Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерні науки_____
(повна назва)

Кафедра _____Програмна інженерія_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти_____перший (бакалаврський)_____

__Програмне забезпечення для мережі роздрібної торгівлі в курортній зоні_

_____
(тема)

Виконав: студент 4 курсу, групи ПЗПІпз–18-1

_____Карпенко Д.М._____
(прізвище, ініціали)

Спеціальність 121 - Інженерія програмного__

_____забезпечення_____
(код і повна назва спеціальності)

Освітня програма Програмна інженерія_____
(повна назва освітньої програми)

Керівник ___ст.викл.Козел Н.Б._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри. проф. _____ З. В. Дудар
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерні науки_____

Кафедра_____Програмна інженерія_____

Рівень вищої освіти____перший (бакалаврський)_____

Спеціальність _____121 – Інженерія програмного забезпечення_____

(код і повна назва)

Освітня програма _____Програмна інженерія_____

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«___»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ (ПРОЄКТ)

студентові ____*Карпенко Денису Михайловичу*_____

(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) *Програмне забезпечення для мережі роздрібної торгівлі в курортній зоні*_____

затверджена наказом по університету від «15»_____травня____ 2020р. № 58 Стз

2. Термін подання студентом роботи до екзаменаційної комісії «15» червня 2020р.

3. Вихідні дані до роботи *розробити програмне забезпечення для мережі роздрібної торгівлі у курортній зоні, середовище об'єктно – орієнтованого проектування та розробки MS Visual Studio 2019, мова програмування C#, СУБД MS SQL Server Local DB, Windows*

4. Перелік питань, що потрібно опрацювати в роботі *вступ, аналіз предметної області та постановка задачі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, опис роботи ПЗ, тестування ПЗ. Додатки: а) приклади програмних кодів; б) слайди презентації; в) електронні матеріали до проекту на CD.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	27.04.2020	виконано
2	Формування вимог до системи	04.05.2020	виконано
3	Проектування системи	11.05.2020	виконано
4	Розробка програмного забезпечення	18.05.2020	виконано
5	Тестування системи	20.05.2020	виконано
6	Підготовка пояснювальної записки	25.05.2020	виконано
7	Підготовка презентації та доповіді	25.05.2020	виконано
8	Перевірка на плагіат	05.06.2020	виконано
9	Нормоконтроль	05.06.2020	виконано
10	Рецензування	09.06.2020	виконано
11	Занесення диплома в електронний архів	12.06.2020	виконано
12	Попередній захист	12.06.2020	виконано
13	Допуск до захисту у зав.кафедри	12.06.2020	виконано

Дата видачі завдання «21» квітня 2020р.

Студент _____ Карпенко Д.М.
(підпис)

Керівник роботи (проекту) _____ ст. викл. Козел Н.Б.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи бакалавра, _62_ стор., _25_ рис., ___3__ табл., __12__ джерел.

БАЗА ДАНИХ, БІЗНЕС, БУХГАЛТЕРСКИЙ ОБЛІК, ІНФОРМАЦІЙНА СИСТЕМА, ТРОГІВЛЯ, C#, WINDOWS FORMS, VISUAL STUDIO 2019.

Об’єктом дослідження є сучасні технології, методи та засоби створення програмного забезпечення інформаційних систем для ведення обліку товарів.

Мета розробки – створити додаток інформаційну систему для оптимізації ведення торгівлі у курортній зоні та ведення обліку товарів.

Метод рішення – платформа .NET, Microsoft Visual Studio 2019, мова C#, Framework 4.8., Windows Forms.

У результаті розробки створено додаток, що дозволяє вести облік основних груп товарів даного підприємства. Додаток розміщується на персональному комп’ютері користувача та використовує локальну базу даних, що дозволяє значно прискорити роботу підприємства.

Explanatory note to the bachelor's attestation work, _62_ pages, _25_ figures, ___3__ tables, __12__ sources.

DATABASE, BUSINESS, ACCOUNTING, TRADE, INFORMATION SYSTEM, C #, VISUAL STUDIO 2019, WINDOWS FORMS.

The object of research is modern technologies, methods and means of creating software information systems for goods accounting.

The purpose of the development is to create an application information system to optimize trade in the resort area and accounting for goods.

Solution method – .NET platform, Microsoft Visual Studio 2019, C # language, Framework 4.8., Windows Forms.

As a result of the work the application which allows to keep the account of the main groups of the goods of the given enterprise was created.

ЗМІСТ

ВСТУП

За останні декілька десятиліть персональні комп’ютери стали невід’ємною частиною життя більшості людей. У наші часи комп’ютер з легкістю заміняє телевізор, радіо, аудіо та відео програвачі, телефони та ще безліч різноманітної техніки. Дуже часто комп’ютери виконують різноманітні автоматизовані дії, які раніше люди були вимушені робити власноруч. То ж зовсім не дивно, що з часом зростають потреби користувачів до обчислювальної техніки, зростає кількість та складність програмного забезпечення, з’являються нові технології, задачі, а також шляхи їх вирішення та впровадження. У наші часи, навряд чи ви знайдете фірму або підприємство, які ведуть облік використовуючи лише зошит то олівець. Сучасні проблеми потребують сучасних рішень. Сучасні умови праці в сфері комерції та обслуговування клієнтів вимагають ведення електронного обліку товарів, послуг, співробітників, клієнтів, різноманітних транзакцій, іншими словами – бази даних. Таким чином, у галузях бізнесу, економіки, виробництва та обліку персональні комп’ютери знайшли своє застосування.

Одним з найрозповсюджених видів програмного забезпечення у цих галузях є інформаційні системи. Вони значно спрощують та прискорюють роботу підприємств, та дозволяють отримувати та надавати актуальну інформацію без особливих зусиль. Інформаційні системи більш надійні, аніж паперові носії інформації. Переважна більшість організацій та фірм використовують електронні базі даних в якості засобу для збереження інформації. Застосування баз даних дуже розповсюджене, а їх розробка є досить об’ємним та складним процесом. За допомогою електронних баз даних доступ до інформації, а також її збереження, модифікування та видалення стає дійсно швидким. Бази даних можна підключити до веб-сайту для того щоб кожен бажаючий міг не тільки ознайомитися з наявністю товарів або послуг, але ще й дізнатися ціну, кількість, вартість доставки та інші характеристики.

Географічно Україна розташована на березі Чорного та Азовського моря. Морські курорти України завжди користувалися популярністю як українців, так і туристів із-за кордону. На території азовського узбережжя розвинута досить широка інфраструктура, що забезпечує комфортний відпочинок громадян. Це зумовлено великим потоком туристів, який за останні роки значно збільшився. Побудовано велика кількість готелів, баз відпочинку, розважальних закладів.

Також розвивається й торгівля. Як відомо, попит породжує пропозицію, тож у приморських зонах формуються торгівельні майданчики, що пропонують туристам широкий асортимент товарів та послуг для комфортного відпочинку. А для ефективної роботи та комерційної діяльності будь якого підприємства потрібна сучасна база даних та програмне застосування, функціонал якого дозволяв би комфортне користування цією базою. Іншими словами – для мережі магазинів роздрібної торгівлі потрібна інформаційна система. Метою даної атестаційної роботи є проектування та розробка саме такого програмного продукту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Предметна галузь інформаційної системи – це матеріальна система або система, що характеризує елементи матеріального світу, інформація про які оброблюється та зберігається. Предметна галузь розглядається як певна сукупність реальних об’єктів та зав’язків поміж ними [1].

Магазини роздрібної торгівлі почали своє існування на косі Арабатська стрілка в Херсонській області у 2003 році. Перша торгова точка була розташована біля геотермального гарячого джерела, куди прибувають люди для відпочинку, оздоровлення та профілактики незалежно від пори року. Магазин спеціалізувався, на продажі товарів для відпочинку. З тих пір мережа розширялася, торгівельні точки будувалися переважно на приморських територіях, біля пляжів та баз відпочинку. Розширявся асортимент товарів, додавалися нові категорії відповідно до потреб людей.

Мережа магазинів курортної зоні спеціалізується на реалізації товарів для відпочинку. В наявності великий асортимент літнього одягу, взуття, головних уборів, косметичних засобів, аксесуарів та сувенірної продукції.

Клієнти магазину – фізичні особи, які можуть зробити як одиничну покупку для особистого користування, так і закупку певних категорій товарів з метою подальшої реалізації.

Кожна одиниця товару має певні властивості та характеристики, такі як назва, ціна, виробник, категорія товару, кількість та наявність товару на складі. Власнику магазина необхідно володіти та оперувати всією вищезазначеною інформацією про товари у наявності для забезпечення продуктивної роботи із клієнтами, організації своєчасної поставки товарів на торгівельні точки та проведення закупок потрібних позицій. Робота дуже трудомістка, адже ведеться облік дуже великого обсягу інформації, яка зберігається у декількох журналах.

Для забезпечення цього необхідна сучасна база даних та додаток, за допомогою якого була б забезпечена робота з базою.

Для ведення оперативного обліку товарів, робітників та транзакцій магазину потребується інформаційна система, що заснована на сучасній базі даних. Наявність такої бази даних значно заощадить час підприємців при веденні обліку товарів, збереженню інформації про робітників, транзакцій та торгівельних точок, а значить підвищить ефективність роботи мережі магазинів роздрібної торгівлі у курортній зоні.

1.2 Аналіз існуючих аналогів

Існують різноманітні системи для обліку продажів, товарів та послуг у магазині. Але більшість програмних продуктів, що вирішують дану задачу коштують досить дорого та деякі з них недостатньо задовольняють потреби конкретного підприємства. Розглянемо деякі із них.

1.2.1 1С: Підприємство

Фірма "1С" випускає тиражні програмні рішення, призначені для автоматизації типових завдань обліку та управління в комерційних підприємствах реального сектора і бюджетних організаціях. У кожному програмному продукті поєднується використання стандартних рішень (загальних для всіх або декількох програм) і максимальне врахування специфіки завдання конкретної галузі або роду діяльності підприємства [2].

Система програм "1С: Підприємство" призначена для вирішення широкого спектра завдань автоматизації обліку та управління, що стоять перед сучасними підприємствами, які динамічно розвиваються.

"1С: Підприємство" являє собою систему прикладних рішень, побудованих за єдиними принципами і на єдиній технологічній платформі. Керівник може вибрати рішення, яке відповідає актуальним потребам підприємства і буде в подальшому розвиватися в міру росту підприємства або розширення завдань автоматизації (рисунок 1.1).

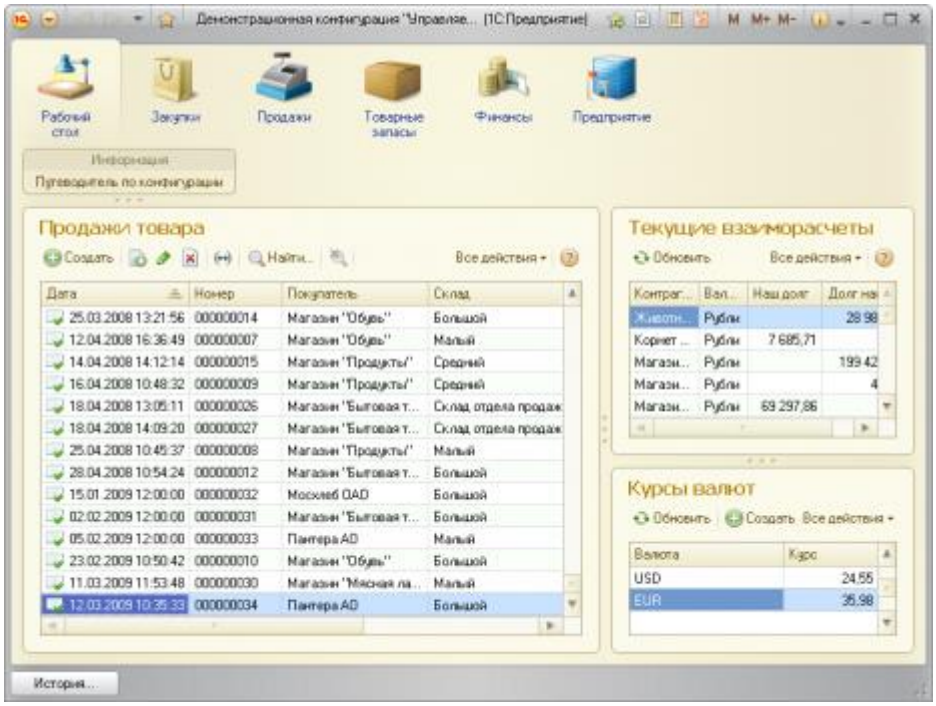

Рисунок 1.1 – Головне вікно програми 1С: Підприємство

Технологічна платформа «1С: Підприємство» являє собою програмну оболонку над базою даних. Використовуються бази на основі DBF-файлів в 7.7, власний формат 1CD з версії 8.0 або СУБД Microsoft SQL Server на будь-який з цих версій . Крім того, з версії 8.1 зберігання даних можливо в PostgreSQL і IBM DB2, а з версії 8.2 додалася і Oracle. Платформа має свій внутрішній мову програмування, що забезпечує, крім доступу до даних, можливість взаємодії з іншими програмами за допомогою OLE і DDE, в версіях 7.7, 8.0 і 8.1 – за допомогою COM-з'єднання [3].

Але, не зважаючи на те, що «1С: Підприємство» в наші часи є найпоширенішою програмою для автоматизації бухгалтерського та управлінського обліку в Україні, вона має ряд недоліків. Розглянемо головні.

– продукт орієнтований в першу чергу на автоматизацію задач бухгалтерського та податкового обліку. Для підтримки роботи об'єктів роздрібної торгівлі, сфери послуг, спец. об'єктів він не підходить. Багато підприємств, що спеціалізуються на розробці під 1С, створюють свої конфігурації, орієнтовані на різні сфери бізнесу. Ціна таких продуктів на порядок вище, а функціонал необхідно буде допрацьовувати, що потребує додаткових витрат на налаштування, підтримку та інше;

– на кожному підприємстві використовується унікальне рішення 1С, що реалізоване не налаштуванням конфігурації, а запрограмоване на мові програмування 1С, що вбудована у платформу. Іншими словами – купуючи будь яке рішення на основі 1С, ви маєте лише платформу 1С, яку будете змушені налаштовувати орієнтуючись на власні вимоги. А такі послуги – це додаткові витрати, які, як правило – незаплановані. І якщо для великих підприємств – це доволі звичайна практика, то для невеликих підприємств – це трудомісткий та затратний процес, що потребує затрат часу та фінансів;

– налаштування, впровадження та запуск здійснюються кваліфікованим програмістом . Реальну вартість продукту можна оцінити лише тоді, коли завершиться обробка коду модулів програми;

– для отримання оновлень продукту користувач вимушений придбати платну підписку;

– завдяки широкій розповсюдженості продукту, у мережі інтернет існують тисячі способів для отримання незаконного доступу до даних, що використовує 1С. Кількість хакерських прийомів для отримання даних постійно збільшується та оновлюється.

1.2.2 Мій склад

Мій склад – спеціальна система для управління торгівлею, оптимізована для мобільних пристроїв Android і iOS. Портал зацікавить власників, як малого, так і середнього бізнесу. Особливо буде корисним тим, у кого є інтернет-магазин і склад з продукцією, який необхідно регулярно оновлювати. Сервіс допоможе керувати кількома магазинами, якщо вони знаходяться в різних кінцях міста. З його допомогою можна відстежувати кількість товару, проводити оновлення продукції і призначати знижки (рисунок 1.2).

