ЗМІСТ

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 Загальний розділ

1.1 Постановка завдання

Темою курсового проекту є програмний продукт “Моделювання роботи банку «Інтелект», моделювання банкомату, вклади і види вкладів, реєстр рахунків фізичних та юридичних осіб”

Метою даного проекту є створення інтерактивного середовища навчального процесу,

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.

1.2 Опис вхідної інформації

Вхідна інформація передбачає певні дані, які необхідні для входу в програму та роботи з особистим рахунком клієнта. Проаналізувавши

предметну область і вимоги до програми, можна сказати, що даний проект повинен мати дві гілки реалізації: користувача, представленого у ролі адміністратора або клієнта банку.

Для роботи клієнта банку вхідною інформацією є сума яку користувач планує зняти з особистого рахунку чи депозиту, пін-код.

Для роботи адмінстратора банку вхідною інформацією є прізвище, ім’я, по-батькові, номер паспорта, пін-код користувача, сума для зняття/внесення на рахунок, обрання типу депозиту.

1.3 Опис результуючої інформації

Результуючу інформацію ми отримуємо після виконання певної функції проекту.

Ввівши пін-код отримаємо відповідну форму з відповідним особистим рахунком.

При редагуванні даних побачимо відредаговані данні на екрані.

При знятті/поповненні коштів отримаємо оновлений рахунок тасписок останніх операцій.

1.4 Стадії та етапи розробки задачі

У ході розробки програмного забезпечення було виділено такі етапи:

1) визначення мети проекту та завдання;

2) аналіз предметної області – будується система у вигляді трьох взаємозалежних моделей;

- об’єктна модель задачі – визначення класів та об’єктів
- динамічна модель задачі – опис змін, що відбуваються з об’єктами та їх зв’язками під час роботи системи;
- функціональна модель задачі – визначення вхідних та результуючих даних, опис функцій та опис обмежень;

3) вибір програм для розробки програмного коду та тестування;

4) створення графічного інтерфейсу програми;

5) програмування;

6) проектування структури вхідних та вихідних даних;

7) методи організації вхідних та вихідних даних.

Визначення мети проекту і його завдання було здійснено ще на початковій стадії планування та вибору теми курсового проекту.

На етапі аналізу предметної області було визначено класи, опис змін, що відбуваються з об’єктами та їх зв’язками, визначення вхідних та вихідних даних.

Особливу увагу приділено розміщенню елементів експерименту. Так як зі збільшенням їх кількості на формі розмір елементів зменшується, що неабияк важливо з естетичної точки зору проекту.

1.5 Опис існуючих методів та 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 , де також можливо лише за натисканням клавіші поповнити рахунок, відредагувати особисту інформацію чи змінити пін-код.

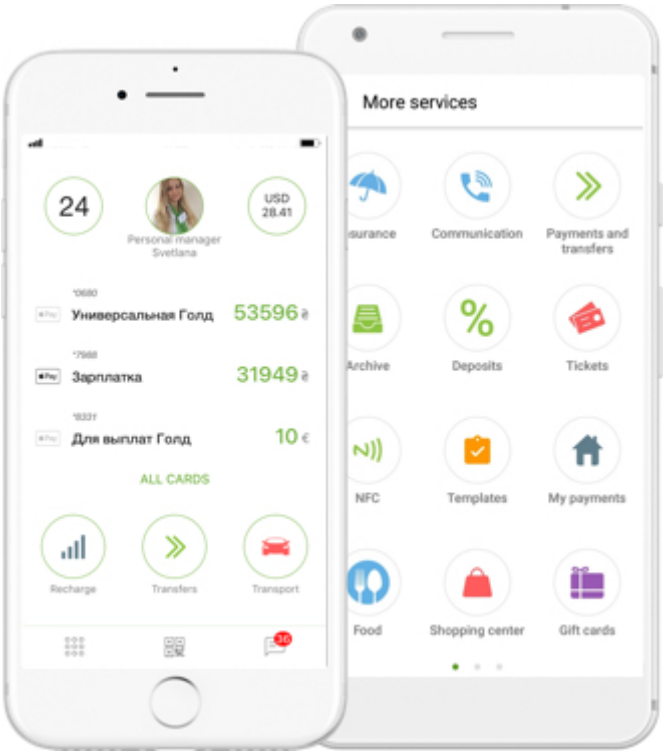


Рисунок 1.1. – Вигляд додатку – аналогу

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.

Серіалізація представляє процес перетворення будь-якого об'єкта в потік байтів. Після перетворення ми можемо цей потік байтів або записати на диск або зберегти його тимчасово в пам'яті. А при необхідності можна виконати зворотний процес - десеріалізацію, тобто отримати з потоку байтів раніше збережений об'єкт.

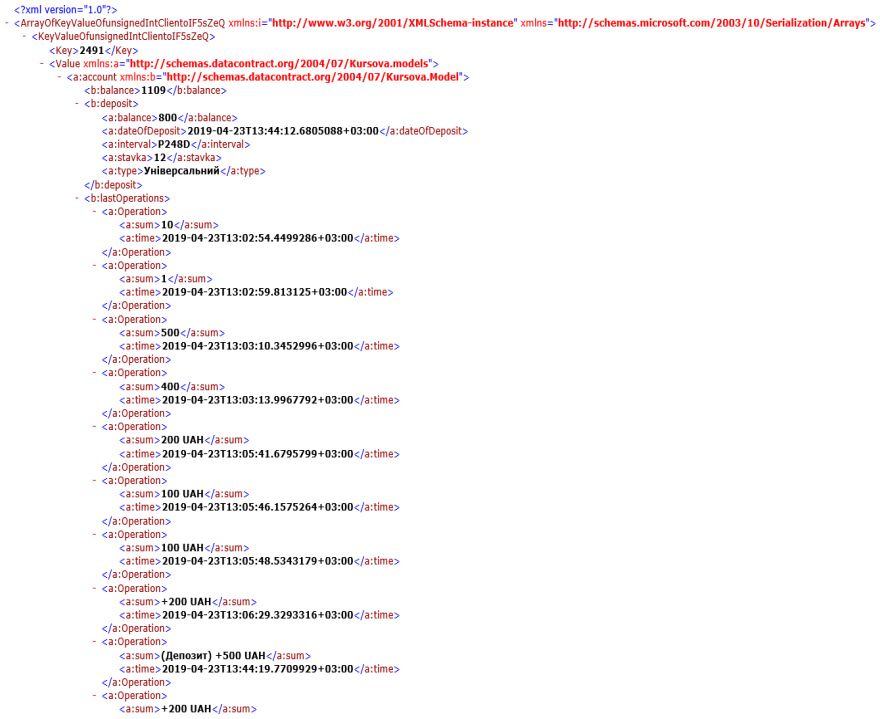
2 Розробка технічного та робочого проекту

2.1 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних

Головними об’єктами проекту є користувач, елемент моделювання.

Всі данні представленні в текстовому форматі а такі данні наприклад як останні операції по особистому рахунку представленні в табличному форматі для зручного сприйняття.

