Хмельницький національний університет
Кафедра інженерії програмного забезпечення

КУРСОВИЙ ПРОЕКТ

з дисципліни “Об’єктно-орієнтоване програмування”

на тему: “Розробка програмного забезпечення, яке реалізує довідник для клієнтів банку на мові програмування C# з використанням принципів ООР”

ПОЯСНЮВАЛЬНА ЗАПИСКА

КПІПЗ.200124.01.12 ПЗ

Спеціальність 121 – Інженерія програмного забезпечення

Студента І курсу, група ІПЗс-20-1 _____ Д. А. Ямборко
Підпис Ініціали, прізвище

Керівник роботи к.п.н., доцент _____ Н.І. Праворська
Підпис Ініціали, прізвище

Кількість балів _____

Оцінка за шкалою:

національною _____/ЄКТС _____

Члени комісії: _____ _____
Підпис Ініціали та прізвище

_____ _____
Підпис Ініціали та прізвище

_____ _____
Підпис Ініціали та прізвище

Хмельницький 2020

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *програмування та комп'ютерних і телекомунікаційних систем*
Кафедра *інженерії програмного забезпечення*
Спеціальність *121«Інженерія програмного забезпечення»*
(Шифр, назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
_____
"____"_____2020 року

ЗАВДАННЯ
НА КУРСОВИЙ ПРОЕКТ

Ямборко Д. А.
Прізвище, ім'я, по батькові студента

1. Тема проекту: Розробка програмного забезпечення, яке реалізує довідник для клієнтів банку на мові програмування C# з використанням принципів ООР.
керівник проекту: Праворська Наталя Іванівна к. п. н, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання
2. Строк подання студентом проекту на кафедру _____
3. Вихідні дані до проекту: методичні вказівки до курсового проектування для студентів спеціальності "Інженерія програмного забезпечення" з курсу "Об'єктно-орієнтоване програмування", тема курсового проекту Розробка програмного забезпечення, яке реалізує довідник для клієнтів банку на мові програмування C# з використанням принципів ООР.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) постановка задачі, короткий аналіз предметної області, побудова структури програмного продукту, програмна реалізація, інструкція користувача.
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
6. Дата видачі завдання: 5.09.2020

Керівник роботи (проекту) _____ _____
(підпис) (П.І.Б.)
Завдання прийняв до виконання _____ _____
(підпис дипломника) (П.І.Б.)

АНОТАЦІЯ

Курсовий проект “Розробка програмного забезпечення, яке реалізує довідник для клієнтів банку на мові програмування C# з використанням принципів ООР”

Автор роботи: Ямборко Д. А.

Керівник роботи: Праворська Н.І.

Обсяг – 57 с., 25 рис., 1 додаток.

Мета курсового проекту: розробка програмного застосунку для автоматизації процесу обліку банківських клієнтів банком.

В даному курсовому проекті вивчено банківську систему та елементарні банківські операції. Для розробки даного програмного продукту використано середовище Microsoft Visual Studio, а саме Windows Forms. За допомогою даних технологій створено інтерфейс для довідника та функціонал програми. Для проектування даної задачі використано UML діаграми, розроблені в середовищі Draw.io.

Дата_____ Підпис_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів курсового проекту	Термін виконання етапів проекту	Відмітка про виконання
1	Дослідження предметної області	09.09 – 10.09	
1.1	Характеристика функціональної структури предметної області	10.09 – 16.09	
1.2	Аналіз останніх публікацій, досліджень та існуючих рішень	17.09 – 23.09	
1.3	Постановка задачі та перелік задач для реалізації	24.09 – 08.10	
2	Проектування структури застосування	08.10 – 09.10	
2.1	Математична модель об’єкту проектування	09.10 – 15.10	
2.2	Вибір засобів розробки інформаційної системи	16.10 – 22.10	
2.3	Розробка структури інформаційної системи	23.10 – 05.11	
3	Програмна реалізація	06.11 – 23.11	
3.1	Структура і функціональне призначення модулів системи, їх взаємозв’язок	24.11 – 25.11	
3.2	Розробка програмних модулів	25.11 – 27.11	
3.3	Налагодження та тестування	27.11 – 29.11	
3.4	Інструкція користувача	30.11 – 01.12	
3.5	Вимоги технічних засобів	02.12 – 05.12	

Студент_____
(підпис)

Керівник проекту_____
(підпис)

ЗМІСТ

Вступ **5**

1 Дослідження предметної області **6**

1.1 Характеристика функціональної структури предметної області 6

1.2 Аналіз останніх публікацій, досліджень та існуючих рішень 7

1.3 Постановка задачі та перелік задач для автоматизації 8

2 Проектування структури інформаційної системи **10**

2.1 Математична модель об’єкту проектування 10

2.2 Розробка структури інформаційної системи 11

2.3 Вибір засобів розробки інформаційної системи 14

3 Програмна реалізація **16**

3.1 Структура і функціональне призначення модулів системи, їх взаємозв’язок 16

3.2 Розробка програмних модулів 17

3.3 Інструкція користувача 23

3.4 Вимоги до технічних засобів 29

Висновки **30**

Перелік посилань **31**

Додатки **32**

ВСТУП

У наш час людство переживає науково-технічну революцію, в якості матеріальної основи якої служить електронно-обчислювальна техніка. На базі цієї техніки з'являється новий вид технологій - інформаційні. До них відносяться процеси, де "вихідним матеріалом" і "продукцією" є інформація. Зрозуміло, що інформація, яка переробляється, зв'язана з визначеними матеріальними носіями і, отже, ці процеси включають також переробку речовини і переробку енергії. Але останнє не має істотного значення для інформаційних технологій. Головну роль тут грає інформація, а не її носій. Як виробничі, так і інформаційні технології виникають не спонтанно, а в результаті технологізації того чи іншого соціального процесу, тобто цілеспрямованого активного впливу людини на ту чи іншу область виробництва і перетворення її на базі машинної техніки. Чим ширше використання ЕОМ, тим вище їхній інтелектуальний рівень, тим більше виникає видів інформаційних технологій, до яких відносяться технології планування і керування, наукових досліджень і розробок, експериментів, проектування, грошово-касових операцій, криміналістики, медицини, утворення та інші.

У кожному банку відбувається облік клієнтів та грошово-касових операцій. Це зумовлює використання певних запам’ятовуючих пристроїв робітниками банку. І саме за допомогою комп’ютерів і програмного забезпечення цей процес можна автоматизувати.

Програмно реалізуємо облік клієнтів банку, що автоматизує роботу працівника банку.

1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Характеристика функціональної структури предметної області

Інформаційні технології дуже широко застосовуються у банках та інших фінансових установах. З розвитком технологій, постало завдання автоматизувати процес обліку клієнтів, надати їм зручний інтерефейс для самостійного маніпулювання своїм особистим кабінетом. При розробці програмних застосунків з графічним інтерфейсом під Windows платформу найбільш популярними є Windows Forms.

Windows Forms це - інтерфейс програмування додатків, відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді.

Всередині .NET Framework, Windows Forms реалізується в межах простору імен System.Windows.Forms.

Для найкращої взаємодії елементів графічного інтерфейсу і відповідних інстансів нашого програмного коду було використано технології ООП.

Важливою складовою програмного застосунку є можливість продовжувати працювати з набором даних, які користувач обробляв під час минулої сесії. Для забезпечення цієї можливості було реалізовано серіалізацію об’єктів додатку в формат XML та запис у файл. Відповідно коли користувач завершує свою роботу з програмою, всі зміни, які було внесено, записуються у файл. І при повторному запуску додатку ці зміни зчитуються з файлу і користувач може продовжити з ними працювати.

Щодо завдань автоматизації, необхідно відповідно автоматизувати процес обліку даних користувача, також автоматизувати можливість обліку банківських операції по певному користувачу.

1.2 Аналіз останніх публікацій, досліджень та існуючих рішень

Для створення будь-якого програмного забезпечення правильним буде спочатку проаналізувати існуючі методи та рішення, а також схоже програмне забезпечення. Це дозволить переглянути які методи використовують розробники для створення аналогічних програм.

У реальному житті напевно всі люди працюють з банками. Звичайно можна прийти у відділення та вирішити все там але набагато зручніше використати додаток на ПК чи смартфоні.

Ідеєю розробки проекту слугував додаток Privat24 (див. рис. 1.2.1), де також можливо лише за натисканням клавіші поповнити рахунок, відредагувати особисту інформацію чи змінити пін-код.

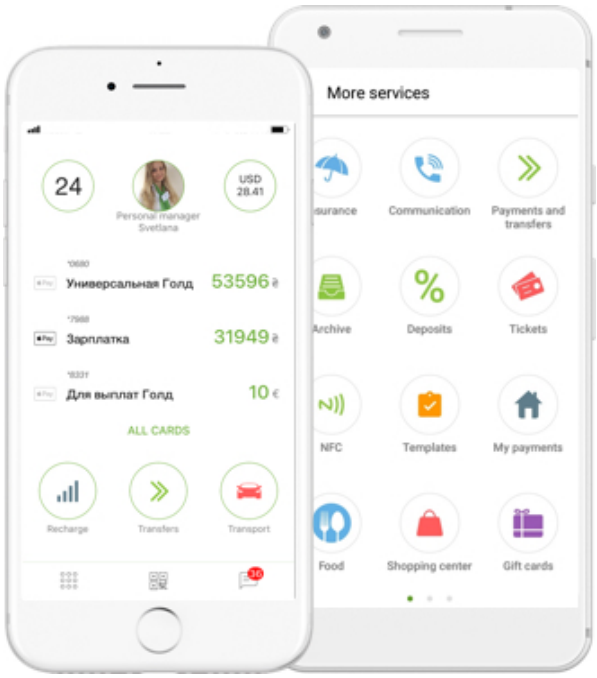

Рисунок 1.2.1 – Вигляд додатку – аналогу

Перевагами даного додатку є можливість керувати всіма банківськими рахунками та переглядати звіти поним, високий рівень захисту, безліч функції та можливостей.

До недоліків цього аналогу, можна віднести – неможливість працювати оффлайн та досить непростий, для звичайного користувача, інтерфейс.

1.3 Постановка задачі та перелік задач для автоматизації

Темою курсового проекту є програмний продукт “ Розробка програмного забезпечення, яке реалізує довідник для клієнтів банку на мові програмування C# з використанням принципів ООР ”

Метою даного проекту є створення інтерактивного середовища де клієнт банку з легкістю зможе вносити данні щодо своїх фінансових операцій і редагувати власну інформацію.

У програмі передбачити, відповідно, два типи користувачів:

- адміністратор;
- клієнт банку.

Можливості обох користувачів дещо різні, для адміністратора існує розширений функціонал – додавання, редагування, вилучення нових клієнтів, зміна особистих даних, а користувач в свою чергу може лише працювати з особистим рахунком.

Додаток повинен містити:

- демонстрація базових функцій проекту (додання, редагування клієнта банку);
- можливість поповнення/зняття коштів з особистого рахунку клієнта банку;
- додання депозиту;
- перегляд даних особистого рахунку клієнту банку;

Програма буде працювати на персональному комп’ютері без необхідності підключення до мережі Інтернет. Для збереження файлів інформації про клієнтів банку буде використовуватись технологія серіалізації XML.

2. ПРОЕКТУВАННЯ СТРУКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Математична модель об’єкту проектування

UML (англ. Unified Modeling Language) – уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем

Цей інструмент формалізації добре підходить для проектування програмної системи курсового проекту.

Спочатку спроектовано діаграму варіантів використання (Use Case Diagram) (див. рис. 2.1.1 і 2.1.2), що демонструє загальний функціонал системи та її використання користувачами (акторами).