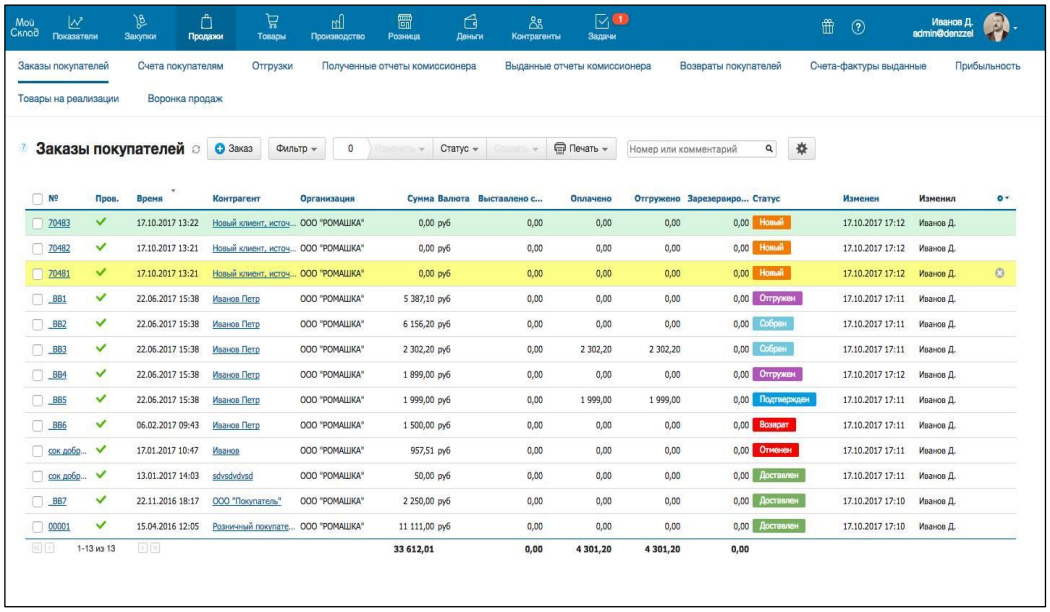

Рисунок 1.2 – Головне вікно програми «Мой склад»

Для автоматизації магазину буде потрібно ПК або ноутбук, USB-сканер штрих-кодів, фіскальний реєстратор і доступ до мережі інтернет для отримання статистичних даних в режимі реального часу. Щоб запустити систему вводиться URL магазину, логін і пароль адміністратора. Так як на порталі синхронізація двостороння, то можна самостійно вибирати категорії для цього процесу. У розділі «Звіти» передбачено створення експорту вільних залишків і цін. За додаткову плату у адміністратора є можливість реєструвати необмежену кількість

менеджерів. Інструменти для настройки прав кожного нового користувача дозволяють відкривати допуск до читання, редагування або створення абсолютно будь-якої операції. CRM система сприяє відстеженню списку покупців і рівня продажів, оформлення поштової розсилки клієнтам, а також створення статусів [4].

Програма «Мой склад» має деякі недоліки, серед яких:

– складний та перевантажений інтерфейс;

– відносно повільна швидкість роботи;

– неможливість налаштування інтерфейсу;

– для роботи з програмою необхідне залучення кваліфікованого спеціаліста.

Критерії, за якими ми проводили порівняння продуктів – аналогів для наглядності наведено у таблиці 1.

Таблиця 1 – Порівняння продуктів-аналогів

	1С:підприємство	Мій склад	Даний продукт
Можливість ведення обліку товарів	+	+	+
Наявність додаткового функціоналу(відправка звітів, сплата податків тощо)	+	+	–
Можливість подальшого розвитку ПЗ	+	+	+
Популярність ПЗ	+	+	–
Можливість використання ПЗ без професійних навичок	–	–	+
Налаштування без залучення фахівця	–	–	+
Продукт безкоштовний	–	–	+

Розглянуті вище продукти є дуже потужними, багатофункціональними інформаційними системами, але вони мають ряд недоліків, які не задовольняють потребам конкретного підприємства. Серед головних відмітимо:

– висока вартість придбання, інтеграції та супроводження;

– необхідність залучення кваліфікованих спеціалістів для роботи з програмою;

– значна частина функціоналу програм просто не потрібна у робочому процесі даного підприємства.

Саме тому було прийняте рішення про розробку власного програмного забезпечення, що дозволить значно скоротити затрати робочого часу та значно підвисити ефективність роботи мережі магазинів роздрібної торгівлі у курортній зоні.

1.3 Постановка задачі

В ході атестаційної роботи необхідно спроектувати та розробити десктопний додаток з базою даних, який відповідає сучасним розробкам інформаційних систем, буде мати зручний та зрозумілий інтерфейс, орієнтований на користувача, що не має наукового ступеня у програмуванні. Інформаційна система має зберігати та оброблювати інформацію щодо товарів в наявності, атрибутів товару, анкети співробітників, дані про торгівельні точки та продажі. Кінцевим користувачем інформаційної системи є персонал та керівництво магазину.

Кінцевий користувач може звернутися до інформаційної системи з такими запитами:

– детальна інформація про конкретний товар;

– детальна інформація про конкретну транзакцію (продаж);

– детальна інформація про певну торгівельну точку;

– детальна інформація про співробітника.

Програмний продукт повинен виконувати такі функції:

– зберігання введеної користувачем інформації щодо товарів;

– зберігання введеної користувачем інформації щодо транзакцій;

– зберігання введеної користувачем інформації щодо співробітників;

– відображення повної інформації щодо товарів;

– пошук товарів за критеріями;

– відображення стану бази даних за іншими таблицями;

– видалення інформації щодо товарів;

– видалення інформації щодо транзакцій;

– видалення інформації щодо робітників;

– оновлення інформації щодо товарів;

– оновлення інформації щодо транзакцій;

– оновлення інформації щодо робітників.

Інформаційна система, як і будь який інший сучасний програмний продукт повинна мати такі властивості:

– зрозумілий та логічно оформлений інтерфейс;

– відмовостійкість;

– безпечність;

– адаптованість.

Для реалізації проекту с вищезазначеною функціональністю була обрана мова програмування C#, фреймворк .NET Framework, та технологія Windows Forms.

C# – це сучасна об’єктно орієнтована мова програмування, яка відноситься до сімейства C подібних мов програмування і використовується для розробки найрізноманітніших проектів, у тому числі і десктопні додатки під ОС Windows, призначених для виконання у середовищі .NET Framework [5].

Розроблюваний продукт повинен мати зручний, логічний та зрозумілий інтерфейс так як планується, що програмним забезпеченням буде користуватися людина, що не має наукової ступені у програмуванні чи комп’ютерних науках.

Для створення інтерфейсу обрана технологія Windows Forms. Windows Forms це інтерфейс програмування додатків, що є частиною Microsoft .NET Framework.

В якості бази даних обрана Microsoft Sql Server 2012 Local DB. Передбачається, що працювати з програмним продуктом буде одна людина в певний період часу, тому саме локальна база даних найкраще підходить до даного проекту.

При розробці даного проекту враховуються такі припущення:

– потенційний користувач має навички роботи з персональним комп’ютером;

– користувач має потребу у зберіганні даних;

– програмний продукт працює без перебоїв.

В даному розділі була проведена постановка задачі.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Сукупність тверджень щодо властивостей, атрибутів або якостей програмної системи, реалізація якої проводиться є вимогами до програмної системи. Вони створюються у процесі розробки програмного забезпечення в результаті аналізу вимог та можуть бути представлені у вигляді графічних моделей або текстових стверджень.

2.1 Функціональні вимоги

Функціональні вимоги до розроблюваної системи були сформовані у вигляді пропозицій користувача. В рамках цього формату маємо такі користувацькі вимоги:

– як користувач додатку я хочу зберігати дані про товари, транзакції, співробітників;

– як користувач, я хочу мати можливість перегляду даних про товари, транзакції та співробітників;

– як користувач, я повинен мати можливість внесення змін та оновлення даних про товари, транзакції, співробітників;

– як користувач, я повинен мати можливість видалення інформації про товари, транзакції, співробітників;

– як користувач, я повинен мати можливість пошуку товарів за критеріями.

Функціональні вимоги було сформовано.

2.2 Вимоги до розроблюваного додатку

Розроблюваний додаток має відповідати наступним функціональним вимогам:

– збереження даних про товари, транзакції, співробітників;

– оновлення даних про товари, транзакції, співробітників;

– видалення даних про товари, транзакції, співробітників;

– відображення детальної інформації про товари, транзакції, співробітників;

– забезпечення пошуку товарів за критеріями.

Серверна частина має задовольняти наступним функціональним вимогам:

– обробка інформації, яку вводить користувач у поля додатку;

– зв’язок за базою даних MS SQL Server Local Db;

– внесення змін у базу даних.

Вимоги до розроблюваного додатку було сформовано.

2.3 Вимоги до обладнання розробника

Для успішної розробки даного програмного забезпечення, розробник повинен мати:

– операційну систему Windows 7 або вище;

– Visual Studio 2019;

– Microsoft Sql Server 2012 Local DB;

– .NET Framework 4.8;

– Microsoft Office Word.

Вимоги до обладнання розробника було сформовано.

2.4 Вимоги до обладнання користувача

Інформаційна система, що розроблюється у даному проекті не потребує багато ресурсів користувацького персонального комп’ютеру. Розроблюване програмне забезпечення орієнтовано на те, що на складах підприємств, частіше за все, знаходиться не дуже потужне обладнання. Робоча станція, на якій буде виконуватися даний продукт повинна мати такі атрибути:

– операційну систему Windows 7;

– .NET Framework;

– Microsoft SQL Server 2012 Local DB.

Вимоги до обладнання користувача було сформовано.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проектування ПЗ

Для більш повного розуміння процесів у проекті наведемо декілька UML діаграм.

UML – це мова для візуалізації, специфікування, конструювання та документування артефактів програмних систем [6].

UML є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, званої UML-моделлю. UML є потужним, гнучким засобом моделювання, опис стандарту якого є відкритим для наступного вдосконалювання [7].

При проектуванні програмного продукту UML моделювання є необхідною частиною. Графічне представлення основних понять і процесів дає змогу розробниками краще зрозуміти суть розроблюваного продукту, що значно заощаджує час і ресурси.

3.1.1 Use case діаграма

Адміністратор БД займається операціями за даними. Він заносить нові дані в базу, оновлює існуючі, видаляє не потрібні. Продавець займається продажом товарів на торгівельній точці та за необхідністю подає заявки на ті позиції товару, що закінчуються.

На рисунку 3.1 представлена use case діаграма. Вона відображає відносини, що існують між акторами (адміністратор бази даних та продавець).

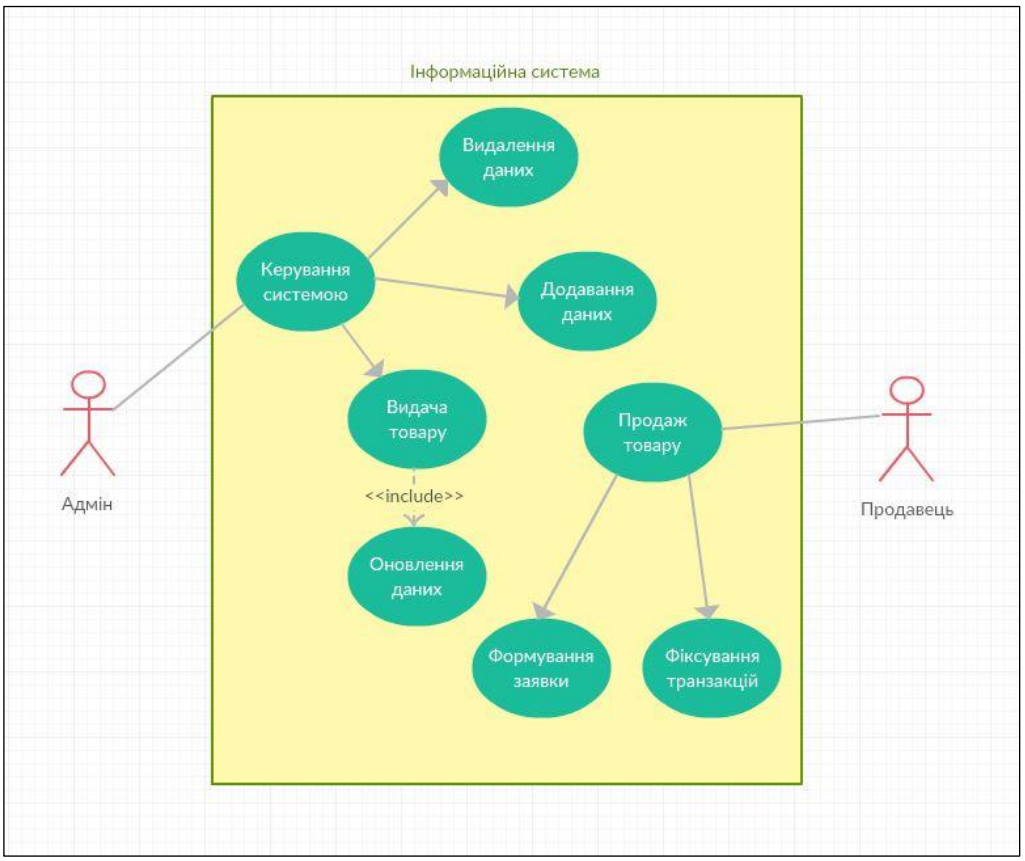

Рисунок 3.1 – Use case діаграма

Основні функції адміністратора є доволі зрозумілими та очевидними. Процес обробки заявки продавця більш детально розглянемо у наступних розділах.

3.1.2 Діаграма активності

Діаграма активності відображає діяльність, яка відбуваються всередині системи. Кожна наступна дія відбувається після завершення першої. Під діяльністю розуміється специфікація виконуваного поведінки у вигляді координованого послідовного і паралельного виконання підлеглих елементів – вкладених видів діяльності і окремих дій, з'єднаних між собою потоками, які йдуть від виходів одного вузла до входів іншого [8].

На рисунку 3.2 зображена діаграма активності.

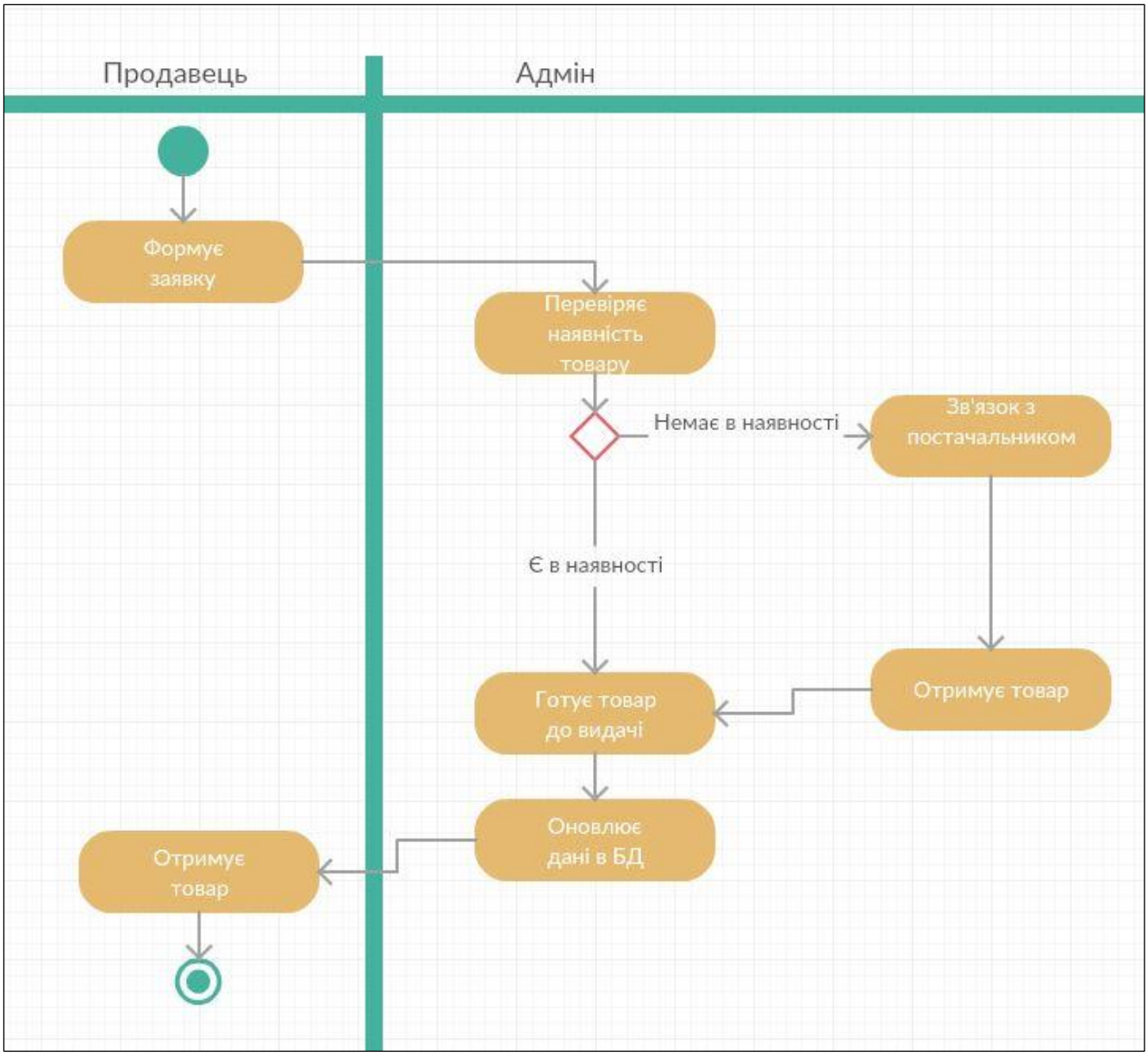

Рисунок3.2 – Діаграма активності

Продавець формує заявку, адміністратор БД перевіряє наявність товару на складі. Якщо товар є в наявності адміністратор готує поставку товару. Якщо товар відсутній – адміністратор зв’язується з постачальником товару для подальшої закупки.

