Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)

(назва факультету)

(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

магістр
(освітній ступінь (освітньо-кваліфікаційний рівень))

на тему:
Розробка автоматизованої системи управління мінімаркетом для оптимізації роботи з продажами на базі Програмного середовища C# . Net

Виконав: студент (ка) 6 курсу, групи СПм-61
спеціальності (напряму підготовки)
121 – Інженерія програмного забезпечення
(шифр і назва спеціальності (напряму підготовки))

	(підпис)	Кухарук Ю. О. (прізвище та ініціали)
Керівник	(підпис)	Бойко І.В. (прізвище та ініціали)
Нормоконтроль	(підпис)	Бойко І.В. (прізвище та ініціали)
Рецензент	(підпис)	Приймак М. В. (прізвище та ініціали)

м. Тернопіль – 2019

АНОТАЦІЯ

Магістерська робота на тему «Розробка автоматизованої системи управління мінімаркетом для оптимізації роботи з продажами на базі програмного середовища C# .Net» Кухарук Юрій Олександрович. – Тернопільський національний технічний університет імені Івана Пулюя, Факультет комп'ютерно-інформаційних систем і програмної інженерії, Кафедра програмної інженерії, група СПм–61 // Тернопіль, 2019.

Дана робота містить 90 сторінок, 10 таблиць, 35 рисунків, список використаної літератури з 20 найменувань та 3 додатки.

Метою магістерської роботи є розробка автоматизованої системи управління мінімаркетом для оптимізації роботи з продажами на базі програмного середовища C# .Net. Система повинна забезпечити максимально зручний та швидкий спосіб продажі товарів та ведення усієї звітності пов’язаною з цією діяльністю.

Під час виконання даної роботи було застосовано мову програмування C# та технології .Net. Середовище розробки Visual Studio дозволили використати усі преваги розробки графічного дизайну з допомогою Windows Form. Для зберігання даних було обрано базу даних MySql.

При виконанні даної роботи було розглянуто існуючі програмні комплекси та проведено їх аналіз. Проаналізовано всі їх переваги та недоліки.

Результатом розробки стало програмне забезпечення яке дозволяє спростити продаж товарів, та дозволяє максимально швидше обслуговувати клієнта мінімаркету. Проведено проектування баз даних та програмної системи.

Ключові слова: WINDOWS, MYSQL, WINDOWS FORM, C# .NET, МІНІМАРКЕТ, АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ, ІНСТРУМЕНТ АВТОМАТИЗАЦІЇ.

ABSTRACT

Master's thesis on "Development of automated mini-market management system for optimization of work with sales on the basis of the C # .Net software environment" Kukharuk Yuriy Aleksandrovich. - Ivan Pulyuy National Technical University of Ternopil, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, SPm-61 group // Ternopil, 2019.

This work contains 90 pages, 10 tables, 35 figures, a list of used literature of 20 titles and 3 appendices.

The purpose of the master's work is to develop an automated system for managing the mini-market to optimize the work with sales on the basis of C # .Net software environment. The system should provide the most convenient and fast way to sell the goods and maintain all reporting related to this activity.

C # programming language and .Net technology was used in this work. The Visual Studio development environment made it possible to take full advantage of graphic design development with Windows Form. MySql database was selected for data storage.

In the course of this work, existing software complexes were considered and analyzed. All their advantages and disadvantages are analyzed.

The result of the development is software that simplifies the sale of goods and allows the customer to serve the minimarket as quickly as possible. Design of databases and software system was performed.

Keywords: WINDOWS, MYSQL, WINDOWS FORM, C # .NET, MINIMARKET, AUTOMATED CONTROL SYSTEM, AUTOMATION TOOL.

ЗМІСТ

ВСТУП

Експлуатація мінімаркету вимагає отримання багатьох адміністративних, управлінських та маркетингових навичок. Зважаючи на те, що достатня кількість записів для складання щомісячних звітів про продажі повинна бути опрацьованою та систематизована, ці навички необхідні для забезпечення безперебійності роботи мінімаркету. Окрім цього варто враховувати велику кількість товару то його систематизацію. У вирішенні цих проблем може допомогти автоматизована система управління продажами мінімаркету.

Автоматизована система управління це місце, де ваш клієнт здійснює оплату за товари чи послуги у вашому магазині, а також місце касира з касовим апаратом який допоможе клієнтові здійснити оплату.

Наскільки очевидними є переваги АСУ, статистика показує, що 56 відсотків роздрібних продавців все ще не використовують її. Натомість, багато хто все ще використовує комбінацію ручних методів, касових апаратів, QuickBooks та Excel для ведення бухгалтерського обліку.

То чому продавці ще не зробили цього кроку до АСУ? Для початку впровадження нової технології - особливо технології, яка є центральною у вашому бізнес-процесі - може бути страшною та непосильною. Роздрібні торговці повинні враховувати негативні наслідки відсутності АСУ.

Гнучкість у роботі є ключовою. Варто завжди переконуватися, що постачальник системи працює з платіжним процесором / шлюзом на вибір, щоб була можливість контролювати витрати. Також варто переконатися, що АСУ може безперешкодно інтегруватися, щоб не було перебоїв у роботі мінімімаркету.

Варто також врахувати, що впровадження системи управління в мінімаркеті вимагає певного устаткування та умов.

На сьогоднішній день впровадження таких систем стає все більше актуальним, адже асортимент товарів та послуг невпинно розширюється, а тримати у пам’яті усі ціни просто неможливо.

Метою магістерської роботи є розробка автоматизованої системи управління мінімаркетом для оптимізації та спрощення продажів. Основним завданням є забезпечення максимально швидкої обслуги клієнта, продажі великої кількості товару максимально зручним для мінімаркету та клієнта способом.

Для того, щоб досягти вище вказану пету вирішити такі завдання:

– дослідити предметну область та здійснити постановку проблеми;

– побудувати модель автоматизованої системи;

– реалізувати програмне забезпечення для автоматичної системи управління;

– провести тестування розробленої системи.

Дана тема є актуальною, оскільки на даний момент існує велика кількість мінімаркетів та магазинів, які потребують оновлення або покращення програмного забезпечення. Велика кількість інформації про товари зберігається в незручному та важкодоступному вигляді, а користувацький інтерфейс ускладнює навчання користувачів даними системами. Важливим також є ведення історії продажів для отримання звітності про прибутковість магазину чи мінімаркету. В даному програмному забезпеченні також реалізовано можливість додавання нового касира в систему, а також отримання інформації та звітності про продажі.

З введенням подібної системи у роботу буде вирішено вище згадані проблеми, а також швидке і інтуїтивно зрозуміле керування автоматичною системою управління.

Для написання магістерської роботи було обрано мову програмування C#. Вибір було зроблено у зв’язку з зручними інструментами, які нам надає ця мова для написання графічного інтерфейсу та роботи з базою даних. Як середовище розробки було вибрано Visual Studio 2017 та технологію Windows Forms.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АСУ	автоматична система управління
UI (User Interface)	користувацький інтерфейс
ПЗ	програмне забезпечення
БД	база даних
SQL (structured query language)	декларативний мова програмування, яка застосовується для створення, модифікації та керування даними в реляційних базах даних, керований відповідною системою управління базами даних
CLI (Common Language Infrastructure)	загальна мовна інфраструктура, також зустрічатися як CLI - це платформа, на якій виконуються програми .Net.
API (Application Programming Interface)	інтерфейс програмування додатків (API) - це інтерфейс або протокол зв'язку між різними частинами комп'ютерної програми, призначений для спрощення впровадження та обслуговування програмного забезпечення.

1 АНАЛІЗ ВИМОГ ТА ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

Завданням магістерської роботи є розробка автоматичної системи управління мінімаркетом для оптимізації та покращення продажів. Система повинна мати зручний та зрозумілий інтерфейс який забезпечить продуктивну роботу користувача та спростить продаж товарів.

Програмні засоби системи повинні забезпечувати справне її функціонування в нормальних умовах. Також потрібно передбачити можливість того, що буде одночасне звернення багатьох запитів до неї. Система повинна гарантувати безпеку введених даних та виконувати те, що від неї вимагають користувачі. Ключовою є можливість роботи з сканером штрих кодів та бар кодів, оскільки саме вони забезпечують максимально швидку та комфортну роботу з системою [16].

Проаналізувавши предметну область, можемо виділити перелік задач, які необхідно вирішити:

- Розробка архітектури системи;
- Проектування бази даних;
- Реалізація роботи сканера штрих кодів;
- Реалізація системи;
- Тестування системи.

До основних функцій системи можна віднести :

- Робота з сканером штрих кодів та бар кодів;
- Швидкий пошук товару за кодом;
- Збільшення асортименту товарів в системі;
- Ведення звітності продажів;
- Можливість автоматично вирахувати знижку за допомогою дисконтної карти постійного клієнта;
- Підтримка роботи декількох користувачів.

Дана предметна область є дуже актуальною, оскільки ключовим завданням мінімаркету вважається вирішення та задоволення базових і не тільки потреб покупців. Великий асортимент базових товарів, мінімаркет може з легкістю замінити інші, менші торгові точки та магазини із спец товарами. Забезпечуючи великий обсяг продукції, зазвичай мінімаркити витісняють привичні для більшості людей магазини, тому даний напрямок розвивається і виникає гостра потреба в автоматизації та розробці додаткового ПО [14].

Сутність постановки задачі полягає в розробці програмного забезпечення для ПК, яке повинне спростити та пришвидшити продаж товарів в мінімаркеті. На рисунку 1.1.1 зображено схему роботи мінімаркету.

Рисунок 1.1.1 - Схема роботи мінімаркету

Сучасний мінімаркет це великий магазин з найрізноманітнішим асортиментом товарів. Для швидкого продажу та оновлення даних про ці товари виникає необхідність в розробці сучасних програмних систем. Тому виникають певні вимоги до програмних систем. Під час створення даної програми потрібно забезпечити максимальну швидкість взаємодії з користувачем, а також простоту і зручність використання користувацького інтерфейсу [17].

1.2 Огляд вимог до програмного забезпечення

1.2.1 Функціональні вимоги

АСУ мінімаркетом – це система яка спрощує та прискорює обслуговування клієнтів. Саме тому то цієї системи висувається ряд вимог, які повинні забезпечувати якісне та коректне функціонування програми. Цей тип вимог описує, що система повинна робити. В переліку нижче перераховано ключові функціональні вимоги до системи:

- кожна функція повинна бути коректно реалізована та виконувати те, що від неї вимагається;
- вибір товару за допомогою штрих коду;
- робота системи з базами даних MySql;
- можливість додати в чек одночасно більше 30 товарів;
- час на додавання одиниці товару не повинен перевищувати 0,5 секунди;
- сканування дисконтної карти за допомогою сканера штрих кодів або при допомозі вводу ідентифікаційного коду картки;
- додавання декількох користувачів в систему;
- формування звітності по продажах за певний період;
- пошук товару за допомогою коду;
- система автоматичну вираховує решту, яку потрібно видати клієнту.

1.2.2 Нефункціональні вимоги

Окрім функціональних вимог, які було описано вище, також велику роль відіграються нефункціональні вимоги. На відміну вид описаного вище, нефункціональні вимоги описують не те, що повинна робити система, а те як вона повинна це робити [16]. Ключовими вимогами до системи є:

- лаконічний та зрозумілий користувацький інтерфейс;
- ключові дані списку товарів повинні бути представлені у вигляді таблиці;

1.3 Розгляд аналогів ПЗ

1.3.1 Програма ShopDesk

ShopDesk – один з аналогів АСУ мінімаркету. Офісна система, для спрощення та вдосконалення роботи касир в магазинах з різною специфікою, а також з самойстійним обслуговуванням. Дозволяє інтегруватися в систему торгового центру, представленя єдиним модулем, який дозволяє працювати з різноманітним обладнанням, в тому чилсі торговими.

Система не є вимогливою до швидкісного каналу з базою даних (при роботі з MSSQL Server), ця особливість дозволяє без проблем налагодити віддалений доступ через інтернет. Окрім цього, також реалізовано функцію роботи з базою даних навіть при відсутності безперервного з'єднання з базою даних на сервері через інтернет. Реалізована підтримка з різними реєстраторами, а також з фіскальними, а також можливість роботи з картками знижок для постійних клієнтів, надання звітів різного типу. Можливість широкою кастомізації роботи касира та ведення повної звітності його дій

Робота додатку відбувається з базою даних безпосередньо, завдяки чому стає можливим отримувати найактуальнішу та найбільш достовірну інформацію про наявний в магазині товар.

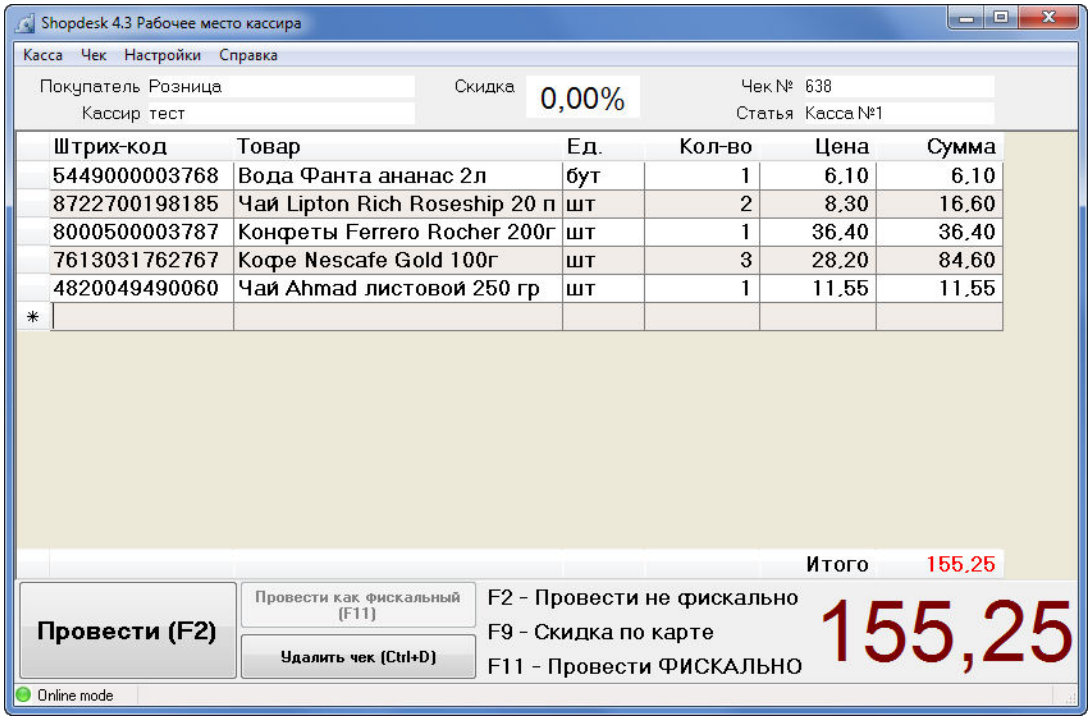

Рисунок 1.3.1 – головне вікно програми ShopDesk

Програмна системи має низькі вимоги до апаратного та програмного обладнання, що дозволяє працювати з різноманітними типами та моделями сканерів та принтерів, не є вийнятком і чекові. Мінімізований час навчання користувачів та касирів даної програми. За допомогою зручного інтерфейсу касир має можливість ідентифікувати товар та внести його в рахунок за допомогою сканера штрих-кодів. Після сканування товару, немає значення чи він сталий чи ваговий, в рахунок записується «1» для сталого товару, тобто новий запис. В залежності від кількості касир може змінити дане значення на необхідну кількість. Якщо мова йде про ваговий товар, то відбувається сканування товару за допомогою штрих-коду, який раніше було надруковано на етикетці спеціальним принтером.

1.3.2 Програма SoftKuB

SoftKuB – інший аналог АСУ мінімаркету. OLAP-система для настільких ПК. Головним призначенням даної програми є аналіз даних х побудовою графіків у реальному часі. Це може бути графік побудований по різних критеріях, наприклад продажі по датах (місяцях, декадах, кварталах), також можуть бути різні групи товарів та агентів. Кількість зусиль для побудови графіку зводиться до мінімуму. Ключовою особливість даної системи є швидкодія. Також одною з особливостей є сумісність з «1С Підприємство» та іншими подібними ПЗ. Підтримується також сумісність з реляційними БД. Що в свою чергу допомагає досягти високої ефективності бізнесу.

Одним з основних компонентів для відображення даних є Pivot Grid – інструмент, який дозволяє будувати графіки або «куби» звичайним перетягуванням мишею потрібної інформації, побудова відбувається в реальному часі.

Завдяки цій функції можна за декілька секунд з звіту продажі по датах утворити звіт продажі по інших критеріях, наприклад по клієнтах [14].

Програма дозволяє виділити мишкою певну ділянку графіка, яка необхідна.

Це дозволяє автоматично відобразити графік або діаграму у потрібному вигляді. Саме ця особливість дозволяє, при потребі, максимально швидко скоротити час побудові графіка з необхідними перерізом даних. За допомогою OLE DB або ODBC можливо передати дані для побудови графіків. Це можуть бути як Excel файли так і бази даних різних форматів, наприклад MBD або бази даних типу SQL.

Для цієї програми на ринку є велика кількість конкурентів, однак перевагою саме SoftKub є ціна та зручність самої роботи з нею. Дане програмне забезпечення дозволяє будувати графіки за допомогою простого перетягування мишкою, що в свою чергу є унікально серед такого роду ПЗ. І не менш приємною є ціна, адже вона у три, чотири рази нижча і ніж у конкурентів, які можуть надати хоча б наближений функціонал.

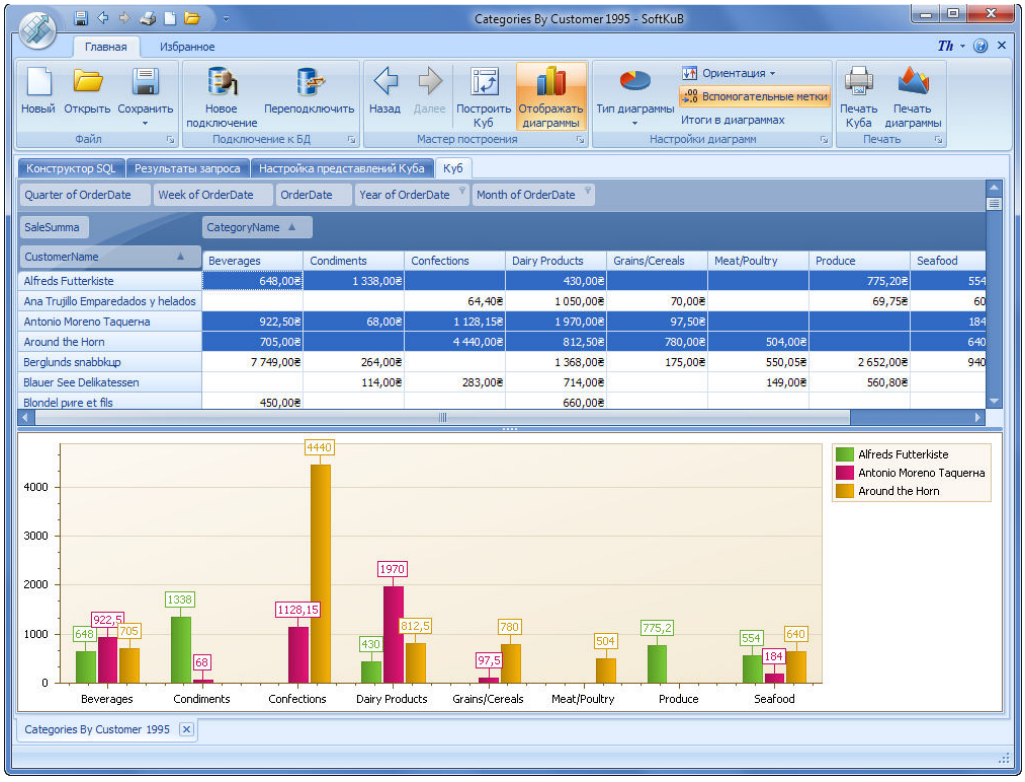

Рисунок 1.3.2 – головне вікно програми SoftKuB

1.3.3 Програма «Товари, Ціни, Облік…»

«Товари, Ціни, Облік…» – ще один з аналогів. Основне призначення програми – облік товару на виробництвах та у магазинах, середній бізнес.

Також програма дозволяє вести облік в торгових точках, складах, а також в мінімаркетах. Дозволяє приймати оплати від клієнтів. Відстежує обсяг здійснених продаж. Листи розпорядження та їх маршрути. Розмежування прав користувачів системи. Дозволяю здійснювати контроль за операторами. В основному програму застосовують індивідуальні підприємства та супермаркети, роздрібна та оптова торгівля. Система працює в мережі з одною базою даних та має можливість підключати необмежену кількість інших користувачів, робочих місць.

«Товари, Ціни, Облік ...» надає широку можливість по розподілу прав у межах системи та організації. Кожному користувачеві надається рівно стільки прав у системі, скільки від нього вимагається для виконання максимально якісної роботи. Також є можливість налаштувати досить якісно та детально, що дозволяє утворювати певні зв’язки для взаємоконтролю працівникам один одного. Саме при умові такої роботи досягається максимальна ефективність роботи колективу. Окрім цього система веде журнал усіє операцій та дій зроблених операторами.

Також покращенню продуктивності сприяє максимально зручний та інтуїтивний інтерфейс, до якого досить швидко звикається.