Рисунок 2.1.1 – Діаграма варіантів використання для клієнта банку

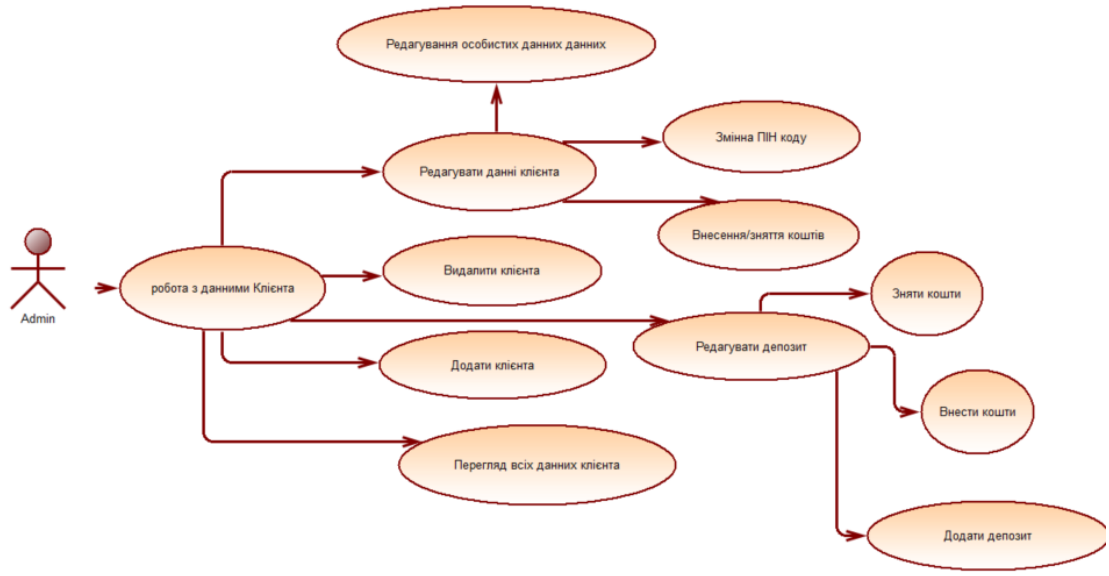

Рисунок 2.1.2 – Діаграма варіантів використання для адміністратора

Програму використовують адміністратор і клієнта, тому створено дві діаграми використання у ролі Адміністратора та Клієнта банку. Клієнта банку позначено відповідним елементом. В овалах описано дії, які може виконувати програма, відношенням асоціації показано з чим може працювати безпосередньо користувач при запуску програми.

2.2 Розробка структури інформаційної системи

Організація вхідних та вихідних даних має велике значення, бо вона формує функціонал самої програми. Структура проекту обов’язково продумуються наперед, бо потрібно знати які методи буде використано.

Оптимальним вибором моделі проекту буде модель, яка базується на подіях. У проекті використовується 2 глобальні змінні які містять інформацію про пакет питань що ми редагуємо та пакет питань для гри. Змінні одразу проініціалізовані для доступу до списків в середині об’єктів пакетів питань. Дані в пакетах це список тем, а в темах список питань, які композитивно залежать від класа предка. Тобто реалізована чітка ієрархія На вершині якого Пакет питань нижче список тем і ще нижче список питань. Дві вищеописаних глобальні змінних серіалізуються в файловий потік у випадку з пакетом для редактора і в мережевий потік у випадку з

пакетом для гри. Крім того існує глобальна змінна що містить покажчик на екземпляр класу користувача, що не взаємодіє з іншими класами.

Для повного розуміння як і яким чином буде реалізовано проект, доцільно створити кілька UML-діаграм. UML-діаграми це один з найкращих способів проектування програмного продукту.

Діаграма станів (Statechart Diagram) описує можливі послідовності станів і переходів, що характеризують поведінку системи.

На діаграмі показано такі стани: запуск програми веде до логіну – ввід пін-коду і відкриття відповідної форми клієнту банку чи адміністратора. (див. рис. 2.2.1)

Перебуваючи у формі Адміністратора, програма у стані обліку клієнта після завершення відбувається.

Перебуваючи у формі клієнта банку, програма у стані редагування або перегляду інформації щодо рахунку, після чого відповідно відбувається збереження.

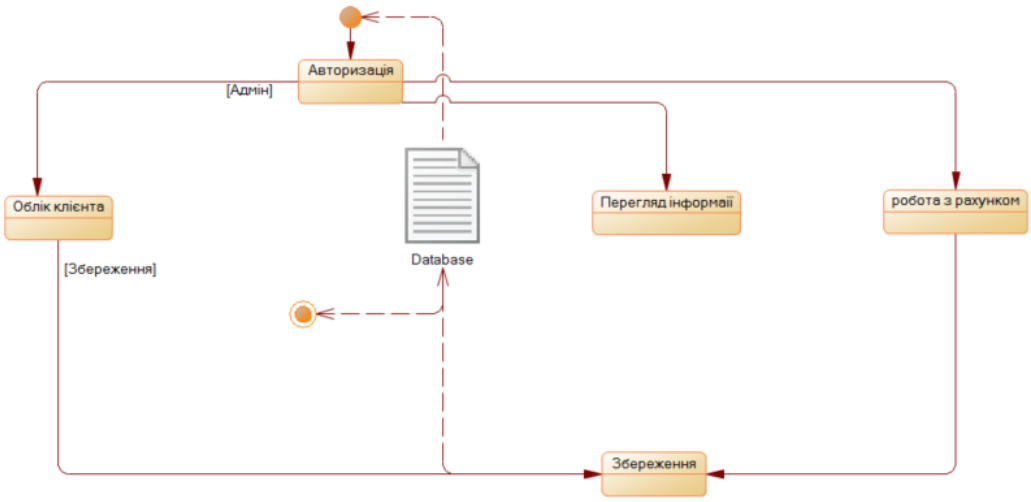

Рисунок 2.2.1. – Діаграма станів

Діаграма активності (Activity Diagram) - це візуальне представлення графу діяльності. Це так звана блок-схема проекту, але без деталізації рішень. (див. рис. 2.2.2)

Після запуску програми відбувається логін, перехід до форми Адміністратора, або ж перехід до форми Клієнта банку.

На формі Адміністратора користувач на першій вкладці має змогу додати клієнта банку а на іншій вкладці працювати з вже існуючими клієнтами.

На формі Клієнта банку користувач має змогу переглянути інформацію по власному рахунку та зняти або поповнити рахунок.

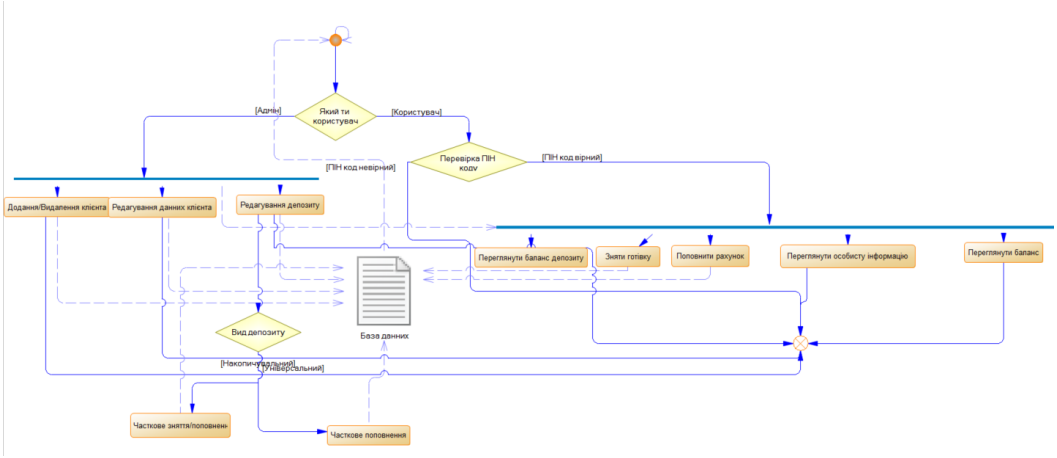

Рисунок 2.2.2. – Діаграма активності

Діаграма послідовності (Sequence Diagram) відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об’єкти та послідовність відправлених повідомлень.

Розглянемо діаграму послідовності для клієнта банку (див. рис. 2.2.3). У діаграмі наявні такі об’єкти як форма логіну, перегляд інформації, зняття/поповнення рахунку.

Дії між об’єктами називаються повідомленнями. Так, наприклад, щоб зняти кошти, користувач надсилає повідомлення, налаштовуючи параметри, до об’єкту зняття коштів. Поміщено блок перевірки чи можлива данна операція, де перевіряється чи правильні були дані зворотнім запитом, і надсилаємо інші, якщо ні.

Щодо Адміністратора, то спочатку відбувається перехід до блоку логіну, де відбувається введення паролю, перевірка і надання доступу до подальшого функціоналу.

Для редагування даних клієнтів користувач просто надсилає запит і відредаговані данні зберігаються у файл.

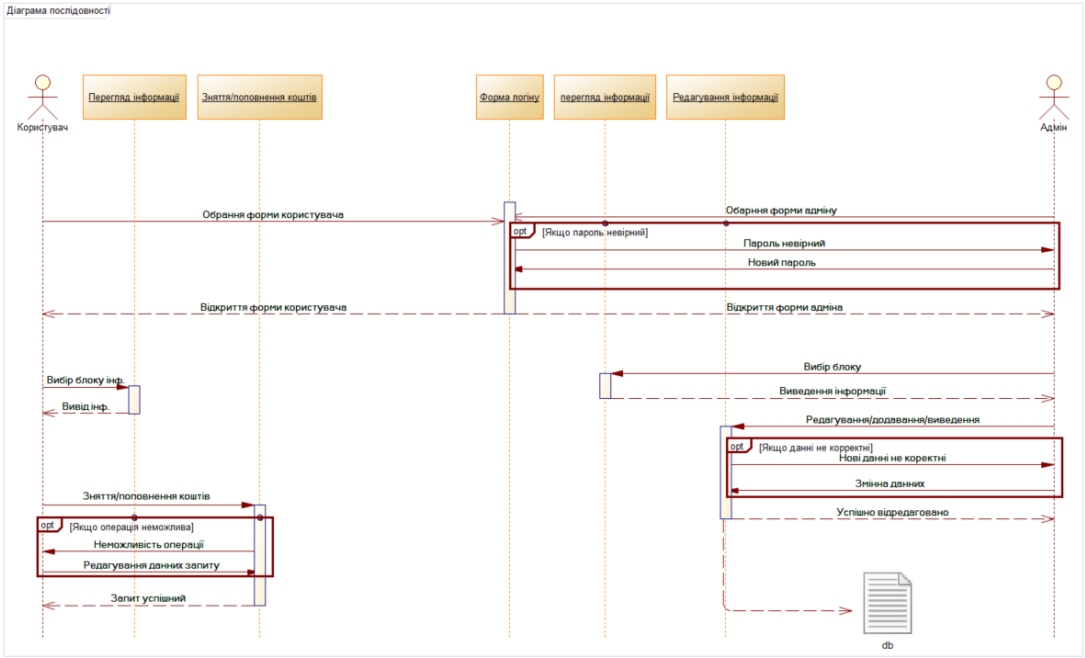

Рисунок 2.2.3. – Діаграма послідовності для клієнта банку і Адміністратора

Діаграма комунікації (Communication Diagram) відображає взаємодії об'єктів (див. рис. 2.2.4), але на відміну від діаграми послідовності, на діаграмі комунікації явно вказуються зв’язки, а час як окремий простір вимірювання не використовується.

