ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: «Розробка програмного забезпечення для касових апаратів коворкінгів мовою C#»

Виконав: студент 4 курсу, групи ПД-43

спеціальності

121 Інженерія програмного забезпечення

(шифр і назва спеціальності/спеціалізації)

Матківський Аскольд Олегович

(прізвище та ініціали)

Керівник Трінтіна Н.А.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Київ –2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти -«Бакалавр»

Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення Негоденко О.В.

“____” _____2022 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТА

Матківському Аскольду Олеговичу

(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для касових апаратів ковіркінгів мовою C#»

Керівник роботи: Трінтіна Н.А., к.т.н., доцент

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «18» лютого 2022 року .

2. Строк подання студентом роботи «3» червня 2022 року

3. Вхідні дані до роботи

Методи обробки зображень;

Науково-технічна література з питань, пов’язаних з програмним забезпеченням щодо створення програм на мові програмування C#

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

4.1 Системи для автоматизації роботи коворкінга.

4.2 Вимоги та оцінка якості системи.

4.3 Опис проектування системи.

4.4 Опис використаних технологій.

5. Перелік демонстраційного матеріалу

1. Актуальність проблеми
2. Існуюче програмне забезпечення
3. Принцип роботи інформаційної системи
4. Розпізнавання та групування даних
5. Архітектура бази даних
6. Логічна діаграма компонентів архітектури програмного забезпечення

6. Дата видачі завдання«11» квітня 2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	11.04-16.04	Виконано
2	Вимоги до системи	17.04-20.04	Виконано
3	Створення та навчання моделі для вилучення полів	21.04-27.04	Виконано
4	Створення та навчання моделі для вилучення таблиць	28.04-07.05	Виконано
5	Концепція та архітектура програмного забезпечення	08.05-13.05	Виконано
6	Вступ, висновки, реферат	14.05-21.05	Виконано
7	Розробка обов‘язкових демонстраційних матеріалів	22.05-30.05	Виконано
8	Попередній захист роботи	03.06	Виконано
9	Здача роботи	17.06.22	

Студент _____ Матківський А. О.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Тринтіна Н.А

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Ключові слова: програмний модуль, програмне забезпечення, термінал-каса,каса, антикафе, модуль, об’єктно-орієнтований аналіз, проектування, класи, об’єкти, C#, Windows Forms, .Net, SQLite.

Об’єкт дослідження : процеси обліку для терміналу-каси антикафе "Compass" .

Предмет дослідження:програмні засоби для роботи з касовими апаратами.

Мета дослідження: дослідити особливості роботи коворкінгу та розробити програмний модуль, який автоматизує та прискорить бізнес-процеси підприємства.

Основними методами є: аналіз та моделювання.

В ході дослідження були одержані: програмне забезпечення для терміналу- каси антикафе, програмні модулі, об'єктно-орієнтовані діаграми специфікаціїпрограмних модулів та процесів підприємства.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ООП (Об'єктоорієнто́ване програмування)— одна з парадигм програмування, яка розглядає програму як множину об'єктів, що взаємодіють між собою

СЕЙВ (З англійської - зберігати.) файл, який зберігає певну інформацію, яку потім можливо зчитати за допомогою певної програми.Найчастіше використовується у комп’ютерних іграх.

ПЗ Програмне забезпечення

ADO Від англ. activex data objects —«об'єкти даних activex» — прикладний програмний інтерфейс для доступу до даних, розроблений компанією microsoft (ms access, ms sql server) і заснований на технології компонентів activex. ado дозволяє представляти дані з різноманітних джерел: (реляційної субд, текстових файлів тощо) в об'єктно-

орієнтованому програмуванні виді.

API Прикладний програмний інтерфейс (англ. application programming interface, скорочено api) — це сукупність засобів та правил, що вможливлюють взаємодію між окремими складниками програмного забезпечення або між програмним та апаратним забезпечення.

RSA Абревіатура від прізвищ Rivest, Shamir та Adleman — криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел.

CRUD З англ. Create Read Update Delete - 4 базові функції управління даними «створення, зчитування, зміна і видалення».

ОС (англ. operating system, os) — це базовий комплекс програм, що виконує керування апаратною складовою комп'ютера або віртуальної машини; забезпечує керування обчислювальним процесом і організовує взаємодію з користувачем.

ЛОГ (англ. log file, від англ. log — колода)— спеціальний файл, у якому накопичується зібрана службова та статистична інформація про події в системі (програмі). операційні системи (особливо це стосується серверних ос) та серверне програмне забезпечення зазвичай мають розвинуту систему ведення

логів. за допомогою них можна змусити систему (програму) реєструвати у лог-файлах фактично будь-які події. відповідно різні типи подій, різну інформацію можна зберігати у своїх спеціалізованих логах.

ВСТУП

Об’єктно-орієнтоване програмування є однією з найважливіших та ефективніших парадигм. ООП дозволяє розробляти та розглядати програмні модулі у вигляді об’єктів і класів, що значно поліпшує сприйняття задачі та дозволяє розробити продуктивні рішення. До того ж, програмне забезпечення, особливо велике корпоративне рішення, розроблене з використанням об’єктно-орієнтованого підходу легше масштабується тасупроводжується.

Така перевага ООП є цілком логічною та очікуваною, оскільки людині набагато легше сприймати вирішувати задачі, коли їх можна розкласти на складові та провести аналогії із реальним світом. Отже використання та розуміння ООП є дуже важливою частиною розробки програмного забезпечення.

Обґрунтування вибору теми і її актуальність базується на відсутності безплатних , або просто дешевих аналогів програм для автоматизації роботи коворкінгів. В просторах інтернету немає програми з таким задачами.

Також важливо зробити програму , яка буде гнучка до всіх вимог. Оскільки , в кожному коворкінгу зазвичай потребуються різні вимоги. Вони можуть залежати , від розміру коворкінгу , типу зайнятих місць, та від послуг які надає коворкінг.

Ступеню вивченої проблеми даної курсової роботи є робота закладу антикафе,бізнес процеси якої потребують автоматизації.

Антикафе - тип громадських закладів соціального спрямування, де основною характеристикою є оплата в першу чергу за проведений у закладі час, до вартості якого входять різні частування, розваги і заходи. Основні процеси, які потребують автоматизаціїце:

- Рахування проведеного клієнтом часу у режимі реального часу

- Рахування суми оплати клієнта за проведений час
- Рахування суми оплати клієнта за додаткові послуги (оренда TV, приставок, комп’ютерів, кімнат, придбання їжі, тощо)
- Збір статистичних даних, які можуть покращити роботу закладу (інформація про робочі зміни та оплати)
- Формування статистичних даних у звіт (наприклад таблиці Excel)
- Створення та формування чеку
- Зберігання усіх активних клієнтів у таблиці, з можливістю додаваннята редагування нотатків.

Методика дослідження: протягом роботи проведено аналіз, що дозволив дослідити інші програми коворкінг центрів. Було використано метод порівняльного аналізу, що задіяний при ознайомленні із даними вітчизняного та закордонного досвіду. При визначенні особливостей формування програм для коворкінг центрів був використаний метод узагальнення.

Наукова новизна дослідження полягає в тому , що дана область з точки зору програмування не є вивченою в достатній мірі.Також всі аналоги використовують для зберіання інформації сервер. Це в свою чергу робить програмний продукт дорогим у використанні.

Практична значущість результатів дослідження є створення правильної програми для користування коворкінгам.

1.АНАЛІЗ НАЯВНИХ ЗАСОБІВ ТА ПРОГРАМ ДЛЯ ОРГАНІЗАЦІЇ АВТОМАТИЗАЦІЇ РОБОТИ КОВОРКІНГІВ.

У вільному доступі немає багато аналогів в інтернеті, оскільки кожна компанія робить програмний продукт під свою фірму на замовлення..

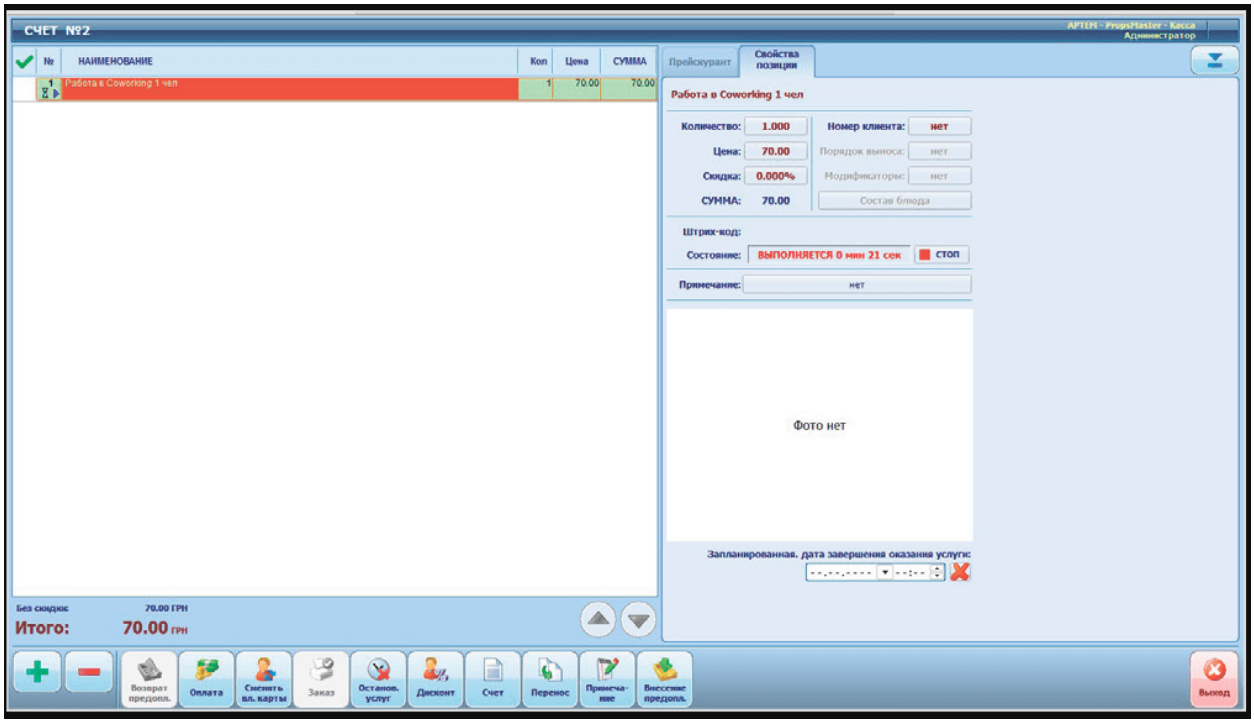


Рисунок 1.1.1 ”Система автоматизації SERVIO”

Програма автоматизації Servio. Вона важка в розумінні , для адміністратора. Інтерфейс написаний російською мовою. Також важливим недоліком є те , що при повищенні цін , потрібно звертатися до програмістів , за подальшим обслуговуванням. Ще одним мінусом програми є відсутність обрахування доходу підприємства.

Програма працює на сервері , тому коли відсутній інтернет , або світло ,вся робота в коворкінгу зупиняється. З плюсів використання сервера , є те що дані зміни , при перебиванні можна швидко відновити.

Програма автоматизації Communa. Інтерфейс написаний українською мовою. Програма формує звіти за день, за місяць ,однак , її недоліком , є те що вона також привязана до використання інтернету.

Вимоги	Порівняльна таблиця		
	Communa	Visio	CashBox
Формування звітів за зміну	+	-	+
Контроль часу	+	+	+
Стійкість до перебивань	-	-	+
Контроль прибутків фірми	+	-	+
система знижок	-	+	-
Гнучкіть до різних коворкінгів	-	-	+

Рисунок 1.1.2 «Порівняльна таблиця аналогів»

1.1.Обгрунтування вибору мови програмування.

Оскільки термінал-каса у більшості випадків представляє з себе персональний комп'ютер (найчастіше моноблок) на базі ОС Windowsверсії 7 та вище й повинно мати мінімальну залежність від інтернет з'єднання – платформа .Net, а саме мова програмування C# разом із прикладним програмним інтерфейсом Windows Forms, відповідальним за інтерфейс користувача, ідеально підходить для створення програмного забезпечення антикафе.

1.2.Моделювання та проектування інформаційної системи.

Загальна структура проекту складається із 3 умовно позначених модулів:

- Джерело (програмне забезпечення терміналу- каси)
- Сховище (модуль відповідальний за зберіганняданих та ідоступ до них)

Звітність (програмне забезпечення для перегляду та формування звітності на основі статистичних даних)

Дані переміщується з одного модуля в інший у чіткійпослідовності . Дані формуються під час роботиПЗ каси (кількість клієнтів, мінімальний час,

максимальний час, суми оплати, типи оплати, данізміни, тощо).

Під час робочої зміни у модулі джерелістворюється дані про сесію, де підраховуються дані зміни, які потім переміщуються у модуль сховище, а усі оплати автоматично відправляються у вищезгаданий модуль.

У сховищі сформовані дані зберігаються для подальшої обробки у модулі звітності (перегляд, пошук, формування звіту).

Дані які зберігаються під час робочої зміни, повинні бути захищені від втрати у випадку відключення електропостачання, тому під час своєї роботи модуль джерело повинен створювати спеціальний файл зліпок, подібний до сейвів у комп’ютерних іграх. Цей файл зберігає дані робочої зміни та дозволяє завантаживши його у ПЗ відновити сесію.

2.МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ.

2.1.Модель предметної галузі.

Для правильного розуміння проєкту , нам необхідно мати цілесне уявлення про моделі галузі. Моделюю предметної галузі розуміють систему , котра візуалізує функціонування або структуру предметної області.

Завчасне моделювання предметної області скорочує термін проєктних робіт,та дозволяє отримати якісніший , та ефективніший програмний продукт. Без неї , збійльшується шанс допущення помилок в подальшому ,що призводять до фінансових витрат при подальшій перепроєктуванні системи.

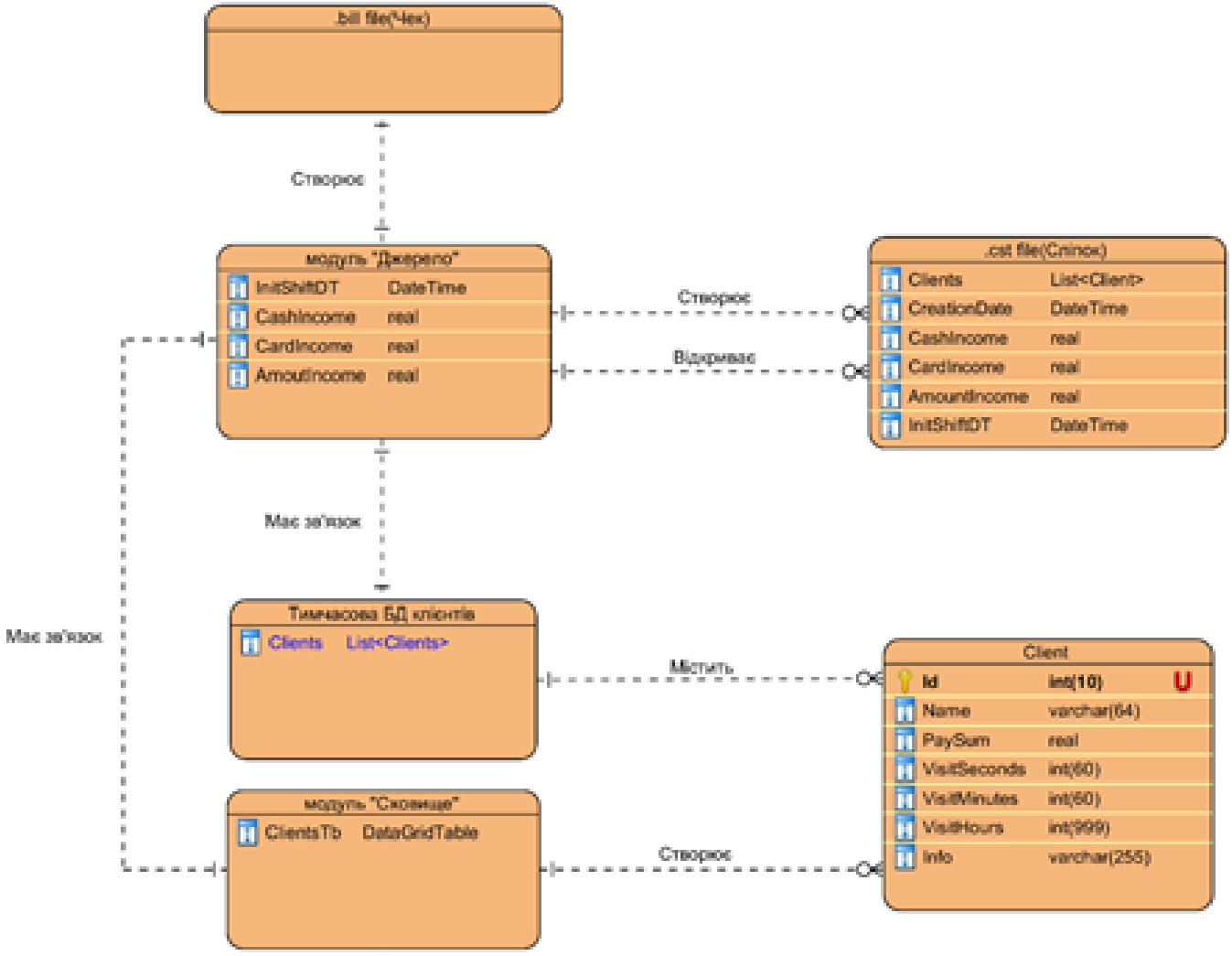


Рисунок 2.1.1 ERD

2.2.Прецеденти

ПЗ має три основних типи користувача: оператор терміналу-каси, менеджер звітності та адміністратор терміналу-каси. Тому щоб показати загальні межі предметної області, загальних вимог до функціональної поведінки та взаємодії ПЗ з користувачем, було використано графічний спосіб специфікування вимог – діаграму прецедентів UML .