Всі вихідні данні які бачить користувач також у текстовому форматі, зміни які відбулись за особистим рахунком зберігаються до XML файлу.



```
<?xml version="1.0"?>
- <ArrayOfKeyValueOfunsignedIntClientoIF5sZeQ xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
  - <KeyValueOfunsignedIntClientoIF5sZeQ>
      <Key>2491</Key>
    - <Value xmlns:a="http://schemas.datacontract.org/2004/07/Kursova.models">
      - <a:account xmlns:b="http://schemas.datacontract.org/2004/07/Kursova.Model">
          <b:balance>1109</b:balance>
        - <b:deposit>
            <a:balance>800</a:balance>
            <a:dateOfDeposit>2019-04-23T13:44:12.6805088+03:00</a:dateOfDeposit>
            <a:interval>P248D</a:interval>
            <a:stavka>12</a:stavka>
            <a:type>Універсальний</a:type>
          </b:deposit>
        - <b:lastOperations>
          - <a:Operation>
              <a:sum>10</a:sum>
              <a:time>2019-04-23T13:02:54.4499286+03:00</a:time>
            </a:Operation>
          - <a:Operation>
              <a:sum>1</a:sum>
              <a:time>2019-04-23T13:02:59.813125+03:00</a:time>
            </a:Operation>
          - <a:Operation>
              <a:sum>500</a:sum>
              <a:time>2019-04-23T13:03:10.3452996+03:00</a:time>
            </a:Operation>
          - <a:Operation>
              <a:sum>400</a:sum>
              <a:time>2019-04-23T13:03:13.9967792+03:00</a:time>
            </a:Operation>
          - <a:Operation>
              <a:sum>200 UAH</a:sum>
              <a:time>2019-04-23T13:05:41.6795799+03:00</a:time>
            </a:Operation>
          - <a:Operation>
              <a:sum>100 UAH</a:sum>
              <a:time>2019-04-23T13:05:46.1575264+03:00</a:time>
            </a:Operation>
          - <a:Operation>
              <a:sum>100 UAH</a:sum>
              <a:time>2019-04-23T13:05:48.5343179+03:00</a:time>
            </a:Operation>
          - <a:Operation>
              <a:sum>+200 UAH</a:sum>
              <a:time>2019-04-23T13:06:29.3293316+03:00</a:time>
            </a:Operation>
          - <a:Operation>
              <a:sum>(Депозит) +500 UAH</a:sum>
              <a:time>2019-04-23T13:44:19.7709929+03:00</a:time>
            </a:Operation>
          - <a:Operation>
              <a:sum>+200 UAH</a:sum>
```


Рисунок 2.1. – XML файл з данними

2.2 Розробка структурної схеми проекту та об’єктної моделі задачі на основі UML діаграм

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),** що демонструє загальний функціонал системи та її використання користувачами (акторами).

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.

Рисунок 2.2. – Діаграма варіантів використання для клієнта банку

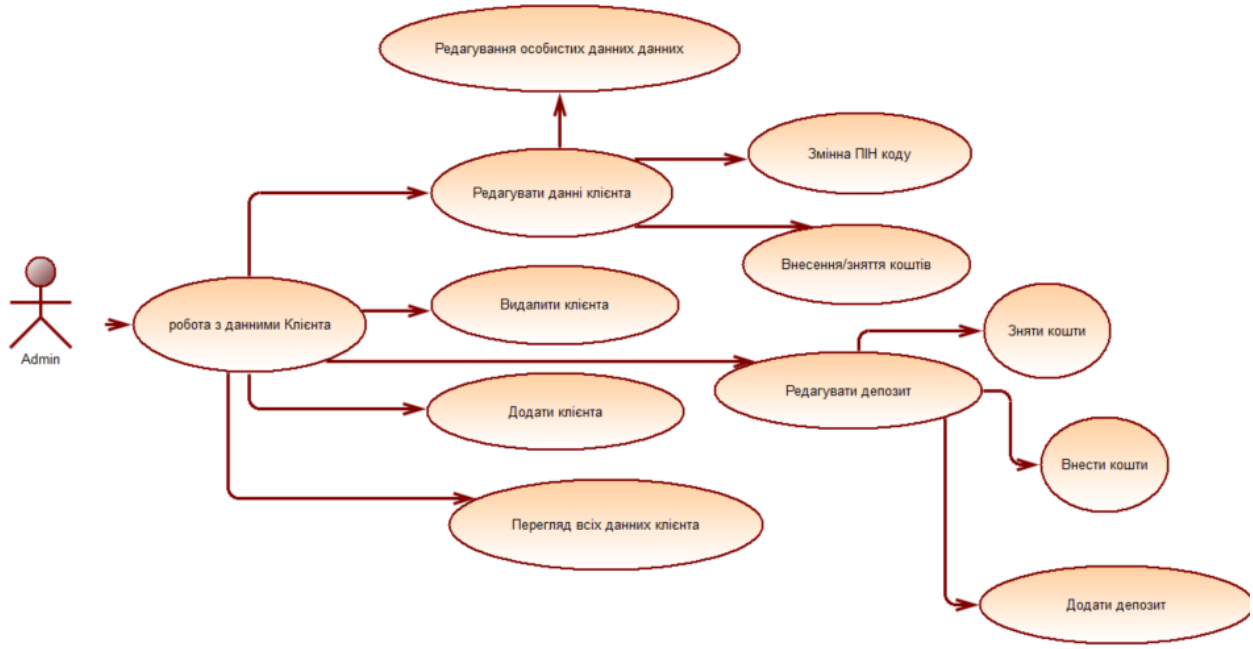


Рисунок 2.3. – Діаграма варіантів використання для адміністратора

Діаграма класів **(Class Diagram)** – статичне представлення структури моделі. Відображає статичні елементи, такі як: класи, типи даних, їх зміст та відношення.

На діаграмі 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.

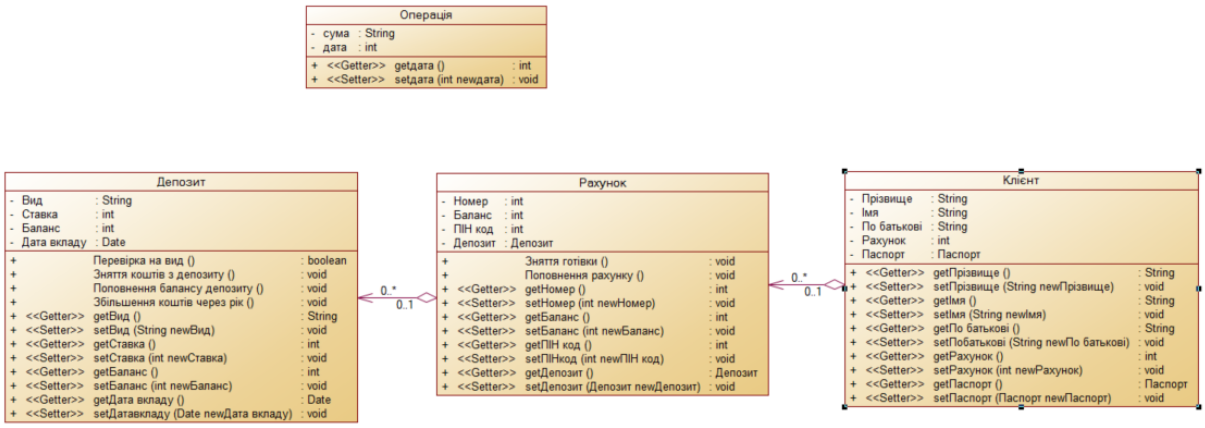


Рисунок 2.7. – Діаграма класів

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.

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.

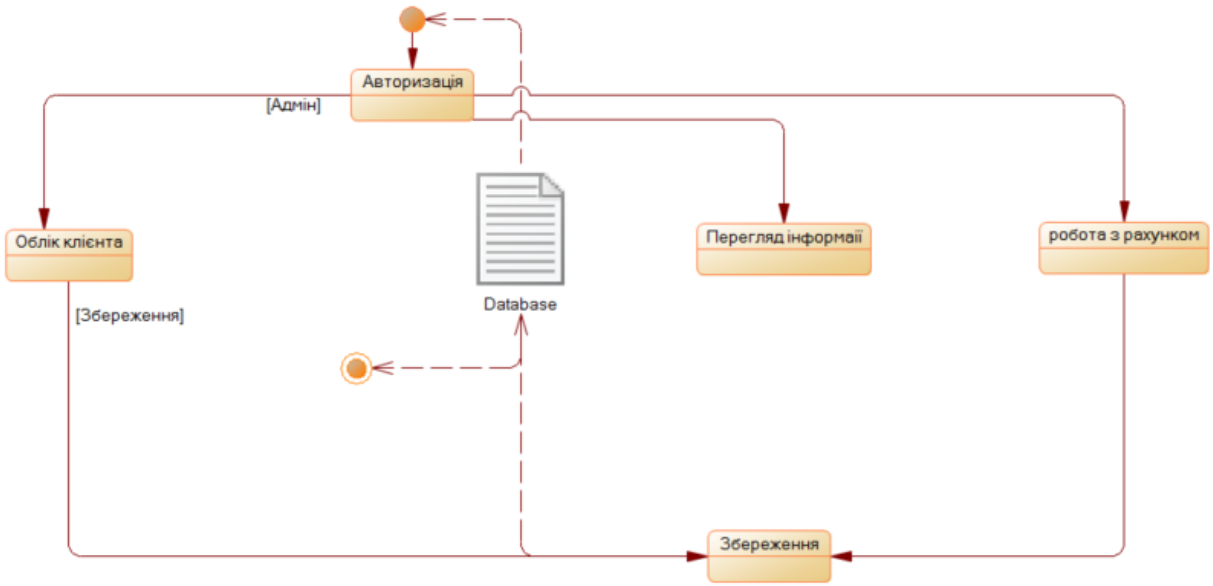


Рисунок 2.4. – Діаграма станів

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

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.