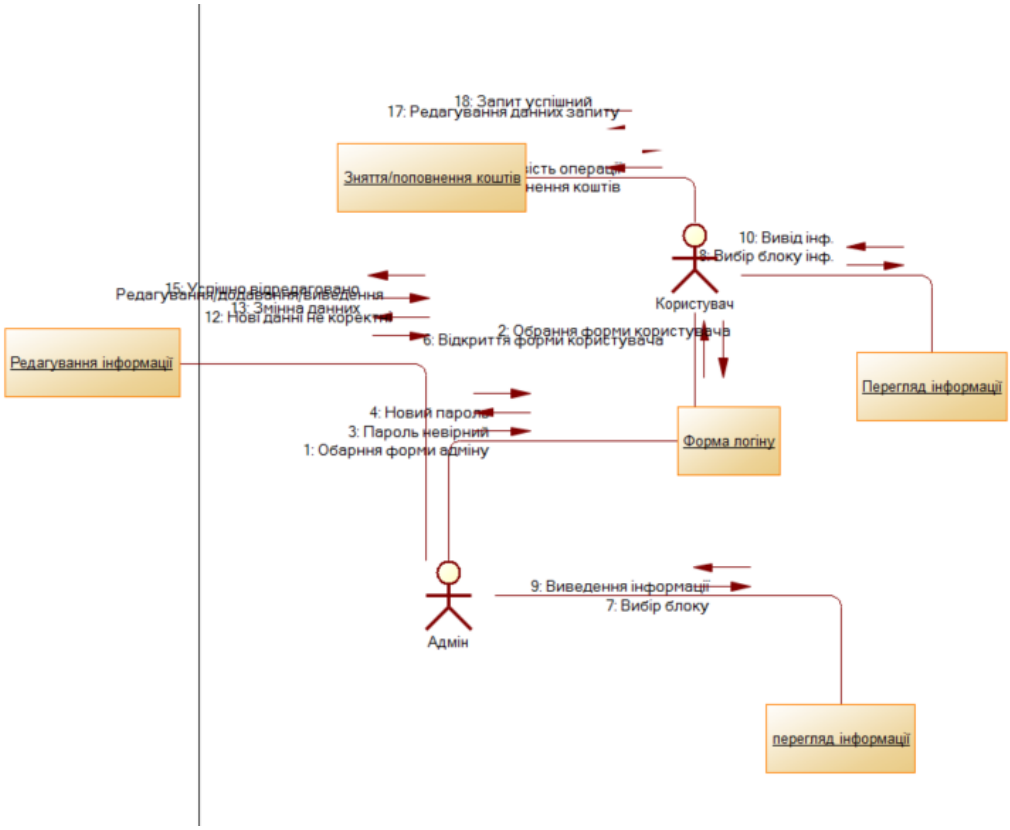

Рисунок 2.2.4. – Діаграма комунікації для Викладача

2.3 Вибір засобів розробки інформаційної системи

Варіантів реалізації програмного засобу є досить багато, проте було обрано об’єктно-орієнтовну мову програмування C#, так як програмне забезпечення було призначено виключно на операційну систему Windows.

Наразі найпопулярнішим середовищем програмування на C# у операційній системі Windows є Visual Studio. Дане програмне середовище надає змогу використовувати нові багатофункціональні модулі для реалізації проектів різної складності з величезною кількістю різноманітних компонентів.

Visual Studio – інтегроване рішення для управління життєвим циклом додатків, яке допомагає організаціям будь-якого масштабу, які працюють у сфері IT та програмного забезпечення, постійно підтримувати конкурентоздатність своєї пропозиції, гарантуючи швидкість і якість. Це комплексне середовище з широкими функціональними можливостями має удосконалений інтерфейс і містить нові інструменти для підтримки багатьох процесів. За його допомоги, розробники можуть створювати інноваційні якісні додатки з привабливим виглядом, або ж консольні прості програми, що дозволяє початківцям ввійти в курс справи перед тим як починати великі серйозні проекти.

Так як даний проект, не містить масштабних даних, що потребують підтримки баз даних MySQL, для збереження певних проміжних результатів було вибрано серіалізацію формату XML.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Структура і функціональне призначення модулів системи, їх взаємозв’язок

Основним структурним елементом системи, як і одиницею інкапсуляції є клас, а інтерфейсні компоненти при використанні технології Windows Forms представленні в вигляді форм, тому основними структурними компонентами системи є класи та форми.

Для створення діаграми класів необхідно визначитись з класами, їх екземплярами та методами. Також необхідно визначитись зі зв'язками між класами. Діаграма класів (Class Diagram) – статичне представлення структури моделі. Відображає статичні елементи, такі як: класи, типи даних, їх зміст та відношення.

На діаграмі (див. рис. 3.1.1) зображено такі класи: Клієнт (містить інформацію про клієнта банку), Рахунок (містить інформацію про рахунок користувача є полем класу Клієнт), Депозит (інформація про депозит, є полем класу рахунок), Операція (інформація про певну операцію її сума та час

Атрибутами класу Клієнт є поле класу Рахунок, відповідно клас Рахунок містить поле Депозит тобто цей клас являє собою сутність, що включає як складені частини інші сутності. Це явище називається відношенням агрегації. Воно позначене на діаграмі стрілкою з незафарбованим ромбом.

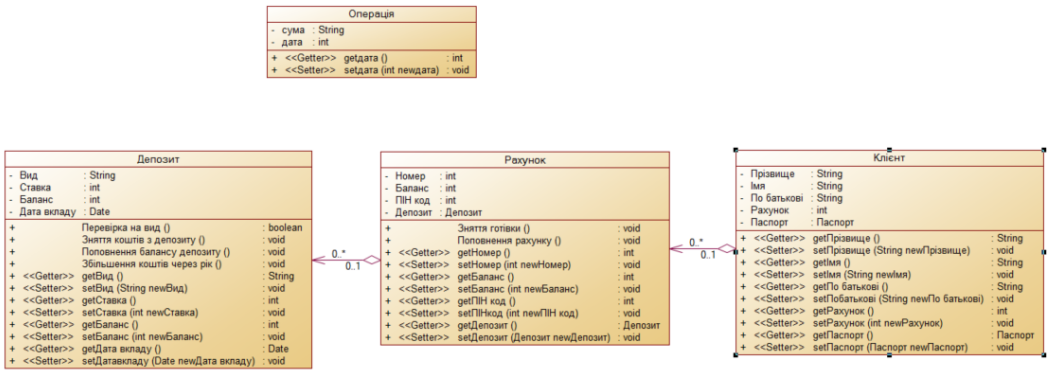

Рисунок 3.1.1. – Діаграма класів

Діаграма компонентів (див. рис. 3.1.2) описує усі компоненти курсової. Це можуть бути форми, бази данних та інші компоненти. Діаграма компонентів (Component Diagram) описує фізичні особливості представлення системи.

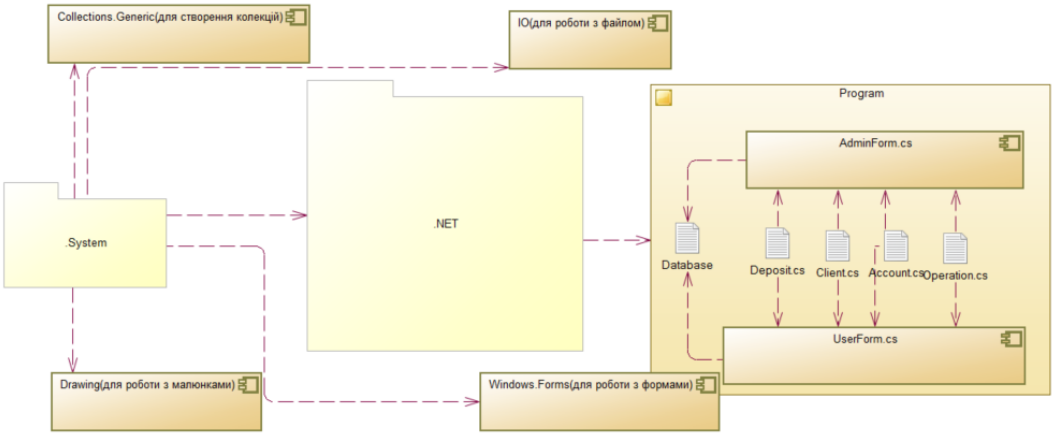

Рисунок 3.1.2. – Діаграма компонентів

Головним компонентом даного додатку є UserForm, який відкривається після валідації данних на формш LoginForm.

Всі використанні класи та компоненти є стандартними класами фреймворку .NET.

3.2 Розробка програмних модулів

Об’єктно-орієнтоване програмування надає велику кількість переваг та можливостей. В особливості розбиття проекту на модулі, агрегація, композиція, наслідування, інкапсуляція, поліморфізм – все це допоможе зробити проект функціонально завершеним.

В даному проекті застосовується чотири класи «User», «Operation», «Deposit», «Account». Між ними зв’язок «композиція». Для роботи з бінарним файлом всі класи серіалізовані.

Перш за все необхідно створити проект в Visual Studio та задати йому відповідне ім’я. Після цього можна почати створювати класи згідно діаграми класів, спроектованої у попередньому розділі.

Під час написання програми було застосовано такі етапи об’єктно-орієнтовного програмування:

- Створення класів та визначення зв’язків між ними;
- Визначення атрибутів, методів, властивостей класу;
- Оголошення екземплярів класу як самостійних об’єктів, або списків типу клас;
- Використання класів та їх методів у класах форм програми.

Для створення класу необхідно натиснути лівою кнопкою миші на пункт меню Проект і обрати створення нового класу:

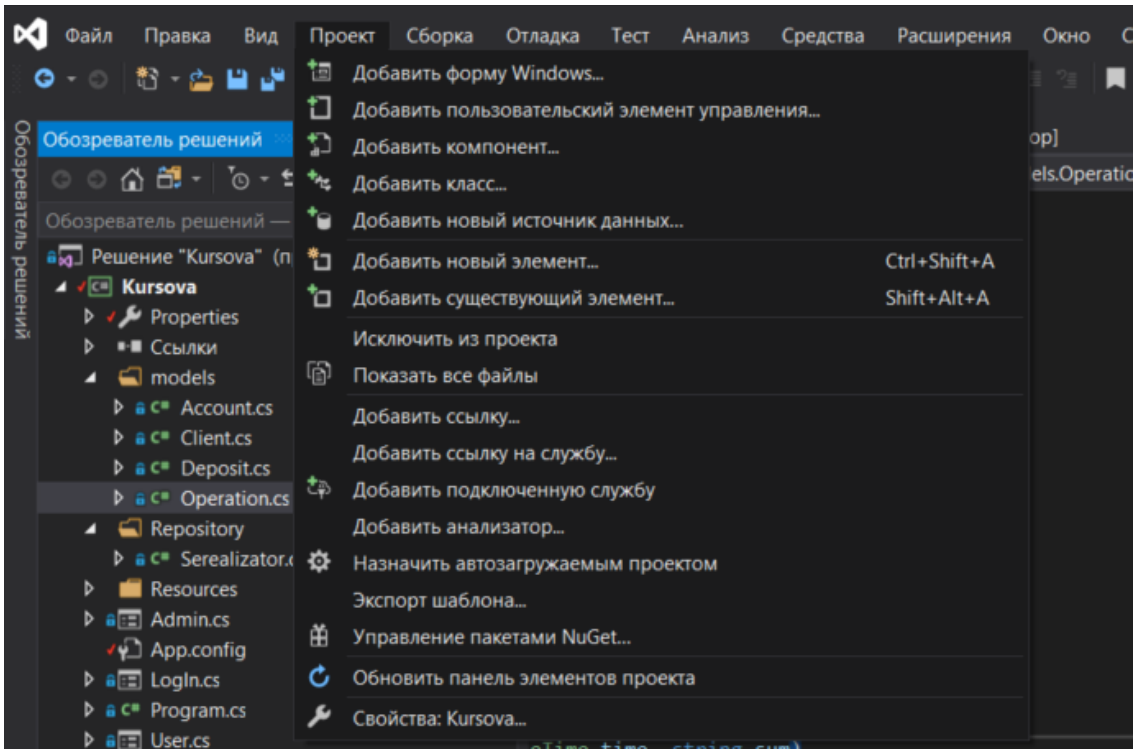

Рисунок 3.2.1. – Створення нового компонента

Після цього треба ввести назву класу в відповідне поле. Програма автоматично створить заготовку.

Аналогічні дії виконано для створення нової форми.

