Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

# Кваліфікаційна робота бакалавра

на тему «Розробка касового додатку з використанням технологій C# та WPF»

Виконав: студент групи _____ІПЗ19-1_____
Спеціальність 121 Інженерія програмного забезпечення
_____Островський Вячеслав Дмитрович_____
(прізвище та ініціали)
Керівник к.т.н., доц. Ульяновська Ю. В._____
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент _____
(місце роботи)
_____
(посада)
_____
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2023

# АНОТАЦІЯ

Островський В. Д. Розробка касового додатку з використанням технологій C# та WPF.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2023.

Зміст анотації:

У результаті виконання кваліфікаційної роботи було створено касовий додаток на мові програмування C# з використанням технології створення графічних інтерфейсів WPF. Цей додаток пропонує рішення для управління продажами та обліку товарів. Використання C# та WPF дозволяє створити потужний та функціональний інтерфейс, забезпечуючи зручну та швидку взаємодію з додатком.

Ця робота пропонує нове технологічне рішення для поліпшення роботи касових систем, що дозволяє покращити продуктивність та точність обліку товарів. Розроблена програма має потенціал для використання у різних галузях, включаючи роздрібну торгівлю, ресторани, супермаркети та інші сфери діяльності, де важливо забезпечити швидкий та точний облік продажу.

Дана роботи відкриває перспективи для подальшого дослідження та розвитку касових систем з використанням сучасних технологій, дозволяючи підвищити ефективність та зручність використання для користувачів.

Ключові слова:

Касова програма, C#, ООП, класи, шифрування, сканування штрих-коду з веб-камери, формування чеку, база даних.

Ostrovskyi Viacheslav. Development of a cash register application using C# and WPF technologies.

Qualification work for obtaining a bachelor`s degree in specialty 121 "Software Engineering". – University of Customs and Finance, Dnipro, 2023.

Abstract content:

As a result of the diploma work, a cashier application was created in the C# programming language using the technology of creating WPF graphical interfaces. This application offers solutions for sales management and product accounting. Using C# and WPF allow us to create a powerful and functional interface, providing convenient and fast interaction with the application.

This work offers a new technological solution to improve the operation of cash register system, which allows to improve the productivity and accuracy of accounting of goods. The developed application has the potential to be used in various industries, including retail trade, restaurants, supermarkets and other areas of activity where it is important to ensure fast and accurate sales accounting.

This work opens prospects for further research and development of cash register systems using modern technologies, allowing to increase efficiency and ease of use for users.

Keywords:

Cashier application, C#, OOP, classes, encryption, scanning a barcode from a webcam, creating a check, database.

# ЗМІСТ

# ВСТУП

На сьогоднішній день існує велика кількість різноманітних програмних продуктів, які допомагають зберігати та обробляти інформацію, однак не завжди вони задовольняють потреби користувачів. У зв’язку з цим, є актуальною розробка нових програмних продуктів, які б враховували потреби різних користувачів та мали високу функціональність та ефективність.

Розробка касового додатку з використанням технологій C# та WPF є однією з актуальних тем в області програмування та розробки програмного забезпечення. Об'єктом даної роботи є процес розробки касового додатку, який використовується в різних галузях бізнесу. Предметом дослідження є розробка програмного забезпечення на базі технологій C# та WPF для створення ефективного та зручного касового додатку.

Актуальність роботи полягає в тому, що касові додатки є невід'ємною частиною бізнес-процесів в різних галузях. Вони дозволяють ефективно і точно обліковувати продажі, управляти запасами, формувати чеки, зчитувати штрих-коди товарів, виконувати операції зі знижками та іншими операціями, пов'язаними з касовими операціями. Крім того, з появою нових технологій та зростанням популярності електронної комерції, касові додатки стають ще більш важливими для підтримки бізнесу.

Метою кваліфікаційної роботи є автоматизація роботи працівників касових апаратів для ефективного обліку покупок клієнтів шляхом розробки програмного забезпечення з використанням технологій C# та WPF. Основними вимогами до ПЗ є розробка привабливого та зручного користувацького інтерфейсу, забезпечення сканування штрих-коду товару, формування чеку та інших операцій, розробка зручної системи управління знижками. Також, метою є забезпечення стабільної роботи програмного забезпечення та забезпечення його безпеки від несанкціонованого доступу.

Об'єктом кваліфікаційної роботи є процеси роботи працівників касових апаратів. Актуальним питанням є автоматизація їх роботи шляхом розробки програмного забезпечення. Робота спрямована на створення функціонального інтерфейсу для касирів, забезпечуючи зручну та швидку взаємодію з додатком. Головна увага виділяється реалізації наступних функцій, як виведення зображення товару на екран касиру для ідентифікації, сканування штрих-коду за допомогою веб-камери та робота з базою даних. Результати дослідження допоможуть визначити переваги та недоліки розробленого додатку і запропонувати рекомендації щодо подальшого вдосконалення та використання в практичних ситуаціях.

Предметом є розробка програмного забезпечення для автоматизації роботи працівників касових апаратів з використанням технологій C# та WPF. Предмет дослідження включає аналіз потреб та вимог користувачів касових систем, проектування то розробка гарного інтуїтивно-зрозумілого інтерфейсу для касирів, реалізацію функціональних можливостей, таких як виведення зображення товару на екран касиру, сканування штрих-коду та робота з базою даних. Предметом касового додатку, є його архітектура, модулі та компоненти, які визначають його функціональність та ефективність.

Для досягнення поставленої мети в роботі ставились та вирішувались наступні завдання:

- Дослідити існуючі касові системи та їх програмні рішення, що використовуються в різних сферах бізнесу.
- Проаналізувати потреби та вимоги користувачів касових систем з метою визначення ключових функцій та можливостей, які має містити розроблений касовий додаток.
- Реалізувати функціональні можливості, включаючи виведення зображення товару на екран касиру для його ідентифікації, сканування штрих-кодів товарів та взаємодією з базою даних.
- Розробити привабливий інтуїтивно-зрозумілий інтерфейс для касирів, забезпечуючи зручну та ефективну роботу з додатком.

- Оцінити переваги та недоліки розробленого додатку з позицій зручності використання, функціональності та продуктивності.
- Запропонувати рекомендації щодо вдосконалення та розширення можливостей касового додатку з урахуванням отриманих результатів та вимог користувача.

Для виконання поставлених завдань використовувались наступні методи:

- Аналіз літературних джерел та наукових досліджень з питань касових систем, що дозволило отримати огляд сучасних рішень та визначити ключові вимоги та тенденції в цій області.
- Проектування інтерфейсу користувача з використанням технологій WPF та Material Design in XAMLЮ що забезпечило створення зручного та привабливого інтерфейсу для користувачів.
- Розробка функціональності додатку з використанням мови програмування C# та платформи WPF, включаючи роботу з базою даних, зображеннями, штрих-кодами та іншими компонентами.
- Використання ADO.NET та EF.Core для роботи з базою даних, що забезпечило ефективне та безпечне зберігання та обробку інформації.
- Використання додаткових бібліотек і фреймворків, таких як BCrypt для шифрування паролів, ZXing.NET для сканування штрих-кодів та AForge.NET для обробки зображень переданих веб-камерою.

Програмне забезпечення, яке розроблювалось в дипломній роботі повинно виконувати наступні завдання:

- Забезпечити можливість додавання, видалення та редагування товарів при формуванні чеку;
- Реалізувати можливість розрахунку за допомогою касового апарату або безпосередньо з програми;
- Розробити зручний та інтуїтивно зрозумілий інтерфейс для користувача;
- Захистити дані користувачів від несанкціонованого доступу;
- Вирішити проблему ідентифікації товарів;

- Реалізувати можливість сканування штрих-коду за допомогою веб-камери.

Для вирішення поставлених завдань використовувались наступні методи, технології та інструменти, зокрема мова програмування C#, платформа .NET Framework, технологія WPF для розробки інтерфейсу користувача, бази даних SQL Server для зберігання даних про продажі, товари та інші операції.

В рамках дослідження буде проведено аналіз існуючих касових додатків та їх можливостей, будуть використані засоби ADO.NET та EF Core для роботи з базою даних, а також буде проведено тестування розробленого додатку з метою оцінки його продуктивності та функціональності.

Основними перевагами використання технологій C# та WPF для автоматизації касового додатку є висока продуктивність та швидкодія, максимальна можлива гнучкість в налаштуванні інтерфейсу користувача, а також можливість розробки багатофункціональних додатків з великою кількістю опцій та можливостей.

Одним із важливих аспектів є відповідність даного продукту потребам користувачів та ринковим потребам. У сучасному світі розвиток програмного забезпечення є динамічним процесом, тому важливо проводити постійний моніторинг ринку та аналізувати досвід розробки аналогічних програмних продуктів.

Отже, автоматизація касового додатку з використанням технологій C# та WPF є актуальною та важливою задачею для сучасної фінансової сфери, а виконання даного дослідження дозволить створити ефективний та надійний продукт, що відповідає всім потребам ринку та законодавства.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновку та додатків. Робота містить 50 сторінок основного тексту, 23 рисунки, 1 таблицю та 23 літературних джерела.

# РОЗДІЛ 1.
# ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

## 1.1 Аналіз публікацій щодо існуючих касових додатків.

Основна мета розробки касового додатку з використанням технологій C# та WPF полягає в створенні функціонального програмного забезпечення для автоматизації роботи каси в торговій точці або іншому закладі обслуговування. Це додаток, який дозволяє швидко і точно проводити операції з продажу товарів та послуг, приймати оплату і зберігати інформацію про транзакції.

Крім того, такий додаток повинен мати зручний інтерфейс для користувача, забезпечувати надійну захист інформації про касирів і операції. Отже, мета полягає в створенні потужного та зручного інструменту для підтримки ефективної та точної роботи касового персоналу та управління бізнесом.

Дослідження предметної області пов'язано з розробкою касових додатків, що використовують технології C# та WPF. Предметна область охоплює автоматизацію роботи касового персоналу в торгових точках та інших закладах обслуговування.

Дослідження повинно включати наступні завдання:

1. Аналіз існуючих касових додатків, що використовують технології C# та WPF. Вивчення їх функціональності, інтерфейсу та можливостей.
2. Вивчення стандартів та законодавчих вимог щодо роботи з касовими апаратами та збереження даних про транзакції.
3. Визначення основних функцій та можливостей, які повинен мати розроблюваний касовий додаток.
4. Вибір оптимальних технологій та архітектури для розробки додатку.

5. Розробка дизайну інтерфейсу користувача та його тестування з різними аудиторіями користувачів.

6. Розробка бази даних для збереження інформації про транзакції та користувачів.

7. Розробка програмного забезпечення для забезпечення безпеки даних, включаючи шифрування та захист від несанкціонованого доступу.

8. Тестування розробленого додатку та аналіз отриманих результатів.

9. Розробка документації для додатку, яка включає опис функціональності, інструкцію користувача та технічну документацію.

10. Розгортання додатку на платформі, що відповідає вимогам його роботи, та надання підтримки користувачам.

У процесі дослідження слід звернути увагу на забезпечення точності та ефективності роботи касового додатку, забезпечення захисту даних користувачів та відповідність законодавчим вимогам щодо обробки фінансових даних.

Також, важливим завданням є розробка додатку з урахуванням потреб користувачів та забезпечення зручності його використання. Необхідно врахувати, що касові додатки використовуються в різних типах закладів, тому інтерфейс додатку повинен бути простим та зрозумілим для різних категорій користувачів.

Окрім того, важливим завданням є забезпечення масштабованості додатку та можливості підтримки різних касових апаратів та операційних систем.

Під час розробки касового додатку необхідно враховувати технічні обмеження та можливості платформи, на якій він буде працювати, а також забезпечувати його сумісність з іншими програмними продуктами, що використовуються в закладі.

Загалом, розробка касового додатку з використанням технологій C# та WPF є складним і відповідальним завданням, що вимагає знань з різних областей, таких як програмування, фінансові технології, дизайн інтерфейсу користувача та безпека даних. Однак, виконання цього завдання може допомогти

покращити ефективність роботи касового персоналу та забезпечити точність обліку фінансових операцій.

Існує велика кількість касових додатків, які використовуються в різних типах закладів, від кафе та ресторанів до магазинів та супермаркетів. Деякі з найпопулярніших касових додатків на сьогоднішній день включають такі:

1. Poster POS - це веб-платформа для управління бізнесом, яка включає касовий додаток, створений на базі технологій React та Node.js. Poster POS дозволяє здійснювати операції з продажу, обліку запасів та управління клієнтською базою [1].

2. Cashier Live - це онлайн-касовий додаток, який можна використовувати на комп'ютерах, планшетах та мобільних пристроях. Cashier Live дозволяє здійснювати операції з продажу, керування запасами та управління клієнтською базою, а також надає звіти та аналітику щодо продажів [2].

3. Square POS - це касовий додаток, який можна використовувати на планшетах та мобільних пристроях. Square POS дозволяє здійснювати операції з продажу, приймати платежі та керувати запасами, а також надає звіти та аналітику щодо продажів [3].

4. Loyverse POS - це безкоштовний касовий додаток для малого та середнього бізнесу, який можна використовувати на планшетах та мобільних пристроях. Loyverse POS дозволяє здійснювати операції з продажу, керувати запасами та управляти клієнтською базою, а також надає звіти та аналітику щодо продажів [4].

5. Shopify POS - це касовий додаток, який можна використовувати на планшетах та мобільних пристроях. Shopify POS дозволяє здійнювати операції з продажу, приймати платежі та керувати запасами, а також надає звіти та аналітику щодо продажів. Крім того, він інтегрується з іншими продуктами Shopify, такими як онлайн-магазин та CRM-система [5].

При виборі касового додатку необхідно враховувати ряд факторів, таких як функціональні можливості, ціна, зручність використання та можливості

інтеграції з іншими системами. Також слід звернути увагу на підтримку та безпеку додатка.

Тема розробки касового додатку з використанням технологій C# та WPF є дуже актуальною, особливо в контексті зростання електронної комерції та розвитку ринку роздрібної торгівлі. На сьогоднішній день касові додатки стали невід'ємною частиною бізнесу, адже вони дозволяють ефективно керувати продажами, контролювати запаси та виконувати фінансові операції.