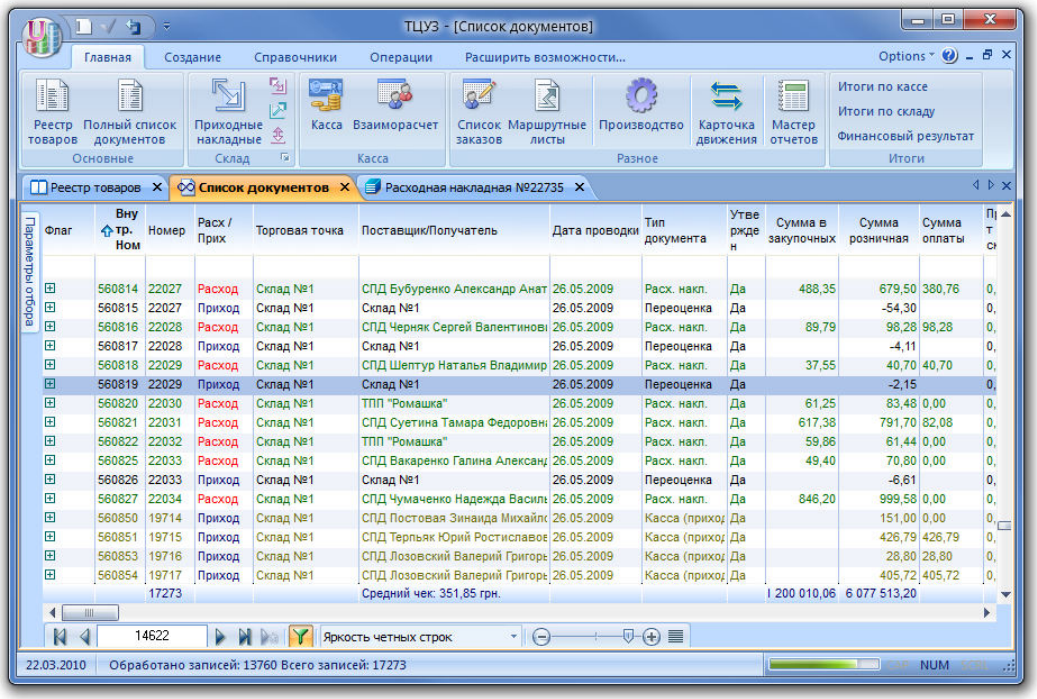

Рисунок 1.3.3 – головне вікно програми «Товари, Ціни, Облік ...»

1.4 Вибір моделі розробки

Після глибокого аналізу вимог до системи та специфіки вибраної предметної області, було вирішено обрати еволюційну модель розробки ПЗ.

Саме завдяки цій методології з’являється можливість для глибокого дослідження предметної області, що в свою чергу дозволяє провести аналіз потреб замовника і придатність даної моделі в межах наявного проекту. Цю модель використовується в основному при розробці некритичних систем і головною метою є реалізація системного функціоналу. Також допускається, що вимоги до системи визначені не повність. Саме через це реалізація ключового функціоналу відбувається ітераційно, що дозволяє на кожному етапі отримувати робочий прототип на якому власне і відбувається перевірка закладених раніше вимог та функціоналу. При дотриманні такого процесу стає можливим розробити максимально якісний продукт, який задовільнить усі вимоги навіть найвибагливішого замовника будь-якої системи. Дещо схожою є спіральна модель розробки ПЗ [7].

Наступним кроком цієї моделі є еволюційне прототипіювання в межах розробки усього ЖЦ програмної системи (рис. 1.4.1).

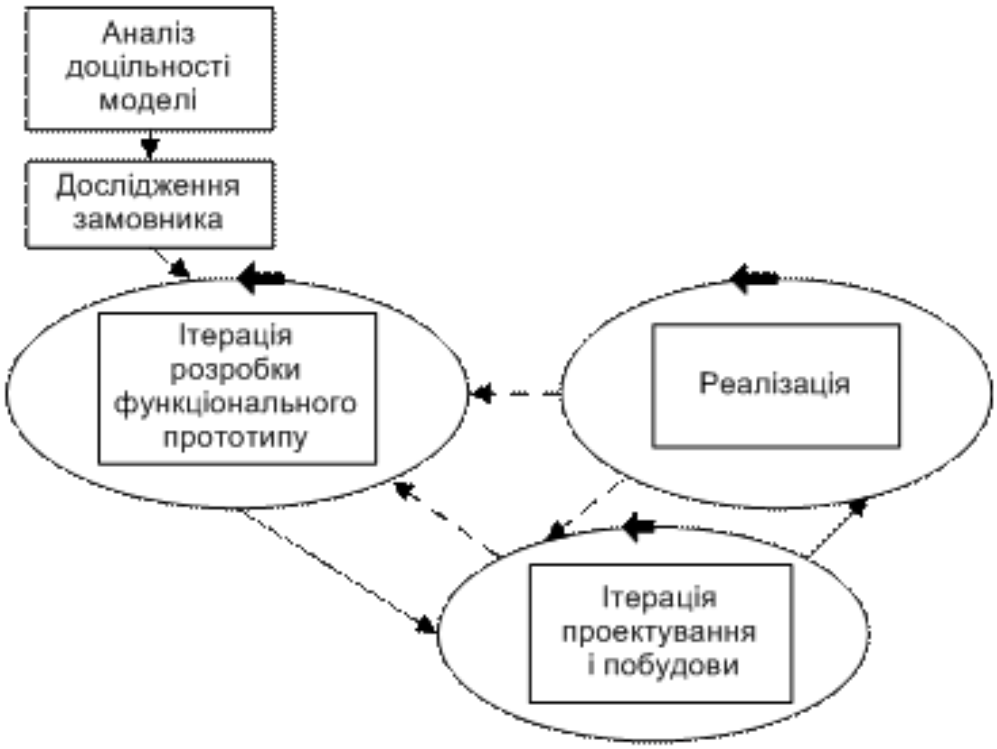

Рисунок 1.4.1 – Принцип роботи еволюційної моделі

Ця модель виконує дії, що дозволяють провести глибокий аналіз предметної області і застосувати його до конкретної системи, окрім цього є можливість підтримувати зв’язок з замовником, щоб більш детально визначити потреби користувачів про створенні прототипів на етапі ітерацій.

Дана модель має два головних етапи, або їх ще називають ітераціями. Починається все з проектування, яке далі перетікає в реалізацію, що в свою чергу дозволяє перевірити чи ПЗ задовольняє усі вимоги. Все зводиться до поступової реалізацію усього функціоналу через прототипи. Кожен наступний прототип доповнює попередній, що в свою чергу допомагає реалізувати увесь функціонал системи.

Переважно відбувається декілька ітерацій, під час яких створюється прототипи з додатковим функціоналом і заново моделюється робота. Це відбувається до тих пір поки не буде протестовано та реалізовано увесь функціонал. Завершенням є фінальна ітерація – отримання готової версії продукту [6].

Дана модель має місце у системах з важливими функціональними можливостями, де потрібно максимально швидко продемонструвати функціонал.

Враховуючи те, що проміжні варіанти системи уже реалізовують певний функціонал, виникає можливість їх перевірки, тестування та супроводу вже під час експлуатації даного прототипу, це означає, що разом з розробкою нової версії ПЗ відбувається тестування частини функціоналу попереднього прототипу..

Однак, варто враховувати наступні ризики:

– можливе нагромаджування коду, даних та версій системи через одночасну реалізацію усіх функцій;

– даний метод є досить затратним в плані часу і потребує велику кількість людей та ресурсів, які в свою чергу будуть зайняті над проектом впродовж досить тривалого часу;

– в певних випадках дороговизна.

Дана модель має наступні переваги:

– висока швидкість реалізації частини функціоналу та їх тестування;

– можливість використовувати попередній прототип у наступній ітерації;

– відокремлення та реалізація частини функціоналу у вигляді готового прототипу;

– при потребі можна розширити систему про збільшенні фінансових надходжень;

– повторне та декілька разове узгодження вимог, без впливу на якість кінцевого продукту;

– можливість легше вносити зміни при реалізації окремих функцій.

Також для даної моделі ключову роль займає захист системи та даних, власне в цьому напрямку вона і розвивається [6].

1.5 Вибір технологій та інструментів для розробки

1.5.1 Мова програмування C#

Для написання магістерської роботи було обрано мову програмування C#. Саме ця мова ідеально підходить для розробки додатків на ПК і має для цього всі необхідні та зручні робочі інструменти.

C # - строго типізована об'єктно-орієнтована мова програмування. Вона являє собою відкритий код, простий, сучасний, гнучкий та універсальний.

Мова програмування - це мова, яка використовується для написання програмних програм.

C # - мова програмування, розроблена та запущена Microsoft у 2001 році, проста, сучасна та об'єктно-орієнтована мова, яка забезпечує сучасну розробку гнучкістю та можливостями для створення програмного забезпечення, яке не тільки працюватиме сьогодні, але застосовуватиметься роками у майбутньому. Вона є проста, сучасна та об'єктно-орієнтована мова програмування [1].

Ключові характеристики мови C # включають:

– Сучасна і легкий;

– Швидкий та відкритий код;

– Мова є кросплатформною;

– Безпечна;

– Універсальна;

– Постійно розвивається.

Якщо подивитися на історію мов програмування та їх особливості, кожна мова програмування була розроблена з певною метою, щоб вирішити конкретну потребу в той час.

Однак мова C # була створена для того, щоб вирішувати потреби бізнесу та підприємств. Мова була розроблена для створення всіх видів програмного забезпечення за допомогою однієї єдиної мови програмування.

Дана мова забезпечує функціональність для підтримки сучасної розробки програмного забезпечення. C # підтримує розробку веб, мобільних додатків та програм для ПК.

Деякі з сучасних функцій мови програмування підтримують дженеріки, типи var, автоматичну ініціалізацію типів і колекцій, лямбда-вирази, динамічне програмування, асинхронне програмування, кортежі, відповідність шаблонів, вдосконалене налагодження та обробку виняткових ситуацій та інше.

На синтаксис мови впливають C ++, Java, Pascal та кілька інших мов, які легко прийняти. C # також уникає складності та неструктурованих особливостей мови.

C # - безпечна мова типу, вона не дозволяє перетворення типів, які можуть призвести до втрати даних або інших проблем, саме це дозволяє розробникам писати безпечний код, а також зосереджується на написанні ефективного коду [8].

Також важливим є програмування між платформами. Можливо створювати .NET програми, які можна розгорнути на платформах Windows, Linux та Mac, а також можна розгорнути у хмарі та контейнерах.

Ось перелік деяких ключових моментів у C #, які допомагають писати безпечний та ефективний код.

- Злиття небезпечних типів заборонене;.
- Підтримка даних типу Null;
- Використовує повернення з типом «тільки для читання», що дозволяє використовувати копії об’єктів підчас компіляції;
- Коли розмір структури, що зчитується більший, ніж IntPtr.Size, її слід передати її як параметр з міркувань продуктивності.

– Ніколи не передає структуру як параметр, якщо вона не оголошена модифікатором readonly, оскільки це може негативно вплинути на продуктивність і може призвести до незрозумілої поведінки.

Також дана мова є універсальною. Хоча більшість мов програмування розроблені з певною метою, C # був розроблений для майже всього. Можна використовувати для створення сучасних програмних систем, для розробки всіх видів програм, включаючи клієнтські, компоненти та бібліотек, сервіси та API, веб-додатки, мобільні додатки, хмарні програми та відеоігри [9].

Ось список типів додатків, які C # може скласти:

– Клієнтські програми Windows;
– Бібліотеки та компоненти Windows;
– Послуги Windows;
– Веб-додатки;
– Веб-сервіси та веб-API;
– Native iOS та мобільні додатки для Android;
– Сервіси з вихідного дня;
– Хмарні програми та сервіси Azure;
– Резервна база даних з використанням засобів ML / Data;
– Штучний інтелект та машинне навчання;

1.5.2 Фреймворк .Net

Фреймворк .Net - це платформа розробки програмного забезпечення, розроблена Microsoft. Фреймворк призначений для створення додатків, які працюватимуть на платформі Windows. Перша версія фреймворку .Net була випущена в 2002 році.

Версія отримала назву .Net Framework 1.0. З цього часу .Net пройшов довгий шлях, а поточна версія - 4.7.1.

Він може використовуватися для створення як програм на основі форм, так і на веб-основі. Веб-сервіси також можуть бути розроблені з його допомогою [1].

Основна архітектура .Net фреймворку наведена нижче на рисунку 1.5.1.

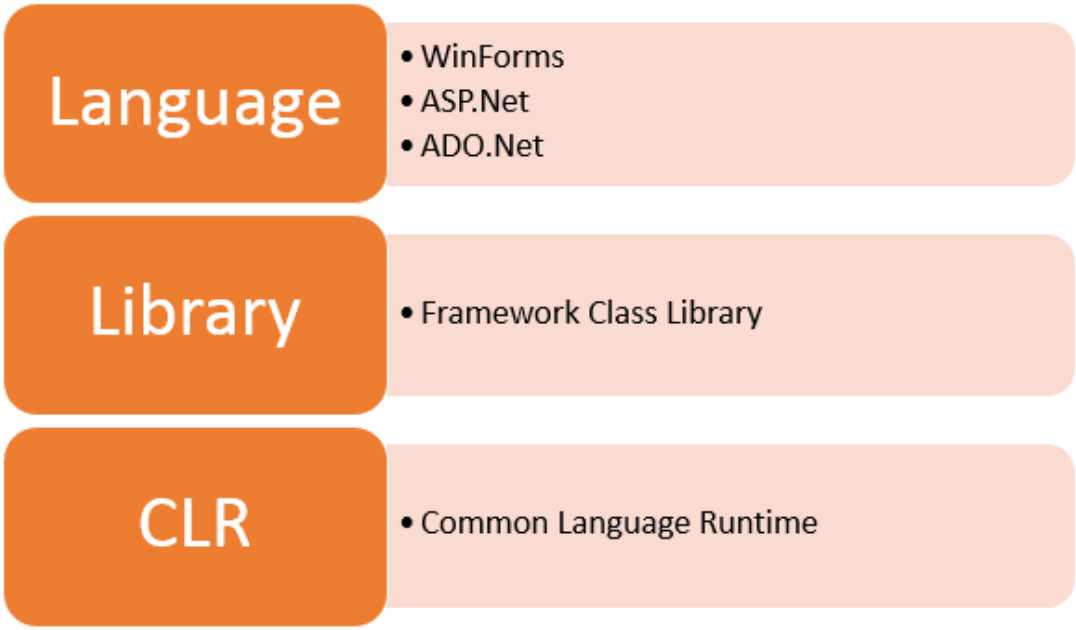

Рисунок 1.5.1 – Основна архітектура .Net фреймворку

Архітектура .Net Framework базується на саме на цих ключових компонентах.

Common Language Infrastructure або "Загальна мовна інфраструктура", також відома як CLI - це платформа, на якій виконуються програми .Net.

CLI має такі основні особливості:

– Обробка винятків - Винятки - це помилки, які виникають під час виконання програми.

Приклади винятків:

- Якщо програма намагається відкрити файл на локальній машині, але файл відсутній;
- Якщо програма намагається отримати деякі записи з бази даних, але підключення до бази даних недійсне.

– Garbage Collection- збирання сміття - це процес видалення небажаних ресурсів, коли вони більше не потрібні.

Приклади збору сміття є

- Файл який більше непотрібен. Якщо програма закінчила всі операції над файлом, обробка файлів більше не потрібна;
- Підключення до бази даних більше не потрібне. Якщо програма закінчила всі операції з базою даних, можливо, більше не потрібно буде з'єднання з базою даних і для оптимізації пам’яті воно закривається [8].

– Робота з різними мовами програмування.

Як зазначалося вище, розробник може розробляти додаток на різних мовах програмування .Net.

1. Мова – це перший рівень, тобто сама мова програмування, найпоширеніші - VB.Net та C #.
2. Компілятор - який буде окремим для кожної мови програмування. Отже, що лежить в основі мови VB.Net - це буде окремим компілятором VB.Net. Аналогічно, для C # передбачається інший компілятор який більш оптимізований під конкретну мову.
3. Інтерпретатор загальної мови - це останній рівень в .Net, який буде використовувався для запуску програми .net, розробленої на будь-якій мові програмування. Тож наступний компілятор відправить програму на рівень CLI для запуску програми .Net.

На рисунку 1.5.2 продемонстровано ключові компоненти CLI. Можна побачити, що спочатку відбувається оптимізація під конкретну мову програмування з усіма її особливостями, а вже потім отримується доступ через загальний інтерпритатор [9].

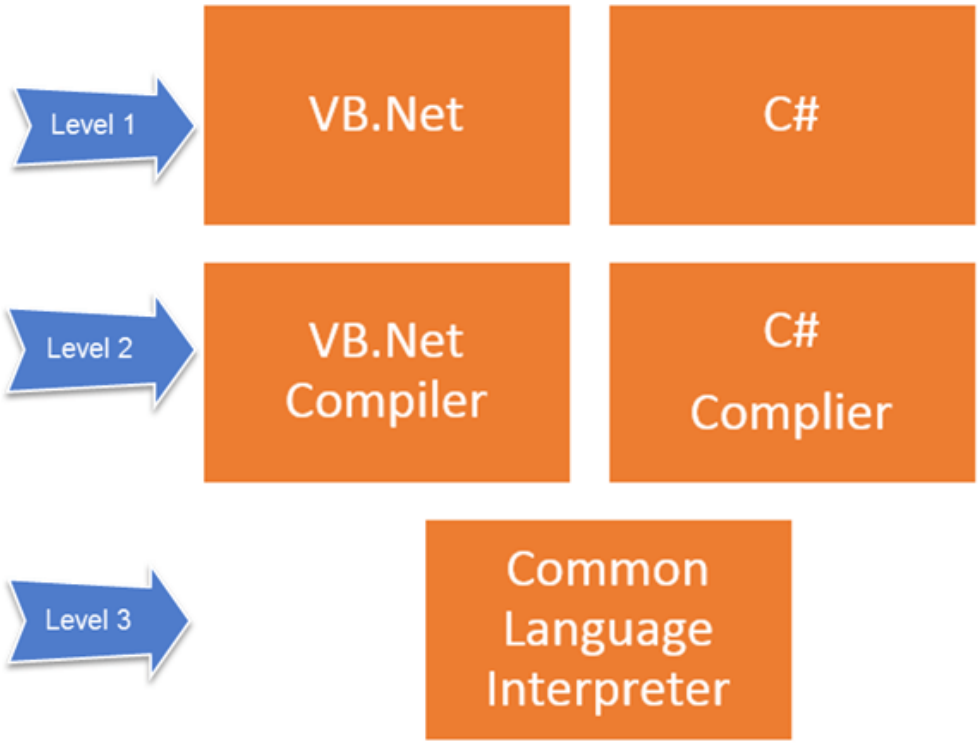

Рисунок 1.5.2 – Особливості CLI

Class Library або Бібліотека класів. Окрім вище сказаного .NET Framework включає набір бібліотек стандартних класів. Бібліотека класів - це сукупність методів та функцій, які можна використовувати для розробки.

Наприклад, є бібліотека класів з методами для обробки всіх операцій на рівні файлів.

Отже, існує метод, який можна використовувати для читання тексту з файлу. Так само існує метод запису тексту у файл.

Languages або мови. Типи програм, які можна вбудувати в .Net Framework, класифікуються в основному на наступні категорії:

- WinForms - використовується для розробки додатків на основі форм, які працюватимуть на машині кінцевого користувача. Блокнот - це приклад клієнтської програми;
- ADO.Net - Ця технологія використовується для розробки програм для взаємодії з базами даних, такими як Oracle або Microsoft SQL Server;
- ASP.Net - використовується для розробки веб-додатків, які запускаються в будь-якому браузері, наприклад, Internet Explorer, Chrome або Firefox [1].

Дану категорію умовно можна поділити ще на:

- Веб-додаток який оброблятиметься на сервері, на якому буде встановлено Інтернет-інформаційні послуги.
- Internet Information Services або IIS - це компонент Microsoft, який використовується для виконання програми Asp.Net.
- Результат виконання надсилається на клієнтські машини, а вихід відображається в браузері.

Microsoft завжди гарантує, що .Net співпрацює з усіма підтримуваними операційним системам Windows.

Окрім всього вище сказаного .Net Framework має певні принципи поведінки та дизайну.

Наступні принципи роблять даний фреймворк дуже актуальним для створення додатків на його базі:

1) Підтримка попередніх версій - .Net забезпечує підтримку попередніх версій платформи. Припустимо, якщо був додаток, побудований на старій версії, скажімо 2.0. Але якщо спробували запустити ту саму програму на машині, яка мала вищу версію фреймворку, 3.5 або вище наприклад, то додаток однаково запрацює. Це пов’язано з тим, що з кожним випуском Microsoft гарантує, що старі версії добре поєднуються з найновішою версією.
2) Портативність. Програми, побудовані на основі .Net, можна запустити на будь-якій платформі Windows. А останнім часом Microsoft також працює над тим, щоб їх продукти працювали на інших платформах, таких як iOS та Linux.
3) Управління пам'яттю - CLI виконує всю роботу по управління пам'яттю. Дана система має всі можливості бачити ті ресурси, які не використовуються запущеною програмою. Тоді вони б відповідно звільнили ці ресурси. Це робиться за допомогою програми під назвою "Збір сміття", яка працює як частина загально механізму та забезпечує цілісність роботи.

4) Безпека – даний фреймворк має хороший механізм захисту. Вбудований механізм допомагає як при верифікації, так і для перевірки програм. Кожна програма може чітко визначити свій механізм захисту. Кожен механізм захисту використовується для надання користувачеві доступу до коду або до запущеної програми.
5) Спрощене розгортання – окрім цього передбачені інструменти, які можна використовувати для упаковки програм, побудованих на даній платформі. Потім ці файли можуть бути розповсюджені на клієнтських машинах, що дозволяю спростити їх встановлення.

Підсумовуючи вище сказане, .Net - фреймворк, створений Microsoft. Він був розроблений для створення додатків, які могли би працювати на платформі Windows. Дана платформа .Net може використовуватися для розробки додатків на Windows forms, веб-додатків та веб-служб. Розробники можуть вибирати з різних мов програмування, доступних на платформі .Net. VB.Net і C# найпоширеніші з них [8].

1.5.3 Windows Forms

Windows Forms - це набір керованих бібліотек в .NET Framework, призначений для розробки клієнтських додатків. Це графічний API для відображення даних та керування взаємодією користувачів з простішим розгортанням та кращою безпекою в клієнтських додатках.

Windows Forms пропонує широку клієнтську бібліотеку, що надає інтерфейс для доступу до власних елементів графічного інтерфейсу Windows та графіки з керованого коду. Він побудований з керованою подіями архітектурою, подібною клієнтам Windows і його програми чекають на ведення даних користувачем для їх обробки та виконання.

Windows Forms подібний до бібліотеки Microsoft Foundation Class (MFC) при розробці клієнтських додатків. Він пропонує обгортку, що складається з набору класів C ++ для розробки програм Windows. Однак він не забезпечує увесь функціонал за замовчуванням, як MFC [2].