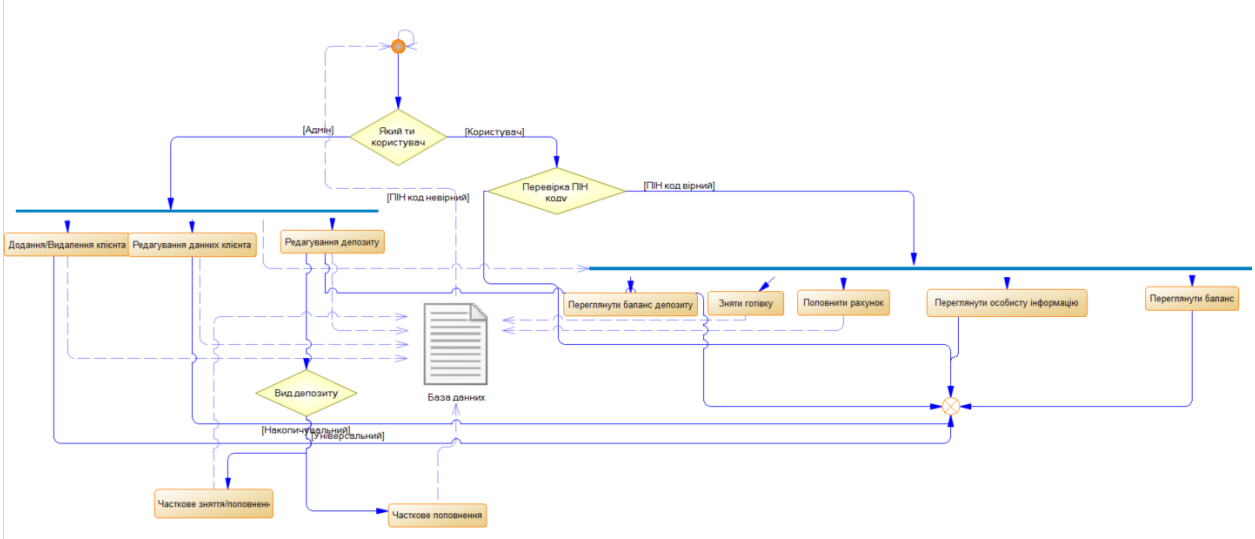


Рисунок 2.5. – Діаграма активності

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

Спершу розглянемо діаграму послідовності для клієнта банку. 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.

