ДИПЛОМНА РОБОТА

БАКАЛАВРА

ДРБ. ІП - 20.00.00.000 ПЗ

Група ІП-18-1

Євстратов Олег

2022

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: "Розробка програмного сервісу аптечного складу"

(назва згідно з наказом ректора)

Спеціальність – 121 "Інженерія програмного забезпечення"

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІП - 20.00.00.000 ПЗ

(позначення)

Консультанти:

з питань нормоконтролю

доц., к.т.н /В.Я. Піх/
(посада) (підпис) (дата)(розшифровка підпису)

Студент

ІП-18-1 /О.Ю.Євстратов/
(шифр групи) (підпис) (дата)(розшифровка підпису)

Керівник роботи

д.т.н.,проф. /М.С.Пасєка/
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту
Завідувач кафедри

д.т.н.,проф. /В.І.Шекета/
(посада) (підпис) (дата) (розшифровка підпису)

2022
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
проф. В.І. Шекета
“ ____ ” _____2022 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Євстратову Олегу Юрійовичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) **“ Розробка програмного сервісу аптечного складу”**

керівник проекту (роботи) Пасєка Микола Степанович

затверджені наказом закладу вищої освіти від **“** ____ ” травня 2022 р. № _____

2. Строк подання студентом проекту (роботи) 17 червня 2022 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Загальна характеристика об’єкту дослідження
2. Реалізація методів та засобів проектування процесу автоматизації підприємства
3. Практична реалізація програмного рішення
4. Представлення інтерфейсу
5. Тестування програмного продукту

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Структура передачі даних між компонентами ПЗ на оптовому складі
2. Структура даних, що отримуються програмним комплексом
3. Структура бази даних, що використовується в програмному комплексі
4. UML діаграма основного модуля (клас TMainForm)
5. UML діаграма класу для роботи з прайсом TPriceForm

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **25 травня 2022 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Загальна характеристика та опис об’єкту дослідження	26.05.2022	виконано
2	Реалізація методів та засобів проектування процесу автоматизації підприємства	30.05.2022	виконано
3	Практична реалізація програмного рішення обліку для аптечного складу	02.06.2022	виконано
4	Опис інтерфейсу розробленого програмного продукту	07.06.2022	виконано
5	Виконання тестування системи	10.06.2022	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	17.06.2022	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

В дипломній роботі розроблено програмний продукт, що забезпечує базові сервіси аптечного складу і містить клієнтську (програмний модуль) і серверну частину (база даних). Реалізація даного проекту була проведена без залучення громіздких засобів роботи з базами даних, які носять універсальний характер і до того ж вимагають необхідну базу знань по теорії баз даних.

КЛЮЧОВІ СЛОВА: БАЗА ДАНИХ, ФАРМАЦЕВТИЧНІ СЕРВІСИ, КЛІЄНТ-СЕРВЕР, МОДЕЛЬ ДАНИХ, АВТОМАТИЗАЦІЯ, ПРОГРАМНИЙ КОМПЛЕКС, АПТЕЧНИЙ СКЛАД.

ANNOTATION

In the thesis a software product was developed that provides basic pharmaceutical services and contains a client (software module) and a server part (database). The implementation of this project was carried out without the use of cumbersome tools for working with databases that are universal in nature and also require the necessary knowledge base on the theory of databases.

KEYWORDS: DATABASE, PHARMACEUTICAL SERVICES, CLIENT-SERVER, DATA MODEL, AUTOMATION, SOFTWARE COMPLEX, PHARMACY WAREHOUSE.

ЗМІСТ

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ 8
ВСТУП 9

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕННЯ. ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОГРАМНИХ РІШЕНЬ
1.1 Коротка характеристика об'єкту проектування 11
1.2 Характеристика і аналіз існуючої системи, перспективи її розвитку 11
1.3 Змістовна постановка завдань, що вирішуються в дипломній роботі 13
1.4 Огляд і аналіз відомих проектних рішень 15
1.5 Аналіз мобільних додатків для пошуку медикаментів……20
Висновки до розділу 24

2 РЕАЛІЗАЦІЯ МЕТОДІВ ТА ЗАСОБІВ ПРОЕКТУВАННЯ ПРОЦЕСУ АВТОМАТИЗАЦІЇ ФАРМАЦЕВТИЧНИХ СЕРВІСІВ АПТЕЧНОГО СКЛАДУ
2.1 Розробка структури моделі ……..……...25
2.2 Методи та підходи до проблеми оптимізації баз даних 29
2.3 Алгоритмізація процесу автоматизації обробки даних для поставок медичних товарів……33
Висновки до розділу 44

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ РОБОТИ АПТЕЧНОГО СКЛАДУ
3.1 Реалізація сервера бази даних 45
3.2 Реалізація моделі клієнта програмного сервісу 47
3.3 Короткий опис основних функцій програми 57

Висновки до розділу 62

ВИСНОВКИ 63

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 64

ДОДАТКИ

БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

АБД – адміністратор бази даних

ЛЗ – лікарські засоби

АСУ – автоматизована система управління

МД – моделі даних

РМД - реляційна модель даних

ММД - мережева модель даних

ІМД - ієрархічна модель даних

ВСТУП

Повнофункціональна система автоматизації оптового підприємства охоплює всю господарську діяльність: постачання, збут, управління матеріальними потоками, облік кадрів, бухгалтерський облік.

Автоматизовані функції прийому, обліку, відвантаження товару по серіях, постачання і збуту, аналізу і статистики, кадрового обліку. Забезпечена комплексна система бухгалтерського обліку, відповідна бізнес-задачам підприємства (мережева версія, більше 20 робочих місць в мережі).

Аптек достатньо багато, тому у покупця завжди є вибір, в яку з найближчих піти. Природно, в ту, де вибір більше, ціни нижче, є консультанти і немає черг. А це залежить від того, наскільки оптимально організовано ведення торгівлі. Отже в хорошій організації зацікавлені не тільки власники аптеки, але і покупці.

Спершу розглянемо типові схеми організації постачань в аптеках. Аптеки ведуть роздрібну торгівлю лікарськими препаратами, купуючи їх безпосередньо у фірм - постачальників або через власний склад. Як правило, аптеки або аптечні кіоски об'єднані в аптечну мережу. У такому разі торгівля ведеться через склад або мережу складів фармацевтичної компанії.

Аптеки, як і інші магазини, можуть бути влаштовані по одній з двох схем обслуговування покупців: або прилавкова торгівля, або метод самообслуговування. Система самообслуговування привертає на 30% більше покупців, але в невеликих аптеках, а тим більше в аптечних кіосках торгівля ведеться з вітрини через касу. Тому оптимізація процесу торгівлі в них досягається за рахунок хорошої організації складу, управління постачаннями і прискорення обслуговування покупців. Тут на допомогу людині приходить комп'ютер.

Автоматизація - знайомий багатьом термін, що на практиці значно полегшує життя власникам і персоналу торгових підприємств. Процес автоматизації для аптек полягає в установці деякого додаткового торгового

устаткування і програмних продуктів, що здійснюють ведення обліку, звітності, документації і т.п. Отже, вартість проведення автоматизації залежить від ціни програмних продуктів, тобто від їх складності; і від кількості необхідного устаткування. Понизити цю вартість можна за рахунок установки програм, які працюють саме з тим устаткуванням, яке є у вашій торговій точці. Оскільки автоматизується, як правило, мережа аптек або аптека із зовнішнім складом, то доводиться об'єднувати різні марки торгового устаткування, отже, бажаний програмний продукт, що працює широким спектром устаткування.

Автоматизація дозволяє не тільки прискорити ведення торгівлі, але значно полегшити його. Ведення торгівлі в аптеках має деякі особливості. У більшості фармацевтичних препаратів є певний термін реалізації, який необхідно враховувати не тільки при закупівлі партії даного товару, але і в самому процесі реалізації. Для торгівлі деякими ліками необхідні сертифікати, які також мають певний термін дії. При організації торгівлі доводиться враховувати і ці чинники, тобто утворюються додаткові документи, ускладнюється ведення документації, це в цілому уповільнює процес. Система автоматизації сама «стежить» за цими даними і дозволяє управляти постачаннями залежно від термінів реалізації і ліцензування. Вся інформація зберігається на жорсткому диску комп'ютера, тобто потрібні цифри знаходяться натисненням декількох клавіш, а не перериванням всієї документації вздовж і поперек.

Автоматизація торгівлі для аптек спрощує весь процес торгівлі і виключає можливість появи помилок, наприклад, при випадкових описках в звітах і накладних. Це обумовлюється тим, що весь облік здійснюється за допомогою програм і спеціального устаткування, а не за допомогою паперів і ручних записів.

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕННЯ. ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОГРАМНИХ РІШЕНЬ

1.1 Коротка характеристика об'єкту проектування

Автоматизація аптечного складу дозволяє автоматизувати консолідований облік і замовлення ліків в рамках групи аптек (оптових складів, торгових залів аптек, аптечних кіосків і т.д.).

Особливості автоматизації аптечного складу:

- безперервна підтримка (автоматизація) реальних залишків ліків;
- автоматизація індивідуального доступу до інформації по аптеці;
- контроль залишків (облік і контроль залишків з моменту приходу);
- відпустка медикаментів аптекам в різних упаковках;
- гнучкі засоби формування знижок в аптеці;
- формування зображення і анотації ліків в аптеці ;
- розвинені засоби доступу до даних;
- настройка на особливості обліку в конкретній аптеці.

Програма автоматизації оптової торгівлі лікарськими засоби повинна виконувати :

- ведення обліку оптового складу і роздрібних підрозділів;
- універсальний механізм ціноутворення залежно від виду підрозділу (опт або роздріб), типу операції (прихід, внутрішнє переміщення), типу і цінової групи товару, групи покупців;
- контроль торгової націнки на перевищення гранично допустимою;
- резервування товару;
- вивантаження прайс-листа в Excel;
- автоматизований контроль фальсифікованих ЛЗ і термінів придатності;
- автоматизований обмін даними між об'єктами аптечної мережі і центральним оптовим складом;
- контроль витрати лікарських засобів;

- формування замовлення товару за наслідками аналізу мінімального залишку, інтенсивності витрати, часу виконання замовлення;

1.2 Характеристика і аналіз існуючої системи, перспективи її розвитку

Дана система призначена для створення замовлення не використовуючи паперовий прайс-лист. Електронний прайс-лист оновлюється декілька раз на добу (максимум до 24 разів).

Легкий пошук препарату, миттєве відображення загальної суми замовлення, вибір комунікаційних програм, можливість збереження незавершеного замовлення дає можливість швидко і зручно зробити замовлення.

Існуючий програмний комплекс призначений для формування замовлень від клієнта (які передаються телефоном, факсом, тощо) і передачі отриманих даних у основну програму для обробки електронних замовлень і формування пакету документів (програмний комплекс «ФАРМ»).

Найважливішими функціями для розроблюваної системи є швидкий пошук як клієнта, так і необхідного товару. Існує можливість пошуку як по коду клієнта чи товару, так і по назві.

Програмний комплекс надає коротку інформацію про стан оплат дебітора (переплата чи заборгованість), умови за якими проводиться продаж лікарських засобів, умови відтермінування клієнта, ліцензії, знижки чи націнки для конкретного клієнта, що дає змогу оператору швидко прийняти і сформувати замовлення від клієнта.

Також забезпечено отримання інформації про конкретний лікарський засіб (ціни, терміни придатності, реєстрація препарату, приналежність конкретного препарату до групи заборонених, група продажу (вітчизняні, зарубіжні, з ПДВ). Відображаються залишки товару на складі, зарезервовані препарати, прихідні

товари, упаковку. Є також можливість отримання ширшої інформації про конкретний препарат.

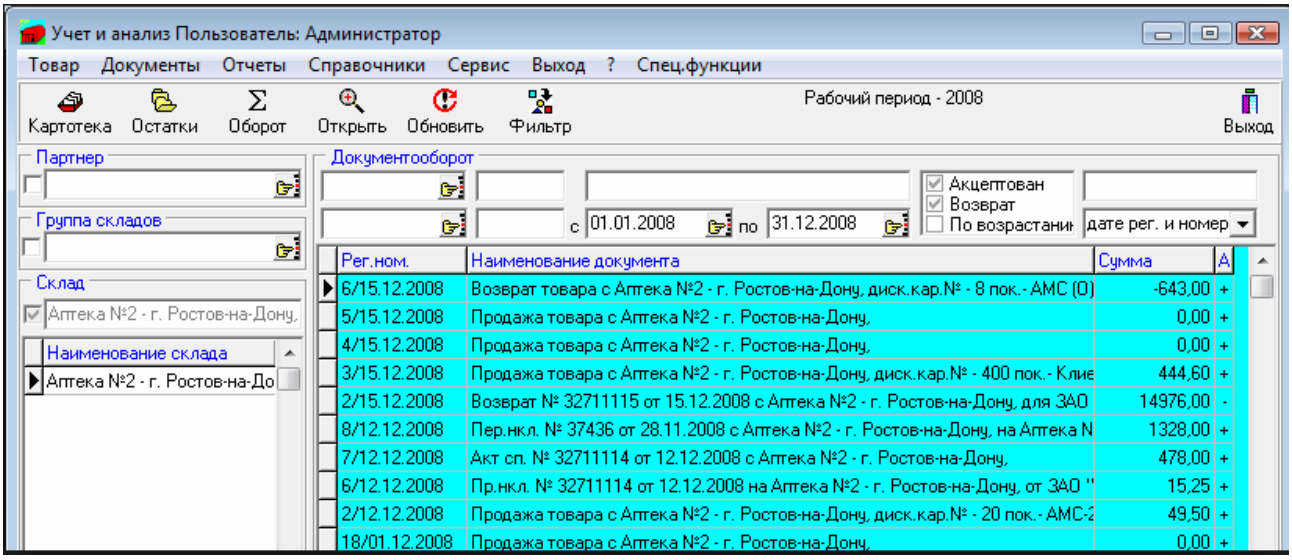


Рисунок 1.1 – Програмний комплекс “ФАРМ”

Незважаючи на вищеописані можливості програмний комплекс перебуває в процесі постійного вдосконалення. Оновлення даних забезпечується за допомогою спеціальних запитів Microsoft SQL (що дозволяють отримувати необхідні дані з основного програмного комплексу «ФАРМ»). Існує також можливість легко перейти на будь-яку іншу систему управління базами даних, що забезпечується легкою конфігурованістю додатку.

1.3 Змістовна постановка завдань, що вирішуються в дипломній роботі

В даній дипломній роботі вирішують завдання щодо побудови системи автоматизації прийому і обробки заявок від клієнтів аптечного складу.

Автоматизація - знайомий багатьом термін, що на практиці значно полегшує життя власникам і персоналу торгових підприємств. Процес автоматизації для аптек полягає в установці деякого додаткового торгового

устаткування і програмних продуктів, що здійснюють ведення обліку, звітності, документації і т.п. Отже, вартість проведення автоматизації залежить від ціни програмних продуктів, тобто від їх складності; і від кількості необхідного устаткування. Понизити цю вартість можна за рахунок установки програм, які працюють саме з тим устаткуванням, яке є у вашій торговій точці.

Автоматизація дозволяє не тільки прискорити ведення торгівлі, але значно полегшити його. Ведення торгівлі в аптеках має деякі особливості. У більшості фармацевтичних препаратів є певний термін реалізації, який необхідно враховувати не тільки при закупівлі партії даного товару, але і в самому процесі реалізації. Для торгівлі деякими ліками необхідні сертифікати, які також мають певний термін дії. При організації торгівлі доводиться враховувати і ці чинники, тобто утворюються додаткові документи, ускладнюється ведення документації, це в цілому уповільнює весь процес. Система автоматизації сама «стежить» за цими даними і дозволяє управляти постачаннями залежно від термінів реалізації і ліцензування. Вся інформація зберігається на жорсткому диску комп'ютера, тобто потрібні цифри знаходяться натисненням декількох клавіш, а не перериванням всієї документації вздовж і поперек.

Повнофункціональна система автоматизації оптового підприємства охоплює всю господарську діяльність: постачання, збут, управління матеріальними потоками, облік кадрів, бухгалтерський облік. Локальна обчислювальна мережа підприємства включає більше 20 комп'ютерів, 1 сервер. Єдина база даних, система "електронне замовлення" - телекомунікаційний зв'язок з покупцями (більше 400 аптек і лікарень).

Проект володіє гнучкою архітектурою; технологією використання, доступною всім фахівцям підприємства; регулярно актуалізується відповідно до стратегії і цілей підприємства; забезпечує логічну цілісність; володіє корпоративними якостями.

Реалізація завдань на адміністративному рівні дозволяє отримувати зріз інформації про роботу підприємства за довільний період часу, оптимізувати

роботу з клієнтами, будувати цінову і товарну політику відповідно до змінної кон'юнктури ринку.

Проект повинен володіти гнучкою архітектурою; технологією використання, доступною всім фахівцям підприємства; регулярно актуалізується відповідно до стратегії і цілей підприємства.

Реалізація завдань на адміністративному рівні повинна дозволяти отримувати зріз інформації про роботу підприємства за довільний період часу, оптимізувати роботу з клієнтами, будувати цінову і товарну політику відповідно до змінної кон'юнктури ринку.

Система повинна забезпечити:

- автоматизацію роботи всіх структурних підрозділів і служб складу;
- інтеграцію інформаційних потоків підприємства;
- доступність ділової інформації в реальному масштабі часу;

Система повинна дозволяти:

- вести щоденний облік стану товарних запасів в кількісному і сумовому виразі, отримувати довідки про рух товару, постачальників, покупців, заборгованість за довільний період часу;
- організувати оптимальний розподіл лікарських засобів і виробів медичного призначення в аптечну мережу;
- проводити інвентаризацію у відділах складу;
- підтримувати і поповнювати довідкові масиви даних;

передавати прайс-листи і приймати замовлення на товари з використанням модемного зв'язку;

1.4 Огляд і аналіз відомих проектних рішень

1.4.1 «АСТОР: Аптека 4.5 DPOS»

Програмний продукт компанії «АСТОР ВЦ» - «Аптека v 4.5» відповідає всім вимогам до системи автоматизації торгівлі. Він підходить для аптек з будь-

якою системою обслуговування покупців. «АСТОР: Аптека 4.5 DPOS» призначений для роботи з фіскальними реєстраторами; «АСТОР: Аптека 4.5 POS» працює з POS- терміналами. Перше рішення є доступнішим, але установка POS - терміналів, устаткування і програмного забезпечення до них дозволяє оперативніше обслуговувати покупців, полегшити ведення обліку. До того ж такі системи швидше окупаються.