Кожен елемент управління в додатку Windows Forms - це конкретний екземпляр класу. Макетом управління в графічному інтерфейсі та його поведінкою керуються за допомогою методів. Windows Forms надає різноманітні елементи керування, такі як текстові поля, кнопки та веб-сторінки, а також параметри створення спеціальних елементів керування. Він також містить класи для створення пензлів, шрифтів, піктограм та інших графічних об'єктів (наприклад, лінії та кола) [2].

Windows Forms Designer - це інструмент, створений для встановлення певних елементів, що керують формою та упорядковують їх відповідно до потрібного стилю або макету з можливістю удосконалювати код для обробки різноманітних подій, які реалізують взаємодію користувачів. Табличні дані, які пов'язані з XML, базою даних тощо, можуть відображатися за допомогою керування DataGrid View у вигляді рядків і комірок.

Налаштування програми - це ще одна особливість Windows Forms для створення, зберігання та підтримки інформації про стан виконання у формі XML, яка може бути використана для отримання бажаних налаштувань користувача, таких як позиції панелі інструментів та останні використані списки. Ці параметри можна повторно використовувати в майбутньому додатку [15].

Деякі найкращі механізми створення програм Windows Forms включають:

- Класи Windows Forms можна розширити, використовуючи успадкування, щоб створити програму, яка може забезпечити високий рівень абстрагування та використання коду;
- Форми повинні бути компактними, при цьому елементи керування обмежені розміром, який може запропонувати мінімальну функціональність.
- Крім того, створення та видалення елементів керування динамічно може зменшити кількість статичних елементів керування;

– Форми можна розбити на шматки, упаковані у склади, які можуть автоматично оновлюватись і з ними можна легко керувати з мінімальними зусиллями;
– Проектування програми без стану забезпечує масштабність та гнучкість з легкістю для налагодження та обслуговування;
– Програми Windows Forms повинні бути розроблені виходячи з необхідного рівня довіри, при потребі запитувати дозволи та обробляти винятки там де це необхідно;
– Форму Windows не можна передавати через межі домену додатків, оскільки вони не розроблені для того.

Система презентації Windows (WPF) - це найновіша технологія візуалізації користувальницьких інтерфейсів у програмах Windows GUI з такими функціями, як підтримка 2D / 3D, інтерактивна візуалізація даних та читаність вмісту. Він покладається на DirectX, а не на GDI (графічний інтерфейс пристрою) для надання моделі програмування, де інтерфейс користувача відокремлений від бізнес-логіки. Однак, маючи можливість взаємодії з WPF (де це потрібно), Windows Forms є хорошим вибором для додатків, які не потребують багатогалузевого графічного інтерфейсу та інших функцій WPF, таких як шаблони даних / керування, типографічні та функції передачі тексту [2]. На рисунку 1.5.3 зображено вікно конструктора.

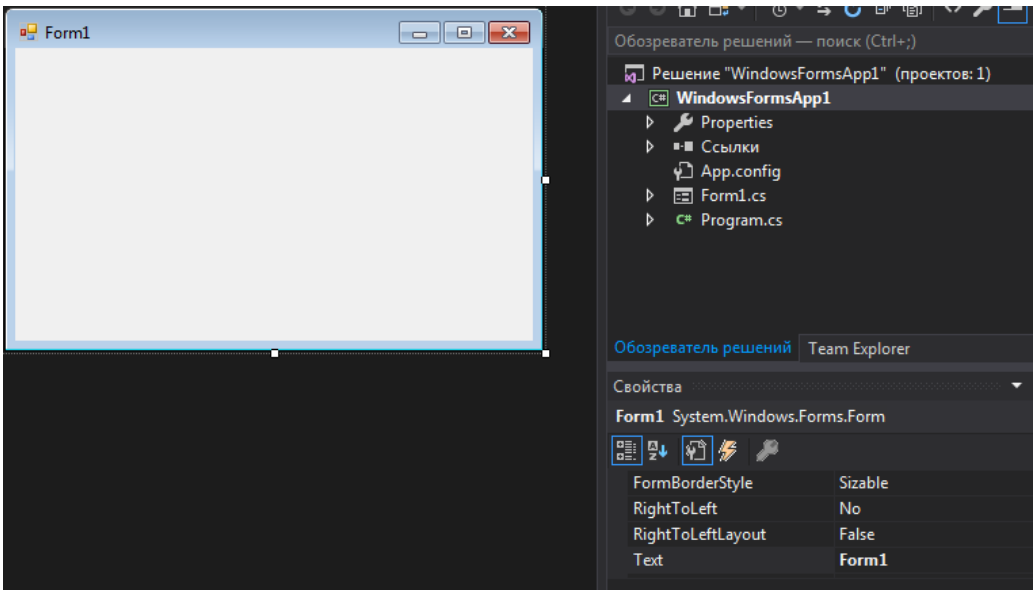

Рисунок 1.5.3 – Конструктор Windows Forms

1.5.4 Бібліотека IronBarcode

IronBarcode дозволяє розробникам читати та записувати штрих-коди та QR-коди в .Net додатках та веб-сайтах. Для читання або запису штрих-кодів потрібен єдиний рядок коду з цієї бібліотеки.

Бібліотека штрих-кодів .Net читає і записує більшість стандартів штрих-коду та QR-кодів. Сюди входять код 39/93/128, UPC A / E, EAN 8/13, ITF, RSS 14, панель даних CodaBar, Aztec, Матриця даних, MaxiCode, PDF417, MSI, Plessey, USPS та QR. Дані результату штрих-коду включають тип, текст, двійкові дані, сторінку та файл зображення.

Набір методів бібліотеки для читання штрих-коду включає автоматичну корекцію зображення та технологію виявлення коду, щоб усунути неточності від локалізації та зчитування з недосконалих сканів. Багатопотокове сканування, обрізка та пакетне сканування забезпечує швидке та якісне сканування документів на декількох сторінках.

API перевірки штрих-коду перевіряє та підтверджує формат, довжину, кількість, контрольну суму, щоб автоматично уникнути помилок кодування. Створення штрих-коду дозволяє створювати стилі, змінювати розмір, поля, межі, перефарбовувати та додавати текстові анотації. Записувати у форматі зображення, PDF чи HTML-файл.

Основні функції бібліотеки включають:

- Читання одиничних чи декількох штрих-кодів та QR-кодів із зображень чи PDF-файлів;
- Корекція зображення для косого, орієнтаційного, зашумленого, низької роздільної здатності, контрастного тощо;
- Створення штрих-коди для зображень чи PDF-документів;
- Вставити штрих-коди в html-документи;
- Стилі штрих-кодів та додавання текстових анотації;
- Написання QR-коду дозволяє додавати логотипи, кольори та розширене вирівнювання QR.

IronBarcode можна використовувати в проектах C #, VB.NET, ASP .NET, MVC, веб-сервісах, консолях та настільних програмах [13].

1.6 Середовище розробки Visual Studio

Інтегроване середовище розробки Visual Studio - це креативний інструмент, який можна використовувати для редагування, налагодження, створення програмних продуктів та їх публікації. Інтегроване середовище розробки (IDE) - це багатофункціональна програма, яка може бути використана для різноманітних напрямків розробки програмного забезпечення. Окрім стандартного редактора та налагоджувача, який надає більшість IDE, тут ще включені компілятори, інструменти для форматування та доповнення коду, графічні середовища та інші функції які значно полегшують процес розробки програмного забезпечення [5].

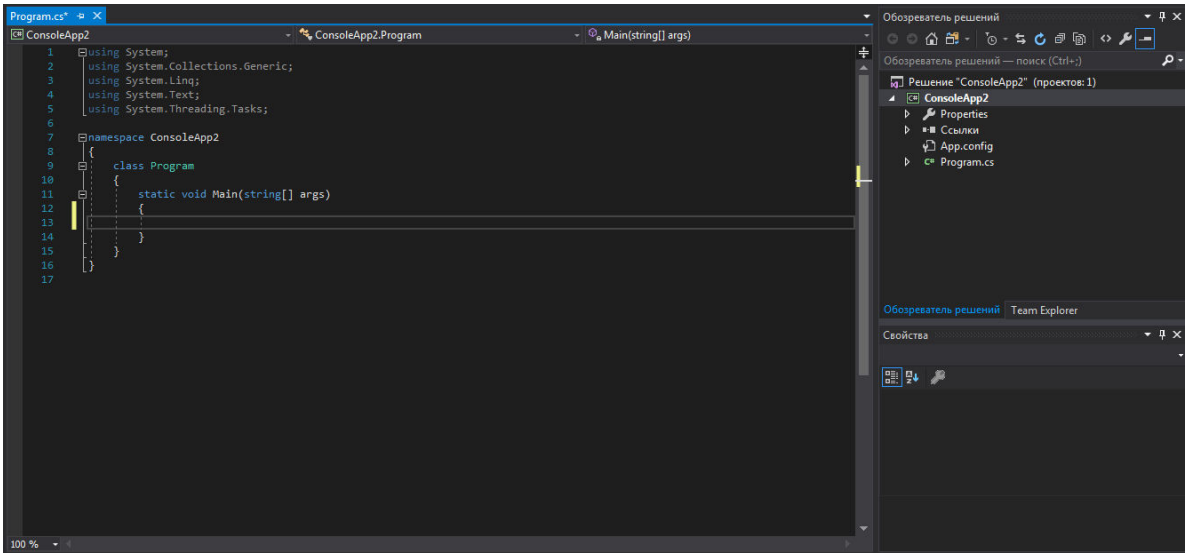

Рисунок 1.6.1 – середовище розробки Visual Studio

На рисунку 1.6.1 зображено Visual Studio з відкритим проектом та кількома ключовими вікнами інструментів, Solution Explorer (угорі праворуч) дозволяє переглядати, переміщуватися та керувати файлами коду. Провідник рішень може допомогти впорядкувати код, згрупувавши файли у рішення та проекти.

- Editor window (в центрі) місце де проводиться більша частина свого часу, відображається вміст файлу.
- Team Explorer (внизу праворуч) дозволяє відстежувати робочі елементи та ділитися кодом з іншими за допомогою технологій контролю версій, таких як Git та Team Foundation (TFVC).

Visual Studio доступний для Windows та Mac. Visual Studio для Mac має багато тих же функцій, що й для Windows і оптимізований для розробки кросплатформних та мобільних додатків.

Деякі з популярних функцій Visual Studio, які допомагають бути більш продуктивними під час розробки програмного забезпечення включають в себе наступні функції [15].

Підкреслення та швидкий доступ. Підкреслення - попереджають про помилки або потенційні проблеми в коді під час введення. Ці візуальні підказки дозволяють негайно виправити проблеми, не чекаючи виявлення помилки під час компіляції або під час запуску програми. Якщо навести курсор миші на клавішу можна побачите додаткову інформацію про помилку. Лампочка може також з'являтися в лівому полі з діями, появиться швидкий доступ, щоб виправити помилку, що власне і продемонстровано на рисунку 1.6.2.

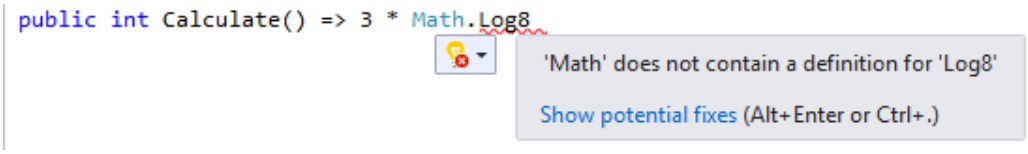

Рисунок 1.6.2 – функція підкреслень та швидкого достопу

Наступною корисною функцією є рефакторинг. Він включає в себе такі операції, як інтелектуальне перейменування змінних, автоматичне перенесення однієї чи декількох рядків коду в новий метод, зміна порядку параметрів методу, або інші корисні речі. Саме це зображено на рисунку 1.6.3.

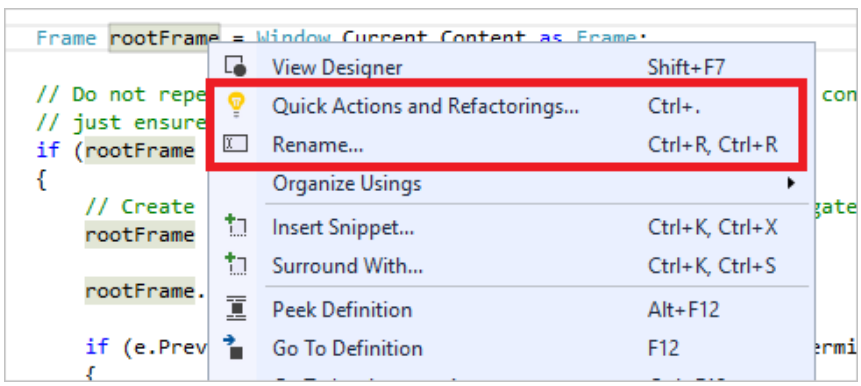

Рисунок 1.6.3 – функція рефакторингу

IntelliSense - це термін для набору функцій, які відображають інформацію про код безпосередньо в редакторі і в деяких випадках пише невеликі шматочки коду замість розробника. Це як би мати в редакторі основну документацію, що позбавляє від необхідності шукати інформацію в інших місцях. Функції IntelliSense залежать від мови. Ілюстрація на рисунку 1.6.4 показує, як IntelliSense відображає список членів для типу.

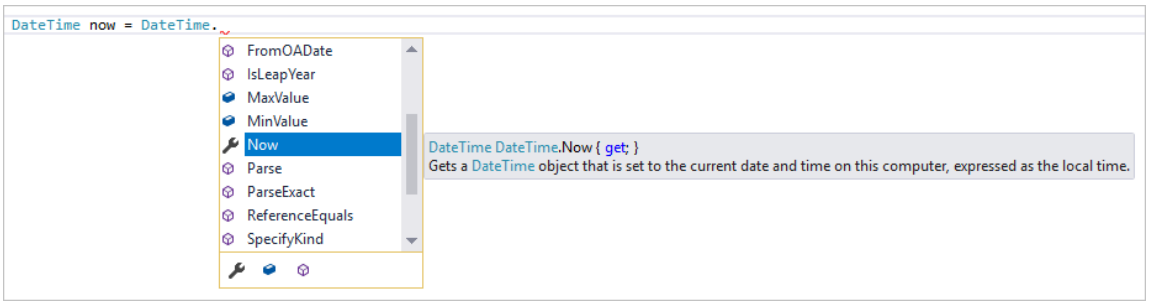

Рисунок 1.6.4 – функція IntelliSense

CodeLens – функція, що допоможе знайти посилання на потрібний код та змінити його на інший, окрім цього можливо знайти пов'язані помилки, робочі елементи, огляди коду та тести одиниць, і все це не виходячи з редактора. Приклад зображено на рисунку 1.6.5.

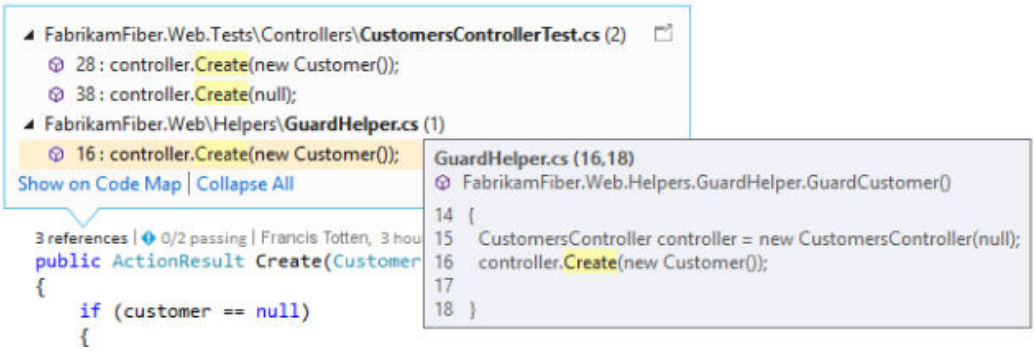

Рисунок 1.6.5 – функція CodeLens

Окрім вище сказаного, існує можливість спільно редагувати та налагоджувати налаштування з іншими користувачами в режимі реального часу, незалежно від типу обраного додатка чи мови програмування. Присутня можливість миттєво та надійно ділитися своїм проектом і при потребі проводити налагодження сесій, веб-додатків, голосових дзвінків та іншого.

Враховуючи все описане вище, дане середовище ідеально підходить для розробки додатку на мові програмування C# [5].

1.7 База даних MySql

Для збереження даних про товари в АСУ необхідний механізм, який може зберігати та обробляти велику кількість даних. Саме з цією метою було обрано бази даних, а саме MySql.

MySQL - реляційна база даних з відкритим кодом, яка є кросплатформною, що означає, що вона працює на багатьох різних платформах, таких як Windows, Linux, Mac OS [3].

На ринку існує багато систем управління реляційними базами даних. Приклади реляційних баз даних можуть слугувати Microsoft SQL Server, Microsoft Access, Oracle, DB2.

Вибір цієї БД випав через ряд факторів:

- Підтримує декілька механізмів зберігання даних, кожен з яких має свої специфіки, тоді як інші системи, такі як SQL-сервер, підтримують лише один механізм зберігання даних;
- Має високу продуктивність порівняно з іншими системами баз даних відношень. Це пов'язано з його простотою в дизайні та підтримці двигунів з декількома сховищами;
- Економічна вигода, порівняно з іншими реляційними базами даних, ця порівняно дешева. Насправді загальна версія безкоштовна. Комерційне видання має плату за ліцензування, яка також є економічно вигідною порівняно з платою за ліцензування таких продуктів, як Microsoft SQL Server;
- Працює на багатьох платформах, що означає, що вона може бути розгорнута на більшості машин. Інші системи, такі як MS SQL Server, працюють лише на платформі Windows;
- Має відкритий код, що означає наявність можливостей для зміни вихідного коду. Встановити програмне забезпечення може кожен. Також можна дізнатися та налаштувати вихідний код, щоб краще відповідав потребам конкретно обраної області, що в свою черго значно покращує ефективність системи [11].

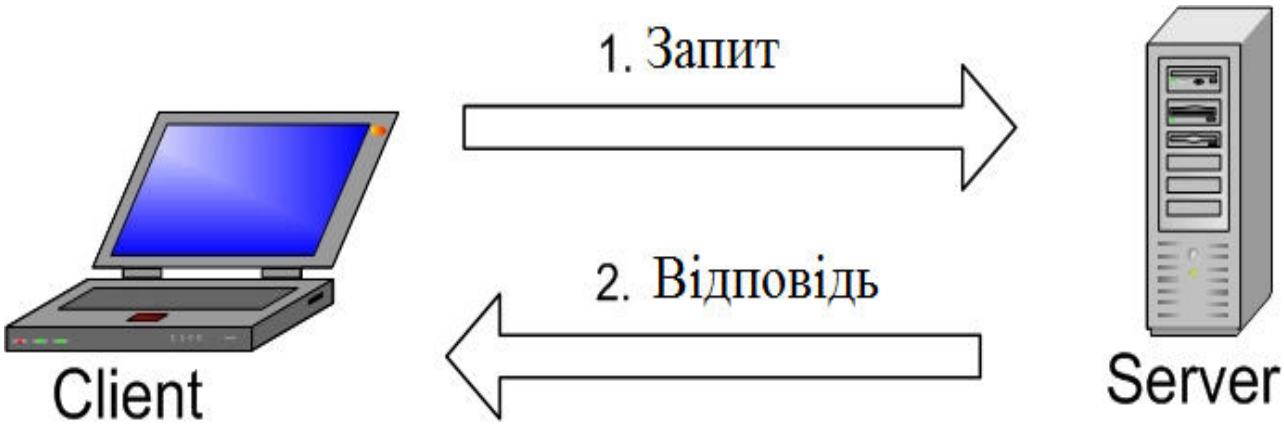

Рисунок 1.7.1 – принцип роботи бази даних

Рисунок 1.7.1 ілюструє основну принцип роботи клієнт-сервер. Один або кілька пристроїв (клієнтів) підключаються до сервера через певну мережу. Кожен клієнт може зробити запит із графічного інтерфейсу користувача (GUI) на своїх екранах, і сервер видасть потрібний результат, доки обидві сторони розуміють інструкцію [3]. Не вдаючись сильно в подробиці конкретної реалізації, головні етами MySQL:

1. MySQL створює базу даних для зберігання та обробки даних, визначаючи взаємозв'язок кожної таблиці;
2. Клієнти можуть робити запити, ввівши конкретні запити SQL на MySQL;
3. Серверна програма відповість на запит та відправляє відповідь на сторону клієнта.

З боку клієнтів зазвичай підкреслюється, яке саме Це майже все. З боку клієнтів вони зазвичай підкреслюють, яким MySQL GUI використовувати. Чим легший і зручніший користувальницький інтерфейс, тим швидше і простіше буде їх діяльність з управління даними.

Одним з найпопулярніших графічних інтерфейсів MySQL є MySQL WorkBench, SequelPro, DBVisualizer та інструмент адміністрування DB Navicat. Деякі з них безкоштовні, а інші - комерційні, деякі працюють виключно для macOS, а деякі сумісні з основними операційними системами. Клієнти повинні вибирати графічний інтерфейс залежно від своїх потреб та апаратних можливостей [11].

Також для оптимізації роботи БД їх нормалізують. Нормалізація - це процес, за допомогою якого великі таблиці поділяються на менші таблиці та визначаються відносини між ними. Він має дві переваги:

- Усуває зайві дані (тобто повторення одних і тих же даних у кількох записах).
- Покращує продуктивність вашої бази даних.

Наприклад, керуючи клієнтами та їхніми замовленнями, неефективно повторювати дані клієнтів з кожним замовленням. Після нормалізації створюються дві таблиці, одна для клієнтів (батьківська таблиця) та друга для замовлень (дочірня таблиця). Первинний ідентифікатор у батьківській таблиці та його відповідний зовнішній ідентифікатор у дочірній таблиці використовуються для формування зв’язку між двома таблицями.

Налаштовуючи зв’язки між пов’язаними таблицями, можна по суті працювати з ними так, ніби вони були однією таблицею. Присутня можливість виконувати запити між об’єднаними таблицями або створювати представлення даних, де автоматично визначаються певні параметри подання, такі як поля приєднання та тип приєднання, і все це не приносить шкоди нормалізації. Взаємозв'язки складаються шляхом об'єднання поля з однієї таблиці до поля з іншої таблиці. Поля, що використовуються у відносинах, повинні бути сумісних типів, інакше взаємозв’язок неможливий. У таблиці нижче показана сумісність відносин між типами даних [17] .