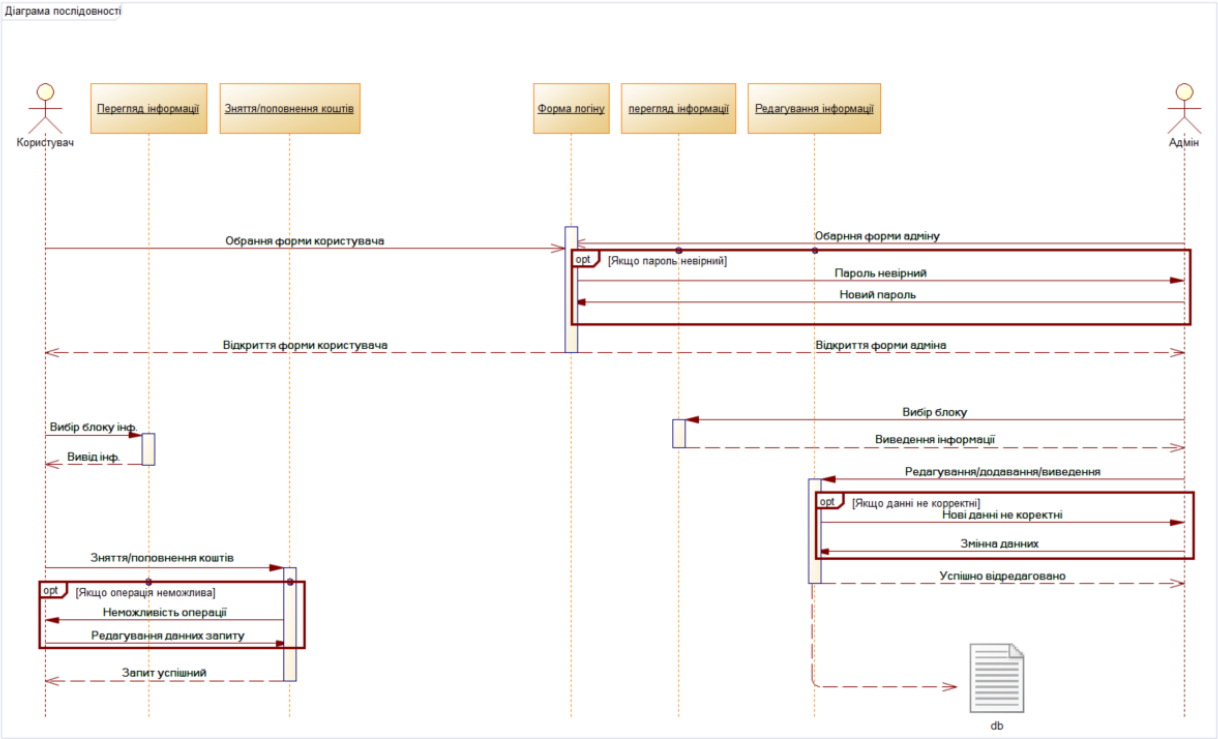


Рисунок 2.6. – Діаграма послідовності для клієнта банку і Адміністратора

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

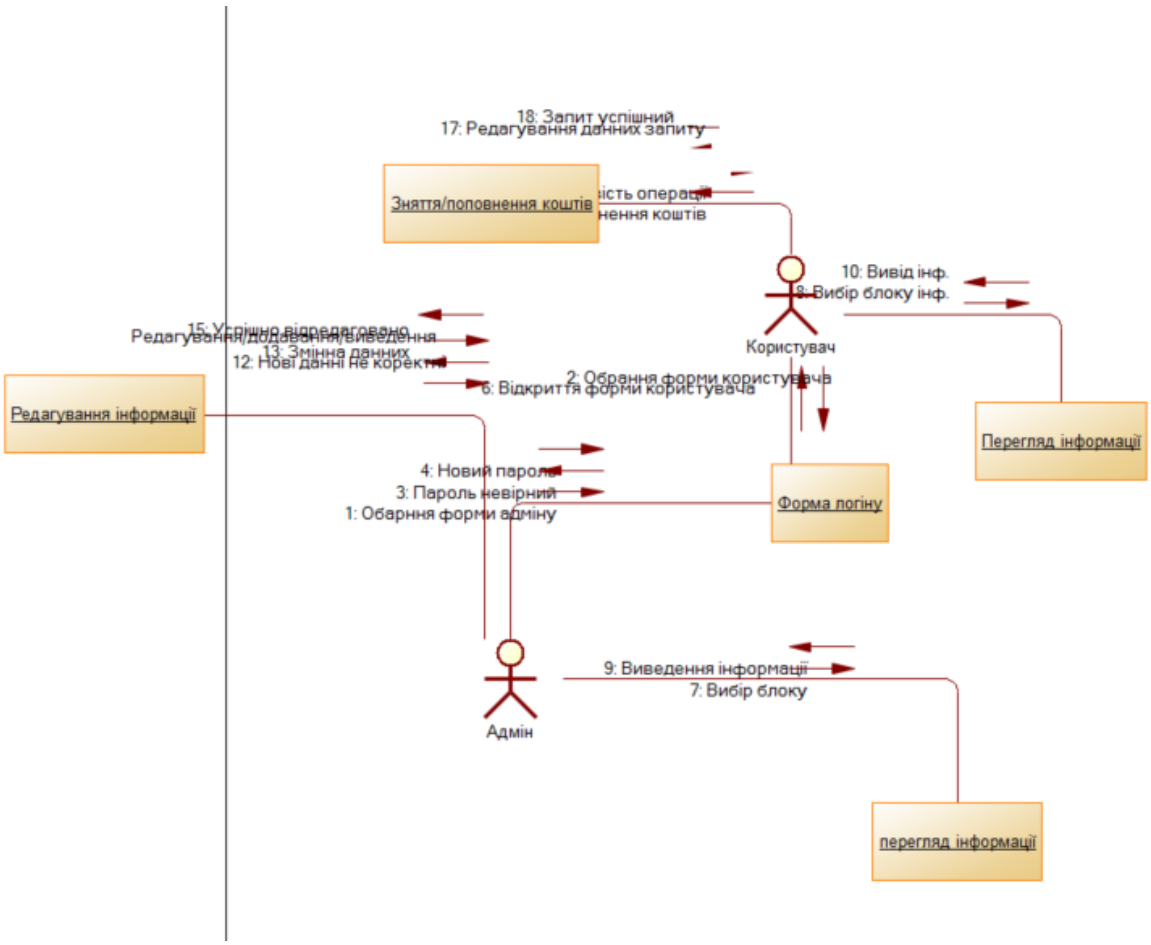


Рисунок 2.7. – Діаграма комунікації для Викладача

Щоб створити діаграму комунікації в середовищі PowerDesigner, варто лише викликати контекстне меню на діаграмі послідовності і вона створиться автоматично.

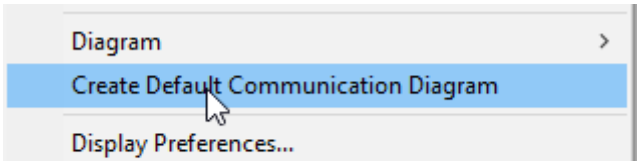


Рисунок 2.8. – Автоматичне створення діаграми комунікації

Діаграма компонентів **(Component Diagram)** описує фізичні особливості представлення системи.

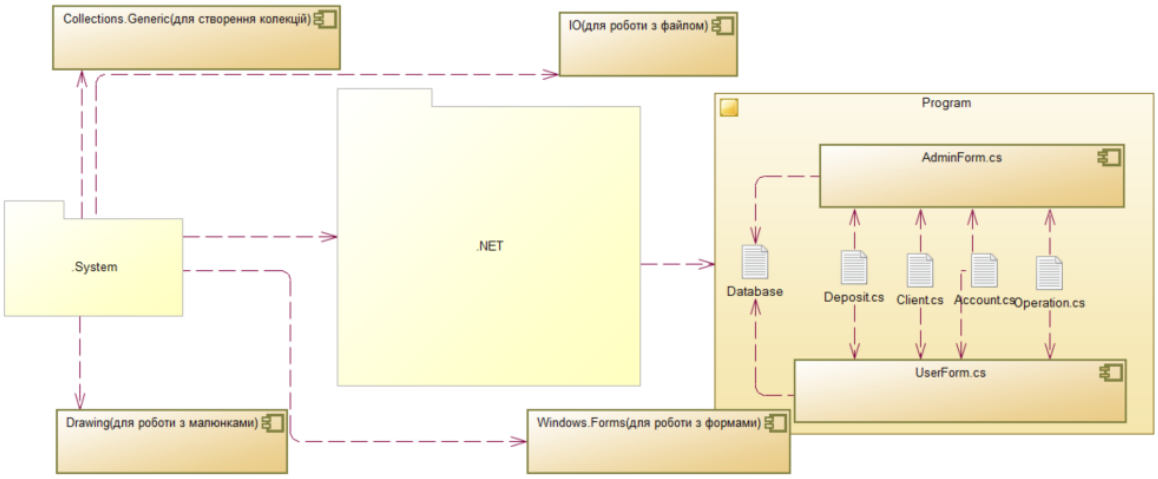


Рисунок 2.9. – Діаграма компонентів

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

Діаграма розгортання **(Deployment Diagram)** призначена для візуалізації елементів і компонентів програми, що існують лише на етапі її виконання.

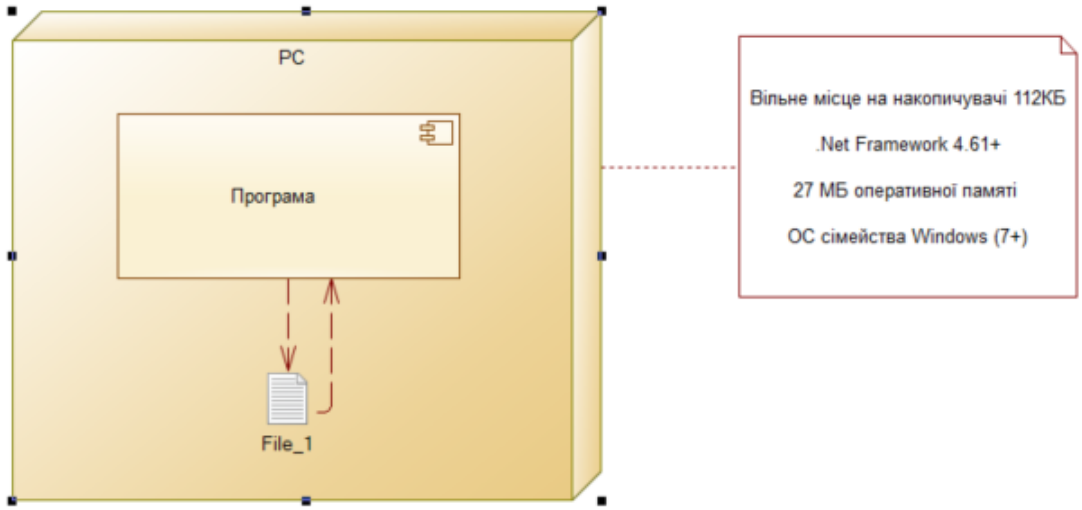


Рисунок 2.10. – Діаграма розгортання

Головним компонентом діаграми є вершини, наприклад сервер, комп'ютер…Програма не має мережевого зв'язку з будь-якими іншими пристроями, тому тут наявна лише одна вершина – комп'ютер. На ньому

збережена безпосередньо сама програма, і файли, які вона може використовувати під час роботи.