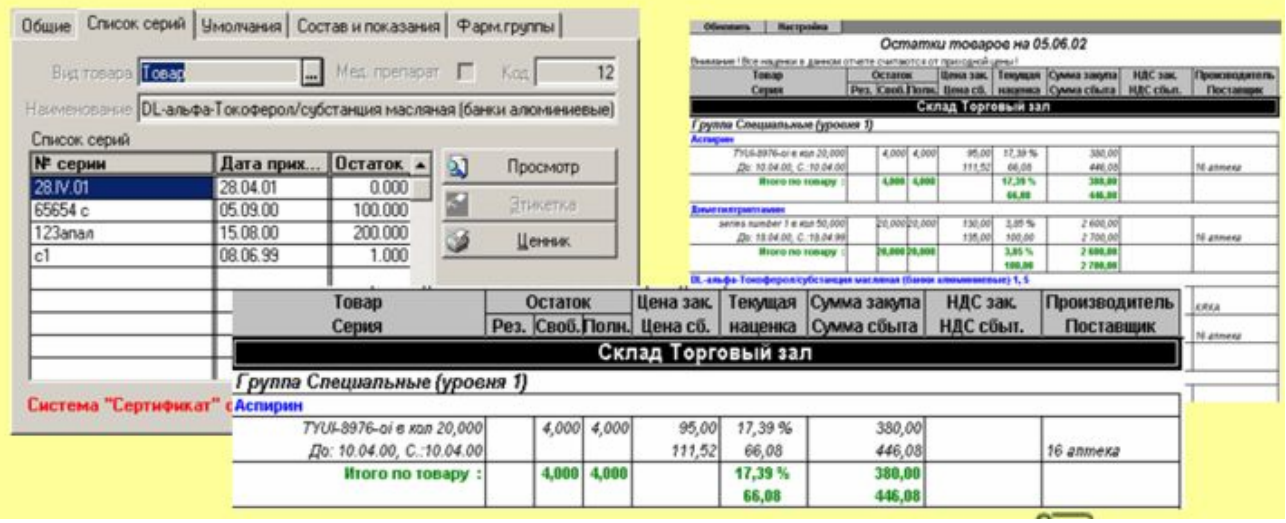


Рисунок 1.2 – Вигляд програмного продукту «АСТОР ВЦ» - «Аптека v 4.5»

Конфігурація «Аптека v.4.5» для 1С:Предприятия 7.7 призначена для автоматизації аптек і підприємств, що займаються роздрібною торгівлею медикаментами і мереж таких підприємств, з можливістю роботи в програмі з різним торговим устаткуванням касовими апаратами, сканерами штрих-кодів, принтерами штрих-кодів. У програмі передбачений облік оптової торгівлі підприємства.

Особливістю даної конфігурації є так званий «партійний» облік товарів на складах. Причому серією товару є товар, що прийшов в один прихід, що володіє однією закупівельною ціною, Даний програмний продукт може застосовуватися для автоматизації наступних торгових підприємств:

1. Аптеки (автоматизовані точки роздрібної торгівлі). Для таких підприємств в системі передбачені механізми обліку, документообігу і

звітності, властиві роздрібній торгівлі медикаментами. Також, в програму інтегровані механізми роботи з широким спектром торгового устаткування і касовими апаратами, що значно знижує трудовитрати при роботі підприємства і отриманні даних для системи обліку.

2. Для мережі аптек з центральним складом (неавтоматизовані точки роздрібної торгівлі) облік оптової торгівлі, система повністю готова для таких організацій. У програмі присутні необхідні механізми обліку оптових операцій і автоматизації робочих місць і обліку товарів на центральному складі і офісі.

3. Оптові організації, які планують перейти до роздрібної торгівлі, і яким необхідно, щоб конфігурація мала механізми обліку роздрібної торгівлі і можливість роботи з ККМ і іншим торговим устаткуванням для швидкого і безболісного переходу до обліку роздрібних продажів.

1.4.2 АСУ "М-Аптека плюс"

В процесі роботи Користувачі здійснюють введення і редагування документів в системі, проведення складських операцій (залежно від прав доступу), введення даних про виконавця, початок і закінчення операції. Існує можливість частково змінювати логістичну схему програми, конфігурації робочих місць і зон на складі за допомогою опису прав доступу по групах користувачів.

Додаткові установки програми включають настройки інтерфейсу, настройки для роботи із списками документів, загальні настройки для роботи на складі і настройки місць зберігання.

АСУ «М-Аптека плюс» розроблена для оптових аптечних складів, а так само – для аптечних мереж, що мають власні централізовані точки приймання товару і розподілу його по підлеглих підрозділах. Система дозволяє повністю автоматизувати процес обліку руху товару на складі: прихід товару на склад, облік товару по місцях зберігання і відвантаження товару одержувачеві (для аптечних мереж - передача в точку роздрібної торгівлі). Надається можливість

включення різних механізмів ціноутворення і ведення операцій на підприємстві будь-якої форми власності.

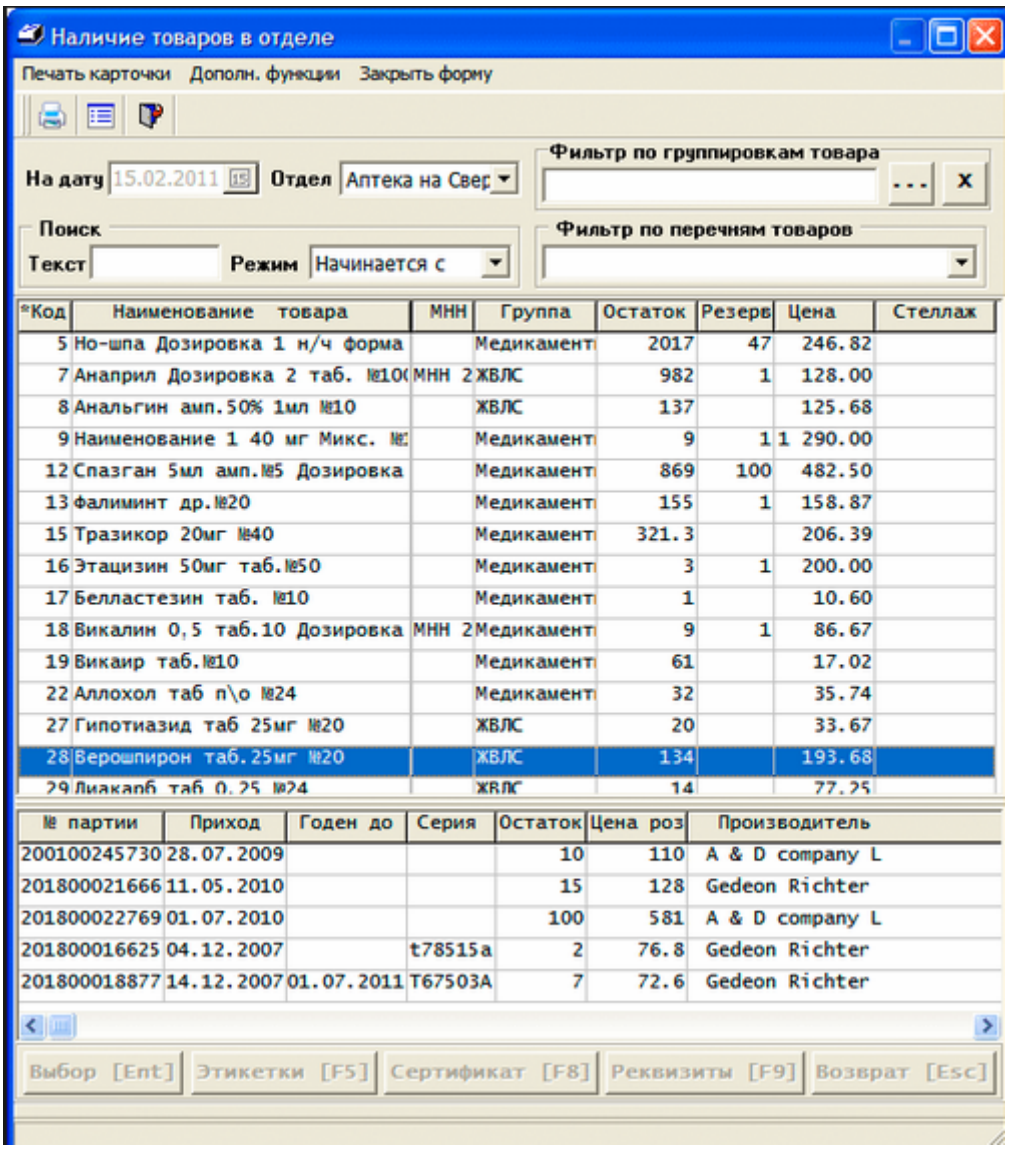


Рисунок 1.3 – Вигляд вікна АСУ «М-Аптека плюс»

Функціональні можливості системи:

- прийом товару по заводському штрих-коду;
- введення документів уручну або прийом електронної накладної від постачальника;
- можливість роботи як по заводському, так і по внутрішньому штрих-коду;

- ведення контролю якості товару, що поступив (перевірка серій, термінів придатності, сертифікатів);
- оформлення документів на проведення аналізу в лабораторії сертифікації;
- розміщення товару по місцях зберігання;
- облік товарів, що знаходяться на складі, зокрема – товару, що вимагає специфічного місця зберігання (забракований товар, товар, що вимагає певних температурних режимів);
- розцінка товару;
- збірка товару для продажу (передачі в роздрібні підрозділи);
- відвантаження товару покупцеві;
- ведення обліку оплати за відвантажений товар;
- проведення інвентаризації наявності товару по місцях зберігання на складі. Існує можливість часткової інвентаризації по місцях зберігання, номенклатурі і спец. групам товару, що дозволяє не зупиняти роботу складу на час її проведення;
- складання форм звітності і інших документів за наслідками операцій приймання-відвантаження товару.

Конфігурація «Оптовий склад» дозволяє:

- отримувати оперативну інформацію про наявність прибуткових і витратних документів, що знаходяться в обробці на різних стадіях операцій з товаром;
- отримувати інформацію про тривалість виконання операцій на складі;
- отримувати інформацію про кількість операцій по виконавцях;
- проглядати історію руху товару по місцях зберігання;
- проглядати залишки товару по конкретних місцях зберігання;
- переглядати і роздруковувати статус оплати і історію платежів по всіх документах;

- оптимізувати роботу розкладальників і складальників товару за рахунок обліку наявності товару по місцях зберігання і підтримки маршрутів розміщення.

1.5 Аналіз мобільних додатків для пошуку медикаментів

1.5.1 Опис програми “Geo Apteka”Geo Apteka””

У додатку представлена інформація про наявність і ціни на ліки в більш ніж 1500 аптек в Києві, Львові, Дніпропетровську, Харкові, Одесі, Миколаєві та ін. Містах України.

Сервіс GeoApteka” пропонує пошук препаратів в найближчих до вас аптеках, підбір ціни і кількості. Також доступні телефони і адреси аптек, маршрути руху.

Основний функціонал додатку “Geo Apteka”Geo Apteka”” включає:

– пошук препаратів у найбільших містах України;

– перегляд на карті трьох найближчих аптек;

– ерегляд адреси аптеки та можливість зателефонувати у відділення;

– прокладання маршруту до вибраної аптеки;

– перегляд інструкції до вибраного препарату;

– налаштування критеріїв пошуку по дозуванню, упаковці та формі випуску.

На рис. 1.4 наведено приклад графічного інтерфейсу користувача. Щоб почати пошук аптек у верхній частині екрану потрібно ввести назву. Після цього буде відкрита карта з найближчими аптеками. Можливо переглянути знайдені результати в списку де показано назву та адресу аптеки, ціну та кількість препарату.

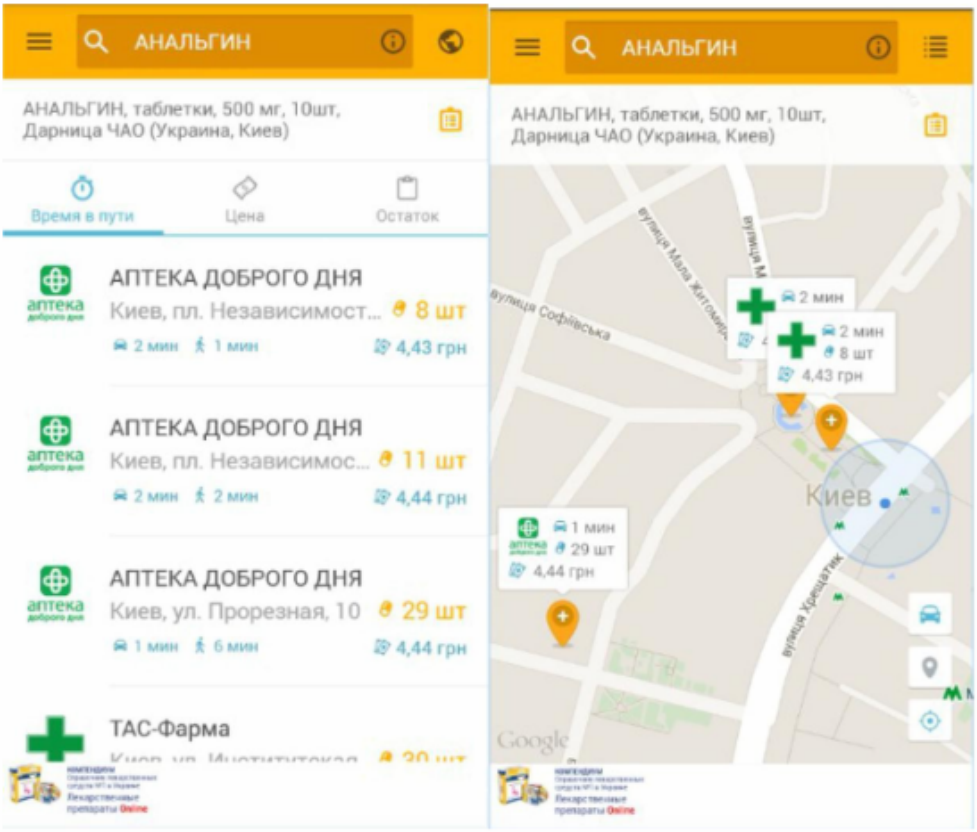


Рисунок 1.4 - Приклад інтерфейсу додатку – Geo Apteka

1.5.2 Опис програми "Geo Apteka"Аптеки України"

В додатку представлено інформаційний опис (форма випуску) ліків, і дані про їх наявність та середню ціну в аптеках мережі "Біокон" (Київ), "DS", "Аптека нашого міста" (Львів), "Конекс" (Вінниця), "Медпрепарати" (Миколаїв, Херсон).

Програма дозволяє:

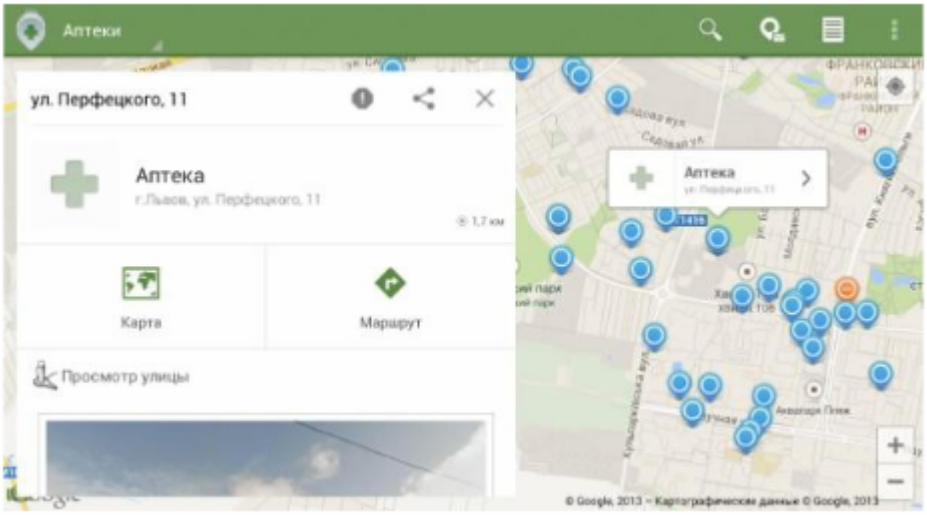
– переглянути список аптек, розташованих біля вашого місце-знаходження (в діапазоні від 1 до 10 кілометрів);

– прокласти зручний маршрут до обраної аптеці (використовуючи карти);

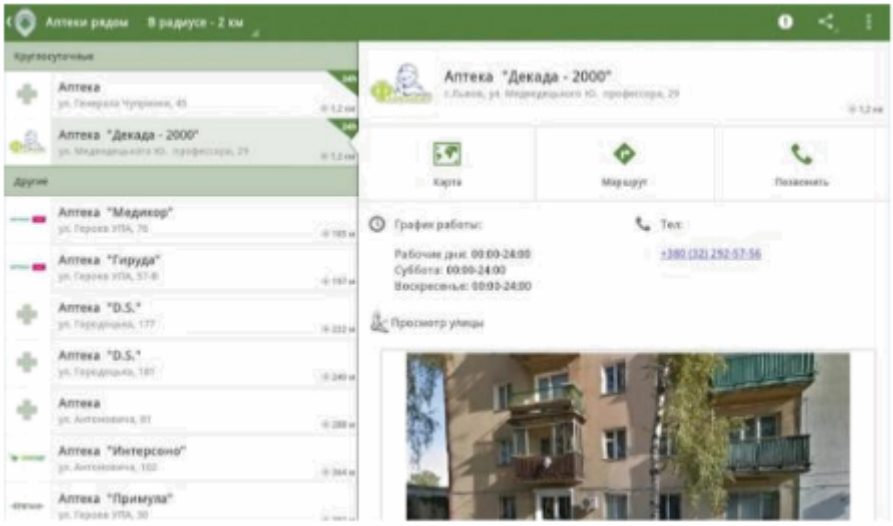
– переглянути місцезнаходження аптеки на карті;

– зателефонувати в аптеку за вказаним номером;

– дізнатися про наявність лікарських засобів та їх середньої вартості в аптеках міст Києва, Львова, Вінниці, Миколаєва, Херсона.



а)



б)

Рисунок 1.5 – Приклад інтерфейсу додатку “Geo Apteka”Аптеки України”
а) пошук на карті б) результат пошуку

На рис. 1.5 а,б наведено приклад графічного інтерфейсу користувача. У верхній частині екранну вводиться назва препарату та задається радіус пошуку аптек. В головному вікні програми можна переглянути результати пошуку на карті та у вигляді списку з детальною інформацією про аптеку. До вибраної аптеки можна прокласти маршрут, зателефонувати, переглянути графік роботи та адресу.

1.5.3 Опис програми “Geo Apteka”Аптечний пункт”

Програма призначена для пошуку лікарських препаратів в аптечних пунктах. Як і в інших додатках можна переглянути результати пошуку на карті та в списку. Є можливість пошуку аналогів якщо бажаних ліків не знайдено.

Також в програмі є функціонал прокладання маршруту до вибраної аптеки транспортом або пішки. Зручною є можливість зберігати інструкції препаратів для подальшого їх перегляду при необхідності (рис. 1.6).

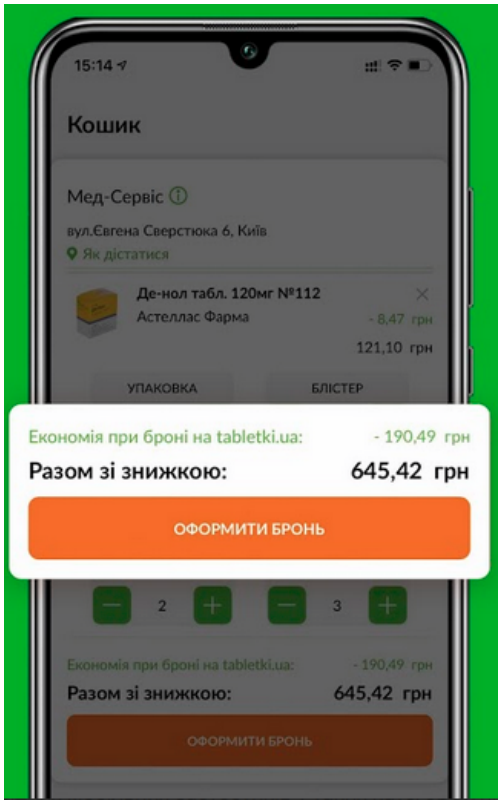


Рисунок 1.6 – Приклад інтерфейсу додатку “Tabletki”

Програма “Tabletki” дозволяє:

– переглянути актуальні ціни на ліки в аптеках;

– порівняння ціни на ліки;

– знайти мінімальні ціни в найближчих аптеках;

– довідник ліків;

– переглянути інструкції до ліків (спосіб застосування, дозування);

– провести пошук аналогів до вибраного препарату;

– можливість зберігати інструкції.