Рисунок 2.2.1 Use case діаграма (Оператор терміналу-каси)(Додаток В)

Рисунок 2.2.2 Use case діаграма (адміністратор терміналу-каси)

2.3.**Нефункціональні вимоги**

«Нефункціональні вимоги описують не те, що софт буде робити, а те, як він буде робити це, наприклад, вимоги до продуктивності, до зовнішнього інтерфейсу, до обмежень дизайну та атрибутів якості. Нефункціональні вимоги важкі для тестування; тому, вони, як правило, оцінюються суб’єктивно.»[8].

Розробка документу специфікація**.**Потрібно створити програму , яка зможе рахувати час , який резидент перебуває у коворкінгу.Цей час має множитися на тариф (ціну/год.),і по закінченні перебування клієнта в будівлі , закінчувати його зміну , та видавати чек, також програма має передбачати в собі можливість зміни тарифу , щоб подальша зміна ціни (враховуючи часту девальвацію) , не

потребувало виклику програміста.

Програма повинна вміти створювати велику кількість таймерів , і обмежити способи шахрайства з боку адміністраторів коворкінгу . для спрощення підрахунків фірмою прибутків , потрібно також розробити звіти , щоденні ,щотижневі, щомісячні . це основні вимоги до програми.

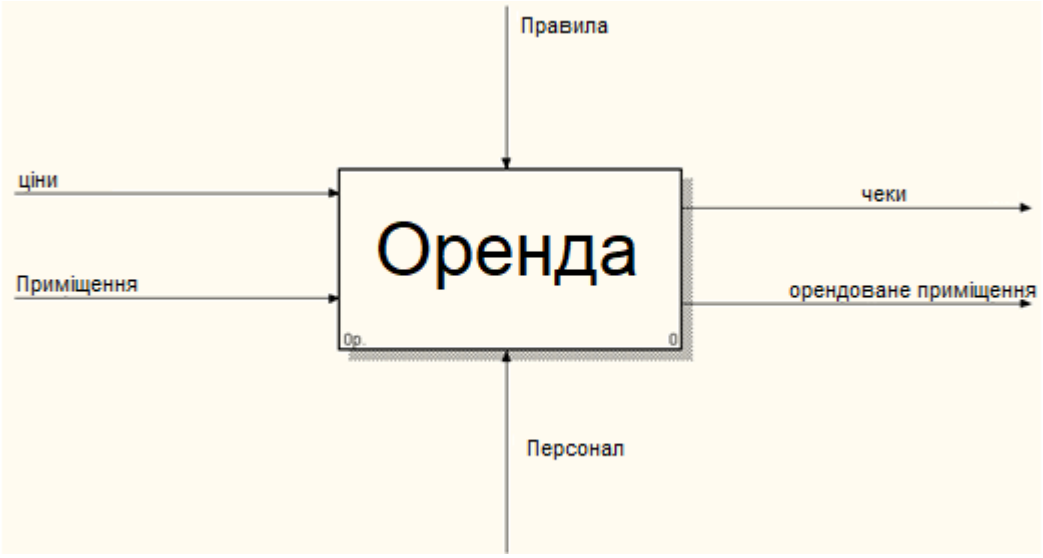


Рисунок2.3.1.”Контекстна Діаграма”

2.4.Модель проектування

Найкращою моделлю програмування для даного типу програми буде ітераційна модель розробки.Ця модель найкраще підходить малим проектам , враховуючи дешевизну цього методу , та подальшу простоту підтримку програми.

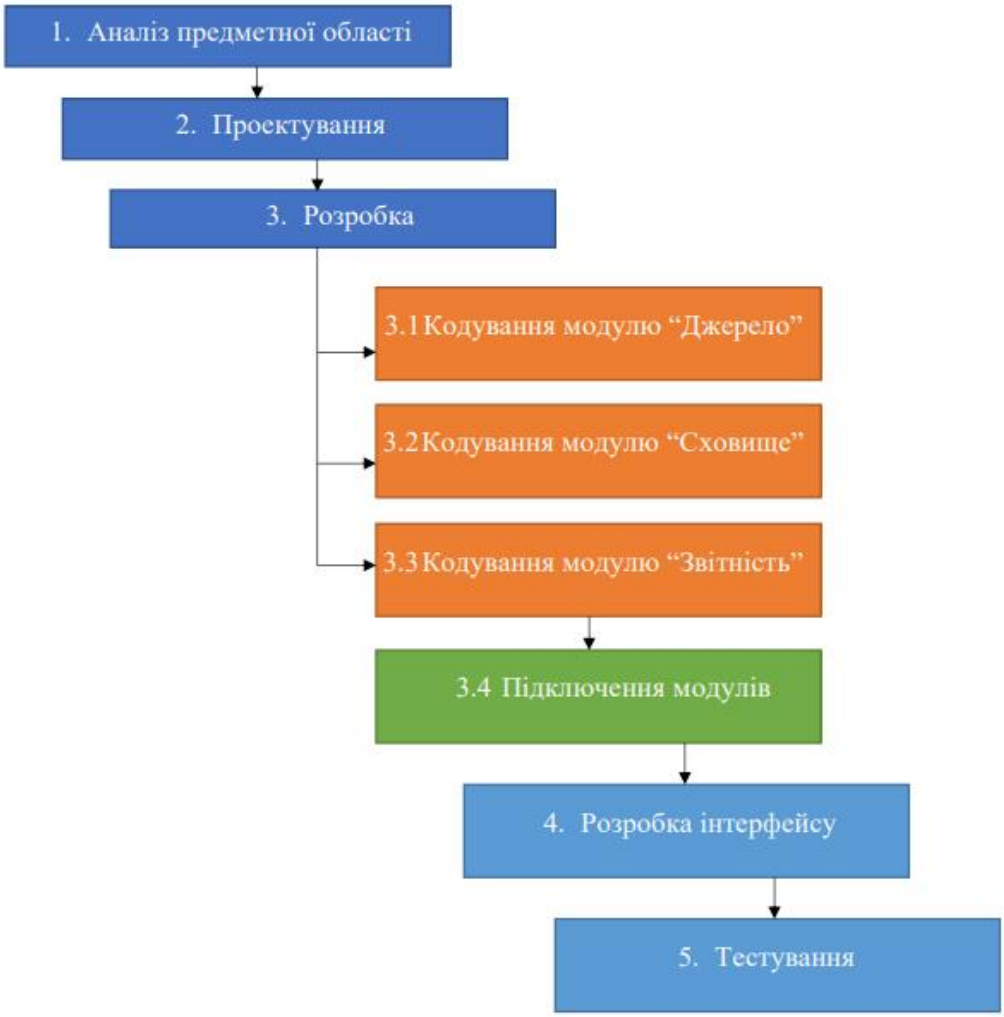


Рисунок2.4.1.“Ітераційний метод розробки”

2.4.1.**Проектування діяльності**

Розгалуження на діагарамі діяльності, показує нам вибір , оплата у коворкінгу зразу , чи по проведеному часу.

Ліва частина діаграми показує вибір , оплати як в анткафе , а права сторона діаграми показує оплату оренди офісу наперед.

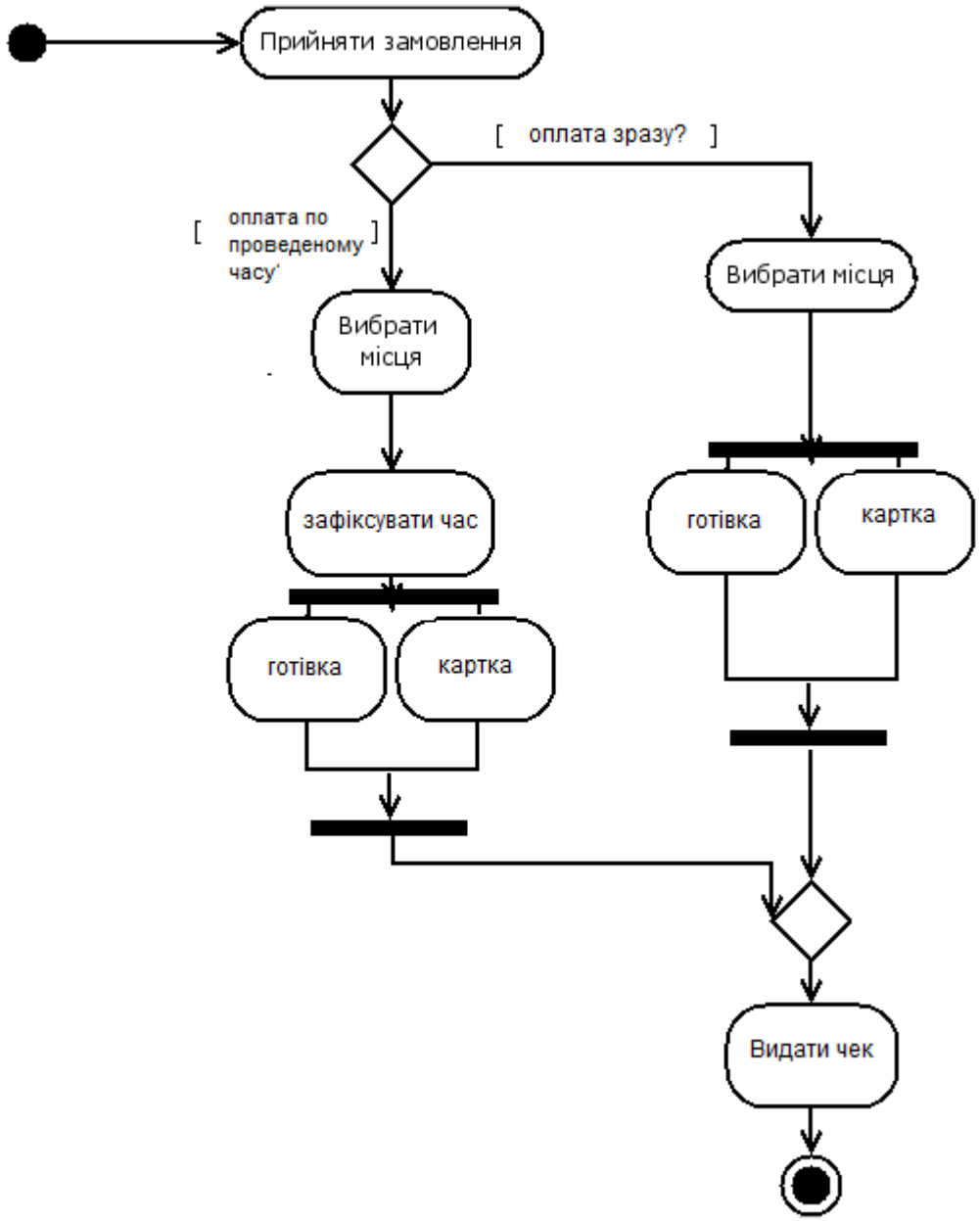


Рисунок2.4.2.»Діаграма діяльності»

2.4.3.**Проектування класів та їх взаємодія**

Оскільки ПЗ “CashBox” має модульну структуру, де кожен модуль є класом та взаємодіє, Створює інші класи та файли, використаємо діаграму класів UML з метою відображення статичної структури ПЗ проекту, описати взаємозв’язки між класами та самі структуру класів в цілому.

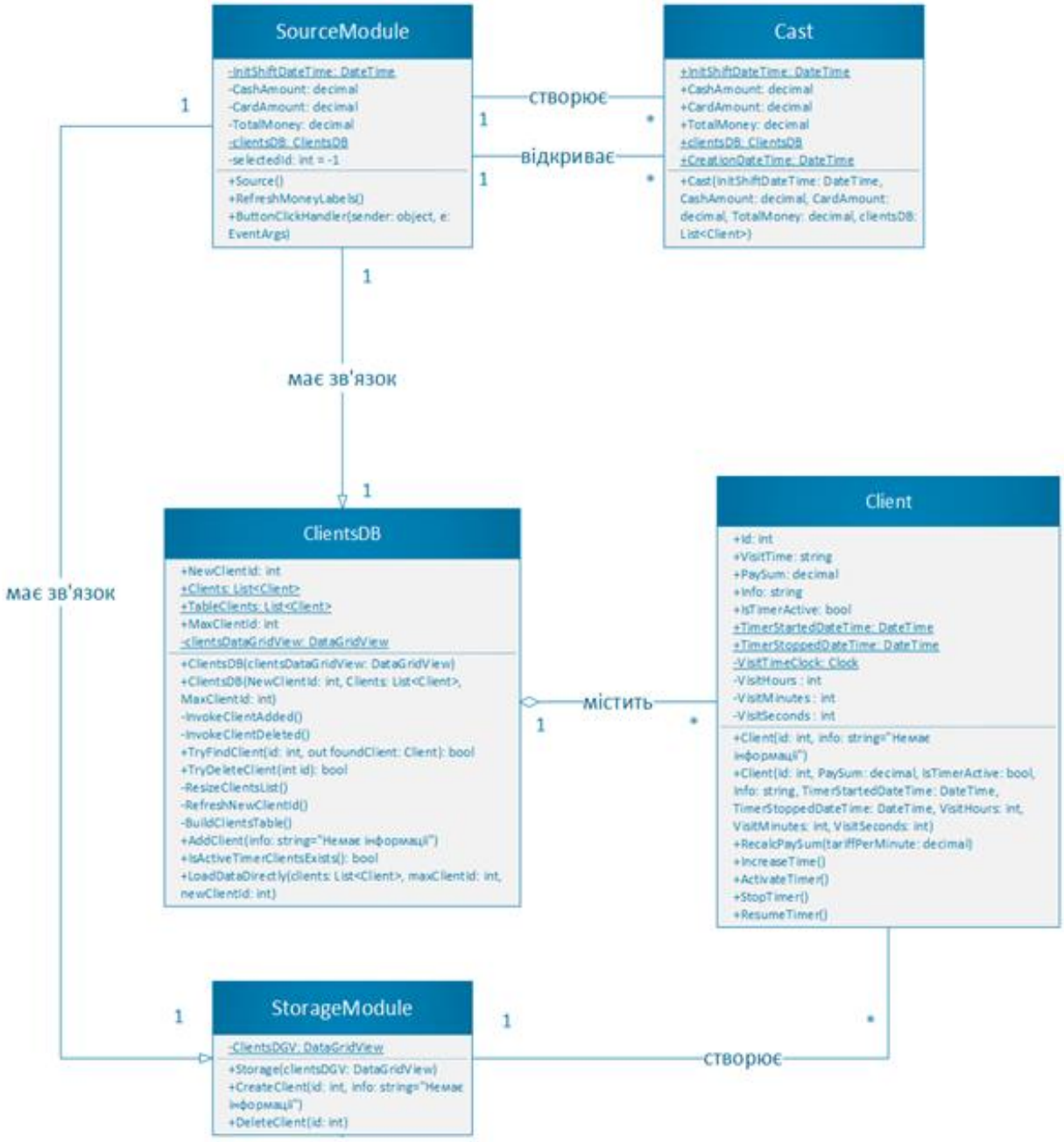


Рисунок2.4.3. «Взаємозв’язки між класами»

Діаграма пакетів

Оскільки пакети відображають структуру ПЗ проекту, використаємо діаграму пакетів.