Зокрема, з поширенням електронних платежів та безготівкових розрахунків касові додатки стають ще більш важливими. Також значна частина бізнесів переходять на онлайн-платформи, що призводить до потреби відповідних касових рішень для їх підтримки.

Крім того, у зв'язку зі зростанням кількості бізнесів, що ведуть діяльність в різних галузях, потреба в належних касових додатках стає дедалі більшою. Використання технологій C# та WPF дозволяє створювати потужні та функціональні додатки зі зручним інтерфейсом, що може допомогти бізнесу бути більш ефективним та конкурентним.

## 1.2 Аналіз методів розробки програми

Аналіз методів вирішення задачі демонструє, що використані технології та інструменти були дуже ефективними в процесі розробки касового додатку.

Дана задача виконується такими методами і технологіями:

C# - це мова програмування, яка забезпечує високу продуктивність та ефективність. Використання C# дозволило розробникам створити потужний та функціональний додаток, який може бути легко розширений та налагоджений.

WPF - це технологія розробки додатків для Windows, яка дозволяє створювати привабливий та інтуїтивно зрозумілий інтерфейс користувача.

Використання WPF дозволило розробникам створити привабливий та зручний для використання інтерфейс користувача.

SQL Server Management - це програмне забезпечення для керування базами даних Microsoft SQL Server. Використання SQL Server Management дозволило розробникам ефективно керувати базою даних та забезпечити високу продуктивність та швидкість доступу до даних.

Material Desing in XAML - це фреймворк, який дозволяє створювати інтерфейс користувача згідно з дизайном Material Design від Google. Використання Material Design in XAML дозволило розробникам створити привабливий та зручний для використання інтерфейс користувача згідно з останніми тенденціями дизайну [20].

BCrypt - це бібліотека, яка забезпечує хешування паролів користувачів. Використання BCrypt дозволило забезпечити високу захищеність даних та паролів користувачів.

ZXing.NET - це бібліотека для створення та розпізнавання QR-кодів. Використання ZXing.NET дозволило розробникам забезпечити можливість швидкого та зручного сканування QR-кодів для здійснення оплат.

AForge.NET - це бібліотека для комп'ютерного зору та обробки зображень. Використання AForge.NET дозволило розробникам забезпечити можливість автоматичного розпізнавання товарів за допомогою камери, що дозволяє спростити процес продажу та зменшити кількість помилок.

ADO.NET - це технологія для роботи з базами даних в середовищі .NET. Використання ADO.NET дозволило розробникам легко та ефективно забезпечити доступ до бази даних та здійснювати операції збереження, оновлення та видалення даних.

EF.Core - це фреймворк для роботи з базами даних в середовищі .NET. Використання EF.Core дозволило розробникам легко та ефективно забезпечити доступ до бази даних та здійснювати операції збереження, оновлення та видалення даних.

Аналіз використаних технологій та інструментів показує, що ці технології дозволили розробникам створити потужний та ефективний касовий додаток, який може забезпечити швидку та зручну оплату товарів, керування даними та забезпечення захищеності користувачів.

На сьогоднішній день є багато програм які вирішують дану задачу. Всі вони мають свої гідності й недоліки.

Наприклад, якщо взяти дану програму [6] (рисунок 1.1), то можна побачити, що вона має весь основний функціонал, такий як, підрахування суми чеку, додавання товару до чеку, формування чеку, змінення кількості товару. Але є наступні недоліки: ця програма не може зчитувати штрих-код товару, потрібно власноруч вводити всі 13 цифр. Також, в даній програмі, важко ідентифікувати товар, для цього потрібно порівнювати всі 13 цифр штрих-коду. Ще, основним недоліком, є те, що вона не має гарного користувальницького інтерфейсу.

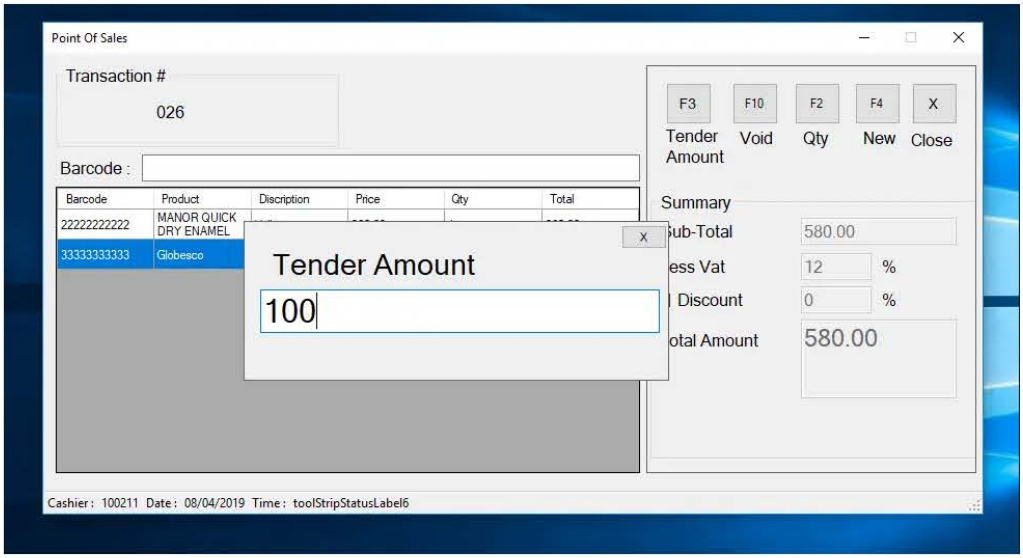


Рисунок 1.1 – Приклад програми «Point of Sales»

Якщо взяти інший приклад [7], який наведений на рисунку 1.2, то в даній програмі є такі ж самі недоліки, як й в попередній, а саме: програма не містить функціоналу зчитування штрих-коду, важко ідентифікувати товар, а ще, вона містить дуже багато полей, кнопок, які відволікають від роботи касира, та не має інтуїтивно-зрозумілого користувальницького інтерфейсу.

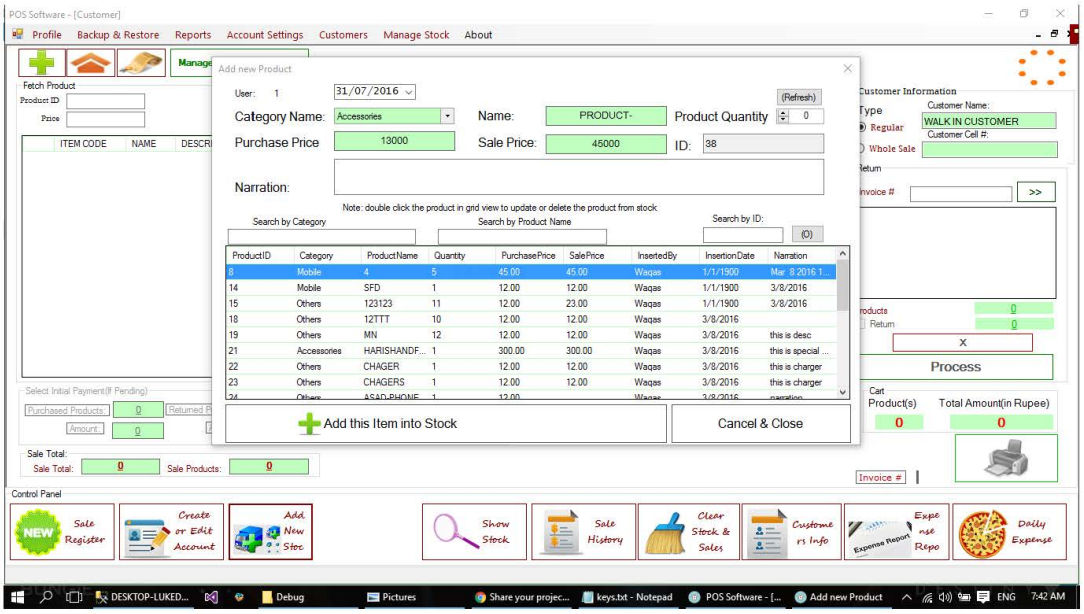


Рисунок 1.2 – Приклад програми «POS Software»

Отже можна зробити висновок, що дані програми мають свої недоліки, які потрібно вирішити. Саме тому дана кваліфікаційна робота присвячена розробці такої програми з урахуванням попередніх робіт і усуненням таких недоліків, як: безпека, ідентифікація товарів, зручний та інтуїтивно-зрозумілий інтерфейс, зчитування штрих-коду.

Основна проблема в касових додатках які були вказані вище це ідентифікація товарів. Покупець може наклеїти штрих-код дешевого товару на дорогий товар, що призведе бізнес до втрати прибутку. Для вирішення даної задачі, було зроблено наступне: виведення зображення товару на екрані касира. Таким чином, касир може порівняти товар який він пробив, та той який пробився. Дане вирішення задачі знижує фактор такого обману до нуля.

Також проблемою є те, що існуючі касові додатки мають не зрозумілий користувальницький інтерфейс, який включає багато елементів, та з якими важко розібратися. Для вирішення цієї проблеми, було обрана саме технологія WPF та Material Desing in XAML, які допомогли вирішити цю проблему.

Ще одна з проблем, це безпека. В указаних вище програмах, немає шифрування даних користувача. Тобто вказаний користувачем пароль

зберігається у явному вигляді. Тому, в даній дипломній роботі, було обрано таку бібліотеку, як BCrypt, яка забезпечує хешування паролів користувачів. Використання цієї бібліотеки, дозволило забезпечити високу захищеність паролів користувачів, що в даний час, є невідмінною частиною кожного додатку.

## 1.3 Висновки до першого розділу. Постановка завдань дослідження

У цьому розділі було проаналізовано та розглянуто тему розробки касового додатку з використанням технологій C# та WPF. Було проведено дослідження предметної області, а також проаналізовано існуючі касові додатки.

Для розв'язання поставленої задачі було використано такі технології та інструменти, як C#, WPF, SQL Server Management, Material Desing in XAML, BCrypt, ZXing.NET, AForge.NET, ADO.NET, EF.Core. Кожна з цих технологій допомогла створити потужний та ефективний касовий додаток, який забезпечує швидку та зручну взаємодію, керування даними та забезпечення захищеності користувачів.

На основі проведених в першому розділі досліджень можна зробити висновок що дана тема дипломної роботи є актуальною в сучасному світі, де робота з касовими апаратами та програмним забезпеченням є необхідною для багатьох підприємств. Касовий додаток, розроблений за допомогою C# та WPF, дозволяє простіше та ефективніше керувати продажами та забезпечує захист даних користувачів.

Постановка завдань дослідження полягала в дослідженні предметної області, аналізі існуючих касових додатків, розробці ефективного та зручного касового додатку з використанням технологій C# та WPF, що забезпечує швидку та зручну оплату товарів, керування даними та захищеність даних користувачів.

# РОЗДІЛ 2.
# АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ КАСОВОГО ДОДАТКУ

## 2.1 Вибір програмних засобів для реалізації проекту

Аналіз засобів реалізації касового додатку дозволяє визначити, які технології та інструменти найбільш ефективно можна використати для розробки потужного та зручного додатку.

Одним з головних критеріїв вибору засобів реалізації є мова програмування. Для розробки касового додатку можна використати різні мови програмування, такі як C#, Java, Python, і т.д. У даній роботі ми використали мову програмування C#, оскільки це мова, яка дозволяє ефективно розробляти додатки для Windows з використанням технології .NET Framework.

Окрім мови програмування, для розробки касового додатку необхідно використовувати спеціальні інструменти, такі як Integrated Development Environment (IDE), системи керування базами даних, бібліотеки, та інші. У данній роботі ми використовували Visual Studio 2019 як Integrated Development Environment, SQL Server Management як систему керування базами даних, Material Desing in XAML та AForge.NET для реалізації візуальної частини додатку.

Для забезпечення безпеки даних користувачів в касовому додатку було використано бібліотеку BCrypt для шифрування паролів, а також бібліотеку ZXing.NET для генерації та сканування штрих-кодів.

Крім того, для забезпечення роботи з базою даних, було використано технології ADO.NET та EF.Core.

Таким чином, під час аналізу засобів реалізації касового додатку, було використано низку різноманітних технологій та інструментів, що дозволило розробити потужний та зручний додаток з використанням сучасних технологій та інструментів.

Вибір правильних програмних засобів є важливим етапом в розробці будь-якого програмного продукту, в тому числі й касового додатку. У цьому розділі будуть розглянуті основні критерії вибору програмних засобів для реалізації проекту.

Одним з основних критеріїв вибору програмних засобів є підтримка мови програмування та технологій, які будуть використовуватися при розробці додатку. Для розробки касового додатку буде використана мова програмування C# та технологія WPF (Windows Presentation Foundation). Отже, програмний засіб повинен підтримувати ці технології.

Також важливим критерієм є можливість інтеграції з різними базами даних, адже касовий додаток повинен зберігати велику кількість даних про продажі та операції. В даному проекті буде використовуватися реляційна база даних Microsoft SQL Server, тому програмний засіб повинен підтримувати роботу з цією базою даних.

Для реалізації графічного інтерфейсу користувача касового додатку буде використовуватися технологія WPF. Це дозволяє створити красивий та зручний інтерфейс з використанням візуальних елементів, таких як кнопки, тексти, картинки тощо. Програмний засіб повинен мати потужні можливості для розробки графічного інтерфейсу на основі технології WPF.

Для забезпечення безпеки та захисту від шахрайства в касовому додатку необхідно забезпечити захист даних користувачів. Програмний засіб повинен мати можливість реалізувати аутентифікацію та авторизацію користувачів, а також захист від недобросовісних програмних засобів та вразливостей.

Також, важливим критерієм є можливість розширення та підтримки додаткових функцій та операцій, що можуть знадобитися у майбутньому. Для касового додатку можуть знадобитися функції, такі як робота з фіскальними реєстраторами, підтримка різних типів оплат, збереження та обробка даних про знижки та промо-акції тощо. Програмний засіб повинен мати гнучку та розширювану архітектуру, що дозволяє додавати нові функції та модулі без значних змін в коді.