При пошуку препаратів відкривається екран перегляду карти. При кліку на певну аптеку можна переглянути інформацію по препаратах, зателефонувати в аптеку, побудувати маршрут за адресою. На головному екрані доступні меню аналогів, де можна знайти препарати тієї ж фармакологічної групи або відкрити інструкцію препарату.

Це офіційний застосунок сайту Tabletki.ua, і ним можна користуватись цілком безкоштовно і без реєстрації. Це також один з провідних мобільних застосунків у таких категоріях як медицина та фармацевтика, онлайн-аптеки та здоров'я.

В додатку представлені крупні мережі аптек України та онлайн-аптеки: Аптека Низьких Цін (АНЦ), Аптека Оптових Цін, Фармація, Аптека Доброго Дня (АДД), Аптека 911, Подорожник, Бажаємо здоров’я, Аптека Копійка, Аптека 36.6, Аптека БАМ, Соціальна аптека, Мед-Сервіс, Аптекарь, Аптека TAS, Аптека D.S та багато інших.

Висновки до розділу

В даному розділі коротко охарактеризовано об'єкт дослідження даного дипломної роботи – систему автоматизації прийому і обробки замовлень від клієнтів аптечного складу. Визначено основні характеристики існуючої системи та перспективи її розвитку. Також описано аналоги даного програмного комплексу. Визначно основні вимоги до системи, що проектується та передбачено функції, які вона повинна виконувати.

2 РЕАЛІЗАЦІЯ МЕТОДІВ ТА ЗАСОБІВ ПРОЕКТУВАННЯ ПРОЦЕСУ АВТОМАТИЗАЦІЇ ФАРМАЦЕВТИЧНИХ СЕРВІСІВ АПТЕЧНОГО СКЛАДУ

2.1 Розробка структури моделі

Розробка структури програмного комплексу для автоматизації прийому заявок від клієнтів оптового аптечного складу є дуже важливим завданням в процесі побудови самого програмного комплексу.

Основною ціллю є визначення основних вимог до даних, методів і видів доступу до них, їх отримання і передача.

Програмний комплекс, що створюється в даній дипломній роботі є частиною складного комплексу автоматизації аптечного складу, що включає: основну програму (програмний комплекс «АФАРМ») для обробки даних і їх передачі в центральне відділення, модуль для передачі даних з бази даних основної програми в СУБД Microsoft SQL або іншу СУБД за вибором клієнта

Структура передачі даних між компонентами програмного забезпечення на оптовому складі зображена на рисунку 2.1.



Рисунок 2.1 – Структура передачі даних між компонентами програмного забезпечення на оптовому складі

Відповідно структуруються дані що отримуються і передаються програмного комплексу (рис. 2.2.)

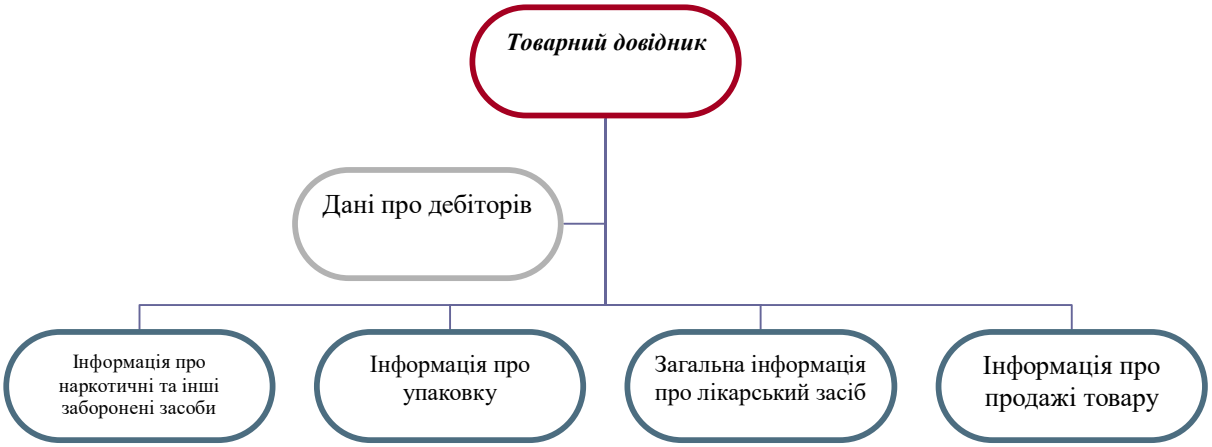


Рисунок 2.2 – Структура даних, що отримуються програмним комплексом

Відштовхуючись від даної структури можна починати власне створення основних компонентів (бази даних і програмного додатку для доступу до бази), їх проектування і вимог до них.

2.1.1 Технологія Клієнт-сервер

Основна думка на якій ґрунтується дана технологія, - мінімізувати об'єм даних, що передаються по мережі, оскільки основні втрати часу і збої відбуваються саме через недостатньо високу пропускну здатність мережі.

Дуже багато СУБД розділяють свою роботу на два рівні по системі "Клієнт-сервер". З погляду виконання програма розділена на 2 частини - клієнтську і серверну. На клієнтській частині (комп'ютері) відбувається контакт із зовнішнім світом. На комп'ютері-сервері розташовані загальні для всіх клієнтів дані і працює спеціальна програма - сервер баз даних, що оптимізує виконання запитів клієнтів.

Отже, дворівнева система "Клієнт-сервер" це:

- Клієнт - програма обробки, вона ж призначена для користувача, вона ж прикладна програма. Займається зазвичай інтерфейсом з користувачем, а всю фактичну роботу з базою даних покладає на плечі сервера баз даних.

- сервер Бази Даних - базис (database engine), він же ядро бази даних. Окрема програма, що виконується як окремий процес. Передає вибрану з бази інформацію по міжпроцесному каналу клієнтові. Саме він, і лише він фактично працює з даними, займається їх розміщенням на диску.

Безперечною перевагою технології є наближеність даних до процесів обчислення. Практично, всі розрахунки виконуються на сервері, що збільшує швидкодію в десятки і сотні разів.

Технологія клієнт-сервер на відміну від файл-серверної дає користувачеві велику безпеку, стійкість, узгодженість, масштабованість, підвищену конфіденційність і надійність обробки і зберігання інформації. Розвиток систем з архітектурою клієнт-сервер неабиякою мірою зобов'язано перевіреному факту: підключення до недорогих серверів недорогих ПК дозволяє отримати оптимальне співвідношення ціни і продуктивності.

В більшості випадків програма обробки (клієнтська частина) розташована на одному комп'ютері, а сама база даних зберігається на іншому. Тут розділення виглядає абсолютно природним: Програма - клієнт (точніше, та її частина, яка відповідає за інтерфейс з користувачем), передає по мережі запити на обробку самих даних на інший комп'ютер, а там БД-сервер їх прочитує, виконує потрібне, і по мережі повертає готові відповіді клієнтові. При цьому по мережі передається тільки корисна інформація.

Інше міркування: постійно йде робота по вдосконаленню самого методу зберігання і обробки інформації, і якщо його реалізація (тобто БД-сервер) змінилася, то не потрібно буде перекомпілювати з новими бібліотеками всі розроблені програми, а досить буде інсталювати новий БД-сервер замість старого і перевести бази даних у формат нового сервера (застосувавши утиліту, що для цього додається до нього). Природно, все це можна виконати, якщо

новий сервер дотримується тих же правил обміну між ним і призначеною для користувача програмою, що і старий, що, втім, напевно має місце.

Використовуючи безліч невеликих комп'ютерів, розробники систем клієнт-сервер можуть емулювати обчислювальну потужність великих ЕОМ, розподіляючи прикладне завдання по різних мікрокомп'ютерах і серверах. Кожен з них бере на себе свою частину обчислювального навантаження, використовуючи інформацію спільно з іншими процесорами мережі. Суть ідеї в тому, щоб підвищити потужність системи, не нарощуючи продуктивність одного комп'ютера, а підсумовуючи засоби багато.

Швидкодія - основний чинник доцільності розробки систем для архітектури клієнт-сервер . Застосування засобів швидкої розробки програм (Rapid Application Development - RAD), таких, як Delphi компанії Borland, дозволяє розробникам "штампувати" прикладні системи для архітектури клієнт-сервер в рекорд короткі терміни. Технологія серверів баз даних також простіша у використанні і поєднується в одних системах із засобами RAD. Таким чином, за допомогою цієї швидкісної і практично захищеної платформи розробки скорочується час, необхідний для підготовки і передачі прикладної програми користувачеві.

Архітектура клієнт-сервер надає розробникам ПО виняткову свободу вибору і узгодження різних типів компонентів для клієнта, сервера і всіх проміжних ланок.

Найбільш проста форма архітектури клієнт-сервер, яка розглядається в контексті даної дипломної роботи - це розділення обчислювального навантаження між двома окремими процесами: клієнтом і сервером. Хоч і клієнт, і сервер можуть знаходитися на одному і тому ж комп'ютері, більшість систем цієї архітектури запускають клієнтський процес на одному комп'ютері, а процес-сервер на іншому, використовуючи для обміну інформацією мережеві зв'язки. У цій моделі один процес може працювати незалежно від іншого, виконувати певні завдання і розділяти обчислювальне навантаження.

Зазвичай клієнтом служить настільний ПК, що виконує програмне забезпечення кінцевого користувача. програмне забезпечення (ПЗ) кінцевого користувача (front-end software) - це будь-яка прикладна програма або пакет, здатні направляти запити по мережі серверу і обробляти отримувану у відповідь інформацію. Сервер, у свою чергу, отримує запити і робить дії від імені клієнта.

ПК, що працює під управлінням Windows XP і клієнт-сервер Delphi, що виконує програму, наприклад, може представити на розгляд запит серверу баз даних (скажімо, Microsoft SQL 2000, запущеній на сервері Windows 2000). Зазвичай клієнт посилає запити базі даних у вигляді пропозицій на мові структурованих запитів (SQL), використовуючи зрозумілий серверу бази даних діалект.

2.2 Методи та підходи до проблеми оптимізації баз даних

У більшості випадків під оптимізацією бази даних розуміють оптимізацію роботи з базою даних, оскільки саме це найбільш є важливим для кінцевого програмного продукту, що виконує обробку даних. В оптимізації роботи з базою даних виділяють три напрями:

- оптимізація файлових серверів та апаратних засобів,
- оптимізація запитів до бази даних,
- оптимізація структури бази даних.

Оптимізація роботи файлових серверів має найменший вплив на оптимізацію бази даних. Звичайно, під оптимізацією серверів розуміють удосконалення та налагодження апаратних і програмних засобів та операційної системи сервера. Оптимізація запитів, напроти, більше відноситься до програмного забезпечення обробки інформації. За допомогою запитів програмний продукт з бази даних отримує необхідні дані. Оптимізація запитів призводить до підвищення швидкості взаємодії між програмним продуктом та

базою даних і зменшує збитковість переданих через лінії зв’язку даних. Ці два напрями є важливими в оптимізації роботи з базами даних. Під оптимізацією бази даних, перш за все, мається на увазі оптимізація структури бази даних, організація інформації в базі даних на логічному рівні. На сьогоднішній день найбільш розповсюдженими є реляційні бази даних, розроблені на основі реляційної моделі даних, створеної доктором Е. Коддом. Теорія реляційної моделі даних описує математичну й логічну модель щодо організації та роботи з інформацією, що зберігається в базах даних.

Кожна наступна нормальна форма вимагає виконання умов попередньої нормальної форми. Перша нормальна форма вимагає від організації таблиці, щоб кожен атрибут таблиці був атомарним, тобто атрибут має складатися з єдиного значення і не може містити в собі переліку значень. Таблиця знаходиться в другій нормальній формі, якщо вона є в першій нормальній формі та всі атрибути таблиці знаходяться у повній залежності від ключового атрибута, а не тільки від деякої його частини. До третьої нормальної форми відносяться такі таблиці другої нормальної форми, в яких неключові атрибути залежать тільки від ключового атрибута. Існує також модифікація третьої нормальної форми, яка називається нормальною формою Бойса-Кодда. Таблиця знаходиться в нормальній формі Бойса-Кодда, якщо вона знаходиться третій нормальній формі та ключовий атрибут не має функціональних залежностей від неключових атрибутів. Таблиця знаходиться в четвертій нормальній формі, якщо вона знаходиться в нормальній формі Бойса-Кодда і всі многозначні залежності є функціональними. Таблиця четвертої нормальної форми знаходиться в п’ятій нормальній формі, якщо у своїй структурі містить обмеження на допустимі комбінації атрибутів. П’ята нормальна форма рідко використовується на практиці і є предметом наукових досліджень. Таблиця знаходиться в доменно-ключовій нормальній формі, якщо забезпечується умова цілісності даних при будь-яких змінах в атрибутах таблиці. Таблиця знаходиться в шостій нормальній формі, якщо

виконуються умови п’ятої нормальної форми та усі залежності є функціональними. Інколи шосту нормальну форму асоціюють з доменно-ключовою нормальною формою .

Описані вище форми нормалізації дозволяють знизити вірогідність появи аномалій даних та збитковість даних, впровадити в саму структуру бази даних реалізацію методів підтримки цілісності даних, зробити базу даних зрозумілою для роботи як людині, так і програмному забезпеченню. Але даний підхід має один значний недолік усіх стандартизованих методів: специфіка конкретних задач не враховується, в результаті чого залишається місце для вдосконалень.

Одними з найбільших баз даних, що існують нині, є бази даних (БД) статистичної інформації. Вони широко використовуються в дослідницькій діяльності, державних установах, бізнесі та багатьох інших сферах. Сучасні системи прогнозування та аналізу також потребують великих об’ємів статистичних даних. На державному рівні навіть створюються архівні бази даних статистичної інформації для накопичення та можливості подальшої обробки.

Характерними рисами цих баз даних є великі об’єми інформації, що носять архівний характер, тобто майже не змінюються:

- нестандартизованість структур таблиць, оскільки найчастіше нова статистична інформація організовується в таблиці, прості для відображення, але складні для їх комп’ютерної обробки;

- низький рівень зв’язності даних між таблицями, викликаний тим, що таблиці статистики рідко пов’язані між собою, а усі близькі дані намагаються розмістити в одній таблиці.

Такі особливості баз даних статистичної інформації породжують ряд специфічних проблем: база даних складається з великої кількості таблиць. При роботі з базою даних програмне забезпечення має знати структуру конкретної таблиці, що при відсутності стандартизації структур викликає складності та

ріст збитковості даних, що передаються по лініях зв'язку; приведення таблиць до нормальних форм реляційної моделі призводить до розбиття однієї логічної таблиці на ряд взаємопов'язаних таблиць бази даних. Велика кількість таблиць бази даних призводить до збільшення часу пошуку, ускладнення запитів до бази даних. Операторам стає складніше відслідковувати й працювати з такою базою даних. Розробка програмного забезпечення теж стикається зі специфічними проблемами. Реалізація інтерфейсу вигляду даних на логічному рівні між системою управління базою даних та програмним рішенням стає задачею розробника прикладного програмного забезпечення, що, у свою чергу, нехтує принципом незалежності реалізації бази даних від реалізації програмного забезпечення, яке працює з базою даних. Загальна складність реалізації як на стороні бази даних, так і на стороні програмного забезпечення підвищують вірогідність появи помилок при роботі з такою системою.

На фоні цих проблем стає очевидною необхідність якісно іншого підходу до розробки та оптимізації баз даних статистичної інформації. Потужний механізм нормалізації реляційних таблиць дає ряд важливих переваг, але стандартизований підхід не враховує специфіки баз даних обліку статистичної інформації і не звільняє від ряду специфічних для даних баз даних недоліків.

Під логічним рівнем оптимізації бази даних маємо на увазі оптимізацію структури бази даних та виділення логічних одиниць (таблиць) у базі даних. Оскільки база даних статистичної інформації складається з набору таблиць статистичних даних, то вони і є логічною одиницею такої бази даних Першим кроком в оптимізації бази даних статистичної інформації є нормалізація вихідних таблиць статистичних даних та службових таблиць. Це дозволяє зменшити збитковість даних та покращити показники однозначності й цілісності інформації. Для цього ми використовуємо другу та третю нормальні форми або нормальну форму Бойса-Кодда. Наступним кроком є створення універсального інтерфейсу для розробників програмного забезпечення, що

дозволить звільнитися від необхідності програмному рішенню знати структуру бази даних.

Для реалізації цього кроку існують два підходи:

- стандартизація структури бази даних та таблиць бази даних;
- надання програмному забезпеченню даних про структуру таблиці бази даних разом із самою таблицею.

Для баз даних обліку статистичних даних пропонується використовувати комбінований підхід, а саме перехід від збереження в базі даних таблиць статистичних даних до збереження опису таблиць статистичних даних. Цей підхід надає ряд переваг. Він дозволяє стандартизувати структуру самої бази даних, тобто різні бази даних будуть мати одну структуру, що значно полегшить роботу з розподіленими базами даних та спростить злиття декількох баз даних. Крім того, база даних буде зберігати та надавати програмному забезпеченню інформацію не тільки про кінцеві значення таблиці, а й опис структури самої таблиці статистичних даних, що підвищує гнучкість як бази даних, так і програмного забезпечення. Також зникає необхідність у створенні службових таблиць, оскільки вся інформація опису може знаходитися в одній таблиці бази даних разом з описом структури таблиці статистичних даних.

2.3 Алгоритмізація процесу автоматизації обробки даних для поставок медичних товарів

Основними вимогами при розробці програмного модуля автоматизації обробки даних для поставок медичних товарів є забезпечення можливості використовувати модуль максимально зручно та доступно, а також отримування якомога ширший обсяг даних.

Для розробки програмного модуля проведено аналіз подібних наявних програм для управління роботи аптечними мережами. Базова конфігурація

подібних програм зазвичай включає в себе всі функції, які дозволяють стежити за наявністю препаратів в аптеці. Додаткові надбудови дозволя¬ють заощадити час на монотонну роботу, аналізувати процеси, отримувати рекомендації та конт¬ролювати цільові коефіцієнти.

У світі існує чимало подібних програм, додатків, надбудов та веб-сервісів. В Україні найбільше поширення з них отримали: «Моріон», «Парацельс» та «АНР Аптека».

Наприклад, основні можливості «АНР Аптека»:

- автоматичне надходження товарів у форматі * mmo, * xls;
- автоматична націнка препаратів;
- реалізація товару в роздріб;
- повернення постачальникам;
- інвентаризація запасів;
- зберігання сертифікатів якості у електронному вигляді;
- гнучка дисконтна система;
- автоматичне формування дефектури;
- контроль виконання дефектури;
- друк цінників /етикеток;
- внутрішнє кодування медикаментів;
- переміщення товару між складами / аптеками;
- майстер перерозподілу товару між аптеками та ін. звіти;
- обмін даними між «АНР-Аптека» і «1С: Бухгалтерія».

Програмний модуль автоматизації обробки даних для поставок медичних товарів має включати в себе функціонал управління і моніторингу великої кількості різних моделей даних, таких як:

- користувачі,
- деталі товару,
- замовлення.