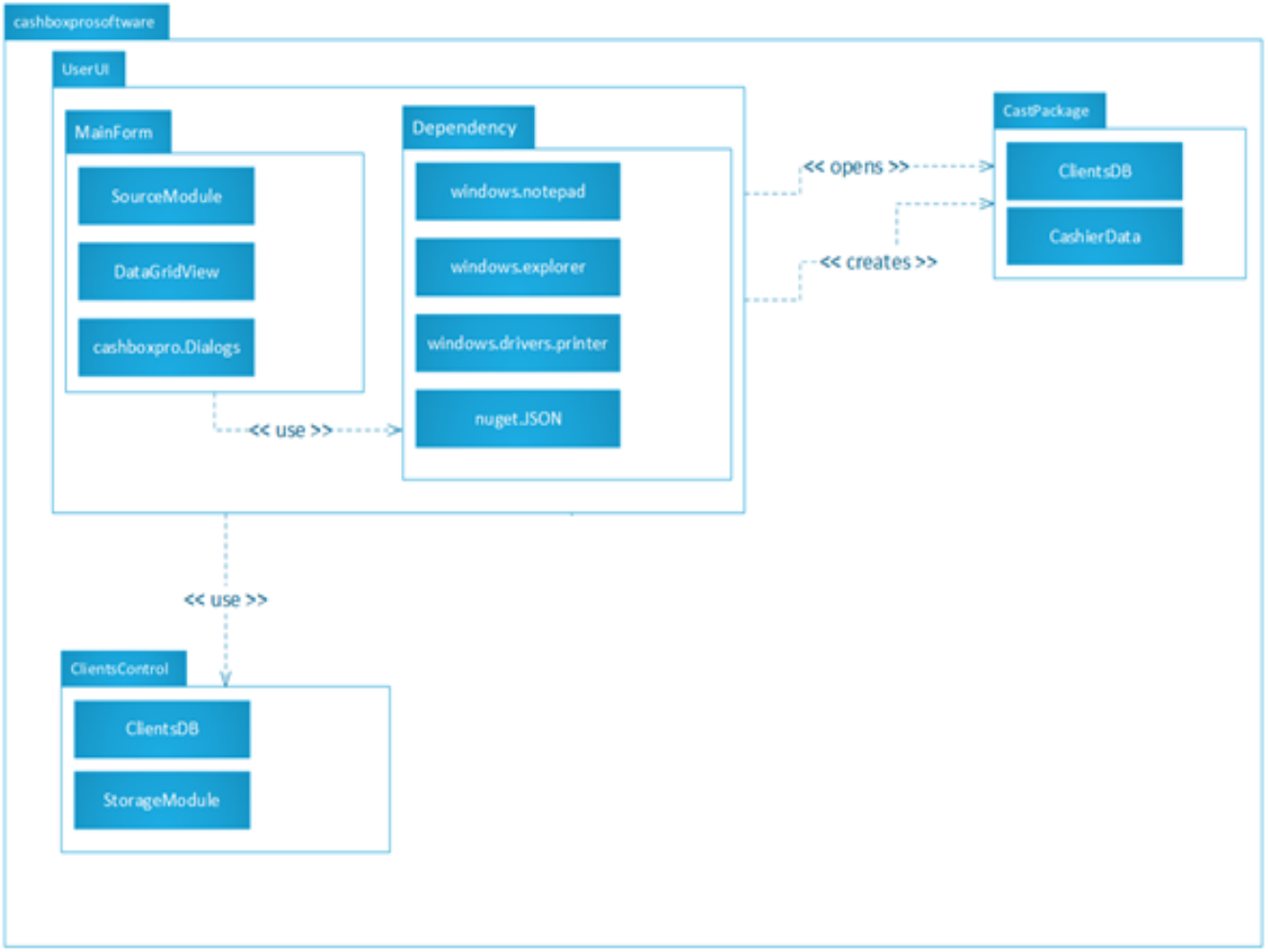


Рисунок2.4.4.»Діаграма пакетів»

Пакет cashboxprosoftware ПЗ “CashBox” є головним пакетом який містить у собі усі пакети програми.

Пакет cashboxprosoftware поділяється на три основні пакети (UserUI, ClientsControl, MainDatabaseControl) та один пакет який створюється та відкривається ПЗ під час роботи програми (CastPackage).

UserUI – є головним підпакетом та містить у собі пакети MainForm та Dependency. MainForm вміщує у собі інтерфейсі засоби та класи для взаємодії користувача із ПЗ. Також цей пакет використовує пакет Dependency**,** який включає у себе усі необхідні бібліотеки для роботи ПЗ. Ці пакети описані у таблиці нижче:

Пакет	Застосування
windows.notepad	Пакет-бібліотека який дозволяє використовувати вбудовану програму Windows – блокнот.
window.explorer	Пакет-бібліотека який дозволяє використовувати вбудовану програму Windows – провідник.
windows.drivers.printer	Пакет-бібліотека який дозволяє використовувати вбудований драйвер друку Windows.
nuget.JSON	Пакет-бібліотека який дозволяє ПЗ створювати та відкривати файли зліпків, які є файлом типу JSON.

Таблиця пакетів

Пакет ClientsControl містить у собі клас ClientsDB та клас-модуль StorageModule. Цей пакет дозволяє керувати Тимчасовою БД клієнтів.

Клас-бібліотеку SQL Adapter, який може взаємодіяти з БД транзакцій та змін, яка є базою даних MySQL та пакет-бібліотеку microsoft.excel, який дозволяє створювати файли типу Excel.

Пакет CastPackage створюється під час роботи ПЗ та використовується як файл зліпку для відновлення робочої зміни. Пакет містить у собі пакети даних БД клієнтів (ClientsDB) та дані каси (CashierData). Під час відкриття пакету ці дані експортуються у ПЗ і таким чином відновлюється робоча зміна.

2.4.4.Діаграми станів ,розгортання.

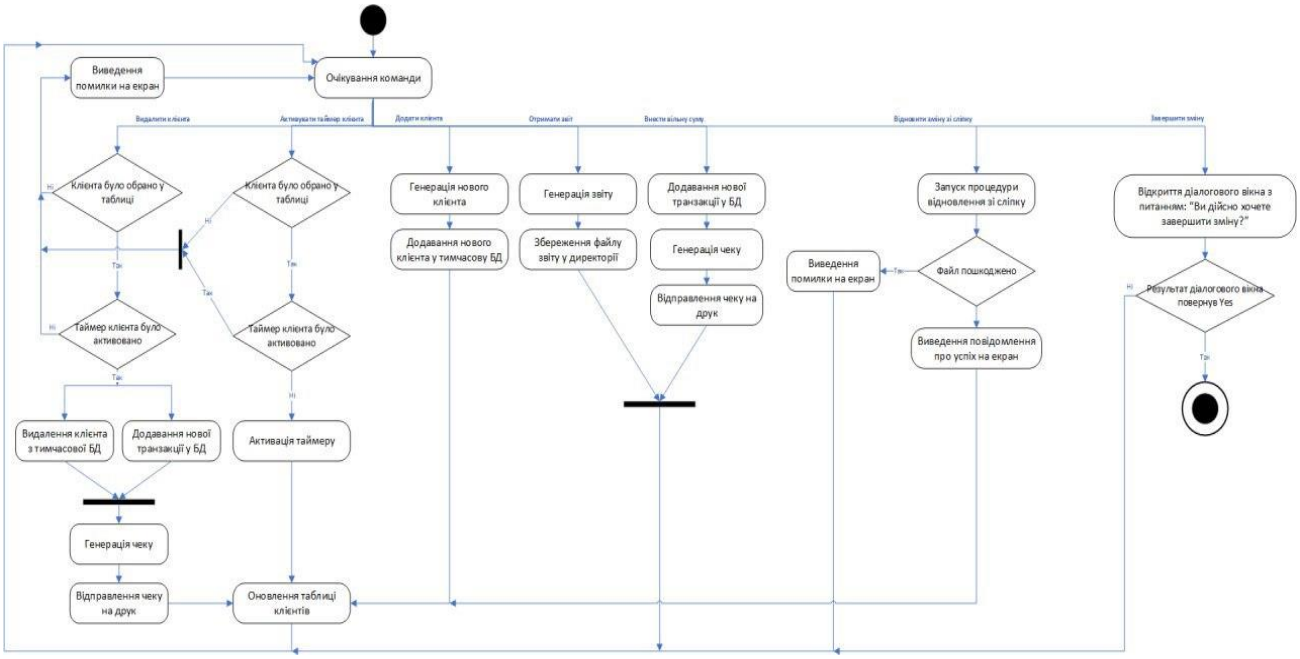


Рисунок 2.4.5.Діаграма станів(Додаток Б)

2.4.5.Розгортання

Оскільки програмне забезпечення буде функціонувати на терміналі (персональний комп'ютер на базі ОС Windows) та використовує різні компоненти та користується системою, створює та зчитує файли, відсилаєзапит на друк, тощо. Важливо побачити як саме програмний продукт будерозгортатися на пристрої

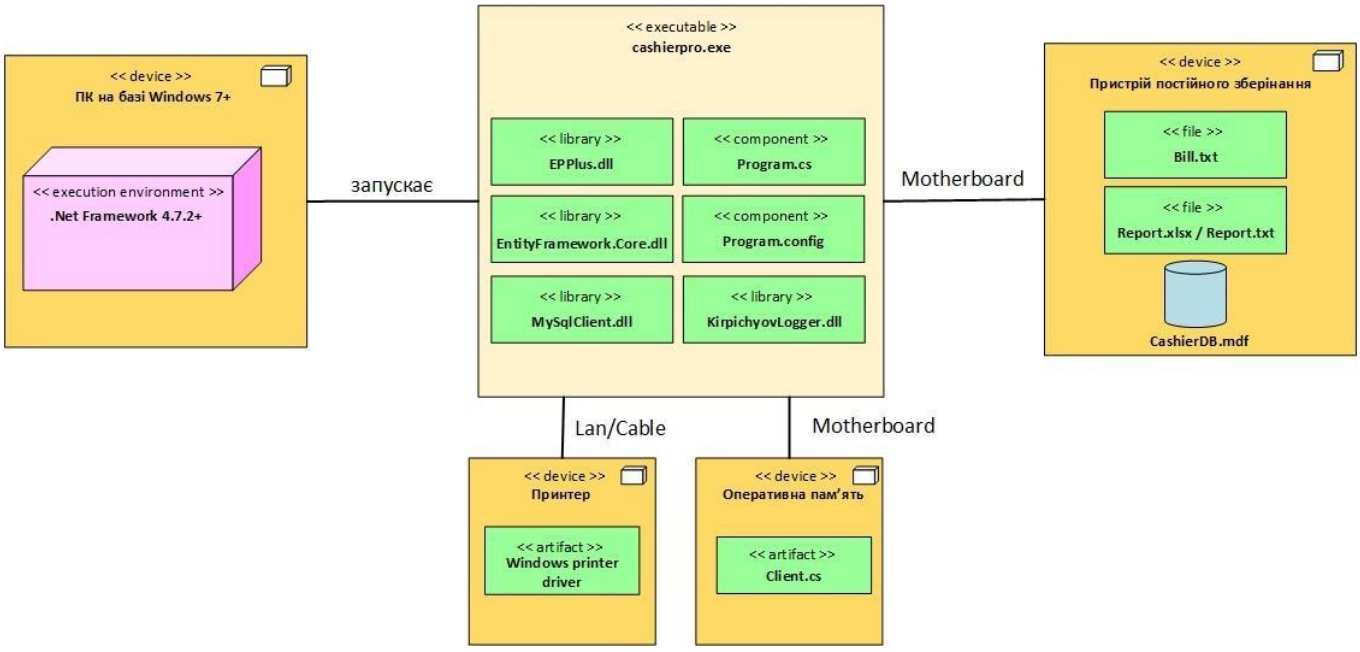


Рисунок 2.4.6 «діаграма розгортання»

2.5.1.Компоненти

Програмний модуль використовує багато компонентів як своїх класів та структур, так і бібліотек, створених іншими розробниками. Отже, щоб побачити основні компоненти програмного продукту, було побудовано діаграму компонентів.

Рисунок 2.5.1. Діаграма компонентів (частина 1)(Додаток Г)

Рисунок 2.5.2. Діаграма компонентів (частина 2)(Додаток Г)

3.РЕАЛІЗАЦІЯ ІНФОРМІЦІЙНОЇ СИСТЕМИ

3.1.Простори імен

У проекті використовується розподіл класів за простором імен, оскільки функціонал чітко розділений. Основними просторами імен є:

1. **CashierPro**

   Кореневий простір імен, що включає в себе весь функціонал та програмні елементи.

2. **CashierPro.CashierSoftware**

   Включає в себе функціонал, що стосується програмних модулів роботикаси.

3. **CashierPro.StartupSoftware**

   Включає в себе функціонал, що стосується програмних модуліввідповідаючих за стартовий екран програмного забезпечення.

4. **CashierPro.DataModels**

   Включає в себе функціонал, що стосується програмних модуліввідповідаючих за взаємодію із БД та її моделями даних.

5. **CashierPro.MainForms**

   Включає в себе головні форми (вікна) програмного продукту, через які відбувається взаємодія із основними функціями ПЗ.

6. **CashierPro.Properties**

   Включає в себе функціонал, що стосується програмних модулів відповідаючих за збереження та зміну користувацьких налаштувань .

7. **System.Windows.Forms**

   Простір імен, що включає в себе усі компоненти, які забезпечуютьстворення та взаємодію із вікнами програмного продукту.

3.2.Робота з бд

В якості сховища статистичних даних про оплати та робочих змін використовується легка локальна база даних SQLite у зв’язці із потужним ADO фреймворком для роботи з даними БД як з об’єктами об’єктно-орієнтованого програмування Entity Framework Core.

Для створення БД використовується підхід Code First – коли БД генерується на основі класів даних, що будуть зберігатися у ній. Для зв’язку із БД використовується спеціальний клас, який називається контекстом бази даних. Шлях до БД прописується у файлі конфігурації проекту, або встановлюється під час створення об’єкту контексту.