Беручи до уваги той факт що курортний сезон є доволі коротким, іноді, у разі відсутності товару на складі, має сенс перекидання товару з однієї точки на іншу для більш оперативної поставки товару. Цей процес зображено на діаграмі активності (рисунок 3.3).

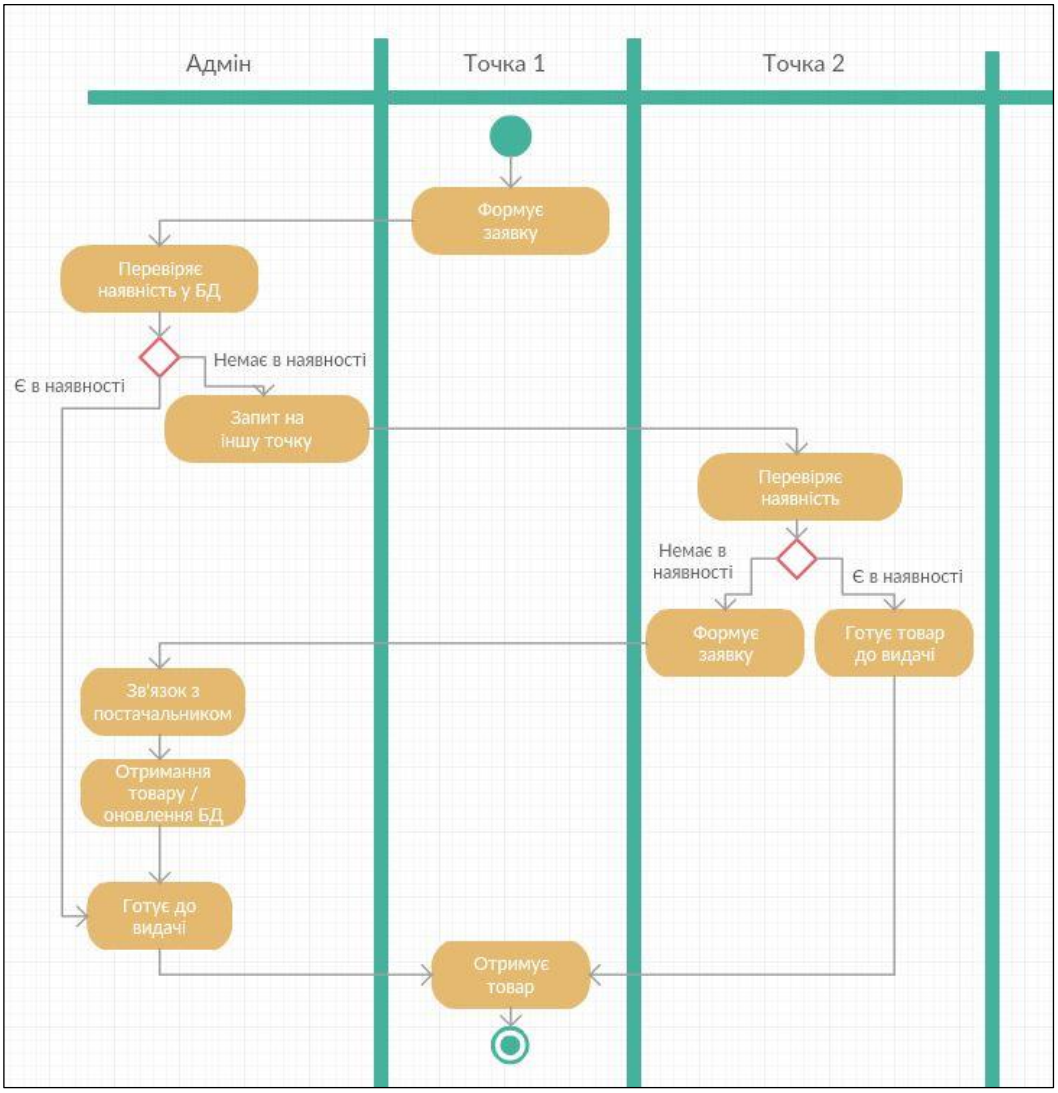

Рисунок 3.3 – Діаграма активності

У разі відсутності товару на складі адміністратор робить запит на інші точки продажу. І у випадку, якщо на точці товар мається у достатній кількості, відбувається перекидання товару з однієї точки на іншу.

В даному розділі було створено діаграми та проведено UML моделювання програмного продукту.

3.2 Проектування архітектури ПЗ

Даний проект буде складатися з двох основних компонентів: база даних та додаток для роботи із базою.

База даних, що проектується за своєю архітектурою має бути централізованою. Всі файли бази даних будуть зберігатися на одній машині. В якості СУБД потрібно використовувати Microsoft Sql Server Local DB.

Local DB – це спрощена версія ядра СУБД SQL Server Express, призначена для розробки програм. LocalDB запускається за запитом в призначеному для користувача режимі, тому налаштування не надто складні.

Для реалізації системи було обрано мову C# та Windows Forms. Основними перевагами яких є :

− об'єктно-орієнтований підхід;

− повноцінна компільована мова програмування C#;

− зручне середовище розробки [9].

3.3 Проектування структури зберігання даних

Найважливішим елементом будь-якої бази даних, незалежно від її подання, є структура даних, яка відображає можливі подання відомостей для зберігання, вибірки і обробки. Структури даних необхідні для роботи з документарними даними, не пов'язаними з комп'ютерною обробкою, і з даними, що подаються в автоматизованих системах [10].

У сенсі терміну "Структура даних" розуміється одиниця відомостей, що дозволяє зберігати і обробляти безліч однотипних і (або) логічно пов'язаних даних.

3.3.1 Концептуальна модель даних

Концептуальне проектування передбачає побудову семантичної моделі предметної галузі, іншими словами інформаційну модель найбільш високого рівня абстракції. Ця модель створюється без орієнтації на будь яку конкретну СУБД та модель даних.

Основними об’єктами інформаційної системи є товар, співробітники та продажі. Робітники продають товар та фіксують продажі. На основі цього створимо концептуальну діаграму (рисунок 3.4)

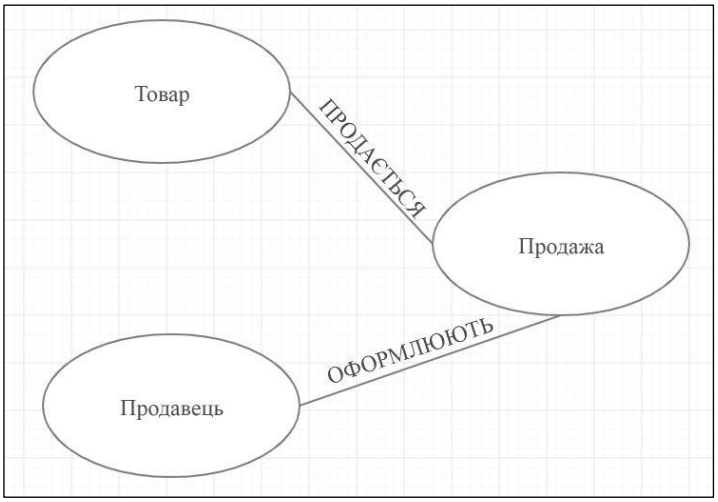

Рисунок 3.4 – Концептуальна модель

Після виявлення основних концептів потрібно побудувати логічну модель бази даних.

3.3.2 Логічна модель даних

Логічна модель даних, або логічна схема – модель даних конкретної предметної області, виражена незалежно від конкретного продукту керування

базами даних або технології зберігання (фізична модель даних), але в термінах структур даних, таких як реляційні таблиці та колонки, об'єктно-орієнтовані класи чи теги XML. Вона є протилежністю концептуальній моделі даних, яка описує семантику організації без посилання на технологію.

Логічні моделі даних подають абстрактну структуру області інформації. Вони часто мають схематичний характер і найтиповіше використовуються у бізнес-процесах, які прагнуть захопити речі, що мають важливе для організації значення, та як вони відносяться одна до одної [11].

Розроблювана в даному проекті інформаційна система має зберігати та оброблювати інформацію щодо товарів, співробітників, продажів, а також інформацію про категорії товарів та точки продажу.

На основі цього визначимо такі відношення з атрибутами:

а) категорія товару (Categories) має атрибути:

1)Category_id (ідентифікатор категорії);

2)Name_c (назва категорії).

б) торгова точка (Outlets) має атрибути:

1)Outlet_id (ідентифікатор торгівельної точки);

2)Address_o (адреса).

в) товари (Products) має атрибути:

1)Product_id (ідентифікатор товару);

2)Category_id (ідентифікатор категорії);

3)Name_p (назва товару);

4)Price_p (ціна товару);

5)Quantity_p(кількість товару).

г) співробітники (Workers) має атрибути:

1)Worker_id (ідентифікатор працівника);

2)Outlet_id (ідентифікатор торгівельної точки);

3)Name_w (ім’я працівника);

4)Phone_w (телефон працівника).

д) продажі (Sales) має атрибути:

1)Sale_id (ідентифікатор продажі);

2)Product_id (ідентифікатор товару);

3)Worker_id (ідентифікатор працівника);

4) Date_s (дата продажі);

5)Quantity_s(кількість товару).

Однозначно ідентифікуємо сутності та виділимо первинні ключі:

– сутність Categories: первинний ключ Category_id;

– сутність Products: первинний ключ Product_id;

– сутність Outlets: первинний ключ Outlet_id;

– сутність Workers: первинний ключ Worker_id;

– сутність Sales: первинний ключ Sale_id;

Тепер потрібно зв’язати відношення по зовнішнім ключам:

– сутності Categories та Products пов’язані між собою по зовнішньому ключу Category_id. І так як в одній категорії може бути декілька товарів і один товар може мати лише одну категорію – то це зв’язок «один до багатьох»;

– сутності Products та Sales пов’язані між собою по зовнішньому ключу Product_id. І так як одна позиція товару може бути продана декілька раз – це зв’язок «один до багатьох»;

– сутності Outlets та Workers пов’язані між собою по зовнішньому ключу Outlet_id. І так як на одній торговій точці може працювати декілька продавців – це зв’язок «один до багатьох»;

– сутності Workers та Sales пов’язані між собою по зовнішньому ключу Worker_id. І так як один продавець може оформити декілька продажів – це зв’язок «один до багатьох».

На основі визначених відносин ми можемо спостерігати наявність зв’язку між таблицями. Зобразимо ці зв’язки у вигляді діаграми entity – relationship.

Entity–relationship діаграма – це модель даних, що дозволяє описувати концептуальні схеми предметної області. Ця модель використовується при високорівневому проектуванні баз даних. З її допомогою ми виділимо ключові сутності і позначимо зв’язки , які можуть встановлюватися між цими сутностями.

Під час проектування баз даних відбувається перетворення ER- моделі в конкретну схему бази даних на основі обраної моделі бази даних. Основні компоненти діаграми entity – relationship – це сутності, атрибути та зв’язки.

Кожна сутність є множиною подібних індивідуальних об’єктів, що є екземплярами. Кожен екземпляр індивідуальний та повинен відрізнятися від решти екземплярів. Атрибут демонструє певну властивість об’єкта.

Діаграму entity – relationship зображено на рисунку 3.5.

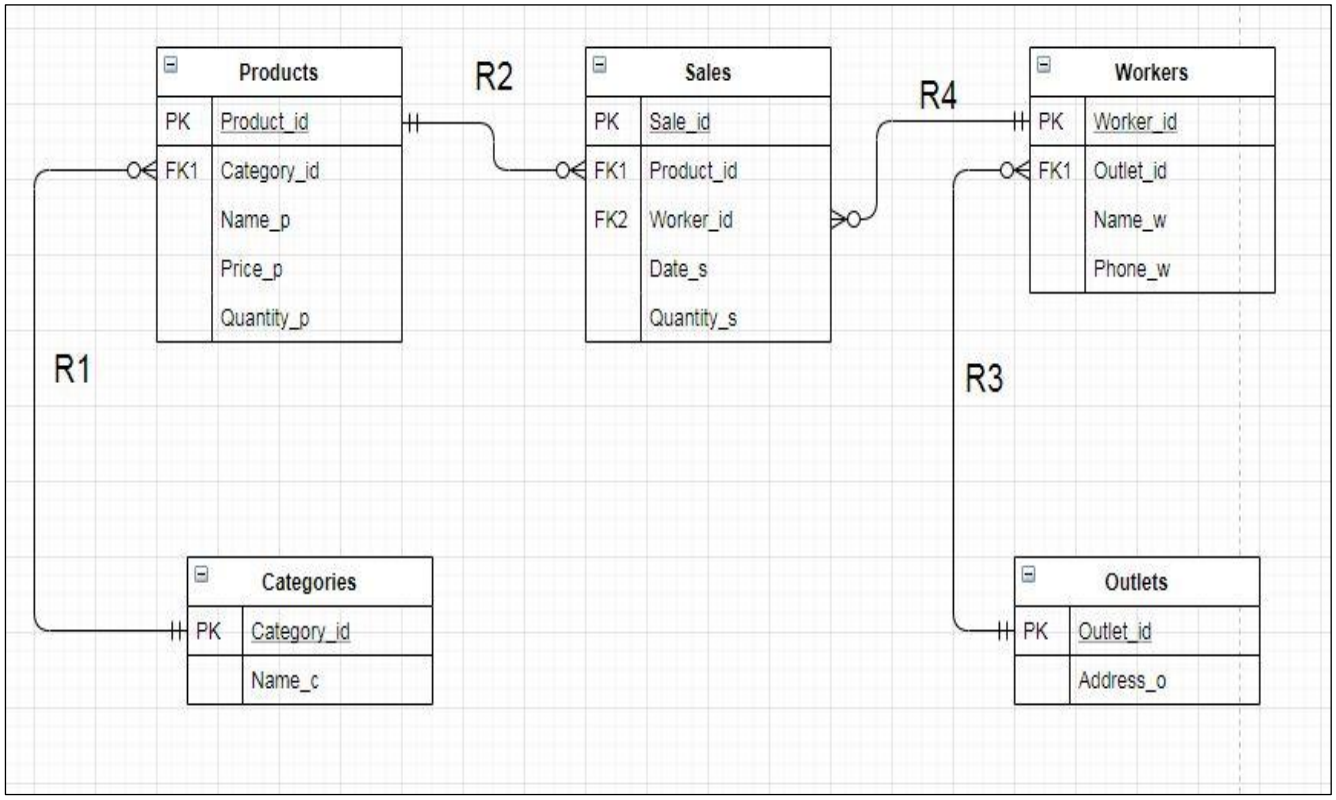

Рисунок 3.5 – Діаграма entity – relationship

Сутність – це реальний, або уявний об'єкт, інформацію про який необхідно зберігати в базі даних. На діаграмі ER-моделі сутність зображується у вигляді прямокутника, що містить ім'я сутності.