Доцільним є розробка додатку з нескладною структурою та відсутністю обов’язкового виходу в мережу Інтернет, що дасть змогу користувачеві зручно і просто працювати з системою і не вимагатиме особливих навичок або знань. Весь функціонал роботи з моделями даних, їх зберігання та обробка, а також процес проведення алгоритмічних розрахунків має виконуватись керуючим модулем. Клієнтська частина цього програмного модуля відповідатиме за зручне відображення даних, швидкість сис¬теми та можливість взаємодії з нею в режимі реального часу.

Для реалізації бази даних програмного модуля автоматизації обробки даних для поставок медичних товарів доцільно використовувати реляційну модель даних. Як відомо, у реляційній моделі даних об’єкти і взаємозв’язки між ними представляються за допомогою таблиць. Взаємозв’язки також подаються як об’єкти. Перевагами реляційної моделі є незалежність від фізичного рівня представлення, зручність і розуміння організації даних користувачами, можливість розширення бази приєднанням нових елементів й записів без зміни при цьому існуючих підсхем та прикладних програм.

Всі технологічні процеси, зокрема й розробка програмного забезпечення є складним процесом, який залежить від вибору структурної моделі майбутнього програмного продукту. Основним завданням програмного модуля автоматизації обробки даних для поставок медичних товарів є забезпечення автоматизованим процесом здійснення замовлень та поставок, а також спрощенням роботи кожної ланки (ролі) робочого процесу:

- адміністратор;
- користувач (наприклад: фармацевт);
- завідувач складом (постачальник).

Надалі розглядатиметься розробка програмного модуля за участі адміністратора та користувача.

Завданнями програми є:

- додавання нових товарів;

- видалення існуючих товарів;
- додавання нових складів;
- видалення існуючих складів;
- додавання нових постачальників;
- видалення існуючих постачальників;
- додавання нових поставок.

Нижче наведена специфікація вимог до системи (в цій специфікації описано вимоги до системи з погляду користувачів):

- адміністратор аптеки може переглядати та видаляти інформацію про товари, склади і постачальників, здійснювати замовлення нових поставок;
- користувач (в особі фармацевта) може переглядати наявну інформацію і здійснювати замовлення нових поставок без права на видалення даних з бази;
- для зручності вибору товару користувачами, система дозволяє переглядати список товарів для подальшого ознайомлення.

Процес поставки товарів проходить через кілька етапів: постачальника, склад, торговельну залу. Аптека взаємодіє з постачальниками від імені адміністратора (або користувача в особі фармацевта), здійснюють постачання ліків, побутової хімії, медичного устаткування й косметичні засоби. Кожний товар має власні: назву, категорію та опис. Постачальник, що забезпечує товарами, взаємодіє з фармацевтом або адміністратором, який, в свою чергу, здійснює замовлення.

Адміністратор має доступ до усіх можливих даних та процесів програмного модуля. Функціонал, що повинен забезпечувати програмний модуль автоматизації обробки даних для поставок медичних товарів для ролі «адміністратор»:

1. Створити, переглянути та редагувати замовлення (поставку);
2. Додати, переглянути та редагувати список товарів;
3. Додати, переглянути та редагувати список складів;

4. Додати, переглянути та редагувати список постачальників.

Отже, весь функціонал, що має забезпечувати програмний модуль автоматизації обробки даних для поставок медичних товарів для ролі адміністратор, описаний в Use Case-діаграмі. UML- діаграма прецедентів програмного модуля показана на рис. 2.3

На рис. 2.4 показана діаграма прецедентів однієї з дій системи керування даними. Діаграми решти дій організовані аналогічним чином.



Рисунок 2.3 - UML-діаграма прецедентів програмного модуля автоматизації обробки даних для поставок медичних товарів

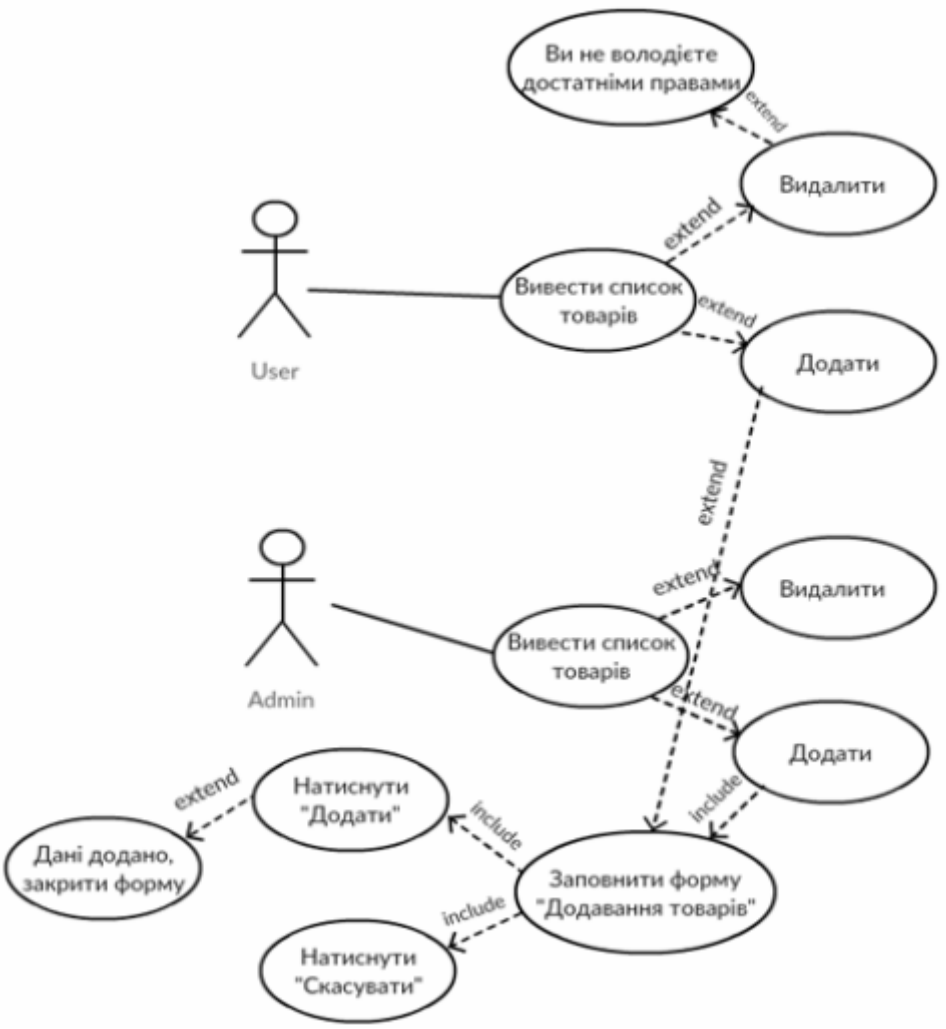


Рисунок 2.4 - Діаграма прецедентів однієї з дій системи керуванням даними

Для того, щоб забезпечити користувачам зручність під час створення або редагування даних чи використання модуля в цілому побудовано UML-діаграми діяльності (Activity Diagram) для кожної дії (процесу), відповідно. Спершу ця програма в разі запуску запитує статус користувача, тобто логін і пароль, що ідентифікують його роль (адміністратор чи користувач), потім підключається до бази даних. В логіці роботи форми входу з даними користувача необхідно також реалізувати перевірку на правильність введених даних та заповнення усіх обов’язкових полів та вивід відповідних повідомлень про помилки в діях.

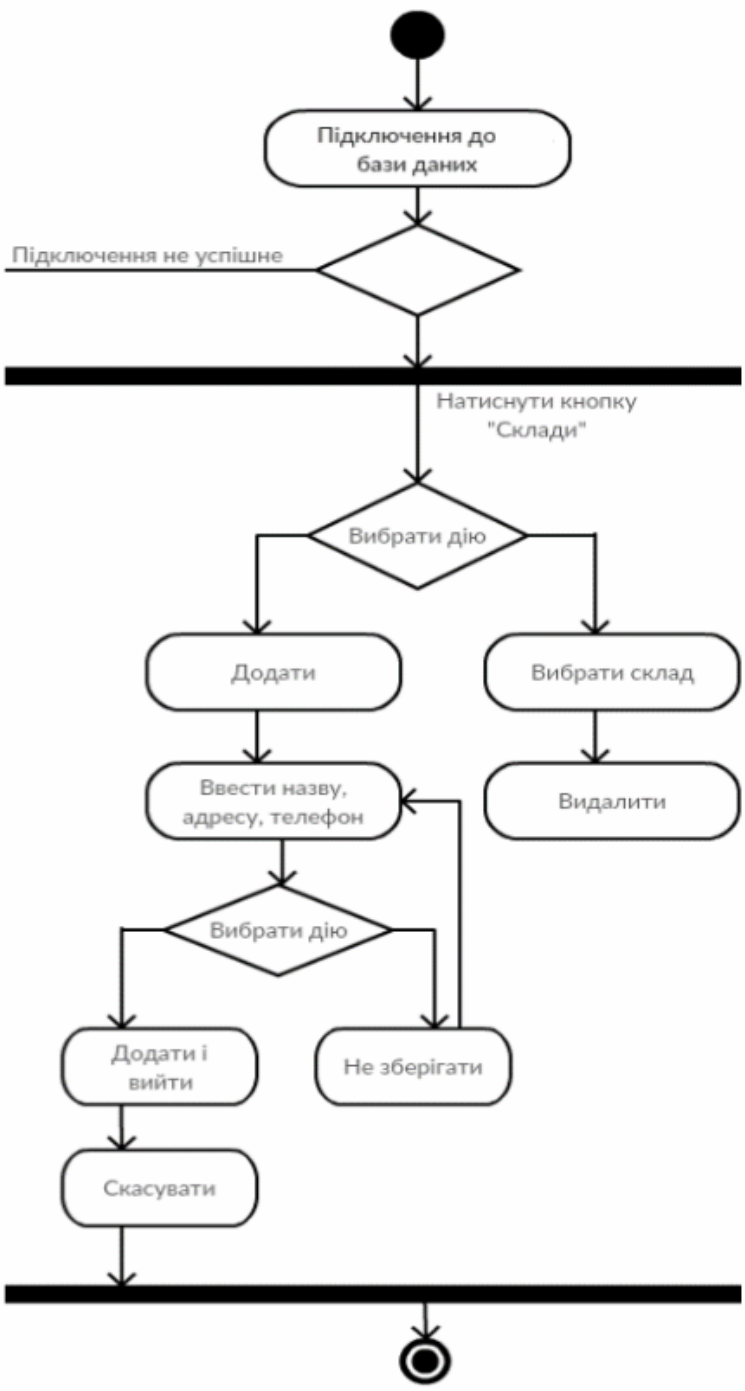


Рисунок 2.5 - Діаграма активності системи керуванням даними

На рис. 2.5 показано діаграму активності си¬стеми керування процесами, пов’язаних з безпосередньою обробкою даних про склади. Варто зазначити, що всі функції видалення, присутні в програмному модулі, будуть доступні лише для адміністратора.

Розробка універсального відношення полягає у виборі інформаційних об’єктів (сутностей), задання найважливіших та суттєвих характеристик бази даних сервісу автоматизації обробки даних для поставок медичних товарів.

Проаналізувавши сформовані запити, в базу даних необхідно включити атрибути, що описують такі сутності:

- користувач (адміністратор або звичайний користувач);
- поставки;
- постачальники;
- товари;
- склади.

Подальшим кроком розробки універсального відношення є введення атрибутів для опису вищезазначених сутностей:

- користувач (id, ім’я користувача; прізвище користувача; роль (адміністратор або користувач; логін, пароль));
- поставки (id, id товару, id складу, дата поставки, id постачальника, ціна, кількість);
- постачальники (id, назва постачальника, адреса постачальника, номер телефону постачальника, статус постачальника);
- товари (id, назва товару, опис товару, категорія товару, статус товару);
- склади (id, назва складу, адреса складу, номер телефону складу, статус складу).

Отже, універсальне відношення буде мати такий вигляд:

R (ім’я користувача; прізвище користувача; роль (адміністратор або користувач); логін, пароль, id товару, id складу, дата поставки, id постачальника, ціна, кількість, назва постачальника, адреса постачальника, номер телефону постачальника, статус постачальника, назва товару, опис товару, категорія товару, статус товару, назва складу, адреса складу, номер телефону складу, статус складу).

Ступінь універсального відношення – 23. Універсальне відношення вміщує в собі всі атрибути, що будуть використовуватись в базі даних.

ER-модель для бази даних для автоматизації обробки даних показана на рис. 2.6.

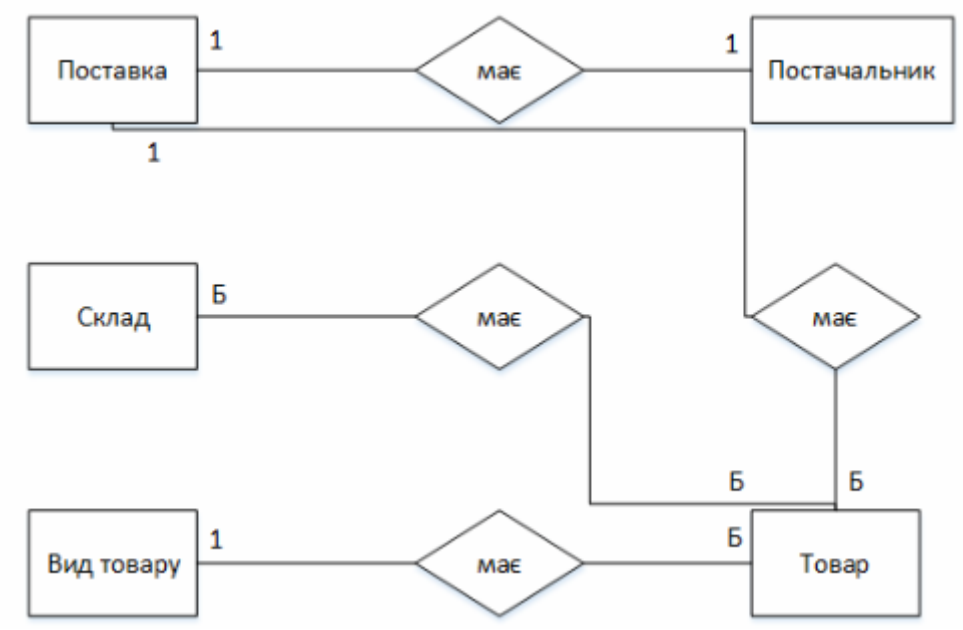


Рисунок. 2.6 - ER-модель предметної області бази даних для автоматизації обробки даних

Для реалізації системи керуванням даними побудовано UML-діаграму класів (Class Diagram) (рис. 2.7). Кожний клас має свої змінні та методи, оскільки використані методи є стандартними, то на діаграмі їх позначати не обов’язково.

База даних для автоматизації поставки товарів до аптеки орієнтована не на пересічних користувачів. Це можуть бути лише фармацевти, провізори та інші особи, пов’язані з фармацевтичною сферою. Тобто базою може користуватись будь-яка особа, але особливим обмеженням є належність до конкретної сфери.

За допомогою цієї бази даних користувачі зможуть ефективніше знайомитись з даними про товар, витрачаючи менше зусиль, оскільки необхідна

інформація буде подаватись точно і в зручному вигляді, а час, необхідний на її пошук, скоротиться в десятки разів, якщо порівнювати з пошуком необхідних документів у книгах, журналах та мережі Інтернет. Інформація, що міститься в базі, охоплює всі аспекти цієї предметної області, а саме:

- інформація про товар,
- дані про склади та постачальників,
- дані про категорію товару,
- дані про поставки,
- інша інформація.

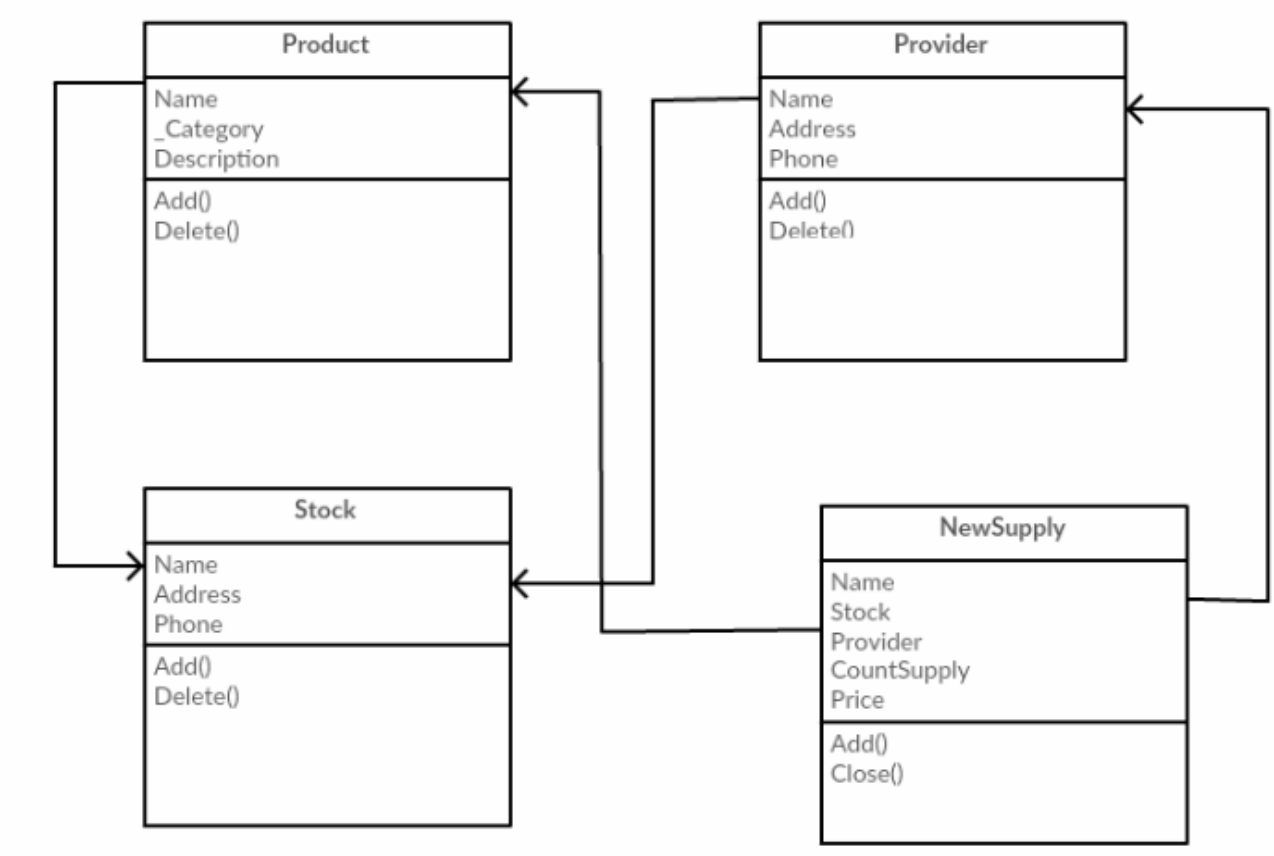


Рисунок 2.7 - Діаграма класів системи керуванням даними

Для інтерактивного використання створеної бази даних буде реалізовано програмне забезпечення. Його використання дасть можливість користування

базою даних особам, що не мають розуміння про структуру бази даних чи фізичне розміщення даних.

Провівши аналіз наявних програм, поєднавши можливості та дані усіх складових, програмного модуля, що розробляється, розроблено алгоритм, за яким має працювати програма для автоматизації поставки товарів до аптеки:

Крок 1. Запускається головна сторінка.

Крок 2. Проводиться вибір дії над базою даних.

Крок 3. Якщо вибрано вивід інформації певного списку, переходимо до Кроку 4.

Крок 4. Виконується виведення необхідної інформації.