Потужні реляційні бази родини SQL підтримують підхід Code First, але не SQLite. Тому у проекті для генерації БД, якщо вона ще не створена, використовуються класи простору імен System.Data.SQLite SQLiteCommand та SQLiteConnection. За допомогою цих класів, ми виконуємо SQL запит на створення бази даних. Код запиту вбудовано в збірку проекту:

```
CREATE TABLE "PayTypes" ( "Id" INTEGER NOT NULL
PRIMARY KEYAUTOINCREMENT UNIQUE, "Label" TEXT NOT
NULL, "Sum" REAL);

CREATE TABLE "Payments" ("Id" INTEGER NOT NULL
PRIMARY KEY AUTOINCREMENT UNIQUE, "Method" INTEGER
NOT NULL, "Type" TEXT NOT NULL,"Sum" REAL NOT NULL,
"InitTime" TEXT NOT NULL);
CREATE TABLE "Shifts" (
"Id"  INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT
UNIQUE,"Clients"  INTEGER NOT NULL,
"FreePayments" INTEGER NOT NULL, "OperatorName"  TEXT NOT NULL,
"InitTime"TEXT NOT NULL,
"FinishTime" TEXT NOT NULL, "TotalIncome"
REAL NOT NULL,"CashIncome" REAL NOT NULL,
"CashlessIncome"  REAL NOT NULL,"MinClientSum"  REAL NOT NULL,
```

```
"MaxClientSum"          REAL NOT NULL,"AvgClientSum"    REAL NOT NULL,
"MinFreeSum"            REAL NOT NULL,"MaxFreeSum"      REAL NOT NULL,
"AvgFreeSum"            REAL NOT NULL,
    "MinClientTime" TEXT NOT NULL,"MaxClientTime" TEXT NOT NULL,"AvgClientTime"
TEXT NOT NULL
    );
```

Доступ до даних, що зберігаються у БД здійснюється через відповіднуданим властивість, яка містить у собі об’єкт Generic класу DbSet.

```
public DbSet<Payment> Payments { get; set; }public DbSet<Shift> Shifts { get; set; }
public DbSet<PayType> PayTypes { get; set; }
```

Під час створення об’єкту контексту здійснюється перевірка наіснування бази даних та будується шлях до файлу БД. При створенні БД, у неї додаються дані, які повинні знаходитися у сховище за замовчуванням.

```
public AppDatabaseContext()
{
dbFilename = Path.Combine(Environment.CurrentDirectory,
"CashierDB.db");connectionString = $"Data
source={dbFilename};Version=3;"; CreateDataBase();
}

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
optionsBuilder.UseSqlite("Filename=CashierDB.db;");
}

private void CreateDataBase()

if (!File.Exists(dbFilename))
{
using (var connection = new SQLiteConnection(connectionString))
```

```
  {
  string commandText = Properties.Resources.SQLCreateTablesQuery;
SQLiteCommand Command = new SQLiteCommand(commandText,
connection);connection.Open();
  Command.ExecuteNonQuery();
connection.Close();
  }
  PayType[] DefaultPayTypes = new PayType[]
  {
  new PayType("Кава", 14.50m),
  new PayType("Оренда на день", 450),
  new PayType("Оренда на тиждень", 2600),
new PayType("Оренда на місяць", 9000),
  };
  foreach (var type in
DefaultPayTypes)
PayTypes.Add(type);
  SaveChanges();
  }
  File.SetAttributes(dbFilename, File.GetAttributes(dbFilename) |
FileAttributes.Hidden);
        }
```

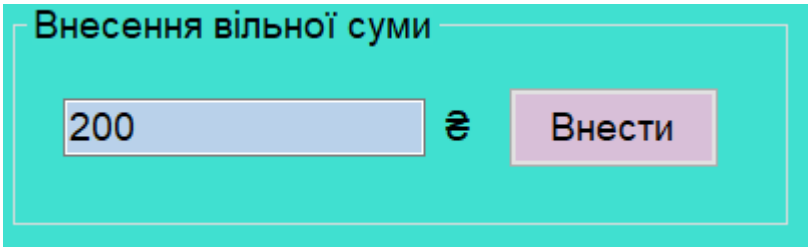


Рисунок 3.2.1 «Внесення вільної суми»

Взаємодія із даними у БД забезпечується спеціальними методами, які дозволяють розробити свій API-подібний функціонал для взаємодії із сховищем даних.

```
public void Drop()
{
if (File.Exists(dbFilename))
{
File.Delete(dbFilename);
CreateDataBase();
}
}

#region PayTypes control
public PayType[] GetPayTypes() => PayTypes.ToArray();

public decimal GetPayTypeSum(string label)
{
foreach (var payType in PayTypes)
if (payType.Label == label) return payType.Sum;
        throw new ArgumentException($"PayType with {nameof(label)}
equal \"{label}\"doesn't exists.", nameof(label));
}

public void AddPayType(string label, decimal sum)
{
PayTypes.Add(new PayType(label, sum));
}

public void DeletePayType(int payTypeid)
```

```
{
var payType = PayTypes.FirstOrDefault(pt => pt.Id ==
payTypeid);if (payType != null)
{
PayTypes.Remove(PayTypes.First(pt => pt.Id == payTypeid));
}
        else throw new ArgumentException($"PayType with Id equal
\"{payTypeid}\" doesn'texists.", nameof(payTypeid));
}

  public void UpdatePayType(int payTypeid, string label, decimal sum)
  {
  var payType = PayTypes.FirstOrDefault(pt => pt.Id ==
  payTypeid);if (payType != null)
  {
  var toEdit = PayTypes.First(pt => pt.Id ==
  payTypeid);toEdit.Label = label;
  toEdit.Sum = sum;
  }
          else throw new ArgumentException($"PayType with Id equal
  \"{payTypeid}\" doesn'texists.", nameof(payTypeid));
  }
  #endregion

  #region ShiftControl
  public void AddShift(Shift shift)
  {
  Shifts.Add(shift);
  }
  #endregion
```

```
#region PaymentsControl
public async void AddPaymentAsync(Payment payment)
{
await Payments.AddAsync(payment);
}
#endregion
```

Варто звернути увагу, що у коді активно використовується Language Integrated .

Після обробки даних через контекст, усі дані трансформуються у звичайні знайомі об’єкти. Класи, що представляють дані розробленого програмного забезпечення – моделі даних:

Payment. Представляє дані про оплату.

```
namespace CashierPro.DataModels
{
class Payment
{
public int Id { get; set; }
public PayMethod Method { get; set; }
public string Type { get; set; } public
decimal Sum { get; set; } public
DateTime InitTime { get; set; }
public string MethodLabel => CashierProExtensions.GetDescription(Method);


public Payment(PayMethod payMethod, string type, decimal sum, DateTime initTime)
        {
Method = payMethod;
Type = type;
```

```
Sum = sum;
InitTime = initTime;
}
public Payment() { }
}
}
```

PayType. Представляє тип даних, що описує варіант для вільної оплати. Варіанти для вільної оплати можна обирати під час здійснення вільної суми, таким чином наповнюється кошик покупок, після чого здійснюється оплата.

```
namespace CashierPro.DataModels
{
public class PayType
{
public int Id { get; set;}
public string Label { get; set; }
public decimal Sum { get; set; }
public PayType(string label, decimal sum)
{
Label = label;Sum =
sum;
}
public PayType() { }
}
}
```

Shift. Представляє дані про робочу зміну. Використовується як модель даних для БД, а також як модель для створення зліпку робочої зміни – тому має в своєму складі атрибути XmlIgnore та Serializable.

```
namespace CashierPro.DataModels
```

```
{
[Serializable]
public class Shift
{
[XmlIgnore]
public int Id { get; set; }
public int Clients { get; set; } public int
FreePayments { get; set; }
public string OperatorName { get; set; }
public DateTime InitTime { get; set; }
public DateTime FinishTime { get; set; }
[XmlIgnore]
public decimal TotalIncome
{set { }
}
public decimal CashIncome { get; set; }
public decimal CashlessIncome { get; set; }
public decimal MinClientSum { get; set; }
public decimal MaxClientSum { get; set; }
public decimal AvgClientSum { get; set; }
public decimal MinFreeSum { get; set; }
public decimal MaxFreeSum { get; set; }
public decimal AvgFreeSum { get; set; }
[XmlIgnore]
public TimeSpan MinClientTime { get; set;
}[XmlIgnore]
public TimeSpan MaxClientTime { get; set;
}[XmlIgnore]
public TimeSpan AvgClientTime { get; set; }
public string MinClientTimeString => MinClientTime.ToTotalTimeString();
```

```
[XmlIgnore]
[NotMapped]
public string MaxClientTimeString => MaxClientTime.ToTotalTimeString();
[XmlIgnore]
[NotMapped]
public string AvgClientTimeString => AvgClientTime.ToTotalTimeString();
[NotMapped]
public double MaxClientTotalSeconds
{
get { return MaxClientTime.TotalSeconds; }
set { MaxClientTime = TimeSpan.FromSeconds(value); }
}
[NotMapped]
public double MinClientTotalSeconds
{
get { return MinClientTime.TotalSeconds; }
set { MinClientTime = TimeSpan.FromSeconds(value); }
}
[NotMapped]
public double AvgClientTotalSeconds
{
get { return AvgClientTime.TotalSeconds; }
set { AvgClientTime = TimeSpan.FromSeconds(value); }
}
public Shift() { }
```

}}Фрагмент коду: додавання нового запису про робочу зміну у БД після закінчення робочої зміни оператора терміналу-каси.

```
using(var db = new AppDatabaseContext())
{
db.AddShift(cashbox.Shift)
```

```
;db.SaveChanges();
}
```

Оскільки клас контекст реалізує інтерфейс IDisposable, ми використовуємо конструкцію using для автоматичного звільнення ресурсів, що були задіяні.

3.3.Перевизначення messageboxbuttons

Діалогові вікна класу MessageBox мають діалогові кнопки ОК, так, ні, відміна, тощо. Особливістю цих кнопок є те, що мова на якій написано текст на цих кнопках залежить від налаштувань системи, тому для того, щоб текст завжди відображався українською, потрібно використати бібліотеку MessageBoxManager, проініцілізувати її при старті програми та налаштувати текст на кнопках вручну. Після підключення бібліотеки, у Program.cs в методіMain було додано:

```
MessageBoxManager.No = "Ні";
MessageBoxManager.Yes =
"Так";
MessageBoxManager.Cancel =
"Відміна";
MessageBoxManager.Register();
```

3.4.Запобігання запуску більше ніж одного екземпляру програми

Оскільки програмне забезпечення розраховано на одного активного користувача, важливо створити механізм запобігання запуску інших екземплярів програмного забезпечення. Для цього було введено перевірку наіснуючий в системі процес перед запуском основних модулів ПЗ в метод Main у Program.cs:

```
if
```

```
(System.Diagnostics.Process.GetProcessesByName(System.Diagnostics.Process
.GetCurrentProc ess().ProcessName).Length > 1)
{MessageBox.Show("Cashier Pro вже запущено! Одночасно запустити можна лише
один додаток!","Cashier Pro помічник", MessageBoxButtons.OK,
MessageBoxIcon.Error); Application.Exit();
return;
}
```

3.6.Налаштування програми

Усі налаштування у вікні налаштування розділені за видами налаштувань, тому для розділення й навігації використовується клас для користувацького інтерфейсу TabControl, який дозволяє створити вкладки як у браузері, для навігації між типами налаштувань (Рис 3.6.1).

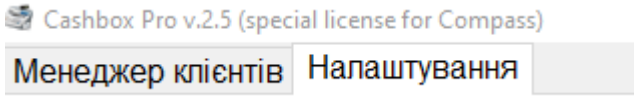


Рисунок 3.6.1 «Навігація між налаштуваннями»

Для налаштування параметрів Каси використовуються елементи TextBox таComboBox, а для підтвердження виконується обробка натискання кнопки зберігання.

Для забезпечення перегляду, додавання, видалення та редагування типів оплати використовується компонент DataGridView, поля TextBox та кнопки Button (Рисунок 3.6.1). Перед виконанням CRUD операцій виконується перевірка на правильність заповнення полів та статус вибраного елемента.

```
private void RefreshPayTypesDGV()
{
payTypesDGV.DataSource = null;
  using (var db = new
```

```
AppDatabaseContext())LoadedPayTypes
= db.GetPayTypes();
payTypesDGV.DataSource = LoadedPayTypes;


payTypesDGV.AutoResizeRows(DataGridViewAutoSizeRowsMode.AllCell
s);
payTypesDGV.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.
AllCells);payTypesDGV.ClearSelection();
PayTypeLabelTextBox.Text =
string.Empty;PayTypeSumTextBox.Text
= string.Empty;
}
private void AddPayTypeButton_Click(object sender, EventArgs e)
{
if (string.IsNullOrWhiteSpace(PayTypeLabelTextBox.Text))
{
        MessageBox.Show("Поле введення назви оплати не може
бути порожнім!","Cashier Pro помічник",
  MessageBoxButtons.OK, MessageBoxIcon.Error);
return;
}


      PayTypeSumTextBox.Text = PayTypeSumTextBox.Text.Replace('.',
',').Replace(" ",string.Empty).Trim();
if (TryParseFreeSumValue(PayTypeSumTextBox.Text, out
decimal sum,out FreeSumParseFailReason parseFailReason))
{
DialogResult dialogResult = MessageBox.Show($"Ви дійсно хочете додати новий
            тип оплати \"{PayTypeLabelTextBox.Text}\"",
"Cashier Pro помічник",MessageBoxButtons.YesNo,
```

```
MessageBoxIcon.Question);
if (dialogResult == DialogResult.Yes)
{
using (var db = new AppDatabaseContext())
{
db.AddPayType(PayTypeLabelTextBox.Text, sum);
db.SaveChanges();
}
PayTypeLabelTextBox.Text = PayTypeSumTextBox.Text =
string.Empty;RefreshPayTypesDGV();
}
}
if (parseFailReason == FreeSumParseFailReason.NonDecimal)
{
MessageBox.Show("Поле введення суми має містити лише числові
значення!","Cashier Pro помічник",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
else if (parseFailReason == FreeSumParseFailReason.OutOfRange)
{
MessageBox.Show("Значення суми повинно бути більшим за 0 та
меншим за 1000 000!", "Cashier Pro помічник",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
private void DeletePayTypeButton_Click(object sender, EventArgs e)
{
if (payTypesDGV.SelectedCells.Count > 0)
{
DialogResult dialogResult = MessageBox.Show($"Ви дійсно хочете видалити тип
```

```
оплати
\"{payTypesDGV.SelectedCells[0].OwningRow.Cells["LabelColumn"].Valu
e.ToString()}\"","Cashier Pro помічник",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
if (dialogResult == DialogResult.Yes)
{
using (var db = new AppDatabaseContext())
{
db.DeletePayType(int.Parse(payTypesDGV.SelectedCells[0].OwningRow.Cells
["IdColumn"].Value.ToString()));
db.SaveChanges();
}
RefreshPayTypesDGV();
}
}
else MessageBox.Show("Спочатку оберіть тип оплати", "Cashier Pro
помічник",MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
private void EditPayTypeButton_Click(object sender, EventArgs e)
{
if (payTypesDGV.SelectedCells.Count > 0)
{
DialogResult dialogResult = MessageBox.Show($"Ви дійсно хочете редагувати тип
оплати
\"{payTypesDGV.SelectedCells[0].OwningRow.Cells["LabelColumn"].Valu
e.ToString()}\"","Cashier Pro помічник",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
if (dialogResult == DialogResult.Yes)
{
if (string.IsNullOrWhiteSpace(PayTypeLabelTextBox.Text))
```

```
{
MessageBox.Show("Поле введення назви оплати не може бути
порожнім!","Cashier Pro помічник",
MessageBoxButtons.OK, MessageBoxIcon.Error);
return;
}
PayTypeSumTextBox.Text = PayTypeSumTextBox.Text.Replace('.',
',').Replace("", string.Empty).Trim();
if (TryParseFreeSumValue(PayTypeSumTextBox.Text, out decimal
sum,out FreeSumParseFailReason parseFailReason))
{
using (var db = new AppDatabaseContext())
{
db.UpdatePayType(int.Parse(payTypesDGV.SelectedCells[0].OwningRow.Cells
["IdColumn"].Value.ToString()),
PayTypeLabelTextBox.Text, decimal.Parse(PayTypeSumTextBox.Text));
db.SaveChanges();
}
RefreshPayTypesDGV();
}
if (parseFailReason == FreeSumParseFailReason.NonDecimal)
{
MessageBox.Show("Поле введення суми має містити лише числові
значення!", "Cashier Pro помічник",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
else if (parseFailReason == FreeSumParseFailReason.OutOfRange)
{
MessageBox.Show("Значення суми повинно бути більшим за 0 та меншим
за1 000 000!", "Cashier Pro помічник",
```

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

else MessageBox.Show("Спочатку оберіть тип оплати", "Cashier Pro помічник",MessageBoxButtons.OK, MessageBoxIcon.Warning);

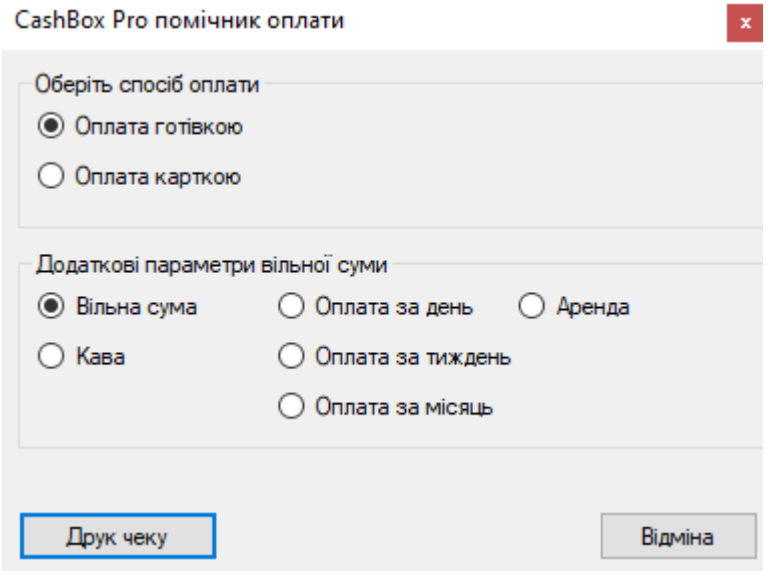


Рисунок 3.6.3. Налаштування (Типи оплати)

Для обирання місця зберігання вихідних файлів програмного забезпечення (чеки, зліпки та звіти) використовується клас FolderBrowserDialog.