Ще одним важливим критерієм є вартість та ліцензування програмного засобу. У залежності від розміру бізнесу, може бути необхідно вибрати програмний засіб з відповідною ліцензією, яка забезпечує потрібний функціонал та рівень підтримки, але не перевищує бюджету проекту.

На основі аналізу вищезгаданих критеріїв було вирішено використати таку інтегровану середу розробки, як Visual Studio який буде основним програмним засобом для розробки касового додатку. Visual Studio підтримує мову програмування C# та технологію WPF, має гнучку архітектуру, дозволяє працювати з базами даних, забезпечує можливість розширення функціоналу та підтримує різні рівні ліцензування, що робить його ідеальним вибором для даного проекту.

## 2.2 Засоби для забезпечення безпеки

Для забезпечення безпеки даних користувача було реалізована форма для авторизації користувача, яка використовує надійний алгоритм шифрування паролю.

Безпосереднє порівняння простих текстових паролів небезпечно. Замість цього слід хешувати пароль і порівнювати хеші. Для цієї мети можна використати бібліотеку на зразок BCrypt або Argon 2.

У роботі була обрана бібліотека BCrypt [8] тому, що вона використовує варіант розкладу введення ключів алгоритму шифрування Blowfish і вводить коєфіцієнт роботи, який дозволяє визначити, наскільки дорогою буде хеш-функція. Саме завдяки цьому, BCrypt може не відставати від закону Мура. У міру того, як комп'ютери стають швидкими, можна збільшити коефіцієнт роботи, і хеш стане повільнішим.

BCrypt – це ключова функція виведення паролів, розроблена Нільсом Провосом і Девідом Мазьєром, заснований на шифрі Blowfish і представлений

на USENIX. Окрім використання солі для захисту від атак веселкової таблиці, BCrypt є адаптивною функцією: з часом кількість ітерацій можна збільшити, що зробить її повільнішою, тож вона залишається стійкою до атак методом "Brute force" навіть із збільшенням обчислювальної потужності [24].

BCrypt.Net – це бібліотека з відкритим кодом, яка забезпечує простий і легкий у використанні інтерфейс для роботи з BCrypt. Його можна встановити у будь-який проект .NET за допомогою NuGet.

Бібліотека надає два основні методи: BCrypt.Net.BCrypt.HashPassword() і BCrypt.Net.BCrypt.Verify(). Метод HashPassword() приймає пароль і робочий фактор і повертає хешований рядок, який можна безпечно зберігати в базі даних. Коефіцієнт роботи визначає кількість ітерацій хеш-функції, що впливає на безпеку хешування. Вищий коефіцієнт роботи означає сильніший хеш, але також повільніше хешування.

Метод Verify() приймає пароль і хешований рядок і повертає true, якщо пароль збігається з хешем, і false в іншому випадку. Цей метод автоматично витягує сіль із хешу та використовує її для повторного хешування пароля, тому немає необхідності зберігати сіль окремо.

BCrypt.Net також підтримує інші функції, такі як спеціальні солі, і може використовуватися з різними платформами .NET, включаючи .NET Core, .NET Framework і Xamarin.

Загалом, BCrypt.Net є чудовим вибором для захисту паролів у програмах .NET завдяки надійній безпеці та простоті використання.

На рисунку 2.1 наведено приклад з програми, який реалізує валідацію паролю:

```
if (!PasswordHelper.ValidatePassword(password, objUser.Password))
{
    ShowErrorMessage("Логін або пароль недійсні!");
    ClearFields();
    return;
}
```

Рисунок 2.1 – Валідація паролю

Реалізацію класу PasswordHelper, яка використовує бібліотеку BCrypt для хешування та валідації паролю наведено на рисунку 2.2.

```
public class PasswordHelper
{
    private const int WorkFactor = 12;

    0 references
    public static string HashPassword(string password)
    {
        string salt = BCrypt.Net.BCrypt.GenerateSalt(WorkFactor);
        return BCrypt.Net.BCrypt.HashPassword(password, salt);
    }

    1 reference
    public static bool ValidatePassword(string password, string hash)
    {
        return BCrypt.Net.BCrypt.Verify(password, hash);
    }
}
```

Рисунок 2.2 – Реалізація класу PasswordHelper

Метод HashPassword генерує сіль і використовує її для хешування заданого пароля за допомогою алгоритму BCrypt. Параметр WorkFactor визначає кількість ітерацій хеш-функції, що впливає на безпеку хешу. Вищий коефіцієнт роботи означає сильніший хеш, але також повільніше хешування.

Метод ValidatePassword приймає простий текстовий пароль і хешований пароль і використовує BCrypt, щоб перевірити, чи відповідає пароль хешу. Якщо паролі збігаються, метод повертає true, інакше повертає false.

## 2.3 Засоби для зчитування штрих-кодів

Штрих-код- це послідовність чорних і білих смуг, що представляє деяку інформацію у вигляді, зручному для зчитування технічними засобами [9].

Для ідентифікації товарів та бонусних карток користувачів, у роботі, був використаний штрих-код EAN-13.

EAN-13 є стандартом міжнародної ідентифікації товарів, який використовується для кодування числової інформації у вигляді штрих-коду на продуктах.

Кожен EAN-13 штрих-код складається з 13 цифр, які можна розділити на три частини: код країни (з трьох цифр), код виробника (з чотирьох або більше цифр, залежно від кількості товарів, що випускається), та код продукту (з п'яти цифр).

Кожна цифра коду представлена у вигляді чорних та білих смуг, які мають різну ширину та розташування. За допомогою спеціального сканера штрих-коду, який може працювати як зі світлодіодним джерелом, так і з лазером, можна декодувати штрих-код та отримати інформацію про продукт, який він ідентифікує.

EAN-13 є одним з найбільш поширених стандартів штрих-кодів, який використовується у багатьох країнах світу для ідентифікації продуктів. Він дозволяє швидко та ефективно відстежувати товари в різних етапах їхнього життєвого циклу, від виробництва до продажу, та забезпечує точність та надійність інформації про продукти.

На рисунку 2.3 можна побачити структуру штрих-коду.


Рисунок 2.3 – Структура штрих-коду

Для програмної реалізації зчитування штрих-коду було використана бібліотека ZXing.NET.

ZXing.NET - це відкрита бібліотека з вільним кодом [10, 11, 12], яка надає засоби для розпізнавання та генерації різних типів штрих-кодів, таких як EAN, UPC, QR-коди та інших. Вона базується на Java-бібліотеці ZXing, але була переписана на мову C# та підтримується .NET Framework та .NET Core.

Бібліотека ZXing.NET дозволяє додавати можливості розпізнавання та генерації штрих-кодів до додатків .NET, що дозволяє створювати різноманітні програмні продукти, що використовують штрих-коди для ідентифікації та відстеження товарів, квитків, етикеток та інших об'єктів.

Крім того, ZXing.NET підтримує розпізнавання штрих-кодів з різних джерел, включаючи зображення, веб-камери та інші пристрої, що дозволяє забезпечувати високу точність розпізнавання для різних умов та форматів.

ZXing.NET є досить популярною бібліотекою, що використовується у багатьох різноманітних додатках, таких як мобільні додатки, системи контролю складу, системи автоматизації та багато інших.

## 2.4 Засоби для обробки зображень

AForge.NET - це відкрита бібліотека з вільним кодом [13], яка надає засоби для розробки програм, що використовують обробку зображень, комп'ютерний зір та нейронні мережі. Вона базується на мові програмування C# та підтримується .NET Framework та .NET Core.

AForge.NET надає широкий набір інструментів для обробки зображень, таких як фільтри, розпізнавання образів, сегментація зображень, визначення країв та інші. Бібліотека також підтримує комп'ютерний зір, що дозволяє

розпізнавати обличчя, визначати кольори, відстежувати рух та інші завдання, пов'язані з обробкою відео.

Крім того, AForge.NET містить модуль для навчання нейронних мереж, що дозволяє створювати та тренувати моделі для класифікації образів та різних завдань машинного навчання.

AForge.NET використовується у багатьох проектах, пов'язаних з обробкою зображень та комп'ютерним зіром, таких як системи відстеження руху, системи контролю якості, системи безпеки та інші. Бібліотека дозволяє розробникам швидко та ефективно створювати програми зі складними завданнями обробки зображень та комп'ютерного зіру.

У даному проекті, ця бібліотека була використана для того, щоб обробляти зображення, яке надходить через веб-камеру, та далі розпізнавати штрих-код.

## 2.5 Засоби для роботи з базою даних

SQL Server Management Studio (SSMS) - це інтегроване середовище розробки та управління базами даних Microsoft SQL Server [14, 15]. Воно дозволяє розробникам, адміністраторам баз даних та користувачам SQL Server управляти та керувати базами даних, зберігати процедури, функції та тригери, виконувати запити та багато іншого.

SSMS дозволяє створювати, редагувати та видаляти бази даних, таблиці, збережені процедури, функції та індекси, керувати безпекою баз даних, налаштувати параметри сервера та виконувати резервне копіювання та відновлення даних. Він також надає засоби для управління реплікацією даних, відстеження запитів та профілювання даних.

Крім того, SSMS містить SQL Server Object Explorer, який дає змогу переглядати та керувати об'єктами баз даних, такими як таблиці, стовпці, індекси

та інші, та Query Editor, який дозволяє виконувати запити до баз даних та отримувати результати в зручному для розуміння форматі.

SQL Server Management Studio є потужним та необхідним інструментом для розробки та управління базами даних Microsoft SQL Server. Він допомагає спростити роботу з базами даних, підвищує продуктивність та ефективність роботи з ними, та забезпечує високий рівень безпеки даних.

SQL Server Management Studio (SSMS) має декілька переваг, серед яких:

1. Зручний інтерфейс: Інтерфейс SSMS є досить зручним та інтуїтивно зрозумілим для розробників, що дозволяє швидко зорієнтуватися в інструментах та об'єктах бази даних.

2. Розширені можливості управління: SSMS надає багато інструментів для управління базою даних, такі як керування безпекою, налаштування параметрів сервера, виконання резервного копіювання та відновлення даних.

3. Керування об'єктами бази даних: SSMS дозволяє розробникам керувати об'єктами бази даних, такими як таблиці, збережені процедури, функції та індекси.

4. Робота з запитами: Інструмент Query Editor в SSMS дозволяє виконувати запити до бази даних та отримувати результати в зручному для розуміння форматі.

5. Висока безпека даних: SSMS забезпечує високий рівень безпеки даних, дозволяючи налаштовувати параметри безпеки, такі як автентифікація користувачів та керування ролями.

6. Підтримка багатьох версій SQL Server: SSMS підтримує багато версій Microsoft SQL Server, що дозволяє працювати з базами даних різних версій.

У загальному, використання SQL Server Management Studio дозволяє розробникам та адміністраторам баз даних ефективно керувати та розробляти бази даних, забезпечувати їх безпеку та продуктивність. Саме тому для виконання роботи було обрано це середовище.

## 2.6 Засоби для роботи з даними

Для роботи з базами даних у C# та .NET можна застосовувати технологію ADO.NET. Для спрощення роботи з базами даних також можна застосовувати Entity Framework – технологія ORM, яка дозволяє зіставляти сутності C# з таблицями у БД.

У дипломній роботі, розробленій з використанням технологій C# та WPF, ADO.NET використовувався для створення з'єднання з базою даних, виконання запитів та отримання даних. EF.Core використовувався для створення моделі даних та роботи з об'єктами в програмі. Обидва засоби дозволили реалізувати функціонал зберігання та взаємодії з даними в касовому додатку.

### 2.6.1 Технологія ADO.NET

ADO.NET є технологією роботи з даними, яка заснована на платформі .NET Framework [16]. Ця технологія представляє нам набір класів, через які ми можемо надсилати запити до баз даних, встановлювати підключення, отримувати відповідь від бази даних та здійснювати низку інших операцій.

Основу інтерфейсу взаємодії з базами даних у ADO.NET представляє обмежене коло об'єктів: Connection, Command, DataReader, DataSet та DataAdapter. За допомогою об'єкта Connection відбувається встановлення підключення до джерела даних. Об'єкт Command дозволяє виконувати операції з даними БД. Об'єкт DataReader зчитує отримані в результаті запиту дані. Об'єкт DataSet призначений для зберігання даних із БД та дозволяє працювати з ними незалежно від БД. І об'єкт DataAdapter є посередником між DataSet та джерелом даних. Головним чином, через ці об'єкти і буде працювати з базою даних.

Схематично архітектуру ADO.NET можна представити наступним чином:

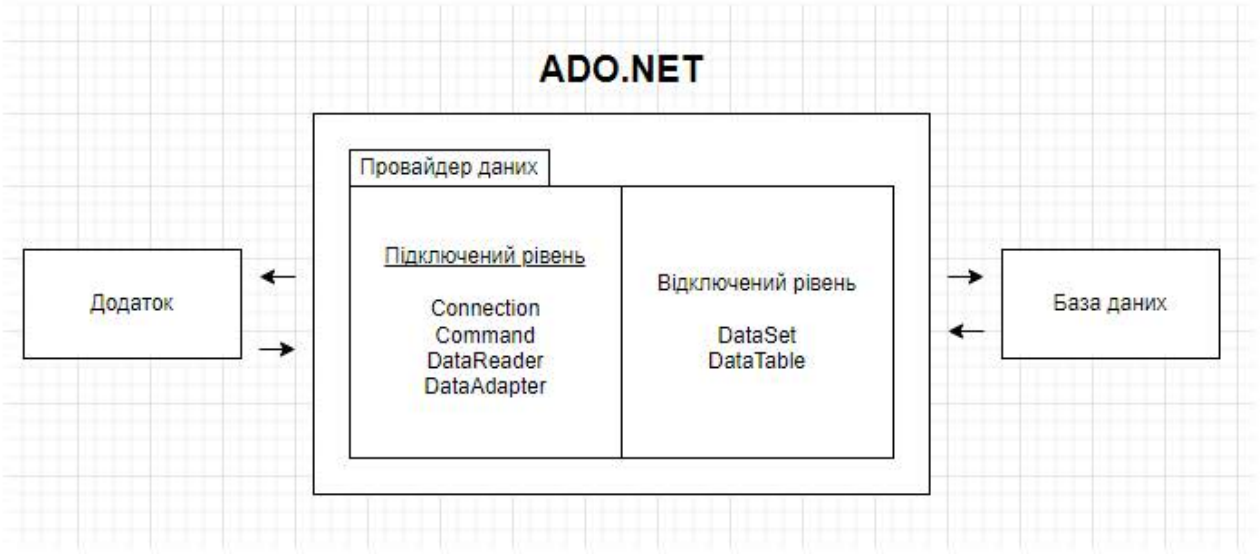


Рисунок 2.4 – Архітектура ADO.NET

Функціонально класи ADO.NET можна розбити на два рівні: підключений та відключений. Кожен провайдер даних .NET реалізує свої версії об'єктів Connection, Command, DataReader, DataAdapter та інших, які становлять підключений рівень. Тобто за допомогою них встановлюється підключення до БД та виконується з нею взаємодія. Як правило, реалізації цих об'єктів для кожного конкретного провайдера у своїй назві мають префікс, який вказує на провайдера.