Крок 5. Якщо вибрано виконання запиту, переходимо до Кроку 6.

Крок 6. Виконується вибір потрібного 7-го запиту.

Крок 7. Введення критеріїв для виконання запиту.

Крок 8. Виконання операцій реляційної алгебри над відношеннями у відповідності до введених критеріїв.

Крок 9. Виведення результатів виконання запиту.

Крок 10. Повернення до Кроку 2.

Крок 11. Вихід з програми.

Вихідні форми повинні бути зручними для користувачів і надавати максимальну кількість необхідної інформації, водночас не перевантажуючи непотрібними даними.

Проаналізовано аналоги для організації роботи аптечних мереж та виявлено такі недоліки, як великі затрати часу на роботу з програмою, а також складність для сприйняття через ті чи інші аспекти роботи з інтерфейсом та сприйняття загалом. Це дозволило поставити задачу розширення функціональних можливостей медичних поставок та полегшення пошуку інформації про товар, постачальника, склад шляхом розробки програмного модуля автоматизації обробки даних про них.

Висновки до розділу

В даному розділі проведена розробка структури програмного комплексу, усіх його основних частин. Визначено клієнт-серверну технологію побудови моделі програмного комплесу та розроблено модель структури бази даних. Визначено етапи проектування бази даних згідно до встановлених завдань.

Розроблено та обґрунтовано UML-моделі та алгоритми автоматизації процесів для забезпечення ефективного оброблення даних в процесі організації поставки медичних товарів. Результатом стала розробка структури бази даних для оптимального представлення всієї інформації предметної області; клієнтської частини зі зручним та зрозумілим інтерфейсом для забезпечення швидкої взаємодії користувача з системою та її базою даних.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ ДЛЯ РОБОТИ АПТЕЧНОГО СКЛАДУ

3.1 Реалізація сервера бази даних

База даних призначена для зберігання інформації про лікарські засоби та клієнтів аптечного складу. Мається на увазі, що ця інформація може змінятся в перебігу всього періоду продажу чи наявності на складі і може викликатись у будь-який час. База даних поза сумнівом носить характер фактографічної інформаційної системи і повинна видавати однозначні відомості на поставлені запити. Кінцевими користувачами бази даних є працівники (оператори) аптечного складу, які відносяться до категорії користувачів не досвідчених в питаннях ведення, адміністрування баз даних і підтримці їх в актуальному стані. Крім того, існуюча база даних диктує необхідність брати до уваги питання сумісності і конвертації даних існуючої бази даних у формат що розробляється в даній дипломній роботі.

Інфологічна модель бази даних легко відображається в реляційну даталогічну модель, використовуючи описані раніше правила по перекладу. В результаті виходить сім таблиць реляційної бази даних, де кожна суть безпосередньо відбивається в окрему таблицю, атрибути кожної суті стають полями цієї таблиці, а первинні ключі суті стають первинними ключами таблиці.

База даних організована в популярному форматі баз даних MS SQL (може підтримуватись і MS Access).

Розглянемо на рисунку 3.1 загальну змодельовану і реалізовану структуру баз даних, що використовуються в даному програмному комплексі

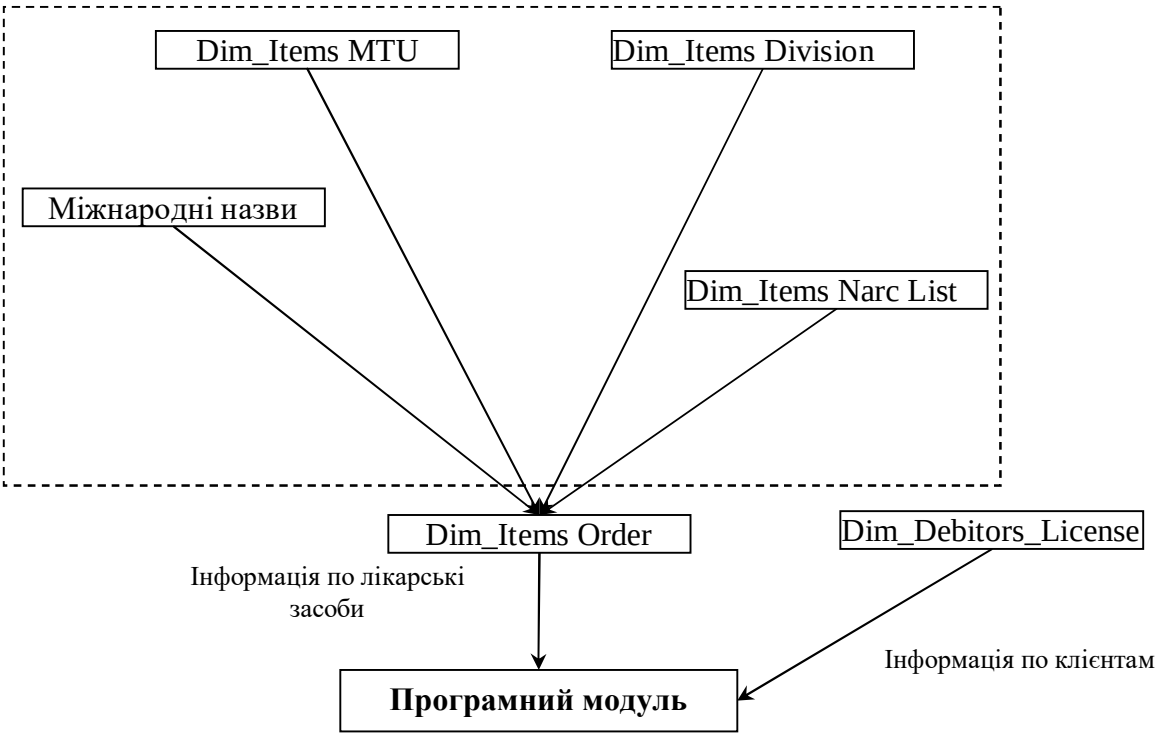


Рисунок 3.1 – Структура бази даних, що використовується в програмному комплексі

Детальніше розглянуто вміст даних таблиць бази даних у таблиці 3.1

Таблиця 3.1 – Інформація про таблиці бази даних і їх вміст

Таблиця бази даних	**Вміст (дані)**
Dim_Items Order	Товарний довідник
Dim_Items Narc List	Наркотичні засоби
Dim_Items MTU	Упаковка (кількість упаковок в ящику)
Dim_Items Division	Дробні упаковки
	Division – на скільки ділиться
	Base – кількість таблеток
Міжнародні назви	Міжнародні назви препаратів і коротка інформація про лікарських засіб
Dim_Debitors_License	Інформація про клієнтів, їх ліцензії та інша додаткова інформація

3.2 Реалізація моделі клієнта програмного сервісу

Оскільки використання баз даних є одним з наріжних каменів, на яких побудовано існування різних організацій, пильну увагу розробників застосувань баз даних викликають інструменти, за допомогою яких такі програмні продукти можна було б створювати. Вимоги, що висуваються до них, в загальному вигляді можна сформулювати як: "швидкість, простота, ефективність, надійність".

Delphi містить повноцінний текстовий редактор типу Brief, призначення клавіш в якому відповідають прийнятим в Windows стандартам, а глибина ієрархії операцій Undo неограниченна. Як це стало вже обов'язковим, реалізовано колірне виділення різних лексичних елементів програми. Процес побудови застосування достатньо простий. Потрібно вибрати форму (у поняття форми входять звичайні, діалогові, батьківські і дочірні вікна MDI), задати її властивості і включити в неї необхідні компоненти (видимі і, якщо знадобиться, що не відображаються): меню, інструментальні панелі, рядок стану і т. п., задати їх властивості і далі написати (за допомогою редактора початкового коду) обробники подій. Object Browser Вікна типу Object Browser стали невід'ємною частиною систем програмування на об'єктно-орієнтованих мовах. Робота з ними стає можливою відразу після того, як ви скомпілювали застосування.

Project Manager - це окреме вікно, де перераховуються модулі і форми, складові проект. При кожному модулі указується маршрут до каталога, в якому знаходиться початковий текст. Жирним шрифтом виділяються змінені, але ще не збережені частини проекту. У верхній частині вікна є набір кнопок: додати, видалити, показати початковий текст, показати форму, задати опції і синхронізувати вміст вікна з текстом файлу проекту, тобто з головною програмою на мові Pascal.

Опції, включаючи режими компіляції, задаються для всього проекту в цілому. В цьому відношенні традиційні make-файли, використовувані в компіляторах мови С, значно гнучкіші.

Visual Component Library (VCL) Багатство палітри об'єктів для побудови призначеного для користувача інтерфейсу - один з ключових чинників при виборі інструменту візуального програмування. При цьому для користувача має значення як число елементів, включених безпосередньо в середу, так і доступність елементів відповідного формату на ринку.

Передбачені спеціальні набори компонент, що відповідають за доступ до даним, і компонент, що відображають дані. Компоненти доступу до даних дозволяють здійснювати з'єднання з БД, проводити вибірку, копіювання даних, і т.п. Компоненти візуалізації даних дозволяють відображати дані виді таблиць, полий, списків. Дані, що відображаються, можуть бути текстового, графічного або довільного формату.

3.2.1 Основний програмний модуль

Наступна UML діаграма показує структуру розроблених класів, їх основні властивості та методи, що використовують. На рисунку 3.2. наведено UML діаграму, що показує і описує структуру класу TMainForm.

-- *фрагмент коду Реалізації модуля клас TMainForm*

```
unit MainFrm;

interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
Menus, ExtCtrls, Grids, ComCtrls, DB, ADODB, VQuery,DateUtils,
StdCtrls, Buttons, DBGrids, DBCtrls, ActnList, ComObj, ExtActns , Mapi,
  IdComponent, IdTCPConnection, IdTCPClient, IdMessageClient, IdSMTP,
  IdBaseComponent, IdMessage, IdIntercept, IdLogBase, IdLogFile, JvMenus,
  JvComponentBase, JvLogFile, JvFormAutoSize;
type
TMainForm = class(TForm)
  StatusBar1: TStatusBar;
  Panel1: TPanel;
  BitBtnNewZakaz: TBitBtn;
  BitBtnSendZakaz: TBitBtn;
  BitBtnExit: TBitBtn;
  Panel3: TPanel;
      Timer1: TTimer;
```

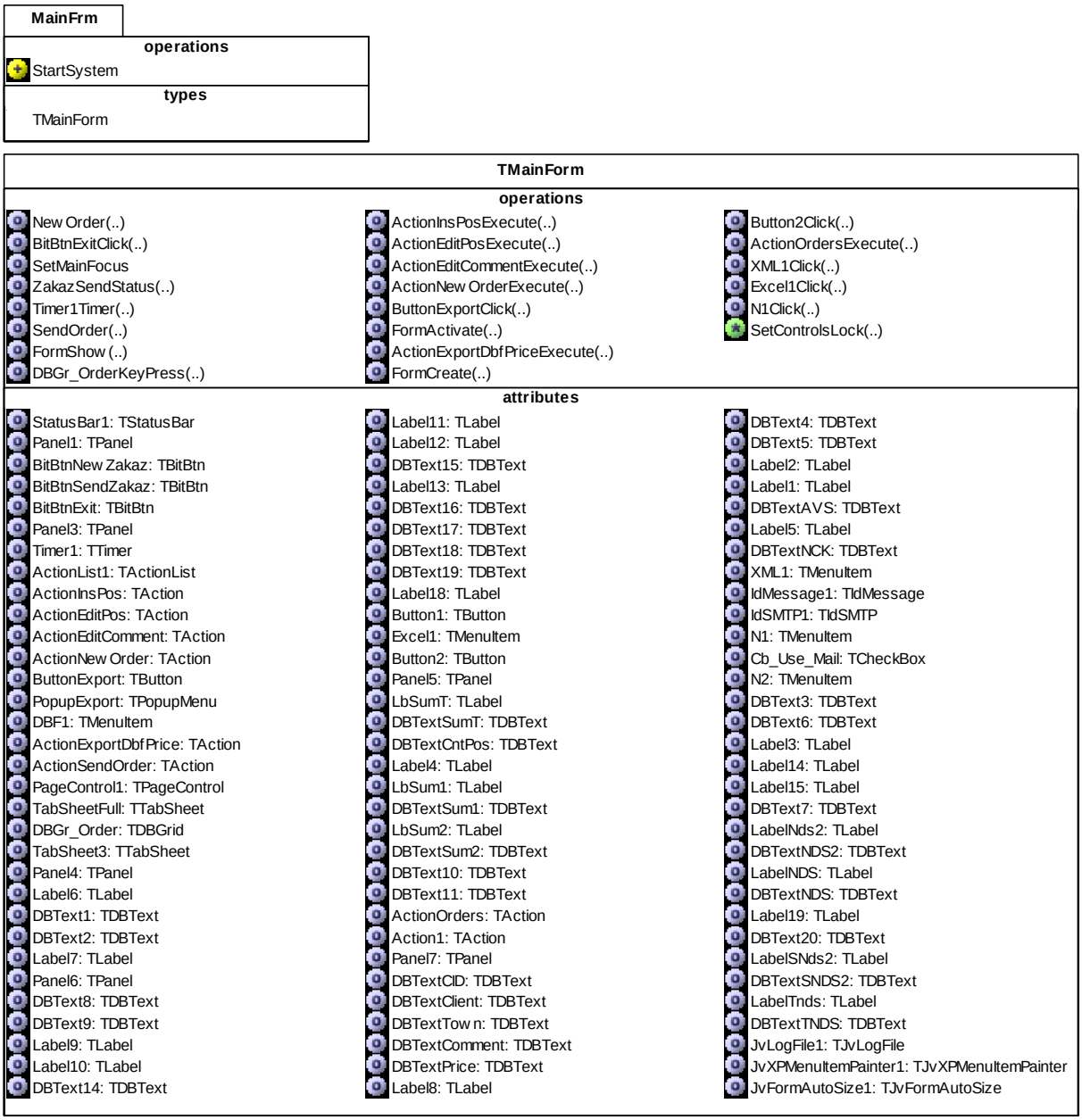


Рисунок 3.2 – UML діаграма основного модуля (клас TMainForm)

```
ActionList1: TActionList;
ActionInsPos: TAction;
ActionEditPos: TAction;
ActionEditComment: TAction;
ActionNewOrder: TAction;
ButtonExport: TButton;
PopupExport: TPopupMenu;
DBF1: TMenuItem;
ActionExportDbfPrice: TAction;
ActionSendOrder: TAction;
PageControl1: TPageControl;
TabSheetFull: TTabSheet;
DBGr_Order: TDBGrid;
TabSheet3: TTabSheet;
```

```
  Panel4: TPanel;
  Label6: TLabel;
  DBText1: TDBText;
  DBText2: TDBText;
  Label7: TLabel;
  Panel6: TPanel;
  DBText8: TDBText;
  DBText9: TDBText;
  Label9: TLabel;
  Label10: TLabel;
  DBText14: TDBText;
      Label11: TLabel;
  Label12: TLabel;
  DBText15: TDBText;
  Label13: TLabel;
  DBText16: TDBText;
  DBText17: TDBText;
  DBText18: TDBText;
  DBText19: TDBText;
  Label18: TLabel;
…
  procedure NewOrder(Sender: TObject);
  procedure BitBtnExitClick(Sender: TObject);
      procedure SetMainFocus;
  procedure Excel1Click(Sender: TObject);
  procedure N1Click(Sender: TObject);
private
  { Private declarations }
  function SelectUmov:boolean;
public
  { Public declarations }

  published
  procedure SetControlsLock(lck: boolean);
end;

procedure StartSystem();
  var
  MainForm: TMainForm;
  implementation

uses PriceFrm, InfoFrm, CommentFrm, UslovFrm,  data_acc,
CountInput, SendStatus, FmxUtils,  IBSQL, LocalDb,IBDatabase, IBCustomDataSet,
IBTable,  VispUnit1, OrderFunc, Variants, Boot, Orders, Shareware,
  ProgressForm;

begin
  // Нове замовлення
  if not(ZakazSend) and (Application.MessageBox('Набране замовлення не
відправлено, створити нове?','Набране Замовлення не
відправлено',MB_YESNO+MB_ICONWARNING+MB_DEFBUTTON2)=IDNO) then
  begin
    exit;
  end;
{Зчитування умов клієнтів з бази}
  with DM_Local.VQ_OrderH do
  begin
    if Active then
      close;
    Open;
    Append;
    DM_Local.VQ_OrderH['Days']:=0;
```

```
    DM_Local.VQ_OrderH['Koef_AVS']:=1;
    DM_Local.VQ_OrderH['Koef_NCK']:=1;

    if not SelectUmov then
    begin
      Cancel;
      last;
      exit;
    end;
ProgresForm:=TProgresForm.Create(self);
ProgresForm.Label1.Caption:='Обробка...';
ProgresForm.Show;
ProgresInc:=True;
ProgresForm.ProgressBar1.Max:=200;
ProgresForm.ProgressBar1.Min:=1;
ProgresForm.Timer1.Enabled:=true;
Application.ProcessMessages;
    FieldByName('BayerID').AsString:=UslovFrm.selbedid;
FieldByName('OrderID').AsString:=IntToStr(Round(Now*10000000)){+IntToStr(data_ac
c.DebCode)};
    FieldByName('BayerName').AsString:=UslovFrm.DebTX;
    FieldByName('BayerTown').AsString:=UslovFrm.DebTown;
    FieldByName('OrderDate').AsDateTime:=Now();
    FieldByName('Sended').AsBoolean:=false;
    Post;
  end;

ProgresForm.ProgressBar1.Position:=30;
Application.ProcessMessages;
  RecalcStat;
ProgresForm.ProgressBar1.Position:=80;
Application.ProcessMessages;
PriceCol:=DM_Local.VQ_OrderH['PriceCol'];
with DataM.DB_Osts do
  begin
    if Active then
      close;
    open;

  end;
ProgresForm.ProgressBar1.Position:=150;
Application.ProcessMessages;
if Not(DataM.DB_Goods.Active)
  then DataM.DB_Goods.Open;
ProgresForm.Timer1.Enabled:=False;
ProgresForm.Close;
ProgresForm.Free;
  SetMainFocus;
  ActionInsPos.Execute
end;
procedure TMainForm.BitBtnExitClick(Sender: TObject);
begin
  Close;
end;
procedure TMainForm.Timer1Timer(Sender: TObject);
begin
  Self.Timer1.Enabled:=false;
  StartSystem;
  Self.Timer1.Free;
end;
procedure TMainForm.SendOrder(Sender: TObject);
Function SaveOrderOptima(DSH,DSL:TVQuery;
```

```
Comment:string;
f2:boolean):integer;
var err:boolean;
user,FinStr,TovLine,headStr,fn,fzn,OrderID:string;
fz:textfile;
i:integer;
cntf:Currency;
at:TIdAttachment;
begin
  SendStatus.SendStatusForm.Memo1.Lines.Add(Comment);
  Application.ProcessMessages;
user:=GetEnvironmentVariable('USERNAME');
if f2 then
  user:='[Ф2] '+user;
//
Randomize;
fn:='Ez'+Inttostr(Round(Random*666666)+111111)+'.txt';
fzn:='c:\'+fn;
assignfile(fz,fzn);
rewrite(fz);
headStr:='0|'+Inttostr(DSH['BayerID'])+'|['+DSH['Comment']+'
'+user+']'{+data_acc.DebTX+'
'+data_acc.DebTown}+'||||||||||||||||||||||'+'['+DSH['Comment']+' '+user+']|||'+
FormatDateTime('yyyymmdd',now)+ '|'+FormatDateTime('yyyymmdd hh:mm',now)+
'|'+FormatDateTime('yyyymmdd',now)+'|||';
writeln(fz,headStr);
DSL.First;
    while not(DSL.eof) do
    begin
try
if f2 then
cntf:=DSL.FieldByName('Count2').AsFloat
else
cntf:=DSL.FieldByName('Count').AsFloat;
cntf:=0;
end;
if cntf>0 then begin
TovLine:='1|'+DSL['TID']+'|KI|1|0|'+cconv(CurrToStrF(cntf,ffFixed,3))+'|'+cconv(
DSL['Price'])+'|';
        writeln(fz,TovLine);
        end;
DSL.Next;
end;

begin
  SendStatus.SendStatusForm:=TSendStatusForm.Create(SELF);
  SendStatus.SendStatusForm.Show;
  SendStatus.SendStatusForm.Memo1.Lines.Clear;
  SendStatus.SendStatusForm.Button1.Visible:=false;
  SaveOrder2(DM_Local.VQ_OrderH,DM_Local.VQ_OrderL,' Ексель 1',false);
  if (OrderType>1) then
    SaveOrder2(DM_Local.VQ_OrderH,DM_Local.VQ_OrderL,' Ексель 2',TRUE);
  ZakazSendStatus(1);
  SendStatus.SendStatusForm.Memo1.Lines.Add('Завершено');
  SendStatus.SendStatusForm.Button1.Visible:=true;
  Application.ProcessMessages;
  Sleep(3000);
  SendStatusForm.Close;
  SendStatusForm.Free;
  SetMainFocus;
end;
end.
```