2.3 Об'єктно-орієнтовне проектування та програмування

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 дівою кнопкою миші на пункт меню Проект і обрати створення нового класу:

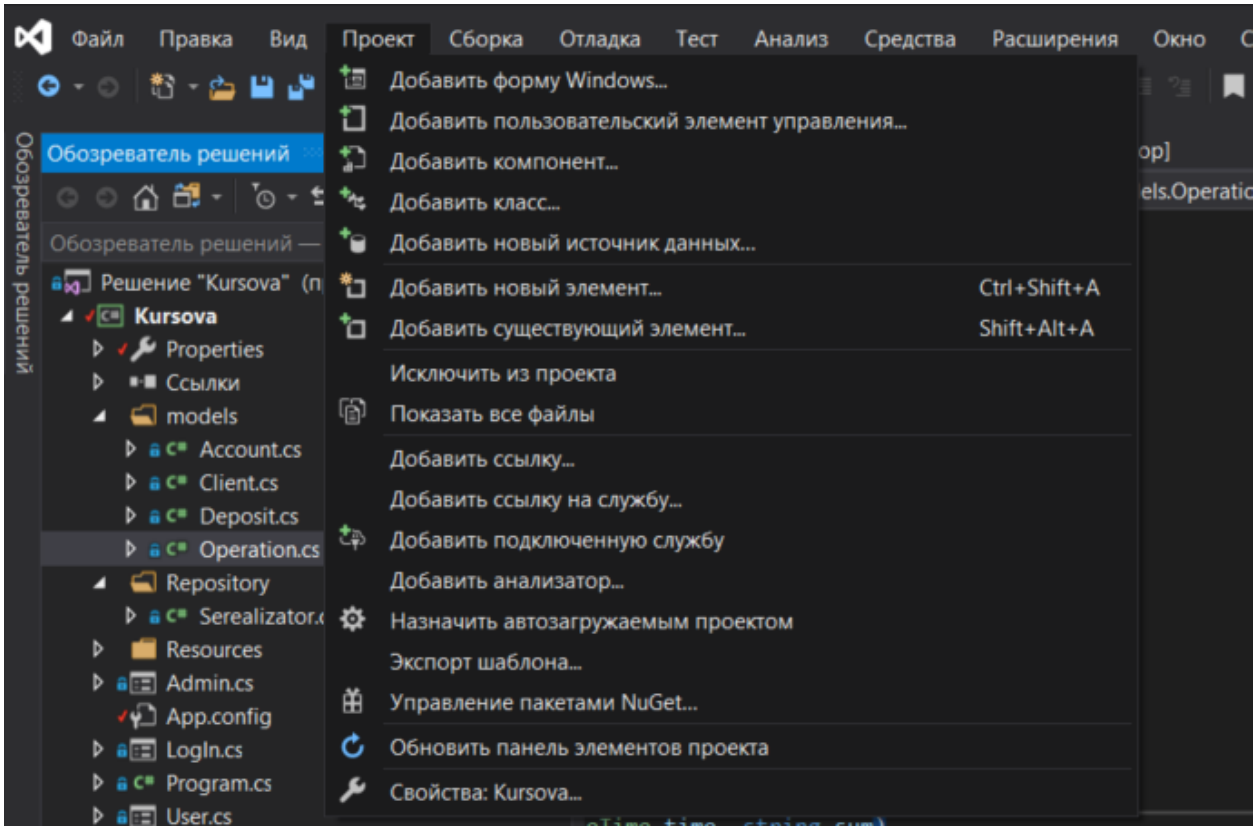


Рисунок 2.11. – Створення нового компонента

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

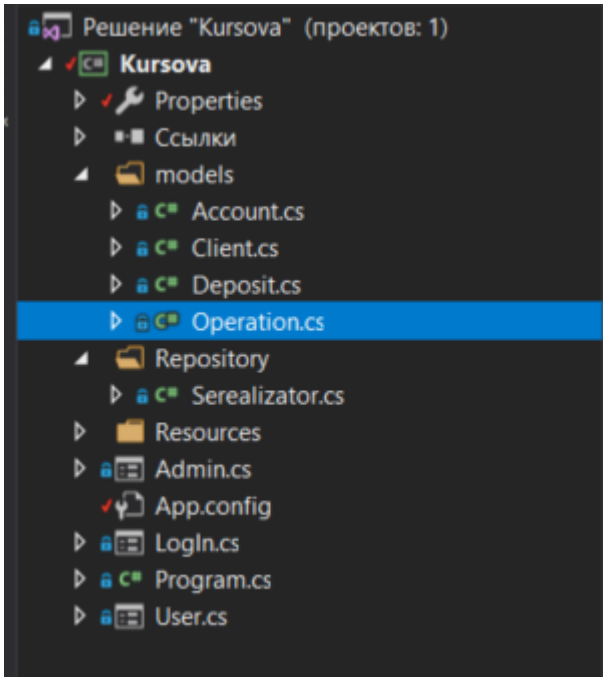


Рисунок 2.12. – Перелік класів і форм програми

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 Спеціальний розділ

3.1 Інструкція з інсталяції розробленого проекту

Спочатку потрібно переконатися в тому, що пристрій буде підтримувати дане програмне забезпечення:

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.

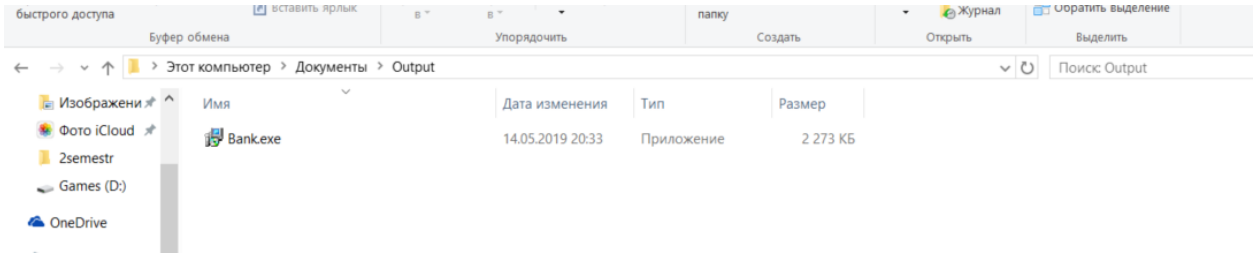


Рисунок 3.1. – Установочний файл

Вибір мови, натискаємо ОК.