### 2.6.2 Технологія Entity Framework Core

Entity Framework Core – це проста, кросплатформова версія популярної технології доступа до даних Entity Framework з відкритим вихідним кодом [17].

Entity Framework Core (EF Core) є об'єктно-реляційною (ORM) бібліотекою для .NET, яка дозволяє розробникам працювати з базами даних, використовуючи об'єктно-орієнтований підхід. EF Core дозволяє зменшити кількість коду,

необхідного для взаємодії з базою даних, і забезпечує більш простий та швидкий розробку програм.

Деякі переваги Entity Framework Core:

1. Зменшення кількості коду: EF Core дозволяє розробникам взаємодіяти з базою даних, використовуючи об'єкти .NET, замість написання SQL запитів вручну. Це зменшує кількість коду, необхідного для розробки, і дозволяє зосередитися на бізнес-логіці програми.

2. Кросплатформеність: EF Core підтримує кросплатформенність, що дозволяє використовувати його на різних операційних системах, таких як Windows, Linux та macOS.

3. Підтримка багатьох провайдерів баз даних: EF Core підтримує багато провайдерів баз даних, включаючи Microsoft SQL Server, MySQL, PostgreSQL, SQLite та багато інших. Це дозволяє розробникам використовувати EF Core з будь-якою базою даних, що підтримується.

4. Простота: EF Core дозволяє легко взаємодіяти з базою даних, не потребуючи великої кількості коду. Наприклад, з EF Core дуже просто створити запити до бази даних, зберігати та видаляти дані.

5. Підтримка лінивої загрузки: EF Core підтримує ліниву загрузку, що дозволяє завантажувати дані з бази даних лише в той момент, коли вони потрібні, що зменшує навантаження на базу даних та підвищує продуктивність.

У EF Core доступ до даних здійснюється за допомогою моделі. Модель складається з класів сутностей та об’єкта контексту, що представляє сеанс взаємодії з базою даних. Об’єкт контексту дозволяє виконувати запити та зберігати дані. EF Core підтримує наступні підходи до розробки моделей [18]:

- створення моделі на основі існуючої БД;
- написання коду моделі вручну відповідно до БД;
- після створення моделі використовуйте міграції EF для створення БД на основі моделі. Міграції дозволяють розвивати БД у міру зміни моделі.

У ході виконання завдання, був використан підхід «створення моделі на основі існуючої БД».

## 2.7 Аналіз методів для розробки дизайну програми

У роботі були проведені дослідження методів та бібліотек для розробки дизайну програмного забезпечення. Однією з найпопулярніших бібліотек для розробки дизайну є Material Design in XAML.

Material Design in XAML є бібліотекою, що дозволяє розробникам створювати програми зі стильним та сучасним інтерфейсом користувача. Бібліотека базується на дизайні Google Material Design, який став популярним у світі програмного забезпечення за останні роки.

Однією з переваг Material Design in XAML є те, що вона пропонує багато стандартних елементів інтерфейсу користувача, таких як кнопки, текстові поля, таблиці та інші, які можуть бути використані для створення програмного забезпечення зі стильним та сучасним дизайном. Крім того, бібліотека пропонує можливість налаштування кольорів та стилів елементів інтерфейсу, що дозволяє розробникам змінювати зовнішній вигляд своїх програм залежно від їх потреб.

Однією з відмінних рис Material Design in XAML є підтримка відомих платформ, таких як Windows, iOS та Android, що дозволяє розробникам створювати кросплатформові програми з однаковим дизайном для різних пристроїв.

Крім того, Material Design in XAML має активну спільноту розробників, що забезпечує підтримку та оновлення бібліотеки. Це означає, що розробники можуть бути впевнені у тому, що їх програми будуть сумісні з майбутніми оновленнями операційної системи та інших залежностей, а також що бібліотека буде підтримувати нові функції та можливості в майбутньому.

Однією з недоліків Material Design in XAML є те, що для повноцінного використання бібліотеки потрібно мати певні знання з програмування та розробки інтерфейсу користувача. Також, для коректної роботи бібліотеки потрібно використовувати середовище розробки Visual Studio, що може бути обмеженням для деяких розробників.

Узагальнюючи, Material Design in XAML є потужною бібліотекою для розробки дизайну програмного забезпечення, що дозволяє створювати програми зі стильним та сучасним інтерфейсом користувача. Бібліотека має багато переваг, включаючи підтримку різних платформ, налаштування кольорів та стилів елементів інтерфейсу, а також активну спільноту розробників. Недоліком є потреба у певних знаннях з програмування та використання середовища розробки Visual Studio.

Material Design in XAML (MDX) - це бібліотека для платформи .NET, яка надає розробникам можливість створювати красиві та сучасні інтерфейси користувача, дотримуючись дизайну Material Design від Google [19, 20]. Material Design - це концепція дизайну, яка поєднує простоту та функціональність, з фокусом на простих, візуально зрозумілих елементах управління.

Деякі переваги Material Design in XAML:

1. Красивий та сучасний дизайн: MDX дозволяє створювати інтерфейси користувача згідно з дизайном Material Design, який має простий та сучасний вигляд.

2. Простота використання: MDX дуже простий у використанні, що дозволяє розробникам легко додавати елементи дизайну Material Design до своїх проектів.

3. Готові компоненти: MDX містить багато готових компонентів, які можна використовувати безпосередньо у своїх проектах, що зменшує кількість коду, необхідного для розробки.

4. Підтримка анімації: MDX підтримує анімацію елементів дизайну Material Design, що дозволяє розробникам створювати більш динамічні та привабливі інтерфейси користувача.

5. Кросплатформеність: MDX підтримує платформи .NET Framework та .NET Core, що дозволяє використовувати його на різних операційних системах, таких як Windows, Linux та macOS.

6. Розширені можливості: MDX дозволяє розробникам розширювати функціональність бібліотеки та створювати власні компоненти, що дозволяє реалізувати власні потреби та ідеї дизайну.

На рисунку 2.3 можна побачити готові текстові поля, які можуть бути використані у програмі:

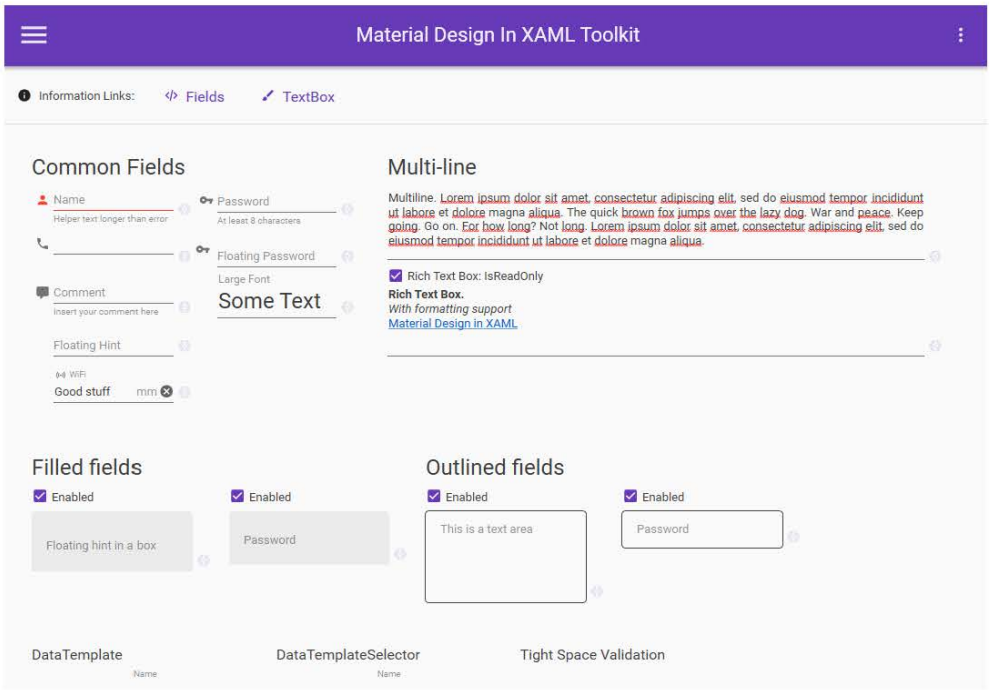


Рисунок 2.5 – Приклад готових текстових полів

Отже, для створення красивого, сучасного та інтуїтивно-зрозумілого інтерфейсу, у роботі було використано саме цю бібліотеку.

## 2.8 Висновки до другого розділу

У другому розділі було проаналізовано технології для вирішення завдання. Було з'ясовано які технології є, та які є найкращими для вирішення завдання.

Будуть використовуватися такі технології як EAN-13 штрих-код, бібліотека ZXing.NET, AForge.Net, SQL Server Management Studio, Entity Framework Core та Material Design in XAML.

За допомогою EAN-13 штрих-коду можна ідентифікувати товари в магазинах. Бібліотека ZXing.NET дозволяє розпізнавати штрих-коди на товарах та інтегрувати цю функціональність в додатки. AForge.Net надає набір інструментів для обробки зображень та розпізнавання об'єктів на них.

SQL Server Management Studio є потужним інструментом для розробки та управління базами даних на платформі Microsoft SQL Server. Завдяки його функціональності можна виконувати запити до баз даних, встановлювати зв'язки між таблицями та слідити за їх роботою.

Entity Framework Core дозволяє спрощувати розробку баз даних та зменшувати кількість написаного коду. Він надає інтерфейс для взаємодії з базою даних та генерує SQL-запити автоматично, що дозволяє зосередитися на розробці функціональності додатку.

Material Design in XAML дозволяє ефективно використовувати готові компоненти та шаблони для створення сучасного та красивого користувальницького ынтерфейсу. Вона підтримує анімацію та має розширені можливості, що дозволяє створювати більш динамічні та інтерактивні дизайни.

Використання бібліотек та інструментів, таких як Entity Framework Core та Material Design in XAML, дозволяє розробникам ефективніше вирішувати складні задачі та прискорює процес розробки.

Крім того, знання баз даних та інформаційних технологій є важливим для успішної кар'єри в індустрії програмування. Отримання навичок в цих областях дозволяє стати конкурентоздатним на ринку праці та розширити свій потенціал як розробник.

Загалом, розглянуті питання є важливими для розуміння технологій, що використовуються в програмуванні та розробці програмного забезпечення. Розуміння цих технологій та їх застосування дозволяє розробникам бути більш продуктивними та ефективними в своїй роботі.

# РОЗДІЛ 3.
# РОЗРОБКА КАСОВОГО ДОДАТКУ

## 3.1 Проектування структури програми

Проектування структури програми – це процес розробки плану, який визначає організацію компонентів програмного коду та способи їх взаємодії для досягнення певної мети. Це може включати визначення функціональних блоків, їх зв'язків, інтерфейсів користувача та алгоритмів обробки даних.

Проектування структури програми є важливим етапом в розробці програмного забезпечення з кількох причин:

1. Допомагає уникнути проблем у майбутньому: добре спроектована структура програми може запобігти проблемам зі скачуванням, швидкістю відгуку та розширенням функціональності програми в майбутньому.
2. Забезпечує більш ефективний процес розробки: структура програми може допомогти розробникам більш ефективно взаємодіяти між собою, ділити код на менші частини та дозволяє кожному розробнику концентруватися на своїй частині проекту.
3. Забезпечує легкість розуміння та підтримки: якщо структура програми добре спроектована, це допомагає зрозуміти логіку та функціональність програми, що полегшує підтримку програми в майбутньому.
4. Дозволяє визначити пріоритети: структура програми може допомогти визначити, які функції важливіші та потребують більшої уваги в розробці.

Узагалі, проектування структури програми є ключовим етапом розробки програмного забезпечення, оскільки від цього залежить успішність програми, ефективність розробки та її підтримка в майбутньому.

Структура проекту програми має наступний вигляд (наведено на рисунку 3.1):

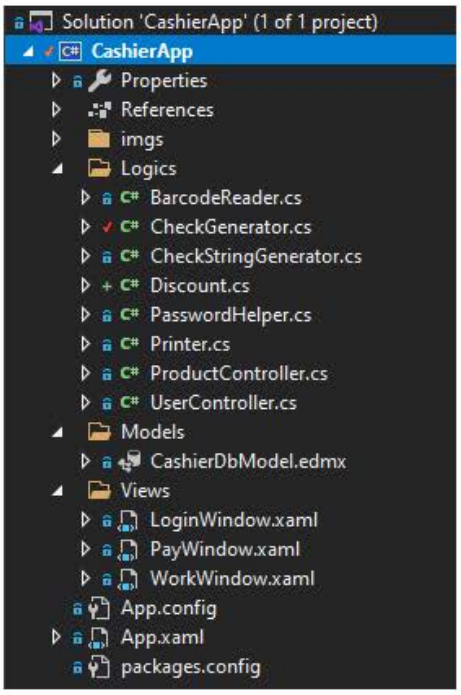


Рисунок 3.1 – Структура проекту програми

У структурі програми присутні такі файли, як Logics, Models, Views та imgs.

