->DataSource=DM1->DSBron;
        DBEditPrice->DataField = "Ціна";
}
void __fastcall TForm16::Button2Click(TObject *Sender)
{
        if(DBEditStatus->Text == "Реалізовано")
        {
                String message = "Дане бронювання вже реалізоване.";
                ShowMessage(message);
                return ModalResult = mrCancel;
        }
}
```

Файл DM.h

```
#pragma hdrstop
#include "DM.h"
#include "Main.h"
#include "Nom.h"
#include "Bron.h"
#include "Orders.h"
#include "Producers.h"
#include "Chek.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma classgroup "Vcl.Controls.TControl"
```

```
#pragma resource "*.dfm"
TDM1 *DM1;
//---------------------------------------------------------------------------
__fastcall TDM1::TDM1(TComponent* Owner)
        : TDataModule(Owner)
{
}
void __fastcall TDM1::DataModuleCreate(TObject *Sender)
{
ADOConnection1->Connected = true;
ADOChek->Open();
ADOOrders->Open();
ADONomenclatura->Open();
ADOKlienti->Open();
ADOBron->Open();
ADOSale->Open();
ADOIncome->Open();
}
void __fastcall TDM1::DataModuleDestroy(TObject *Sender)
{
 ADOConnection1->Connected = false;
}
```

ДОДАТОК Б. Інтерфейс системи товарообігу магазину одягу

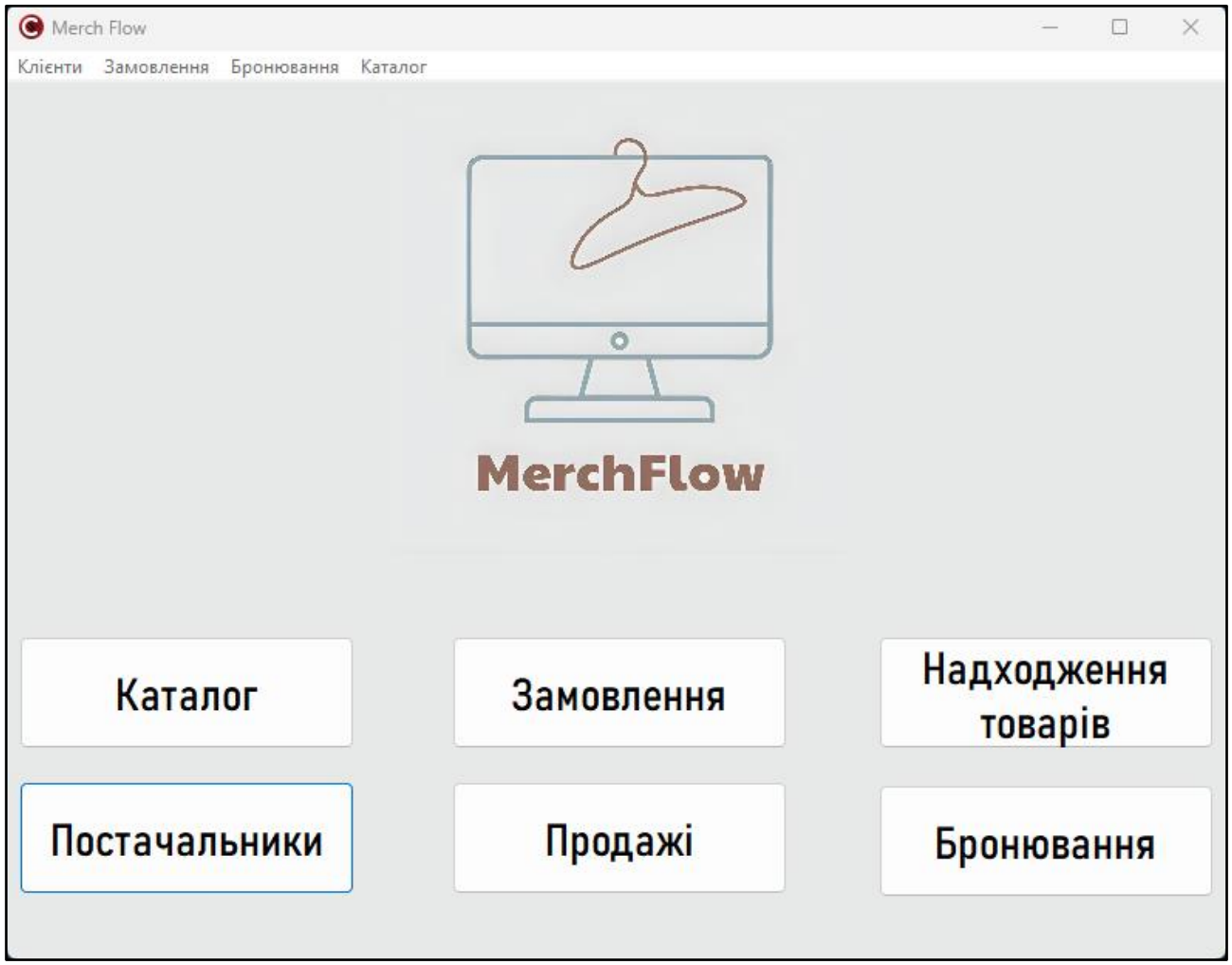


Рисунок Б.1 – Головна сторінка

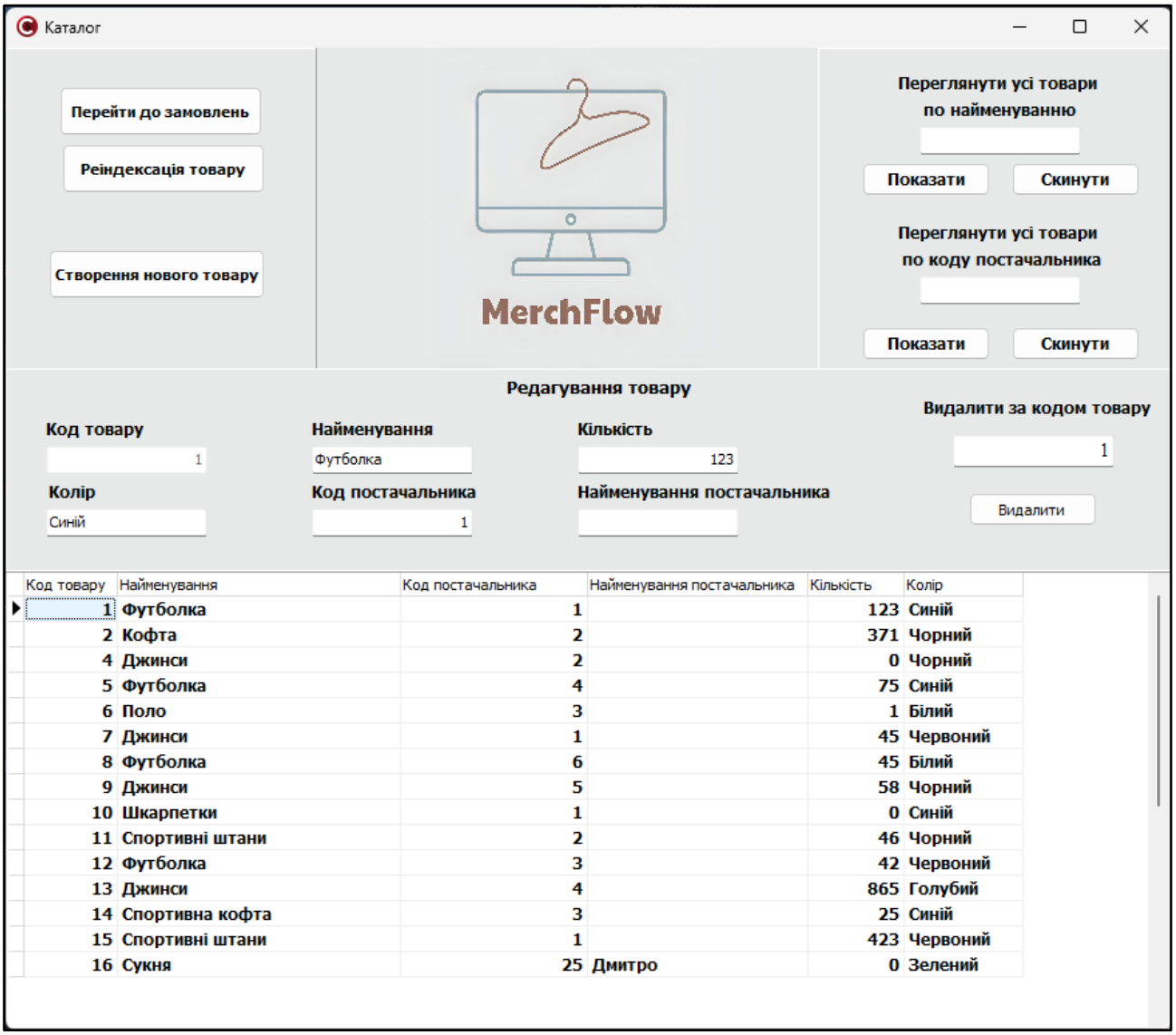


Рисунок Б.2 – Вікно «Каталог»

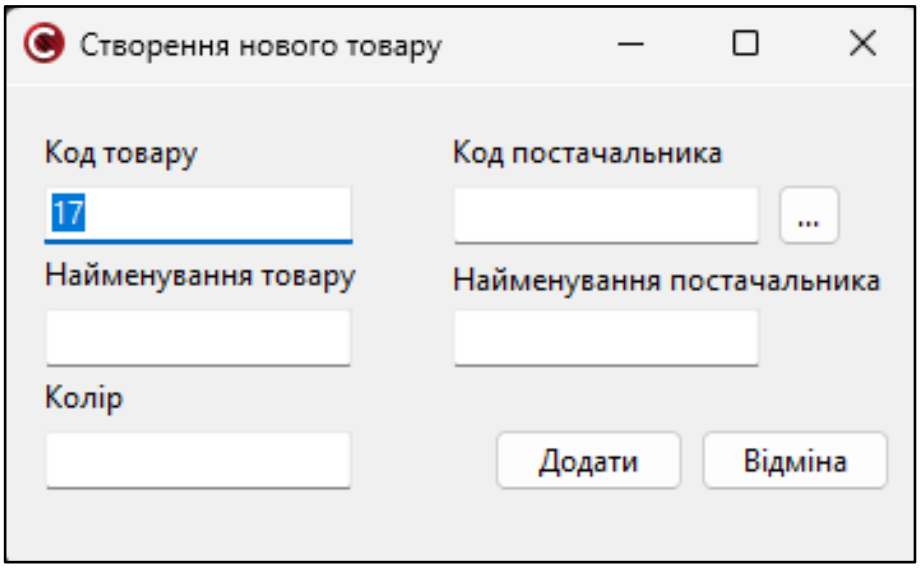


Рисунок Б.3 – Вкладка «Створення нового товару»

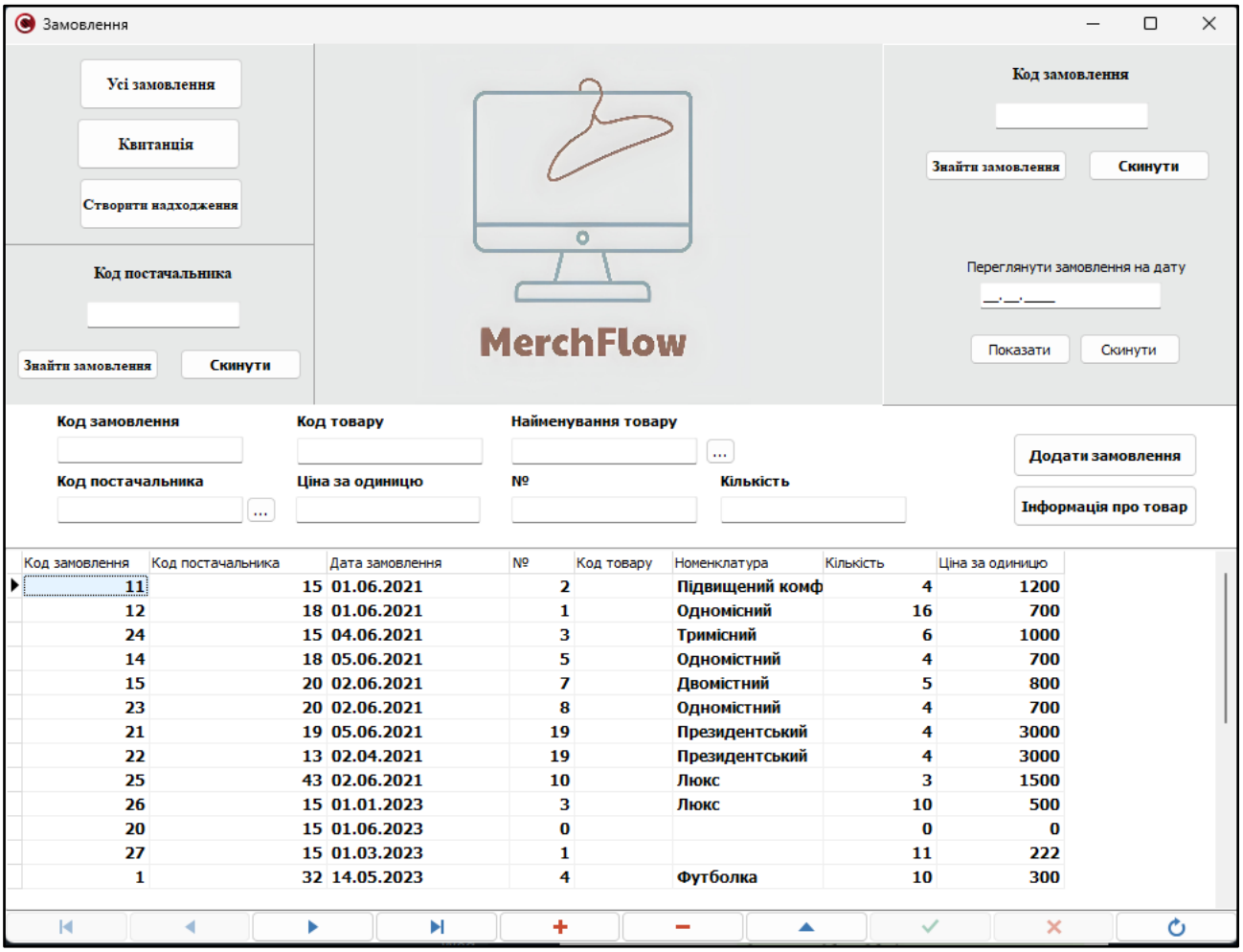