Data Types	Relationship Compatibility
Integer	Integer, Autonumber
Autonumber	Autonumber, Integer
Number	Number
Currency	Currency
Text (255)	Text (255), Prefixed Autonumber, Random ID, GUID
Prefixed Autonumber	Prefixed Autonumber, Text (255)
Random ID	Random ID, Text (255)
GUID	GUID, Text (255)
Date/Time	Date/Time, Timestamp
Timestamp	Timestamp, Date/Time

Рисунок 1.7.2 – сумісні типи в MySql

1.8 MySQL WorkBench

Оскільки працювати напряму з БД досить незручно та ресурсо затратно, хорошим рішенням було використати графічний інтерфейс для роботи з БД. Для цих цілий було обрано MySQL WorkBench. MySQL Workbench - це інструмент для проектування та моделювання візуальних баз даних для реляційної бази даних сервера MySQL. Це полегшує створення нових фізичних моделей даних та модифікацію існуючих баз даних з інженерною підтримкою попередніх та наступних версій, допомагає в керуванні змінами, рисунок 1.8.1.

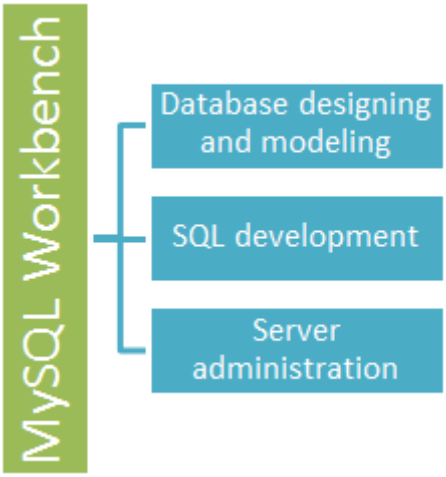

Рисунок 1.8.1 – принцип роботи бази даних

Моделі лежать в основі більшості дійсних і високопродуктивних баз даних. У MySQL Workbench є інструменти, які дозволяють розробникам та адміністраторам баз даних візуально створювати фізичні моделі дизайну баз даних, які можна легко перевести в бази даних MySQL за допомогою звичайного перетягування мишкою або вбудованих функцій. Окрім цього підтримує створення декількох моделей в одному середовищі. Підтримуються всі об'єкти, такі як таблиці, представлення даних, збережені процедури, тригери, власне все те, що складає базу даних [4].

Окрім цього, підтримується вбудована утиліта для перевірки моделі, яка повідомляє про будь-які проблеми, які можуть бути знайдені при моделювання даних. Він також дозволяє застосовувати різні позначення моделювання і може бути розширений за допомогою мови скриптів LUA.

Structured Query Language (SQL) або структурована мова запитів дозволяє маніпулювати нашими реляційними базами даних. SQL лежить в основі всіх реляційних баз даних.

Перелік найкорисніших функцій та інструментів:

1. MySQLworkbench, має вбудований візуальний редактор SQL;
2. Редактор Visual SQL дозволяє розробникам створювати, редагувати та запускати запити баз даних MySQL-сервера. У ньому є утиліти для перегляду даних та їх експорту;
3. Підсвічування синтаксису кольорами допомагає розробникам легко писати та налагоджувати запити SQL;
4. Можна запустити кілька запитів, а результати відображатимуться автоматично на різних вкладках;
5. Запити також зберігаються на панелі історії для подальшого пошуку та запуску.

Адміністрація сервера відіграє вирішальну роль у забезпеченні безпеки даних системи. Основними проблемами, що стосуються адміністрування сервера, є управління користувачами, налаштування самого сервера, журналу сервера та багато іншого [16]. Workbench MySQL має такі функції, які спрощують процес адміністрування сервера, до їх переліку можна віднести:

- Адміністрація користувачів - візуальна утиліта для керування користувачами, яка дозволяє адміністраторам баз даних легко додавати нових та видаляти існуючих користувачів;
- Конфігурація сервера - дозволяє розширити конфігурацію сервера і провести тонку настройку для досягнення оптимальної продуктивності;
- Резервне копіювання та відновлення баз даних - візуальний інструмент для експорту / імпорту файлів MySQL. Файли дампів містять сценарії SQL для створення баз даних, таблиць, представлень;

2. ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Виявлення ризиків при проектуванні

Розробка будь-якої системи, незалежно від складності, несе в собі певні ризики, які завжди потрібно враховувати як розробнику і замовнику [14]. В таблиці 2.1.1 описані ключові ризики розробки АСУ мінімаркету.

Таблиця 2.1.1 – Ризики при розробці ПЗ

Категорії ризиків	Ризики та їх приклади
Ризики технологічного характеру	Несправності апаратного забезпечення робочих машин персоналу; Збій у роботі бази даних при обробці запитів від користувачів; Не можливе повторне використання раніше розроблених модулів чи компонент в подібних задачах.
Ризики організаційного характеру	Не правильно сформований бюджет для розробки проекту; Перед тим як почати розробку не були остаточно ухвалені усі потреби і побажання замовника; Замовник змінює орієнтацію проекту в іншу сферу. Не можливість знайти хостинг для сайту.
Персонал та ризики пов’язані з його наймом	Кількість набраного персоналу є малою для того, щоб розпочати реалізацію поставленого проекту; Труднощі у підборі висококваліфікованих працівників; Основні технології побудови програмної системи заміняються новими Неможливо реалізувати навчання персоналу.
Бізнес-Ризик	На ринку програмних продуктів до закінчення проекту з'явилася конкуруюча програмна система

2.2 Пошук акторів системи

Розробка будь-якої системи вимагає детального аналізу та розуміння не лише програмних засобів, але і предметної області в якій ведеться розробка [10]. Проаналізувавши предметну область та усі функціональні, нефункціональні вимоги, можна виділи двох чітких акторів в системі. Цими акторами є касир та адміністратор, рисунок 2.2.1.

Рисунок 2.2.1 – головні актори системи

Саме ці актори виступають головними користувачами АСУ мінімаркетом. В таблиці 2.2.1 наведено ключових акторів та описано їх головну роль в системі.

Таблиця 2.2.1 - Виявлення акторів

Актор	Короткий опис
Касир	Головний користувач програми який має змогу виконувати пошук товарів в системі по штрих-коді або по звичайному цифровому коді і додавати ці товари в чек. Основною його задачею є обслуга клієнтів. Окрім цього також може виконувати в системі пошук персоналізованих дисконтних карток.
Адміністратор	Головний користувач програми який має змогу керувати даними в системі, а також додавати та видаляти в системі користувачів, відвідаю за надання прав та отримує доступ до звітності.

2.3 Виявлення основних варіантів використання

Окрім визначення акторів в системі, також необхідно встановити варіанти використання для кожного окремого користувача. Саме це описано в таблиці 2.3.1.

Таблиця 2.3.1 - Опис варіантів використання

К1	Касир	Авторизація в системі	Касир має змогу авторизуватися в системі для початку роботи з нею, а також завершити сеанс.
К2	Касир	Застосувати дисконтну карту	Підчас продажу товарів, касир маю змогу застосувати дисконтну карту клієнта, при її наявності.
К3	Касир	Змінити кількість товарів в чеку	Якщо товар відскановано і необхідно змінити його кількість в чеку, касир редагує дані в графі «Кількість» без потреби повторного сканування.
К4	Касир	Пошук товару в системі	При необхідності, користувач може знайти необхідний товар в системи за певними критеріями
К5	Касир	Додати товар в чек	Користувач додає товар в чек. Це можна зробити декількома способами: відсканувати штрих-код, знайти товар по назві або по номері.
К6	Касир	Додати товар в систему	При наявності доступу, користувач може вносити новий товар в систему при необхідності.
А1	Адміністратор	Адміністрування користувачів	Адміністратор може додати, видалити, редагувати дані про користувачів.
А2	Адміністратор	Робота з звітністю	Доступ до даних по продажі товарів.

Аналіз представлених варіантів використання встановив наступні взаємозв’язки.

Варіанти використання для касира «Додати товар в чек за допомогою штрих-коду», «Додати товар в чек за допомогою коду» та «Додати товар в чек по назві» можна узагальнити до «Додати товар в чек». Це пов’язано з тим, що по великому рахунку виконується одна і та сама дія, користувач «Касир» шукає в системі товар, для того, щоб потім додати його в чек. На етапі а проектування не є критичним, чи це буде зроблено за допомогою сканера, чи за допомогою вводу коду товару в системі [12]. Результат досягається один і той самий товар додається в чек, узагальнення представлене на рисунку 2.3.1.

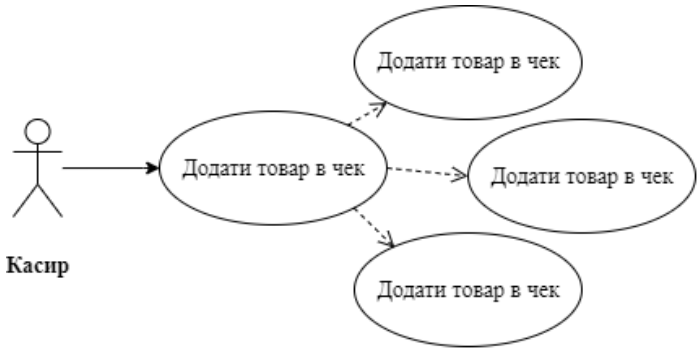

Рисунок 2.3.1 – Узагальнення варіантів «Додати товар в чек»

Аналогічна ситуація виникає і для варіанту використання «Застосувати дисконтну карту». Оскільки варіанти використання «Застосувати карту за кодом» та «Застосувати карту по штрих-коду» в плані логіки є однаковими, вони лиши дозволяють отримати дані про дисконтну карту користувача. Це означає, що ці варіанти можна узагальнити до одного варіанту використання «Застосувати дисконтну карту». Це дозволить наглядно спростити представлення про систему, рисунок 2.3.2.

Рисунок 2.3.2 – Узагальнення пов’язане з дисконтною картою

Варто також звернути увагу на узагальнення «Адміністрування користувачів». Сюди входять усі варіанти пов’язані з керуванням

користувачами системи. До них можна віднести «Добавити користувача», «Видалити користувача», «Редагувати дані», «Зміна прав доступу».

Адміністратор це ключовий користувач який відповідає за цілісність та коректність роботи системи. Одним з основних його завдань є менеджмент нових користувачів. Також варто додати, що саме адміністратор керує правами доступу. Мається на увазі, що касири поділені на дві групи. Перша з яких має доступ лише до продажу товарів і може лише додавати товари в чек та продавати їх. Інша категорія має доступ на внесення нових товарів в систему. За допомогою сканера штрих-кодів та простого вводу коду товару, ці користувачі можуть додавати нові товари в систему [16]. Узагальнення зображене на рисунку 2.3.3. Адміністратор може як надавати ці права так і забирати їх, ніякої попередньої підготовки для цього не потрібно. Варто також зазначити, що передбачено обмеження у розмірі 30 касирів для даної системи. Оскільки наявність більшої кількості користувачів може значною мірою сповільнити систему, що призведе до некоректного функціонування. Окрім цього адміністратор має можливості, які не зазначені у варіантах використання. Наприклад це моніторинг за часом, який проводять за робочим місцем інші користувачів системи. Адміністратор може бачити сумарну кількість годин, які проводили касири за робочим місцем .

Рисунок 2.3.3 – Узагальнення пов’язане з варіантом використання «Адміністрування користувачів»

Наступне узагальнення це «Робота з звітністю». Даний варіант використання включає в себе такі можливості як «Звітність за відрізок часу», «Вивести звітність на екран», «Зберегти у форматі PDF». Даний варіант використання описує логіку пов’язану з переглядом даних про продажі магазину. Доступ до цих даних має лише адміністратор, рисунок 2.3.4.

Рисунок 2.3.4 – Узагальнення пов’язане з варіантом використання «Робота з звітністю»

Аналіз даних варіантів використання показав, що з погляду потенційних ризиків і архітектурної значущості найбільш істотними є прецеденти, пов'язані з певними варіантами використання акторів «Касир» та «Адміністратор». Певні прецеденти є ключовими в системі, адже саме на них зав’язана ключова логіка. Тому варто детальніше розглянути їх, саме ці користувачі користуються системою і певні варіанти використання мають найбільше значення. Варто крім цього звернути увагу, що для можливості доступу до певного варіанта використання потрібно зробити дії, щоб отримати доступ до усього функціоналу. В залежності від прав доступу, «Касир» може лише отримувати можливість для додавання товару в чек та застосування дисконтних карток, а також може мати права, щоб додавати новий товар в систему, варто розмежовувати ці два моменти, адже від цих прав, задачі даної групи користувачів можуть відрізнятися.

Для подальшої деталізації вибрано такі прецеденти:

- К1. Авторизація в системі;
- К2. Застосувати дисконтну карту;
- К5. Додати товар в чек;
- А2. Робота з звітністю.

В таблиці 2.3.2 описаний варіант використання «Авторизація в системі». При у спішному завершенні прецеденту користувач отримує змогу надалі працювати з системою, якщо за певних причин логін або пароль були введені неправильно, у доступі до системи буде відмовлено. Для того, щоб відновити пароль або доступ до системи, користувач повинен зв’язатися з адміністратором та повідомити про неможливість входу в систему. Дану проблему адміністратор вирішує на свій розсуд. Можливій варіант, що касирові повідомлять старий пароль з допомогою якого відновіться можливість авторизації [6]. Інший варіант, це заміна паролю на новий. Примітка, в самій системі відсутні інструменти, механізми для користувач заміни користувачем даних для авторизації, тобто логіна або пароля.

Таблиця 2.3.2. – Опис варіанту використання «Авторизація в системі»

Прецедент: Авторизація в системі
ID: К1
Короткий опис: Касир має змогу авторизуватися в системі для початку роботи з нею, а також завершити сеанс.
Головні актори: Касир
Другорядні актори: Немає
Передумови: Користувач зареєстрований в системі
Основний потік: 1. Касир вмикає програму АСУ мінімаркетом 2. Після увімкнення появляється поле для вводу логіна та пароля, відбувається процес авторизації. 3. Користувач вводить логін та пароль у відповідні поля. 4. Система авторизує користувача. 5. Система відмічає в базі даних час початку роботи користувача. 6. Запускається головний інтерфейс для роботи з програмою.
Постумови: При успішному закінченні прецеденту робляться відмітки в базі даних про час початку роботи користувача
Альтернативні потоки: 1.1 Невірно вказаний логін або пароль Якщо при виконанні п. 3 було введено невірно логін або пароль, система видає повідомлення про це в діалоговому вікні.

Наступний важливий прецедент, це «Застосувати дисконтну карту», таблиця 2.3.3. При наявності дисконтної карти клієнт може отримати знижку. Для цього необхідно надати касиру цю карту, після чого він просканує її і система автоматично вирахує знижку. Якщо карту не вдається просканувати, завжди можна ввести її код, який знаходиться на самій карті. Картки додає в систему касир з відповідними правами доступу.

Таблиця. 2.3.3 – Опис варіанту «Застосувати дисконтну карту»

Прецедент: Застосувати дисконтну карту
ID: К2
Короткий опис: Підчас продажу товарів, касир маю змогу застосувати дисконтну карту клієнта, при її наявності.
Головні актори: Касир
Другорядні актори: Немає
Передумови: Користувач зареєстрований та авторизований в системи, покупець купляє принаймні один або більше товарів і вони внесені в чек.
Основний потік: 1. Касир вмикає програму АСУ мінімаркетом 2. Касир авторизується в системі. 3. Додається товар в чек за допомогою сканера штрих-кодів. 4. Натискається на кнопку, що дозволяє оформити знижку по дисконтній картці. 5. Відкривається діалогове вікно. 6. За допомогою сканера штрих-кодів касир сканує карту. 7. Вираховується нова загальна сума чеку з врахуванням знижки.
Постумови: При успішному закінченні прецеденту вираховується нова ціна в чеку з врахуванням знижки.
Альтернативні потоки: 1.1 Невірно вказаний логін або пароль Якщо при виконанні п. 3 було введено невірно логін або пароль, система видає повідомлення про це в діалоговому вікні. 1.2 Не знайдено карти Якщо при виконанні п. 6 не вдалося розпізнати карту, касир може ввести вручну номер карти. 1.3 Жодного товару не додано в чек Якщо при виконанні п. 4 в списку товарів, в чеку, не було жодного товару, система видасть повідомлення, що необхідно додати товар.

«Додати товар в чек» - один з ключових варіантів використання, описаний в таблиці 2.3.4. Даний прецедент відіграє ключову роль в цілій системі, адже він відповідає за додавання товару в чек. У разі якщо не вдається просканувати штрих-код або він пошкоджений, користувач може ввести код товару та додати його в чек. Також в системі можуть бути товари без штрих-коду. В такому випадку додати їх в чек можливо лише за наявності коду товару. Якщо один товар було просковано декілька раз, позначка в графі «Кількість» буде рівна кількості сканувань.

Таблиця 2.3.4 – Опис варіанту « Додати товар в чек »

Прецедент: Додати товар в чек
ID: К5
Короткий опис: Касир сканує товар та додає його в чек
Головні актори: Касир
Другорядні актори: Немає
Передумови: Користувач зареєстрований та авторизований в системи, покупець купляє принаймні один або більше товарів.
Основний потік: 1. Касир вмикає програму АСУ мінімаркетом 2. Касир авторизується в системі. 3. Додається товар в чек за допомогою сканера штрих-кодів. 4. Товар появляється в таблиці, яка відображає список усього чеку. 5. При необхідності касир змінює кількість конкретного товару в таблиці. 6. Вираховується загальна сума чеку.
Постумови: При успішному закінченні прецеденту товар додається в чек, а також вираховується загальна сума чеку
Альтернативні потоки: 1.1 Невірно вказаний логін або пароль Якщо при виконанні п. 3 було введено невірно логін або пароль, система видає повідомлення про це в діалоговому вікні. 1.2 Не знайдено товару Якщо при виконанні п. 3 не вдалося розпізнати товар, касир може ввести вручну номер товару. 1.3 Жодного товару не додано в чек Якщо не було додано жодного товару в чек, система видає повідомлення про необхідність додати принаймні одного товару для закриття чеку.

Також адміністратор має можливість працювати з інформацією по продажах, описано в таблиці 2.3.5. За допомогою цього може отримати доступ до інформації пов’язаною з продажами за обраний період часу. Можна перегляну який саме товар, та в якій кількості було продано. При необхідності цю інформацію можна зберегти на власний комп’ютер. За допомогою цієї інформації можна провести детальний аналіз діяльності магазину та усіх продажів і при необхідності провести оптимізаційні кроки для покращення роботи підприємства [14].

Таблиця 2.3.5 – Опис варіанту використання « Робота з звітністю »

Прецедент: Робота з звітністю
ID: A2
Короткий опис: Доступ до даних по продажі товарів. Присутня можливість роздрукувати або вивести на екран усі дані.
Головні актори: Адміністратор
Другорядні актори: Немає
Передумови: Користувач зареєстрований та авторизований в системи, було здійснено принаймні однин продаж
Основний потік: 1. Адміністратор вмикає програму АСУ мінімаркетом 2. Авторизується в системі. 3. Вибирає вкладку «Звітність». 4. Встановлю проміжок за який потрібно отримати дані по звітності. 5. Система виводить інформацію по звітності на екран. 6. При необхідності, адміністратор зберігає її на комп’ютер.
Постумови: При успішному закінченні прецеденту адміністратор отримує на своєю ПК документ з звітністю за обраний період
Альтернативні потоки: 1.1 Невірно вказаний логін або пароль Якщо при виконанні п. 3 було введено невірно логін або пароль, система видає повідомлення про це в діалоговому вікні. 1.2 Не знайдено звітності Якщо при виконанні п. 4 системі не вдалося нічого знайти або виникла помилка, користувач отримує повідомлення про це.

Рисунок 2.3.5 – ключові варіанти використання

На рисунку 2.3.5 описано ключові варіанти використання. Частина з них вже була описана вище, розглянемо решту. «Адміністрування користувачів» - один з прецедентів який забезпечує функціонал пов’язаний з додаванням інших користувачів в систему. Дозволяє адміністратору додати в систему нового користувача. А також при необхідності видалити вже існуючого. Також ключовою функцією є надання прав доступу вже наявним користувачам. Завдяки цьому можна керувати правами доступу вже наявних користувачів та надавати або забирати в них частину функціональних можливостей. «Пошук товару в систему» - забезпечує касиру можливість знайти наявний товар в системі. Це можна зробити за допомогою штрих-коду або просто ввівши частину назви цього товару в стрічку для пошуку. Це дозволяє перевірити його наявність та ціну. Також важливу роль відіграє «Додати товар в систему». Цією функцією можуть скористатися лише користувачі з відповідними правами доступу, які надає адміністратор. За допомогою цієї функції касир може додавати новий товар в систему. Для цього необхідно натисну «Додати товар». Після чого появиться відповідна форма для заповнення. У цій формі необхідно вказати назву товару, код для ідентифікації, а також відсканувати штрих-код [6].

2.4 Проектування діаграми послідовності

Визначивши усі варіанти використання, можна описати саму роботу системи більш детально. Щоб це описати використаємо один з різновидів діаграми UML, а саме діаграму послідовності. Ця діаграма дає змогу краще зрозуміти взаємодію об’єктів, які впорядковані за часовим критерієм. Ця діаграма найкраще описує, які об’єкти були задіяні та в якій послідовності відбувався обмін повідомленнями між ними. В основному опис відбувається за допомогою горизонтальних та вертикальних ліній. Де вертикальні – це процеси чи об’єкти, а в свою чергу горизонтальні – це повідомлення які були надіслані. Ці повідомлення, як правило впорядковані за відправленням [6].

Переважно на цій діаграмі показують лише ті об’єкти, що беруть участь у обміні інформацією, можливі або статичні об’єкти не враховуються. Для діаграми даного типу є важливою послідовність об’єктів та динаміка з якою вони взаємодіють [7].

На рисунку 2.4.1 представлена діаграма послідовності для варіанту використання «Додати товар в чек». Спочатку касир дає запит на пошук товару, за допомогою сканування штрих-коду, далі система шукає в базі даних відповідний товар.