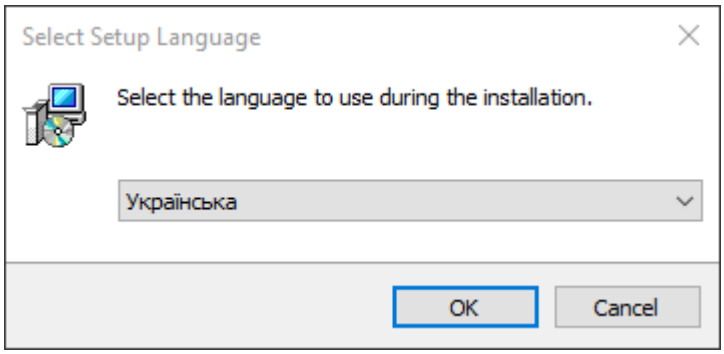


Рисунок 3.2. – Встановлення програми

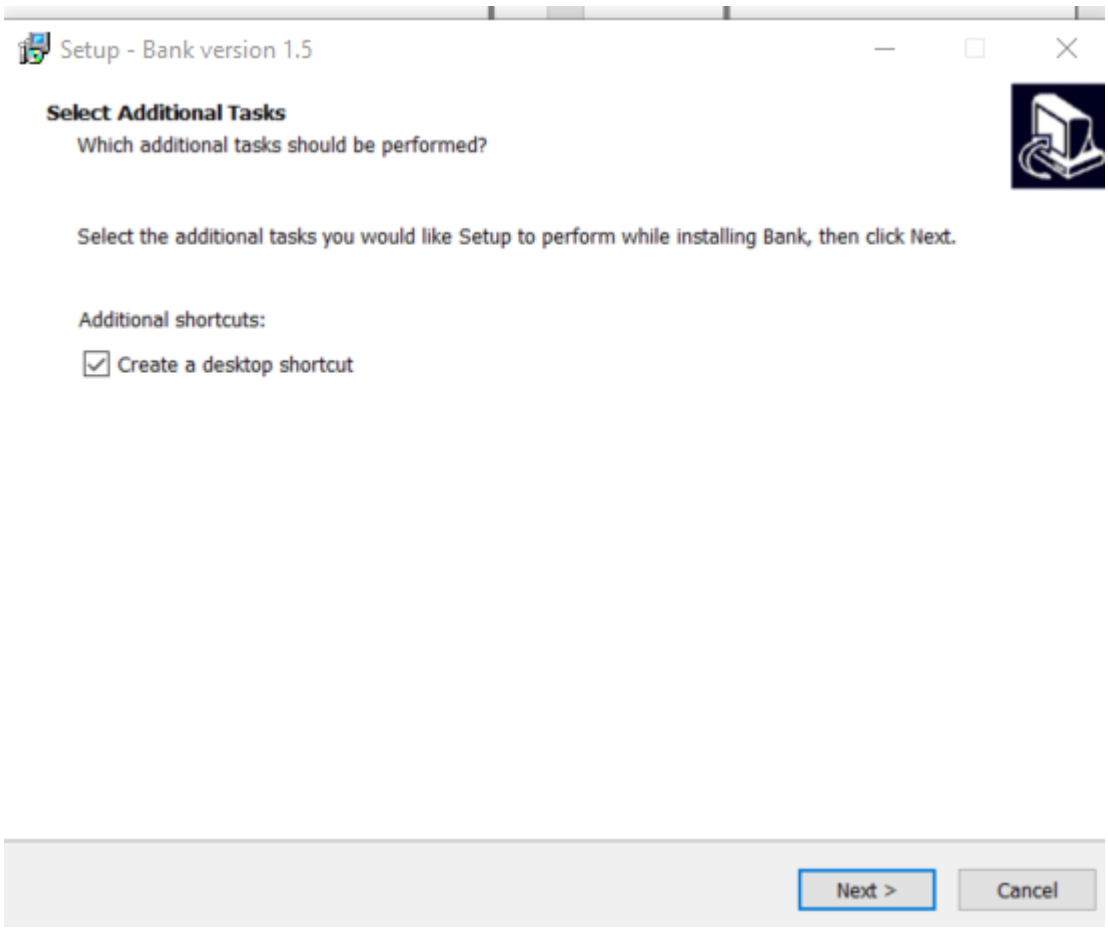


Рисунок 3.3. – Встановлення програми

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

Перевіряємо дані, натискаємо Встановити.

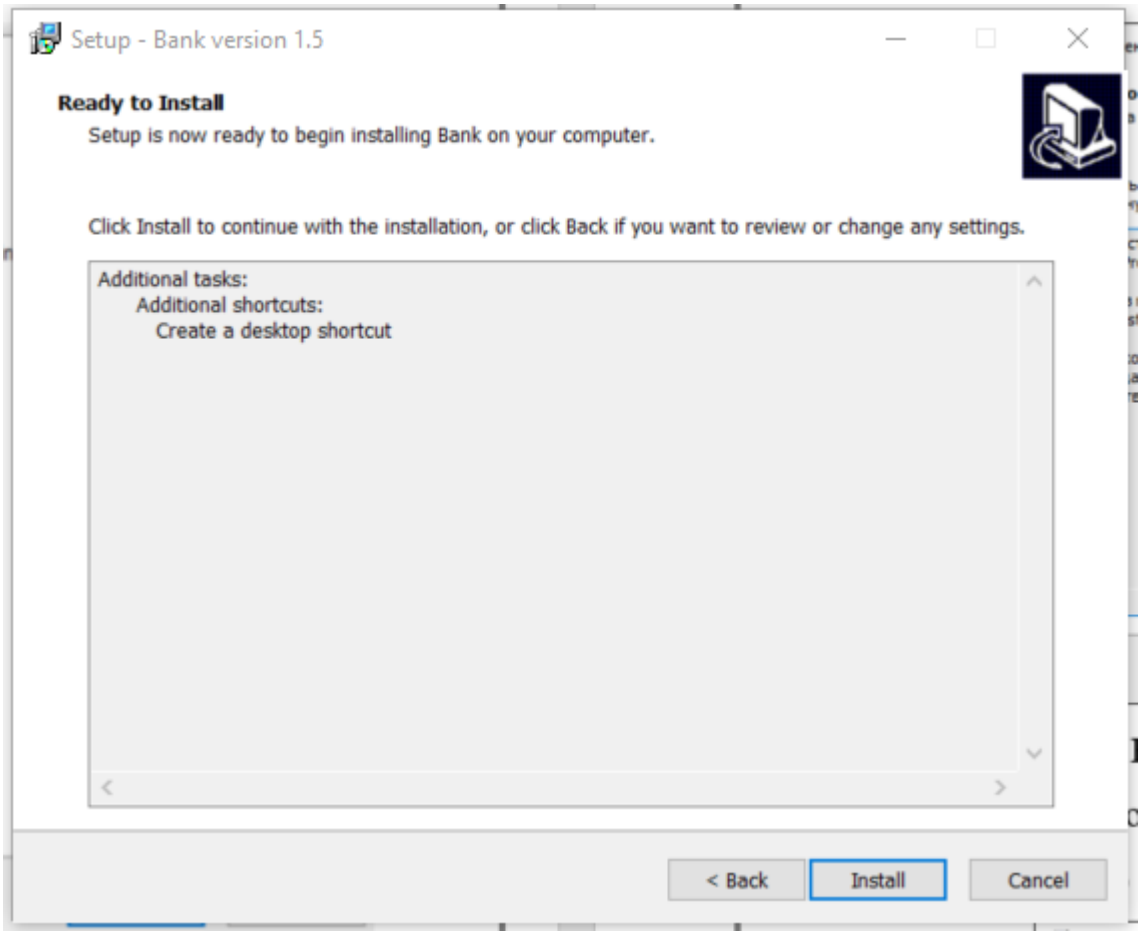


Рисунок 3.4. – Встановлення програми

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

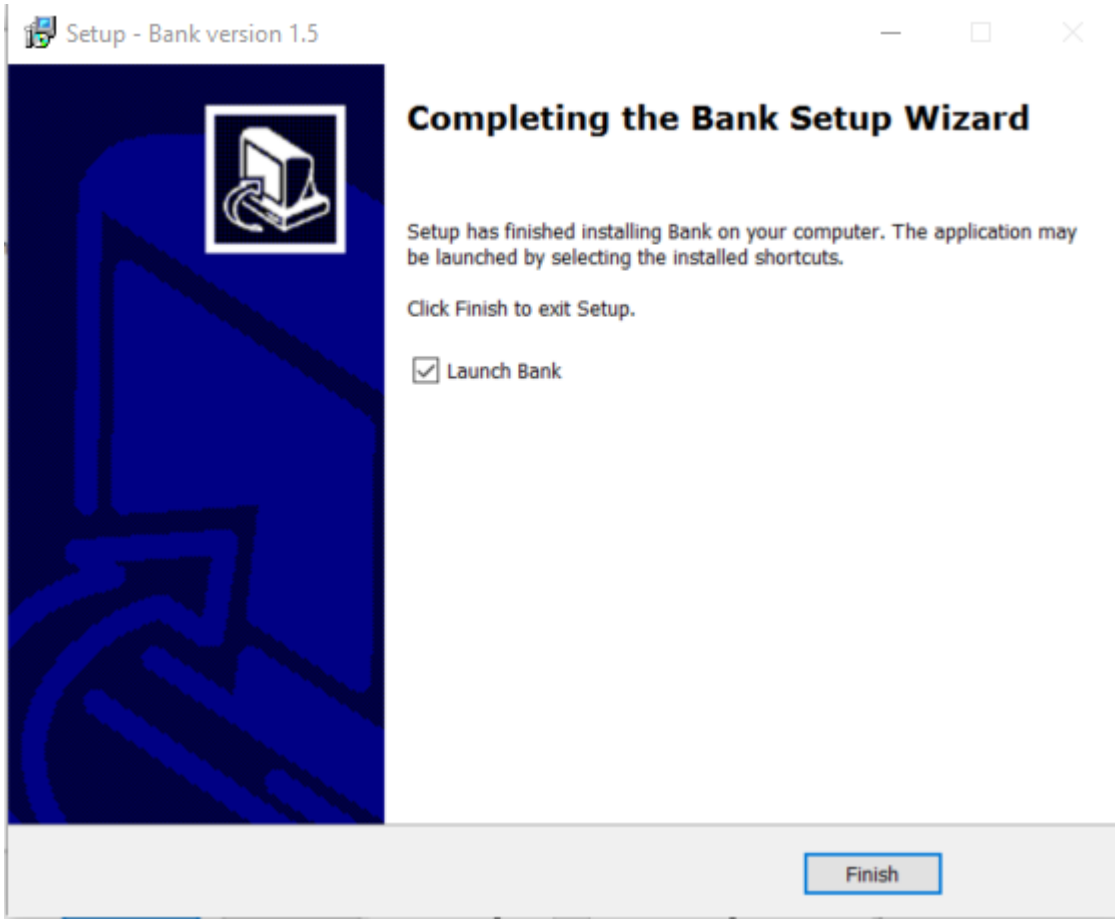


Рисунок 3.5. – Встановлення програми

Рисунок 3.6. – Вигляд ярлика встановленої програми

3.2 Інструкція з експлуатації проекту

У цьому розділі описано всі можливі варіації роботи програми та показано їх результат.