Так як в проекті передбачено різний функціонал, він розміщується на різних формах: User.cs, Admin.cs, LogIn.cs (див. рис. 3.2.2)

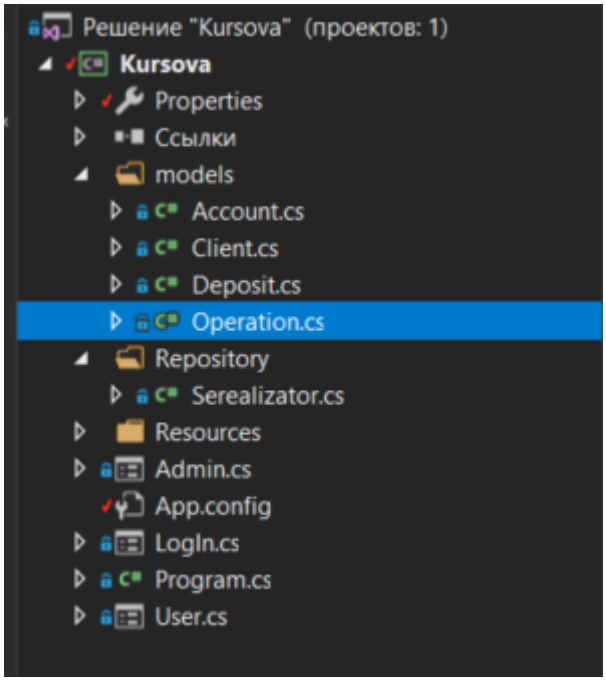

Рисунок 3.2.2. – Перелік класів і форм програми

На формах використано такі стандартні компоненти: MenuStrip, Button, StatusStrip, Label, DataGridView, TextBox, ComboBox, ToolStripMenu, RadioButton.

Для додавання нового елементу на форму достатньо вибрати його зі списку та клацнути мишкою в потрібному місці. Після цього елемент керування відобразиться на формі. Якщо двічі клацнути на елемент, відкриється обробник подій, де можна задати властивості, дії, які цей елемент виконуватиме.

Розглянемо на прикладі кнопки Зняти кошти.

```
private void Button2_Click(object sender, EventArgs e)
    {
       string suma = Interaction.InputBox("Введіть суму", "", "");
       if (!tryToUint(suma))
       {
          return;
       }

       if (Convert.ToDouble(suma) == 0)
```

```
                {
                    showMes("Не може бути 0");
                    return;
                }
                if (client.account.balance.Equals(0) || ToUInt(suma) > client.account.balance)
                {
                    showMes("Недостатньо коштів на рахунку");
                    return;
                }
                else
                {
                    client.account.minusMoney(Convert.ToDouble(suma));
                    client.account.lastOperations.Add(new Operation(DateTime.Now, ("-" + suma + " UAH")));
                    clientsDictionary[key] = client;
                    Serealizator.serealizable(clientsDictionary);
                    label7.Text = "Баланс " + client.account.balance.ToString();
                    refreshDatagrid(client.account.lastOperations);
                    tSS2.Text = "Знято" + suma + " UAH";
                }
            }
```

Як бачимо, використовуючи атрибути класів, тут реалізовані методи зняття коштів відповідного особистого рахунку.

Клас Операція у попередньому фрагменті коду:

```
public class Operation
    {
        [DataMember]
        public DateTime time;
```

```
        [DataMember]
        public string sum { get; set; }

        public Operation(DateTime time, string sum)
        {
            this.time = time;
            this.sum = sum;
        }
    }
```

Розглянемо код Глобального класу, який містить в собі практично всі функціонуючі елементи програми:

```
    public class Account
    {
        [DataMember]
        public double balance { get; set; }
        [DataMember]
        public Deposit deposit { get; set; }

        [DataMember]
        public List<Operation> lastOperations { get; set; }

        public void minusMoney(double sumOperation)
        {
            if (balance < sumOperation)
            {
                MessageBox.Show("Недостатньо коштів на рахунку", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
```

```
            else {
                balance -= sumOperation;
            }
        }

        public void plusMoney(double sumOperation)
        {
            balance += sumOperation;
        }

        public Account(double balance, Deposit deposit, List<Operation> lastOperations)
        {
            this.balance = balance;
            this.deposit = deposit ?? throw new ArgumentNullException(nameof(deposit));
            this.lastOperations = lastOperations ?? throw new ArgumentNullException(nameof(lastOperations));
        }
    }
}
```

Весь код програми, з усіма методами реалізації можна переглянути в Додатку А.

3.3 Інструкція користувача

Інсталяцію програмного продукту можна розпочати відкривши файл Bank.exe (див. рис. 3.3.1).

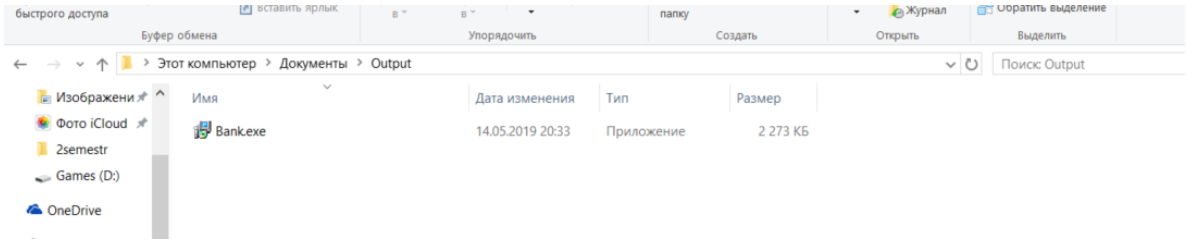

Рисунок 3.3.1. – Установочний файл

Вибір мови, натискаємо ОК. (див. рис. 3.3.2)

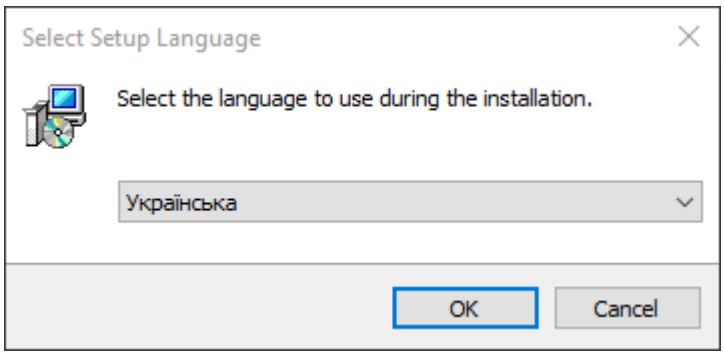

Рисунок 3.3.2. – Встановлення програми

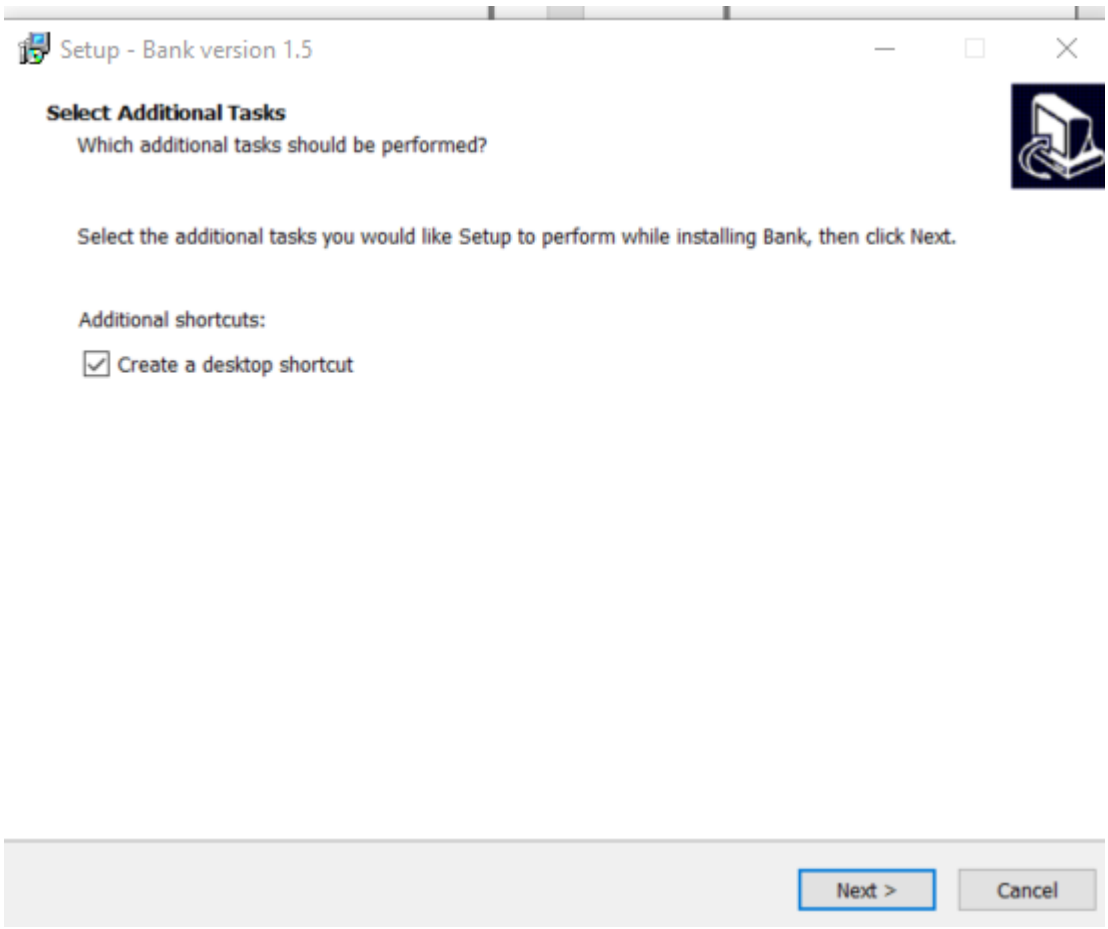

Рисунок 3.3.3. – Встановлення програми

Відмічаємо галочкою, якщо потрібен ярлик на робочому столі, натискаємо Далі.

Перевіряємо дані, натискаємо Встановити. (див. рис. 3.3.4)

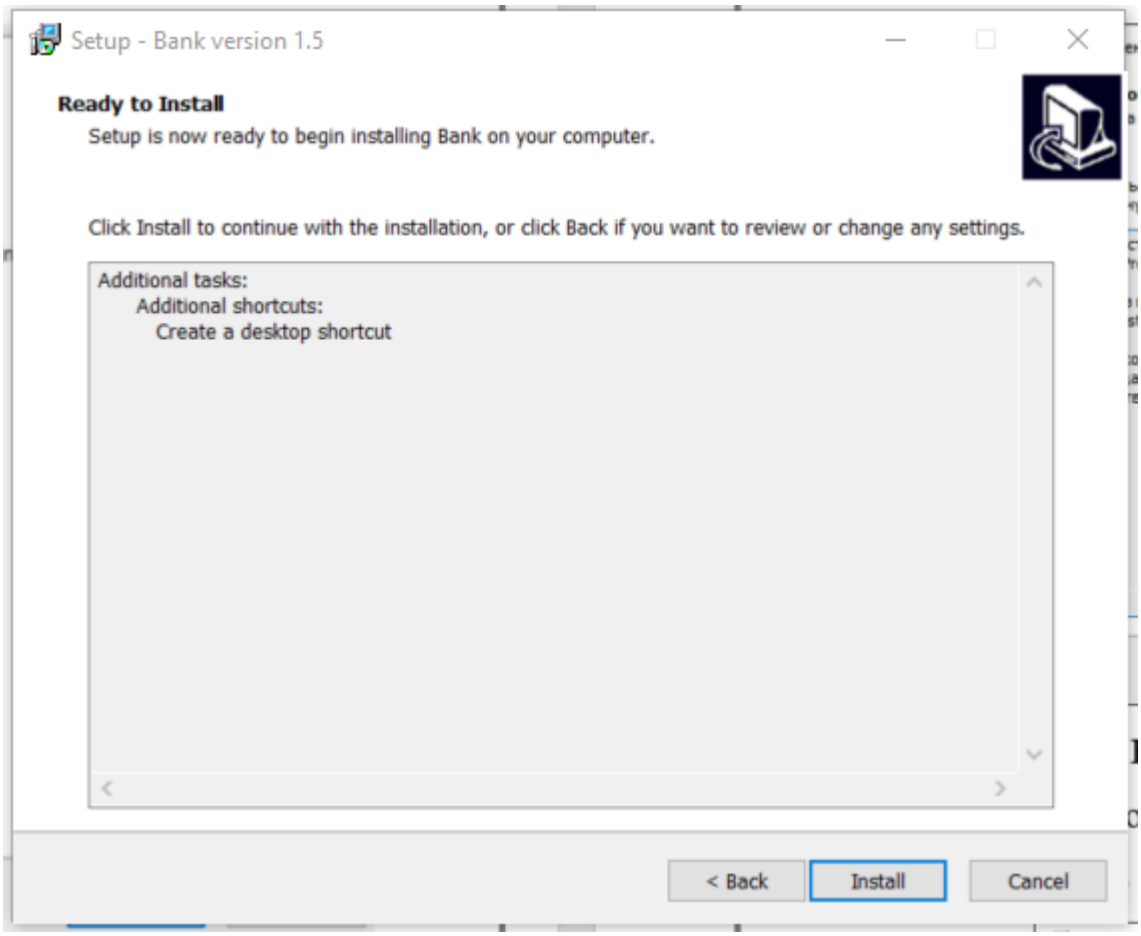

Рисунок 3.3.4. – Встановлення програми

Готово. Програма встановлена і її можна запустити з ярлика на робочому столі або з меню Пуск.