3.2.2 Модуль роботи з прайсом

Наступна UML діаграма показує структуру класів, що використовуються в модулі боботи з прасом (клас TPriceForm), їх основні властивості та методи, що використовують. На рисунку 3.3. наведено UML діаграму, що показує і описує структуру класу TPriceForm

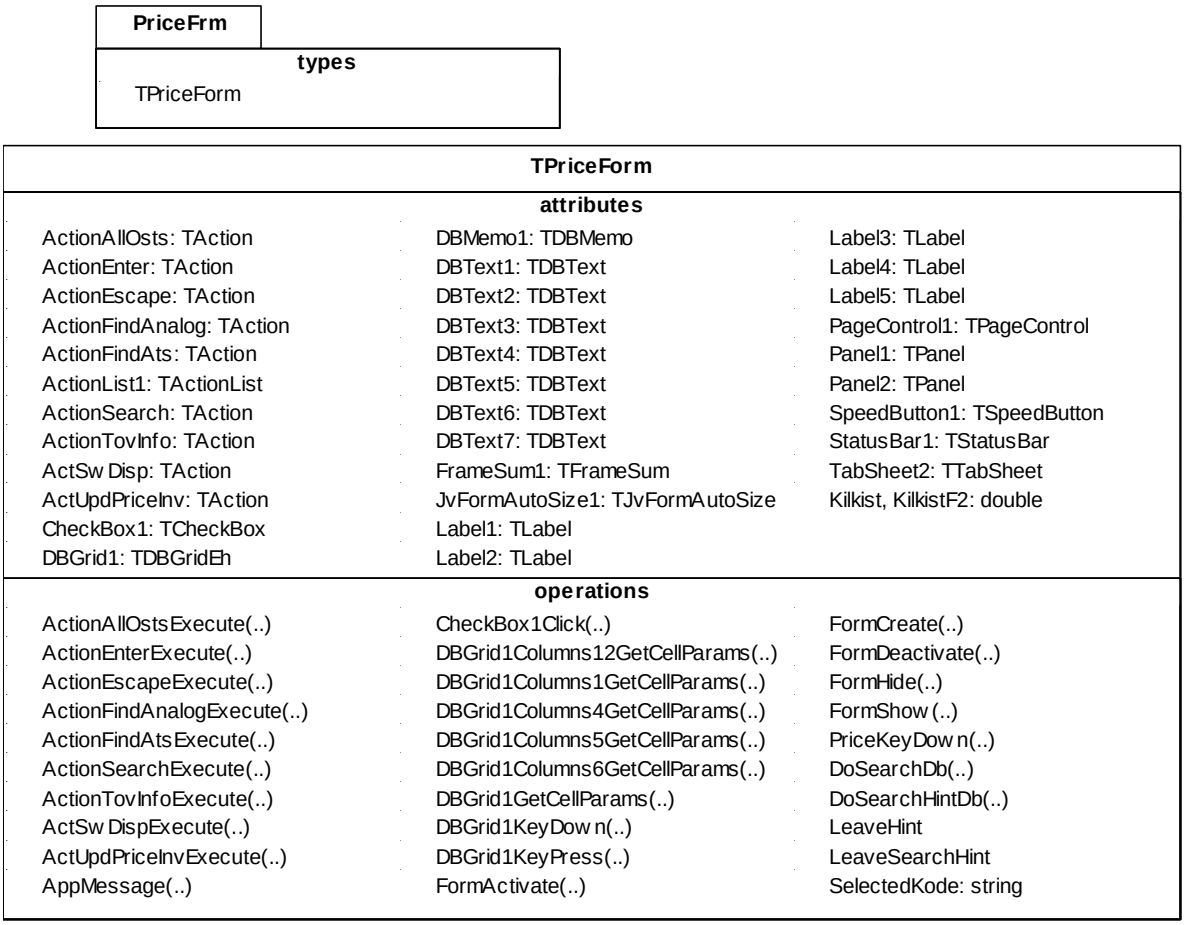


Рисунок 3.3 – UML діаграма класу для роботи з прайсом TPriceForm

Фрагмент реалізації програми

```
unit PriceFrm;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
Grids, Menus, StdCtrls, Buttons, ExtCtrls, ComCtrls, DB, ADODB, VQuery,
DBGrids,DateUtils,DBTables,Math,       VListView,       DBCtrls,       SumFrame,
ActnList,MemTableEh,
  DBGridEh, JvExControls, JvComponent, JvLED, JvComponentBase,
  JvFormAutoSize;

type
TPriceForm = class(TForm)
  StatusBar1: TStatusBar;
```

```
  Panel1: TPanel;
  FrameSum1: TFrameSum;
  ActionList1: TActionList;
  ActionEscape: TAction;
  ActionEnter: TAction;
  PageControl1: TPageControl;
  TabSheet2: TTabSheet;
  ActionSearch: TAction;
  procedure FormShow(Sender: TObject);
  procedure ActionEscapeExecute(Sender: TObject);
  procedure ActionEnterExecute(Sender: TObject);
  procedure ActionSearchExecute(Sender: TObject);
    procedure AppMessage(var Msg: TMsg; var Handled: Boolean);
    procedure DBGrid1Columns6GetCellParams(Sender: TObject;
      EditMode: Boolean; Params: TColCellParamsEh);
    procedure DBGrid1Columns5GetCellParams(Sender: TObject;
      EditMode: Boolean; Params: TColCellParamsEh);
    procedure ActionTovInfoExecute(Sender: TObject);
    procedure ActionAllOstsExecute(Sender: TObject);
    procedure ActionFindAnalogExecute(Sender: TObject);
    procedure ActionFindAtsExecute(Sender: TObject);
    procedure DBGrid1Columns1GetCellParams(Sender: TObject;
      EditMode: Boolean; Params: TColCellParamsEh);
    procedure DBGrid1Columns12GetCellParams(Sender: TObject;
      EditMode: Boolean; Params: TColCellParamsEh);
    procedure DBGrid1GetCellParams(Sender: TObject; Column: TColumnEh;
      AFont: TFont; var Background: TColor; State: TGridDrawState);
    procedure DBGrid1Columns4GetCellParams(Sender: TObject;
      EditMode: Boolean; Params: TColCellParamsEh);
    procedure ActSwDispExecute(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
    procedure ActUpdPriceInvExecute(Sender: TObject);
  Procedure DoSearchHintDb(s:string);
  procedure DoSearchDb(ch:string);
  function SelectedKode:string;
end;

begin
  inherited;
  //PriceSelectCell(self,Price.col,price.Row,cs);
      if (Key=VK_F10) then
    ModalResult:=mrCancel;

  //   if (Key=VK_Right) then DoSearchNext;

  if (Key=VK_Up) or (Key=VK_Down) then
  begin
    // Сброс при стрелках
    SearchStr:='';
    LeaveSearchHint;
  end;

  if (Key=VK_BACK) and (Shift=[]) then
  begin
    SearchStr:=Copy(SearchStr,1,Length(SearchStr)-1);
    if SearchStr<>'' then
      //         DoSearchHint(Price.Row,Price.Col,SearchStr)
    else
    begin
      LeaveSearchHint;
      /// Поиск а-ля Афарм
      s:='';
```

```
      InputQuery('Пошук а-ля Афарм _','Наименование',s);

      if s<>'' then
      begin
        s:=AnsiUpperCase(s);
        setlength(saArr,0);


        if pos(' ',s)>1 then
        begin
          s1:=copy(s,1,pos(' ',s)-1);
          s2:=copy(s,pos(' ',s)+1,length(s));
        end
        else
        begin
          s1:=s;
          s2:='';
        end;

        BEGIN
        END;
      END;
    end;
  end;

end;
{

procedure TPriceForm.DoSearchNext;
var i,Len,CurCol:integer;
    SUp:string;
begin
   fgSearch:=true;
   try
     if SearchStr<>'' then
     begin
       SUp:=AnsiUpperCase(SearchStr);
       Len:=Length(SearchStr);
       CurCol:=Price.Col;
       i:=Price.Row+1;
       if N1.Checked then
       begin
         while (i<Price.RowCount) and
           (AnsiUpperCase(Copy(Price.Cells[CurCol,i],1,Len))<>SUp)
              do inc(i);
       end else
       begin
         while (i<Price.RowCount) and
           (Pos(SUp,AnsiUpperCase(Price.Cells[CurCol,i]))=0)
              do inc(i);
       end;
       if i=Price.RowCount then
       begin
         Beep;
       end else
       begin
         LeaveSearchHint;
         LeaveHint;
         Price.Row:=i;
         DoSearchHint(i,CurCol,SearchStr);
       end;
    end;
```

```
   finally
     fgSearch:=false;
   end;
end;  }

{procedure TPriceForm.PriceDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
var s,s1:string;
dd:Tdatetime;

begin
dd:=DBGrid1.DataSource.DataSet['FITDATE'];
//6
if dd<IncMonth(now,12) then
  begin
  Params.Background:=clGreen;
  Params.Font.Color:=clWhite;
  end;

if dd<IncMonth(now,1) then
  begin
  Params.Background:=$00003C6B;
  Params.Font.Color:=clWhite;
  end;

if dd<now() then
  Params.Background:=clRed;
end;

end;

procedure TPriceForm.ActionTovInfoExecute(Sender: TObject);
begin
FmInstr:=TFmInstr.Create(self);
FmInstr.ButtonWord.Enabled:=false;
if DataM.DB_GoodsAnnotDoc.BlobSize>100 then begin
        FmInstr.ButtonWord.Enabled:=true;
        end;

FmInstr.ShowModal;
end;
procedure TPriceForm.ActionAllOstsExecute(Sender: TObject);
var user,s:Ansistring;
begin
user:=GetEnvironmentVariable('USERNAME');
user:=AnsiUpperCase(user);
try
s:='SELECT dbo.vw_Remains_Enterplans.Item_code,
dbo.vw_Remains_Enterplans.Warehouse_Code, dbo.vw_Remains_Enterplans.Remain_psc,
';
s:=s+'dbo.vw_Remains_Enterplans.Bad_Remain_psc,
dbo.vw_Remains_Enterplans.CntBig, dbo.DIM_Warehouses.Filial,
dbo.DIM_Warehouses.City ';
s:=s+'FROM dbo.vw_Remains_Enterplans INNER JOIN dbo.DIM_Warehouses ON
dbo.vw_Remains_Enterplans.Warehouse_Code = dbo.DIM_Warehouses.Warehouse_code
WHERE (dbo.vw_Remains_Enterplans.Item_code = ';
DM_K.VQAllRemains.SQL.Text:=s+Trim(SelectedKode)+') and
dbo.vw_Remains_Enterplans.Remain_psc>0';

        if not(DM_K.VQAllRemains.Active) then DM_K.VQAllRemains.Open
        else DM_K.VQAllRemains.ExecSQL;
finally
```

```
if DM_K.VQAllRemains.RecordCount>0 then
        FormRemaice.ShowModal
        else
        MessageBeep($fffff);
DM_K.VQAllRemains.Clse;

end;
```

3.3 Короткий опис основних функцій програми

Розглянемо основні функціональні можливості програмного комплексу. В основному вікні (рисунок 3.4.) ми можемо виконувати наступні функції (таблиця 3.2)

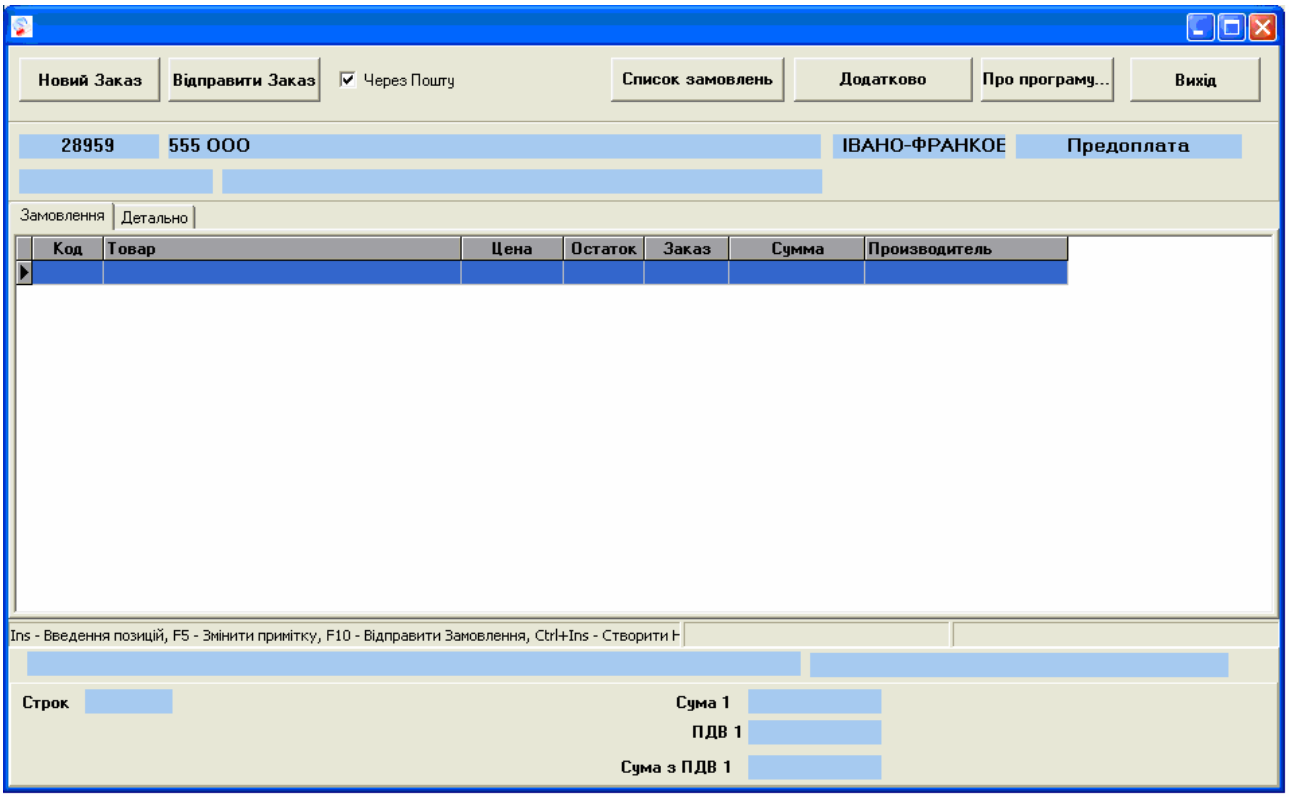


Рисунок 3.4 - Головне вікно «Набране замовлення»

Таблиця 3.2 - Функції головного вікна «Набране замовлення

Кнопка	Клавіша	Дія
«Нове замовлення»	[Ctrl+Ins]	Створити нове замовлення
«Відправити замовлення»	[F10]	Відправити поточне замовлення
«Список замовлень»	[Space]	Виклик вікна вибору замовлення з архіву
«Експорт»		Можливість експортувати замовлення в Excel
	[F5]	Змінити примітку

- Код клієнта

- Найменування клієнта

- Місто клієнта

- Відстрочення платежу

- Дата і час створення замовлення

- Примітка замовлення

Нижче знаходитися перегляд замовлення в двох можливих режимах

- «**Замовлення»** список набраних товарних позицій для вибраного замовлення.

- «**Детально»** детальна інформація про умови, суми, відмови.

Далі вказана кількість рядків і суми замовлення

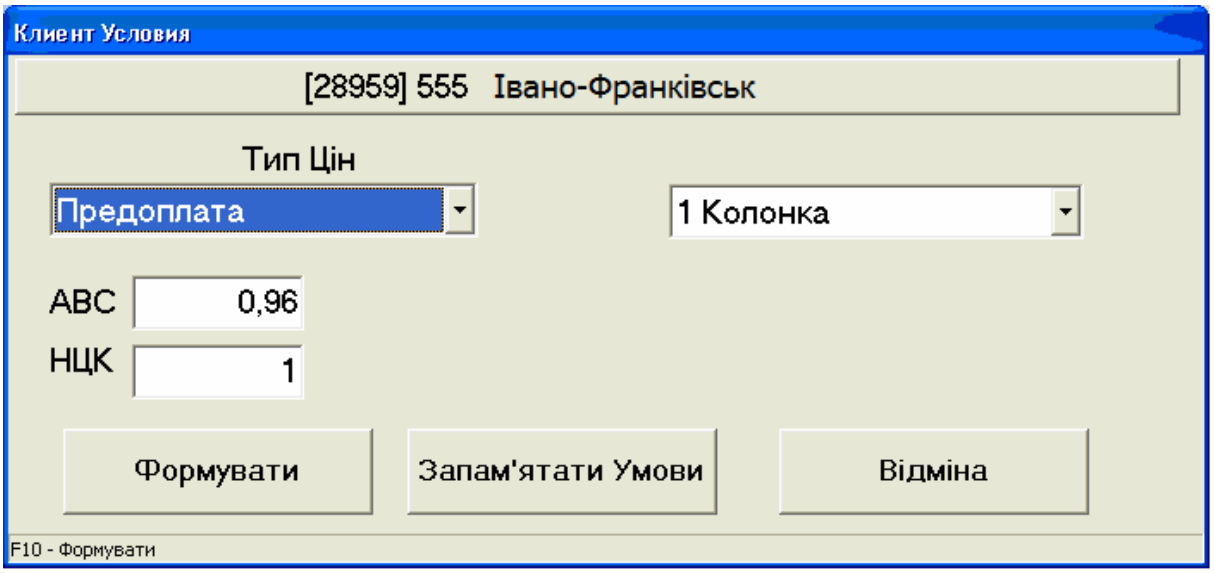


Рисунок 3.5 – Вікно роботи з клієнтом (Вибір клієнта, умов роботи і цін)

Таблиця 3.3 - Функції вікна для роботи з клієнтом

	Клавіша	Дія
Клієнт	[Enter]	Виклик вікна вибору клієнта
Тип Цін	[F10]	Вибір відстрочення
Колонки		Вибір роботи з 1-ою або 2-ма колонками замовлення

АВС		Коефіцієнт знижки
НЦК		Коефіцієнт націнки
Формувати		Почати роботу з новим замовленням вибраного клієнта на вказаних умовах
Запам'ятати умови		Запам'ятати умови роботи з даним клієнтом. (Наступного разу будуть встановлені автоматично для даного клієнта)
Відміна		Відмовитися від створення замовлення

Перехід між полями проводитися за допомогою клавіші Enter або Tab. При появі даного вікна перше натиснення Enter викликає вікно вибору клієнта (див. далі), після переходить по полях в такому порядку Тип Цін, АВС, НЦК і переходить на кнопку Формувати. В кожному випадку можна або вибрати інше значення або натиснути Enter і перейти до наступного. Поле Колонки можна змінити тільки за допомогою миші. Натиснення Enter на кнопці Формувати створює нове замовлення.