У файлі Logics знаходиться уся логіка програми, яка потрібна для її реалізації. Файл Models містить у собі модель бази даних, яка було згенерована автоматично за допомогою Entity Framework Core. Файл Views містить усі візуальні вікна програми, та файл imgs в якому знаходяться картинки, які були використані у програмі.

У таблиці 3.1 можна побачити всі класи програми та їх методи (тільки публічні методи), які є ключовими модулями логіки програми:

Таблиця 3.1

Логічні класи програми та їх опис

| Назва класу | Методи класу | Опис |
|---|---|---|
| UserController | public async Task<User> GetUserAsync(string username) | Використовується для отримання користувача за його username за допомогою асинхронного методу |
| PasswordHelper | public static bool ValidatePassword(string password, string hash) | Робить перевірку введеного паролю та паролю що знаходиться у базі даних за допомогою шифрування Bcrypt |
| Discount | public void CalculateDiscount() | Підраховує витрачені кошти користувача за весь час, та відповідно з цим змінює відсоток знижки користувача |
| BarcodeReader | 1. public void Start(string barcodeDeviceName);<br>2. public void Stop(). | 1. Запускає процес роботи веб-камери, з якої зчитується штрих-код<br>2. Завершує роботу веб-камери |
| ProductController | public async Task<Product> | Використовується для отримання |

| | | |
|---|---|---|
| | GetProductAsync(string barcode) | продукту за його штрих-кодом за допомогою асинхронного методу |
| Printer | public void Print() | Починає процес друкування чеку |
| CheckGenerator | 1. public Check GenerateCheck(Card card)<br>2. public decimal CalculateSum()<br>3. public decimal CalculateSumWithDiscount() | 1. Починає процес генерування чеку<br>2. Підраховує суму всіх товарів<br>3. Підраховує суму чеку згідно з відсотком знижки користувача |
| CheckStringGenerator | public string GenerateCheck() | Формує всі дані, які потрібні у чеці та передає строку до друку |

## 3.2 Проектування візуального інтерфейсу користувача

Проектування візуального інтерфейсу користувача (UI) - це процес створення графічного інтерфейсу, який дозволяє користувачам взаємодіяти з програмою або системою за допомогою графічних елементів, таких як кнопки, меню, поля введення та інші.

UI дизайн важливий для багатьох причин:

1. Поліпшення користувальницького досвіду: добре проектований UI дозволяє користувачам легко зрозуміти, як взаємодіяти з програмою, забезпечує зручність та швидкість використання програми.

2. Збільшення ефективності роботи: зручний та простий у використанні інтерфейс дозволяє користувачам швидко знайти та виконати необхідну функцію, що підвищує їх продуктивність та ефективність роботи.

3. Підвищення конкурентоспроможності: дизайн UI є важливим елементом в розробці програмного забезпечення та систем. При добре проектованому інтерфейсі користувачі більш схильні до використання вашої програми порівняно з конкурентами.

4. Зменшення помилок та запобігання неправильному використанню: якщо UI добре спроектований, користувачі зменшать ризик зробити помилку при взаємодії з програмою, що забезпечує більш точний та надійний результат використання.

Узагалі, проектування візуального інтерфейсу користувача є важливим етапом розробки програмного забезпечення, який дозволяє забезпечити ефективну та зручну взаємодію користувача з програмою, підвищити її конкурентоспроможність та забезпечити більш точний результат використання.

Для виконання завдання було обрано технологію WPF для створення добре проектованого UI, який є зручним та простим у використанні. Також була використана бібліотека Material Design in XAML, що також покращує показники при проектуванні візуального інтерфейсу користувача.

Для зручності розробки візуального інтерфейсу користувача було створено три вікна, які виконують свої функції.

На рисунок 3.2 можна побачити вікно авторизації касира:

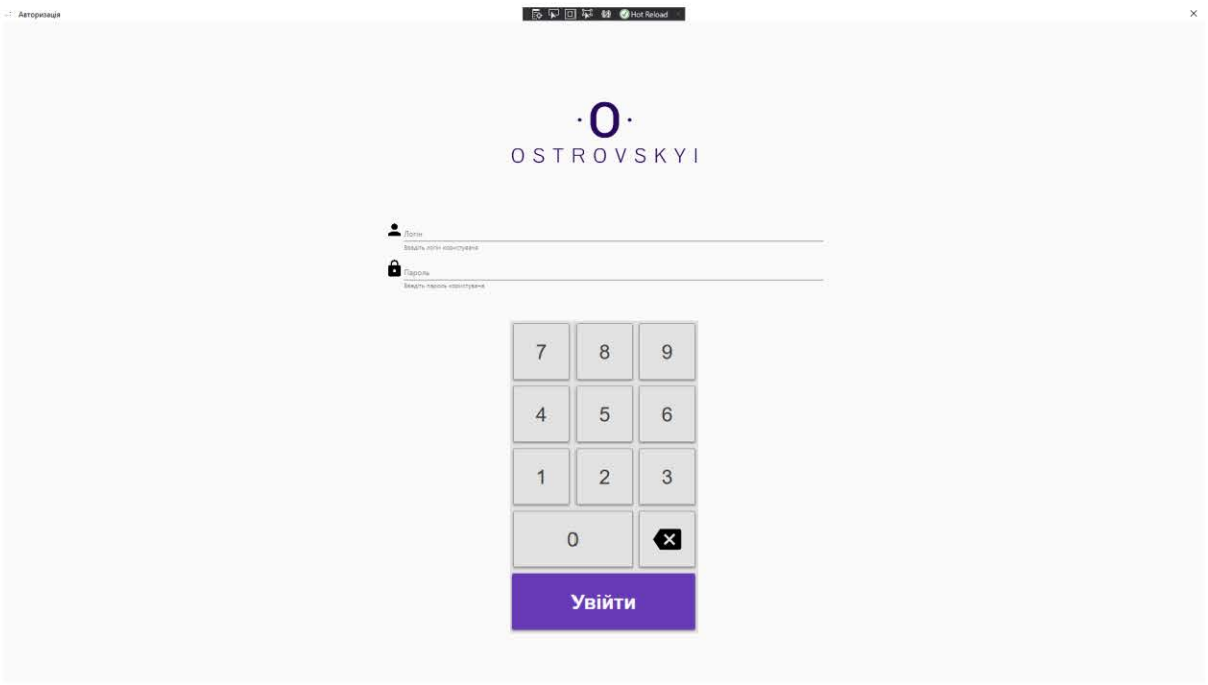


Рисунок 3.2 – Вікно авторизації касира

Далі, на рисунок 3.3, наведено робоче вікно касира. На ньому можна побачити таблицю, яка містить в собі найменування, кількість та ціну товару що були додані до чеку. Також присутнє поле для введення штрих-коду товару та кнопка додати продукт. Є кнопки для видалення продукту з таблиці, вихід до вікна авторизації і кнопка до сплати, яка відкриває нове вікно, де формується чек. Для зручності введення касиром цифр штрих-коду були додані цифри від 0 до 9 у правій частині форми. У нижній частині можна побачити ПІБ касира, поточну дату та час.

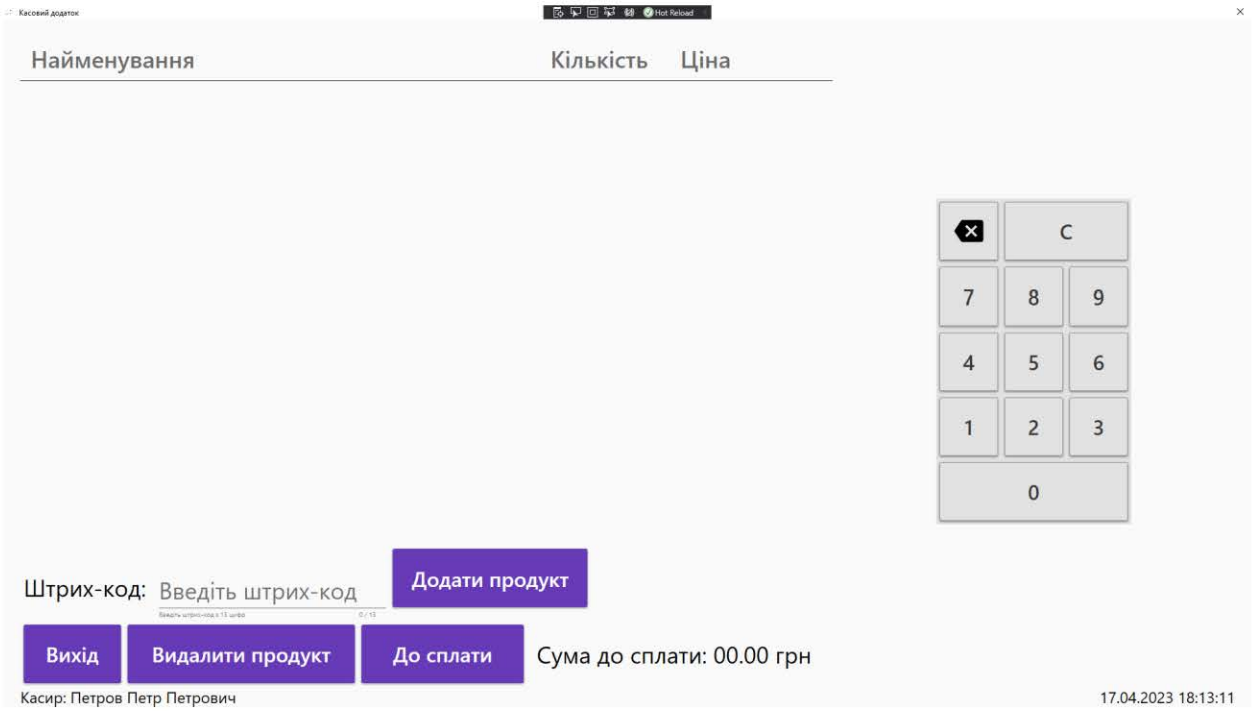


Рисунок 3.2 – Робоче вікно касира

Вікно формування чеку має наступний вигляд (рисунок 3.3). На ньому відображається сума до сплати, сума до сплати зі знижкою. Присутнє поле для введення дисконтної картки користувача, для отримання знижки, кнопка додавання цієї картки та формування чек.

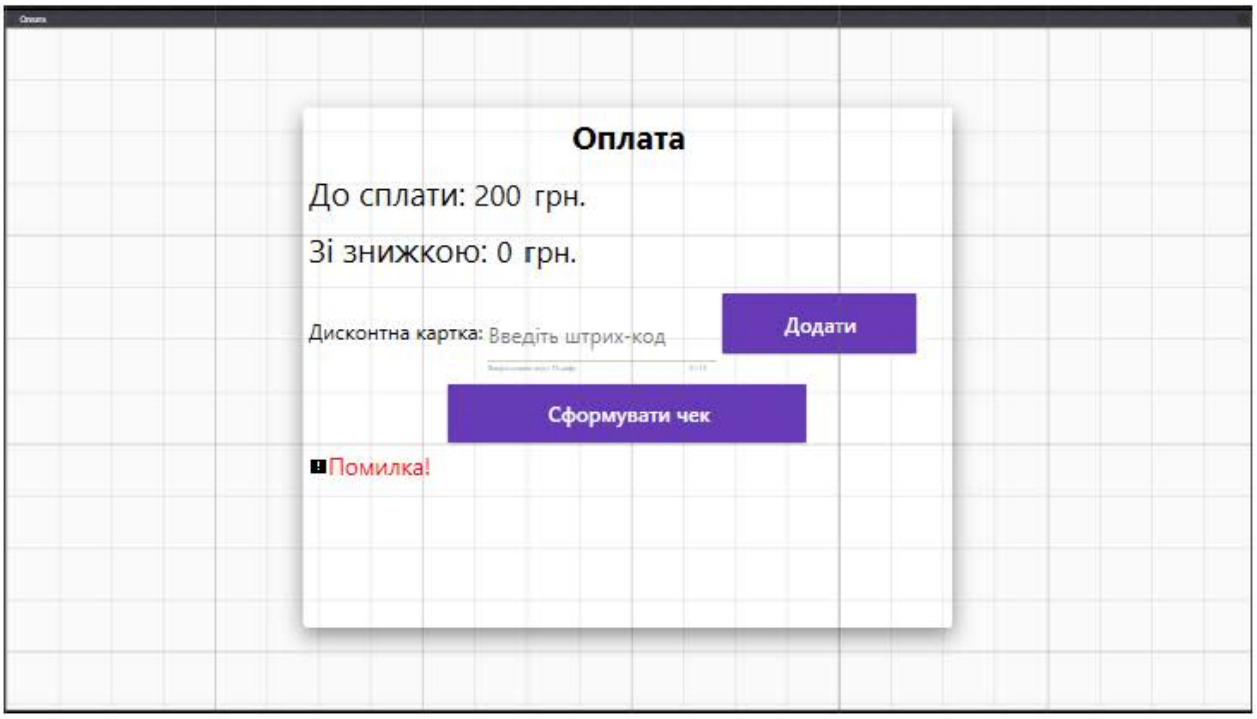


Рисунок 3.3 – Вікно формування чеку

## 3.3 Проектування та розробка бази даних

Для вирішення завдання потрібно було спроектувати та розробити базу даних. У якості інструменту для створення бази даних була обрана Microsoft SQL Server Management Studio. Саме у цьому інструменті можна легко та швидко розробляти базу даних, вносити правки та інше.

У ході проектування та розробки бази даних була створена база даних «CashierDB» які містить 5 таблиць, а саме:

1. Таблиця «Roles», яка містить два стовпці: «Id» та «Role». Стовпець «Id» є первинним ключем і автоматично генерується за допомогою IDENTITY, «Role» є рядком, що містить найменування ролі користувача. Ця таблиця вміщає в собі такі ролі, як: «admin» «cashier» та «customer».

2. Таблиця «Users», яка містить шість стовпців: «Id», «Username», «Password», «FirstName», «LastName», «Patronymic» та «RoleId». «Id» є

первинним ключем і автоматично генерується за допомогою IDENTITY, «Username» та «Password» є рядками, що містять ім'я користувача та пароль відповідно, «FirstName», «LastName» і «Patronymic» містять відповідні дані користувача, а «RoleId» є зовнішнім ключем, що зв'язує користувача з його роллю.