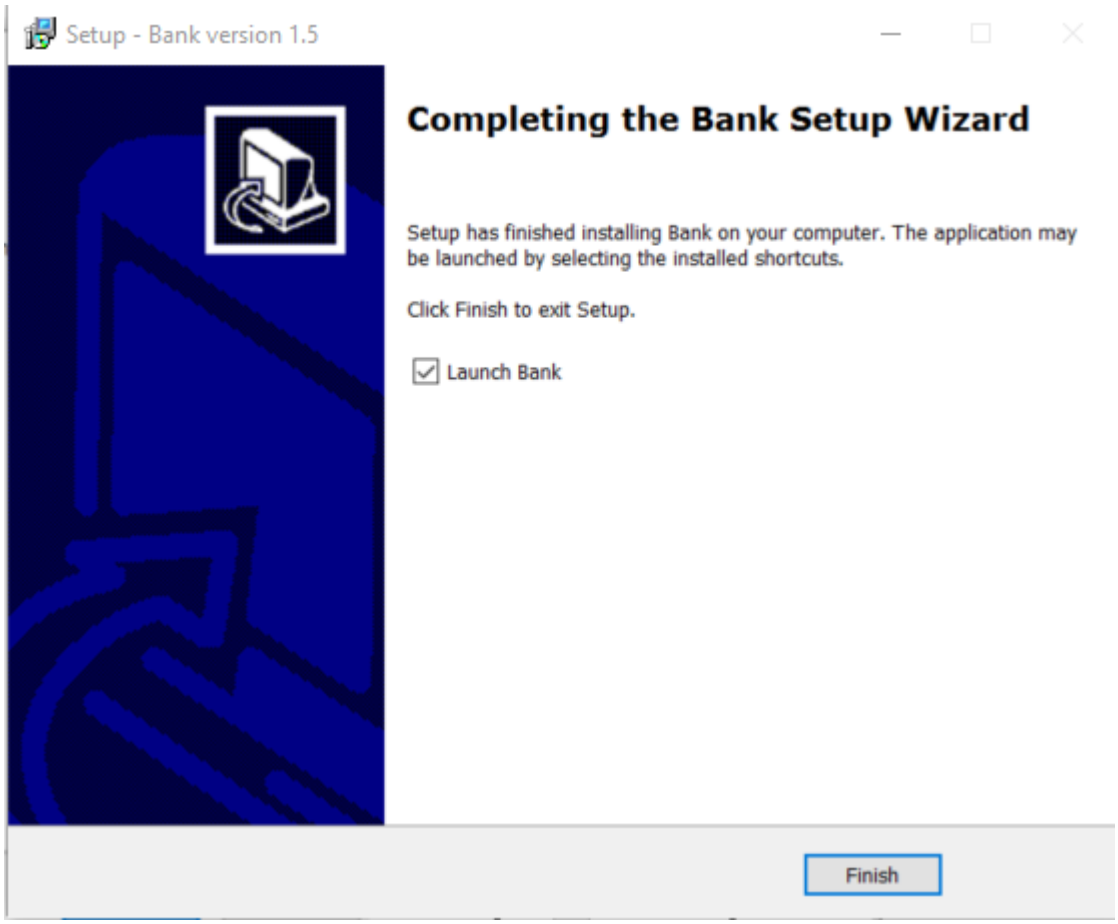

Рисунок 3.3.5. – Встановлення програми

Рисунок 3.3.6. – Вигляд ярлика встановленої програми

При запуску програми відображається головне вікно логіну. (див. рис. 3.3.7)

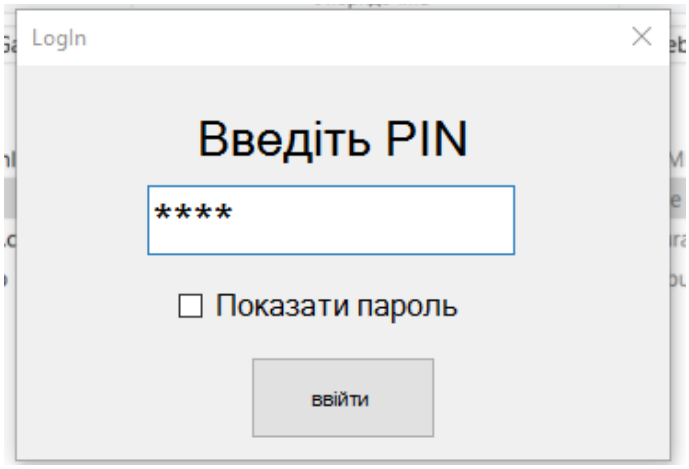

Рисунок 3.3.7. – Головне вікно програми

Після введення пін-коду відповідно відкриється форма Адміністратора або клієнта банку (див. рис. 3.3.8)

Рисунок 3.3.8. – Форма Адміністратора перша вкладка Додання нового клієнта

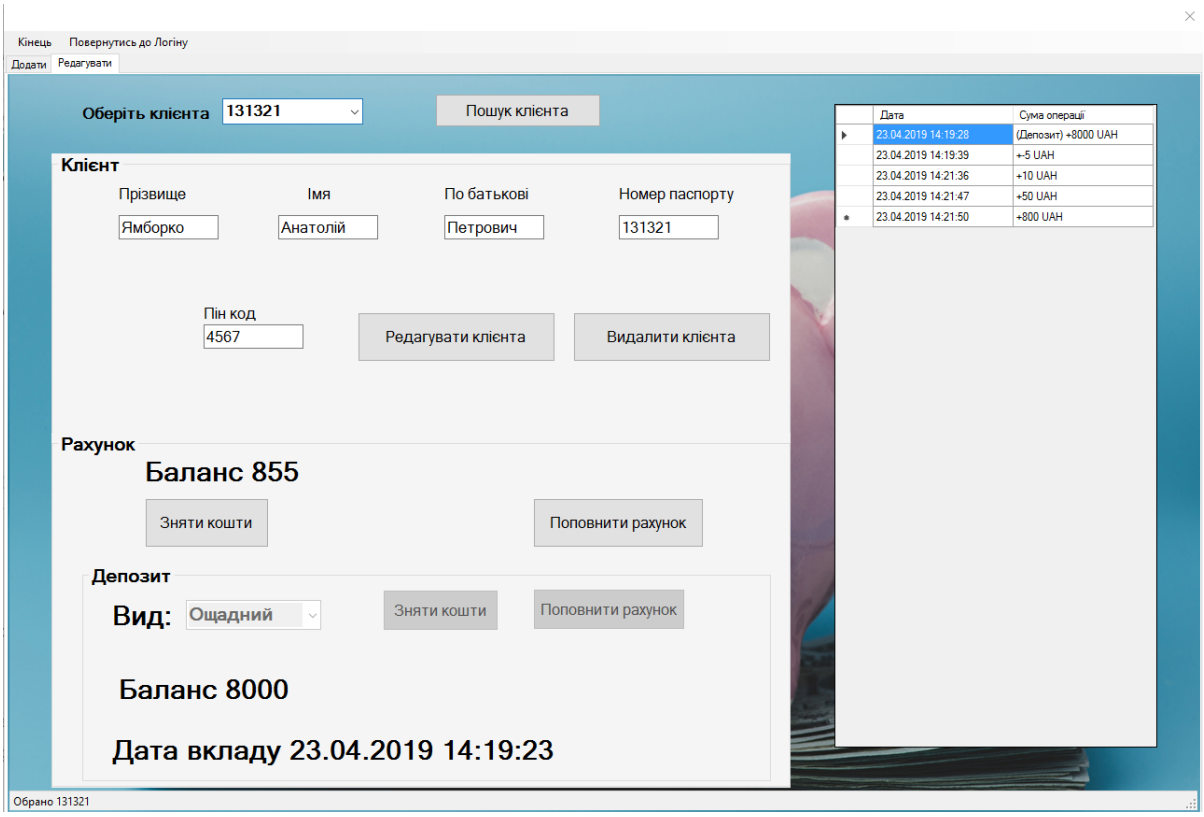

Рисунок 3.3.9. – Форма Адміністратора перша вкладка Редагування інформації існуючих клієнтів

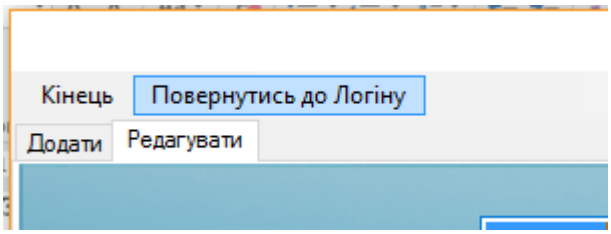

Рисунок 3.3.10. – Форма Адміністратора кнопка для повернення до форми Логіну

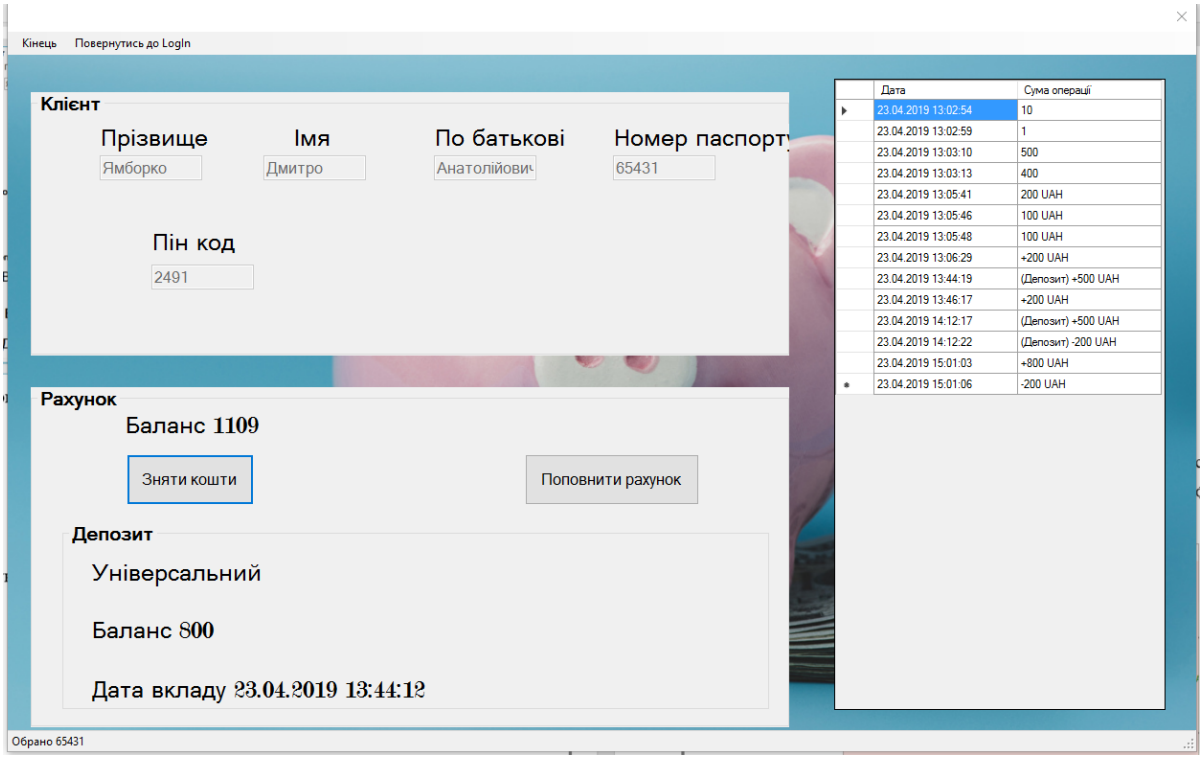

Рисунок 3.3.11. – Форма клієнта банку

Знімемо з рахунку 100 грн та поповнимо на 200 (див. рис. 3.3.12)