У вікні вибору клієнта відображають наступні Дані

- Код клієнта
- Статус клієнта (Заблокований і причина блокування, Розблокований)
- Найменування клієнта
- Місто клієнта
- Ліміт Кредиту
- Загальна заборгованість
- Прострочена заборгованість

У вікні пошуку вводитися частину назви або код клієнта якого необхідно знайти, якщо пошук дає однозначний результат те проводитися позиціонування в списку клієнтів на нім інакше якщо пошук дав декілька варіантів результату те виводитися вікно уточнення вибору в якому необхідно вказати необхідного клієнта і натиснути **[Enter]**

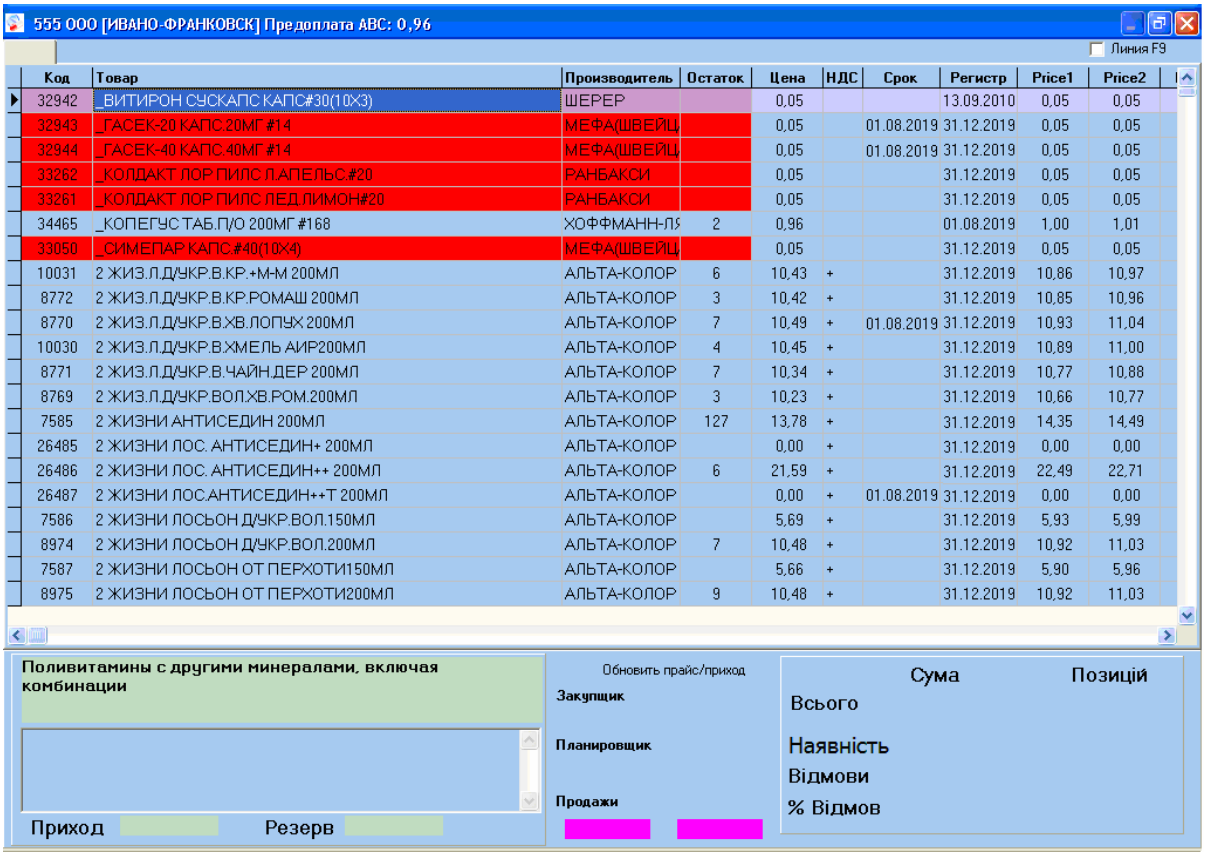


Рисунок 3.6 – Вікно роботи з з прайс-листом

Таблиця 3.4 - Функції вікна роботи з прайс-листом

Кнопка	Клавіша	Дія
Швидкий пошук	[букви, цифри]	Швидкий пошук товару по перших буквах
Пошук товарів по АТС аналогам	[F7]	Виклик вікна пошуку товарів по АТС аналогам
Пошук товарів по АТС	[Alt+F7]	Виклик вікна пошуку товарів по АТС
Введення позиції	[Enter]	Виклик вікна введення або зміни позиції замовлення
Інформація про товар	[F3]	Виклик вікна інформації про товар
Пошук товару	[<BackSpace]	Виклик вікна пошуку товару
Вихід	[Esc]	Повернутися до вікна замовлення

Дані, що відображаються

- Код Товару
- Найменування Товару
- Виробник
- Залишок Товару
- Ціна клієнта
- Ознака ПДВ
- Термін Придатності
- Термін реєстрації
- Ціни 1,2,3 колонок прайс-листа
- Митна ціна
- Кількість товару в упаковці
- Група товару по АТС класифікації
- Примітка до товару
- Очікуваний прихід товару
- Інформація про відповідальних осіб по закупівлі і плануванню товару
- Інформація про продажі товару у минулому і поточному місяцях
- Статистика поточного замовлення суми і процентне відношення замовлення і відмов по ньому.

У даному вікні пошуку вводитися частину назви або код товару якого необхідно знайти, якщо пошук дає однозначний результат те проводитися позиціонування в списку товарів на нім інакше якщо пошук дав декілька варіантів результату те виводитися вікно уточнення вибору в якому необхідно вказати необхідний товар і натиснути [Enter]

Можливо також указувати декілька частин назви товару розділених пропусками, наприклад пошукова фраза «шпр 10 1» допоможе знайти всі товари назви яких містять вказані частини тексту – наприклад «шприц 10 мл №1» і т.п.

Вікно уточнення вибору Товару містить дві частини:

- Точний збіг – список товарів назва яких починається з шуканої фрази.

- Частковий збіг – список товарів назву яких містять шукану фразу.

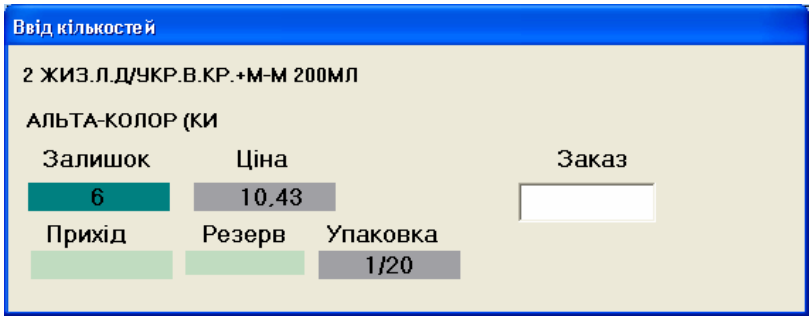


Рисунок 3.7 – Вікно введення/зміни кількості

Дані, що відображаються:

- Найменування Товару
- Виробник
- Залишок Товару
- Ціна клієнта
- Очікуваний прихід товару
- Кількість в упаковці

Залежно від типу замовлення (1 або 2 колонки) містить відповідно одне або два поля введення для кількостей.

Висновки до розділу

В даному розділі показано практичну реалізацію сервера бази даних та моделі клієнта із застосування мови програмування Delphi. Вказано основні частини і наведено лістинг програмного комплексуі його взаємозв'язок із серверною базою даних. Описано основні функціональні можливості програмного комплексу.

ВИСНОВКИ

Під час виконання дипломної роботи було розширено теоретичні і практичні знання новітніх технологій доступу до даних і мов програмування. Було створено програмний продукт забезпечення базових фармацевтичних сервісів для аптечного складу.

Клієнт-серверна взаємодія проекту побудована на основі платформи доступу до даних на сервері ADO, що включає об'єкти і елементи управління часу розробки і контекст середовища виконання. Крім того розширено знання мови формування і роботи з базами даних SQL (забезпечена можливість роботи на MS SQL та MS Access і легка зміна конфігурації).

Реалізація даного проекту була проведена без залучення громіздких засобів роботи з базами даних, які носять універсальний характер і до того ж вимагають необхідну базу знань по теорії баз даних.

Використання ООП засобів по створенню додатків що працюють в операційній системі Windows і зокрема додатків баз даних, дозволило створити програмний продукт максимально орієнтований на кінцевого користувача, який не досвідчений в питаннях теорії баз даних.

Вся необхідна робота по здійсненню методів доступу до інформації що зберігається в базі даних, її модифікації, підтримці бази даних в цілісному вигляді прихована усередині і користувачеві немає необхідності знати про неї, щоб успішно вирішувати все коло завдань, що виникають і пов'язані з використанням інформації базі даних, що зберігається. Більш того, програмний інтерфейс максимально полегшує роботу по роботі з базою даних. Навіть звернення до бази даних з складними запитами здійснюється у такому вигляді, що структура даних, що повертаються, видно ще до його виконання.

Завдання вирішуються в рамках даної СУБД, з максимальною простотою, зручністю і швидкістю. Даний програмний продукт без сумніву може конкурувати з існуючими аналогами даних і засобами управління ними.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Багрова І.В. Нормування праці. Навч. посібник. – К:Центр навч. літератури, 2003. – 212 с.
2. Бедрій Л.І., Геврик Є.О., Кіт Я.І., Мурін О.С. Охорона праці. Навчальний посібник для студенів вищих навчальних закладів . – Львів, 2000 – 279 с.
3. Будівельні норми і правила СНиП II-4-79 «Естественное и искуственное освещение. Нормы проектирования»
4. Горев А., Ахаян Р., Макашарипов С. «Эффективная работа с СУБД».СПб.:Питер, 1997.— 704 с.,ил.
5. Дейт К., "Введение в системы баз данных", Москва, 'Наука', 1980 г.
6. Дж. Тельман, "Основы систем баз данных", Москва, Финансы и статистика', 1983г.
7. Дж. Мартин., "Организация баз данных в вычислительных системах" М: Мир 1978г.
8. Кириллов В.В. Структуризованный язык запросов (SQL). – СПб.: ИТМО, 1994. – 80 с.
9. Тиори Т., Фрай Дж. Проектирование структур баз данных. В 2 кн., – М.: Мир, 1985. Кн. 1. – 287 с.: Кн. 2. – 320 с.
10. Цикритизис Д., Лоховски Ф. Модели данных. – М.: Финансы и статистика, 1985. – 344 с.
11. Диго С.М. Проектирование и использование баз данных. – Москва: Финансы и статистика, 1995. – 592 с.
12. Мейер М. Теория реляционных баз данных. – Москва: Мир, 1998. – 608 с.
13. Джексон Г. Проектирование реляционных баз данных для использования с микро-ЭВМ. – Москва: Мир, 2001. – 256 с.
14. Шумаков П.В. Delphi и создание баз данных. – Москва: Нолидж, 1997. – 320 с.

15. Larson B. Delivering Business Intelligence with Microsoft SQL Server 2008. – McGraw-Hill Osborne Media; 2 edition. – 792 p.
16. K. Sommer, N. O. Friedrich, S. Bietz, M. Hilbig, T. Inhester, and M. Matthias Rarey, “UNICON: A Power¬ful and Easy-to-Use Compound Library Converter,” Journal of Chemical Information and Modelling, no. 56, pp. 1105-1111, 2016.
17. M. Sud, “Maya Chem Tools: An Open Source Package for Computational Drug Discovery,” Journal of Chemical Information and Modelling, no. 56 (12), pp. 2292-2297, 2016. https://doi.org/: 10.1021/acsjcim.6b00505.
18. Бази даних та інформаційні системи. [Електронний ресурс]. Режим доступу: http ://oracle.co m. edge suite. net/timeline/j ava/ .

ДОДАТКИ

Додаток А

Лістинг файлу UslovFrm.pas

```
unit UslovFrm;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, Spin, ExtCtrls,  Mask, Menus, ComCtrls, DBCtrls;

type
TUslovForm = class(TForm)
  Button1: TButton;
  Button2: TButton;
  ComboBoxPriceSelect: TComboBox;
  Label3: TLabel;
  Label4: TLabel;
  Label5: TLabel;
  ComboBoxFormZak: TComboBox;
  Label6: TLabel;
  Button3: TButton;
  Label2: TLabel;
  PopupMenu1: TPopupMenu;
  N1: TMenuItem;
  StatusBar1: TStatusBar;
  Button4: TButton;
  DBEditAVS: TDBEdit;
  DBEditNCK: TDBEdit;
  DBTextKoefPrice: TDBText;
  DBEditDays: TDBEdit;
  Label1: TLabel;
  procedure FormShow(Sender: TObject);
  procedure FormKeyPress(Sender: TObject;
  var Key: Char);
  procedure Button3Click(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure DBEditDaysChange(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
  Predopl:integer;
  MaxDefer:integer;
  DayCost:double;
  DeferArray:array [1..5] of integer;

end;

var
UslovForm: TUslovForm;
debsf:boolean;
selbedid:string;
DebTX,DebTown:string;

implementation
```

```
uses debselect, data_acc, MainFrm, ProgressForm, DB, LocalDb, OrderFunc;

{$R *.DFM}

procedure TUslovForm.FormShow(Sender: TObject);
begin




  DM_Local.ServerInfoDebs.Open;

  UslovFrm.UslovForm.Button3.Caption:='['+(selbedid)+'] '+DebTX+' '+DebTown;

  if DM_Local.ServerInfoDebs.Locate('DebCodeInt',selbedid,[]) then
  begin
    // Восстановление

MainForm.DBTextPrice.DataSource.DataSet.FieldByName('PriceCol').AsInteger:=DM_Lo
cal.ServerInfoDebs.FieldValues['PriceCol'];


self.ComboBoxPriceSelect.ItemIndex:=DM_Local.ServerInfoDebs.FieldValues['PriceCo
l']-1;

MainForm.DBTextPrice.DataSource.DataSet.FieldByName('PriceCol').AsInteger:=self.
ComboBoxPriceSelect.ItemIndex+1;
    DBEditAVS.Field.Value:=DM_Local.ServerInfoDebs.FieldValues['AVS'];
    DBEditNCK.Field.Value:=DM_Local.ServerInfoDebs.FieldValues['NCK'];
  end
  ELSE
  begin

MainForm.DBTextPrice.DataSource.DataSet.FieldByName('PriceCol').AsInteger:=1;
    self.ComboBoxPriceSelect.ItemIndex:=0;
    //         self.ComboBoxPriceSelect.ItemIndex:=0;
    DBEditAVS.Field.Value:=1;
    DBEditNCK.Field.Value:=1;

  end;

  DM_Local.ServerInfoDebs.Close;

  if MainForm.DBTextAVS.Field.IsNull then
    MainForm.DBTextAVS.Field.Value:=1;
  if MainForm.DBTextNCK.Field.IsNull then
    MainForm.DBTextNCK.Field.Value:=1;

  self.Button3.SetFocus;
end;


procedure TUslovForm.FormKeyPress(Sender: TObject;
var Key: Char);
begin
  If Key=#13 then
  begin
    Key:=#0;
    Perform(WM_NextDlgCtl,0,0);
  end;
```

```
end;

procedure TUslovForm.Button3Click(Sender: TObject);
label retr;

var errl:boolean;
usfr:integer;

begin
  self.Button3.Enabled:=false;
  retr:
  errl:=false;
  debsf:=false;
  if debsf then
    exit;
  { DONE : Загрузка Дебиторов }
  setlength(DebitrDov,0);

  try
{    DM_Local.DB_Clients.SQL.Text:='SELECT * '+
    ' FROM CLIENTS '+
    ' ORDER BY PayerName';}

    DM_Local.DB_Clients.Open;

    if DataM.VQ_Debtors.Active then
    DataM.VQ_Debtors.Close;

    DataM.VQ_Debtors.Open;

    setlength(DebitrDov,0);

    while not(DM_Local.DB_Clients.Eof) do
    begin
      try
        setlength(DebitrDov,length(DebitrDov)+1);

DebitrDov[high(DebitrDov)].DebCode:=DM_Local.DB_Clients.fieldbyName('ID').AsStri
ng;

DebitrDov[high(DebitrDov)].DebCodeS:=DM_Local.DB_Clients.fieldbyName('OKPO').AsS
tring;

DebitrDov[high(DebitrDov)].EnFL:=DM_Local.DB_Clients.fieldbyName('ZAP_OBSCH').As
Integer;

DebitrDov[high(DebitrDov)].DebName:=DM_Local.DB_Clients.fieldbyName('PAYERNAME')
.AsString;

DebitrDov[high(DebitrDov)].Town:=DM_Local.DB_Clients.fieldbyName('Town').AsStrin
g;
      finally
      end;

      DM_Local.DB_Clients.Next;
    end;
    DM_Local.DB_Clients.Close;
    except
    errl:=true;
  end;
```

```
  if errl then
    goto retr;

  //ProgresForm.Hide;

  self.StatusBar1.SimpleText:='F10 - Формировать';
  Application.ProcessMessages;

  FormDebSel:=TFormDebSel.Create(self);


  usfr:=FormDebSel.ShowModal;

  FormDebSel.Free;

  if usfr=mrOk then
  begin
    // Сохр Услов

    DM_Local.ServerInfoDebs.Open;

    UslovFrm.UslovForm.Button3.Caption:='['+(selbedid)+'] '+DebTX+' '+DebTown;

    if DM_Local.ServerInfoDebs.Locate('DebCodeInt',selbedid,[]) then
    begin
      // Восстановление условий

MainForm.DBTextPrice.DataSource.DataSet.FieldByName('PriceCol').AsInteger:=DM_Lo
cal.ServerInfoDebs.FieldValues['PriceCol'];


self.ComboBoxPriceSelect.ItemIndex:=DM_Local.ServerInfoDebs.FieldValues['PriceCo
l']-1;

MainForm.DBTextPrice.DataSource.DataSet.FieldByName('PriceCol').AsInteger:=self.
ComboBoxPriceSelect.ItemIndex+1;
      DBEditAVS.Field.Value:=DM_Local.ServerInfoDebs.FieldValues['AVS'];
      DBEditNCK.Field.Value:=DM_Local.ServerInfoDebs.FieldValues['NCK'];
      DBEditDays.Field.Value:=DM_Local.ServerInfoDebs.FieldValues['Days'];
    end
    ELSE
    begin

MainForm.DBTextPrice.DataSource.DataSet.FieldByName('PriceCol').AsInteger:=1;
      self.ComboBoxPriceSelect.ItemIndex:=0;
      DBEditAVS.Field.Value:=1;
      DBEditNCK.Field.Value:=1;

    end;

    DM_Local.ServerInfoDebs.Close;
    self.ComboBoxPriceSelect.SetFocus;
  end;

  if usfr=mrYesToAll  then
  begin
    self.ComboBoxPriceSelect.SetFocus;
    debsf:=false;
    self.Button3.Enabled:=true;
    self.Button1Click(self);
```

```
  end;

  debsf:=false;
  self.Button3.Enabled:=true;
end;




procedure TUslovForm.Button1Click(Sender: TObject);
begin
  if not(selbedid>'') then
  begin
    self.Button3.Click;
    exit;
  end;


  MainForm.DBTextComment.Field.Value:=' ';

  DM_Local.VQ_OrderH['PriceCol']:=self.ComboBoxPriceSelect.ItemIndex+1;

  //MainForm.DBTextPrice.Field.Value:=self.ComboBoxPriceSelect.ItemIndex+1;

  MainForm.DBTextCID.Field.Value:=selbedid;

  DM_Local.VQ_OrderH['Type']:=ComboBoxFormZak.ItemIndex+1;
  //SetOrderType();

  Self.ModalResult:=mrOk;

end;

procedure TUslovForm.Button4Click(Sender: TObject);
begin
  //if data_acc.DebCode<5 then exit
  try;
    DM_Local.ServerInfoDebs.Open;
    if DM_Local.ServerInfoDebs.Locate('DebCodeInt',selbedid,[]) then
      // Корректировка
    DM_Local.ServerInfoDebs.edit
    else
      // Добавление
    DM_Local.ServerInfoDebs.Append;

    DM_Local.ServerInfoDebs.FieldValues['DebCodeInt']:=selbedid;

DM_Local.ServerInfoDebs.FieldValues['PriceCol']:=self.ComboBoxPriceSelect.ItemIn
dex+1;
    DM_Local.ServerInfoDebs.FieldValues['AVS']:=DBEditAVS.Field.Value;
    DM_Local.ServerInfoDebs.FieldValues['NCK']:=DBEditNCK.Field.Value;
    DM_Local.ServerInfoDebs.FieldValues['Days']:=DBEditDays.Field.Value;
    DM_Local.ServerInfoDebs.Post;

    showmessage('Условия Сохранены');
    except
    showmessage('Ошибка Сохранения');
  end;
```

```
  DM_Local.ServerInfoDebs.Close;
  self.Button1.SetFocus;
end;

procedure TUslovForm.FormCreate(Sender: TObject);
begin
  DataM.VQ_PriceCols.Open;
  self.ComboBoxPriceSelect.Items.Clear;

  while not(DataM.VQ_PriceCols.eof) do
  begin
    self.ComboBoxPriceSelect.Items.Add(DataM.VQ_PriceCols['PriceColName']);
    DataM.VQ_PriceCols.Next;
  end;
  DataM.VQ_PriceCols.Close;
  self.ComboBoxPriceSelect.ItemIndex:=0;
end;

procedure TUslovForm.DBEditDaysChange(Sender: TObject);
var i:integer;
begin
  try
    i:=DBEditDays.Field.Value;
    if i<0 then
      DBEditDays.Field.Value:=0;
    except
    DBEditDays.Field.Value:=0;

  end;


end;

initialization
debsf:=false;

end.
```