Рисунок Б.4 – Вікно «Замовлнння»

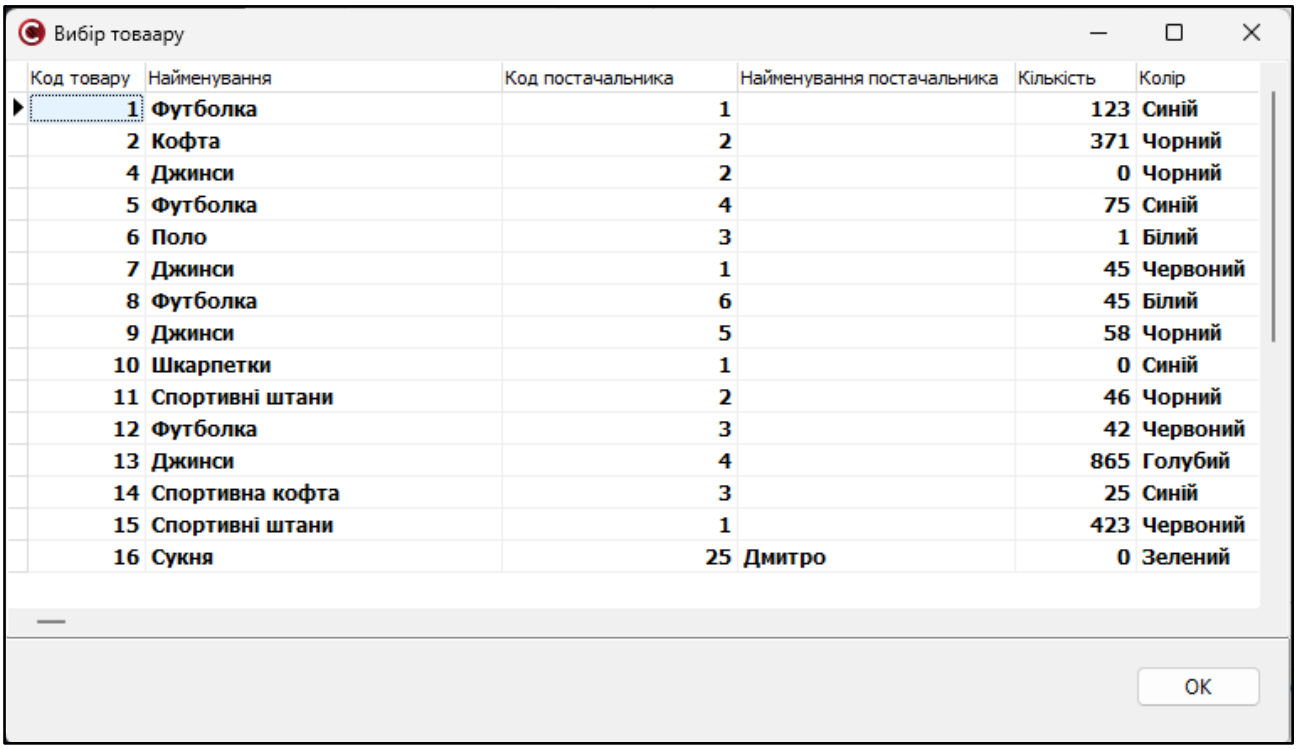


Рисунок Б.5 – Вкладка «Вибір товару»

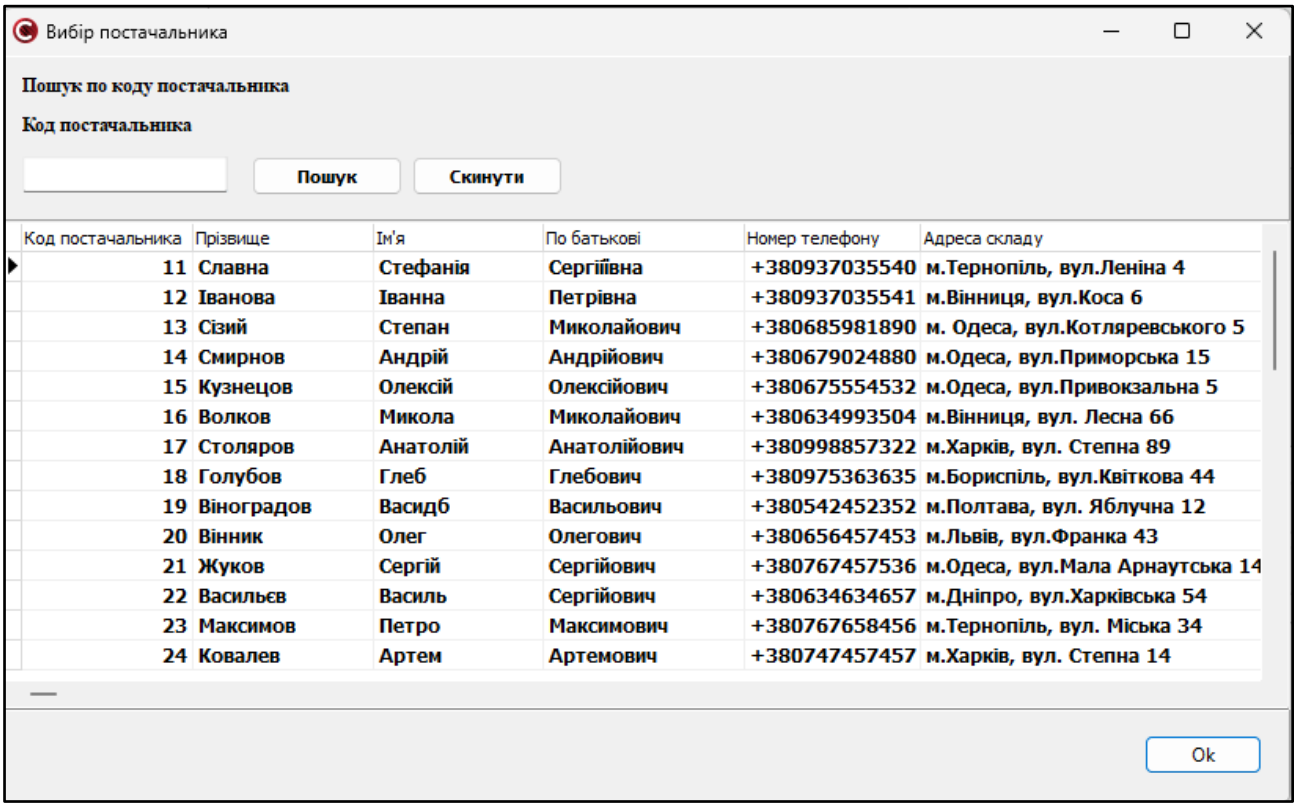


Рисунок Б.6 – Вкладка «Вибір постачальників»

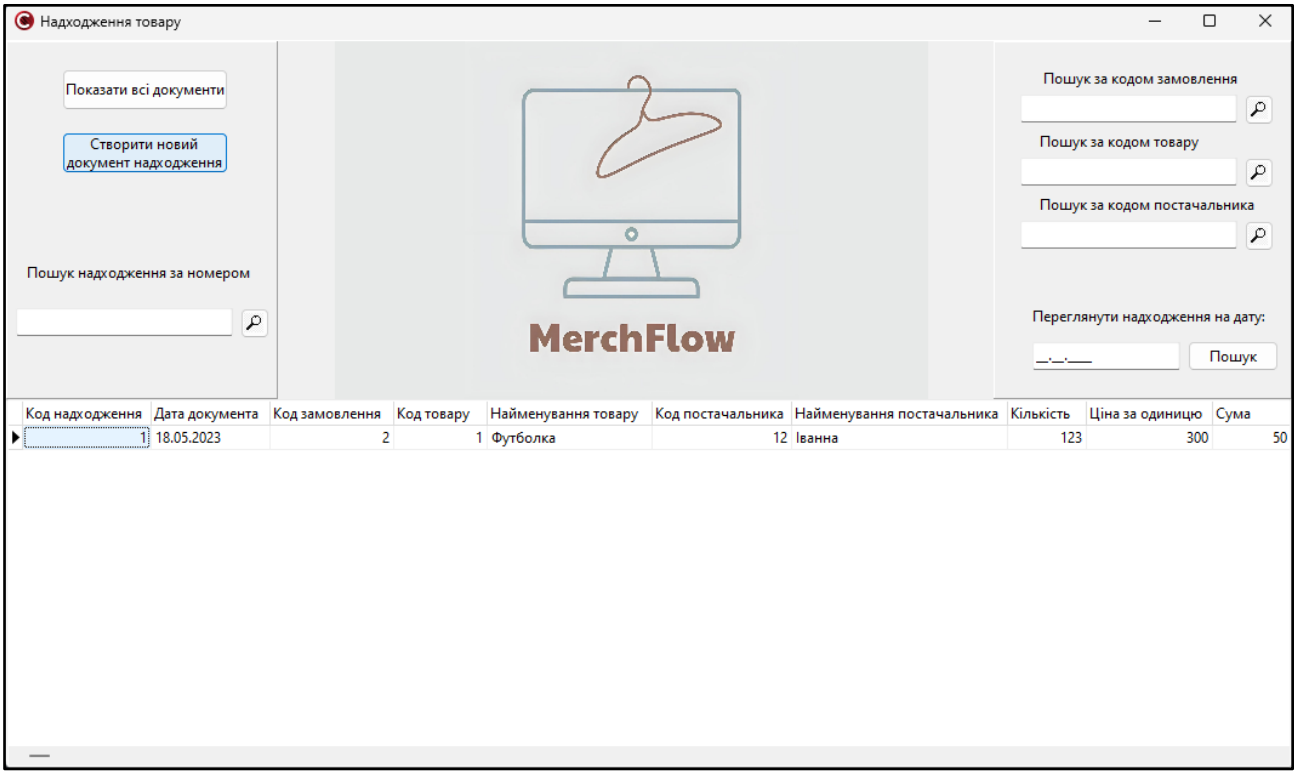


Рисунок Б.7 – Вікно «Надходжння товарів»

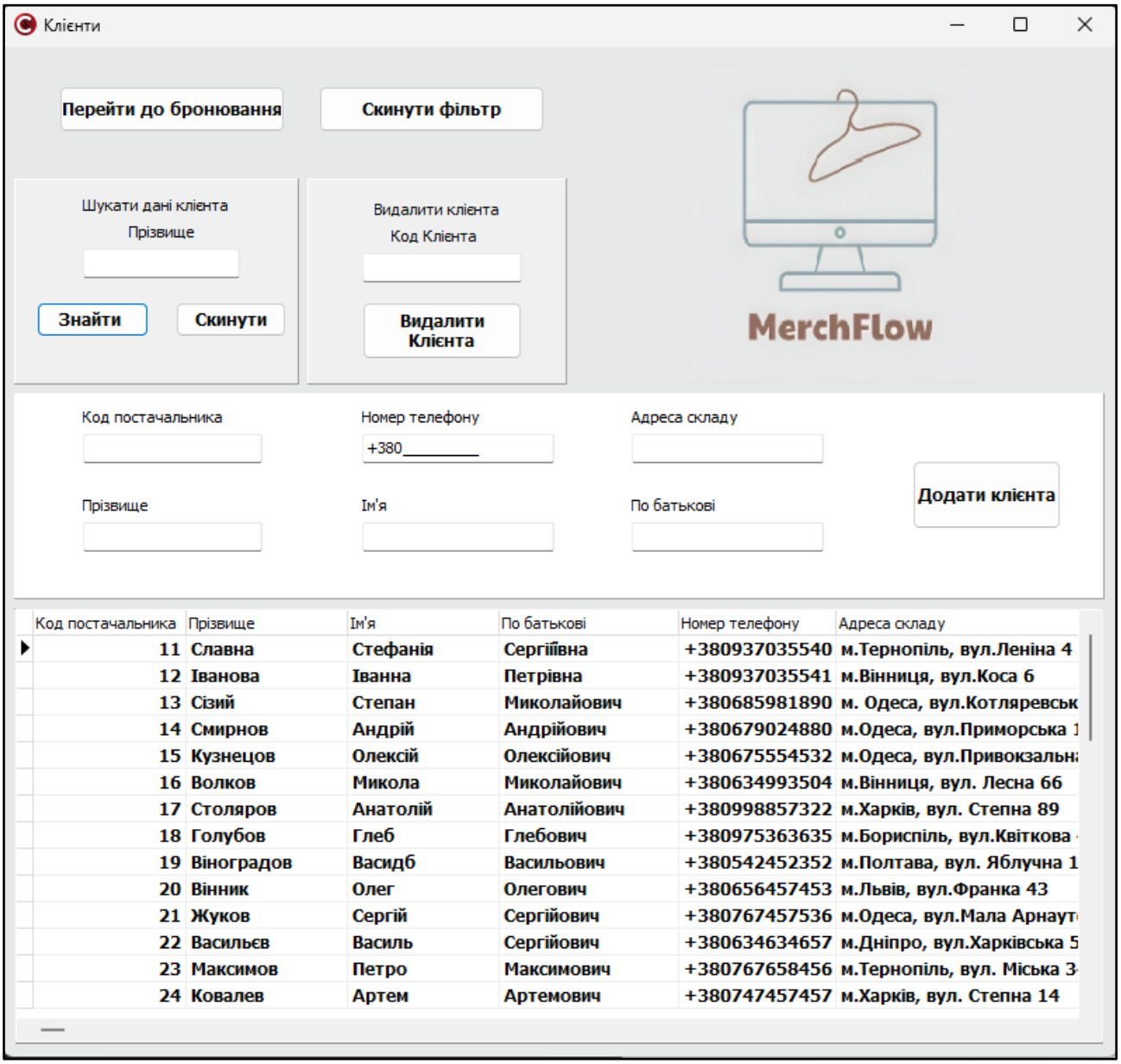


Рисунок Б.8 – Вікно «Клієнти»

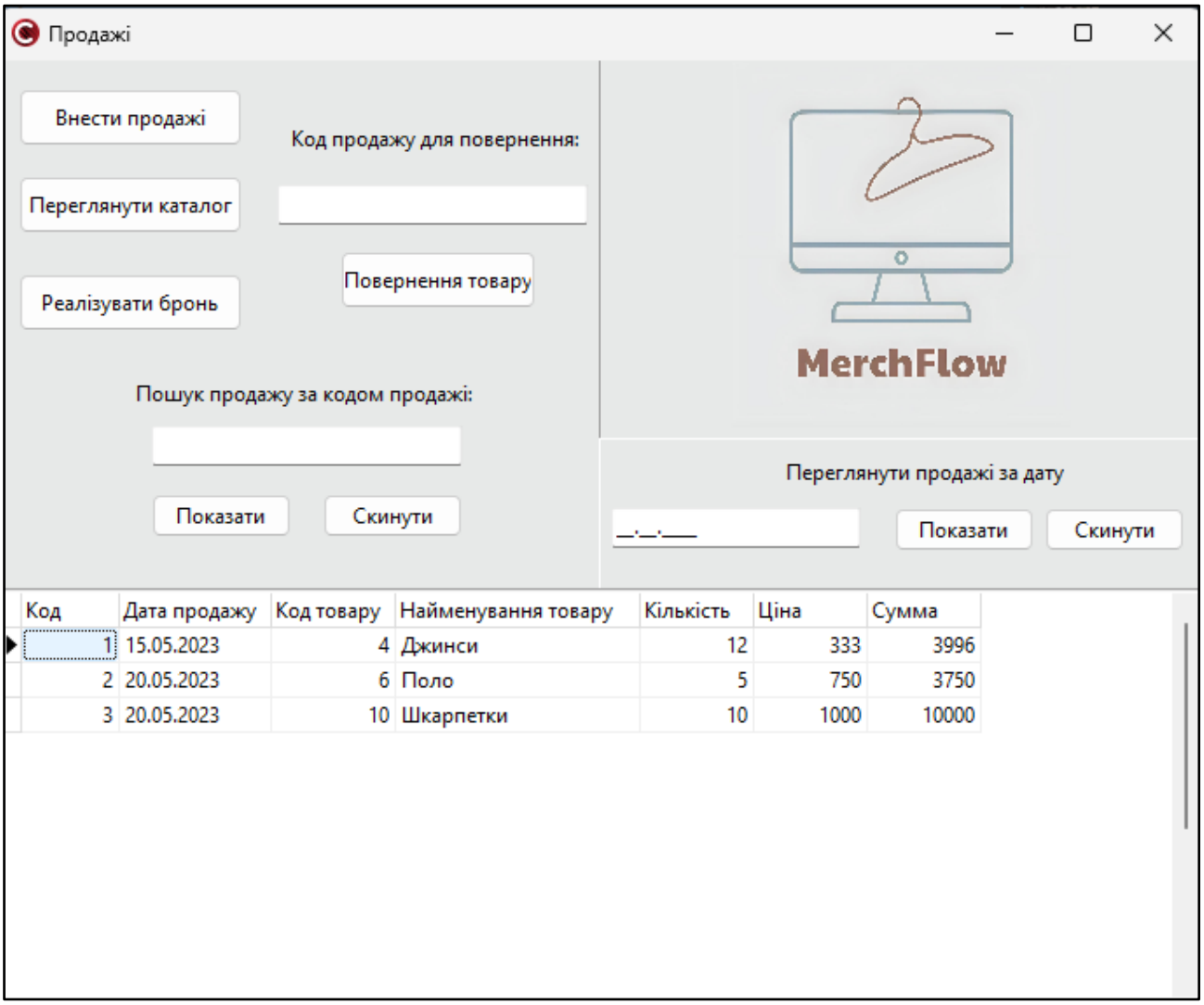


Рисунок Б.9 – Вікно «Продажі»

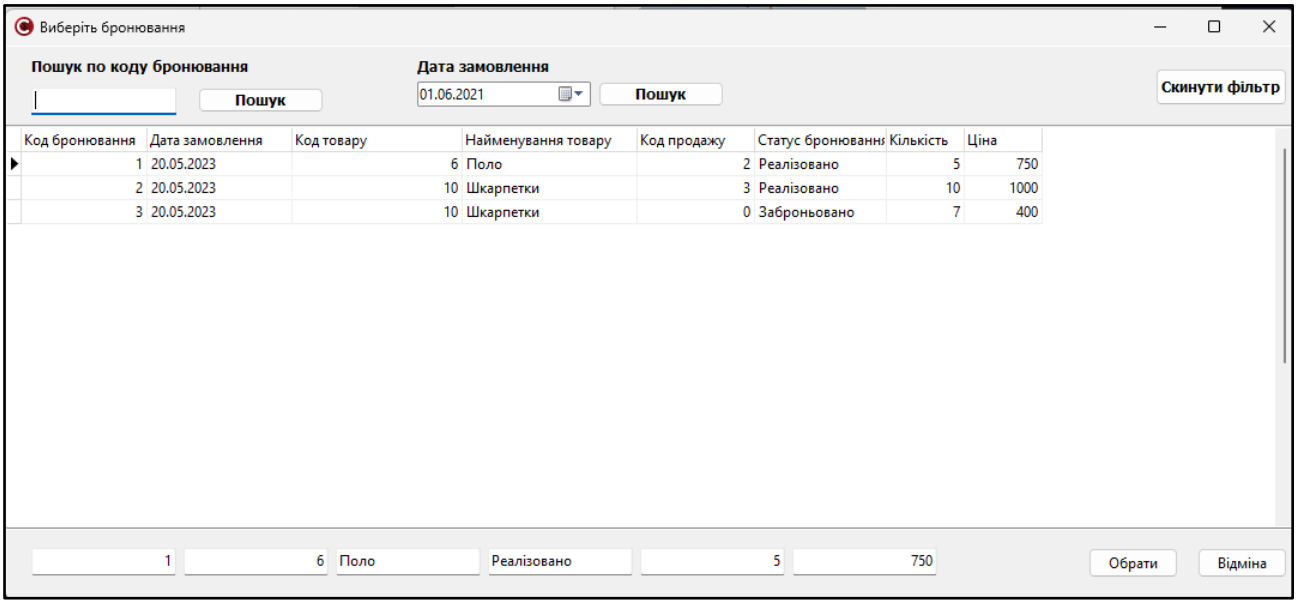