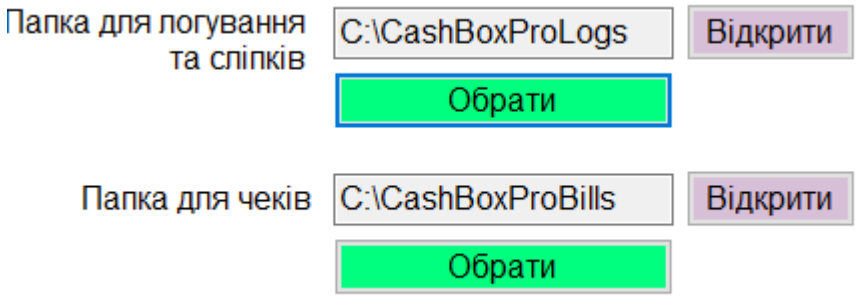


Рисунок 3.6.4 Налаштування (Сховище)

private void BillFolderOpenButton_Click(object sender, EventArgs e)

{

Process.Start("explorer", AppConfig.BillFolder);

```
}
private void ReportFolderOpenButton_Click(object sender, EventArgs e)
{
Process.Start("explorer", AppConfig.ReportFolder);
}
private void CastFolderOpenButton_Click(object sender, EventArgs e)
{
Process.Start("explorer", AppConfig.CastFolder);
}
private void BillFolderEditButton_Click(object sender, EventArgs e)
{
using (var fbd = new FolderBrowserDialog())
{
DialogResult result = fbd.ShowDialog();
if (result == DialogResult.OK && !string.IsNullOrWhiteSpace(fbd.SelectedPath))
{
if(IsCanUseFolder(fbd.SelectedPath))
{
AppConfig.BillFolder = fbd.SelectedPath;
AppConfig.ApplyChanges();
MessageBox.Show("Успішно змінено!", "Cashier Pro помічник",
MessageBoxButtons.OK, MessageBoxIcon.Information);
RefreshConfigUI();
}
MessageBox.Show("Обрана папка є системною та не може бути використана!
Оберіть іншу папку!", "Cashier Pro помічник",
MessageBoxButtons.OK,MessageBoxIcon.Warning);
}
}
}
```

```
private void ReportFolderEditButton_Click(object sender, EventArgs e)
{
using (var fbd = new FolderBrowserDialog())
{
DialogResult result = fbd.ShowDialog();
if (result == DialogResult.OK && !string.IsNullOrWhiteSpace(fbd.SelectedPath))
{
if (result == DialogResult.OK && !string.IsNullOrWhiteSpace(fbd.SelectedPath))
{
if (IsCanUseFolder(fbd.SelectedPath))
{

AppConfig.ReportFolder = fbd.SelectedPath;
AppConfig.ApplyChanges();
MessageBox.Show("Успішно змінено!", "Cashier Pro помічник",
MessageBoxButtons.OK, MessageBoxIcon.Information);
RefreshConfigUI();
}

MessageBox.Show("Обрана папка є системною та не може бути використана!
Оберіть іншу папку!", "Cashier Pro помічник",
MessageBoxButtons.OK,MessageBoxIcon.Warning);
}
}
}
}
private void CastFolderEditButton_Click(object sender, EventArgs e)
{
using (var fbd = new FolderBrowserDialog())
{
```

```
DialogResult result = fbd.ShowDialog();
if (result == DialogResult.OK && !string.IsNullOrWhiteSpace(fbd.SelectedPath))
{
if (IsCanUseFolder(fbd.SelectedPath))
{
AppConfig.CastFolder = fbd.SelectedPath;
AppConfig.ApplyChanges();
MessageBox.Show("Успішно змінено!", "Cashier Pro помічник",
MessageBoxButtons.OK, MessageBoxIcon.Information);
RefreshConfigUI();
}
MessageBox.Show("Обрана папка є системною та не може бути використана!
Оберіть іншу папку!", "Cashier Pro помічник",
MessageBoxButtons.OK,MessageBoxIcon.Warning);
}
}
}
```

Для забезпечення логування системи використовується бібліотека CompassLib. Бібліотека формує логи у файл XML. Сформовані логи, містять повідомлення та тип логу. На основі структури LogType, можна вивести інформацію про події у системі, які потрібно відстежувати. Для відображення логів системи було використано ListView. У залежності від типу логу, таблицянаповнюється записом із відповідним кольором. Пошук логів забезпечується за допомогою ComboBox. Залежно від обраного варіанту відображаються логи у таблиці. Для пошуку використовуються засоби LINQ

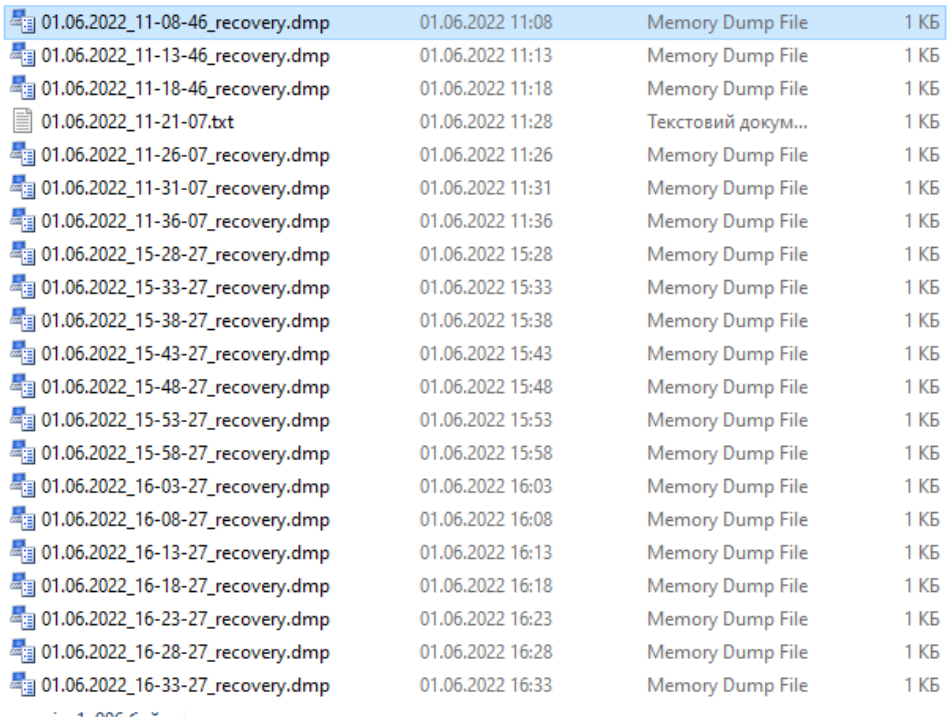


Рисунок 3.6.5 Папка з Логуванням

```
private void FillLogsView(LogType logType = LogType.Debug)
{
logsView.Items.Clear();
IEnumerable<LogRecord>
logRecords;
logRecords = Logs.Where(l => l.Type == logType);

foreach (var log in logRecords)
{
string[] row = { log.CreationTime.ToString(), log.Type.ToString(),
log.Text };var listViewItem = new ListViewItem(row);
listViewItem.BackColor = Color.Black;
switch (log.Type)
{
case LogType.Info:
listViewItem.ForeColor = Color.Blue;
```

```
break;
case LogType.Error:
listViewItem.ForeColor = Color.OrangeRed;
break;
case LogType.Warning:
listViewItem.ForeColor = Color.Yellow;
break;
case LogType.Debug:
listViewItem.ForeColor = Color.White;
break;
case LogType.Critical:
listViewItem.ForeColor = Color.Red;
break;
}
logsView.Items.Add(listViewItem);
}
}

private void FillLogsView()
{
logsView.Items.Clear();
foreach (var log in Logs)
{
string[] row = { log.CreationTime.ToString(), log.Type.ToString(),
log.Text };var listViewItem = new ListViewItem(row);
listViewItem.BackColor = Color.Black;
switch (log.Type)
{
        case LogType.Info: listViewItem.ForeColor = Color.Blue;break;
case LogType.Error:
```

```
listViewItem.ForeColor = Color.OrangeRed;
break;
        case LogType.Warning: listViewItem.ForeColor = Color.Yellow;break;
        case LogType.Debug: listViewItem.ForeColor = Color.White;break;
        case LogType.Critical:
listViewItem.ForeColor = Color.Red;
break;
}
logsView.Items.Add(listViewItem);
}
}
private void findLogsButton_Click(object sender, EventArgs e)
{
FillLogsView((LogType)logsTypeCombobox.SelectedItem);
}
private void showAllLogsButton_Click(object sender, EventArgs e)
{
FillLogsView();
        }
```

3.7.Програмне забезпечення каси

Програмне забезпечення каси представляє з себе головне вікно (Рисунок 2.3) ізтаблицею клієнтів (DataGridView), інформаційної панелі виручки та панелі керування (Panel), кнопок керування (Button) та поля для інформації клієнтів (RichTextBox).

У даному програмному модулі постійно використовується основний клас Cashier, який дозволяє налаштувати взаємодії із усіма компонентами каси, а також зберігає у собі та підраховує статистичні дані (клас Shift).

Головне вікно активно використовує два класи, які представляють вікно розрахування суми клієнта за проведений час (ClientPayDialog) та вікно вільної суми (FreeSumPayDialog). Ці класи успадковані від класу Form.

Для валідації інформації у додаткових вікнах, використовується компонент ErrorProvider, що дозволяє відображати помилки валідації користувачу.

```
namespace CashierPro.CashierSoftware

{

public enum PayMethod

{

[Description("Готівка")]
Cash = 0,
[Description("Безготівка")]
Cashless = 1
public enum ClientTimerState

{

Start, Stop, Resume

}

public class Cashier

{

public List<Client> Clients { get; set;
}public Shift Shift { get; set; }
public event EventHandler ClientAdded;
public event EventHandler ClientDeleted;
public event EventHandler
ClientInfoChanged;public event
EventHandler IncomeReceived;
```

```
public const string DefaultClientInfo = "Інформації
немає";private int releasedClientsCount = 0;
private int releasedFreeSumCount = 0;


public Cashier(string operatorName)
{
Clients = new List<Client>();
Shift = new Shift { OperatorName = operatorName, InitTime = DateTime.Now };
}
public Client this[int clientId] => Clients.First(c => c.Id == clientId);
public int GetNewClientId()
{
int curNewId = 1;
if (Clients.Count == 0) return 1;
for (int i = 0; i < Clients.Count; i++)
{
if (Clients[i].Id == curNewId)
curNewId++;
else return curNewId;
}
return curNewId;
}
public void AddNewClient(string info = DefaultClientInfo)
{
```

```
Clients.Add(new Client(GetNewClientId(), info));
Clients = Clients.OrderBy(c => c.Id).ToList();
ClientAdded?.Invoke(this, EventArgs.Empty);
}


public bool TryDeleteClient(int id)
{
if(Clients.Any(c => c.Id == id))
{
Client toDelete = Clients.First(c => c.Id == id);
Clients.Remove(toDelete);
ClientDeleted?.Invoke(this, EventArgs.Empty);
releasedClientsCount++;
CalcClientsShiftInfo(toDelete);
return true;
}
return false;
}
public bool TryChangeClientInfo(int id, string newInfo)
{
if (Clients.Any(c => c.Id == id))
{
Clients.First(c => c.Id == id).Info = newInfo;
ClientInfoChanged?.Invoke(this, EventArgs.Empty);
return true;
```

```
}
return false;
}

public bool TrySetTimerState(int id, ClientTimerState timerState)
{
if (Clients.Any(c => c.Id == id))
{
switch (timerState)
{
case ClientTimerState.Start:
Clients.First(c => c.Id == id).Timer.Start();break;
case ClientTimerState.Stop:
Clients.First(c => c.Id == id).Timer.Stop();break;
case ClientTimerState.Resume:
Clients.First(c => c.Id == id).Timer.Resume();break;
}
return true;
}
return false;
}
public void AddIncome(decimal income, PayMethod payMethod, bool freeIncome)
{
```

```
if (payMethod == income;
else if (payMethod == PayMethod.Cashless)
Shift.CashlessIncome += income;
if (freeIncome)
releasedFreeSumCount++;
CalcFreePaymentsShiftInfo(income);
}
IncomeReceived?.Invoke(this, EventArgs.Empty);
}

private void CalcClientsShiftInfo(Client releasedClient)
{
//load
decimal minClientSum = Shift.MinClientSum;
decimal maxClientSum = Shift.MaxClientSum;
decimal avgClientSum = Shift.AvgClientSum;
double minClientTimeSeconds =
Shift.MinClientTime.TotalSeconds; double
maxClientTimeSeconds = Shift.MaxClientTime.TotalSeconds;
double avgClientTimeSeconds =
Shift.AvgClientTime.TotalSeconds;

//calculate
if (releasedClientsCount == 1)
{
minClientSum = maxClientSum = avgClientSum =
```

```
releasedClient.SumToPay;minClientTimeSeconds =
maxClientTimeSeconds =
releasedClient.Timer.TotalSeconds;

avgClientTimeSeconds = releasedClient.Timer.TotalSeconds;

}

else

{

minClientSum = minClientSum > releasedClient.SumToPay ?
releasedClient.SumToPay : minClientSum;

maxClientSum = maxClientSum < releasedClient.SumToPay ?
releasedClient.SumToPay : maxClientSum;

decimal totalSum = avgClientSum * (releasedClientsCount - 1);

avgClientSum = (totalSum + releasedClient.SumToPay) / releasedClientsCount;



minClientTimeSeconds = minClientTimeSeconds >
releasedClient.Timer.TotalSeconds ?
releasedClient.Timer.TotalSeconds :minClientTimeSeconds;

maxClientTimeSeconds = maxClientTimeSeconds <
releasedClient.Timer.TotalSeconds ?
releasedClient.Timer.TotalSeconds :
maxClientTimeSeconds;
double totalTimeSeconds = avgClientTimeSeconds * (releasedClientsCount -
1); avgClientTimeSeconds = (totalTimeSeconds +
releasedClient.Timer.TotalSeconds)
/ releasedClientsCount;
```

```
}


//save
Shift.Clients = releasedClientsCount;
Shift.MinClientSum = minClientSum;
Shift.MaxClientSum = maxClientSum;
Shift.AvgClientSum = avgClientSum;
Shift.MinClientTime =
TimeSpan.FromSeconds(minClientTimeSeconds);
Shift.MaxClientTime =
TimeSpan.FromSeconds(maxClientTimeSeconds);
Shift.AvgClientTime =
TimeSpan.FromSeconds(avgClientTimeSeconds);
}


private void CalcFreePaymentsShiftInfo(decimal income)
{
decimal minFreeSum = Shift.MinFreeSum;
decimal maxFreeSum = Shift.MaxFreeSum;
decimal avgFreeSum = Shift.AvgFreeSum;
if (releasedFreeSumCount == 1)
{
minFreeSum = maxFreeSum = avgFreeSum = income;
}
else
```

```
{
minFreeSum = minFreeSum > income ? income : minFreeSum;
maxFreeSum = maxFreeSum < income ? income : maxFreeSum;
decimal totalSum = avgFreeSum * (releasedFreeSumCount - 1);
avgFreeSum = (totalSum + income) / releasedFreeSumCount;
}


Shift.FreePayments = releasedFreeSumCount;
Shift.MinFreeSum = minFreeSum;
Shift.MaxFreeSum = maxFreeSum;
Shift.AvgFreeSum = avgFreeSum;
}
public void DevShowClients()
{
if (Clients.Count == 0)
{
System.Diagnostics.Trace.WriteLine("No clients");
return;
}
System.Diagnostics.Trace.WriteLine(" ----------------");
foreach (var c in Clients)
{
System.Diagnostics.Trace.WriteLine(
$"{c.Id}|{c.Info}");
}
```