Зв'язок – відображається графічно на діаграмі, це є асоціація між двома сутностями

В цьому розділі було створено схему бази даних проекту. Наступним кроком буде нормалізація відношень бази даних.

3.3.3 Нормалізація відношень бази даних

Нормальна форма є властивістю відношень у реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може привести до логічних помилкових результатів вибірки або збору даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення.

Нормалізація бази даних потрібна для того, щоб не описувати одні й ті самі дані у базі більше одного разу. Як правило, щоб уникнути надмірності бази даних, достатньо привести її до третьої нормальної форми.

Проаналізуємо відношення з урахуванням їх первинних ключів та функціональних залежностей для доказу нормалізації.

У відношенні «Categories» атрибутами є наступні поля: Category_id, name_c. Серед них немає атрибутів що повторюються та атрибутів з однаковою по змісту інформацією. Значення атрибутів атомарні. Поле «Category_id» однозначно визначає кожний кортеж таблиці. Таким чином визначаємо, що відношення «Categories» знаходиться у першій нормальній формі.

У відношенні «Categories» первинний ключ складається з єдиного атрибуту «Category_id». Це означає, що відношення знаходиться у другій нормальній формі.

Всі неключові атрибути залежать від первинного ключа у цілому. Усі неключові атрибути не залежать від інших неключових атрибутів, а залежать тільки від первинного ключа. Таким чином ми довели, що відношення «Categories» знаходиться у третій нормальній формі.

У відношенні «Products» атрибутами є наступні поля: Product_id, Category_id, Name_p, Price_p, Quantity_p. Серед них немає атрибутів що повторюються та атрибутів з однаковою по змісту інформацією. Значення атрибутів атомарні. Поле «Product_id» однозначно визначає кожний кортеж таблиці. Таким чином визначаємо, що відношення «Products» знаходиться у першій нормальній формі.

У відношенні «Products» первинний ключ складається з єдиного атрибуту «Product_id». Це означає, що відношення знаходиться у другій нормальній формі.

Всі неключові атрибути залежать від первинного ключа у цілому. Усі неключові атрибути не залежать від інших неключових атрибутів, а залежать тільки від первинного ключа. Таким чином ми довели, що відношення «Products» знаходиться у третій нормальній формі.

У відношенні «Workers» атрибутами є наступні поля: Worker_id, Outlet_id, Name_w, Phone_w. Серед них немає атрибутів що повторюються та атрибутів з однаковою по змісту інформацією. Значення атрибутів атомарні. Поле «Worker_id» однозначно визначає кожний кортеж таблиці. Таким чином визначаємо, що відношення «Workers» знаходиться у першій нормальній формі.

У відношенні «Workers» первинний ключ складається з єдиного атрибуту «Worker_id». Це означає, що відношення знаходиться у другій нормальній формі.

Всі неключові атрибути залежать від первинного ключа у цілому. Усі неключові атрибути не залежать від інших неключових атрибутів, а залежать тільки від первинного ключа. Таким чином ми довели, що відношення «Workers» знаходиться у третій нормальній формі.

У відношенні «Sales» атрибутами є наступні поля: Sale_id, Product_id, Worker_id, Date_s, Quantity_s Серед них немає атрибутів що повторюються та атрибутів з однаковою по змісту інформацією. Значення атрибутів атомарні. Поле «Sale_id» однозначно визначає кожний кортеж таблиці. Таким чином визначаємо, що відношення «Sales» знаходиться у першій нормальній формі.

У відношенні «Sales» первинний ключ складається з єдиного атрибуту «Sale_id». Це означає, що відношення знаходиться у другій нормальній формі.

Всі неключові атрибути залежать від первинного ключа у цілому. Усі неключові атрибути не залежать від інших неключових атрибутів, а залежать тільки від первинного ключа. Таким чином ми довели, що відношення «Sales» знаходиться у третій нормальній формі.

У відношенні «Outlets» атрибутами є наступні поля: Outlet_id, Address_o. Серед них немає атрибутів що повторюються та атрибутів з однаковою по змісту інформацією. Значення атрибутів атомарні. Поле «Outlet_id» однозначно визначає

кожний кортеж таблиці. Таким чином визначаємо, що відношення «Outlets» знаходиться у першій нормальній формі.

У відношенні «Outlets» первинний ключ складається з єдиного атрибуту «Outlet_id». Це означає, що відношення знаходиться у другій нормальній формі.

Всі неключові атрибути залежать від первинного ключа у цілому. Усі неключові атрибути не залежать від інших неключових атрибутів, а залежать тільки від первинного ключа. Таким чином ми довели, що відношення «Outlets» знаходиться у третій нормальній формі.

Таким чином було доведено, що база даних магазину роздрібної торгівлі знаходиться у третій нормальній формі. Завершуючи нормалізацію переконуємось що базі даних не власна надмірність та дублювання інформації.

3.3.4 Фізична модель даних

Створимо структуру бази даних у інтегрованому середовищі розробки Visual Studio 2019, використовуючи засоби MS SQL Server Express 2012 Local Db. У першу чергу додаємо відношення у наш проект та атрибути до них. Також необхідно визначити тип даних кожного атрибуту.

Таблиця Categories містить у собі дані про категорії товарів та має атрибути Category_id, Name_c. Category_id є первинним ключем відношення, він буде автоматично інкрементуватись. Типи даних, зовнішні ключі та скрипт створення відношення зображено на рисунку 3.6.

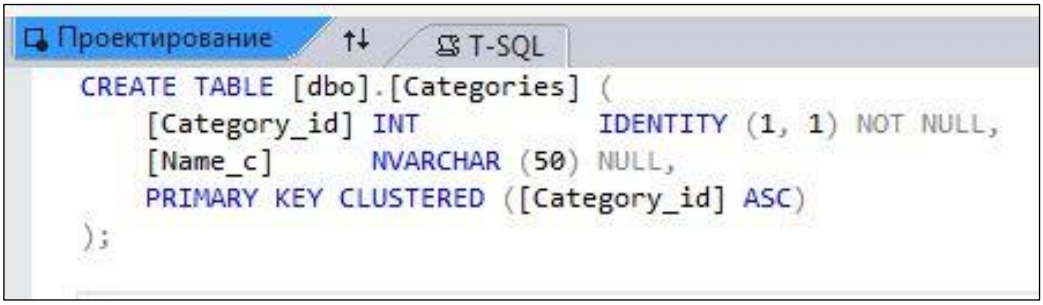

Рисунок 3.6 – Скрипт створення відношення Categories

Таблиця Outlets містить у собі дані про торгові точки та має атрибути Outlet_id, Address_o. Outlet_id є первинним ключем відношення, він буде автоматично інкрементуватись. Типи даних, зовнішні ключі та скрипт створення відношення зображено на рисунку 3.7.

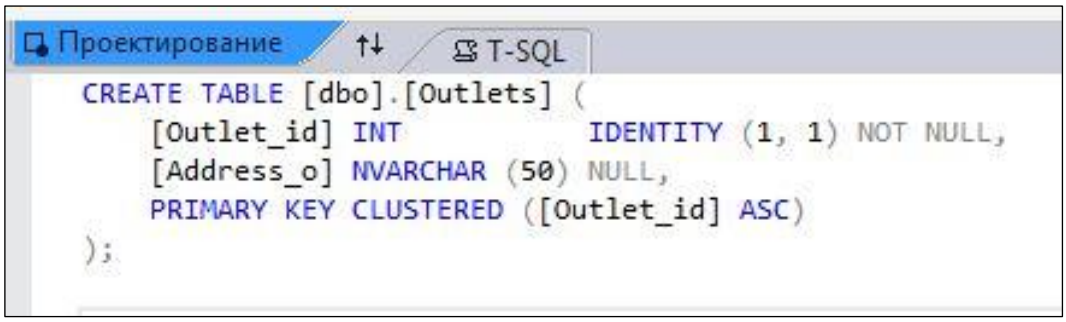

Рисунок 3.7 - Скрипт створення відношення Outlets

Таблиця Products містить у собі дані про товари та має атрибути Product_id, Category_id, Name_p, Price_p, Quantity_p. Product_id є первинним ключем відношення, він буде автоматично інкрементуватись. Типи даних, зовнішні ключі та скрипт створення відношення зображено на рисунку 3.8.

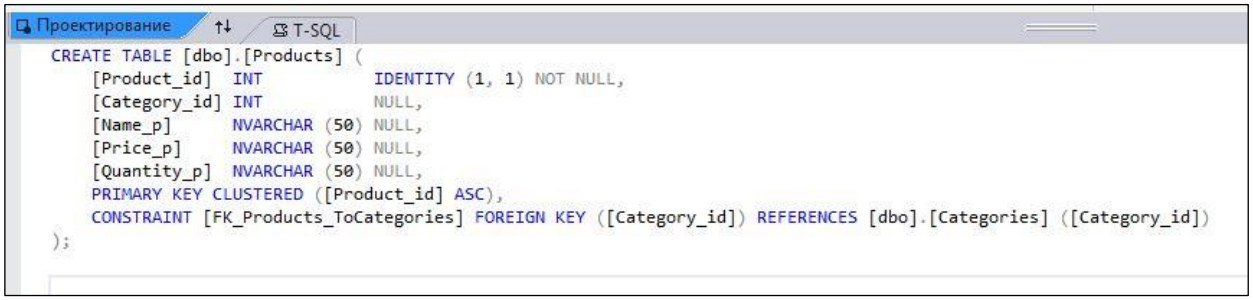

Рисунок 3.8 - Скрипт створення відношення Products

Таблиця Workers містить у собі дані про працівників та має атрибути Worker_id, Outlet_id, Name_w, Phone_w. Worker_id є первинним ключем відношення, він буде автоматично інкрементуватись. Типи даних, зовнішні ключі та скрипт створення відношення зображено на рисунку 3.9.

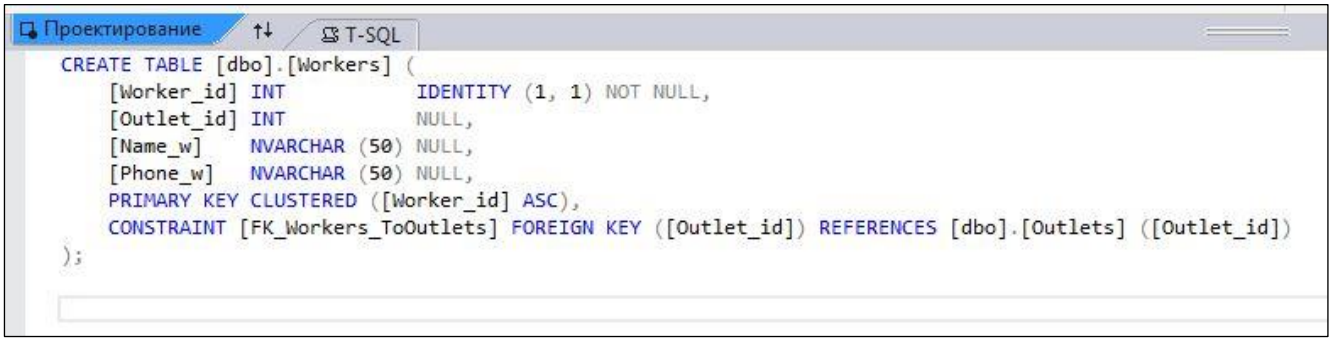

Рисунок 3.9 – Скрипт створення відношення Workers

Таблиця Sales містить у собі дані з продаж та має атрибути Sale_id, Product_id, Worker_id, Date_s, Quantity_s Sale_id є первинним ключем відношення, він буде автоматично інкрементуватись. Типи даних, зовнішні ключі та скрипт створення відношення зображено на рисунку 3.10.

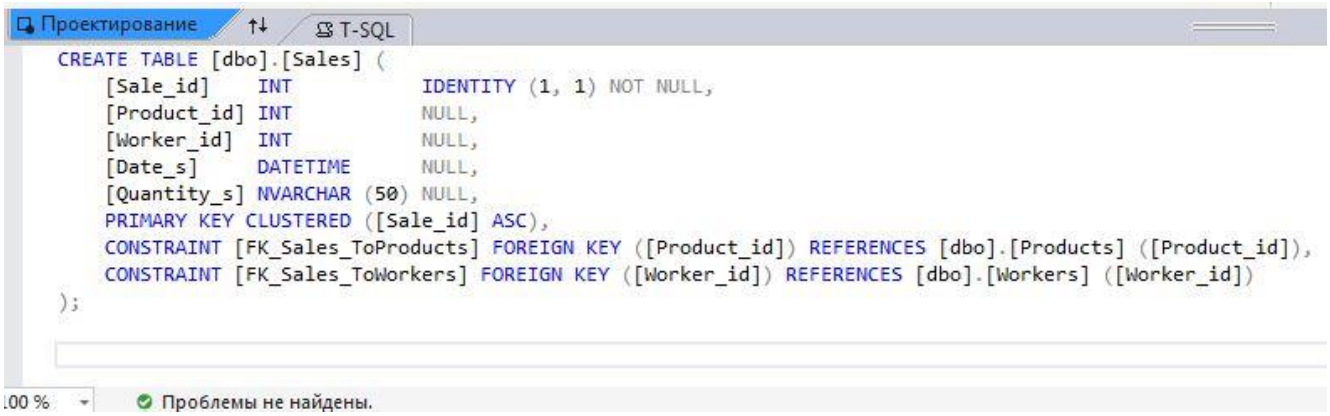

Рисунок 3.10 – Скрипт створення відношення Sales

На даному етапі фізичне моделювання бази даних завершено. Наступним кроком буде проектування користувацького інтерфейсу.

3.4 Проектування UI

Для створення користувацького інтерфейсу було обрано технологію Windows Forms.

Інтерфейс користувача, в галузі дизайну, взаємодії між людиною та комп'ютером – це простір, де відбувається співпраця між людьми та машинами. Мета цієї співдії, полягає у тому, щоби забезпечити досконалу роботу та керування машиною з боку людини, а машина одночасно, надає інформацію, яка допомагає прийняттю рішень операторами [12].

Інтерфейс був розроблений у мінімалістичному стилі. Додаток складається із декількох вкладок. Кожна вкладка має поля вводу та кнопки для проведення маніпуляцій із даними. Головне вікно програми зображено на рисунку 3.11.

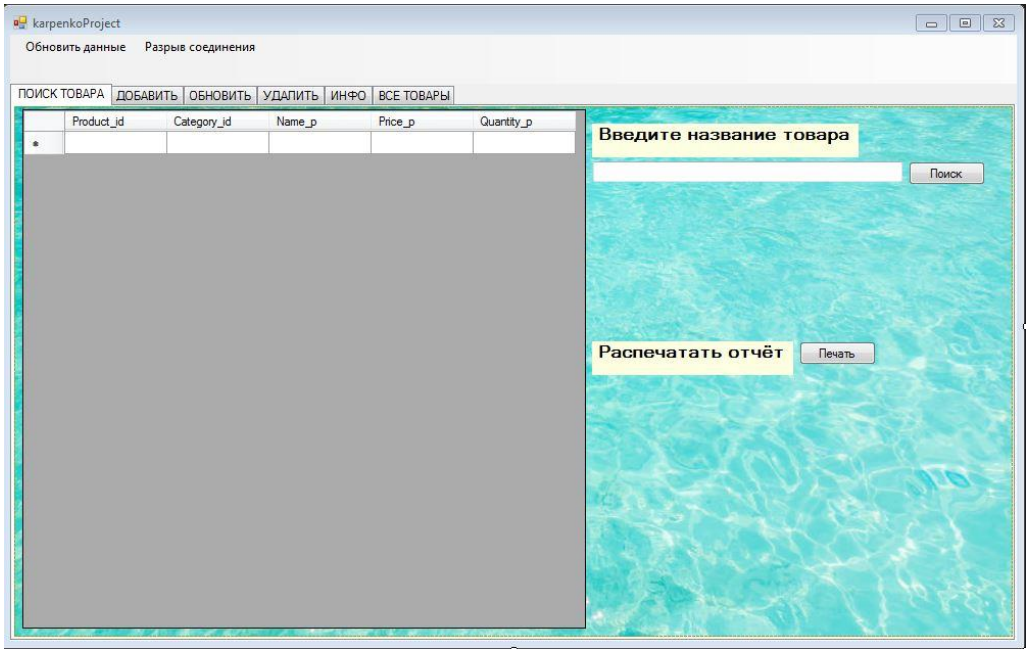

Рисунок 3.11 – Головне вікно програми

Інтерфейс додатку орієнтований на звичайного користувача, його розділи логічні, логіка полів та кнопок максимально зрозуміла.