3. Таблиця «Cards», яка містить чотири стовпці: «Id», «UserId», «CardNumber» та «Discount». «Id» є первинним ключем і автоматично генерується за допомогою IDENTITY, «UserId» є зовнішнім ключем, що зв'язує карту з відповідним користувачем, «CardNumber» являє собою рядок, що містить номер картки, а «Discount» є значенням знижки на карту.

4. Таблиця «Products», що містить п'ять стовпців: «Id», «Nomitation», «Barcode», «Amount», «Price» та «Image». «Id» є первинним ключем і автоматично генерується за допомогою IDENTITY, «Nomitation» є рядком, що містить найменування продукту, «Barcode» являє собою рядок, що містить штрих-код продукту, «Amount» представляє ціну продукту, а «Image» представляє зображення продукту.

5. Таблиця «Checks», яка містить п'ять стовпців: «Id», «UserId», «OperationTime», «Discount», «Sum» та «SumWithDiscount». «Id» є первинним ключем і автоматично генерується за допомогою IDENTITY, «UserId» є зовнішнім ключем, що зв'язує перевірку з відповідним користувачем, «OperationTime» представляє дату та час виконання операції формування чеку, «Discount» являє собою знижку відповідно до знижці покупця, «Sum» представляє суму чеку без урахування знижки, а «SumWithDiscount» представляє суму з урахуванням знижки.

Primary Key (первинний ключ) та Foreign Key (зовнішній ключ) - це два ключові поняття, що використовуються в базах даних для зв'язування даних між таблицями.

Первинний ключ (Primary Key) - це унікальний ідентифікатор для кожного запису в таблиці. Він може складатися з одного або декількох атрибутів (стовпців), і його значення не може повторюватися для різних записів.

Первинний ключ дозволяє однозначно ідентифікувати кожен запис у таблиці, що забезпечує унікальність та цілісність даних.

Зовнішній ключ (Foreign Key) - це атрибут (стовпець) в одній таблиці, який посилається на первинний ключ в іншій таблиці. Використання зовнішнього ключа дозволяє створювати зв'язки між таблицями та робити запити, які об'єднують дані з різних таблиць. Коли значення зовнішнього ключа змінюється, система автоматично змінює значення відповідних записів у залежній таблиці.

Зовнішній ключ також може мати значення NULL, якщо запис не має відповідного запису в таблиці, на яку він посилається. Це називається зовнішнім ключем з необов'язковим значенням.

Використання первинного та зовнішнього ключа дозволяє зв'язувати дані між таблицями та забезпечує їх цілісність. Вони є ключовими компонентами в проектуванні баз даних і є необхідними для роботи з багатьма типами баз даних.

Створена база даних відповідає третій нормальній формі (3НФ).

Нормалізація бази даних - це процес організації даних в базі даних таким чином, щоб вони були більш ефективними для зберігання та використання. Головною метою нормалізації є уникнення аномалій даних та забезпечення цілісності даних.

Існує кілька типів нормалізації бази даних:

1. Перша нормальна форма (1НФ): Кожен атрибут таблиці повинен містити тільки атомарні значення, тобто значення, які не можуть бути поділені на менші частини.

2. Друга нормальна форма (2НФ): Кожен неключовий атрибут таблиці повинен залежати від цілого первинного ключа, а не від частини його значення.

3. Третя нормальна форма (3НФ): Кожен неключовий атрибут таблиці повинен залежати від первинного ключа безпосередньо, а не через інші неключові атрибути.

4. Нормальна форма Бойса-Кодда (НФБК): Кожен атрибут таблиці повинен залежати від первинного ключа, а не від будь-якого альтернативного ключа.

5. Четверта нормальна форма (4НФ): Кожен множинний залежний атрибут таблиці повинен бути представлений окремо від інших атрибутів.

6. П'ята нормальна форма (5НФ): Кожен атрибут таблиці повинен залежати від первинного ключа та не може бути розкладений на менші компоненти без втрати інформації.

7. Нормалізація даних є важливою частиною проектування баз даних, оскільки допомагає забезпечити ефективну та надійну роботу з даними, зменшує можливість виникнення помилок та забезпечує консистентність даних.

Для кожної нормальної форми є певні правила, яким вона повинна відповідати. Наприклад, для першої нормальної форми є 5 правил, а саме:

1. кожна комірка таблиці повинна містити атомарне значення;
2. всі стовпці таблиці повинні мати унікальну назву;
3. значення, що зберігаються в стовпці, повинні бути одного типу;
4. кожен рядок у таблиці має бути унікальним;
5. порядок у якому зберігаються дані – не важливий.

На рисунку 3.4 можна побачити створену діаграму бази даних:

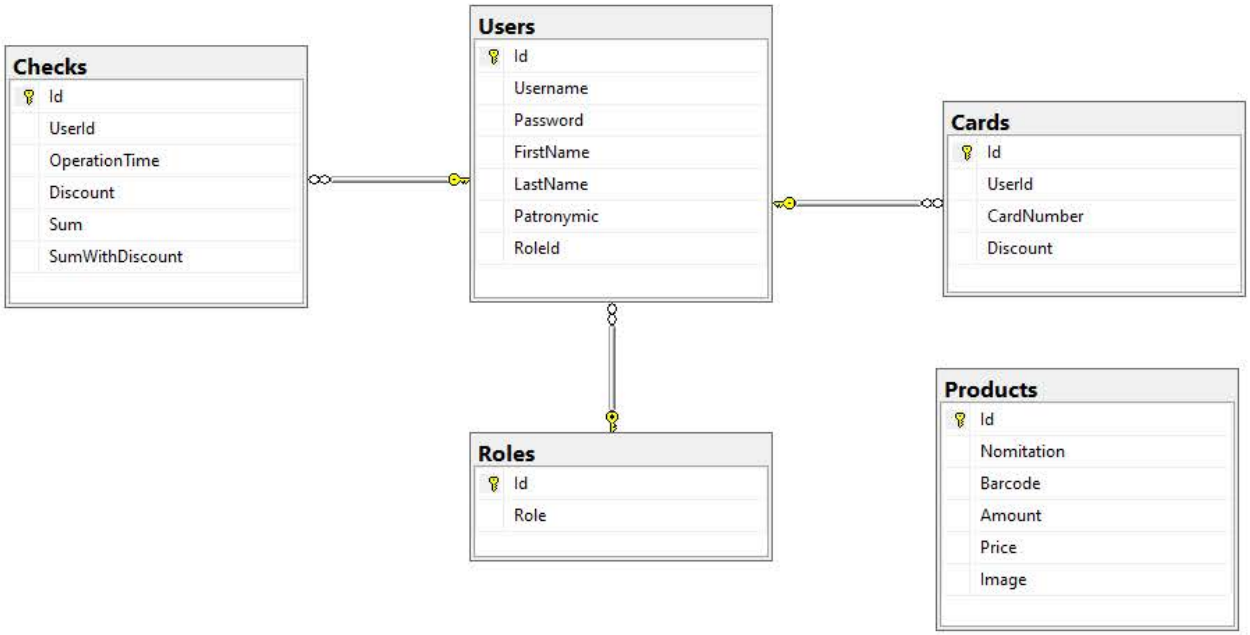


Рисунок 3.4 – Діаграма бази даних

## 3.4 Впровадження бази даних у проект

Так як база даних була створена до написання коду програми, можна використати наступний підхід до розробки моделі, а саме «створення моделі на основі існуючої БД», яка доступна за допомогою Entity Framework Core.

Для цього потрібно додати до проекту «ADO.NET Entity Data Model». Далі обрати модель, а саме «EF Designer from database», як показано на рисунку 3.5:

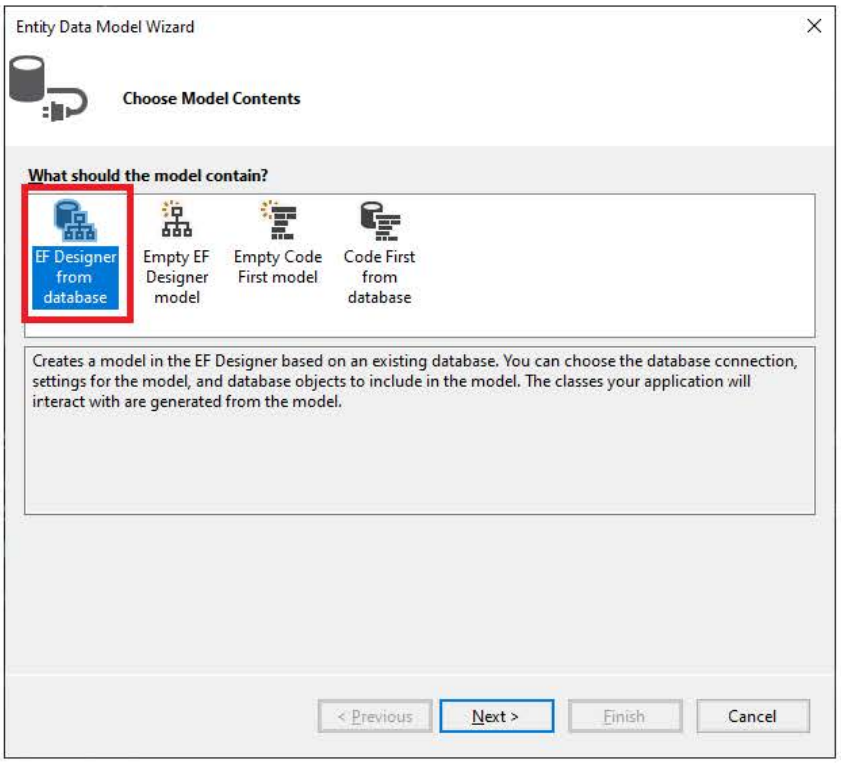


Рисунок 3.5 – Вибір моделі

Далі потрібно обрати строку підключення до бази даних, приклад наведено на рисунку 3.6 :

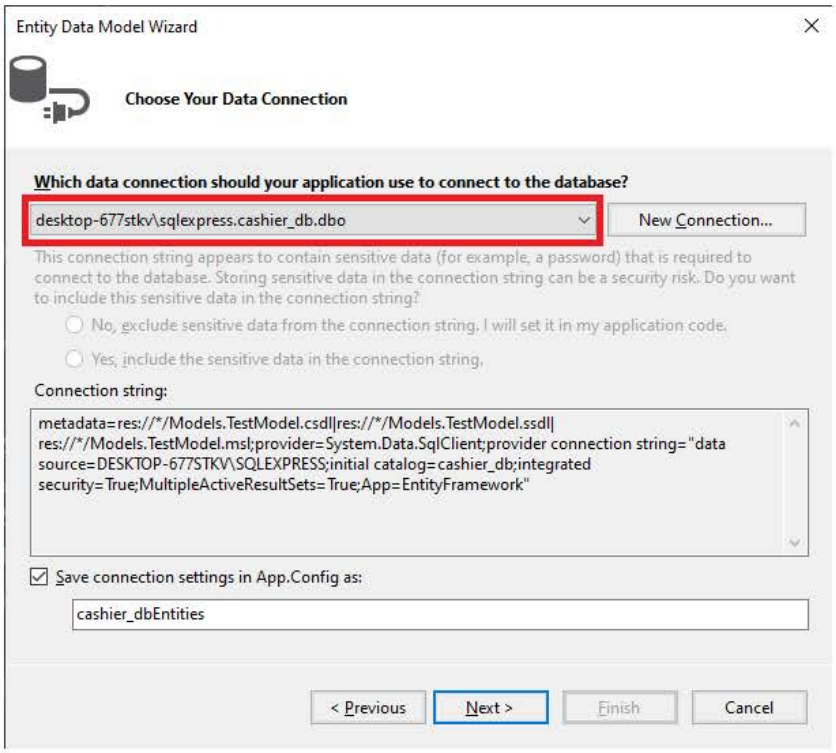


Рисунок 3.6 – Вибір строки підключення

Наступним кроком потрібно обрати всі необхідні створені таблиці, та натиснути кнопку «Finish»:

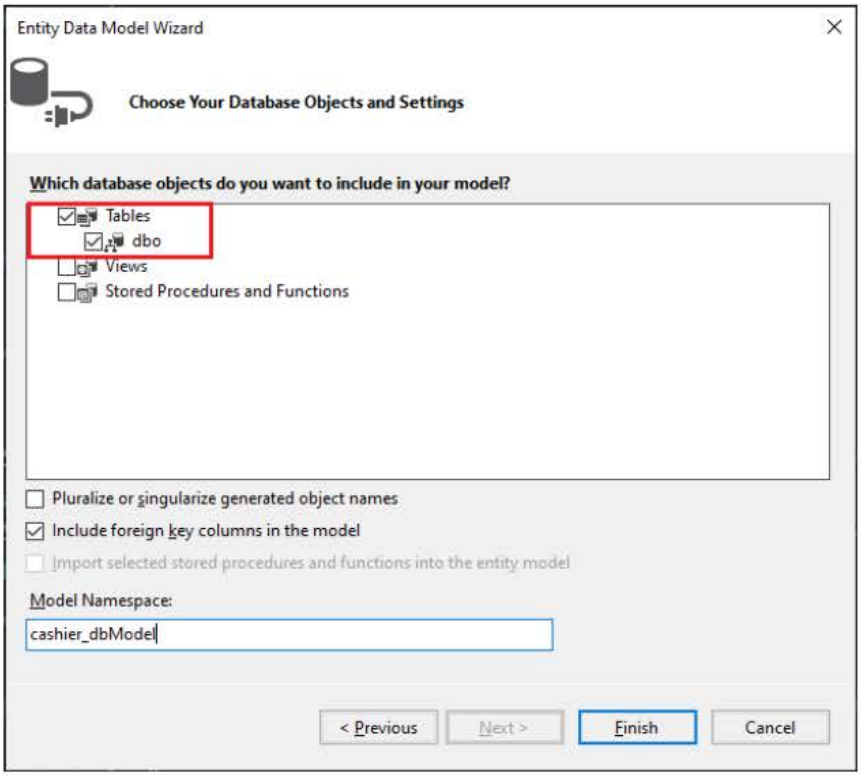


Рисунок 3.7 – Вибір таблиць

Після цього процес впровадження бази даних до проекту завершений. На рисунку 3.8 можна побачити структуру моделі бази даних:

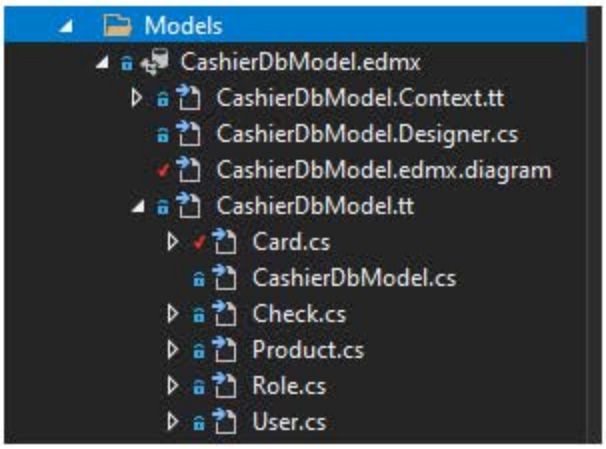


Рисунок 3.8 – Структура моделі бази даних

Якщо обрати, наприклад, User.cs файл, то в ньому можна побачити клас, який був створений автоматично за допомогою Entity Framework Core. Цей клас включає в себе всі поля, які відповідають структурі створеної таблиці у базі даних.

```
public partial class User
{
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]
    public User()
    {
        this.Cards = new HashSet<Card>();
        this.Checks = new HashSet<Check>();
    }

    public int Id { get; set; }
    public string Username { get; set; }
    public string Password { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Patronymic { get; set; }
    public int RoleId { get; set; }

    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Card> Cards { get; set; }
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
    public virtual ICollection<Check> Checks { get; set; }
    public virtual Role Role { get; set; }
}
```

Рисунок 3.9 – Структура таблиці User

На рисунку 3.10 можна побачити діаграму бази даних, яка було створена автоматично за допомогою Entity Framework Core, яку можна відкрити у проекті у IDE Microsoft Visual Studio 2022:

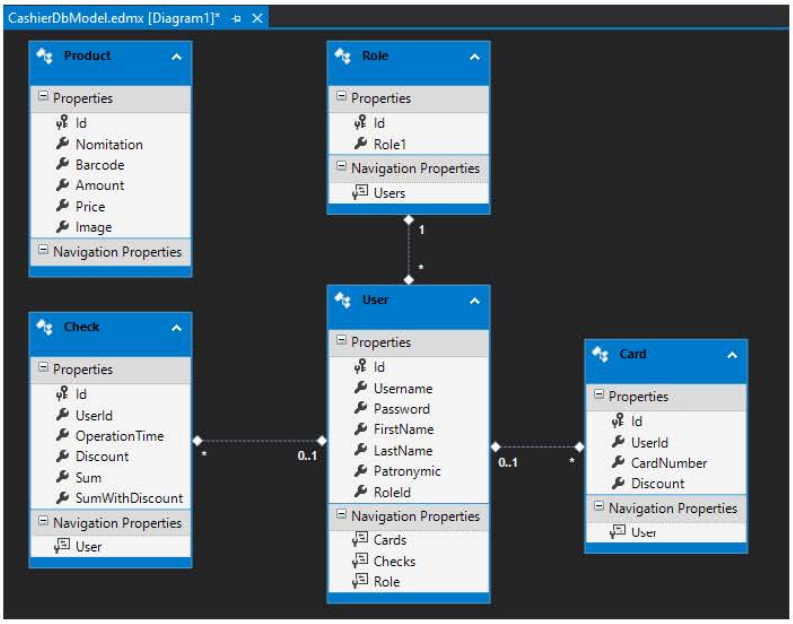


Рисунок 3.10 – Діаграма бази даних

## 3.5 Контрольний приклад та аналіз комп'ютерної реалізації програми

Після запуску програми, касиру потрібно пройти авторизації (рисунок 3.11), для цього йому необхідно ввести логін та пароль. Після цього потрібно натиснути кнопку «Увійти».

Для зручності введення логіну та/або паролю, можна використати цифри, що наведені нижче поля «Логін» та «Пароль».

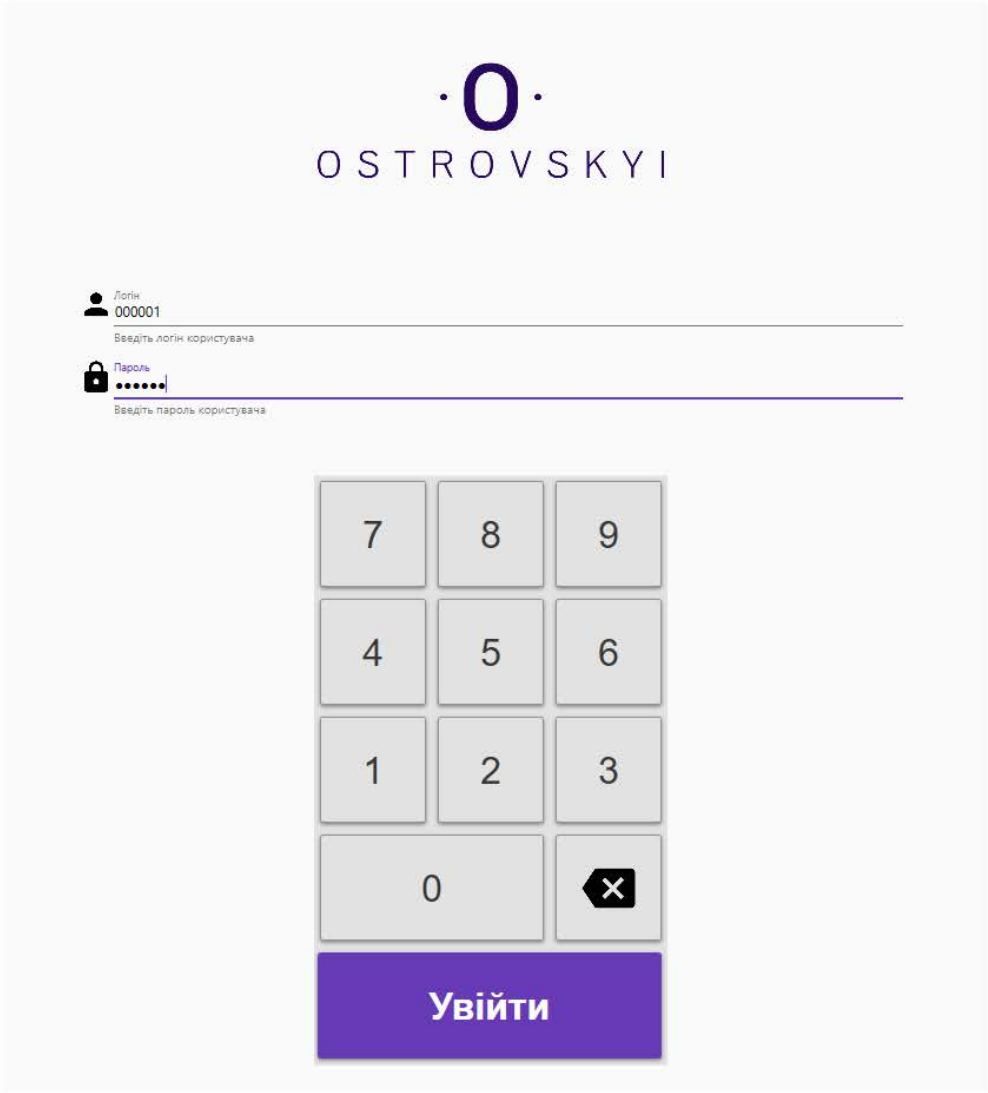


Рисунок 3.11 – Вікно авторизації

Якщо дані були введені некоректні, або якесь поле не було заповнено, під полями, буде виведено текстове повідомлення помилки, приклад наведений на рисунку 3.12).


Рисунок 3.12 – Повідомлення про помилку

Якщо ж касир пройшов авторизацію успішно, відкриється робоче вікно, наведене на рисунку 3.2.

Для сканування штрих-коду продукту, касиру потрібно піднести штрих-код до веб-камери, після цього здійсниться зчитування штрих-коду і касиру потрібно буде натиснути кнопку «Додати продукт».

Якщо зчитати штрих-код через веб-камеру не виходить, можна ввести штрих-код руками, ввевши 13 цифр штрих-коду.

Після додавання продукту, у вікні буде відображено доданий продукт у таблиці і його рисунок у лівому нижньому куті вікна. Також «Сума до сплати» зміниться відповідно з ним (рисунок 3.13).

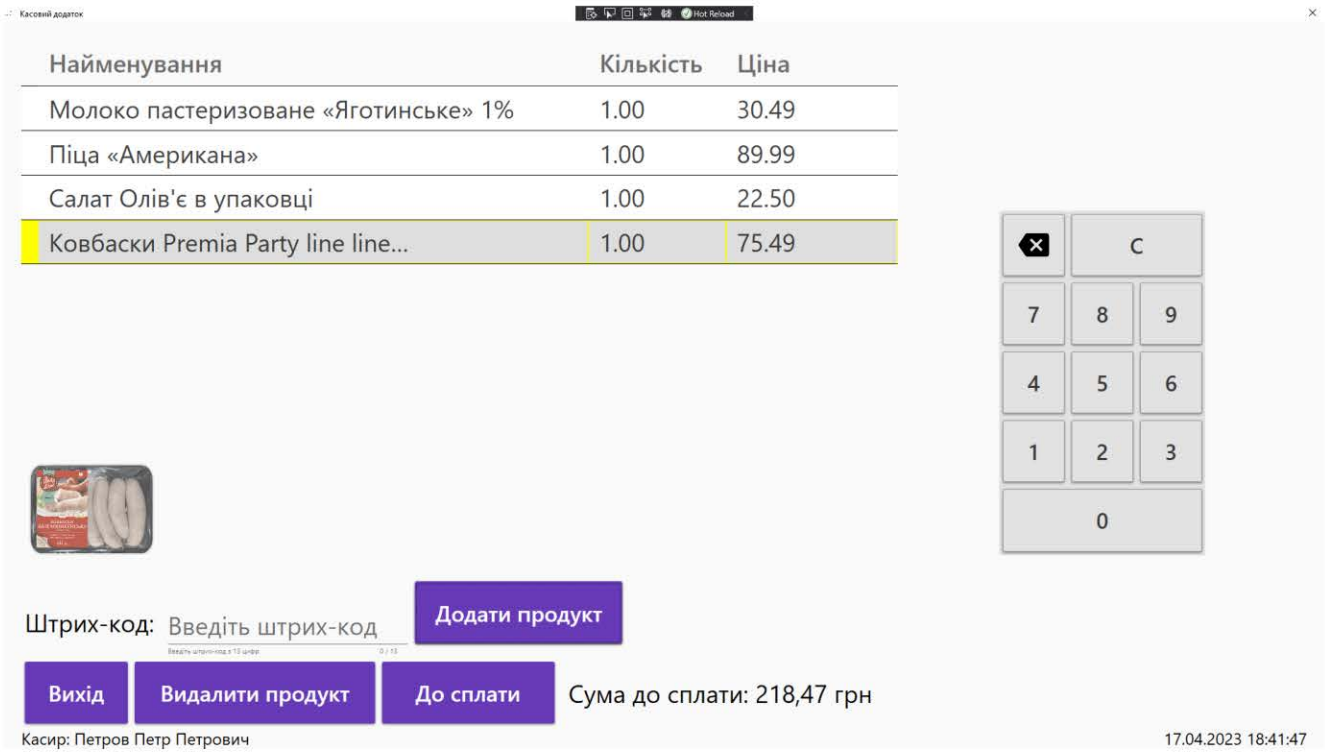


Рисунок 3.13 – Робоче вікно після додавання продуктів

Якщо продукт був додан помилково, касир має можливість видалити його. Для цього необхідно натиснути на цей продукт у таблиці, після чого продукт буде виділено жовтим кольором, потім, залишиться лише натиснути кнопку «Видалити продукт».

За необхідністю, касир може виконати сортування за одним з полів, таких як «Найменування», «Кількість», «Ціна». Для цього потрібно лише натиснути на назву «Найменування», «Кількість» або «Ціна».

Коли всі продукти були додані, касиру потрібно натиснути кнопку «До сплати». Після цього відкриється вікно «Оплата» (рисунок 3.14).

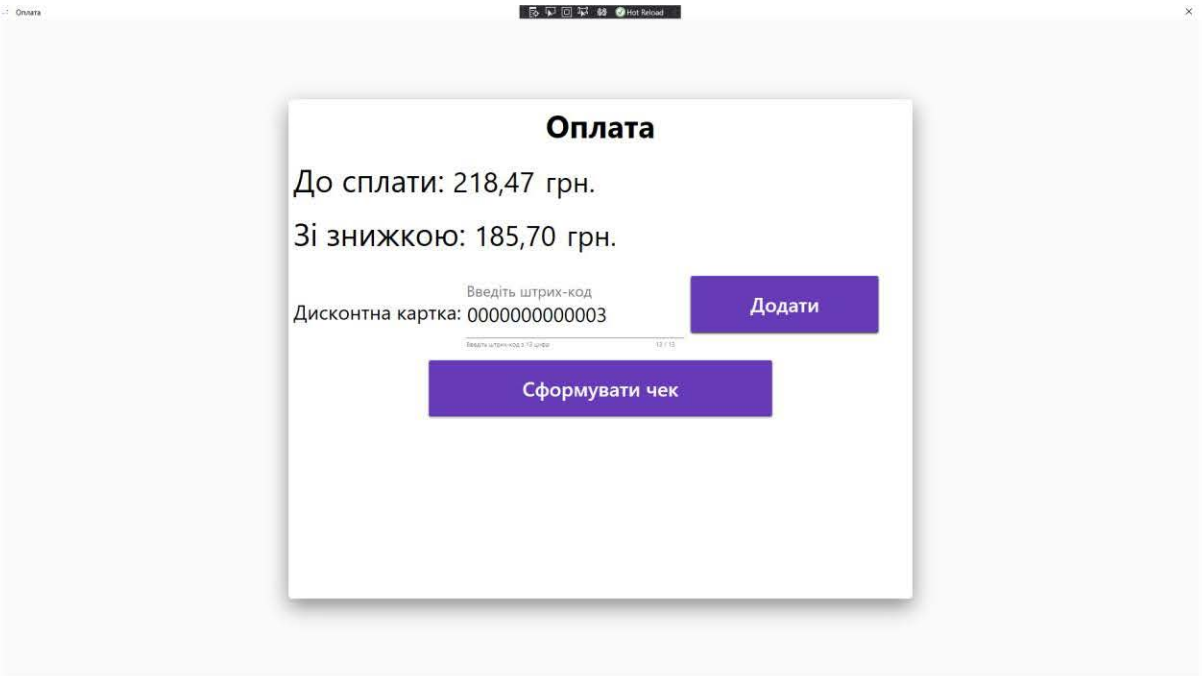


Рисунок 3.14 – Вікно «Оплата»

У цьому вікні надається сума до сплати та сума до сплати зі знижкою, якщо вона є.