}

}

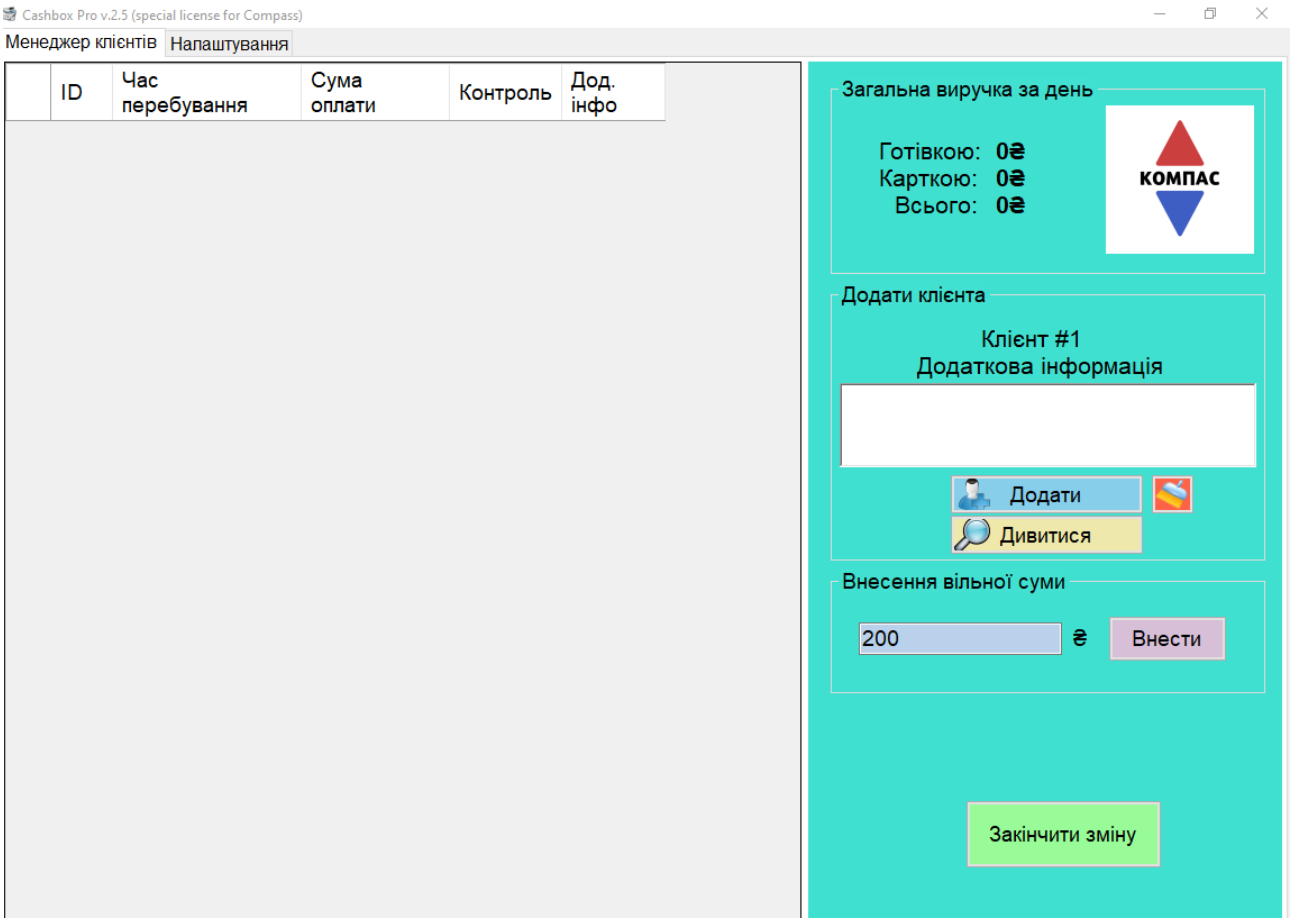


Рисунок 3.7.1 Головне вікно ПЗ каси

3.8.Клас-розширення

Деякі функції у програмному модулі є загальними та не відносяться до якогось певного функціоналу, або мають загальні типи даних, об’єкти й перерахування, а деякий функціонал взагалі потребує модифікування стандартний засобів об’єктно-орієнтованої мови програмування C#. Оскільки C# дозволяє створювати статичні класи та розширювати вже створений функціонал, було створено клас розширення.

namespace CashierPro

```
{public static class CashierProExtensions
{
public static Logger AppLogger;
public enum FreeSumParseFailReason { None, NonDecimal, OutOfRange }
static CashierProExtensions()
{
AppLogger = new Logger();
}
public static bool TryParseFreeSumValue(string sumString, out
decimal sum,out FreeSumParseFailReason parseFailReason)
{
parseFailReason = FreeSumParseFailReason.None;
sumString = sumString.Replace('.', ',').Replace(" ", string.Empty).Trim();
if (decimal.TryParse(sumString, out sum))
{
        if (sum > 0 && sum < 1000000)return
true;
 else
{
parseFailReason = FreeSumParseFailReason.OutOfRange;
return false;
}
}
else
{
```

```
parseFailReason = FreeSumParseFailReason.NonDecimal;
return false;
}
}



public static string[] ForbiddenFolders = new string[]
{
"Program Files", "Program Files (x86)",
"Windows", "WindowsApps", "System32",
"Users", "Program Data"
};


public static bool IsCanUseFolder(string folderpath)
{
for(int i=0; i<ForbiddenFolders.Length; i++)
{
                if(folderpath.Contains(ForbiddenFolders[i]))return
false;
}
return true;
}
```

```
            public static string ToTotalTimeString(this TimeSpan t) =>
                $"{((int)(t.TotalHours)).ToString().PadLeft(2, '0')}:" +
  $"{(t.Minutes).ToString().PadLeft(2, '0')}" +
  $":{(t.Seconds).ToString().PadLeft(2, '0')}";


public static int GetTotalSeconds(this TimeSpan t)
  =>(t.Seconds +
 (t.Minutes * 60) + (t.Hours * 3600));


  public static string GetDescription<T>
  (this T enumerationValue)
   where T : struct
  {
  var type = enumerationValue.GetType();
  if (!type.IsEnum)
  {
                throw new ArgumentException($"{nameof(enumerationValue)}
  must be of Enumtype", nameof(enumerationValue));
  }
  var memberInfo =
  type.GetMember(enumerationValue.ToString());if
  (memberInfo.Length > 0)
  {
  var                    attrs  =
```

```
                memberInfo[0].GetCustomAttributes(typeof(DescriptionAttribute),
false);
if (attrs.Length > 0)
{
        return ((DescriptionAttribute)attrs[0]).Description;
}
}
return enumerationValue.ToString();
}
}
}
```

3.9.Створення та читання зліпку

Для створення механізму зберігання робочої зміни та її відновлення із файлу зліпку було розроблено клас CastManager та допоміжні класи контейнери для серіалізації ClientSerializable та Cast. Для створення зліпку використовується механізм серіалізації, а для читання – десеріалізації. Файл зліпку представляє із себе файл XML, який легко та швидко зчитується засобами мови програмування C#. Щоб запобігти підробці файлу поза програмним забезпеченням, використовується механізм шифрування RSA (вкладений клас Encryptor).

```
namespace CashierPro.CashierSoftware
{
public static class CastManager
```

```
{
static public string CreateCast(Client[] clients , Shift shift)
{
        string xmlFilename =
$"{AppConfig.CastFolder}cast_{DateTime.Now.ToString("dd-MM-yyyy_h-mm-
ss")}.cst";
List<ClientSerializable> xmlClients = new List<ClientSerializable>();foreach (var
client in clients)
{
xmlClients.Add(new ClientSerializable()
{
Id = client.Id,
Hours = client.Timer.Hours, Minutes = client.Timer.Minutes,Seconds =
client.Timer.Seconds,Info = client.Info,
IsEnabled = client.Timer.IsEnabled, StartDateTime = client.Timer.StartDateTime
});
}
Cast cast = new Cast(xmlClients.ToArray(), shift); XmlSerializer formatter = new
XmlSerializer(typeof(Cast));

using (FileStream fs = new FileStream(xmlFilename, FileMode.Create))
{
formatter.Serialize(fs, cast);
}
string fileContent = File.ReadAllText(xmlFilename);
```

```
        File.WriteAllText(xmlFilename, Encryptor.Encrypt(fileContent,
"Compass_acidsnake_the_creator"));
return xmlFilename;
static public bool TryReadCast(string path, Cashier cashier)
{
try
{
string fileContent = File.ReadAllText(path); File.WriteAllText(path,
Encryptor.Decrypt(fileContent,
"Compass_acidsnake_the_creator"))

using (FileStream fs = new FileStream(path, FileMode.Open))
{
XmlSerializer formatter = new XmlSerializer(typeof(Cast));var loadedCast =
(Cast)formatter.Deserialize(fs);

cashier.Clients = new List<Client>(); foreach (var client in loadedCast.Clients)
{
cashier.Clients.Add(new Client
{
Id = client.Id, Info = client.Info,
                Timer = new ClientTimer(client.Hours, client.Minutes, client.Seconds,
client.IsEnabled, client.StartDateTime)
});
}
```

```
cashier.Shift = loadedCast.Shift;

}

fileContent = File.ReadAllText(path);

            File.WriteAllText(path, Encryptor.Encrypt(fileContent,
"Compass_acidsnake_the_creator"));

catch (Exception e)

{

File.AppendAllText("error.log", $"\terror occured
  {DateTime.Now.ToString()}\n{e.ToString()}\n\n");return false;
}

return true;

}

static internal void CreateDevCopy(string path1, string path2)

{
string fileContent = File.ReadAllText(path1); File.WriteAllText(path2,
Encryptor.Decrypt(fileContent,
"Compass_acidsnake_the_creator"));

}
[Serializable]

public class ClientSerializable

{

public int Id { get; set; } public int Hours { get; set; } public int Minutes { get; set; }
public int Seconds { get; set; }
public bool IsEnabled { get; set; }public string Info { get; set; }
public DateTime? StartDateTime { get; set; }
```

```
public ClientSerializable() { }

}

[Serializable] public class Cast

{

public ClientSerializable[] Clients { get; set; }public Shift Shift { get; set; }

public Cast() { }

public Cast(ClientSerializable[] clients, Shift shift)

{

Clients = clients;

Shift = shift;

}

}

internal static class Encryptor

{

private const string InitVector = "T=A4rAzu94ez-dra";private const int KeySize =
256;
private const int PasswordIterations = 1000;

        private const string SaltValue =
"d=?ustAF=UstenAr3B@pRu8=ner5sW&h59_Xe9P2za-eFr2fa&ePHE@ras!a+uc@";
public static string Decrypt(string encryptedText, string passPhrase)

{

byte[] encryptedTextBytes = Convert.FromBase64String(encryptedText);byte[]
initVectorBytes = Encoding.UTF8.GetBytes(InitVector);
byte[] passwordBytes = Encoding.UTF8.GetBytes(passPhrase);string plainText;
```

```
byte[] saltValueBytes = Encoding.UTF8.GetBytes(SaltValue);
        Rfc2898DeriveBytes password = new Rfc2898DeriveBytes(passwordBytes,
saltValueBytes, PasswordIterations);
byte[] keyBytes = password.GetBytes(KeySize / 8);

        RijndaelManaged rijndaelManaged = new RijndaelManaged { Mode =
CipherMode.CBC };
try
{
            using (ICryptoTransform decryptor =
rijndaelManaged.CreateDecryptor(keyBytes,initVectorBytes))
{
                using (MemoryStream memoryStream = new
MemoryStream(encryptedTextBytes))
{
                    using (CryptoStream cryptoStream = new
CryptoStream(memoryStream,decryptor, CryptoStreamMode.Read))
{
                        //TODO: Need to look into this more. Assuming encrypted text is
longerthan plain but there is probably a better way
byte[] plainTextBytes = new byte[encryptedTextBytes.Length];
                        int decryptedByteCount = cryptoStream.Read(plainTextBytes, 0,
plainTextBytes.Length);
   plainText = Encoding.UTF8.GetString(plainTextBytes, 0,decryptedByteCount);
}
```

```
}

}

}

catch (CryptographicException)

{

plainText = string.Empty; // Assume the error is caused by an invalid password

}

return plainText;

}



public static string Encrypt(string plainText, string passPhrase)

{

string encryptedText;

byte[] initVectorBytes = Encoding.UTF8.GetBytes(InitVector);byte[] passwordBytes
= Encoding.UTF8.GetBytes(passPhrase);byte[] plainTextBytes =
Encoding.UTF8.GetBytes(plainText); byte[] saltValueBytes =
Encoding.UTF8.GetBytes(SaltValue);

   Rfc2898DeriveBytes password = new Rfc2898DeriveBytes(passwordBytes,
saltValueBytes, PasswordIterations);

byte[] keyBytes = password.GetBytes(KeySize / 8);
RijndaelManaged rijndaelManaged = new RijndaelManaged { Mode =
CipherMode.CBC };
        using (ICryptoTransform encryptor =
rijndaelManaged.CreateEncryptor(keyBytes,initVectorBytes))
```

```
{
using (MemoryStream memoryStream = new MemoryStream())
{
    using (CryptoStream cryptoStream = new CryptoStream(memoryStream, encryptor, CryptoStreamMode.Write))
{
cryptoStream.Write(plainTextBytes, 0, plainTextBytes.Length);
cryptoStream.FlushFinalBlock();
byte[] cipherTextBytes = memoryStream.ToArray(); encryptedText = Convert.ToBase64String(cipherTextBytes);
}
return encryptedText;
```

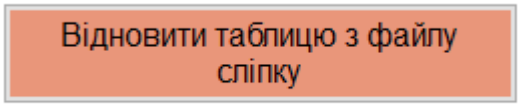


Рисунок 3.9.1 Колонка відновлення змін

ВИСНОВКИ

Дана робота була спрямована на проектування та реалізацію інструменту для автоматизації роботи обліку коворкінгів..

1. Обґрунтовано актуальність розробленого продукту та його наукову новизну шляхом аналізу інших схожих додатків. Було досліджено типові потреби користувачів інтернет-ресурсів для комунікації та виявлено, що аудиторія потребує інструментів, які можуть спростити пошук людей по схожим інтересам.
2. Розроблено розділену архітектуру додатку з використанням клієнт серверну архітектуру, оскільки це обумовлено стандартами розробки веб-додатків. Обрано інструменти для побудови продукту, обґрунтовано перевагу та ефективність над іншими аналогами.

Для того щоб розробити успішний продукт, було проаналізовано потреби користувача при роботі з аналогічними додатками та створено набір вимог у вигляді UML діаграми.

В ході розробки системи було реалізовано можливість обрахування коштів , відновлення зміни , зберігання чеків.

3. Описано програмні засоби та інструменти, котрі було застосовано для розробки програмного забезпечення. Виявлено що зручним середовищем розробки є VS. Підібрано допоміжні інструменти розробки

Відстеження статусу виконання проекту було за допомогою інструменту Jira.

4. Відповідність системи визначеним вимогам забезпечує набір тестових сценаріїв, котрий покриває всі функціональні вимоги системи, з урахуванням граничних випадків, та перевірки їх обробки системою.
5. Створений продукт цілеспрямований на допомогу у обрахуванні обліку коштів коворкінгу. Визначено, що інструмент може використовуватись для, розвитку особистого бренду.
6. Роботу було апробовано серед реальних користувачів. Було зібрано статистику використання, проаналізовано та виявлено ряд переваг та недоліків системи. Також було розроблено план розвитку продукту. Виявлено, що

основними перевагами програми є:

– простота використання; ;

– різноманітність для використання;

– стабільність роботи основних функцій; – легкий для розуміння інтерфейс.

Результати дослідження бакалаврської роботи апробована на IV Міжнародній студентській науковій конференції «Наука сьогодення: від досліджень до стратегічних рішень» (м. Івано-Франківськ, Україна).