На рисунку 3.12 зображена вкладка програми, що відповідає за внесення нових даних у систему.

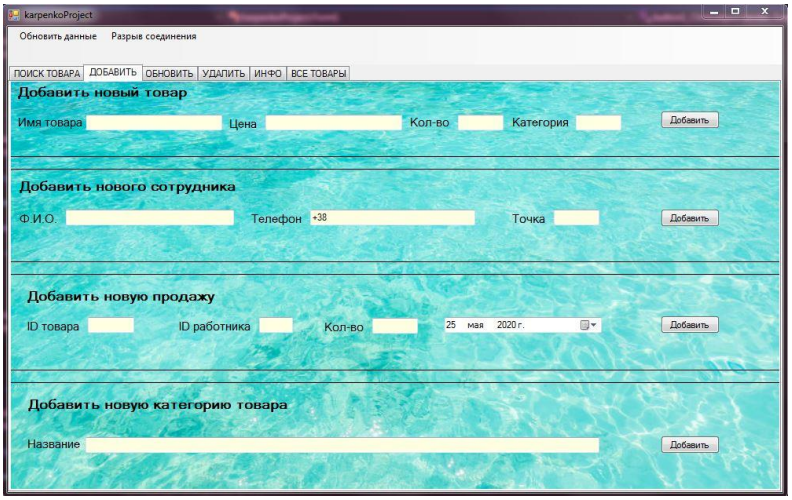

Рисунок 3.12 – Вкладка «ДОБАВИТЬ»

Кожна вкладка поділена на логічні розділи. У якості фонового зображення обраний малюнок з морською тематикою.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Загальні відомості

Даний програмний продукт призначений для прискорення та підвищення якості робочого процесу мережі магазинів роздрібної торгівлі у курортній зоні. Він дозволяє швидко зберігати, відображати та оброблювати різноманітні дані, пов’язані з даною сферою діяльності. Продукт в першу чергу призначений для підприємця, що є власником мережі. Але даною інформаційною системою також можуть користуватися співробітники, з метою внесення та обробки потрібних даних.

Для написання цього продукту використовувалась мова програмування C#, яка є дуже зручна для написання прикладних програм та веб-сайтів та має дуже простий та зручний синтаксис, що значно полегшує розробку та підтримку програмного продукту.

Для розробки даного продукту в якості технології була обрана Windows Forms, що дозволяє створити інтерфейс та за допомогою програмного коду описати логіку взаємодії компонентів інтерфейсу с базою даних. Також додатки, розроблені за допомогою Windows Forms працюють значно швидше, бо вони менш вимогливі до ресурсів системи. Окрім того, для інформаційних систем та подібних бізнес – додатків функціональність та швидкодія системи більш пріоритетні, аніж зовнішній вигляд.

Для створення системи використовувалась середа розробки Visual Studio 2019, .NET Framework версії 4.8. База даних будується на основі Microsoft SQL Server 2012 Local Db. Відповідно, мова програмування C#.

Робота програмного продукту виглядає таким чином : користувач, залежно від своїх потреб, обирає певну вкладку додатку та вносить у відповідні поля необхідні дані. Після цього користувач натискає певну кнопку, та дані підлягають обробці у базі даних.

Для реалізації обробки даних та внесення їх в базу даних була використана мова SQL (structured query language). SQL – «мова структурованих запитів», декларативна мова програмування, що застосовується для створення, модифікації, управління даними в реляційних базах даних, керованих відповідними системами управління базами даних.SQL, перш за все, інформаційно-логічна мова, призначена для опису, зміни і вилучення даних, що зберігаються в реляційних базах даних.

4.2 Опис прийнятих програмних рішень

Використовуючи Microsoft SQL Server 2012 Local Db створимо базу даних для додатку. Процес створення бази даних більш детально описаний у розділі 3 підпункті 3.3 атестаційної роботи.

У середовищі розробки Visual Studio 2019 створимо інтерфейс додатку, використовуючи технологію Windows Forms. Додаток складається із шести розділів, кожен з яких відповідає за певну функціональність:

– пошук товарів;

– додати;

– оновити;

– видалити;

– інфо;

– всі товари.

Після створення форми Windows Forms та бази даних Local Db потрібно організувати з’єднання форми та бази даних для організації передачі, збереження та обробки даних. Для реалізації цього у мові C# передбачений клас «SqlConnection». Щоб його використовувати потрібно імпортувати простір імен «using System.Data» та «using System.Data.SqlClient».

Необхідно створити екземпляр класу «SqlConnection» та в якості аргументу передати йому створену змінну типу string, в яку поміщуємо інформацію про розташування бази даних. В даному проекті використовується локальна база даних, тому вказуємо фізичний шлях до неї. В даному випадку база даних знаходиться у папці з проектом. Програмна реалізація строки з'єднання наведена на рисунку 4.1.

```
private async void Form1_Load(object sender, EventArgs e)
{
    // TODO: данная строка кода позволяет загрузить данные в таблицу "databaseDataSet.Products".
    //При необходимости она может быть перемещена или удалена.
    this.productsTableAdapter.Fill(this.databaseDataSet.Products);
    string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\mypc" +
        @"\Desktop\Diploma\karpenkoProject\karpenkoProject\Database.mdf;Integrated Security=True";

    sqlConnection = new SqlConnection(connectionString);

    await sqlConnection.OpenAsync();
```

Рисунок 4.1 – Програмна реалізація строки з'єднання

Після того як з'єднання організовано, необхідно розробити робочі форми додатку.

Вкладка «пошук товарів» – це перша вкладка яку бачить користувач при першому запуску програми. Даний розділ програми призначений саме для пошуку товарів у базі даних. Вкладка «пошук товарів» зображена на рисунку 4.2.

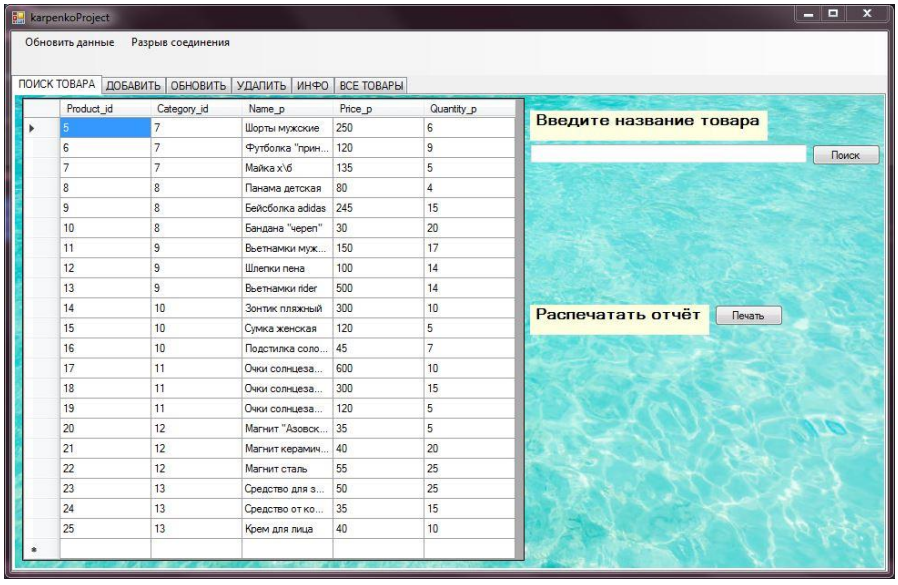

Рисунок 4.2 – Вкладка «пошук товарів»

В даному розділі інформаційної системи реалізовано пошук товарів за допомогою таких елементів як datagridview, textbox, button та label. Користувач набирає у textbox назву товару (або частину назви) та натискає кнопку «пошук». Елемент datagridview відображає результат операції. Фільтрацію даних реалізовано за допомогою методу Binding.Source.Filter з додаванням відповідного SQL запиту. Програмну реалізацію фільтрації даних зображено на рисунку 4.3.

```
ссылка: 1
private void button11_Click(object sender, EventArgs e)
{
    productsBindingSource.Filter = "[Name_p] LIKE '" + textBox25.Text + "%'";
}
```

Рисунок 4.3 – Фільтрація даних

Також у даній вкладці реалізовано можливість роздрукування звіту. Коли користувач натискає кнопку «роздрукувати звіт», то таблиця datagridview, у тому стані, в якому вона є на даний момент роздруковується на доступному принтері робочої станції. Приклад програмної реалізації обробника події кнопки «роздрукувати звіт» зображено на рисунку 4.4.

```
private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    Bitmap bmp = new Bitmap(dataGridView1.Size.Width + 10, dataGridView1.Size.Height + 10);
    dataGridView1.DrawToBitmap(bmp, dataGridView1.Bounds);
    e.Graphics.DrawImage(bmp, 0, 0);
}

ссылка: 1
private void button12_Click(object sender, EventArgs e)
{
    printDocument1.Print();
}
```

Рисунок 4.4 – Програмна реалізація друкування

Наступна вкладка програмного продукту – «додати». Даний розділ програми призначений для внесення та збереження нової інформації до інформаційної системи. Інтерфейс реалізовано за допомогою таких елементів

Windows Forms як textbox, button, label. Для внесення нових даних користувач має заповнити необхідні поля та натиснути кнопку «додати». Вкладку «додати» зображено на рисунку 4.5.

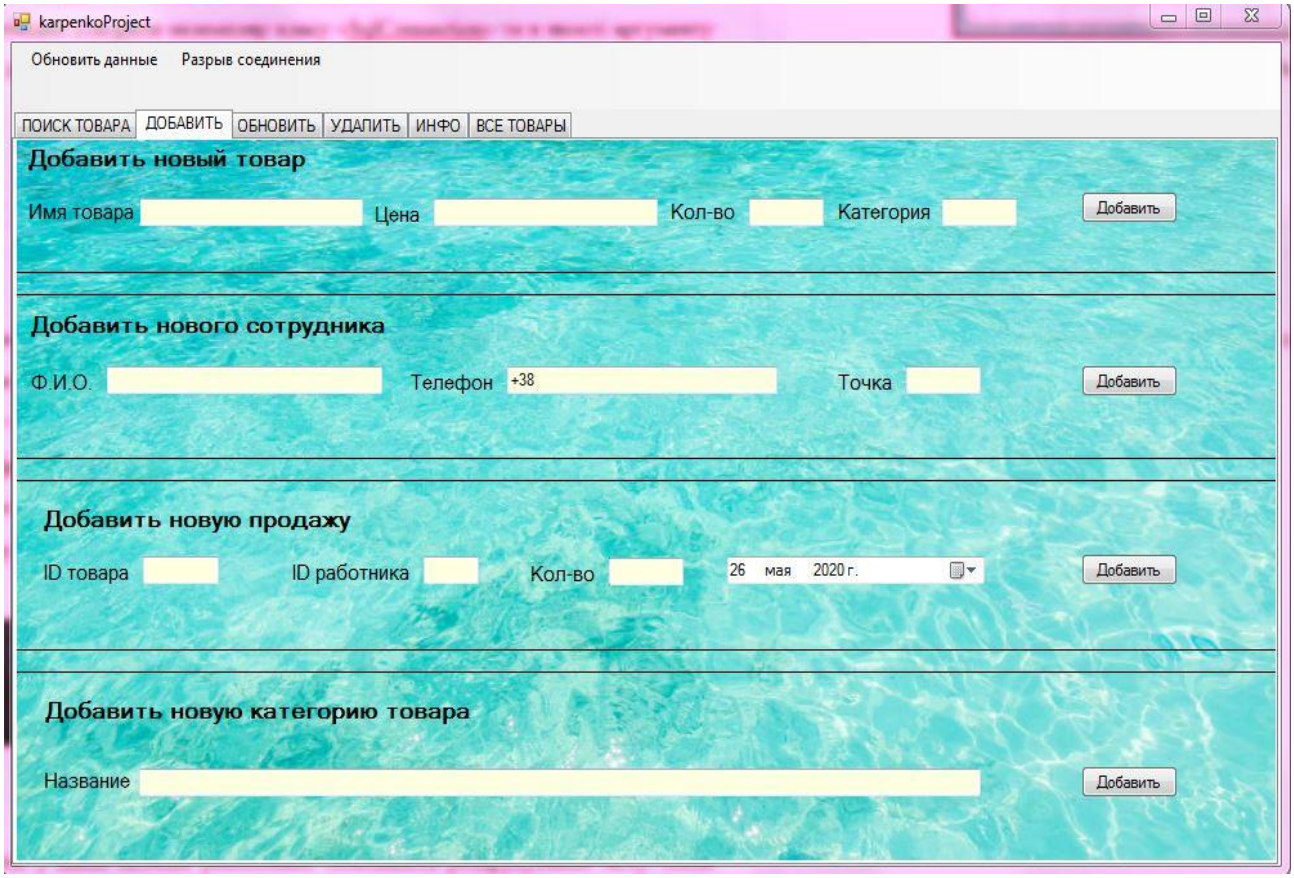

Рисунок 4.5 – Вкладка «додати»

Для зручності користування програмою вкладку візуально поділено на чотири частини, кожна з котрих відповідає за додавання певної інформації:

– додати новий товар;
– додати нового співробітника;
– додати нову транзакцію;
– додати нову категорію.

Алгоритм додавання даних через елементи Windows Forms у базу даних реалізовано за допомогою класу SqlCommand та Sql команди INSERT. Клас SqlCommand являє собою об'єктно-орієнтоване уявлення SQL-запиту, імені таблиці або збереженої процедури. При створенні екземпляру класу ми можемо

одразу передати у якості параметру потрібний SQL-запит, а також строку підключення.

У випадку, якщо користувач не заповнив необхідні поля та натискає кнопку «додати» реалізовано обробник помилки – відображається повідомлення про те, що необхідні поля повинні бути заповнені даними, а також які саме поля. Приклад програмної реалізації кнопки «додати товар» зображено на рисунку 4.6.

```
#region INSERT buttons
ссылка: 1
private async void button1_Click(object sender, EventArgs e)
{
    if (label7.Visible)
        label7.Visible = false;

    if (!string.IsNullOrEmpty(textBox1.Text) && !string.IsNullOrWhiteSpace(textBox1.Text) &&
        !string.IsNullOrEmpty(textBox2.Text) && !string.IsNullOrWhiteSpace(textBox2.Text) &&
        !string.IsNullOrEmpty(textBox7.Text) && !string.IsNullOrWhiteSpace(textBox7.Text) &&
        !string.IsNullOrEmpty(textBox8.Text) && !string.IsNullOrWhiteSpace(textBox8.Text))
    {
        SqlCommand command = new SqlCommand("INSERT INTO [Products] (Name_p, Price_p, Quantity_p, Category_id)" +
            "VALUES (@Name, @Price, @Quantity, @Category)", sqlConnection);

        command.Parameters.AddWithValue("Name", textBox1.Text);
        command.Parameters.AddWithValue("Price", textBox2.Text);
        command.Parameters.AddWithValue("Quantity", textBox8.Text);
        command.Parameters.AddWithValue("Category", textBox7.Text);

        await command.ExecuteNonQueryAsync();
    }

    else
    {
        label7.Visible = true;
        label7.Text = "Все поля должны быть заполнены!";
    }
}
```

Рисунок 4.6 – Програмна реалізація додавання товару

Більш детально лістинг коду системи приведено у додатку А.