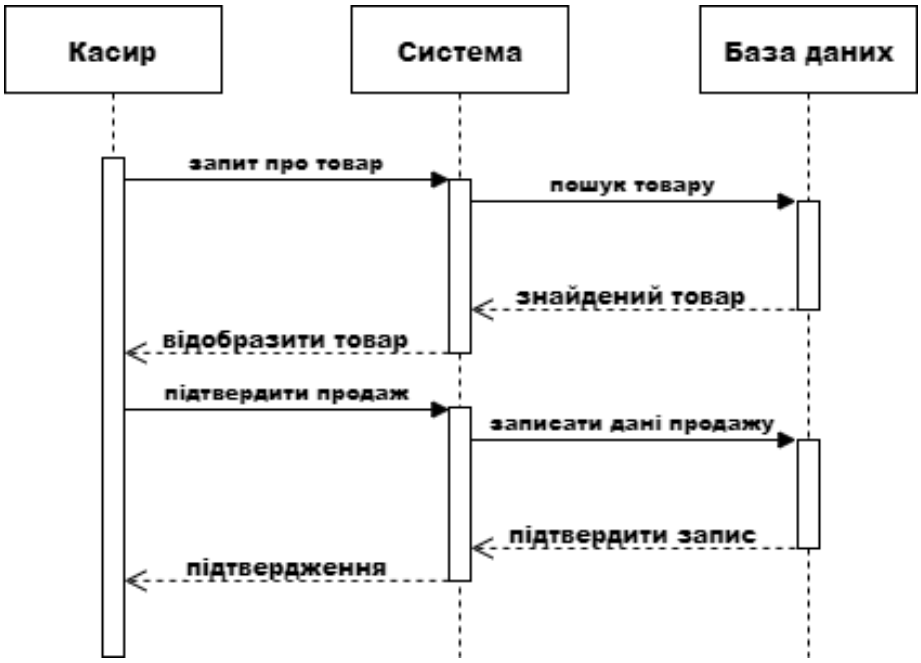

Рисунок 2.4.1 – діаграма послідовності « Додати товар в чек»

Також варто розглянути варіант використання «Робота з звітністю». Роботу починає адміністратор, вибираючи вкладку «Інформація про звітність». Після чого відправляється відповідний запит в систему. Коли система отримала та опрацювала запит, він передається далі у базу даних. В залежності від обраних критеріїв виконується пошук даних. Знайдені дані назад повертаються в систему, яка опрацьовує їх відповідним чином та відображає користувачеві у табличному вигляді. Після отримання інформації користувач має вибір, завершити перегляд інформації про звітність або зберегти її на копм’ютер.

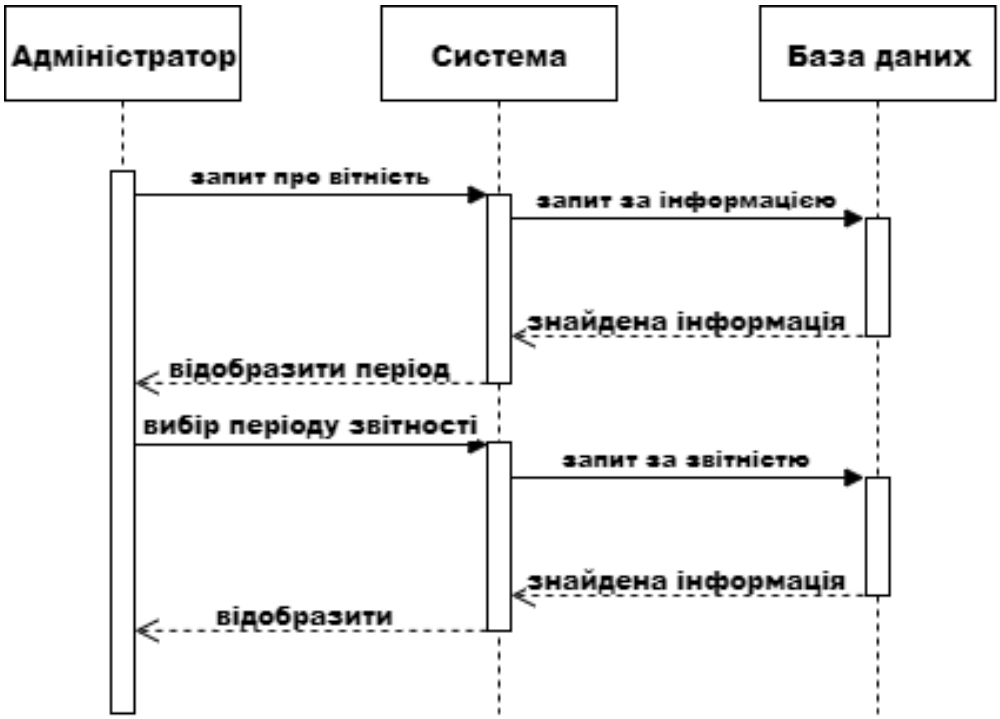

Рисунок 2.4.2 – діаграма послідовності «Інформація про звітність»

Дана діаграма дозволяє поглиби розуміння роботи окремих елементів програми та спростити конструювання та розробку ПЗ. Окрім головних сценаріїв використання, варто також пам’ятати про альтернативні потоки, які можуть значною мірою ускладнити проектування та реалізацію. При роботі з таким типом потоків завжди потрібно пам’ятати про необхідність якщо не всіх, то практично всіх альтернативних сценаріїв, адже навіть одне недоопрацювання в котромусь з напрямків може привести до критичної помилки та краху усієї системи. Тому також варто пам’ятати про обробку помилок в системі [6].

2.5 UML-діаграми активності

Для більш детального розуміння ВВ «Застосувати дисконтну карту» представимо його за допомогою діаграми активності, рисунок 2.5.1. Ця діаграма дозволяє покроково відтворити потрібний нам сценарій з наглядною демонстрацією усіх кроків та етапів виконання даної операцій.. Діаграма активності починається з стартової позиції, з якої отримуються початкові дані і продовжується робота з ними. Після чого події можуть розвиватися поступово або паралельно. Також можливі розгалуження тоді, коли можливий альтернативний сценарій.

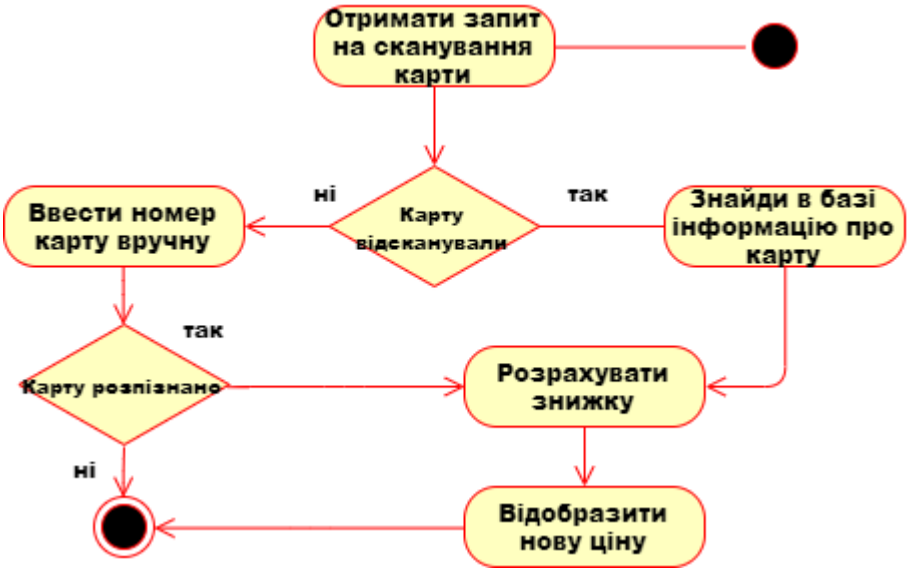

Рисунок 2.5.1 – діаграма активності «Застосувати дисконтну карту»

Дана діаграма починається з отримання запиту на сканування карти. Після чого касир сканує дисконтну картку клієнта. Якщо карту не вдається знайти касир може ввести дані карти вручну. Якщо ж знову карти не знаходить, в діалоговому вікні відображається інформація про те, що не вдалося знайти такої карти. Коли карту вдалося просканувати номер цієї карти передається в базу даних для пошуку точної інформації про суму знижки [17]. Після того як інформацію було знайдено база даних повертає її назад в систему, яка в свою чергу розраховує нову ціну вже з врахуванням знижки. Після чого на екрані касира відображається остаточна ціна з врахуванням знижки.

Діаграма активності визначається як діаграма UML, яка фокусується на виконанні та потоці поведінки системи замість реалізації. Її також називають об'єктно-орієнтованою блок-схемою. Діаграми діяльності складаються з заходів, що складаються з дій, які стосуються технології поведінкового моделювання.

Вона дозволяє створювати події як діяльність, що містить набір вузлів, з'єднаних ребрами. Діяльність може бути приєднана до будь-якого елемента моделювання, щоб змоделювати її поведінки. Діаграми активності використовуються для моделювання:

- Варіантів використання;
- Класів;
- Інтерфейсів;
- Компонентів;
- Взаємодії.

Діаграми активності використовуються для моделювання процесів та робочих потоків. Суть діаграми зосереджена на повідомленні конкретного аспекту динамічної поведінки системи. Вони фіксують свою увагу на динамічних елементах.

Вона схожа на блок-схему, яка візуалізує потік від однієї діяльності до іншої діяльності. Крім того, може бути ідентична блок-схемі, але вона не є блок-схемою. Потік активності можна керувати за допомогою різних елементів керування на діаграмі UML. Простими словами, діаграма використовується для потоків діяльності, які описують потік виконання між декількома видами діяльності [6].

Щоб скласти діаграму такого типу, потрібно зрозуміти та дослідити всю систему. Користувачі повинні знати всі елементи та об'єкти, які будуть використовуватися всередині діаграми. Користувачеві має бути зрозуміло центральне поняття, яке є не що інше, як принцип роботи системи. Проаналізувавши всі види діяльності, щоб знайти різні обмеження, що застосовуються до реалізації. Якщо існує таке обмеження, то це слід зазначити перед розробкою.

2.6 Проектування бази даних

Для зберігання великої кількості про товари та історію їх продажів було спроектовану базу даних, яка дозволить оперувати усією інформацією. Всі таблиці перебувають у певних відношеннях. Було застосовано два ключові відношення.

Перше це відношення «один до один» воно було використане, коли запис в одній таблиці має тільки один запис у іншій таблиці. Цей взаємозв'язок зустрічається досить рідко, розмістивши всі дані в одній таблиці можна досягти більшої продуктивності, проте він може застосовуватися для розділення логіки. Коли створюється відшення даного типу і поля, що використовуються також унікальні – відношення даного типу може бути створене [11].

Окрім цього також було використано відношення «один до багатьох», цей тип відношень зустрічається найчастіше. Дане відношення необхідне, коли, запис у одній таблиці є частиною запису одного або декількох у іншій таблиці, проте кожен окремо вибраний запис у іншій таблиці стосується одного і тільки одного запису в першій таблиці. Наприклад, відносини між таблицею "Клас" та таблицею "Учень" - адже кожен клас може мати декілька учнів, але кожне учень належить лише до одного класу.

Відношення «багато до багатьох» виникає, коли кожен запис у таблиці 1 стосується декількох записів у таблиці 2, а кожен запис у таблиці 2 стосується декількох записів у таблиці 1. Наприклад, зв'язок між таблицею «Товар» та таблицею «Постачальник» є багато до багатьох, тому що в одного постачальника може бути багато товарів для поставки, і кожен товар в свою чергу може мати багато постачальників замовленнях. Зазвичай таке відношення реалізується через третю таблицю [3].

В даній розробці воно теж було реалізоване. Воно використається для створення та зберігання історії продажів кожного продавця. Оскільки кожен продавець може продати безліч товарів і також кожен товар може продати багато продавців.

Для початку розглянемо сутності «activities» та «users» - рисунок 2.6.1. Ці сутності описують ключові дані пов’язані з збереженням даних користувачів та їх активності. Сутність users описую дані пов’язані з самим користувачем. Сюди входить унікальний ідентифікатор idusers, який виступає первинним ключем таблиці. А users_type це тип користувача, в даній системі користувач може бути двох типів, якщо 1, то це адміністратор, якщо 2 – касир. Інформація про ім’я самого користувача зберігається в графі users_name. В permission_status описується права доступу касира. Якщо 1 то касир має право лише пробивати чеки, якщо 2 – можливість додавати нові товари в систему.

Дані про дату реєстрації користувача зберігається в reg_data. В свою чергу сутність activities складається з ідентифікатор idactivities, який визначає унікальність даного запису. Дана сутність створена для збереження інформації про те, коли користувачі заходили в систему, тому для визначення який саме користувач заходив в систему було додано вторинний ключ, а саме id_users. Він посилається на сутність users, а точніше на поле idusers. Завдяки цьому ці дві таблиці можна зв’язати між собою. Поле start_time містить в собі інформацію пов’язану з датою та часом початку роботи користувача.

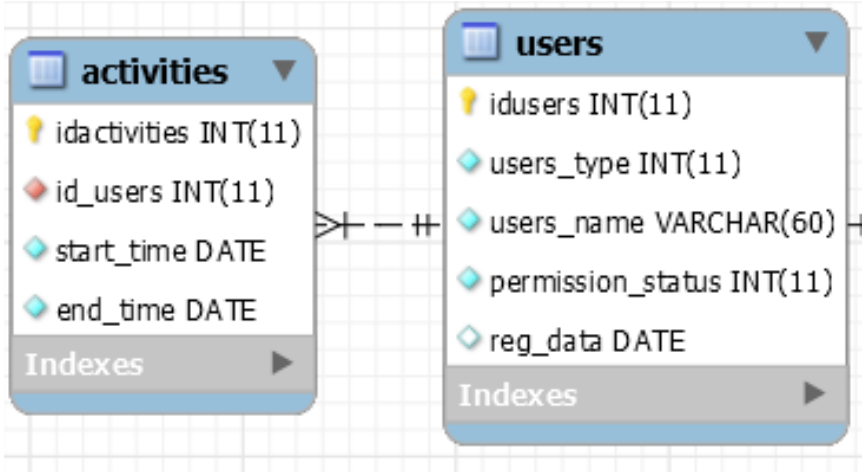

Рисунок 2.6.1 – таблиця сутностей «activities» та «users»

А в end_time міститься інформація про те, коли користувач завершив роботу з системою. Як можна помітити, ключова інформація про користувача зберігається в сутності users, в свою чергу інформація про активність цього користувача зберігається в сутності activities [3].

Відношення між ними встановлене як один до багатьох. Рахується, що один користувач може мати багато записів про активність, проте в одній активності інформація може зберігатися тільки про одного користувача [7].

Також необхідна сутність для збереження даних про товар в системі. Нею виступає таблиця під назвою items. В цій таблиці зберігається уся інформація пов'язана з товар мінімаркету. Первинним ключем в цій таблиці виступає iditems. Для збереження назви про сам товар передбачене поле під назвою items_name. В incomeDay зберігається інформація про дату коли товар було додано в систему. Графа price відповідає за збереження інформації пов'язаної з ціною товару. Також, кожен товар має свого поставника, в полі id_supplier зберігається зовнішній ключ на таблицю з інформацією про поставника товарів. Ця інформація зберігається в таблиці supplier. Вона складається з первинного ключа idsupplier та поля name, в якому власне і зберігається інформація про назву поставника. Між цими сутностями встановлений зв'язок один до багатьох. Це пов'язано з тим, що один поставник може надавати багато товарів, але в товару може бути лише один один поставник. Якщо виникає колізія, при який один товар можуть надавати двоє різних поставників, товар необхідно ще раз додати

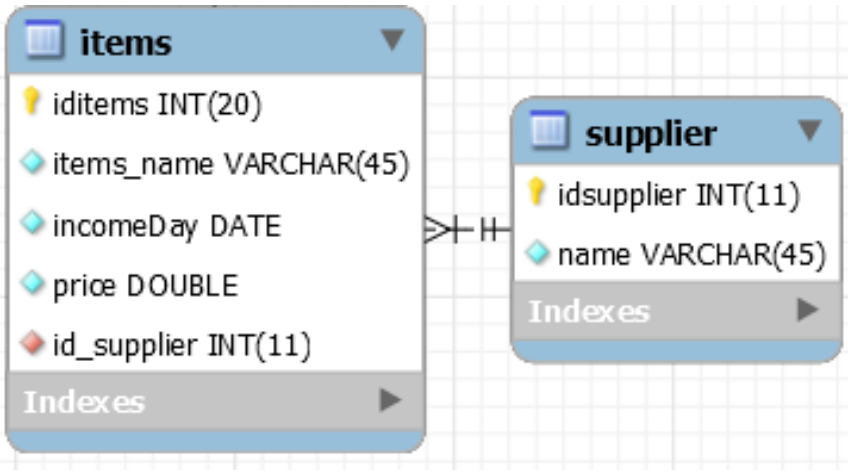

Рисунок 2.6.2 – таблиця сутностей «items» та «supplier»

Вже встановлено, що в системі потрібно зберігати інформація про користувачів та про товари.

Однак окрім цього, для ведення звітності необхідно забезпечити зберігання даних про історію продажів.

Саме це забезпечує сутність history, в якій зберігається інформація про продані товари. Первинним ключем виступає поле idhistory. В полі list зберігається весь список покупок. Для збереження інформації про вартість чек використовується поле amounth.

Для збереження даних про дату закриття чеку використовується поле date. Також, якщо покупку робить клієнт, який має карту для знижок, відсоток знижки записується у поле discount_percent. Для того, щоб мати статистику про те, який скільки чеків закрив касир, необхідно зберігати інформацію також і про касира, в полі id_seller зберігається зовнішній ключ з посиланням на продавця, який закрив цей чек.

Окрім цього в полі id_card_discount зберігається зовнішній ключ таблиці discount_card, в якій зберігається детальна інформація про карту знижки, якщо така була задіяна. В сутності discount_card роль первинного ключа виконує поле iddiscount_card, а поле short_id зберігає запасний та коротший код по якому можна знайти додатково знайти картку. За допомогою discount_percent можна дізнатися яку знижку надає дана картка, а поле amount зберігає в собі інформацію про загальну суму на яку було куплено в магазині за допомогою цієї картки на знижки.

Дані сутності знаходяться у відношенні оди до багатьох, адже для багатьох покупок може бути лише одна карта.

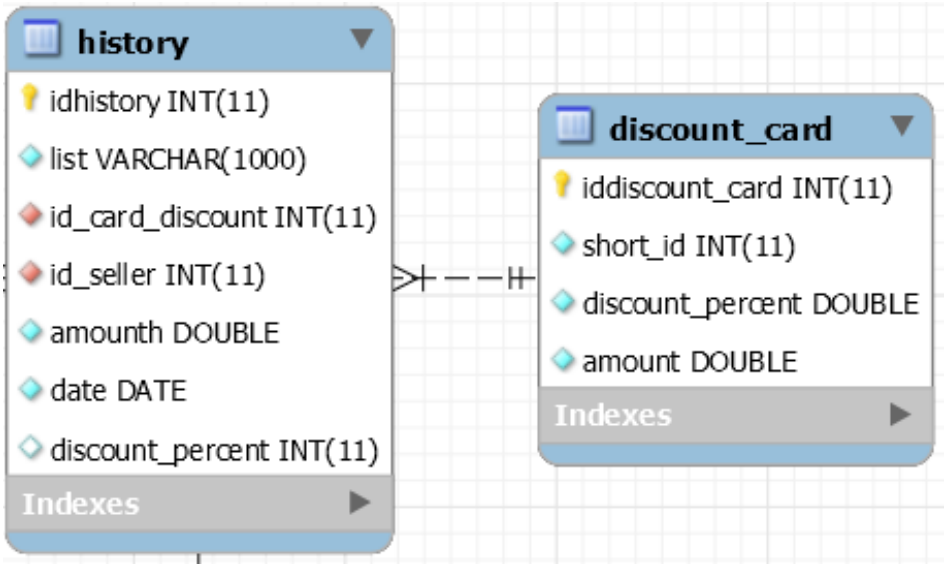

Рисунок 2.6.3 – таблиця сутностей «history» та «discount_card»

На рисунку 2.6.4 представлена загальна схема сутностей та їх зв’язки. Вище вже було описано загальні сутності та їх ролі. Залишилося ще розглянути відношення багато до багатьох у таких сутностей як history та items. Таке відношення було створене у зв’язку з необхідністю збереження історії покупок для ведення звітності у системі. Це відношення реалізується за допомогою додаткової сутності history_has_items. В ній присутні два поля, а саме history_idhistory та items_iditems. Це означає, що в одній історії може міститися інформація про багато товарів. І в свою чергу, багато товарів можуть входити в одну звітність.

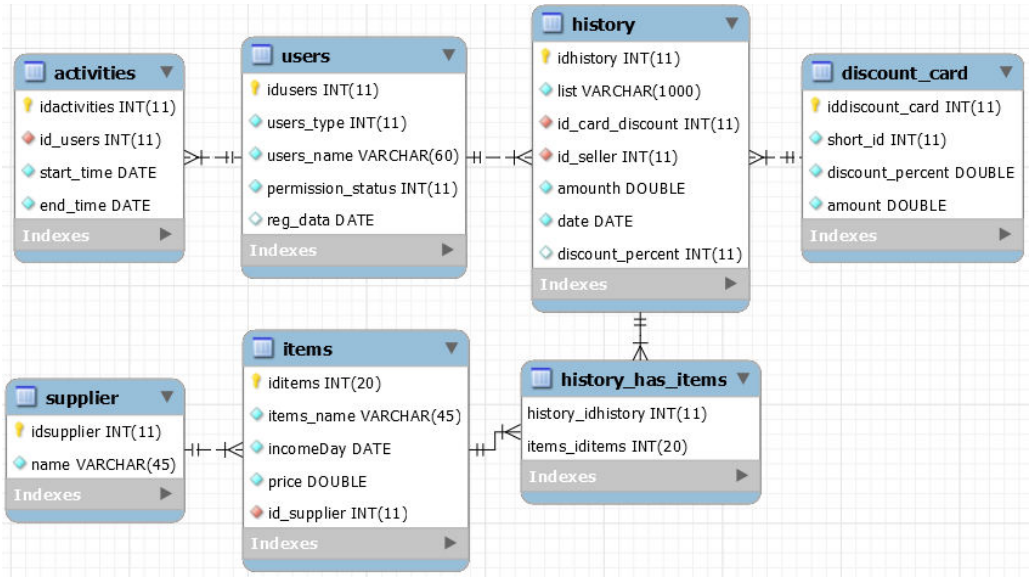

Рисунок 2.6.4 – загальна схема сутностей та їх зв’язків

Для пришвидшення виконання певних елементів програми, було прийнято рішення помістити частину логіки на сторону сервера. На сервері було розміщено вбудовані процедури, тригери та представлення.