Рисунок Б.10 – Вкладка «Вибір бронювання»

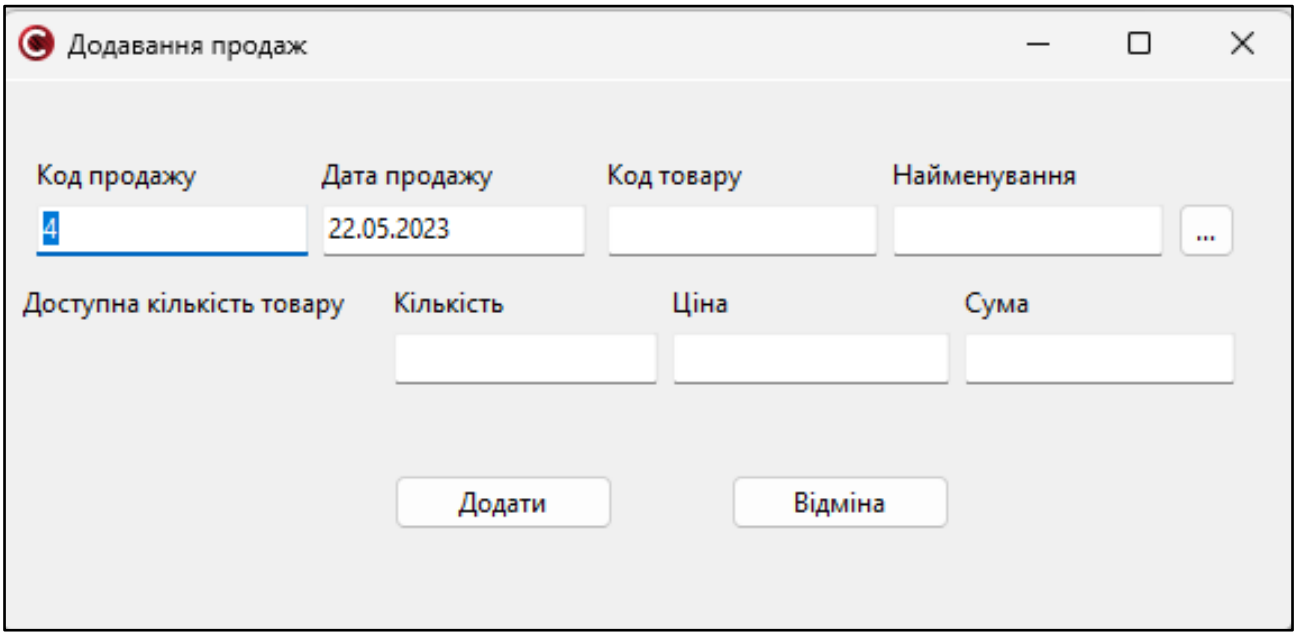


Рисунок Б.11- Вкладка «Додавання продаж»

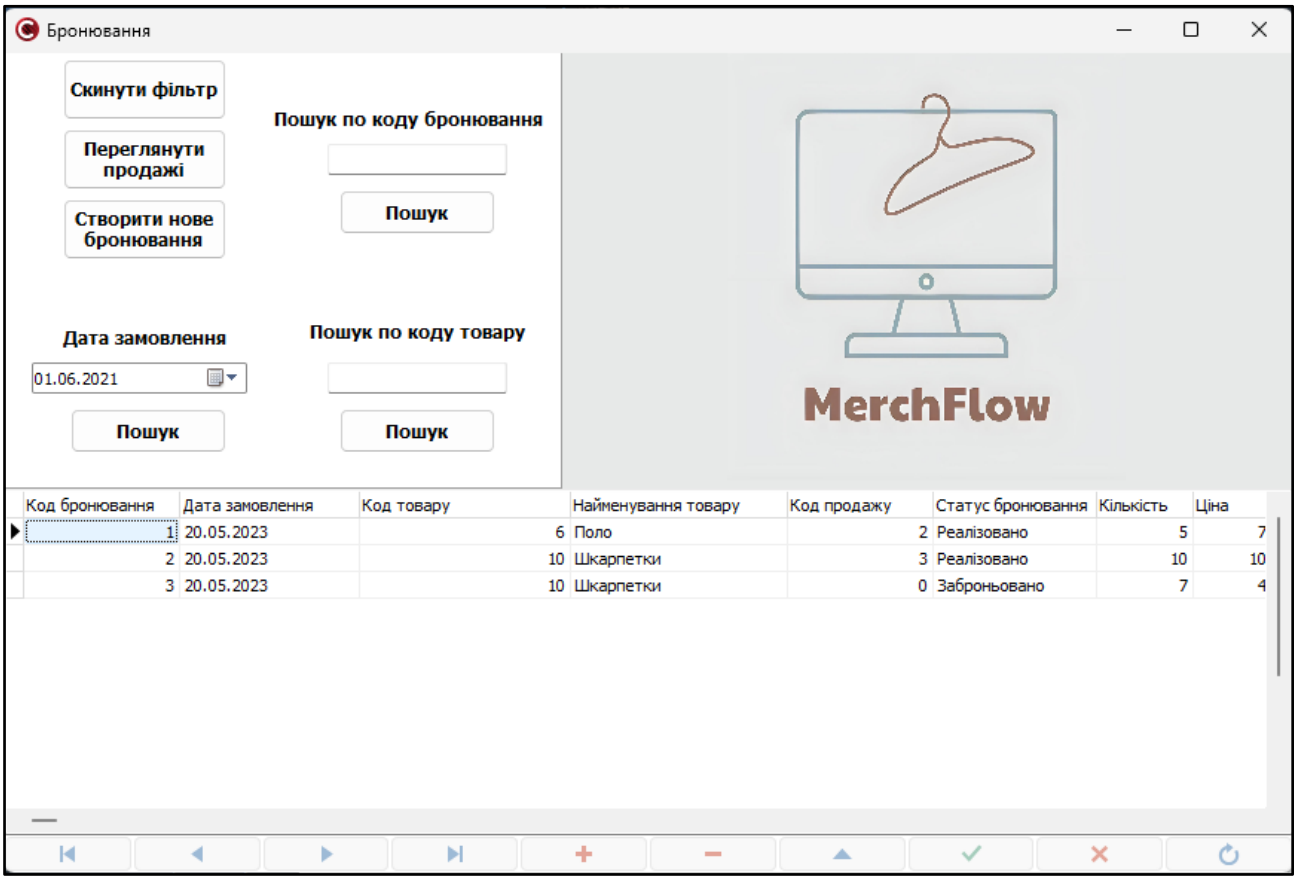


Рисунок Б.12 – Вікно «Бронювання»

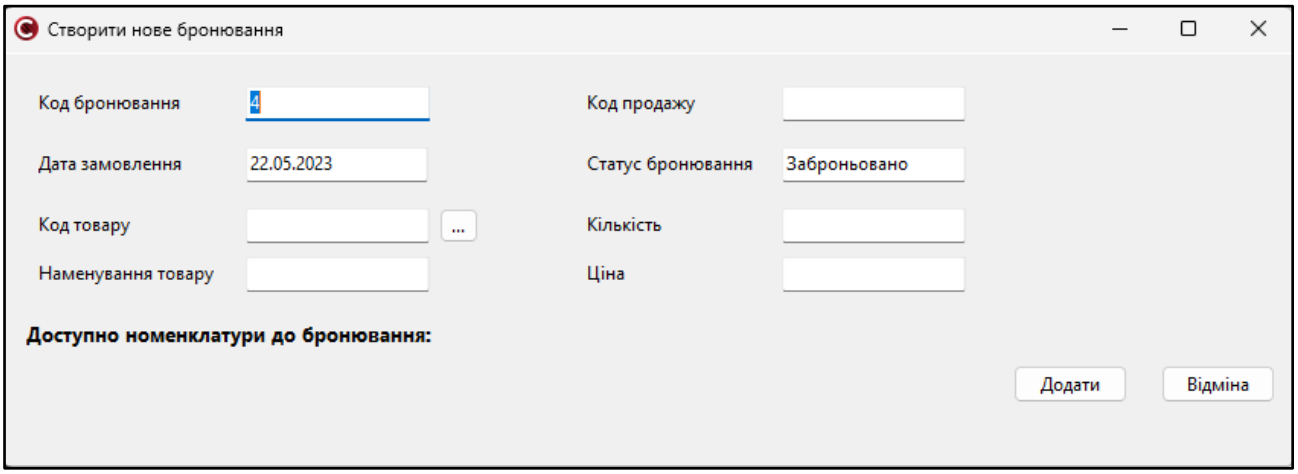


Рисунок Б.13 – Вкладка «Створити бронювання»

ДОДАТОК В. Презентація

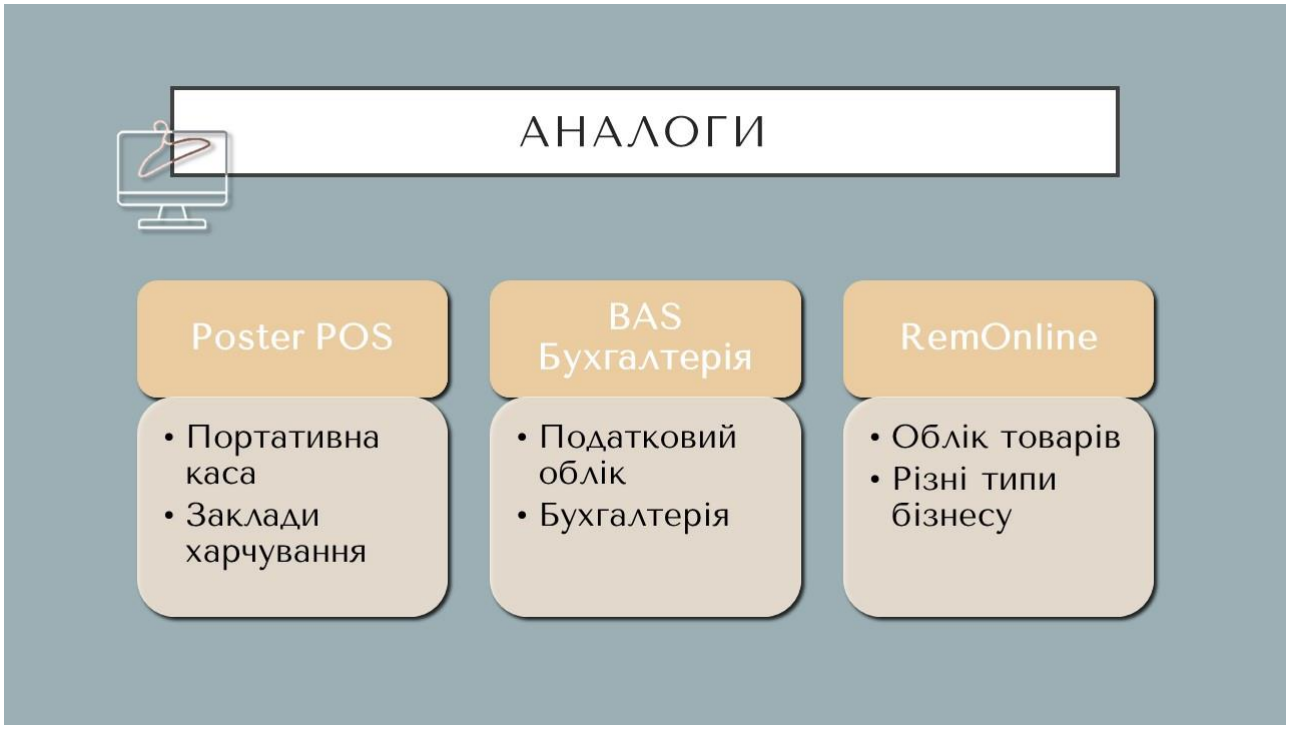
СИСТЕМА ТОВАРООБІГУ МАГАЗИНУ ОДЯГУ «MERCHFLOW»

Виконала — Аріана СЛАВІЦЬКА
Науковий керівник — Тетяна ДЕМКІВСЬКА

МЕТА РОБОТИ