Додаток Б – Лістинг файлу SearchAccept.pas

```
unit SearchAccept;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ComCtrls, VListView, ExtCtrls, StdCtrls, Grids, DBGrids, DBCtrls;

type
TFormSrchAcc = class(TForm)
    PanelBottom: TPanel;
    PanelTop: TPanel;
  PageControl1: TPageControl;
  TabSheet1: TTabSheet;
  DBGrid_StName: TDBGrid;
  DBText1: TDBText;
  PageControl2: TPageControl;
  TabSheet2: TTabSheet;
  DBGrid_InName: TDBGrid;
  DBText2: TDBText;
```

```
    Timer1: TTimer;
    PanelAnalog: TPanel;
    PageControl3: TPageControl;
    TabSheet3: TTabSheet;
    DBText3: TDBText;
    DBGridAn: TDBGrid;
    DBText4: TDBText;
  procedure FormShow(Sender: TObject);
  procedure DBGrid_StNameKeyDown(Sender: TObject;
  var Key: Word;
  Shift: TShiftState);
  procedure DBGrid_StNameKeyPress(Sender: TObject;
  var Key: Char);
  procedure DBGrid_InNameKeyPress(Sender: TObject;
  var Key: Char);
  procedure DBGrid_InNameKeyDown(Sender: TObject;
  var Key: Word;
  Shift: TShiftState);
  procedure FormKeyPress(Sender: TObject;
  var Key: Char);
    procedure Timer1Timer(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure DBGridAnKeyPress(Sender: TObject; var Key: Char);
private
  { Private declarations }
public
  { Public declarations }
end;

var
SearchStr,SelID:string;
sanalog:boolean;

FormSrchAcc: TFormSrchAcc;
saArr:array of record
  kod,pos:integer;
  tov,suppw:String;
  ost:real;
end;

implementation

uses data_acc, LocalDb, StrUtils;

{$R *.dfm}

procedure TFormSrchAcc.FormShow(Sender: TObject);
var escch:char;
sid:string;
id:integer;
begin

if sanalog then
begin
PanelBottom.Visible:=false;
PanelTop.Visible:=false;
DataM.Q_Goods_Analog.Parameters.ParamByName('AGRP').Value:=(SearchStr);
DataM.Q_Goods_Analog.Open;
if DataM.Q_Goods_Analog.RecordCount>0 then
begin
  DBGridAn.SetFocus;
```

```
  DBGridAn.DataSource.DataSet.First;
end
else
            begin
            escch:='~';
                  self.FormKeyPress(self,escch);
      Self.Timer1.Enabled:=true;
            end;
end
else
begin
PanelAnalog.Visible:=false;
PanelTop.Height:=300;
PanelBottom.Height:=300;
  with DataM.Q_Goods_StName do
  begin
    if Active then
      close;
{    if Not(DM_Local.VQ_OrderH.FieldByName('PriceCol').IsNull) then
      Parameters.ParamValues['Pcol']:=DM_Local.VQ_OrderH['PriceCol']
    else
      Parameters.ParamValues['Pcol']:=8;          }
    Parameters.ParamValues['SStr']:=AnsiReplaceStr(SearchStr,' ','%')+'%';
    try
    id:=strtoint(SearchStr);
    except
    id:=-1;
    end;
    sid:=IntToStr(id);


    Parameters.ParamValues['SID']:=sid;
    Open;
  end;

  with DataM.Q_Goods_InName do
  begin
            if Active then
      close;
{    if Not(DM_Local.VQ_OrderH.FieldByName('PriceCol').IsNull) then
      Parameters.ParamValues['Pcol']:=DM_Local.VQ_OrderH['PriceCol']
    else
      Parameters.ParamValues['Pcol']:=8;           }

    Parameters.ParamValues['SStr']:='%'+AnsiReplaceStr(SearchStr,' ','%')+'%';
    Parameters.ParamValues['SStr2']:=AnsiReplaceStr(SearchStr,' ','%')+'%';

    Open;
  end;



  if DBGrid_StName.DataSource.DataSet.RecordCount>0 then
  begin
    DBGrid_StName.SetFocus;
    DBGrid_StName.DataSource.DataSet.First;
            DBGrid_StName.SelectedField.FocusControl;
            if Not(DBGrid_InName.DataSource.DataSet.RecordCount>0) then
            begin
            // no bottom
      PanelBottom.Height:=30;
```

```
      PanelBottom.Visible:=false;
      PanelTop.Align:=alClient;
            end;
      end
      else
      begin
      // no top
      PanelTop.Height:=30;
      PanelTop.Visible:=false;
      PanelBottom.Align:=alClient;

            if DBGrid_InName.DataSource.DataSet.RecordCount>0 then
            begin
                  DBGrid_InName.SetFocus;
                  DBGrid_InName.DataSource.DataSet.First;
                  DBGrid_InName.SelectedField.FocusControl;
            end
            else
            begin
            escch:='~';
//    Beep;
                  self.FormKeyPress(self,escch);
      Self.Timer1.Enabled:=true;
//                ModalResult:=mrCancel;
//                Application.ProcessMessages;
//                Self.Close;
            end;
      end;
end;

end;

procedure TFormSrchAcc.DBGrid_StNameKeyDown(Sender: TObject;
var Key: Word;
Shift: TShiftState);
begin
  if key=vk_DOWN then
    if DBGrid_StName.DataSource.DataSet.Eof then
      if DBGrid_InName.DataSource.DataSet.RecordCount>0 then
  begin
    DBGrid_InName.SetFocus;
    DBGrid_InName.DataSource.DataSet.First;
    DBGrid_InName.SelectedField.FocusControl;
  end;

  if key=vk_Up then
    if DBGrid_StName.DataSource.DataSet.Bof then
      if DBGrid_InName.DataSource.DataSet.RecordCount>0 then
  begin
    DBGrid_InName.SetFocus;
    DBGrid_InName.DataSource.DataSet.Last;
    DBGrid_InName.SelectedField.FocusControl;
  end;

end;

procedure TFormSrchAcc.DBGrid_StNameKeyPress(Sender: TObject;
var Key: Char);
begin
  if key=#13 then
    if Not(DBGrid_StName.DataSource.DataSet.Fieldbyname('ID').IsNull) then
```

```
  begin
    SelID:=DBGrid_StName.DataSource.DataSet['ID'];
    self.ModalResult:=mrOk;
  end;


end;

procedure TFormSrchAcc.DBGrid_InNameKeyPress(Sender: TObject;
var Key: Char);
begin
  if key=#13 then
    if Not(DBGrid_InName.DataSource.DataSet.Fieldbyname('ID').IsNull) then
  begin
    SelID:=DBGrid_InName.DataSource.DataSet['ID'];
    self.ModalResult:=mrOk;
  end;

end;

procedure TFormSrchAcc.DBGrid_InNameKeyDown(Sender: TObject;
var Key: Word;
Shift: TShiftState);
begin
  if key=vk_DOWN then
    if DBGrid_InName.DataSource.DataSet.Eof then
      if DBGrid_StName.DataSource.DataSet.RecordCount>0 then
  begin
    DBGrid_StName.SetFocus;
    DBGrid_StName.DataSource.DataSet.First;
    DBGrid_StName.SelectedField.FocusControl;
  end;

  if key=vk_Up then
    if DBGrid_InName.DataSource.DataSet.Bof then
      if DBGrid_StName.DataSource.DataSet.RecordCount>0 then
  begin
    DBGrid_StName.SetFocus;
    DBGrid_StName.DataSource.DataSet.Last;
    DBGrid_StName.SelectedField.FocusControl;
  end;

end;

procedure TFormSrchAcc.FormKeyPress(Sender: TObject;
var Key: Char);
begin
  if key=#27 then
    self.ModalResult:=mrAbort;
  if key='~' then
    self.ModalResult:=mrCancel;

end;

procedure TFormSrchAcc.Timer1Timer(Sender: TObject);
begin
Timer1.Enabled:=false;
Self.ModalResult:=mrCancel;
end;

procedure TFormSrchAcc.FormClose(Sender: TObject;
```

```
  var Action: TCloseAction);
begin
DataM.Q_Goods_InName.Close;
DataM.Q_Goods_StName.Close;
DataM.Q_Goods_Analog.Close;
end;

procedure TFormSrchAcc.DBGridAnKeyPress(Sender: TObject; var Key: Char);
begin
  if key=#13 then
    if Not(DBGridAn.DataSource.DataSet.Fieldbyname('ID').IsNull) then
  begin
    SelID:=DBGridAn.DataSource.DataSet['ID'];
    self.ModalResult:=mrOk;
  end;

end;

end.
```

Додаток В – Лістинг файлу OrderFunc.pas

```
unit OrderFunc;

interface

function getFileVersionStr(fn:string):String;
function getFileVersion64(fn:string):INT64;

function GetOstByCode(Code: string; OnLine:boolean): Real;

function GetTName:String;
procedure AddPos();
procedure EditPos();

function ZakazSend:Boolean;
Procedure SetZakazSend(b:boolean);

procedure RecalcStat;

function OrderType:Integer;
procedure SetOrderType(i:Integer);

implementation

uses PriceFrm, SysUtils, VispUnit1, data_acc, LocalDb, DB, CountInput, Controls
,
Math, MainFrm,Types, Windows, Classes;

function GetOstByCode(Code: string;
OnLine:boolean): Real;

begin
  result:=0;
  case OnLine of
    false:
    begin
      DataM.DB_Osts.Open;
      Result:=DataM.DB_Osts.Lookup('ID',Code,'Total');
      DataM.DB_Osts.Close;
    end;
    true: begin

    end;

  end;

end;


function OrderType:Integer;
begin
  if DM_Local.VQ_OrderH.FieldByName('Type').IsNull=false then
    Result:=DM_Local.VQ_OrderH['Type']
  else
    result:=0;
end;

procedure RecalcStat;
var o2:boolean;
begin
```

```
  o2:=OrderType=2;
  mainform.DBGr_Order.Columns[6].Visible:=O2;
  mainform.DBGr_Order.Columns[7].Visible:=O2;

  mainform.LbSum2.Visible:=o2;
  mainform.DBTextSum2.Visible:=o2;

  mainform.LbSumT.Visible:=o2;
  mainform.DBTextSumT.Visible:=o2;

  mainform.DBTextNDS2.Visible:=o2;
  mainform.DBTextSNDS2.Visible:=o2;
  mainform.DBTextNDS.Visible:=o2;
  mainform.DBTextTNDS.Visible:=o2;

  mainform.LabelNDS.Visible:=o2;
  mainform.LabelNds2.Visible:=o2;
  mainform.LabelTnds.Visible:=o2;
  mainform.LabelSNds2.Visible:=o2;

  with DM_Local.VQ_OrderL do
  begin
    if Active then
      close;
    Parameters.ParamByName('OrdID').Value:=DM_Local.VQ_OrderH['OrderID'];
    Open;
  end;

  {with DM_Local.VQ_OrderL_Nac do begin
if Active then close;
Parameters.ParamByName('OrdID').Value:=DM_Local.VQ_OrderH['OrderID'];
Open;
end;

with DM_Local.VQ_OrderL_NDS do begin
if Active then close;
Parameters.ParamByName('OrdID').Value:=DM_Local.VQ_OrderH['OrderID'];
Open;
end;

with DM_Local.VQ_OrderL_bNDS do begin
if Active then close;
Parameters.ParamByName('OrdID').Value:=DM_Local.VQ_OrderH['OrderID'];
Open;
end;}


  with DM_Local.VQ_Ord_Stat do
  begin
    if Active then
      close;
    Parameters.ParamByName('OrdID').Value:=DM_Local.VQ_OrderH['OrderID'];
    open;
  end;

end;

procedure SetOrderType(i:Integer);
var ed:boolean;
begin
```

```
  ed:=Not(DM_Local.VQ_OrderH.State=dsEdit);
  if ed then
    DM_Local.VQ_OrderH.Edit;
  DM_Local.VQ_OrderH['Type']:=i;
  if Not(ed) then
    DM_Local.VQ_OrderH.Post;
end;

Procedure SetZakazSend(b:boolean);
begin
  if Not(DM_Local.VQ_OrderH.State=dsedit) then
  begin
    DM_Local.VQ_OrderH.Edit;
    DM_Local.VQ_OrderH['Sended']:=b;
    DM_Local.VQ_OrderH.Post
  end
  else
    DM_Local.VQ_OrderH['Sended']:=b;

end;

function ZakazSend:Boolean;
begin
  if DM_Local.VQ_OrderH.FieldByName('Sended').IsNull
  then
    result:=True
  else
    Result:=DM_Local.VQ_OrderH['Sended'];
end;


function GetTName:String;
begin
  Result:=DataM.DS_Goods.DataSet['Tovar'];
end;

function GetTID:String;
begin
  Result:=DataM.DS_Goods.DataSet['ID'];
end;

procedure AddPos();
var
ok:Boolean;
begin

  with DM_Local.VQ_OrderL do
  begin

    if Locate('TID',GetTID,[])
    then
      edit
    else
    Append;

    FieldByName('TID').AsString:=GetTID;
    FieldByName('Tname').AsString:=GetTName;
            FieldByName('Tvendor').AsString:=DataM.DS_Goods.DataSet['Proizv'];
            FieldByName('Price').Value:=DataM.DS_Goods.DataSet['PriceVal'];
            FieldByName('NDS').AsString:=DataM.DS_Goods.DataSet['NDS'];
```

```
    if DataM.DS_Goods.DataSet.FieldByName('Total').IsNull
    then
      FieldValues['Ost']:=0
    else
      FieldValues['Ost']:=DataM.DS_Goods.DataSet['Total'];

    FormCntInput:=TFormCntInput.Create(nil);

    ok:=FormCntInput.ShowModal=mrok;

            if ok Then
            begin
                  Post;
      edit;
      Post;
      RecalcStat;
{      Edit;
      post;}
    end
    else
      Cancel;

    if ((FieldByName('Count').IsNull) or (FieldByName('Count').Value=0)) and
    ((FieldByName('Count2').IsNull) or (FieldByName('Count2').Value=0)) then
     DM_Local.VQ_OrderL.Delete;
    FormCntInput.Free;
  end;

end;

procedure EditPos();

begin

  with DM_Local.VQ_OrderL do
  begin
    edit;
    //FieldValues['Ost']:=DataM.DB_Goods['Total'];
    FormCntInput:=TFormCntInput.Create(nil);
    try
      case FormCntInput.ShowModal of
        mrok:Post;
        mrCancel:Cancel;
      end;
      except

    end;
    FormCntInput.Free;
  end;


end;

function GetFileVersionL(const AFileName: string): Cardinal;
var
  FileName: string;
  InfoSize, Wnd: DWORD;
  VerBuf: Pointer;
  FI: PVSFixedFileInfo;
  VerSize: DWORD;
begin
```

```
  Result := Cardinal(-1);
  // GetFileVersionInfo modifies the filename parameter data while parsing.
  // Copy the string const into a local variable to create a writeable copy.
  FileName := AFileName;
  UniqueString(FileName);
  InfoSize := GetFileVersionInfoSize(PChar(FileName), Wnd);
  if InfoSize <> 0 then
  begin
    GetMem(VerBuf, InfoSize);
    try
      if GetFileVersionInfo(PChar(FileName), Wnd, InfoSize, VerBuf) then
        if VerQueryValue(VerBuf, '\', Pointer(FI), VerSize) then
          Result:= FI.dwFileVersionLS;
    finally
      FreeMem(VerBuf);
    end;
  end;
end;

function getFileVersionStr(fn:string):String;
var
ver:Cardinal;
cr:packed record
w1,w2:word
end;
vers,v1,v2,v3,v4:string;

begin
  ver:=getFileVersion(fn);
  dword(cr):=ver;
  v1:=IntToStr(cr.w2);
  v2:=IntToStr(cr.w1);

  ver:=GetFileVersionL(fn);
  dword(cr):=ver;
  v3:=IntToStr(cr.w2);
  v4:=IntToStr(cr.w1);

  vers:=v1+'.'+v2+'.'+v3+'.'+v4;

result:=vers;

end;

function getFileVersion64(fn:string):INT64;
var
ver:Cardinal;

MR:packed record
D1,D2:LongWord
END;

cr1,CR2:packed record
w1,w2:Word;
end;


begin
  ver:=getFileVersion(fn);
  dword(cr1):=ver;
```

```
  ver:=GetFileVersionL(fn);
  dword(cr2):=ver;

MR.D1:=dword(cr2);
MR.D2:=dword(cr1);

RESULT:=INT64(MR);

end;

end.
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “Розробка програмного сервісу аптечного складу”

Обсяг пояснювальної записки: 65 аркушів

Дата закінчення "17" червня 2022 р.

Підпис: _____