При запуску програми відображається головне вікно логіну.

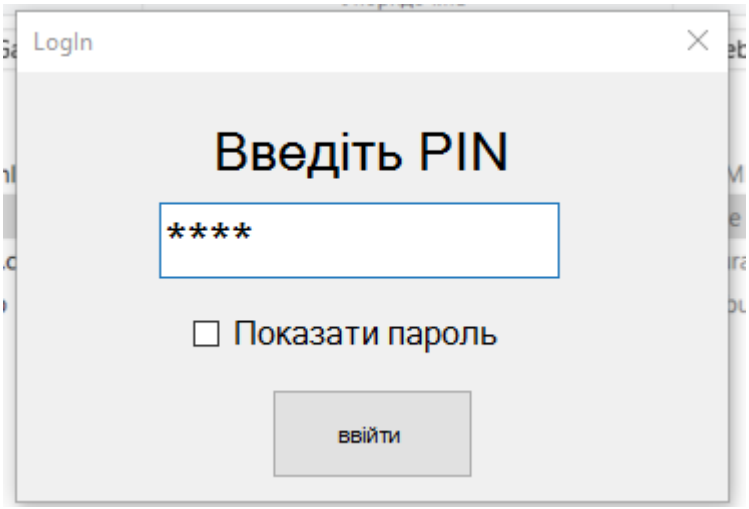


Рисунок 3.7. – Головне вікно програми

Після введення пін коду відповідно відкриється форма Адміністратора або клієнта банку

Рисунок 3.8. – Форма Адміністратора перша вкладка Додання нового клієнта

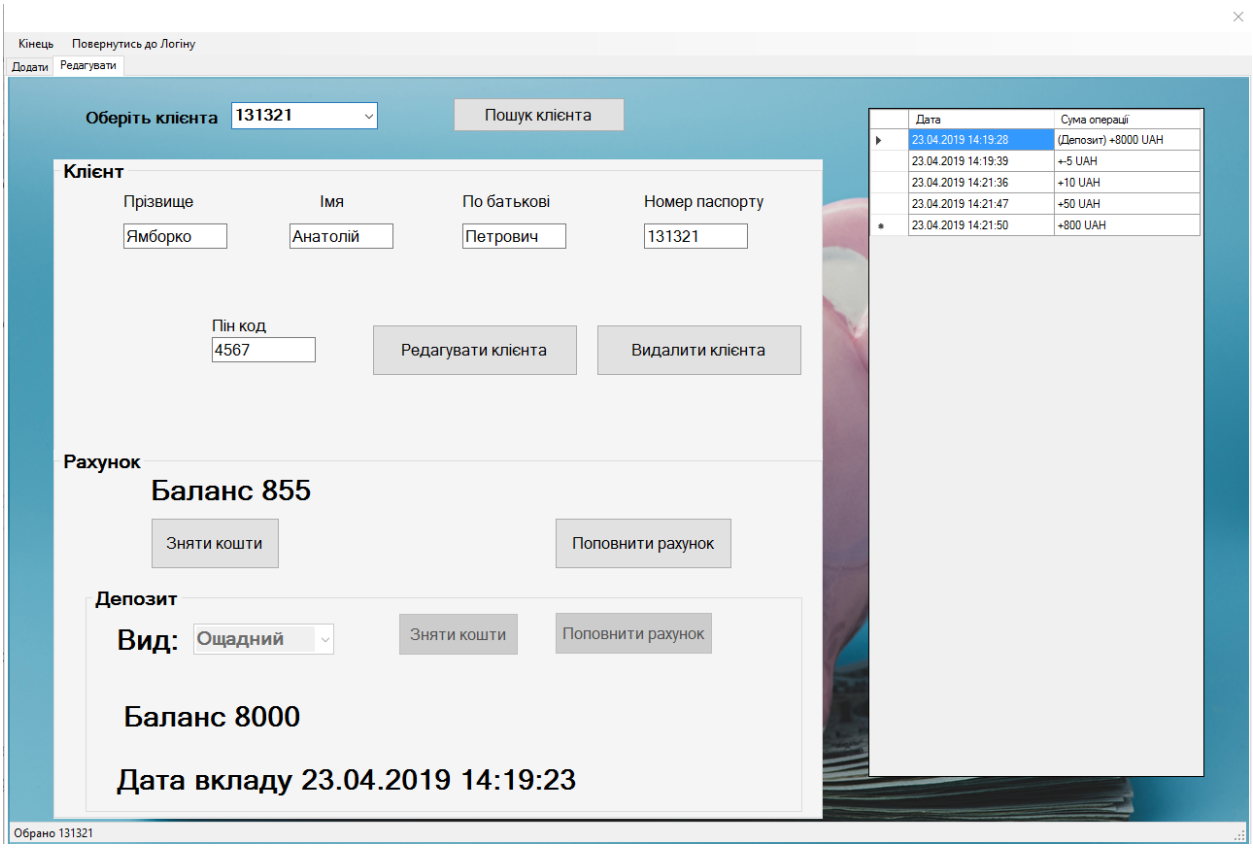


Рисунок 3.9. – Форма Адміністратора перша вкладка Редагування інформації існуючих клієнтів