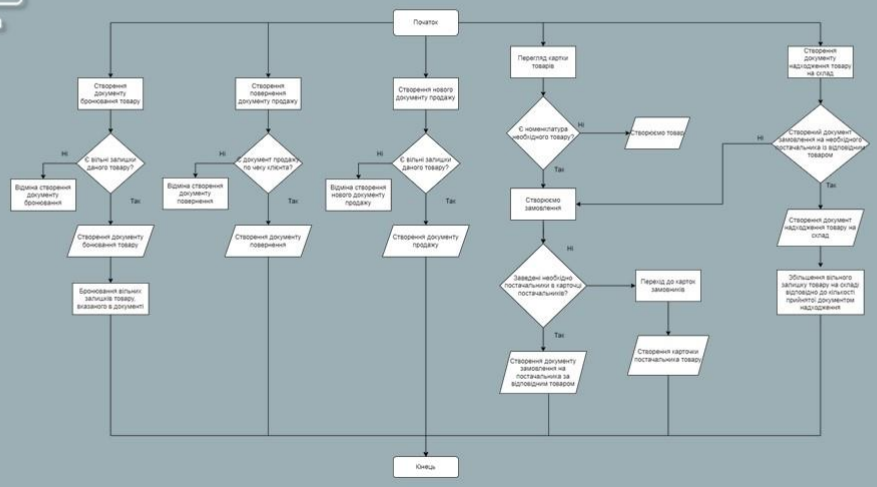
Розробка та створення системи товарообігу.

Аналіз проблем автоматизації процесу обробки інформації щодо обліку товарів та розробка пропозицій щодо вдосконалення її діяльності.


АНАЛОГИ
Poster POS
• Портативна каса
• Заклади харчування
BAS Бухгалтерія
• Податковий облік
• Бухгалтерія
RemOnline
• Облік товарів
• Різні типи бізнесу


ОСНОВНІ ФУНКЦІЇ
Каталог товарів
База постачальників
Надходження товарів
Список замовлень
Продажі
Бронювання товарів