Вбудовані процедури мають мету пришвидшити та полегшити виконання певних алгоритмів пов’язаних з маніпуляціями над даними з бази даних, тому вони виконують операції, які користувач програми, в теорії, буде використовувати досить часто. Тригери мають мету забезпечити цілісність даних, які записуються у базу даних та, при необхідності, певним чином редагують записи при додаванні їх у базу даних. Представлення мають на меті полегшити вибірку даних які досить часто вибираються, та скоротити досить громісткий синтаксис при об’єднанні таблиць [11].

3. РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Розробка та тестування головних функцій системи

Для тестування буде використовуватись скомпільований .exe файл. Тестування програми проводилося на операційній системі Windows 7. Програма тестувалася на коректне відображення компонентів та правильну роботу ключових елементів.

Після запуску програми відкривається початкове діалогове вікно для авторизації у систему (рис. 3.1.1).

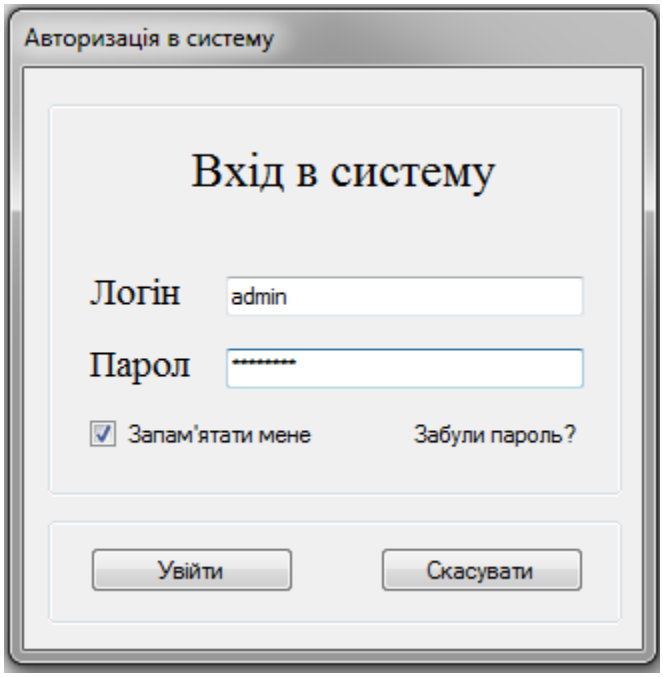

Рисунок 3.1.1 – авторизація в систему

Дана форма складається з двох ключових полів, одне для вводу логіна, інше для пароля. Внизу форми знаходиться дві кнопки: «Увійти» та «Скасувати». Після того як дані в поля були введені, користувач може натиснути «Увійти», якщо дані ведено коректно, система пропустить далі і користувач почне працювати з іншим вікном. Якщо дані були введені некоректні, появиться діалогове вікно, яке проінформує про це користувача. Також присутня функція, яка дозволяє запам’ятати для того, щоб не доводилося повторно вводити логін та пароль. Ця функція діє до перезавантаження комп’ютера.

При натиску на кнопку «Скасувати», закривається діалогове вікно та система завершує свою роботу. При наведені курсором на напис «Забули пароль» появляється підказка, яка допоможе відновити доступ до системи.

Саме за цим вікном користувач проводить найбільше часу. Умовно його можна розділити на три частини: верхню, центральну та нижню. У верхній частині знаходиться панель керування під назвою «menuStrip». Це один з компонентів фреймворку Windows Form. В ній розміщені дві кнопки «Каса» та «Довідка». За допомогою першої кнопки користувач має змогу додати новий товар в систему, за умови відповідних прав доступу, за допомогою другою – отримати довідкову інформацію. Під цим компонентом знаходиться інформація про касира який зараз працює, та про номер чека з яким даний касир зараз працює. В центральній частині програми знаходиться таблиця, яка відображає інформацію про товар, який відсканував касир. Даний компонент має назву «listView». ListView - це прекрасний спосіб відображення інформації та даних файлової системи з XML-файлу чи бази даних. Елемент керування ListView зазвичай використовується для відображення графічного значка, що представляє елемент, а також тексту елемента [2]. Крім того, ListView може використовуватися для відображення додаткової інформації про елемент у підпункті. Наприклад, якщо в елементі управління ListView відображається список файлів, присутня можливість налаштувати елемент управління ListView таким чином, щоб відображати деталі, такі як розмір файлу та атрибути як підпункти. Щоб відобразити інформацію про підрозділ в елементі управління ListView, потрібно встановити властивість View на View.Details. Крім того, повинні бути створені об'єкти ColumnHeader і призначені їх властивості управління. Після встановлення цих властивостей елементи відображаються у форматі рядків та стовпців, подібних до елемента керування DataGrid. Можливість відображення елементів таким чином робить ListView швидким та простим рішенням для відображення даних із будь-якого типу джерел даних.

При необхідності дані можна відсортувати. Сортування для елемента ListView здійснюється за допомогою метода сортування.

Це дає змогу визначити тип сортування, який слід застосувати до елементів. Це відмінна функція, якщо сортувати лише за елементами. Якщо сортувати за підпунктами, повинні використовувати спеціальні функції сортування елемента [8].

З його допомогою можна створювати таблиці необхідних форм та розмірів. Таблиця складається з 6 стовпців, які відображають ключову інформацію про куплений товар. Після успішною авторизації в систему, касиру відкривається наступне вікно для роботи з системою – рисунок 3.1.2.

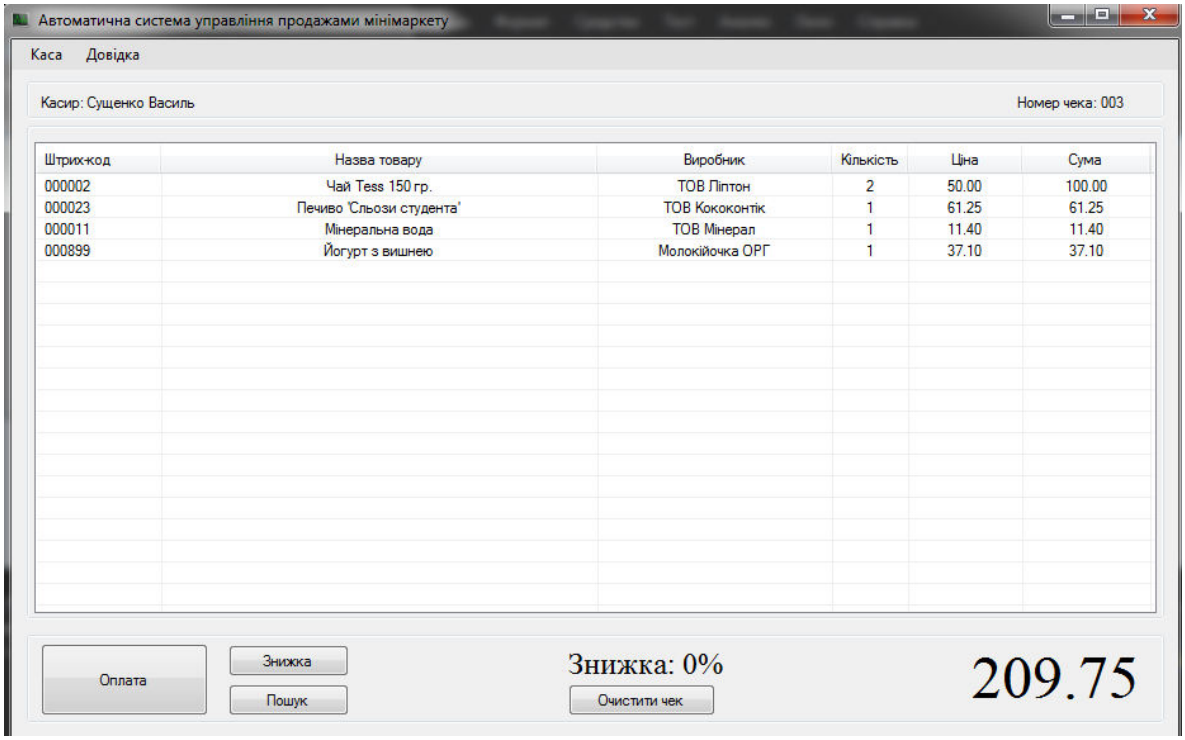

Рисунок 3.1.2 – головне вікно програми для касира

Штрих-код відображає інформацію пов’язану з відсканованим кодом. «Назва товару» - поле в якому міститься інформацію про назву купленого товару. В «Виробник» - інформація про поставника або виробника даної продукції. Наступні три колонки містять інформацію про кількість та вартість купленої продукції. Також встановлене обмеження на кількість даних в таблиці в розмірі 1000. В середній чек входить близько 25 товарів, для того, щоб уникнути непередбачених помилок було встановлене це обмеження. Для завантаження даних в таблицю необхідно встановити зв’язок з базою даних, в якій зберігається вся інформація [9].

У лістингу 3.1.1 продемонстровано метод openConnect, який відповідає за підключення програми до бази даних. Також цей метод використовується при верифікації користувачів.

```
public static void openConnect()
        {
            myConnectionString =
"server=localhost;uid=root;pwd=1234;database=store;"; //підключення до
сервера
            try
            {
                connect = new MySql.Data.MySqlClient.MySqlConnection();
                connect.ConnectionString = myConnectionString;
                connect.Open();
            }
            catch (MySql.Data.MySqlClient.MySqlException ex)
            {
                MessageBox.Show(ex.Message);
            }
```

Лістинг 3.1.1 – реалізація методу openConnect

Після того як встановлено з'єднання з базою даних, кожен раз коли користувач сканує товар, система отримує код – унікальний ідентифікатор товару. Після чого робиться запит в базу даних і цьому коду вся інформація про товар. Після чого вона поміщається в компонент таблиці listView. Якщо декілька одиниць одного і того самого товару необхідно просканувати, касир має два варіанта. Перший – провести кожен товар через сканер, тоді число в колонці «Кількість» збільшується на 1 після кожного сканування.

Або інший варіант – вручну змінити значення в полі «Кількість». Система автоматично вирахує суму цього товару.

Ця дія виконується кожен раз після сканування штрих-коду. Загальна сума вираховується на основі даних заповнених в таблиці.

В нижній частині вікна знаходяться чотири кнопки управління. «Очистити чек» - видаляє всі дані з чеку, якщо виникає така необхідність. «Знижка» дозволяє відкрити діалогове вікно, в якому є можливість ввести код для пошуку карти на знижки або просканувати картку. Якщо це вдалося, тоді вираховується нова сума з врахуванням знижки і відображається знизу в правому куті. В відповідному полі для знижки вказується її відсоток [10].

Заповнення таблиці даними з бази даних описано в лістингу 3.1.2

```
MySqlCommand command = new MySqlCommand(query, connect);  //виконання запиту
            using (MySqlDataReader reader = command.ExecuteReader()){
                while (reader.Read()){
                    string[] row = { reader.GetInt32(0).ToString(),
reader.GetString(1), reader.GetString(2),
                     reader.GetString(3), reader.GetString(4),
reader.GetString(5),reader.GetString(6)};
                    var listViewItem = new ListViewItem(row);
                    bookList.Items.Add(listViewItem);
                }
            }
```

Лістинг 3.1.2 – заповнення таблиці даними

Також присутня кнопка «Оплатити». Якщо в чеку немає жодного товару, вона є неактивною. Якщо додається принаймні один товар – активується. При натиску цієї кнопки відкривається діалогове вікно з формою для оплати. Сума яка передається в це вікно вже включає в себе знижку. Користувач може вписати суму, яку йому дав покупець і система автоматично вирахує здачу після натиску на кнопку «Розрахувати». Після цього стає доступно кнопка «Оплатити». Якщо користувач її натискає, то вважається, що операція завершена і вносяться зміни в БД. За замовчуванням вважається, що покупець розраховується готівкою, якщо ж це не так, касиру просто потрібно поставити галочку на «Безготівковий». Сценарієм використання передбачається, що користувач після цього надасть термінал для безготівкового розрахунку, а квитанції про операцію збереже в касі [7]. На рисунку 3.1.3 представлена дана форма.

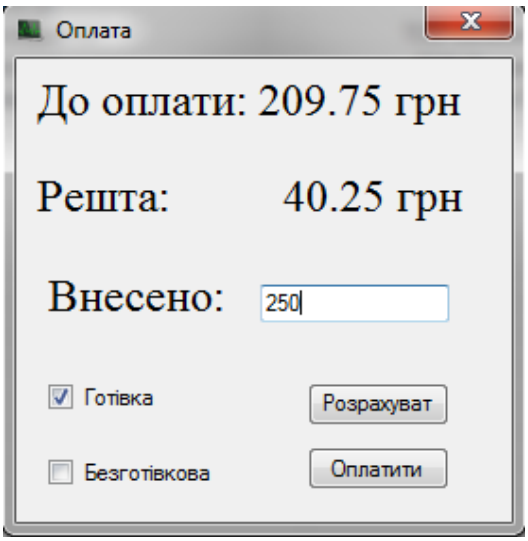

Рисунок 3.1.3 – вікно оплати

Користувач відповідними правами доступу має можливість додавати новий товар в систему. Для цього, на головній форма, на верхній панелі, потрібно натиснути «Каса», а далі з списку вибрати «Додати товар». Якщо у користувача немає прав доступу, ця функція буде недоступною, якщо є відкриється діалогове вікно для введення інформації про новий товар – рисунок 3.1.4.

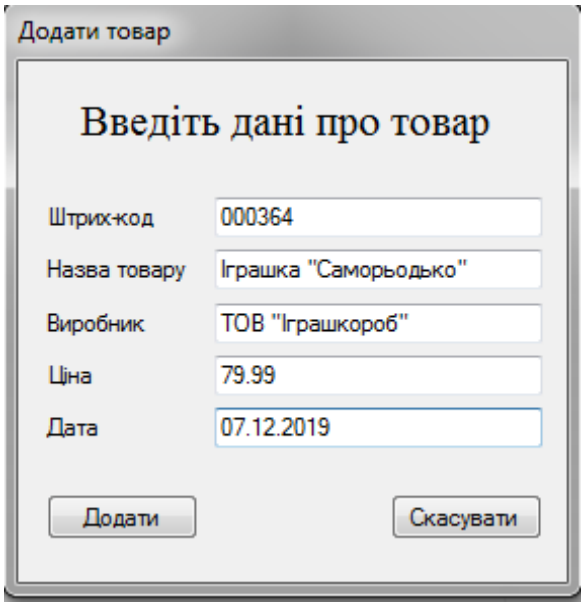

Рисунок 3.1.4 – форма додавання товару

Як можна побачити, форма містить три поля для додавання товару, це:

- Штрих-код;
- Назва товару;
- Виробник;
- Ціна;
- Дата.

Це ключові дані які описують товар. Користувач маже ввести дані про штрих-код в ручну, або просто просканувати штрих-код потрібного товару, при відкритій формі додавання товару, це код автоматично внесеться в систему. У разі, якщо такий штрих-код уже існує, система видає діалогове вікно з попередженням. Окрім цього, в полі «Ціна» можуть бути введені лише цифрові значення, система не дозволить ввести інші символи. Це саме стосується і графи «Дата» [1]. Сюди можуть бути записані дані лише такого типу, як на рисунку.

3.2 Тестування системи та інших функцій

Також варто проаналізувати скільки пам’яті займає дана програма. Використовуючи стандартні засоби для діагностики Visual Studio на операційній системі Windows 7 було встановлено, що запущена програма використовує орієнтовно 23мб (рисунок пам’яті в режимі очікування, а під навантаженням близько 100мб в залежності від кількості відкритих вікон та виконуваної роботи. Даний результат повністю задовольняє всі поставлені до системи вимоги. Адже використання пам’яті є досить низьким, що дозволяє програму запускати на ПК з не надто потужним апаратним забезпеченням [5].

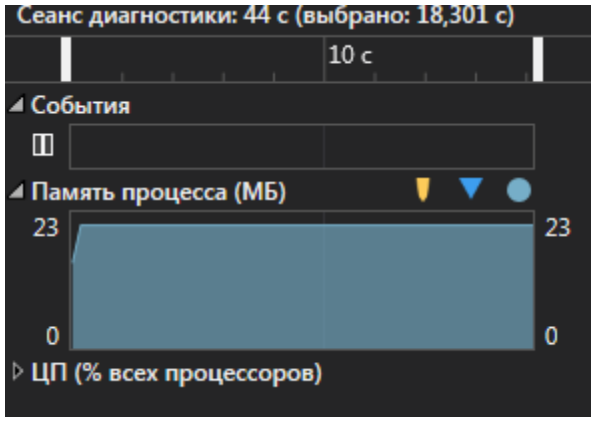

Рисунок 3.2.1 – тестування системи засобами Visual Studio

Також була виконана перевірка за допомогою перегляду навантаження системи в диспетчері завдань – рисунок 3.2.2. Як видно, при відкритому головному вікні програми та встановленому з’єднанні з базою даних програма використовує не більше 15мб. Проте варто враховувати, що дані з БД можуть використовуватися безпосередньо на локальному сервері, який встановлений разом з додатком АСУ на ПК. Враховуючи, що в базі даних буде не більше 100 000 записів про товари, 1гб оперативної пам’яті, та пам’яті на жорсткому диску повинно вистарчити для комфортної роботи з даною системою. При розрахунку мінімальних системних вимог, також потрібно врахувати затрати, на встановлення стороннього програмного забезпечення для роботи з сканером штрих-кодів [16].

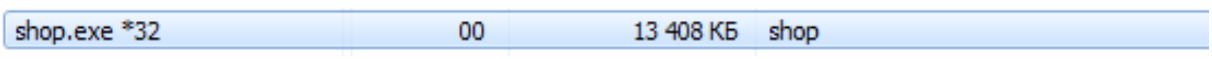

Рисунок 3.2.2 –дані з диспетчера завдань

4 ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

4.1 Розрахунок норм часу на виконання науково-дослідної роботи

Головною метою розділу виступає встановлення економічної доцільності додатку АСУ мінімаркету та розрахунок витрат.

На ринку програмного забезпечення представлена велика кількість продуктів, проте з часом міняються вимоги до існуючих систем. Для впровадження нових функцій, які задовільнять ці вимоги, можна дописувати уже існуюче програмне забезпечення або розробити нове.

Розробка нового програмного продукту вимагає свого управління і контролю з боку керівника. Таким чином, складання та організація економічної чатсини є актуальною проблемою сучасного менеджменту.

Планування потребує будь-яке підприємство, будь-яке виробництво, економіка в цілому. Спланувати – означає оцінити можливості, необхідність і обсяги випуску конкурентоспроможної продукції, визначити місткість ринку і його конкретного сегмента, оцінити попит на продукцію, що випускається підприємством, результативність його роботи на ринку.

Швидкий розвиток технологій, ускладнення і різноманіття пропонованої продукції та послуг, скорочення їх життєвого циклу, поява великої кількості конкурентоспроможних компаній, підвищення вимог з боку споживачів, збільшення обсягів і швидкості отримання інформації, нових знань – всі ці і інші зміни в світі змушують господарюючі суб'єкти шукати методи для кращої адаптації до нових умов.

Головною метою розділу є встановлення економічної доцільності розробки програмного забезпечення, розрахувати передбачені витрати та оцінити ризики непередбачуваних витрат.

Ефективне використання часу має велике значення тому, що коефіцієнт корисної дії залежить від оптимального використання часу.

Розробку поділяють на декілька етапів, що дозволить полегшити і структуризацію виконання розробки.

Для виконання проекту може буде залучено 3 керівника, 8 розробників та 8 тестувальник. На проекти такою складності виділяється 152 робочих годин, тому в подальшому будемо орієнтуватися саме на таку часову одинцю.

Витрати часу по окремих етапах розробки програмного забезпечення відображено в таблиці 4.1.

Таблиця 4.1 – Операції процесу розробки ПЗ і часові затрати

	Місячна зарплата грн.	Денна зарплата грн	Трудомісткість, людино-дні		Основна заробітня плата, грн	
			Процедур-ний підхід	ООП підхід	Процедур-ний підхід	ООП підхід
Керівник	10000	470	3	3	2280	2280
Розробник	10000	470	8	10	3760	4700
Тестуваль-ник	6000	285	8	9	2280	2565
Всього			19	22	8320	9545

Процедурний підхід передбачає використання більшої кількості ресурсів, це пов’язано з використанням застарілих принципів та методів розробки програмного забезпечення. В даній таблиці він наведений для наглядної демонстрації та як один з можливих варіантів.

В основному під час проекту та розробки програмного продукту головний нахил робився на об’єктно – орієнтований метод розробки, через його простоту ти потребу в меншій кількості ресурсів. З використанням

цього підходу можна оптимізувати вартість розробки програмного продукту на більше як 10%.

4.2 Визначення ключових витрат

Першою ключовою витратою є заробітна плата.

Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та господарської діяльності підприємства. Заробітна плата складається з основної та додаткової оплати праці.

Основна заробітна плата нараховується на виконану роботу за тарифними ставками, відрядними розцінками чи посадовими окладами і не залежить від результатів господарської діяльності підприємства.

$$ЗП_{осн1} = 8320 грн;\ ЗП_{осн2} = 9545\ грн$$

Додаткова заробітна плата – це складова заробітної плати працівників, до якої включають витрати на оплату праці, не пов'язані з виплатами за фактично відпрацьований час. Нараховують додаткову заробітну плату залежно від досягнутих і запланованих показників, умов виробництва, кваліфікації виконавців [19]. Джерелом додаткової оплати праці є фонд матеріального стимулювання, який створюється за рахунок прибутку.

$$ЗП_{дод} = 0{,}2 \cdot ЗП_{осн} \qquad (4.1)$$

$$ЗП_{дод1} = 0{,}2 \cdot ЗП_{осн1} = 1664\ грн.;$$

$$ЗП_{дод1} = 0{,}2 \cdot ЗП_{осн2} = 1909\ грн.$$

Таким чином загальний фонд заробітної плати, що обчислюється за формулою:

$$ФЗП = ЗП_{осн} + ЗП_{дод}. \qquad (4.2)$$

$$ФЗП_1 = 8320 + 1664 = 9984\ грн.;$$

$$ФЗП_2 = 9545 + 1909 = 11454\ грн.$$

Крім того, слід визначити відрахування на соціальні заходи:

- єдиний соціальний внесок – 3,6 %;
- військовий збір – 1,5 %;

– ПДФО (прибутковий податок) – 15 %.