Для додавання дисконтної картки покупця, потрібно сканувати її штрих-код, або ж ввести її цифри власноруч у відповідне поле «Дисконтна картка», яка вміщує в себе 13 цифр, після чого треба натиснути кнопку «Додати». Після чого, сума зі знижкою зміниться на відповідну до відсотку знижки покупця.

Після виконаних потрібних дій, потрібно сформувати чек, натиснувши кнопку «Сформувати чек». Після чого буде сформовано чек, та буде можливість роздрукувати його для покупця. Сам чек має наступний вигляд, наведений на рисунку 3.15.

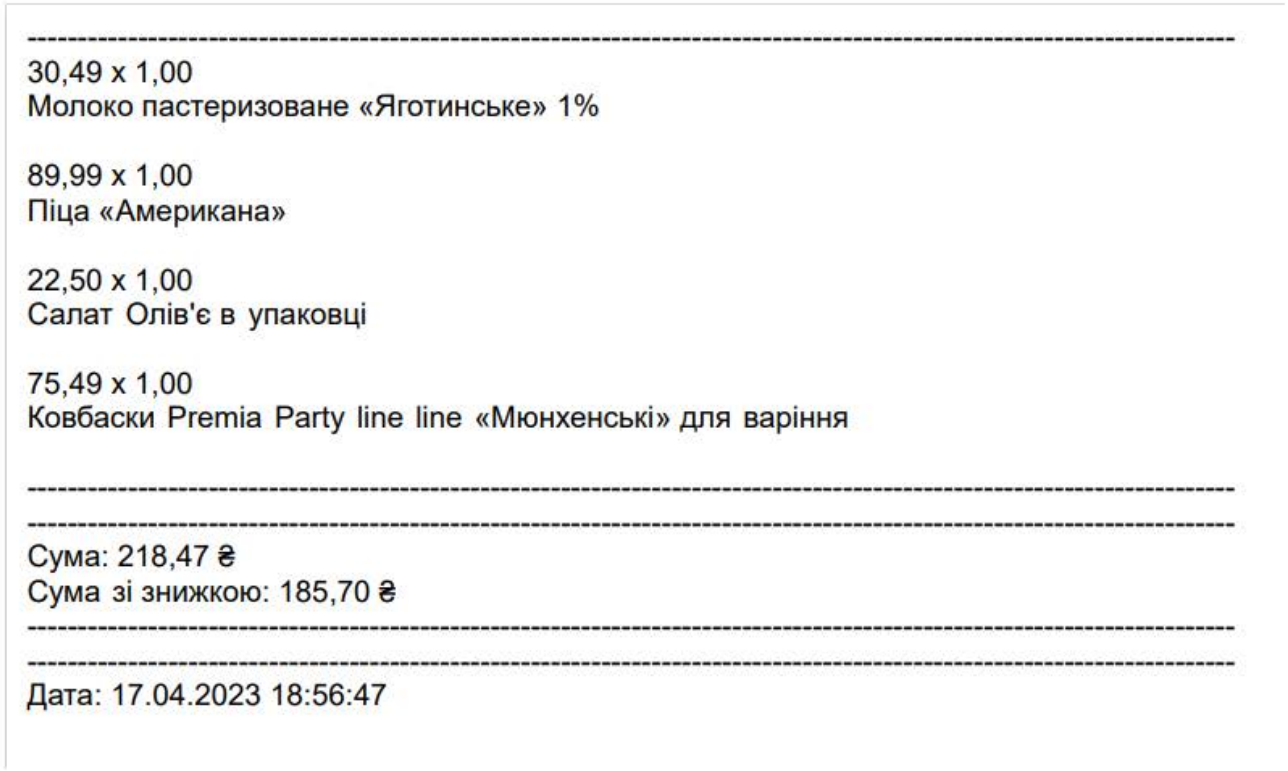


Рисунок 3.15 – Чек

У чеку можна побачити найменування всіх продуктів, що були куплені покупцем, загальну суму та суму зі знижкою, яку потрібно сплатити покупцю, а також дату та час формування чеку.

## 3.6 Висновки до третього розділу

У третьому розділі узагальнюються отримані результати та формулюються висновки, що стосуються цього розділу. Розглядаються основні досягнення, важливі технічні аспекти та результати програми.

Було розглянуті такі питання, як:

1. Проектування структури програми та чому це важливо. Було надано структуру проекту програми, наведені логічні класи програми та їх опис у вигляді таблиці.

2. Проектування візуального інтерфейсу користувача та чому це важливо. Були наведені вікна програми.

3. Проектування та розробка бази даних. Було надано структуру бази даних та її таблиць, а також діаграму бази даних. Розглянуте питання нормалізації БД та її форми нормалізації.

4. Впровадження бази даних у проект. Було наведено алгоритм впровадження готової бази даних до проекту програми за допомогою підходу «створення моделі на основі існуючої БД». Також було наведено приклад структури таблиці, діаграму бази даних.

5. Контрольний приклад та аналіз комп'ютерної реалізації програми. Було розглянута робота програми по крокам, від авторизації касира до формування чеку.

# ВИСНОВКИ

У результаті роботи було розроблено касовий додаток який був написаний на мові програмування C# з використанням технології Windows Presentation Foundation. Його основні переваги, це виведення зображення продукту на екран касиру, для ідентифікації товару, що зменшує шанс сканування та продаж неправильного товару, що в свою чергу, збільшує дохід закладу, який використовує даний додаток. Наступний, але не менш важлива перевага це інтуїтивно-зрозумілий та привабливий інтерфейс додатку, що дозволяю виконувати операції дуже швидко та точно. Ще одна перевага даного додатку полягає в тому, що є можливість додавати товар до чеку як ввівши 13 цифр штрих-коду, так і сканування за допомогою веб-камери, що прискорює процес та зменшує шанс, додавання неправильного товару до чеку, до нуля, що є дуже важливим фактором у таких родах додатків. Також, це ідентифікація користувача, та шифрування його даних за допомогою алгоритму шифрування BCrypt.

Оцінюючи продукт, можна зазначити його численні переваги, так і його недоліки, такі як:

1. потенційна нестабільність програми під час роботи зі скануванням штрих-коду за допомогою веб-камери, порівнюючи з сканером штрих-кодів, через те, що веб-камери не мають спеціальної оптики та зчитувачів, які були розроблені спеціально для сканування штрих-коду. Спеціалізовані сканери штрих-кодів зазвичай мають вбудовані лазерні сканери, які забезпечують швидке та точне зчитування інформації з штрих-коду. По-друге, веб-камери зазвичай не мають такої ж швидкості, як сканери штрих-кодів. Швидкість зчитування є важливим фактором, особливо в області роздрібної торгівлі. Отже, хоча сканування штрих-коду через веб-камеру може бути зручним у певних випадках, воно може бути менш ефективним та менш точним порівняно зі спеціалізованими сканерами штрих-кодів;

2. можливість збоїв під час підключення до бази даних. Це може виникнути через багато причин, наприклад: введені неправильні дані автентифікації, проблеми з мережою, недостатність ресурсів, відсутність дозволів на доступ та застарілий драйвер. Для вирішення цих проблем необхідно аналізувати причини та вживати відповідні заходи.

Розроблений програмний продукт, в рамках виконання дипломної роботи, може знайти своє застосування в різних галузях, як, наприклад, торгівля, ресторани та кафе, та будь-які інші сфери, де потрібна точна обробка платежів та зберігання даних.

Одними з основних технічних рішень, що були запропоновані в роботі, є інтуїтивно-зрозумілий та привабливий інтерфейс, це можливість сканування штрих-коду товару за допомогою веб-камери, та можливість ідентифікації товару за допомогою відображення товару на екрані касового апарату. Ці рішення значно спрощують процес обробки продажу, ідентифікації товару, та швидкість роботи з додатком, який потрібен для обробки платежів.

На підставі отриманих результатів можна запропонувати покращення продукту, наприклад, вдосконалення алгоритму сканування штрих-коду, щоб уникнути можливих збоїв. Також, сканування товару за допомогою спеціалізованих сканерів штрих-кодів, а не сканування через веб-камеру. Використання цього продукту можна рекомендувати у різних галузях, де він може бути корисним для обробки платежів та зберігання даних.

На базі отриманих висновків можна запропонувати наступні рекомендації:

1. При розробці програмного забезпечення, що використовує бази даних, необхідно забезпечувати максимальну стійкість до помилок, що виникають при роботі з базою даних. Для цього можна використовувати технології підключення з автоматичним відновленням, перевірку доступність бази даних перед початком роботи з нею тощо.

2. При проектуванні бази даних необхідно враховувати можливість росту обсягів даних і планувати відповідну ємність бази даних, а також запобігати збоїв у роботі бази даних шляхом оптимізації запитів, індексування тощо.

3. Для підвищення стійкості роботи бази даних рекомендується використовувати програми-моніторингу, які відслідковують стан бази даних та оперативно сповіщають про можливі проблеми.

4. Необхідно забезпечувати регулярне збереження даних та резервне копіювання бази даних, щоб у разі виникнення помилки можна було відновити роботу системи з мінімальними втратами.

У результаті дослідження та розробки касового додатку було успішно досягнуто поставлені завдання. Програмне забезпечення було розроблене з урахуванням вимог до привабливого та зручного користувальницького інтерфейсу. Була впроваджена фунціональність сканування штрих-кодів товарів, що дозволяє швидко та точно ідентифікувати продукти. Крім того, було реалізовано систему формування чеків та інших операцій, що забезпечує ефективний облік покупок клієнтів.

В процесі розробки було приділено особливу увагу стабільності та безпеці програмного забезпечення. Були використані методи шифрування для збереження паролів користувачів та захисту конфіденційної інформації.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Poster. [Інтернет-ресурс] URL: https://joinposter.com/ua
2. CashierLive. Smart POS software. [Інтернет-ресурс] URL: https://www.cashierlive.com/
3. Square. A point of sale for however you sell. [Інтернет-ресурс] URL: https://squareup.com/us/en/point-of-sale
4. Loyverse. [Інтернет-ресурс] URL: https://loyverse.com/
5. Shopify POS. [Інтернет-ресурс] URL: https://www.shopify.com/pos
6. POS in C# and SQL Server Free Source Code. [Інтернет-ресурс] URL: https://www.sourcecodester.com/c/13150/pos-c-and-sql-server-2017.html
7. Point of Sale project in C# .NET with source code. [Інтернет-ресурс] URL: https://www.kashipara.com/project/c-net/882/complete-point-of-sale-pos-shop-management-system-c-net-project-download
8. Bcrypt. [Інтернет-ресурс] URL: https://github.com/BcryptNet/bcrypt.net
9. Про що нам говорять штрих-коди? [Інтернет-ресурс] URL: https://consumerhm.gov.ua/544-pro-shcho-nam-govoryat-shtrikh-kodi
10. Бібліотека ZXing.Net. [Інтернет-ресурс] URL: https://github.com/micjahn/ZXing.Net
11. The C# Barcode Library. [Інтернет-ресурс] URL: https://ironsoftware.com/csharp/barcode/blog/compare-to-other-components/zxing-net-generate-qr-code-barcode-alternatives/
12. AForge.NET Framework. [Інтернет-ресурс] URL: http://www.aforgenet.com/framework/
13. AForge.NET Framework. [Інтернет-ресурс] URL: https://github.com/andrewkirillov/AForge.NET
14. SQL Server Management Studio. [Інтернет-ресурс] URL: https://www.tutorialsteacher.com/sqlserver/sql-server-management-studio

15. Microsoft Docs. Download SQL Server Management Studio. [Інтернет-ресурс] URL: https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16
16. ADO.NET. [Інтернет-ресурс] URL: https://metanit.com/sharp/adonet/1.1.php
17. Entity Framework Core. [Інтернет-ресурс] URL: https://github.com/dotnet/efcore
18. Офіційний сайт Material Design in XAML. [Інтернет-ресурс] URL: http://materialdesigninxaml.net/
19. Material Design in XAML. [Інтернет-ресурс] URL: https://github.com/MaterialDesignInXAML/MaterialDesignInXamlToolkit
20. Нормалізація відношень при проектуванні БД. [Інтернет-ресурс] URL: https://rdb.dp.ua/uk/chapter_03
21. Point of Sale (POS) – Inventory & Super Shop Management System – C#. [Інтернет-ресурс] URL: https://github.com/livealvi/.NET-Point-of-Sale-POS--Csharp#database-diagram
22. Building a Point-of-Sale Application in Blazor WebAssembly .Net 7 C# Tutorial. [Інтернет-ресурс] URL: https://trystanwilcock.com/2023/02/20/building-a-point-of-sale-application-in-blazor-webassembly-dot-net-7-c-sharp-tutorial/
23. P. Sriramya and R. A. Karthika. Providing password security by salted password hashing using BCRYPT algorithm [Інтернет-ресурс] URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=9d82620f0866ec773b12c30f90d70b74661f972f