Наступна вкладка програмного продукту «оновити» відповідає за оновлення вже існуючої інформації у базі даних. Для зручності користування вікно візуально поділене на дві частини : оновлення даних про товар та оновлення даних про співробітника. Для проведення процедури оновлення користувачеві потрібно лише знати відповідний ідентифікаційний номер товару або співробітника і, після введення його у відповідне поле, він може починати заповнювати ті поля, інформацію про які потрібно оновити.

Вкладку програмного продукту «оновити» зображено на рисунку 4.7.

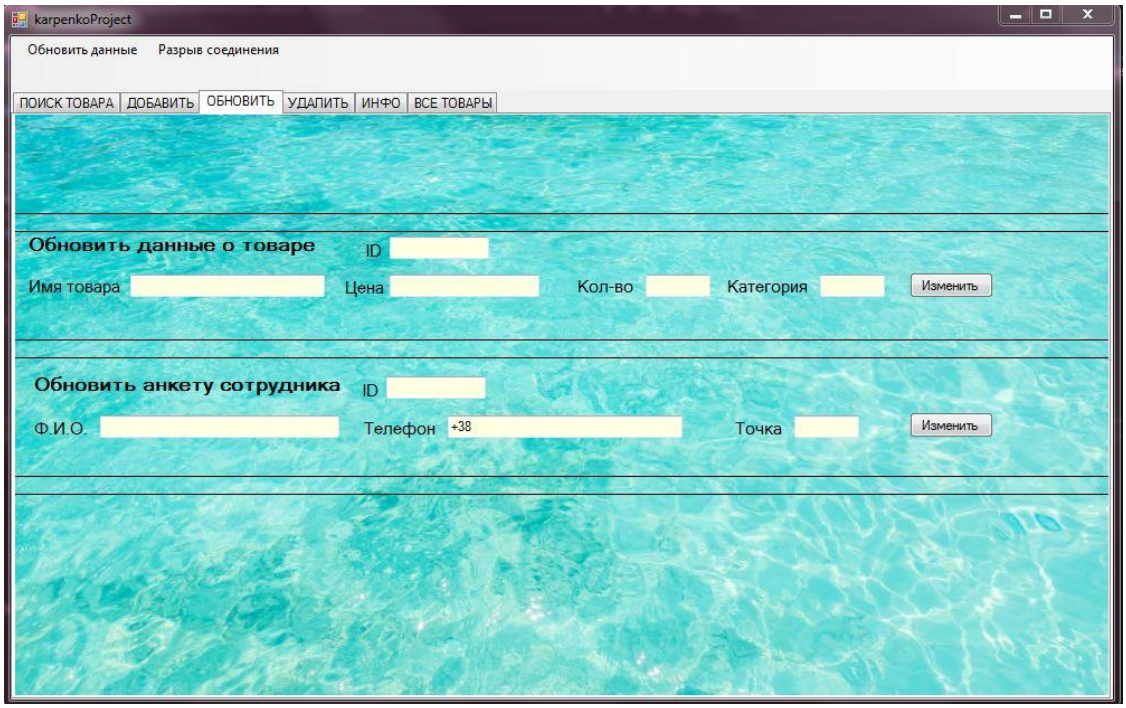

Рисунок 4.7 – Вкладка «оновити»

У випадку, якщо необхідні поля не заповнені, повідомлення про необхідність заповнення цих полів буде відображено. Оновлення даних реалізовано за допомогою класу SqlCommand та SQL-запиту UPDATE. Приклад програмної реалізації оновлення даних про товар зображено на рисунку 4.8.

```
private async void button2_Click(object sender, EventArgs e)
{
    if (label8.Visible)
        label8.Visible = false;

    if (!string.IsNullOrEmpty(textBox3.Text) && !string.IsNullOrWhiteSpace(textBox3.Text) &&
        !string.IsNullOrEmpty(textBox4.Text) && !string.IsNullOrWhiteSpace(textBox4.Text) &&
        !string.IsNullOrEmpty(textBox5.Text) && !string.IsNullOrWhiteSpace(textBox5.Text) &&
        !string.IsNullOrEmpty(textBox9.Text) && !string.IsNullOrWhiteSpace(textBox9.Text) &&
        !string.IsNullOrEmpty(textBox10.Text) && !string.IsNullOrWhiteSpace(textBox10.Text))
    {
        SqlCommand command = new SqlCommand("UPDATE [Products] SET [Name_p] = @Name, [Price_p] = @Price, [Quantity_p] = @Quantity,

        command.Parameters.AddWithValue("Id", textBox5.Text);
        command.Parameters.AddWithValue("Name", textBox4.Text);
        command.Parameters.AddWithValue("Price", textBox3.Text);
        command.Parameters.AddWithValue("Quantity", textBox9.Text);
        command.Parameters.AddWithValue("Category", textBox10.Text);

        await command.ExecuteNonQueryAsync();
    }
    else if (string.IsNullOrEmpty(textBox5.Text) && string.IsNullOrWhiteSpace(textBox5.Text))
    {
        label8.Visible = true;
        label8.Text = "Поле 'ID' должно быть заполнено!";
    }

    else
    {
        label8.Visible = true;
        label8.Text = "Все поля должны быть заполнены!";
    }
}
```

Рисунок 4.8 – Приклад програмної реалізації оновлення даних

Наступна вкладка програми «видалити» призначена для видалення існуючих у базі даних про продавця, товар, категорію товару або транзакції. Вкладку «видалити» зображено на рисунку 4.9.

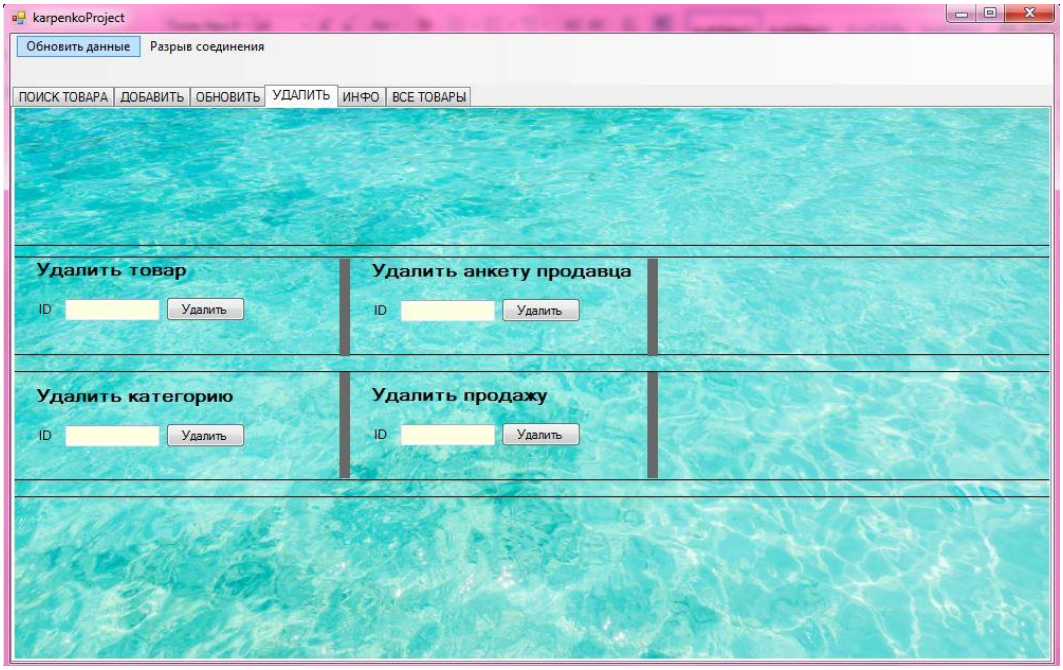

Рисунок 4.9 – Вкладка «видалити»

Все що потрібно користувачеві для здійснення видалення даних – це знати ідентифікаційний номер бажаного запису. Після натискання кнопки – запис видаляється. На рисунку 4.10 наведено програмну реалізацію кнопки видалення.

```
private async void button3_Click(object sender, EventArgs e)
{
    if (label9.Visible)
        label9.Visible = false;

    if (!string.IsNullOrEmpty(textBox6.Text) && !string.IsNullOrWhiteSpace(textBox6.Text))
    {
        SqlCommand command = new SqlCommand("DELETE FROM [Products] WHERE [Product_id] = @Id", sqlConnection);

        command.Parameters.AddWithValue("Id", textBox6.Text);

        await command.ExecuteNonQueryAsync();

    }
    else
    {
        label9.Visible = true;
        label9.Text = "Поле 'ID' должно быть заполнено!";
    }

}
```

Рисунок 4.10 – Приклад програмної реалізації видалення даних

Видалення реалізоване за допомогою класу SqlCommand та SQL-запиту DELETE. У випадку, якщо не заповнене поле з ідентифікатором користувачеві буде виведено повідомлення про помилку.

Вкладки «інфо» та «всі товари» призначені для відображення усіх даних, які є в даний момент у базі. Тут користувач може швидко отримати дані про категорії товарів, адреси торгівельних точок, телефони працівників. Вся ця інформація буде доступна щойно користувач відкриє дану вкладку (рисунок 4.11)

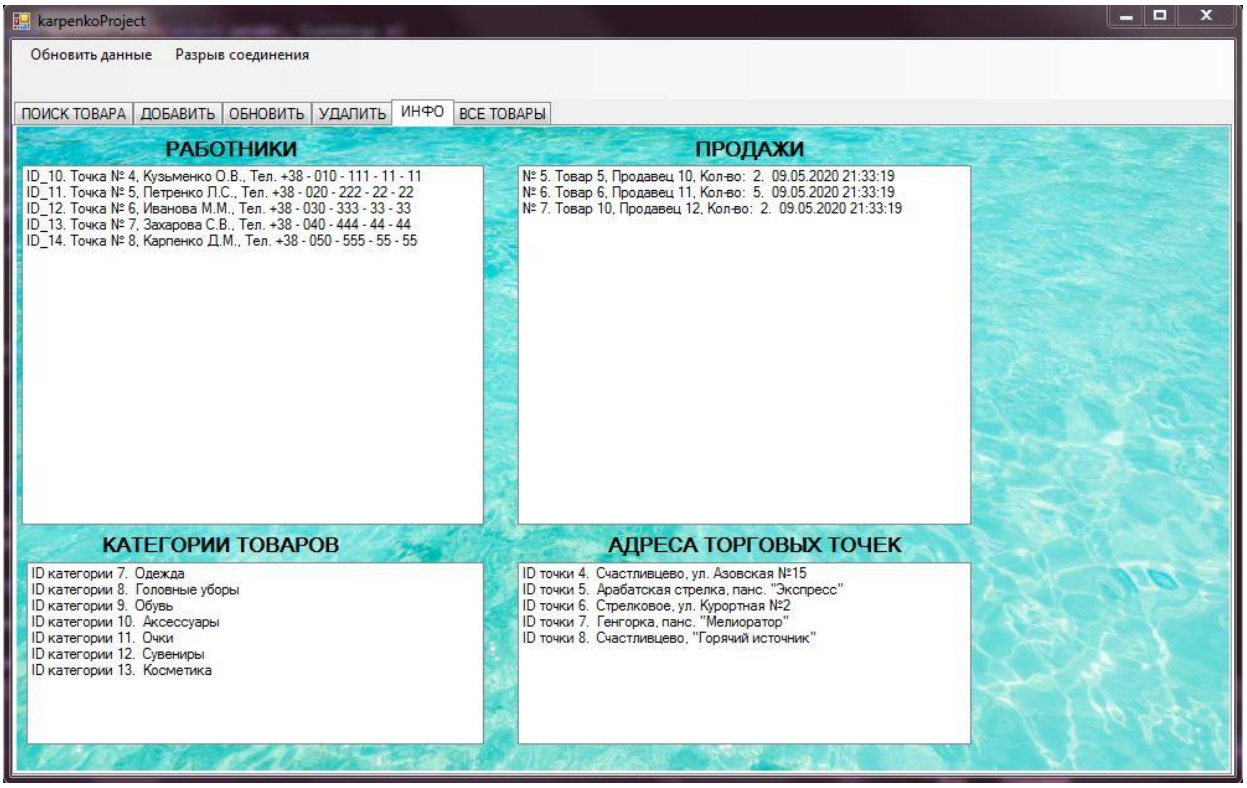

Рисунок 4.11 – Вкладка «інфо»

Інтерфейс реалізовано за допомогою таких елементів як listbox та label. Вибірка даних реалізована за допомогою класу SqlCommand та SQL-запиту SELECT. Детально лістинг коду системи приведено у додатку А.

В даному розділі було здійснено опис прийнятих програмних рішень при розробці інформаційної системи для мережі роздрібної торгівлі у курортній зоні.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування є невід'ємною частиною розробки програмного забезпечення. Це дуже важливий етап розробки, на якому перевіряється, чи всі частини програмного забезпечення ведуть себе таким чином, як і очікувалось.

Враховуючи особливості додатку, що розроблений у даній атестаційній роботі, було проведено функціональне та нефункціональне тестування. В ході функціонального тестування перевірялась функціональність програмного забезпечення. В нефункціональному тестуванні буде перевірено графічний інтерфейс користувача програми.

5.1 Функціональне тестування

Функціональне тестування – вид тестування, який базується на функціях та особливостях, а також взаємодією з іншими системами та може бути представлений на усіх рівнях тестування: компонентному (модульному), інтеграційному, системному та приймальному тестуванні. Функціональні тести – це тести, які розглядають зовнішню поведінку системи .

В якості початкових даних для функціонального тестування буди взяті діаграми «use case», діаграма активності та діаграма послідовності, що наведені у третьому розділі атестаційної роботи.

Для всіх варіантів використання програмного продукту були створені окремі тестові сценарії (TestCase). В ході функціонального тестування була перевірена практично вся функціональність додатку, а саме:

– функція збереження даних;

– функція оновлення даних;

– функція видалення даних;

– функція відображення даних;

– вибірка та фільтрація даних;

– коректна робота обробника помилок.

В ході перевірки функціональності додатку було виправлено незначні недоліки програмного коду.

У таблиці 2 наведено приклад тест кейсу пошуку товарів за критеріями.

Таблиця 2 – Приклад тест кейсу

Інформація про тестовий випадок			
Ідентифікатор тестового випадку		TC 0.7	
Власник тесту		Карпенко Д.М.	
Дата останнього перегляду		27.05.2020	
Мета тесту		Пошук та фільтрація товару	
Методика тестування			
Налаштування прогону тесту	Не проводиться	N/A	
Крок	Дія	Очікувальний результат	Відмітка (V/X)
1	Запустити додаток на робочій станції	Відкривається головна вкладка додатку	V
2	Ввести параметри пошуку у текстовому полі «назва товару»	Введенні дані відображаються у текстовому полі	V
3	Натиснути кнопку «пошук»	В таблиці головної вкладки показано список відфільтрованого товару	V

Кінець таблиці 2

Результати тесту		
Тестувальник: Карпенко Д.М.	Дата прогону тесту: 25.05.2020	Результат тесту (P/F/b): ПРОЙДЕНО (Passed)

Під час тестування було виявлено, що у певні категорії неможливо додати товар. Ця проблема була вирішена корегуванням помилки у зв’язках відношень по зовнішнім ключам.

5.2 Нефункціональне тестування

В ході нефункціонального тестування були протестовані графічний інтерфейс користувача та основні елементі форми додатку. Метою тестування було підтвердження того факту, що користувацький інтерфейс є логічно побудований, комфортний з точки зору юзабіліті, зрозумілий. Перевірялось розташування текстбоксів та кнопок, наявність перевірки заповнення необхідних полів.

У таблиці 3 наведено контрольний список, що містить у собі ряд необхідних перевірок для тестування.

Таблиця 3 –«Check list» тестування GUI

Критерій		Відмітка (V/X)
1	Збереження розташування елементів при зміні розміру форми	V
2	Чи зрозуміло призначення	V

	розділів програмного продукту	

Кінець таблиці 3

Критерій		Відмітка (V/X)
3	Відмічені обов’язкові поля	V
4	Наявність перевірки обов’язкових полів	V
5	Перевірка текстбоксів на переповнення	V

Внаслідок нефункціонального тестування було виявлено, що інтерфейс користувача відповідає усім вимогам юзабіліті, користуватися додатком зручно, всі розділи додатку логічно сформовані. При зміні розміру вікна додатку, елементи форми зберігають встановлені позиції.