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Pro C# 7: With .Net and .Net Core - Andy Pole, Mike West, Andrew Troelsen& Philip Japikse Apress – 2017.: P.52-670
2. Microsoft Visual C# Step by Step (9th Edition) (Developer Reference) - John Sharp O'Riley – 2018.: P.100-325
3. Windows Forms Programming in C# - Chris Sells Addison Wesley PC – 2003.: P.21-402
4. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition) - Craig Larman PrenticeHall – 2003.: P.100-201
5. Повне керівництво по мові програмування C # 8.0 і платформі .NET Core [Електронний ресурс]. - Режим доступу:htmlhttps://metanit.com/sharp/tutorial
6. Керівництво з програмування в Windows Forms. [Електронний ресурс]. - Режим доступу: https://metanit.com/sharp/windowsforms
7. C# documentation. [Електронний ресурс]. - Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp
8. https://www.quality-assurance-group.com/requirement-types/
9. A4 Company [Електронний ресурс]: BAS Управління торгівлею – Режим доступу: https://a4.com.ua/ru/bas-upravlenie-torgovlej/
10. Документація Microsoft [Електронний ресурс]: Boolean logical operators (C# reference) – Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/boolean-logical-operators
11. Документація Microsoft [Електронний ресурс]: Using domain analysis to model microservices – Режим доступу: https://docs.microsoft.com/uk-ua/azure/architecture/microservices/model/domain-analysis

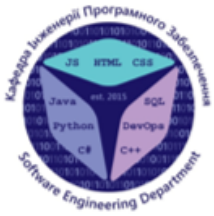
ЕКРАННІ ФОРМИ

ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Розробка програмного забезпечення для касових апаратів коворкінгів мовою C#

Виконав студент 4 курсу
Групи ПД-43:
Матківський Аскольд Олегович
Керівник роботи:
к.т.н., доц. Трінтіна Наталя Альбертівна.

Київ – 2022

Аналоги

Розглянуті програми	Communa coworking	SERVIO HMS
Платформи	Windows всі версії	Windows всі версії
Складність	складний	складний
Доступність	недоступний фріланс 100 $	980 грн
Зручність інтерфейсу	-	+
українська мова	+	-
мова програмування	JS	JS
Гнучкіть	-	+
система знижок	-	+
Стійкість до відутності інтернету	-	-
подальші витрати	Оренда сервера	Оренда сервера

Мета, об’єкт та предмет дослідження

Мета роботи:розробити програмний модуль, який автоматизує та прискорить бізнес- процеси підприємства.

- **Об’єкт дослідження:** процеси обліку для терміналу-каси антикафе "Compass"
- **Предмет дослідження:** програмні засоби для роботи з касовими апаратами.

Технічне завдання

1. Створити механізм обрахування коштів , які мають заплатити користувачі антикафе.
2.Створити можливість зміни тарифу коворкінга.
3.Зробити механізм звітів , за кожну завершену зміну.
4. Розробити механізм відновлення зміни.
5.Розробити можливість внесення довільної суми , за різні послуги(оренда кімнати коворкінгу ,робочого місця , тощо)