Отже, сума відрахувань на соціальні заходи буде становити:

$$\text{Відр}_{\text{ЄСВ1}} = 0{,}036 \cdot \text{ФЗП} = 336{,}96 \text{ грн.};$$

$$\text{Відр}_{\text{ЄСВ2}} = 0{,}036 \cdot \text{ФЗП} = 412{,}40 \text{ грн.};$$

$$\text{Відр}_{\text{вз1}} = 0{,}015 \cdot \text{ФЗП} = 140{,}40 \text{ грн.};$$

$$\text{Відр}_{\text{вз2}} = 0{,}015 \cdot \text{ФЗП} = 171{,}81 \text{грн.}$$

$$\text{Відр}_{\text{ПДФО1}} = 0{,}15 \cdot \text{ФЗП} = 1400 \text{ грн.};$$

$$\text{Відр}_{\text{ПДФО2}} = 0{,}15 \cdot \text{ФЗП} = 1718{,}1 \text{грн..}$$

Нарахування на фонд оплати праці, які включають відрахування до Пенсійного фонду, фонду з тимчасової втрати працездатності, фонду з безробіття і фонду страхування від нещасних випадків на виробництві; для бюджетної організації тариф на фонд оплати праці встановлено на рівні 36,3%.

Зокрема, видання програмного забезпечення – 36,77%.

Нарахування на Фонд оплати праці (ФОП): $\text{ФОП}_{\text{ЄСВ}} = 0{,}3677 \cdot \text{ФЗП}$

$$\text{ФОП}_{\text{ЄСВ1}} = 0{,}3677 \cdot \text{ФЗП} = 3441{,}67 \text{ грн.};$$

$$\text{ФОП}_{\text{ЄСВ2}} = 0{,}3677 \cdot \text{ФЗП} = 4211{,}64 \text{ грн}$$

Вього витрат:

$$\text{В}_{\text{ЗП1}} = \text{ФЗП}_1 + \text{ФОП}_{\text{ЄСВ1}} = 8320 + 3441{,}67 = 11761{,}67 \text{грн.};$$

$$\text{В}_{\text{ЗП2}} = \text{ФЗП}_2 + \text{ФОП}_{\text{ЄСВ2}} = 9545 + 4211{,}64 = 13756{,}64 \text{ грн.};$$

Також важливою складовою витрат є матеріальні витрати. Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни:

$$M_{ві} = q_{\text{і}} \cdot p_{\text{і}}\,, \tag{4.3}$$

де: $q_{\text{і}}$ – кількість витраченого матеріалу і–го виду;

$p_{\text{і}}$ – ціна матеріалу і–го виду.

Звідси, загальні матеріальні витрати можна визначити:

$$З_{м.в.} = \sum M_{вi} \, . \qquad (4.4)$$

Таблиця 4.2 – Зведені розрахунки матеріальних витрат.

Найменування матеріальних ресурсів	Один. Виміру	Фактично витрачено матеріалів	Ціна 1-ці., грн.	Загальна сума витрат, грн
Папір формату А4	листів	1000	0,3	300
Тонер для принтера	шт	1	110	110
Флеш-накопичувач	шт	2	110	220
Всього				630

Отже, загальна сума матеріальних витрат становить 630 гривень.

В багатьох випадках існує ряд додаткових витрат які пов’язані із реалізацією проекту, але в більшості випадків їхня сума не перевищує десяти відсотків від загальної собівартості реалізації проекту.

Також варто врахувати електроенергію. Затрати на електроенергію використану 1-цею обладнання визначаються за формулою:

$$З_{е} = W \cdot T \cdot S \, , \qquad (4.5)$$

де W – необхідна потужність, кВт;

T – кількість годин роботи обладнання;

S – вартість кіловат-години електроенергії.

Вартість кіловат-години електроенергії слід приймати згідно існуючих на даний час тарифів. Отже, 1 кВт з ПДВ коштує 2,50 грн.

Потужність комп’ютера для створення проекту – 750 Вт, кількість годин роботи необхідних для проекту – 150 години при процедурному підході та 170 годин при об'єктно орієнтованому підході.

$$З_{е1} = 0{,}4 \cdot 152 \cdot 2{,}50 = 152$$

$$З_{е2} = 0{,}4 \cdot 176 \cdot 2{,}50 = 176$$

Характерною особливістю застосування основних фондів у процесі виробництва є їх відновлення.

Для відновлення засобів праці у натуральному виразі необхідне їх відшкодування у вартісній формі, яке здійснюється шляхом амортизації.

Для визначення амортизаційних відрахувань застосовуємо формулу:

$$\text{А} = \frac{\text{С}_\text{Б} \cdot N_A \cdot T_{\text{ФАК}}}{\text{Т}_{\text{год}}} \quad (4.6)$$

де $\text{С}_\text{Б}$ – балансова вартість обладнання, грн;

N_A – норма амортизаційнихвідрахувань в рік, %;

$\text{Т}_{\text{год}}$ – річний робочий фонд часу, год;

$T_{\text{ФАК}}$ –фактичний час роботи обладнання по написанню програми, год.

Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Отже, використовуючи в роботі 1 комп'ютер балансовою вартістю 15000 грн. Отже, амортизаційні відрахування будуть рівні:

$$A_1 = (15000 \cdot 0{,}6 \cdot 152)/2080 = 657{,}7 \text{ грн.}$$

$$A_2 = (15000 \cdot 0{,}6 \cdot 176)/2080 = 761{,}54 \text{ грн.}$$

Варто врахувати і накладні витрати, адже вони пов'язані з обслуговуванням виробництва, утриманням апарату управління спілкою та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20-60 % від суми основної та додаткової заробітної плати працівників.

$$\text{НВ} = 0{,}5 \cdot \text{ЗП}_{\text{осн}} \quad (4.7)$$

де $H_в$ – накладні витрати.

Отже, накладні витрати:

$H_{в1}$ = 8320· 0,5 = 4160 грн. $H_{в2}$ = 9545 · 0,5 = 4772,5 грн.

4.3 Визначення періоду окупності та собівартості

Проведемо розрахунок вартості створюваного програмного продукту. Вартість продукції включає у собі собівартість і планований прибуток.

Приймемо прибуток на рівні 30%. Для нових інноваційних продуктів, що користуються високим попитом на ринку, ринкову вартість Вр можна втановити вищу.

Отже, вартість розробленого програмного забезпечення:

Вр1,2= Св1,2 + 0,3 · Св1,2 = 11000 + 0,3 · 11000 = 14300 грн..

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу [20]. Економічна ефективність ($E_р$) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_р = \frac{П}{C_в} \tag{4.8}$$

де $П$ – прибуток;

$C_в$ – собівартість.

Плановий прибуток ($П_{пл}$) знаходимо за формулою:

$$П_{пл} = В_р - C_в \tag{4.9}$$

Розраховуємо плановий прибуток:

$П_{пл}$ = 14300 – 11000 = 3300 грн.

Отже, формула для визначення економічної ефективності набуде вигляду:

$$E_р = \frac{П_{пл}}{C_в}. \tag{4.10}$$

$$E_р = 3300 / 11000 = 0,3.$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($T_р$):

$$Т_{ок} = \frac{1}{Е} \qquad (4.11)$$

Термін окупності дорівнює:

$$Т_{ок} = 1 / 0,3 = 3,3 \text{ роки}$$

У нашому випадку Ток1= Ток2 =1/0,30=3,33 років, що є нормальним, оскільки допустимим вважається термін окупності до 5 років.

Даний розрахунок виконаний у розрахунку на 1 екземпляр програмного продукту без врахування його тиражування.

Загальна вартість пропонованих робіт по розробці програмного продукту становить 14300 грн. Оскільки ефективність для обидвох проектів відповідно до встановленого рівня прибутку становить 0,3, що є високим показником, то проводити дані роботи варто і вкладені кошти окупляться за 3,33 року. Також слід врахувати можливість не одиничного замовлення програми, відповідно її ціна в такому випадку значно понизиться, а при продажі понад план прибуток зросте.

Виходячи із експертних оцінок і складності програми, приймемо величину витрат на супровід і модернізацію програмного забезпечення, створеного за процедурним методом 60% від початкових витрат, а за об’єктно-орієнтованим – 20%.

Собівартість модернізації:

$$С_вМ_1 = 0,6 \cdot С_{в1} = 0,6 \cdot 11000 = 6600 \text{ грн.},$$

$$С_вМ_2 = 0,2 \cdot С_{в2} = 0,2 \cdot 11000 = 2200 \text{ грн.}$$

Для споживача вартість модернізації:

$$М_1 = 0,6 \cdot В_1 = 0,6 \cdot 14300 = 8580 \text{ грн};$$

$$М_2 = 0,2 \cdot В_1 = 0,2 \cdot 14300 = 2860 \text{грн.}$$

Таким чином, уже після першої модернізації, загальні витрати на створення і супровід ПЗ для виробника за об’єктно-орієнтованим методом менші, ніж за процедурним, навіть якщо його собівартість є дещо дорожчою.

$ЗВ_{1\ (вир)} = 11000 + 6600 = 17600$ грн.;

$ЗВ_{2\ (вир)} = 11000 + 2200 = 13200$ грн..

Як і для споживача:

$ЗВ_1 = 14300 + 8580 = 22880$ грн.;

$ЗВ_2 = 14300 + 2860 = 17160$ грн..

Річна економія витрат за всіма можливими напрямами і додатковими витратами, пов'язаними з супроводом і тільки одноразовою модернізацією (у розрахунку на одиницю продукції) при об’єктно-орієнтованому методі порівняно із процедурним:

$\Delta С_{\ (вир)} = ЗВ_{1\ (вир)} - ЗВ_{2\ (вир)} = 17600 - 13200 = 4400$ грн.;

$\Delta С = ЗВ_1 - ЗВ_2 = 22880 - 17160 = 5720$ грн..

Чистий приведений дохід (ЧПД) визначається як різниця між сукупними доходами (сукупний грошовий потік) і сукупними витратами (сукупними інвестиціями) взятими за весь період життя інвестицій і дисконтова ними в кожному році на фактор часу. Дисконтування являє собою визначення вартості майбутніх грошових потоків у теперішній момент часу.

Коефіцієнт дисконтування показує, яку величину грошових коштів ми отримаємо з урахуванням фактору часу та ризиків. Він дозволяє перетворити майбутню вартість у вартість на даний момент [18].

Для розрахунку коефіцієнта дисконтування (коефіцієнта приведення) грошових потоків за роками періоду економічного життя інвестицій використовується формула:

$$\alpha = \frac{1}{(1+i)^n}; \quad (4.12)$$

де i – ставка дисконтування або норма дисконту, i = 0,2;

n – час або кількість періодів (років), протягом якого планується отримання доходу.

$$\alpha_0 = 1, \alpha_1 = \frac{1}{1+0,2} = 0,83.$$

Вважатимемо, що обидва програмних продукта одинаково забезпечують потреби і вимоги споживача, і тому придбання першої чи другої програми однаково вплинуть на розмір його додаткових доходів на вкладений капітал. Тому приймемо цю величину за постійну, а порівняння дохідності двох проектів проведемо тільки за витратами.

$ЧПД_1' = ГП + 0,83 \cdot ГП = 14300 - 0,83 \cdot 8580 = 1,83ГП - 7178,6$ грн.;

$ЧПД_1' = ГП + 0,83 \cdot ГП = 14300 - 0,83 \cdot 2860 = 1,83ГП - 11926,2$ грн..

Чим менші витрати, тим більша дохідність проекту.

$ЗВ_1 = 14300 + 8580 = 22880$ грн.;

$ЗВ_2 = 14300 + 2860 = 17160$ грн..

Таблиця 4.3 – Техніко–економічні показники програмного продукту

Показник	Процедурний підхід	Об'єктно-орієнтований підхід
Зарплата основна, грн	8320	9545
Зарплата додаткова, грн	1664	1909
Фонд заробітньої плати, грн	9984	11454
Відрахування на ФОП, грн	3441,67	4211,64
Разом на виплату плаці, грн	11761,67	13756,64
Матеріальні витрати, грн	630	630
Електроенергія, грн	152	176
Амортизація, грн	657,7	761,54
Накладні витрати, грн	4160	4772,5
Разом на ін.витрати, грн	3651,44	4467,84
Собівартість	11000	14300
Прибуток	3300	3660
Вартість розробленого ПЗ	11000	14300
Економічна ефективність	0,30	0,30

Термін окупності, років	3,33	3,33
Собівартість модернізації	8580	2860
Супровід і модернізація	22880	17160
Загальні витрати на розробку	18411,15	14080,86
Порівняльна економія витрат (для виробника)	-	4330,29
Загальні витрати (для споживача, на придбання програмного прод.)	22880	17160
Порівняльна економія витрат для споживача)	-	5 720
Дохідність проекту для споживача за витратною частиною	−4468,85	−13129,43
Економія	-	8660,58

Економія витрат у випадку придбання, супроводу і одноразової модернізації програмного продукту, створеного за об’єктно-орієнтованим підходом, становить 8660,58грн.

Оскільки ефективність для обидвох проектів відповідно до встановленого рівня прибутку становить 0,3, що є високим показником, то проводити дані роботи варто і вкладені кошти окупляться за 3,3 року, бо нормальним терміном окупності є термін, який коливається від 1 до 3 років, тоді розробка вважається доцільною і економічно вигідною [19].

При використанні об’єктно-орієнтовного підходу зменшується кількість працівників, які залучаються у проект, та зменшуються витрати на реалізацію проекту, але для підтримки проекту і його подальшої модернізації.

Отже, програмний продукт може бути впроваджений та мати подальший розвиток, оскільки він є економічно вигідною за всіма основними техніко-економічними показниками.

4.4 План маркетингу

Головною метою є проникнення на ринок і наступне розширення ринкової частки. Головною стратегією повинна стати комплексна стратегія по наданню продукції більш високої якості і за нижчими цінами. Виходячи з цього, стратегією маркетингу обирається стратегія розширення попиту за рахунок стимулювання обсягу продажів, цінової політики і нецінових факторів конкурентної боротьби, створення позитивного іміджу програмного забезпечення. Виходячи з цілей і стратегії маркетингу, а також з урахуванням еластичності попиту, встановлення цін буде здійснюватися методом "витрати + прибуток», з урахуванням величини очікуваного попиту і поведінки конкурентів. Ціни на продукцію будуть розраховуватися виходячи з рівня попиту і витрат та цільового прибутку.

Одним з важелів може бути посилення рекламної компанії і застосування незвичайного прийому маркетингу: кожному покупцеві забезпечимо можливість 5% знижки на продукцію, при покупці 2х і більше одиниць ПЗ.

Для вступу на ринок потрібно робити основний акцент в маркетинговій стратегії на проведення рекламної кампанії. Від цього залежить успіх просування товару. Потрібно масштабна рекламна компанія, яка буде орієнтуватися на вище перераховані категорії.

Переваги продукції, що випускається передбачається доводити до потенційного покупця і споживача декількома шляхами:

- Відео кліпи;
- Радіореклама;
- Анкетування;
- Рекламні оголошення в журналах і газетах.

Планується відраховувати на ці потреби від 5% до 10% обсягу річного прибутку.

5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Охорона праці

Враховуючи, що при використання автоматичної системи управління мінімаркетом, користувачі проводять більшу частину свого робочого часу за комп'ютером, необхідно дотримуватися правил охорони праці при роботі з персональним комп'ютером.

Питання охорони праці регулюються певними законодавчими та нормативно-правовими актами, які, зокрема, визначають обов'язки роботодавця із забезпечення робітникам комфортних та безпечних умов для здійснення роботи. Ці обов'язки, а також права робітників на таких умовах праці передбачені частиною 2 ст.2 і частиною 1 ст.21 КЗпП, а також ст.13 Закону України «Про охорону праці», у яких визначаються основні положення з реалізації конституційного права робітників.

Існує цілий ряд вимог, які визначають специфіку заходів з охорони праці при роботі з персональним комп'ютером. Законодавчі та нормативно-правові акти, які за участі відповідних органів державної влади регулюють відносини між роботодавцем та робітником з питань безпеки, гігієни праці та виробничого середовища, а також встановлюють єдиний порядок організації охорони праці в Україні. На їх основі розроблені чисельні документи: правила, інструкції, норми, державні санітарні правила та ін., якими мають керуватись роботодавці та які регламентують певні питання щодо конструкції електронно-обчислювальної техніки, та особливостей їх розміщення [19].

На сьогодні основними документами, які регламентують питання охорони праці при використанні працівниками персональних комп'ютерів, можна вважати наступні підзаконні акти:

1) НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями»;

2) ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи

з візуальними дисплейними терміналами електронно-обчислювальних машин»;

У відповідності з цими актами, при розробці будь-якого програмного забезпечення, в тому числі при розробці автоматизованої системи управління мінімаркетом, необхідно вжити всіх необхідних заходів з охорони праці та розробити відповідні документи, зокрема, але не виключно:

- положення про охорону праці;
- інструкції з охорони праці для касира та адміністратора;
- накази з охорони праці;
- журнали інструктажу, реєстрацій та інше.

Окрім цього, в залежності від кількості працівників в ІТ компанії та від кількості працівників задіяних в проекті по розробці системи управління мінімаркетом служба охорони праці виглядає наступним чином:

- В ІТ компанії або проекті з кількістю працюючих 50 і більше осіб, роботодавець створює службу охорони праці відповідно до типового положення, що затверджується центральним органом виконавчої влади, що забезпечує формування державної політики у сфері охорони праці.
- В ІТ компанії або проекті з кількістю працюючих менше 50 осіб, функції служби охорони праці можуть виконувати в порядку сумісництва особи, які мають відповідну підготовку.
- В ІТ компанії або проекті з кількістю працюючих менше 20 осіб, для виконання функцій служби охорони праці можуть залучатися сторонні спеціалісти на договірних засадах, які мають відповідну підготовку.

Підпорядковується служба охорони праці згідно із законодавством безпосередньо роботодавцеві [20].

Проте роботодавець може доручити функціональне управління (кураторство) діяльністю служби іншій посадовій особі, скажімо, головному інженерові, заступникові директора з охорони праці тощо.

Покладення таких обов’язків потрібно закріпити наказом або відобразити в посадовій інструкції уповноваженої особи.

Робота служби охорони праці підприємства має здійснюватися відповідно до плану роботи та графіків обстежень, затверджених роботодавцем.

Функції служби охорони праці:

1. Підготовка проектів наказів (розпоряджень) з питань охорони праці і внесення їх на розгляд роботодавцю. Проведення спільно з представниками інших структурних підрозділів і за участю представників професійної спілки підприємства або, за її відсутності, уповноважених найманими працівниками осіб із питань охорони праці перевірок дотримання працівниками вимог нормативно-правових актів з охорони праці.

2. Проведення з працівниками вступного інструктажу з питань охорони праці та супутніх інструктажів.

3. Ведення обліку та проведення аналізу причин виробничого травматизму, професійних захворювань, аварій на виробництві, заподіяної ними шкоди.

4. Забезпечення належного оформлення і зберігання документації з питань охорони праці, а також своєчасної передачі її до архіву для тривалого зберігання згідно з установленим порядком.

5. Складання звітності з охорони праці за встановленими формами.

6. Складання за участю керівників підрозділів підприємства переліків професій, посад і видів робіт, на які повинні бути розроблені інструкції з охорони праці, що діють в межах підприємства, надання методичної допомоги під час їх розроблення.

7. Інформування працівників про основні вимоги законів, інших нормативно-правових актів та актів з охорони праці, що діють в межах підприємства.

8. Розгляд питань про підтвердження наявності небезпечної виробничої ситуації, що стала причиною відмови працівника від виконання дорученої роботи відповідно до законодавства (у разі необхідності).

9. Організація - забезпечення підрозділів нормативно-правовими актами з охорони праці та актами з охорони праці, що діють в межах підприємства, посібниками, навчальними матеріалами з цих питань [19].

Дотримання даних пунктів є необхідним при роботі з автоматизованою системою управління мінімаркетом.

5.2 Безпека в надзвичайних ситуаціях

Загальні ергономічні вимоги для організації робочого місця користувача ПЕОМ (ГОСТ 12.2.049-80, ГОСТ 122032-78, ГОСТ 22269-76). Ці вимоги встановлюють основні параметри робочого місця, оснащеного дисплеєм, і враховують особливість виконуваних робіт.

5.2.1 Параметри робочого місця

Площа кабінету, в якому буде проходити робота повинна бути не менш 6 м², а об’єм не менш 24 м³. Для внутрішньої обробки приміщення повинні використовуватися дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі – 0,7-0,8; для стін – 0,5-0,6; для підлоги – 0,3-0,5.

Конструкція робочого столу повинна забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання. Конструкція крісла повинна забезпечувати підтримку раціональної робочої пози під час роботи з відео-дисплейним терміналом (Далі ВДТ) і ПЕОМ, дозволяти змінювати позу з метою зниження статичного напруження м'язів шийно-плечової області і спини для попередження розвитку втоми працюючого (згідно з ГОСТ 12.2.032-78). Поверхня сидіння, спинки та інших елементів стільця (крісла) повинна бути напівм'якою, з покриттям, що не електризується, неслизьке та повітронепроникне, що забезпечує легке очищення від забруднення.

Висота робочої поверхні столу, за відсутності можливості її регулювання повинна складати 725 мм. Робочий стіл повинен мати простір для ніг висотою не менше 600 мм, шириною – не менше 500 мм, не менше 450 мм в глибину на рівні колін і на рівні простягнутої ноги – не менше 650

мм. Робоче місце має бути обладнане підставкою для ніг, має ширину не менше 300 мм, глибину не менше 400 мм, регулювання по висоті в межах 150 мм за кутом нахилу опорної поверхні підставки до 20 градусів.

Відстань від очей користувача до екрану дисплея має становити 500-700 мм.

Кут зору 10-20°, але не більше 40°; кут між верхнім краєм дисплея і рівнем очей користувача має становити не менше 10°. Кращим є розташування екрану перпендикулярно до лінії зору користувача. Робочі місця по відношенню до світлових прорізів повинні розташовуватися не ближче 3 м так, щоб природне світло падало збоку, переважно зліва. Освітленість також впливає на стан здоров'я і працездатність людини. У відповідності зі СНіП 11-4-79 встановлені наступні вимоги до освітленості:

Для штучного освітлення:

- Комбіноване освітлення – освітленість 1500 лк;
- Загальне освітлення – освітленість 400 лк.