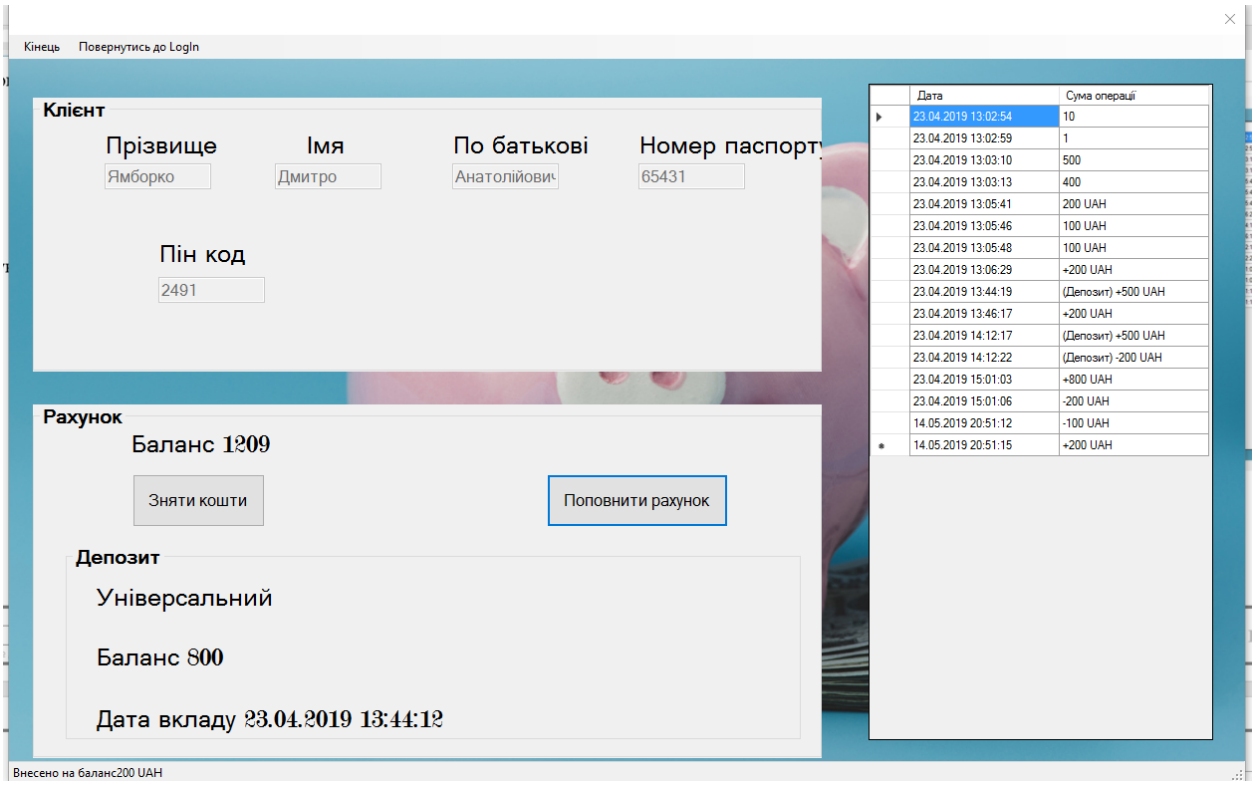

Рисунок 3.3.12. – Форма клієнта банку з оновленим списком операцій

3.4 Вимоги до технічних засобів

Для використання програмного забезпечення не потрібно ніяких специфічних засобів, програма запускається на звичайному персональному, системні вимоги необхідні для правильної робити програми описані в діаграмі розгортання Діаграма розгортання (див. рис. 3.4.1) описує потрібні мінімальні ресурси для нормальної роботи програми (наявність персонального комп'ютера , вихід в інтернет, вихід в локальну мережу, мінімальні характеристики комп'ютера та інше).

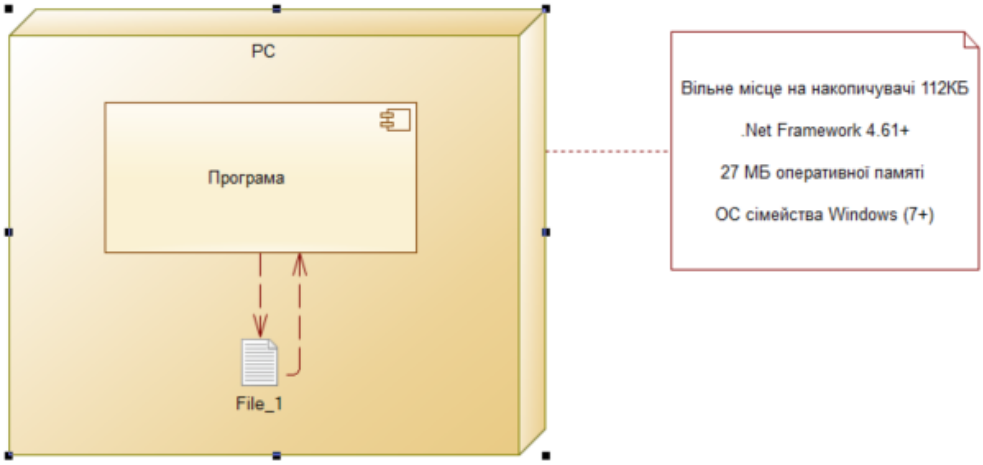

Рисунок 3.4.1 – Діаграма розгортання

Системні вимоги:

- операційна система: Windows 7 і вище;
- Microsoft .NET Framework: Версія 4.7 і вище;
- тактова частота процесора: 1 ГГц і вище;
- оперативна пам'ять: 128 МБ і більше;
- вільне місце на диску: 5 МБ.

Після того як переконались, що комп'ютер наділений необхідними системними вимогами можна безпосередньо перейти до інсталяції програмного продукту, для цього запустивши файл "Bank.exe" подвійним натисканням клавіші миші, розміщений на диску С.

ВИСНОВКИ

Використовуючи об'єктно-орієнтовне програмування, в даному курсовому проекті було реалізовано навчальну програму на тему: “Моделювання роботи банку «Інтелект», моделювання банкомату, вклади і види вкладів, реєстр рахунків фізичних та юридичних осіб”, яка дозволяє ввести облік клієнтів банку, моделювати роботу банкомату, додавати депозити.

У процесі проектування було створено вісім UML діаграм та шість класів ООП. Мовою розробки була мова програмування C# у середовищі Visual Studio, уніфікована мова моделювання UML в середовищі PowerDesigner.

Звісно, як і всі програмні засоби, даний проект має ряд переваг та недоліків, наведених нижче.

До переваг можна віднести:

- легкий в користуванні;
- багатофункціональність;
- доступний інтерфейс;
- можливість працювати оффлайн.

Недоліки програмного забезпечення:

- невеликий функціонал клієнта банку;
- відсутність дизайну.

В цілому цей додаток задовольняє всі поставлені до нього вимоги, розроблено та протестовано проект, який автоматизує рутинну роботу клієнтів та робітників банку пов’язану з банківськими процесами процесом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Эндрю Троелсен - Язык программирования C# 5.0 и платформа .NET 4.5. – М.: И.Д. Вильямс, 2013 – 1312 с.

2. Нейгел К., Ивьен Б., Глинн Дж., Уотсон К. - C# 4.0 и платформа .NET 4 для профессионалов. – М.: И.Д. Вильямс, 2011 – 1440 с.

3. Рихтер Дж. - CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. – СПб.: Питер, 2013 – 896 с.

4. Інтернет-ресурс для вивчення програмування – https://stackoverflow.com/

ДОДАТКИ

Додаток А – Код проекту

(Довідковий)

Account.cs

```
namespace Kursova.Model
{
    [DataContract]
    public class Account
    {
        [DataMember]
        public double balance { get; set; }
        [DataMember]
        public Deposit deposit { get; set; }

        [DataMember]
        public List<Operation> lastOperations { get; set; }

        public void minusMoney(double sumOperation)
        {
            if (balance < sumOperation)
            {
                MessageBox.Show("Недостатньо коштів на рахунку", "Ошибка!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            else {
                balance -= sumOperation;
            }
        }

        public void plusMoney(double sumOperation)
        {
            balance += sumOperation;
        }

        public Account(double balance, Deposit deposit, List<Operation> lastOperations)
        {
            this.balance = balance;
            this.deposit = deposit ?? throw new ArgumentNullException(nameof(deposit));
            this.lastOperations = lastOperations ?? throw new
ArgumentNullException(nameof(lastOperations));
        }
    }
}
```

Client.cs

```
namespace Kursova.models

{

    [DataContract]

    public class Client

    {

        [DataMember]
```

```
        public string name { get; set; }
        [DataMember]
        public string prizv { get; set; }
        [DataMember]
        public string surname { get; set; }
        [DataMember]
        public uint pasport { get; set; }

        [DataMember]
        public Account account { get; set; }
        public Client(string name, string prizv, string surname, uint pasport, Account
account)
        {
            this.name = name ?? throw new ArgumentNullException(nameof(name));
            this.prizv = prizv ?? throw new ArgumentNullException(nameof(prizv));
            this.surname = surname ?? throw new ArgumentNullException(nameof(surname));
            this.pasport = pasport;
            this.account = account ?? throw new ArgumentNullException(nameof(account));
        }
    }
}
```

Deposit.cs

```
namespace Kursova.models
{
    [DataContract]
    public class Deposit
    {

        [DataMember]
        public string type { get; set; }
        [DataMember]
        public double balance { get; set; }
        [DataMember]
        public double stavka { get; set; }
        [DataMember]
        public DateTime dateOfDeposit { get; set; }
```

```
        [DataMember]
        public TimeSpan interval { get; set; }

        public void initStavka(string str)
        {
            stavka = str == "Накопичувальний" ? 12.5 : 0;
            stavka = str == "Ощадний" ? 12.75 : 0;
            stavka = str == "Універсальний" ? 12 : 0;
        }

        public void minusMoney(double sumOperation)
        {
            if(type != "Універсальний")
            {
                MessageBox.Show("Вам недоступна ця операція", "Ошибка!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            if (balance < sumOperation)
            {
                MessageBox.Show("Недостатньо коштів на рахунку", "Ошибка!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            else
            {
                balance -= sumOperation;
            }
        }

        public void plusMoney(double sumOperation)
        {
            if (type == "Ощадний")
            {
                MessageBox.Show("Вам недоступна ця операція", "Ошибка!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
```

```
            }
            balance += sumOperation;
        }
        public void augmentMoney()
        {
            if ((DateTime.Now - dateOfDeposit).Equals(interval))
            {
                balance *= stavka;
            }
            else return;
        }

        public Deposit(string type, double balance, double stavka, DateTime
dateOfDeposit, TimeSpan interval)
        {
            this.type = type ?? throw new ArgumentNullException(nameof(type));
            this.balance = balance;
            this.stavka = stavka;
            this.dateOfDeposit = dateOfDeposit;
            this.interval = interval;
        }
    }
}
```