В даному розділі було проведено функціональне та нефункціональне тестування розроблюваного програмного забезпечення.

ВИСНОВКИ

У ході атестаційної роботи був розроблений програмний продукт, котрий допомагає в діяльності мережі магазинів роздрібної торгівлі в курортній зоні.

Всі компоненти програмного продукту розроблені з використанням мови програмування C#, технології Windows Forms в середовищі Microsoft Visual Studio 2019, .NET Framework 4.8, Microsoft SQL Server 2012 Local Db. У процесі виконання роботи проаналізовано усі переваги роботи з обраними технологіями, а саме зручність, швидкість, простота та легкість у використанні.

У результаті розробки вдалось виконати поставлену задачу. Створений десктопний додаток підходить для діяльності мережі магазинів роздрібної торгівлі, та виконує всі необхідні функції, пов’язані з цією сферою діяльності.

Інтерфейс та функціонал програмного забезпечення є зрозумілим та комфортним для користувача. Повне використання функцій програми не вимагає наявності спеціальної освіти у комп’ютерних науках або інформаційних технологіях.

Створений додаток активно використовується для підтримки діяльності підприємства у курортному сезоні 2020 року.

Попри виконання поставленої задачі, проект має шляхи для подальшого розвитку. З розвитком підприємства планується розвиток та підтримка додатку, додавання нового функціоналу, такого як автоматизація закупок товару, складання звітів. Можливий перехід на платформу Android для можливості використання додатку на планшетах та смартфонах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дубнов, П.Ю. Access 2000. Проектирование баз данных / П.Ю. Дубнов. - М.: ДМК, 2000. - 272 с.

2. О фирме «1С». Офіційний сайт URL: https://1c.ru/rus/firm1c/firm1c.htm (дата звернення 10.05.2020).

3. 1С:Бухгалтерия предприятия 8.1. Практическое пособие; КноРус - Москва, 2012. - 368 с.

4. «Мой склад». Офіційний сайт URL: https://www.moysklad.ru/ (дата звернення 10.05.2020).

5. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 115 с

6. Буч Г. Язык UML. Руководство пользователя. М.: ДМК Пресс, 2006. – 248 с.

7. Фаулер, Мартин UML. Основы. Краткое руководство по стандартному языку объектного моделирования / Мартин Фаулер. - Москва: СИНТЕГ, 2011. - 192 с.

8. ТЕЗИ ДОПОВІДЕЙ VIII Міжнародної науково-технічної конференції «Інформаційно-комп ютерні технології 2016» - 14с.

9. Подбельский, В. В. Язык C#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, Инфра-М, 2011. - 77 с.

10. Грэй, П. Логика, алгебра и базы данных / П. Грэй. - М.: Машиностроение, 2015. - 195 с.

11. Редько, В.Н. Базы данных и информационные системы / В.Н. Редько, И.А. Бассараб. - М.: Знание, 2004. - 150 с.

12. Михаил Куртов Генезис графического пользовательского интерфейса. К теологии кода / Михаил Куртов. - М.: ТрансЛит, 2014. - 217 с.

ДОДАТОК А
ПРОГРАМНИЙ КОД

Приклад C# коду, який відповідає за відображення даних.

```
public partial class Form1 : Form
   {
      SqlConnection sqlConnection;
      public Form1()
      {
         InitializeComponent();
      }
      private async void Form1_Load(object sender, EventArgs e)
      {
         this.productsTableAdapter.Fill(this.databaseDataSet.Products);
         string connectionString = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\mypc\Desktop\Dipl
oma\karpenkoProject\karpenkoProject\Database.mdf;Integrated Security=True";
         sqlConnection = new SqlConnection(connectionString);
         await sqlConnection.OpenAsync();
      #region SELECT displaying the data
         {
            SqlDataReader sqlReader = null;
            SqlCommand command = new SqlCommand("SELECT * FROM
[Products]", sqlConnection);
            try
            {
               sqlReader = await command.ExecuteReaderAsync();
               while (await sqlReader.ReadAsync())
               {
                  listBox1.Items.Add("ID_" + Convert.ToString(sqlReader["Product_id"])
+ ". " + Convert.ToString(sqlReader["Name_p"]) + ", Цена " +
Convert.ToString(sqlReader["Price_p"]) + ", Кол-во " +
Convert.ToString(sqlReader["Quantity_p"]) + ", Категория " +
Convert.ToString(sqlReader["Category_id"]) + ".");
               }
            }
            catch (Exception ex)
            {
               MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
```

```
            finally
            {
                if (sqlReader != null)
                    sqlReader.Close();
            }
        }
```

Приклад C# коду, який відповідає за збереження даних.

```
private async void button1_Click(object sender, EventArgs e)
    {
        if (label7.Visible)
            label7.Visible = false;
        if (!string.IsNullOrEmpty(textBox1.Text) &&
!string.IsNullOrWhiteSpace(textBox1.Text) &&
            !string.IsNullOrEmpty(textBox2.Text) &&
!string.IsNullOrWhiteSpace(textBox2.Text) &&
            !string.IsNullOrEmpty(textBox7.Text) &&
!string.IsNullOrWhiteSpace(textBox7.Text) &&
            !string.IsNullOrEmpty(textBox8.Text) &&
!string.IsNullOrWhiteSpace(textBox8.Text))
        {
            SqlCommand command = new SqlCommand("INSERT INTO [Products]
(Name_p, Price_p, Quantity_p, Category_id)" +
                "VALUES (@Name, @Price, @Quantity, @Category)", sqlConnection);
            command.Parameters.AddWithValue("Name", textBox1.Text);
            command.Parameters.AddWithValue("Price", textBox2.Text);
            command.Parameters.AddWithValue("Quantity", textBox8.Text);
            command.Parameters.AddWithValue("Category", textBox7.Text);
            await command.ExecuteNonQueryAsync();
        }
        else
        {
            label7.Visible = true;
            label7.Text = "Все поля должны быть заполнены!";
        }
    }
```

Приклад C# коду, який відповідає за оновлення даних.

```
private async void button2_Click(object sender, EventArgs e)
    {
        if (label8.Visible)
```

```
            label8.Visible = false;
        if (!string.IsNullOrEmpty(textBox3.Text) &&
!string.IsNullOrWhiteSpace(textBox3.Text) &&
            !string.IsNullOrEmpty(textBox4.Text) &&
!string.IsNullOrWhiteSpace(textBox4.Text) &&
            !string.IsNullOrEmpty(textBox5.Text) &&
!string.IsNullOrWhiteSpace(textBox5.Text) &&
            !string.IsNullOrEmpty(textBox9.Text) &&
!string.IsNullOrWhiteSpace(textBox9.Text) &&
            !string.IsNullOrEmpty(textBox10.Text) &&
!string.IsNullOrWhiteSpace(textBox10.Text))
        {
            SqlCommand command = new SqlCommand("UPDATE [Products] SET
[Name_p] = @Name, [Price_p] = @Price, [Quantity_p] = @Quantity, [Category_id] =
@Category WHERE [Product_id] = @Id", sqlConnection);
            command.Parameters.AddWithValue("Id", textBox5.Text);
            command.Parameters.AddWithValue("Name", textBox4.Text);
            command.Parameters.AddWithValue("Price", textBox3.Text);
            command.Parameters.AddWithValue("Quantity", textBox9.Text);
            command.Parameters.AddWithValue("Category", textBox10.Text);
            await command.ExecuteNonQueryAsync();
        }
        else if (string.IsNullOrEmpty(textBox5.Text) &&
string.IsNullOrWhiteSpace(textBox5.Text))
        {
            label8.Visible = true;
            label8.Text = "Поле 'ID' должно быть заполнено!";
        }
        else
        {
            label8.Visible = true;
            label8.Text = "Все поля должны быть заполнены!";
        }
    }
```

Приклад C# коду, який відповідає за видалення даних.

```
private async void button3_Click(object sender, EventArgs e)
    {
        if (label9.Visible)
            label9.Visible = false;

        if (!string.IsNullOrEmpty(textBox6.Text) &&
!string.IsNullOrWhiteSpace(textBox6.Text))
        {
```

```
            SqlCommand command = new SqlCommand("DELETE FROM [Products] WHERE [Product_id] = @Id", sqlConnection);
            command.Parameters.AddWithValue("Id", textBox6.Text);
            await command.ExecuteNonQueryAsync();
        }
        else
        {
            label9.Visible = true;
            label9.Text = "Поле 'ID' должно быть заполнено!";
        }
    }
```

ДОДАТОК Б

СЛАЙДИ ПРЕЗЕНТАЦІЇ

Рисунок Б.1 – Титульна сторінка презентації

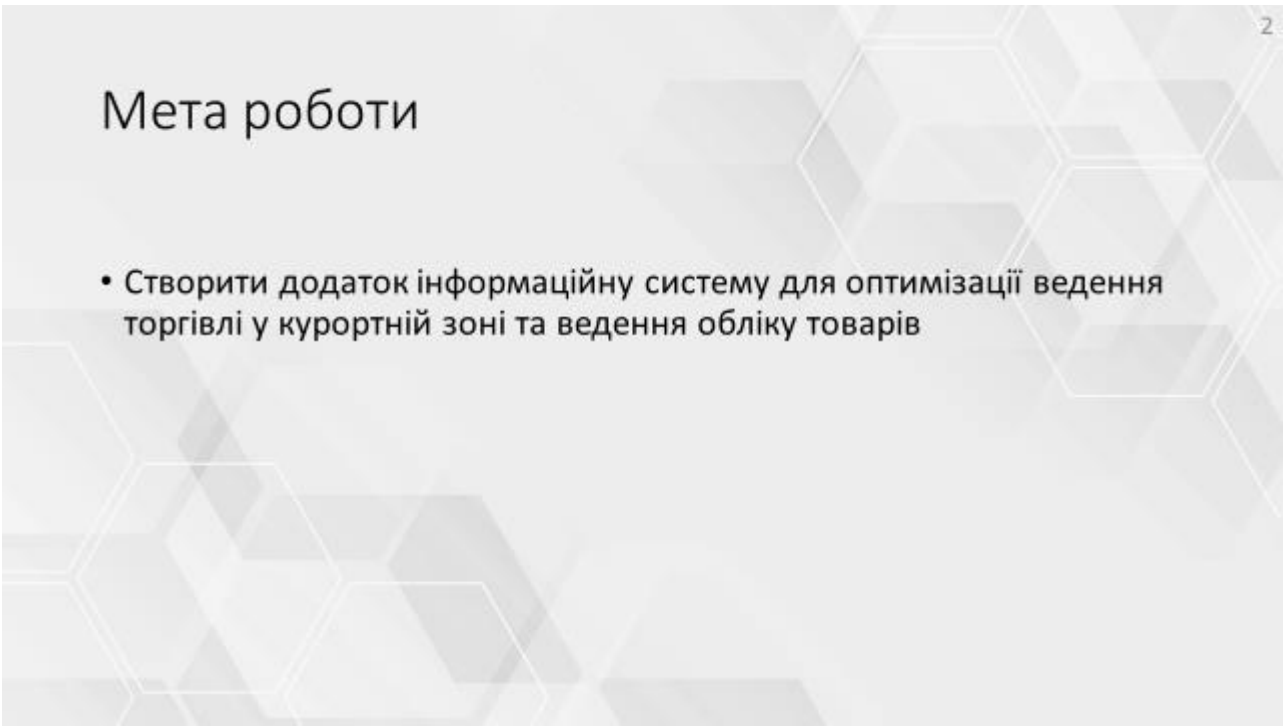

Рисунок Б.2 – Слайд з метою роботи

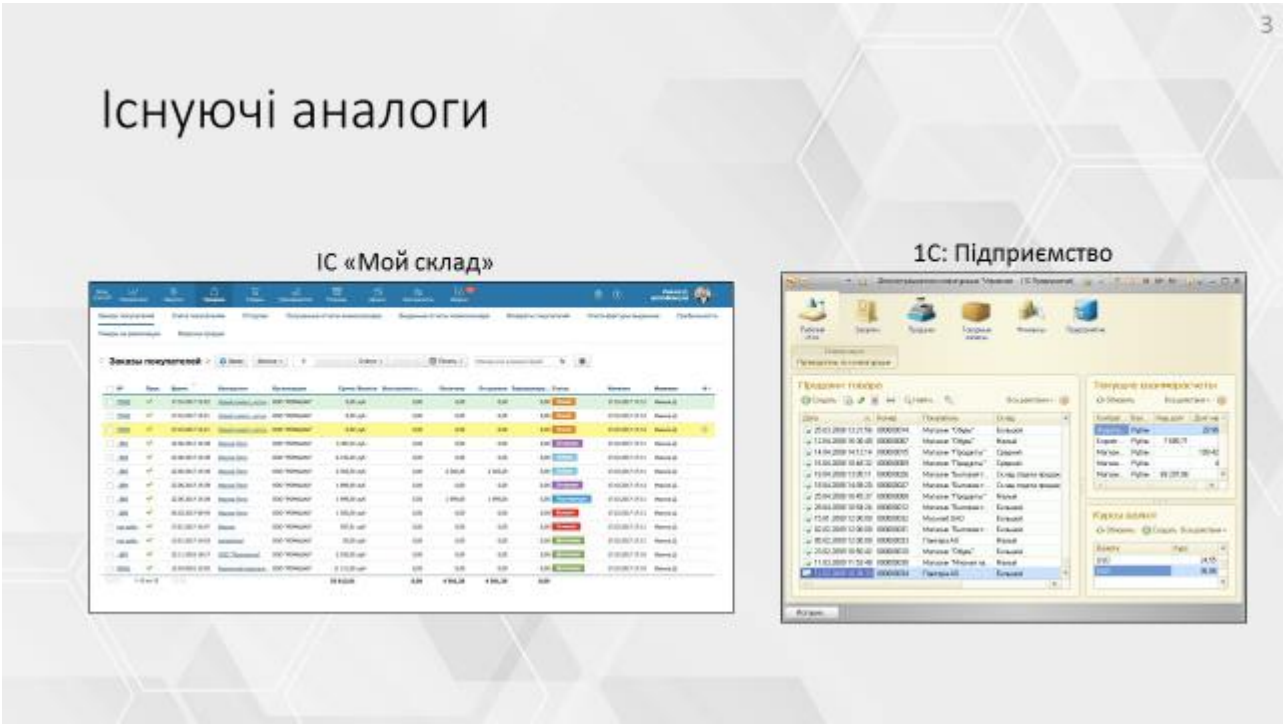

Рисунок Б.3 – Слайд з переліком аналогів

	Можливість ведення обліку товарів	Наявність додаткового функціоналу (відправка звітів, сплата податків тощо)	Можливість подальшого розвитку ПЗ	Популярність ПЗ	Можливість використання ПЗ без професійних навичок	Налаштування ПЗ без залучення фахівця	Продукт безкоштовний
1С: Підприємство	Так	Так	Так	Так	Ні	Ні	Ні
Мой склад	Так	Так	Так	Так	Ні	Ні	Ні
Розроблюваний продукт	Так	Ні	Так	Ні	Так	Так	Так