Використані інструменти

Мова програмування C#

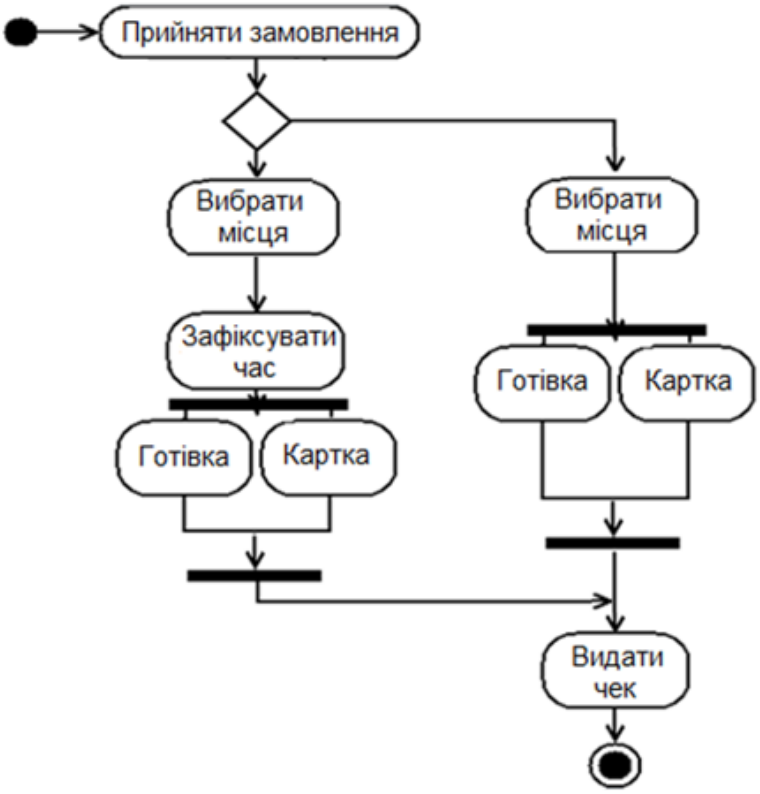
Середовище розробки Visual Studio

Платформа .NET із прикладним інтерфейсом Windows Forms

Система керування базами даних SQLite

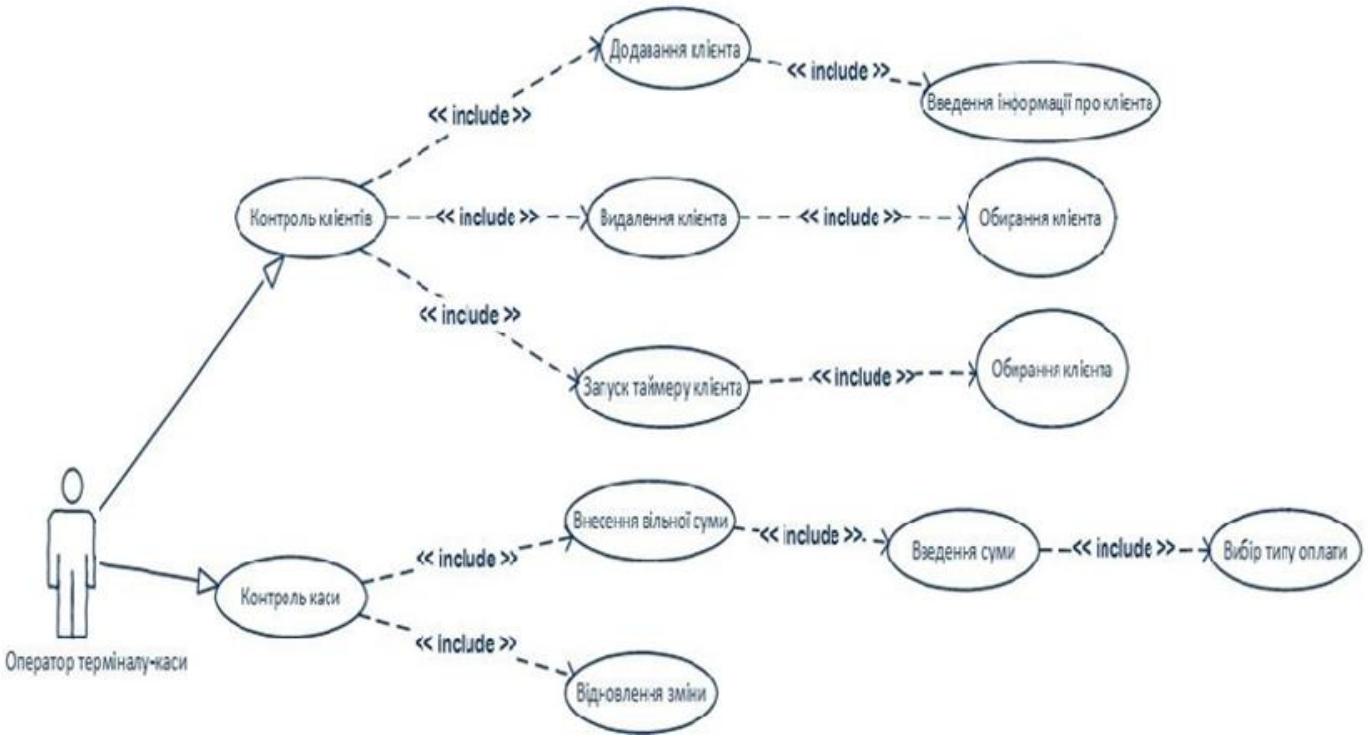
5

Діаграма діяльності роботи коворкінгу



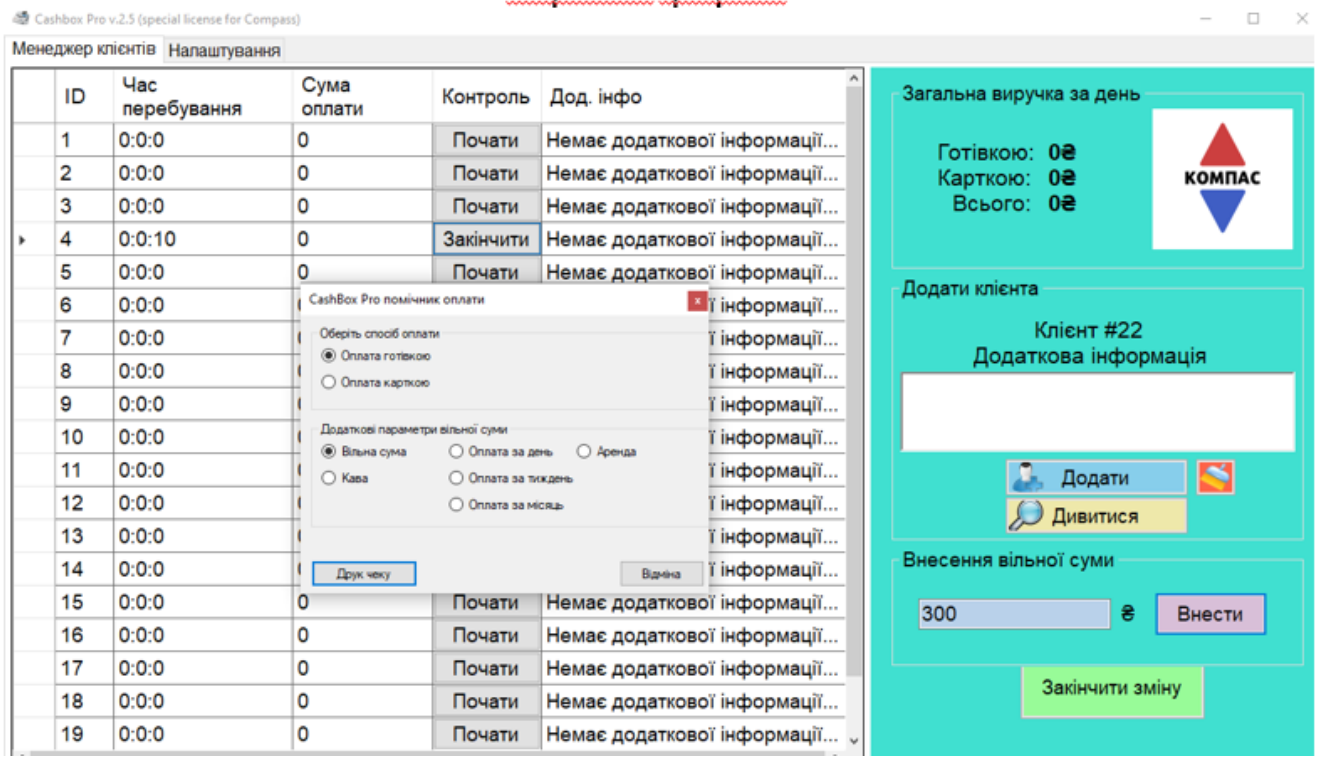
6

Діаграма прецедентів



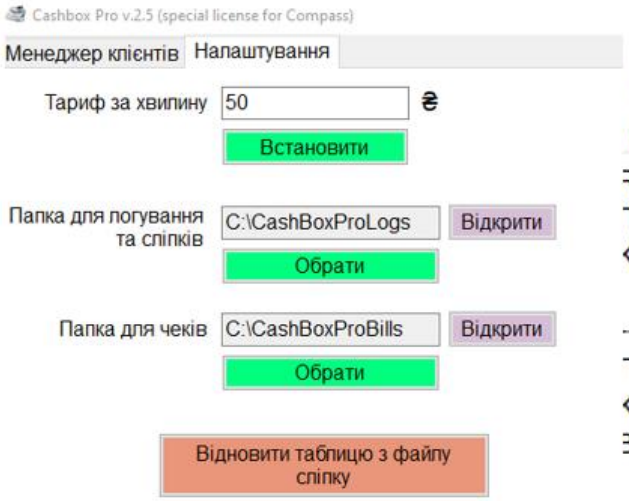
7

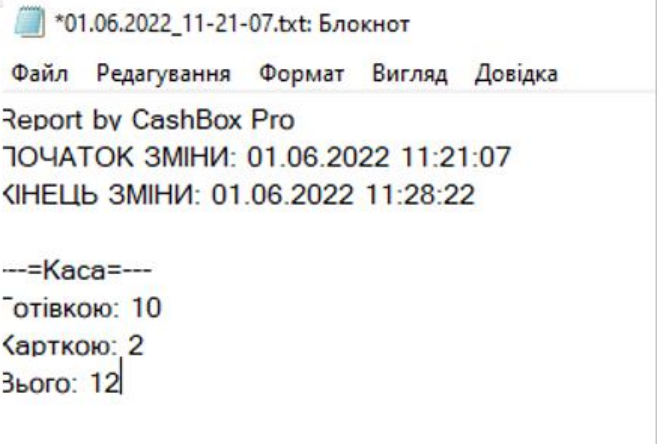
Екранні форми



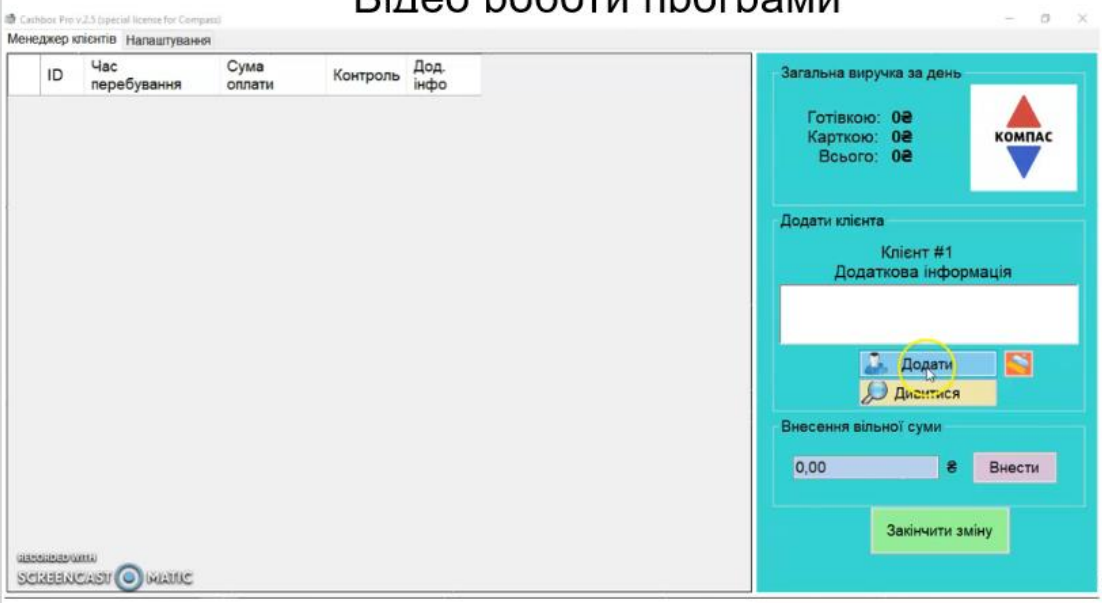
8

Екранні форми





Відео роботи програми



10

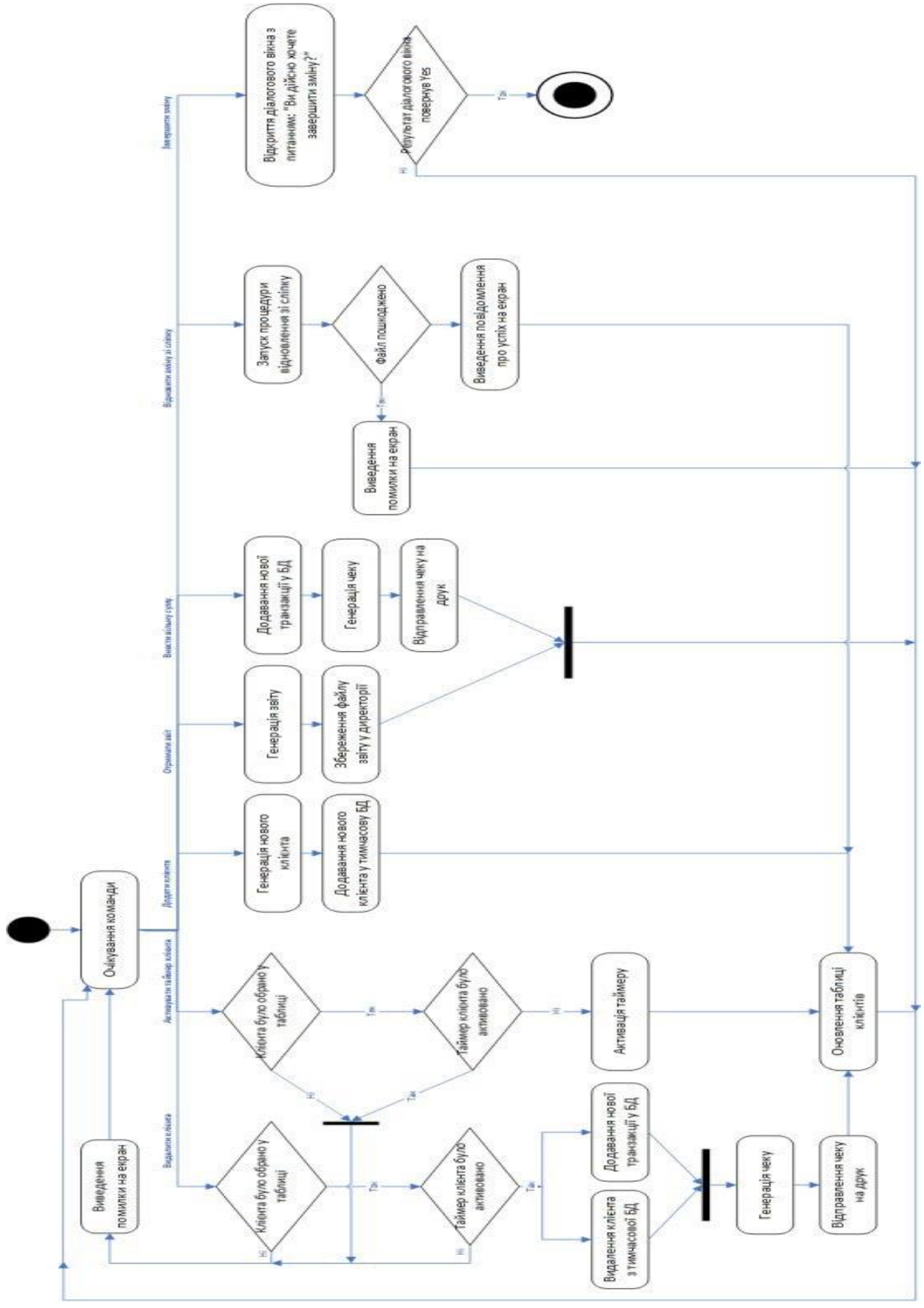
Висновки

1. Створено механізм обрахування коштів , які мають заплатити користувачі антикафе.
2. Створено можливість зміни тарифу коворкінга.
3. Зроблений механізм звітів , за кожну завершену зміну.
4. Розроблений механізм відновлення зміни.
5. Розроблено можливість внесення довільної суми

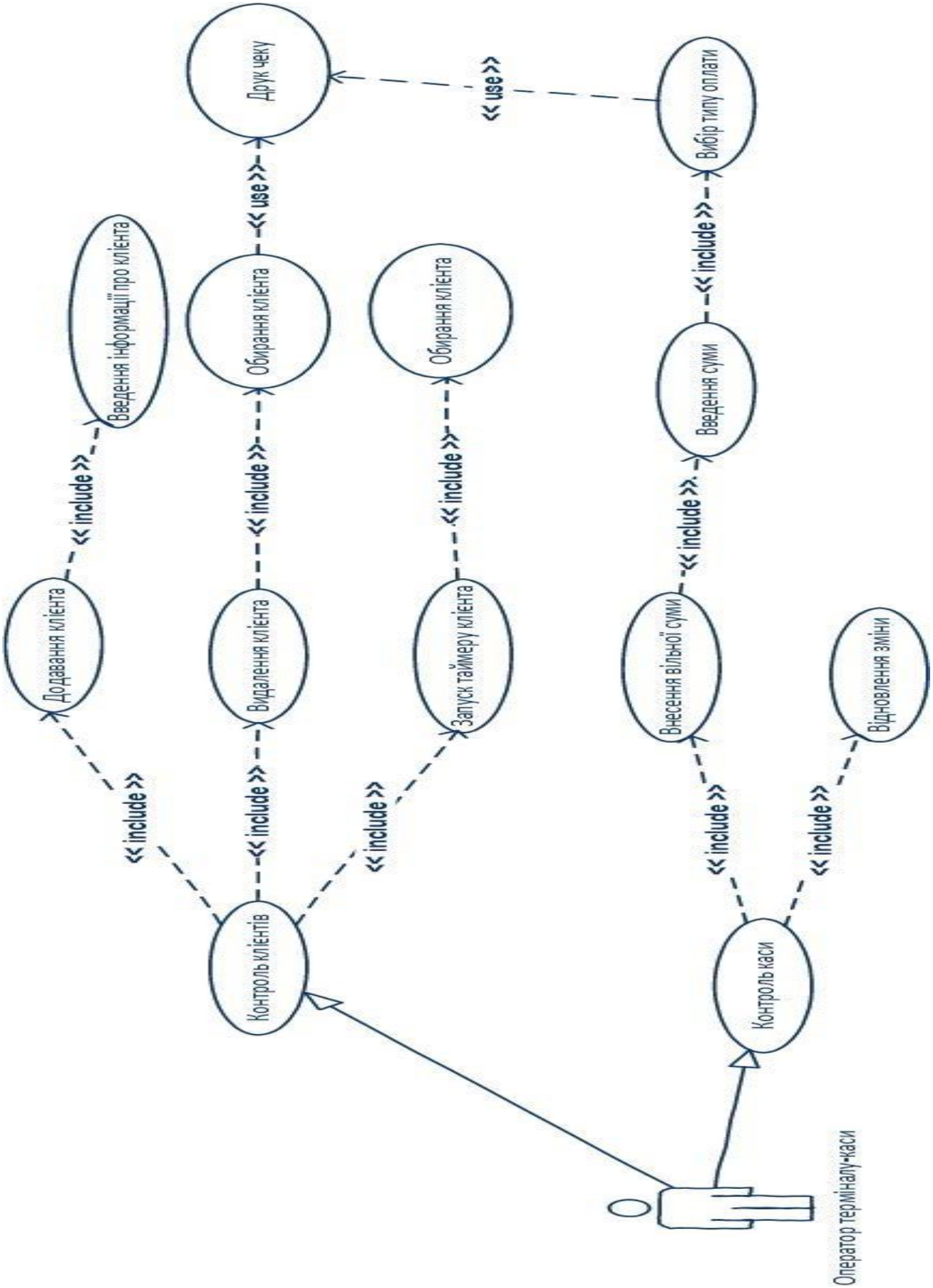
11

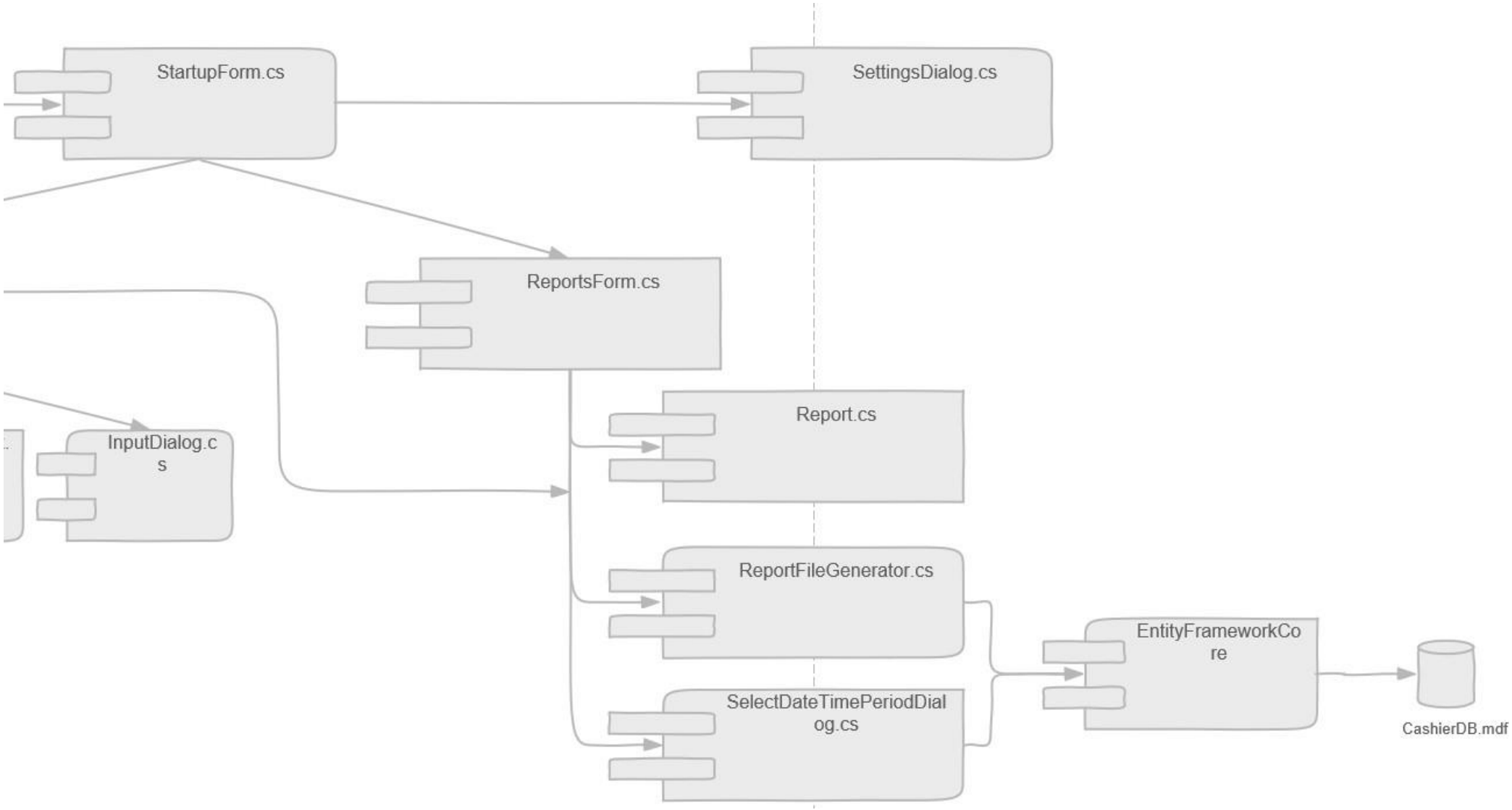
ДЯКУЮ ЗА УВАГУ!

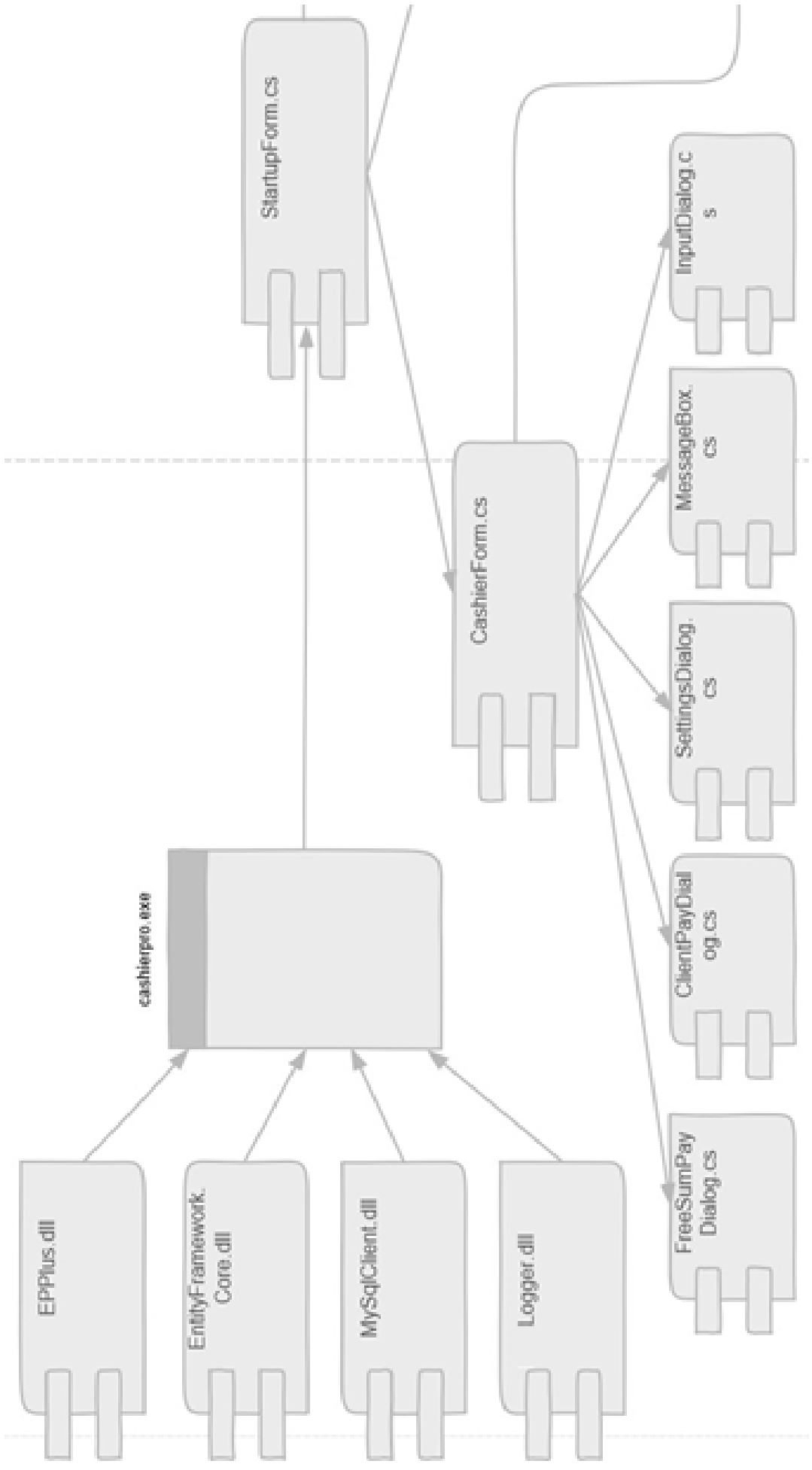
ДОДАТОК Б


Очікування команди
Завершити роботу
Відкриття діалогового вікна з питанням: "Ви дійсно хочете завершити зміну?"
Результат діалогового вікна повернув Yes
Так
Відкрити вікно зі слайду
Запуск процедури відновлення зі слайду
Файл пошкоджено
Так
Виведення повідомлення про успіх на екран
Виведення помилки на екран
Внести зміну суми
Додавання нової транзакції у БД
Генерація чеку
Відправлення чеку на друк
Отримати звіт
Генерація звіту
Збереження файлу звіту у директорії
Додати клієнта
Генерація нового клієнта
Додавання нового клієнта у тимчасову БД
Активувати таймер клієнта
Клієнта було обрано у таблиці
Так
Таймер клієнта було активовано
Так
Активація таймеру
Оновлення таблиці клієнтів
Видалити клієнта
Клієнта було обрано у таблиці
Так
Таймер клієнта було активовано
Так
Додавання нової транзакції у БД
Видалення клієнта з тимчасової БД
Генерація чеку
Відправлення чеку на друк
Оновлення таблиці клієнтів
Виведення помилки на екран

ДОДАТОК В


Оператор терміналу-каси
Контроль клієнтів
Контроль каси
Додавання клієнта
Видалення клієнта
Запуск таймеру клієнта
Внесення вільної суми
Відновлення зміни
Введення інформації про клієнта
Обирання клієнта
Обирання клієнта
Введення суми
Вибір типу оплати
Друк чеку
<< include >>
<< use >>


StartupForm.cs
SettingsDialog.cs
ReportsForm.cs
InputDialog.cs
Report.cs
ReportFileGenerator.cs
SelectDateTimePeriodDialog.cs
EntityFrameworkCore
CashierDB.mdf


StartupForm.cs
InputDialog.cs
MessageBox.cs
CashierForm.cs
SettingsDialog.cs
ClientPayDialog.cs
FreeSumPayDialog.cs
cashierpro.exe
EPPlus.dll
EntityFramework.Core.dll
MySqlClient.dll
Logger.dll