Operation.cs

```
namespace Kursova.models
{
    [DataContract]
    public class Operation
    {
        [DataMember]
        public DateTime time;
        [DataMember]
        public string sum { get; set; }

        public Operation(DateTime time, string sum)
        {
```

```
            this.time = time;
            this.sum = sum;
        }
    }
}
```

Admin.cs

```
namespace Kursova
{
    public partial class Admin : Form
    {
        Dictionary<uint, Client> clientsDictionary = new Dictionary<uint, Client>();
        public Admin()
        {
            InitializeComponent();
        }

        private void кінецьToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void режимКористувачаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LogIn login = new LogIn();
            login.Show();
            this.Hide();
        }
        public void clearTb(TextBox tb)
        {
            tb.Text = "";
        }

        private uint ToUInt(String str)
        {
            return Convert.ToUInt32(str);
        }
```

```
public void showMes(String str)
{
    MessageBox.Show(str, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
}

public bool tryToUint(string str)
{
    try
    {
        Convert.ToUInt32(str);
        return true;
    }
    catch (Exception)
    {
        showMes("Некоректне введення");
        return false;
    }
}

/* public bool tryToDouble(string str)
{
    try
    {
        Convert.ToDouble(str);
        return true;
    }
    catch (Exception)
    {
        showMes("Некоректне введення");
        return false;
    }
}*/
private void refreshDatagrid(List<Operation> operationList)
{
    if (operationList.Count == 0)
```

```
            {
                showMes("Список операцій пустий пустий");
                return;
            }
            dataGridView1.RowCount = operationList.Count;
            int i = 0;
            foreach (Operation a in operationList)
            {
                dataGridView1[0, i].Value = a.time.ToString();
                dataGridView1[1, i].Value = a.sum.ToString();
                i++;
            }
        }

        private void refreshCbEditClient()
        {
            cbEditClient.Items.Clear();
            cbEditClient.Items.AddRange(clientsDictionary.Values.Select(a =>
a.pasport.ToString()).ToArray());
            cbEditClient.SelectedItem = null;
            cbEditClient.Text = "";
            cbEditClient.Update();
        }

        private void clearAllFields()
        {
            tbEditName.Text = "";
            tbEditPrizv.Text = "";
            tbEditSurname.Text = "";
            tbEditPasport.Text = "";
            tbEditPinKod.Text = "";
            label7.Text = "";
            label13.Text = "";
            label15.Text = "";
            refreshDatagrid(new List<Operation>() { new Operation(default(DateTime), "")
});
```

```
            refreshCbEditClient();
        }
        private void Admin_Load(object sender, EventArgs e)
        {
            this.Width = 866;
            this.Height = 426;
            clientsDictionary = Serealizator.desearealizable();
            if (clientsDictionary == null)
            {
                clientsDictionary = new Dictionary<uint, Client>();
            }
            tSS2.Text = "Прочитано клієнтів з файлу";
        }

        private void TabControl1_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (tabControl1.SelectedIndex == 0)
            {

                this.Width = 866;
                this.Height = 426;
                cbEditClient.SelectedItem = null;
            }
            else
            {
                clearAllFields();
                this.Width = 1302;
                this.Height = 880;
                cbEditClient.Items.Clear();
                if (clientsDictionary == null)
                {
                    showMes("Список пустий");
                    return;
                }
                refreshCbEditClient();
```

```
                refreshDatagrid(new List<Operation>() { new Operation(default(DateTime),
"") });
            }
        }

        private void Button5_Click(object sender, EventArgs e)
        {
            if (tbAddName.Text == "" || tbAddPrizv.Text == "" || tbAddSurname.Text == ""
|| tbAddPasport.Text == "" || tbAddPinKod.Text == "")
            {
                    showMes("Не всі рядки заповнено!");
                    return;
            }
            if (!tryToUint(tbAddPasport.Text) || !tryToUint(tbAddPinKod.Text))
            {
                    return;
            }
            if (clientsDictionary != null)
            {

                if (!clientsDictionary.Values.All(a => a.pasport !=
ToUInt(tbAddPasport.Text)))
                {
                    showMes("Такий клієнт вже існує");
                    return;
                }

                if (!clientsDictionary.Keys.All(a => a != ToUInt(tbAddPinKod.Text)))
                {
                    showMes("Такий пін Код вже існує");
                    return;
                }
            }
            Client client = new
Client(tbAddName.Text,tbAddPrizv.Text,tbAddSurname.Text,ToUInt(tbAddPasport.Text),new
Account(0,new Deposit("",0,0,default(DateTime),default(TimeSpan)),new
List<Operation>()));
            clientsDictionary.Add(ToUInt(tbAddPinKod.Text), client);
```

```
            Serealizator.serealizable(clientsDictionary);

            clearTb(tbAddName); clearTb(tbAddPrizv); clearTb(tbAddSurname);
clearTb(tbAddPasport); clearTb(tbAddPinKod);

            tSS2.Text = "Додано клієнта і записано у файл";

        }


        public void findClient(Client client)

        {

            tbEditName.Text = client.name;

            tbEditPrizv.Text = client.prizv;

            tbEditSurname.Text = client.surname;

            tbEditPasport.Text = client.pasport.ToString();

            tbEditPinKod.Text = clientsDictionary.FirstOrDefault(x => x.Value ==
client).Key.ToString();

            label7.Text = "Баланс " + client.account.balance.ToString();

            cbEditDeposit.Text = client.account.deposit.type;

            if (client.account.deposit.type == "Універсальний")

            {

                btEditClientMinusMoneyDeposit.Enabled = true;

                btEditClientPlusMoneyDeposit.Enabled = true;

            }

            else if (client.account.deposit.type == "Накопичувальний")

            {

                btEditClientMinusMoneyDeposit.Enabled = false;

                btEditClientPlusMoneyDeposit.Enabled = true;

            }

            else

            {

                btEditClientMinusMoneyDeposit.Enabled = false;

                btEditClientPlusMoneyDeposit.Enabled = false;

            }

            if (client.account.deposit.type == "")

            {

                cbEditDeposit.Enabled = true;

                btEditClientDeposit.Visible = true;

            }
```

```
            label13.Text = "Баланс " + client.account.deposit.balance.ToString();
            label15.Text = "Дата вкладу " +
client.account.deposit.dateOfDeposit.ToString();
            if (client.account.lastOperations.Count == 0)
            {
                refreshDatagrid(new List<Operation>() { new Operation(default(DateTime),
"") });
            }
            refreshDatagrid(client.account.lastOperations);
            tSS.Text = "Обрано " + tbEditPasport.Text.ToString();
        }
        private void CbEditClient_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (cbEditClient.SelectedItem == null)
            {
                return;
            }
            if (cbEditClient.Items.Count == 0)
            {
                showMes("Список клієнтів пустий");
                return;
            }
            Client client = clientsDictionary.Values.ToList().Find(a => a.pasport ==
ToUInt(cbEditClient.SelectedItem.ToString()));
            findClient(client);

        }
        private void BtEditFindClient_Click(object sender, EventArgs e)
        {
            string str = Interaction.InputBox("Введіть номер паспорта клієнта?", "", "");
            if(!tryToUint(str))
            {
                return;
            }
            uint pasport = ToUInt(str);
            Client client = clientsDictionary.Values.ToList().Find(a => a.pasport ==
pasport);
```

```
            findClient(client);


        }


        private void BtEditClient_Click(object sender, EventArgs e)
        {
            if (tbEditName.Text == "" || tbEditPrizv.Text == "" || tbEditSurname.Text ==
"" || tbEditPasport.Text == "" || tbEditPinKod.Text == "")
            {
                showMes("Не всі рядки заповнено!");
                return;
            }
            if (!tryToUint(tbEditPasport.Text) || !tryToUint(tbEditPinKod.Text))
            {
                return;
            }


            if (cbEditClient.SelectedItem == null)
            {
                return;
            }
            if (cbEditClient.Items.Count == 0)
            {
                showMes("Список клієнтів пустий");
                return;
            }
            Client client = clientsDictionary.Values.ToList().Find(a => a.pasport ==
ToUInt(cbEditClient.SelectedItem.ToString()));
            client.name = tbEditName.Text;
            client.prizv = tbEditPrizv.Text;
            client.surname = tbEditSurname.Text;
            client.pasport = ToUInt(tbEditPasport.Text);
            uint key = clientsDictionary.FirstOrDefault(x => x.Value == client).Key;
            clientsDictionary.Remove(key);
            clientsDictionary.Add(ToUInt(tbEditPinKod.Text), client);
```

```
            Serealizator.serealizable(clientsDictionary);
            clearAllFields();
            refreshCbEditClient();
            tSS.Text = "Віредаговано";
        }

        private void BtEditClientMinusMoney_Click(object sender, EventArgs e)
        {
            if (cbEditClient.SelectedItem == null)
            {
                showMes("Оберіть клієнта");
                return;
            }
            if (cbEditClient.Items.Count == 0)
            {
                showMes("Список клієнтів пустий");
                return;
            }
            Client client = clientsDictionary.Values.ToList().Find(a => a.pasport ==
ToUInt(cbEditClient.SelectedItem.ToString()));
            uint key = clientsDictionary.FirstOrDefault(x => x.Value == client).Key;
            string suma = Interaction.InputBox("Введіть суму", "", "");
            if (!tryToUint(suma))
            {
                return;
            }

            if (Convert.ToDouble(suma) == 0)
            {
                showMes("Не може бути 0");
                return;
            }
            if (client.account.balance.Equals(0) || ToUInt(suma) >
client.account.balance)
            {
                showMes("Недостатньо коштів на рахунку");
```

```
                return;
            }
            else
            {
                client.account.minusMoney(Convert.ToDouble(suma));
                client.account.lastOperations.Add(new Operation(DateTime.Now,("-"+suma +
" UAH")));
                clientsDictionary[key] = client;
                Serealizator.serealizable(clientsDictionary);
                label7.Text = "Баланс " + client.account.balance.ToString();
                refreshDatagrid(client.account.lastOperations);
                tSS.Text = "Знято" + suma + " UAH";
            }
        }

        private void BtEditClientPlusMoney_Click(object sender, EventArgs e)
        {
            if (cbEditClient.SelectedItem == null)
            {
                showMes("Оберіть клієнта");
                return;
            }
            if (cbEditClient.Items.Count == 0)
            {
                showMes("Список клієнтів пустий");
                return;
            }
            Client client = clientsDictionary.Values.ToList().Find(a => a.pasport ==
ToUInt(cbEditClient.SelectedItem.ToString()));
            uint key = clientsDictionary.FirstOrDefault(x => x.Value == client).Key;
            string suma = Interaction.InputBox("Введіть суму", "", "");
            if (!tryToUint(suma))
            {
                return;
            }
            if (Convert.ToDouble(suma) == 0)
```

```
            {
                showMes("Не може бути 0");
                return;
            }
            client.account.plusMoney(Convert.ToDouble(suma));
             client.account.lastOperations.Add(new Operation(DateTime.Now, ("+"+suma + "
UAH")));
             clientsDictionary[key] = client;
             Serealizator.serealizable(clientsDictionary);
             label7.Text = "Баланс " + client.account.balance.ToString();
             refreshDatagrid(client.account.lastOperations);
             tSS.Text = "Внесено на баланс" + suma + " UAH";
        }

        private void CbEditDeposit_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (cbEditDeposit.Text == "Універсальний")
            {
                btEditClientMinusMoneyDeposit.Enabled = true;
                btEditClientPlusMoneyDeposit.Enabled = true;
            }
            else if (cbEditDeposit.Text == "Накопичувальний")
            {
                btEditClientMinusMoneyDeposit.Enabled = false;
                btEditClientPlusMoneyDeposit.Enabled = true;
            }
            else
            {
                btEditClientMinusMoneyDeposit.Enabled = false;
                btEditClientPlusMoneyDeposit.Enabled = false;
            }

        }

        private void BtEditClientDeposit_Click(object sender, EventArgs e)
        {
```

```
            if (cbEditClient.SelectedItem == null)
            {
                showMes("Оберіть клієнта");
                return;
            }
            if (cbEditClient.Items.Count == 0)
            {
                showMes("Список клієнтів пустий");
                return;
            }
            Client client = clientsDictionary.Values.ToList().Find(a => a.pasport ==
ToUInt(cbEditClient.SelectedItem.ToString()));
            uint key = clientsDictionary.FirstOrDefault(x => x.Value == client).Key;
            if (cbEditDeposit.Text == "")
            {
                showMes("Оберіть тип депозиту!");
                return;
            }
            client.account.deposit.type = cbEditDeposit.Text;
            client.account.deposit.initStavka(cbEditDeposit.Text);
            client.account.deposit.dateOfDeposit = DateTime.Now;


            string interval = Interaction.InputBox("Введіть кількість місяців: на скільки
ви хочете покласти: ", "", "");
            if (!tryToUint(interval))
            {
                return;
            }
            if (Convert.ToDouble(interval) == 0)
            {
                showMes("Не може бути 0");
                return;
            }
            TimeSpan interv = new TimeSpan((int)(ToUInt(interval)*31),0,0,0);
            client.account.deposit.interval = interv;
            string suma = Interaction.InputBox("Введіть суму яку ви хочете покласти: ",
"", "");
```

```
            if (!tryToUint(suma))
            {
                return;
            }
            if (Convert.ToDouble(suma) == 0)
            {
                showMes("Не може бути 0");
                return;
            }
            client.account.deposit.balance = Convert.ToDouble(suma);
            client.account.lastOperations.Add(new Operation(DateTime.Now, ("(Депозит) +"
+ suma + " UAH")));
            clientsDictionary[key] = client;
            Serealizator.serealizable(clientsDictionary);
            label13.Text = "Баланс " + client.account.deposit.balance.ToString();
            refreshDatagrid(client.account.lastOperations);
            tSS.Text = "Внесено на депозит" + suma + " UAH";
            label15.Text = client.account.deposit.dateOfDeposit.ToString();
            btEditClientDeposit.Visible = false;
            cbEditDeposit.Enabled = false;
        }

        private void Button1_Click(object sender, EventArgs e)
        {
            if (cbEditClient.SelectedItem == null)
            {
                showMes("Оберіть клієнта");
                return;
            }
            if (cbEditClient.Items.Count == 0)
            {
                showMes("Список клієнтів пустий");
                return;
            }
            Client client = clientsDictionary.Values.ToList().Find(a => a.pasport ==
ToUInt(cbEditClient.SelectedItem.ToString()));
```

```
            uint key = clientsDictionary.FirstOrDefault(x => x.Value == client).Key;
            clientsDictionary.Remove(key);
            Serealizator.serealizable(clientsDictionary);
            clearAllFields();
            refreshCbEditClient();
            tSS.Text = "Видалено";
        }

        private void BtEditClientMinusMoneyDeposit_Click(object sender, EventArgs e)
        {
            if (cbEditClient.SelectedItem == null)
            {
                showMes("Оберіть клієнта");
                return;
            }
            if (cbEditClient.Items.Count == 0)
            {
                showMes("Список клієнтів пустий");
                return;
            }
            Client client = clientsDictionary.Values.ToList().Find(a => a.pasport ==
ToUInt(cbEditClient.SelectedItem.ToString()));
            uint key = clientsDictionary.FirstOrDefault(x => x.Value == client).Key;
            string suma = Interaction.InputBox("Введіть суму : ", "", "");
            if (!tryToUint(suma))
            {
                return;
            }
            if (Convert.ToDouble(suma) > client.account.balance)
            {
                showMes("недостатньо коштів на депозиті");
                return;
            }
            if (Convert.ToDouble(suma) == 0)
            {
                showMes("Не може бути 0");
```

```
                return;
            }
            client.account.deposit.minusMoney(Convert.ToDouble(suma));
            client.account.lastOperations.Add(new Operation(DateTime.Now, ("(Депозит) -"
+ suma + " UAH")));
            clientsDictionary[key] = client;
            Serealizator.serealizable(clientsDictionary);
            label13.Text = "Баланс " + client.account.deposit.balance.ToString();
            refreshDatagrid(client.account.lastOperations);
            tSS.Text = "Знято з депозиту на депозит " + suma + " UAH";
        }

        private void BtEditClientPlusMoneyDeposit_Click(object sender, EventArgs e)
        {
            if (cbEditClient.SelectedItem == null)
            {
                showMes("Оберіть клієнта");
                return;
            }
            if (cbEditClient.Items.Count == 0)
            {
                showMes("Список клієнтів пустий");
                return;
            }
            Client client = clientsDictionary.Values.ToList().Find(a => a.pasport ==
ToUInt(cbEditClient.SelectedItem.ToString()));
            uint key = clientsDictionary.FirstOrDefault(x => x.Value == client).Key;
            string suma = Interaction.InputBox("Введіть суму : ", "", "");
            if (!tryToUint(suma))
            {
                return;
            }

            if (Convert.ToDouble(suma) == 0)
            {
                showMes("Не може бути 0");
```

```
                return;
            }
            client.account.deposit.plusMoney(Convert.ToDouble(suma));
            client.account.lastOperations.Add(new Operation(DateTime.Now, ("(Депозит) +"
+ suma + " UAH")));
            clientsDictionary[key] = client;
            Serealizator.serealizable(clientsDictionary);
            label13.Text = "Баланс " + client.account.deposit.balance.ToString();
            refreshDatagrid(client.account.lastOperations);
            tSS.Text = "Внесено на депозит " + suma + " UAH";
        }

        private void TbAddPasport_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar !=
'.'))
            {
                e.Handled = true;
            }
        }

        private void TbAddPinKod_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar !=
'.'))
            {
                e.Handled = true;
            }
        }

        private void TbEditPasport_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar !=
'.'))
            {
                e.Handled = true;
            }
```

```
        }


        private void TbEditPinKod_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar != '.'))
            {
                e.Handled = true;
            }
        }
    }

}
```