Рисунок Б.4 – Слайд з таблицею порівняння

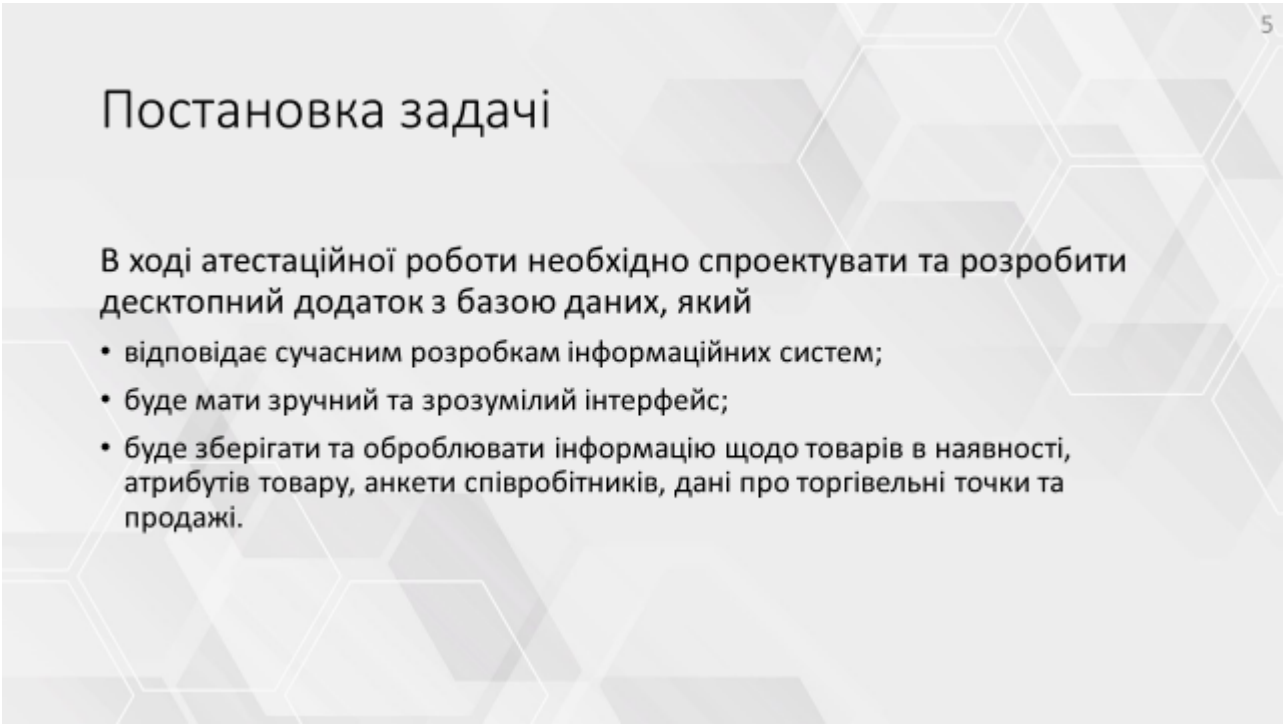

Рисунок Б.5 – Слайд з постановкою задачі

Рисунок Б.6 – Слайд з діаграмою прецедентів

Рисунок Б.7 – Слайд з діаграмою активності обробки заявки

Рисунок Б.8 – Слайд з діаграмою активності перекидання товару

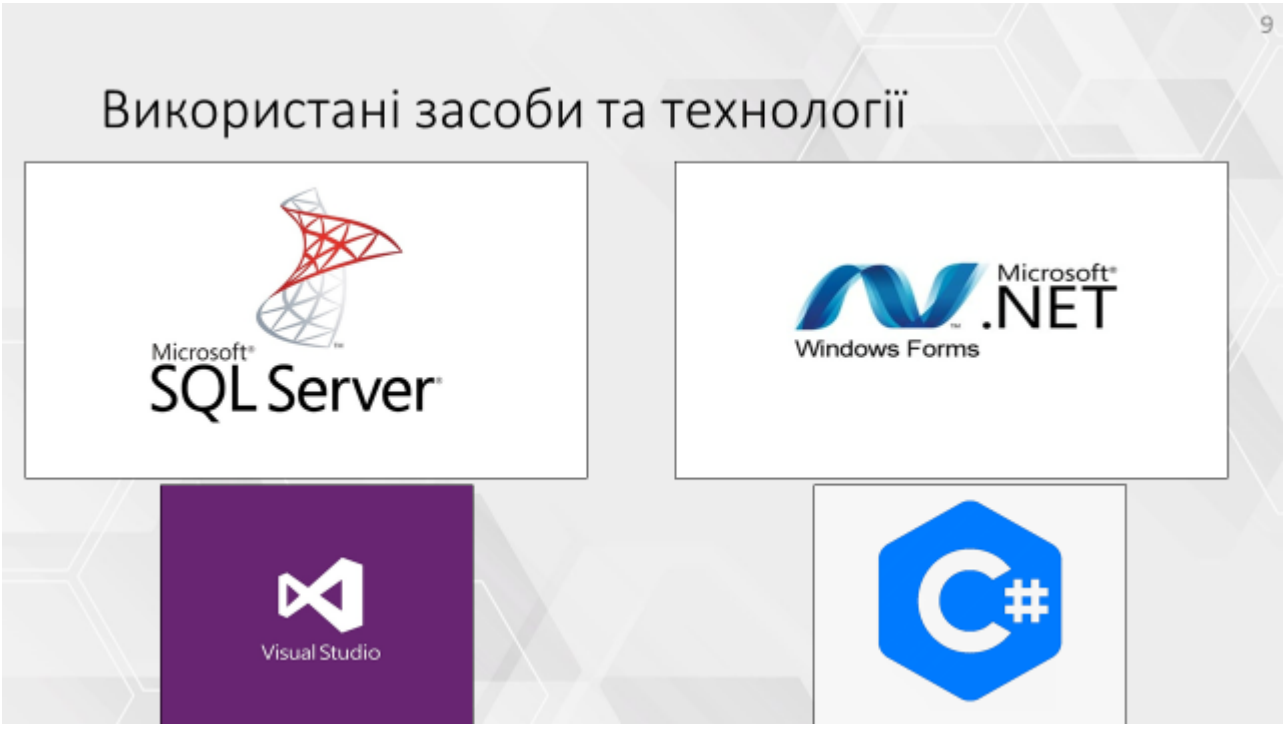

Рисунок Б.9 – Слайд з використаними засобами та технологіями

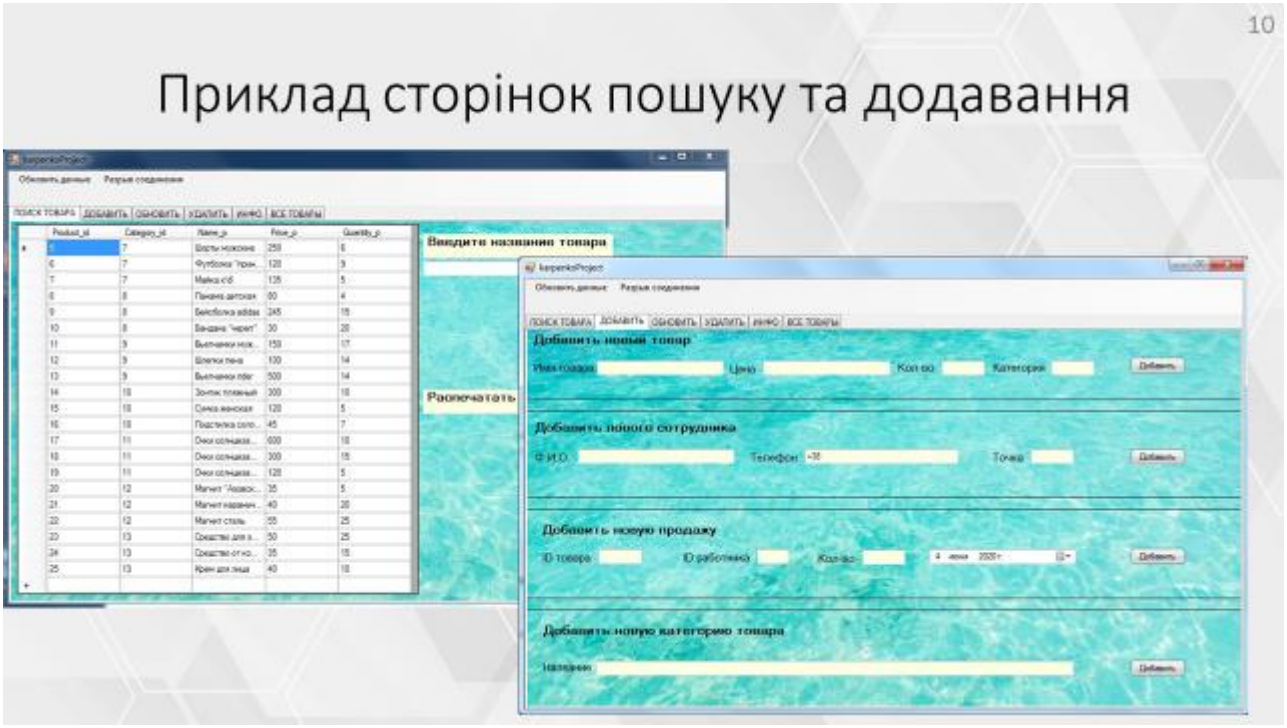

Рисунок Б.10 – Слайд з сторінками пошуку та додавання

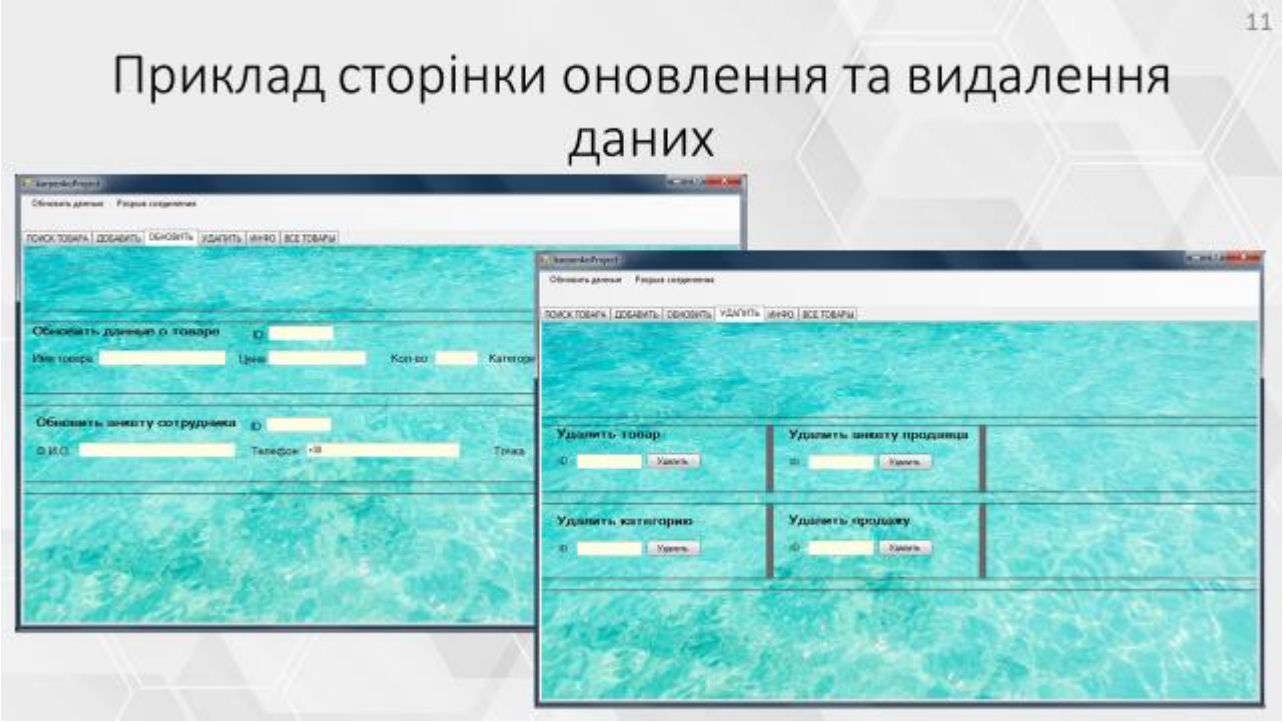

Рисунок Б.11 – Слайд з сторінками оновлення та видалення

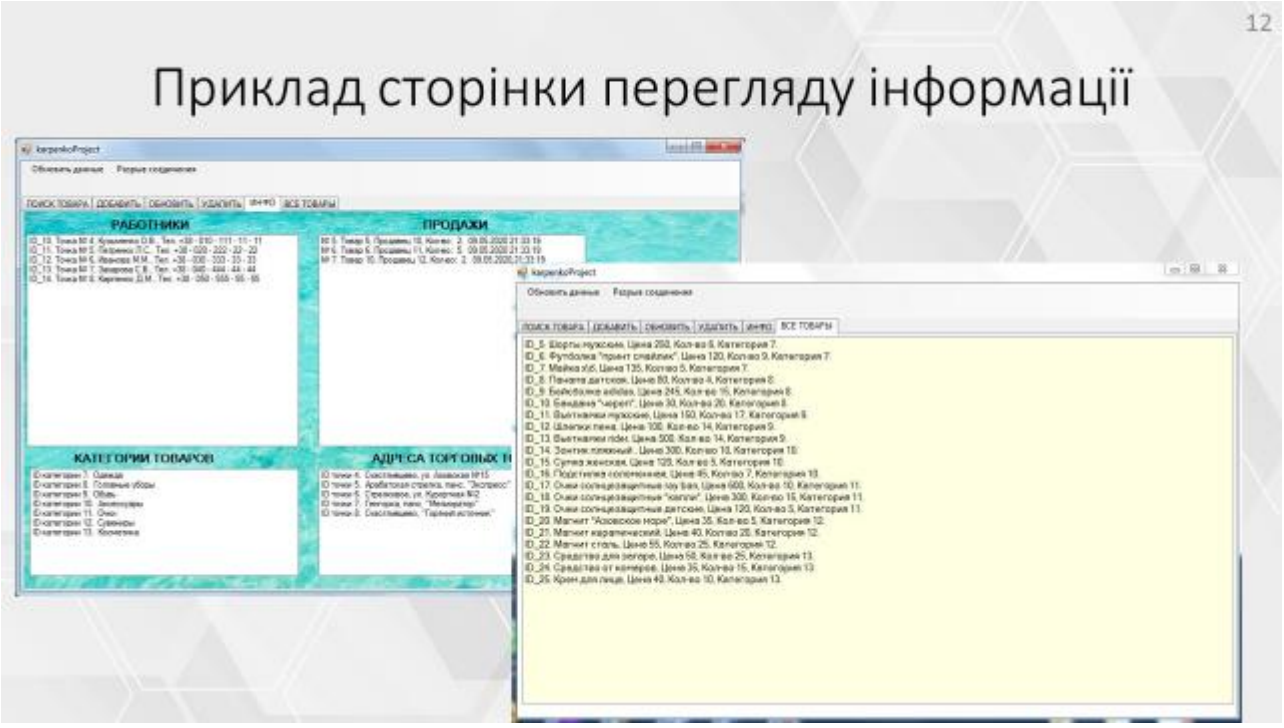

Рисунок Б.12 – Слайд з сторінками перегляду інформації

13

Тестування

- Функціональне тестування
- Тестування UI

Приклад функціонального тестового випадку

Інформація про тестовий випадок			
Ідентифікатор тестового випадку		TC 0.7	
Власник тесту		Карпенко Д.М.	
Дата останнього перегляду		27.05.2020	
Мета тесту		Пошук та фільтрація товару	
Методика тестування			
Налаштування прогону тесту		Не проводиться	N/A
Крок	Дія	Очікуваний результат	Відмітка (V/X)
1	Запустити додаток на робочій станції	Відкривається головна сторінка додатку	V
2	Ввести параметри пошуку у полі «назва товару»	Введені дані відображаються у текстовому полі	V
3	Натиснути кнопку «пошук»	В таблиці відображено список відфільтрованого товару	V
Результати тесту			
Тестувальник: Карпенко Д.М.	Дата прогону тесту: 25.05.2020		Результат тесту: ПРОЙДЕНО(Р)

Рисунок Б.13 – Слайд з тестуванням

14

Висновки

- У ході атестаційної роботи був розроблений програмний продукт, котрий допомагає в діяльності мережі магазинів роздрібної торгівлі в курортній зоні
- Інформаційна система підходить для діяльності мережі магазинів роздрібної торгівлі, та виконує всі необхідні функції, пов’язані з цією сферою діяльності
- Додаток вже використовується з початку курортного сезону 2020 року
- Програмне забезпечення має перспективи для розвитку:
  - додавання нового функціоналу в залежності від вимог підприємства;
  - перехід на мобільні платформи.

Рисунок Б.14 – Слайд з висновками

ДОДАТОК В

ЕЛЕКТРОННІ МАТЕРІАЛИ