Для природного освітлення:

- Верхнє або комбіноване освітлення – коефіцієнт природної освітленості (далі КПО) 10%;
- Бічне освітлення – КПО 3.5%.

Для суміщеного освітлення:

- Верхнє або комбіноване освітлення – КПО 3-6%;
- Бічне освітлення – КПО 1.1-2%.

До основних показників, що визначають умови здорової роботи, належать: фон, контраст об'єкта з фоном, видимість, показник осліпленості, коефіцієнт пульсації освітленості.

Фон характеризується коефіцієнтом відбиття. Контраст об'єкта з фоном (К) характеризується співвідношенням яскравості розглянутого об'єкта (точки, лінії, знаки) і фону. Оскільки роботи користувача ПЕОМ відносяться до категорії 1а – легкі фізичні роботи (роботи проводяться сидячи і супроводжуються незначним фізичним напруженням, з енерговитратами до 120 ккал / годину), необхідно дотримуватися наступних

норм: коефіцієнт відображення більше 0,4, тобто світлий фон; контраст об'єкта з фоном великий і середній при К більше 0,2 (згідно СНіП 11-4-79).

У полі зору користувача ПЕОМ має бути забезпечений відповідний розподіл яскравості. Відношення яскравості екрана до яскравості оточуючих його поверхонь не повинно перевищувати у робочій зоні 3:1 (СНіП 11-4-79).

У зв'язку з цим дисплей ПЕОМ повинен відповідати наступним вимогам:

- Яскравість свічення екрану не менше 100 кд/м;
- Мінімальний розмір світної точки для кольорового дисплея не більше 0,6 мм ;
- Контрастність зображення знаку – не менше 0,8;
- Низькочастотне тремтіння зображення в діапазоні 0,05-1,0 Гц повинно знаходитися в межах 0,1 мм;
- Екран повинен мати покриття антивідблиску;
- Відеомонітор повинен бути обладнаний поворотним майданчиком, що дозволяє переміщати відеотермінал в горизонтальній і вертикальній площинах в межах 130-220 мм і змінювати кут нахилу на 10-15 мм.

Коефіцієнт відбиття світла матеріалами і обладнанням всередині приміщень має велике значення для освітлення: чим більше світла відбивається від поверхонь, тим вище освітленість. Коефіцієнт відображення відповідно повинен бути для: стелі 60-70%, стін 40-50%, підлоги 30%, для інших поверхонь 30-40%.

Результати досліджень показують, що найбільшою мірою негативний фізіологічний вплив на операторів ПК пов'язаний з дискомфортними зоровими умовами через неправильно спроектоване освітлення. Згідно СНіП II-4-79 освітленість на горизонтальній площині робочого місця оператора ЕОМ повинна складати 400 лк при висоті цій площині 0,8 м над підлогою.

5.2.2 Вимоги до освітленості і повітряного середовища в робочій зоні

Світловий клімат визначає зоровий дискомфорт. Запобігти шкідливому впливу освітлення можна шляхом правильного підбору системи освітлення, джерел світла (за їх спектрального складу випромінювання), світильників. Коли штучне світло змішується з природним, рекомендується використовувати лампи за спектральним складом найбільш близькі до сонячного світла.

Світильники слід вибирати з розсіювачами, а блискучі деталі освітлювального обладнання, що можуть потрапити в поле зору оператора, повинні бути замінені на матові. Розташовувати робоче місце, обладнане дисплеєм, необхідно таким чином, щоб у полі зору оператора не потрапляли вікна або освітлювальні прилади; вони не повинні знаходиться і безпосередньо за спиною оператора. Вікна в приміщеннях з дисплеями обладнають шторами з коефіцієнтом відображення 0,5 ... 0,7, стіни фарбують матовою фарбою з коефіцієнтом відображення 0,4 ... 0,6. Світловий клімат може бути поліпшений шляхом встановлення спеціальних антивідблискових контрастних фільтрів, однак при виборі типу фільтра необхідно враховувати умови роботи з комп'ютером, оскільки оптимальні значення коефіцієнтів пропускання і дзеркального відображення фільтрів залежать від освітленості робочого місця і типу джерела світла.

Враховуючи великий вплив освітлення на працездатність оператора при роботі з комп'ютером, проведемо розрахунок необхідної освітленості в приміщенні з дисплеями при наступних умовах: гігієнічна норма освітленості на горизонтальній поверхні на рівні робочого місця оператора – 400 лк; ширина приміщення – 7 м, довжина – 8 м, висота – 3 м. Коефіцієнт відбиття від стелі – 70, від стін – 50, від робочих поверхонь – 30. Повітряне середовище – нормальне (вміст пилу, диму й кіптяви не більше 5 мг/м^3).

Повітряне середовище в робочій зоні визначається мікрокліматом виробничого приміщення. Величини температури, відносної вологості та швидкості руху повітря на робочих місцях з дисплеями повинні відповідати

допустимим значенням, які встановлені ГОСТ 12.1.005-88 ССБТ для категорії робіт 1а (легкі фізичні роботи, вироблені сидячи і супроводжуються незначною фізичною напругою до 120 ккал/год.). Згідно з цим документом допустимі значення температури повітря в приміщенні становлять 19-25 °С, відносної вологості повітря – 55%, швидкості руху повітря на рівні особи – 0,1 м/С. При наявності досить комфортного робочого середовища атмосферний тиск по ГОСТ 21552-84 ССБТ може змінюватися від 84 до 107 кПа (630 ... 800 мм рт. ст.).

Шум несприятливий для людини, особливо при тривалому впливі. В оператора це виражається в зниженні працездатності (наприклад, швидкість обробки тексту зменшується на 10-15%), у прискоренні розвитку зорового стомлення, зміну відчуття кольору, підвищенні витрати енергії (на 17%). Тривалий та інтенсивний шум значно знижує продуктивність праці і призводить до зростання кількості помилок у роботі. У відділі головного економіста шум може створюватися телефонними дзвінками та розмовами, системними блоками та клавіатурою ПЕОМ. Так само джерелами шуму можуть бути системи кондиціювання та вентилювання повітря, існують і зовнішні джерела шуму (наприклад, працюють агрегати на вулиці).

5.2.3 Допустимі рівні звуку на робочих місцях

Допустимі рівні звуку та еквівалентні рівні звуку на робочих місцях повинні відповідати вимогам «Санітарних норм допустимих рівнів шуму на робочих місцях» № 3223-85. Згідно з цими нормами в приміщенні, де працює користувач ПЕОМ для забезпечення оптимальної робочої середовища рівень шуму не повинен перевищувати 60 дБ. Основними заходами боротьби з шумом згідно з ГОСТ 12.1.029-80 ССБТ є ліквідація або ослаблення джерела шуму шляхом застосування звукопоглинаючих матеріалів у конструкціях механізмів, використання коштів звукопоглинання і раціональна планування виробничого приміщення.

Випромінювання ПК можуть бути небезпечними для здоров'я. Низькочастотні поля при тривалому опроміненні сидять біля ПК людей можуть привести до порушень самих різних фізіологічних процесів. Відповідно до ГОСТ 27016-86 і ГОСТ 27954-88 потужність дози рентгенівського випромінювання в будь-якій точці простору на відстані 5 см від екрану відеомонітора при 41 годинному робочому тижні не повинна перевищувати 100 мкР/год (0,03 мкР/с), а інтенсивність ультрафіолетового випромінювання – 10 Вт/м².

В даний час випускаються вибухобезпечні відеомонітори. За способом захисту людини від ураження електричним струмом дисплеї виготовляються відповідно з 1-м класом захисту за ГОСТ 25861-84, тому кабель живлення дисплея має вилку з трьома виводами, один з яких заземлюючий.

Для забезпечення ПДУ чинників робочого середовища на робочих місцях у необхідних випадках використовуються спеціальні засоби захисту працюючих. Способи захисту бувають активними і пасивними. Способи активного захисту засновані на виявленні джерел несприятливих факторів і вплив на них. У випадках неможливості здійснення активного захисту застосовується пасивна, за якої джерела несприятливих факторів залишаються, але здійснюються заходи, спрямовані на попередження вплив цих факторів на людину. Пасивна захист може бути колективного та індивідуального. Розглянемо колективні засоби захисту оператора ПК.

Висока температура повітря негативно позначається на функціональному стані людини. Всі основні електронні блоки ПК мають вбудовані вентилятори для забезпечення стабільних температурних режимів їх функціонування, тому при створенні комфортних умов роботи особливу увагу необхідно приділити шляхам відводу повітря (припливно-витяжної вентиляції).

Для захисту від електростатичного потенціалу і, певною мірою, від електричної складової змінного електромагнітного поля (ЕМП) можуть бути

використані згадані вище антиблискові контрастуючі фільтри на екрани дисплеїв.

ВИСНОВКИ

Під час виконання даної роботи було реалізовано додаток для управління мінімаркетом для оптимізації та спрощення продажів. Проаналізовано додатки зі схожим функціоналом та застосовано в даному додатку і компенсовано слабкі сторони. Специфіка даного сегменту полягає у тому, що потрібно забезпечити якісну роботу з базою даних та сканером штрих-кодів. Актуальність тему пов’язана з великою кількістю мінімаркетів, які потребують автоматизації і покращення роботи.

Для розробки даного додатку було застосовано технології C# та .Net. Було використано ключові можливості даних інструментів. Фреймворк windows form дозволив максимально швидко, якісно та зручно розробити графічний інтерфейс система, продемонструвавши всі свої сильні сторони. Було застосовано велику кількість функцій пов’язаних з форматуванням таблиць. Окрім цього було досліджено на застосовано ключові можливості мови програмування C# та інший її функціонал.

Варто також зазначити, що для такого типу програмного забезпечення, надзвичайно зручною виявилася база даних MySql. Одною з її найбільших переваг можна вважати швидкодію, яка на повністю забезпечували всі поставлені перед нею задачі. Workbench – дозволив якісно та швидко створити ключові представлення для роботою з базою даних.

Широкий функціонал дозволяє за допомогою декількох простих кроків створювати представлення та взаємозв’язки між ними. Також зручною виявилася функція автоматичної побудови діаграм для представлення ключових сутностей системи.

Діаграми UML дозволяють максимально якісно та доступно продемонструвати ключові можливості системи да її акторів, в зв’язку з чим і були застосовані при проектування системи. Діаграми станів та послідовностей допомогли більше заглибитися в специфіку та глибше в ній розібратися, що дозволили розробити максимально якісний продукт.

Visual studio дозволило пришвидшити процес розробки та спростити певні аспекти розробки. Розроблена програма дозволяє користувачам дізнаватися вартість та іншу важливу інформація про продукт за допомогою штрих-коду. Після дослідження предметної області, було встановлено, що програмою будуть користуватися два типи користувачів, це касир та адміністратор. Саме для цих користувачів і розроблений даний додаток, який спростить процес розробки та дозволить максимально збільшити ефективність їх роботи. Також була розроблена форма авторизація користувачів, яка допомагає максимально спростити процес входу в систему та роботу з нею. Весь функціонал системи було протестовано на швидкодію так коректність роботи. Тести показали, що швидкість роботи програмного забезпечення дозволить максимально якість та швидко виконувати ті функції, які вимагаються від програми. Розроблена система не є ресурсо затратною, що дозволить запустити її навіть на комп’ютерах з слабким апаратним забезпеченням. Мінімальні вимоги передбачають, що версія операційно системи повинна бути принаймні Windows 7.

Завдяки грамотному проектуванні та використання коректних та прагматичних інструментів було створено програмний продукт, який дозволяє спростити роботу з продажами товарів в магазинах і як наслідок оптимізувати його роботу. Використавши аналізи економічної частини розробки програмного продукту вдалося розробити доступний продукт, який знайде своїх покупців на ринку, адже існуючі вже програми не у всіх

випадках дозволяють оптимізувати або автоматизувати роботу для збільшення прибутку та полегшення роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація C#. – [Електронний ресурс] - 2019 Режим доступу https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/.
2. Офіційна документація Windows Forms. – [Електронний ресурс] 2019 Режим доступу https://docs.microsoft.com/en-us/dotnet/framework/winforms/.
3. Офіційна документація MySql. – [Електронний ресурс] - 2019 Режим доступу: https://dev.mysql.com/doc/refman/8.0/en/.
4. Офіційна документація Workbench – [Електронний ресурс] - 2019 Режим доступу https://dev.mysql.com/doc/workbench/en/.
5. Офіційна документація Visual Studio. – [Електронний ресурс] - 2019 Режим доступу https://docs.microsoft.com/en-us/visualstudio/ide/?view=vs-2019.
6. Scott W. Amble. The Elements of UML(TM) 2.0 Style / Scott W. Amble.. – 202 c.
7. Craig Larman. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition) / Craig Larman.. – 736 c.

8. Michael B. White. Mastering C# (C Sharp Programming): A Step by Step Guide for the Beginner, Intermediate and Advanced User, Including Projects and Exercises / Michael B. White. – Centerville Road Suite 400 Wilmington: Amazon Digital Services LLC. – 517 с.

9. Joseph Albahari. C# 7.0 in a Nutshell: The Definitive Reference 1st Edition / Joseph Albahari. – 41 E University Ave, Champaign, IL 61820,: O'Reilly Media; 1 edition. – 1088 с.

10. Martin Kleppmann. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems / Martin Kleppmann. – 41 E University Ave, Champaign, IL 61820,: O'Reilly Media; 1 edition. – 616 с.

11. Jamie Chan. SQL: Learn SQL (using MySQL) in One Day and Learn It Well. SQL for Beginners with Hands-on Project. / Jamie Chan. – 2711 Centerville Road Suite 400 Wilmington, DE 198: Amazon Digital Services LLC. – 166 с.

12. Don Norman. The Design of Everyday Things: Revised and Expanded Edition / Don Norman.. – 368 с.

13. Офіційна документація фреймворку Iron Barcode. – [Електронний ресурс] - 2019 Режим доступу https://ironsoftware.com/csharp/barcode/.

14. Luke Fatooros. 1-PAGE PLANNER: How to Discover your Red-Hot Niche of Cash-Paying Raving Fans. Dominate it. Become the Business Customers Want to Buy From / Luke Fatooros. – 2711 Centerville Road Suite 400 Wilmington, DE 198: Amazon Digital Services LLC. – 84 с.

15. Chris Sells. Windows Forms Programming in C# / Chris Sells. – Boston: Addison-Wesley Professional. – 736 с. – (1st Edition).

16. Robert C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert C. Martin.. – 432 с. – (1 edition)..

17. Michael Keeling. Design It!: From Programmer to Software Architect / Michael Keeling.. – 360 с.

18. Богданюк В.Є. Методичні вказівки до виконання організаційно-економічного розділу дипломних проектів / Богданюк В.Є., Березовський К.В., Пашін В.П. – К.: НТУУ "КПІ", 1999. – 66 с.

19. Методичні вказівки до виконання магістерської роботи освітнього рівня “магістр” студентами усіх форм навчання для напряму підготовки 121 – “Інженерія програмного забезпечення” / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 26 с.

20. Методичні рекомендації по виконанню розділу техніко-економічного обґрунтування дипломних робіт студентами технічних спеціальностей напряму підготовки 8.05010302 «Інженерія програмного забезпечення» освітньо-кваліфікаційного рівня «Магістр» / Укладачі: Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

ДОДАТКИ

ДОДАТОК А - ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра програмної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедрою
програмної інженерії

"___ "_____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської роботи

на тему: «Розробка автоматизованої системи управління мінімаркетом для оптимізації роботи з продажами на базі програмного середовища C# .Net»
Кухаруку Юрій Олександровичу

Керівник роботи:
к.ф.-м.н., доцент Бойко І. В.
"___" _____ 2019 р.
_____

Виконавець:
студент групи СПм-61
Кухарук Юрій Олександрович
"___" _____ 2019 р.
_____

м. Тернопіль – 2019

ЗМІСТ

1. ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану на 2019 рік, та згідно наказу на виконання магістерської роботи студента-магістра.

Тема проекту: «Розробка автоматизованої системи управління мінімаркетом для оптимізації роботи з продажами на базі програмного середовища C# .Net».

2. ПРИЗНАЧЕННЯ РОЗРОБКИ

Автоматична система управління продажами мінімаркету дозволяє максимально спростити та пришвидшити процес обслуговування клієнта, що в свою чергу сприяє посиленню продуктивності праці та збільшенню прибутку.

Метою магістерської роботи є розробка автоматичної системи управління продажами мінімеркату. Необхідно проаналізувати предметну область та дослідити програмні системи з аналогічним функціоналом.

В якості автоматично системи управління мінімаркетом необхідно розробити систему, яка забезпечить користувачів максимальним функціоналом. Система повинна працювати з сканером штрих-кодів та забезпечувати можливість входу в систему різних користувачів. Окрім цього необхідно забезпечити зручність системи та інтуїтивно зрозумілий інтерфейс.

За результатами виконаної роботи необхідно розробити автоматизовану систему управління мінімаркетом, яка дозволить пришвидшити та спростити роботу персоналу магазину.

3. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Функціональні характеристики

Програмне забезпечення має виконувати наступні дії:

- сканувати товари за допомогою сканера штрих-кодів;
- працювати з дисконтними картами клієнтів;
- підтримувати можливість виходу та входу в систему, запам’ятовувати користувача в системі;
- автоматично вираховувати знижку на базі наданої дисконтної карти;
- додавати, видаляти та редагувати дані про користувачів;
- додавати дані про новий товар;
- формувати звітність за певний проміжок часу;
- підтримувати одночасну роботу з 40 та більше користувачами.

3.2 Склад та параметри технічних засобів

1) ПК або планшетний комп’ютер з 1024 Мб оперативної пам‘яті, встановленою системою Windows 7, 8, 8.1, 10, MacOS, Не менше 200 Мб вільного місця на жорсткому диску. Двоядерний процесор з тактовою частотою від 1.2 GHz і більше.

2) наявність встановленої бібліотеки .Net.

3.3 Інформаційна та програмна сполучність

Програмний продукт повинен коректно функціонувати в операційних системах Windows, різних поколінь, на яких доступне для встановлення бібліотека фреймворку .Net. Розроблювана бібліотека класів повинна бути пристосована до використання у інформаційних системах та програмних засобах. Розробку виконувати з використанням бібліотек та технологій мови C# в середовищі програмування Visual Studio 2017 з використанням .Net.

4. СТАДІЇ РОЗРОБКИ

В ходів реалізації роботи проект повинен пройти крізь наступні стадії розробки:

- аналіз предметної області;
- проектування архітектури;
- реалізація архітектури;
- реалізація графічного інтерфейсу;
- тестування результатів розробки;
- оформлення супровідної документації;
- здача роботи.

5. ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для програмного продукту повинні бути розроблені наступні документи:

- Пояснювальна записка;
- Технічне завдання;
- Презентаційний матеріал;
- Додатки.

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Розроблений програмний продукт має виконувати всі вимоги, що складаються з перерахованих у п. 3.1 характеристик.

Приймання проводиться спеціально створеною екзаменаційною комісією в термін до:

“__” грудня 2019р.

УДК 004.422.83
Ю.О. Кухарук – магістрант
Тернопільський національний технічний університет імені Івана Пулюя, Україна
І.В. Бойко – кандидат фізико – математичних наук, доцент
Тернопільський національний технічний університет імені Івана Пулюя, Україна

УДОСКОНАЛЕННЯ АВТОМАТИЧНИХ СИСТЕМ УПРАВЛІННЯ ПРОДАЖАМИ В МАГАЗИНАХ

Y.O. Kukharuk – magistrate I.V. Boyko – Ph.D, Assoc. Prof.

IMPROVING AUTOMATIC SHOP MANAGEMENT SYSTEMS

Сучасний магазин представляє собою великий вибір товар різних груп та категорій де покупець має можливість знайти все, що його цікавить в одному місці. Враховуючи кількість пропозицій товарів у магазинах та потребу в акційних продажах для підвищення прибутку, можна зробити висновок, що не всі системи задовільнять потреби сучасного покупця. Існуючі системи досить часто не пристосовані до великої кількості товарів та постійних цінових змін в середині системи [1]. Також у великих маркетах не завжди вчасно відбувається зміна цінників. Проте за допомогою сучасних комунікаційних систем та мереж появляється безліч можливостей для вдосконалення існуючих систем та впровадження нових технологій.

Як вже було описано вище, магазини залучають нових клієнтів використовуючи акційні пропозиції [1]. Це один з способів модернізації існуючих систем. Варто включати механізми, які роблять можливим повідомлення покупців про ці пропозиції. В даному випадку існує декілька шляхів вдосконалення. Можливі електронні банери, на яких рекламу можна транслювати через систему, також зібравши клієнтську бази номерів можлива СМС розсилка інформації про знижки та розпродажі [2]. Ще один спосіб, ще інтеграція існуючих систем та мобільних додатків для покупців. Це дозволить бачити усі акційні пропозиції та ціни на товари.

Наступною моментом, який потрібно вдосконалити, є актуальність цін. Варто додати можливість підключати до системи додаткові сканери штрих-кодів з дисплеєм для відображення ціни та встановити їх по цілому магазині [2]. Це дозволить покупцям самостійно перевіряти ціну на товари. Іншим варіантом вдосконалення є інтеграція функціоналу в мобільний додаток який дозволить зчитувати коди з товару та за допомогою мобільного пристрою перевіряти ціну.

Підсумовуючи весь матеріал можна дійти висновку, що програмна модернізація програмних систем в магазинах є необхідною для сучасного бізнесу та комерції. Особливу увагу потрібно приділити вирішенню проблем описаних вище. Проте перед цим потрібно також модернізувати і наявні системи , щоб вони відповідали всім вимогам, були зручними і зрозумілими у використанні та встановленні пересічному користувачеві персонального комп’ютерів та мобільних пристроїв.

Література

1. Апопій В. В. Сучасні проблеми та стратегічні пріоритети розвитку внутрішньої торгівлі України / В. В. Апопій // Вісник Донецького державного університету економіки і торгівлі ім. М. Туган-Барановського. – 2005. – № 4. – С. 145–153.
2. Леффінгуелл Д., Уідріг Д. Принципи роботи з вимогами до програмного забезпечення. Уніфікований похід, М .: «Вільямс», 2002. - 448с.

ДОДАТОК В - ДИСК