User.cs

```
namespace kursova
{
    public partial class User : Form
    {
        Dictionary<uint, Client> clientsDictionary;
        Client client;
        uint key;
        public User(Dictionary<uint, Client> dictionary, Client client, uint key)
        {
            clientsDictionary = dictionary;
            this.client = client;
            this.key = key;
            InitializeComponent();
        }
        private uint ToUInt(String str)
        {
            return Convert.ToUInt32(str);
        }
        public bool tryToUint(string str)
        {
            try
            {
```

```
            Convert.ToUInt32(str);
            return true;
        }
        catch (Exception)
        {
            showMes("Некоректне введення");
            return false;
        }
    }
    public void showMes(String str)
    {
        MessageBox.Show(str, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    private void refreshDatagrid(List<Operation> operationList)
    {
        if (operationList.Count == 0)
        {
            showMes("Список операцій пустий пустий");
            return;
        }
        dataGridView1.RowCount = operationList.Count;
        int i = 0;
        foreach (Operation a in operationList)
        {
            dataGridView1[0, i].Value = a.time.ToString();
            dataGridView1[1, i].Value = a.sum.ToString();
            i++;
        }
    }

    private void User_Load(object sender, EventArgs e)
    {
        tbEditName.Text = client.name;
        tbEditPrizv.Text = client.prizv;
        tbEditSurname.Text = client.surname;
        tbEditPasport.Text = client.pasport.ToString();
```

```
            tbEditPinKod.Text = clientsDictionary.FirstOrDefault(x => x.Value ==
client).Key.ToString();

            label7.Text = "Баланс " + client.account.balance.ToString();

            label8.Text = client.account.deposit.type;


            label13.Text = "Баланс " + client.account.deposit.balance.ToString();

            label4.Text = "Дата вкладу " +
client.account.deposit.dateOfDeposit.ToString();

            if (client.account.lastOperations.Count == 0)

            {

                refreshDatagrid(new List<Operation>() { new Operation(default(DateTime),
"") });

            }

            refreshDatagrid(client.account.lastOperations);

            tSS2.Text = "Обрано " + tbEditPasport.Text.ToString();

        }


        private void кінецьToolStripMenuItem_Click(object sender, EventArgs e)

        {

            Application.Exit();

        }


        private void повернутисьДоLogInToolStripMenuItem_Click(object sender, EventArgs
e)

        {

            LogIn log = new LogIn();

            log.Show();

            this.Hide();

        }


        private void Button2_Click(object sender, EventArgs e)

        {

            string suma = Interaction.InputBox("Введіть суму", "", "");

            if (!tryToUint(suma))

            {

                return;

            }
```

```
            if (Convert.ToDouble(suma) == 0)
            {
                showMes("Не може бути 0");
                return;
            }
            if (client.account.balance.Equals(0) || ToUInt(suma) >
client.account.balance)
            {
                showMes("Недостатньо коштів на рахунку");
                return;
            }
            else
            {
                client.account.minusMoney(Convert.ToDouble(suma));
                client.account.lastOperations.Add(new Operation(DateTime.Now, ("-" + suma
+ " UAH")));
                clientsDictionary[key] = client;
                Serealizator.serealizable(clientsDictionary);
                label7.Text = "Баланс " + client.account.balance.ToString();
                refreshDatagrid(client.account.lastOperations);
                tSS2.Text = "Знято" + suma + " UAH";
            }
        }

        private void Button3_Click(object sender, EventArgs e)
        {
            string suma = Interaction.InputBox("Введіть суму", "", "");
            if (!tryToUint(suma))
            {
                return;
            }
            if (Convert.ToDouble(suma) == 0)
            {
                showMes("Не може бути 0");
                return;
```

```
            }
            client.account.plusMoney(Convert.ToDouble(suma));
            client.account.lastOperations.Add(new Operation(DateTime.Now, ("+" + suma + "
UAH")));
            clientsDictionary[key] = client;
            Serealizator.serealizable(clientsDictionary);
            label7.Text = "Баланс " + client.account.balance.ToString();
            refreshDatagrid(client.account.lastOperations);
            tSS2.Text = "Внесено на баланс" + suma + " UAH";
        }
    }
}
```

LogIn.cs

```
namespace Kursova
{
    public partial class LogIn : Form
    {
        public LogIn()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Dictionary<uint, Client> clientsDictionary = Serealizator.desearealizable();
            if (tb1.Text == "")
            {
                MessageBox.Show("Заповніть поле з паролем!", "Ошибка!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            if (tb1.Text == "0000")
            {
                Admin admin = new Admin();
                admin.Show();
                this.Hide();
                return;
```

```
            }
            Client client = null;
            try
            {
               client = clientsDictionary[Convert.ToUInt32(tb1.Text)];
            }
            catch
            {
                MessageBox.Show("Такого пін коду не знайдено!", "Ошибка!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                tb1.Text = "";
                return;
            }
            User user = new User(clientsDictionary,client, Convert.ToUInt32(tb1.Text));
            user.Show();
            this.Hide();
        }

        private void Tb1_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar !=
'.'))
            {
                e.Handled = true;
            }
        }

        private void CheckBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked == true)
            {
                tb1.PasswordChar = '\0';
            }
            else
            {
                tb1.PasswordChar = '*';
```

```
            }
        }
    }
}
```

Program.cs

```
namespace Kursova
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LogIn());
        }
    }
}
```