БЛОК-СХЕМА



ЗАСОБИ РОЗРОБКИ

Мова програмування

- C++ (С плюс плюс) є скомпільованою, статично типізованою мовою програмування загального призначення

Середовище програмування

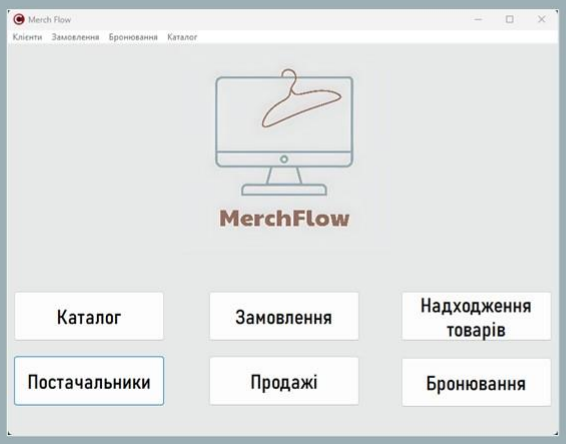
- RAD Studio - це інтегроване середовище розробки програмного забезпечення (IDE), яке дозволяє розробникам створювати програми для різних платформ

База даних

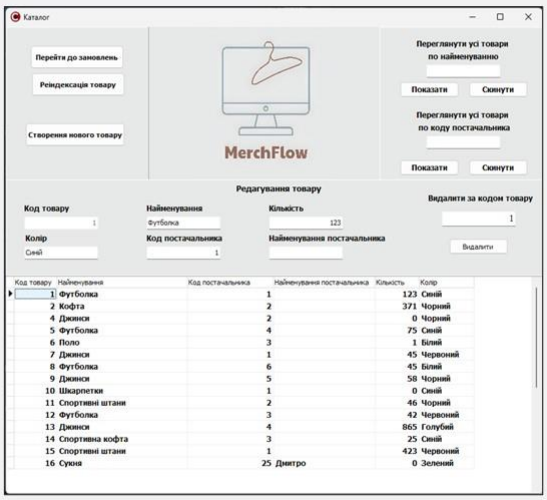
- Microsoft Access – це система управління реляційними базами даних (СУБД)

ІНТЕРФЕЙС «MERCHFLOW»

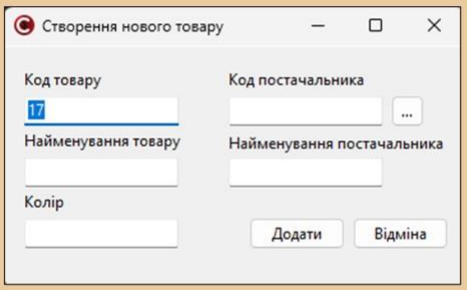
Головна сторінка «MERCHFLOW»



Вікно «Каталог»



Вкладка «Створення нового товару»




ВИСНОВКИ
В результаті було проведено детальне ознайомлення та дослідження теоретичної та практичної частин розробки системи «Merchflow», а саме:
Постановка мети та задачі
Розгляд та аналіз аналогів
Розробка основних функцій
Конструювання блок-схеми
Вибір засобів розробки
Створення системи «Merchflow»
©2023 Аріана СЛАВІЦЬКА