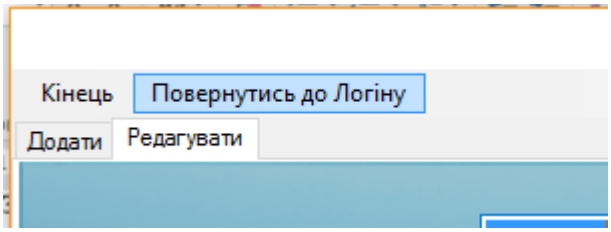


Рисунок 3.10. – Форма Адміністратора кнопка для повернення до форми Логіну

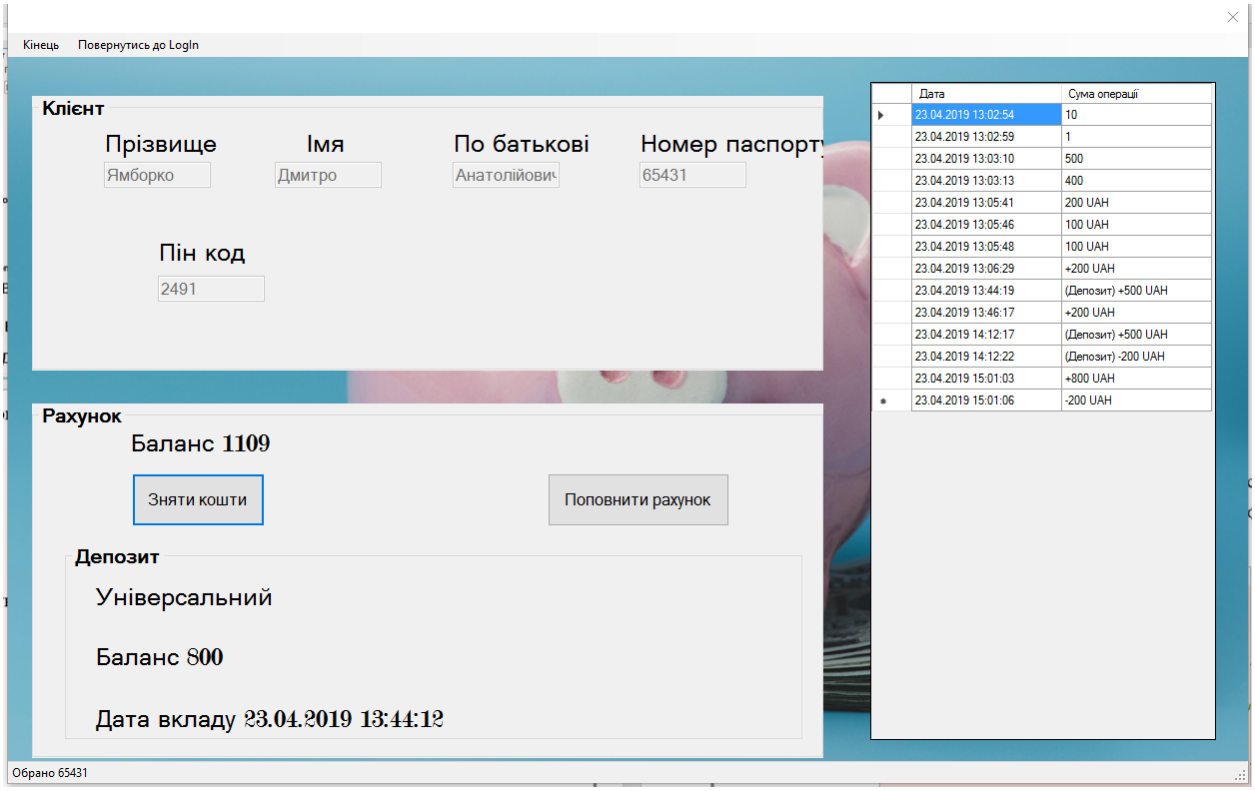


Рисунок 3.11. – Форма клієнта банку

Знімемо з рахунку 100 грн та поповнимо на 200

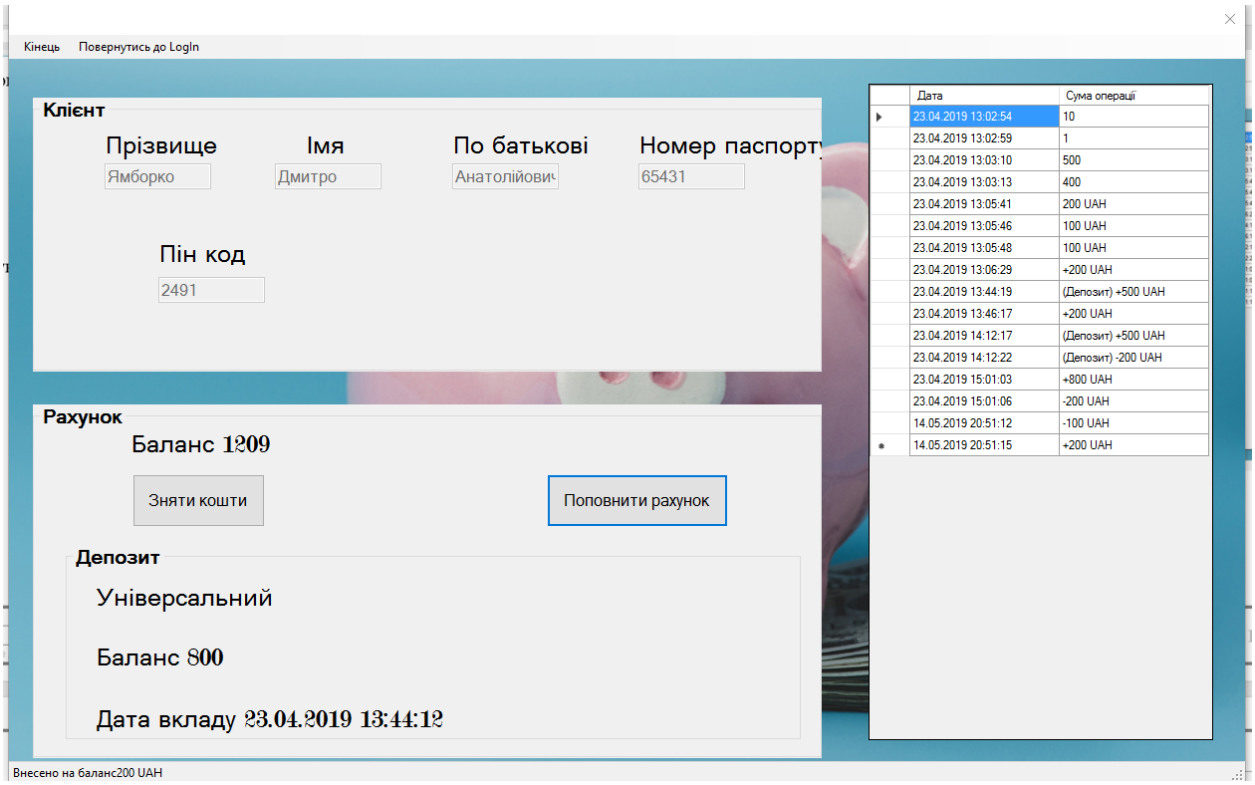


Рисунок 3.12. – Форма клієнта банку з оновленим списком операцій

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.

ЛІТЕРАТУРА

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 – Лістинг програми

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(lastOperatio
ns));
        }
    }
}
```

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(o
bject 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(clientsDictio
nary.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,tbA
ddSurname.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(clientsDictiona
ry);
            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.ToSt
ring();
            if
(client.account.lastOperations.Count ==
0)
            {
                refreshDatagrid(new
List<Operation>() { new
Operation(default(DateTime), "") });
            }

refreshDatagrid(client.account.lastOperat
ions);
            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(clientsDictiona
ry);
            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.ToDoubl
e(suma));

client.account.lastOperations.Add(new
Operation(DateTime.Now,("-"+suma + "
UAH")));
                clientsDictionary[key] =
client;

Serealizator.serealizable(clientsDictiona
ry);
                label7.Text = "Баланс " +
client.account.balance.ToString();

refreshDatagrid(client.account.lastOperat
ions);
                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(clientsDictiona
ry);
            label7.Text = "Баланс " +
client.account.balance.ToString();

refreshDatagrid(client.account.lastOperat
ions);
            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(cbEditD
eposit.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(clientsDictiona
ry);
            label13.Text = "Баланс " +
client.account.deposit.balance.ToString()
;

refreshDatagrid(client.account.lastOperat
ions);
            tSS.Text = "Внесено на
депозит" + suma + " UAH";
            label15.Text =
client.account.deposit.dateOfDeposit.ToSt
ring();
            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(clientsDictiona
ry);
            clearAllFields();
            refreshCbEditClient();
            tSS.Text = "Видалено";
        }

        private void
BtEditClientMinusMoneyDeposit_Click(objec
t 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(clientsDictiona
ry);
            label13.Text = "Баланс " +
client.account.deposit.balance.ToString()
;

refreshDatagrid(client.account.lastOperat
ions);
            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(clientsDictiona
ry);
            label13.Text = "Баланс " +
client.account.deposit.balance.ToString()
;

refreshDatagrid(client.account.lastOperat
ions);
            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.ToSt
ring();
            if
(client.account.lastOperations.Count ==
0)
            {
                refreshDatagrid(new
List<Operation>() { new
Operation(default(DateTime), "") });
            }

refreshDatagrid(client.account.lastOperat
ions);
            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.ToDoubl
e(suma));

client.account.lastOperations.Add(new
Operation(DateTime.Now, ("-" + suma + "
UAH")));
                clientsDictionary[key] =
client;

Serealizator.serealizable(clientsDictiona
ry);
                label7.Text = "Баланс " +
client.account.balance.ToString();

refreshDatagrid(client.account.lastOperat
ions);
                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(clientsDictiona
ry);
            label7.Text = "Баланс " +
client.account.balance.ToString();

refreshDatagrid(client.account.lastOperat
ions);
            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.Te
xt)];
            }
            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.SetCompatibleTextRenderingDef
ault(false);
            Application.Run(new LogIn());
        }
    }
}
```