МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ Андрій ФОРСЮК	_____ Сергій ГРИБКОВ
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«___» _____ 2024р.	«___» _____ 2024р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)
освітньо-професійної програми Комп'ютерні науки
на тему: «Розроблення інформаційної системи, призначеної для підвищення ефективності функціонування банку»

Виконав: здобувач 3 курсу, групи 5-ІТ

Пожидаєв Владислав Олександрович _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Чевська Крістіна Сергіївна _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти _____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач _____
(підпис)

Київ – 2024 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки
Освітній ступінь Бакалавр
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри
Інформаційних технологій, штучного інтелекту і кібербезпеки
Сергій ГРИБКОВ

" 15 " Листопада 2023 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Пожидаєв Владислав Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розроблення інформаційної системи, призначеної для підвищення ефективності функціонування банку»*,
керівник роботи *Чевська Крістіна Сергіївна*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «14» листопада 2023 року № 934-кс
2. Строк подання здобувачем роботи: *26.01.2024 р.*
3. Вихідні дані до роботи: *дані про діяльність та особливості функціонування банку та банківських інформаційних систем*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
системний аналіз банківських інформаційних систем та виявлення задач автоматизації, технічне завдання на проєктування, опис комплексу задач автоматизації, охорона праці та техніка безпеки
5. Перелік графічного матеріалу:
1. Приклади роботи інформаційної системи
2. Фрагменти коду

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Чевська Крістіна Сергіївна		
2	Чевська Крістіна Сергіївна		
3	Чевська Крістіна Сергіївна		
4	Чевська Крістіна Сергіївна		

7. Дата видачі завдання *15 листопада 2023 року*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Системний аналіз об’єкта дослідження та виявлення задач автоматизації	27.11.2023	Виконано
2	Створення технічного завдання	11.12.2023	Виконано
3	Розроблення інформаційної системи	18.12.2023	Виконано
4	Написання інструкцій користувача	08.01.2024	Виконано
5	Оформлення пояснювальної записки	15.01.2024	Виконано
6	Оформлення презентації	26.01.2024	Виконано

Здобувач _____ Владислав ПОЖИДАЄВ
(підпис) (прізвище та ініціали)

Керівник роботи _____ Крістіна ЧЕВСЬКА
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пожидаєв В.О. «Розроблення інформаційної системи, призначеної для підвищення ефективності функціонування банку».

Кваліфікаційна робота складається з 74 сторінок, 25 рисунків, 6 таблиць, 1 додатку та 10 джерел.

Метою даної кваліфікаційної роботи є розроблення інформаційної системи, призначеної для підвищення ефективності функціонування банку.

У даній роботі було проведено аналіз банківської діяльності, щоб визначити області, де автоматизація може принести найбільший ефект, розроблено функціональну модель інформаційної системи, обрано технологічне рішення для реалізації інформаційної системи та виконано програмну реалізацію інформаційної системи.

Для розробки інформаційної системи використано мову програмування C# та середовище програмування Visual Studio 2019.

Впровадження розробленого програмного продукту значно підвищить ефективності функціонування банку.

КЛЮЧОВІ СЛОВА: БАНКІВСЬКА ІНФОРМАЦІЙНА СИСТЕМА, MS SQL SERVER, VISUAL STUDIO 2019, C#.

SUMMARY

Pozhidayev V. "Development of an information system designed to increase the efficiency of the bank's functioning."

The qualification work consists of 74 pages, 25 figures, 6 tables, 1 appendix and 10 sources.

The purpose of this qualification work is to develop an information system designed to increase the efficiency of the bank's functioning.

In this work, an analysis of banking activity was carried out to determine the areas where automation can bring the greatest effect, a functional model of the information system was developed, a technological solution was chosen for the implementation of the information system, and the software implementation of the information system was performed.

The C# programming language and the Visual Studio 2019 programming environment were used to develop the information system.

Implementation of the developed software product will significantly increase the efficiency of the bank's functioning.

KEYWORDS: BANKING INFORMATION SYSTEM, MS SQL SERVER, VISUAL STUDIO 2019, C#.

ЗМІСТ

ВСТУП

Банківська сфера є однією з найбільш конкурентоспроможних галузей економіки. Вони відіграють важливу роль у забезпеченні фінансової стабільності та розвитку економіки.

У сучасних умовах банки постійно конкурують між собою за клієнтів та ринкову частку. Для того, щоб бути успішними, банкам необхідно підвищувати свою ефективність та постійно вдосконалювати свою діяльність.

Одним із ключових напрямків такого вдосконалення є автоматизація банківських процесів.

Інформаційні системи відіграють важливу роль у підвищенні ефективності банківської діяльності. Вони дозволяють банкам автоматизувати виконання основних банківських операцій, таких як відкриття та обслуговування рахунків, здійснення платежів, надання кредитів тощо. Покращити якість обслуговування клієнтів, наприклад, за рахунок надання нових послуг, підвищення доступності інформації тощо. Знизити витрати, наприклад, за рахунок зменшення кількості паперових документів, оптимізації робочих процесів тощо. Покращити управління банком, наприклад, за рахунок прийняття більш ефективних рішень, виявлення ризиків тощо.

Розробка ефективних банківських інформаційних систем є важливим завданням для банків. Це завдання може бути вирішено за допомогою кваліфікованих фахівців у галузі інформаційних технологій та банківської справи.

Метою даної кваліфікаційної роботи є розроблення інформаційної системи, призначеної для підвищення ефективності функціонування банку.

Для досягнення цієї мети були поставлені такі завдання:

Провести аналіз банківської діяльності, щоб визначити області, де автоматизація може принести найбільший ефект. Розробити функціональну модель інформаційної системи, яка буде відповідати потребам банку. Вибрати

технологічне рішення для реалізації інформаційної системи. Виконати програмну реалізацію інформаційної системи. Провести тестування та налагодження інформаційної системи.

Об’єктом дослідження є інформаційна система, призначена для підвищення ефективності функціонування банку.

Предметом дослідження є теоретичні та практичні аспекти розроблення інформаційних систем у банківській сфері.

Розробка інформаційної системи, призначеної для підвищення ефективності функціонування банку, є актуальною темою для кваліфікаційної роботи бакалавра. Дана тема дозволяє закріпити теоретичні знання та отримати практичні навички в галузі інформаційних технологій.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ БАНКІВСЬКИХ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ВИЯВЛЕННЯ ЗАДАЧ АВТОМАТИЗАЦІЇ

1.1 Загальна характеристика банківських інформаційних систем

Банківські інформаційні системи (БІС) відіграють важливу роль у розвитку банківської справи. Вони дозволяють банкам підвищувати свою конкурентоспроможність, забезпечувати безпеку фінансових ресурсів та надавати якісні послуги клієнтам.

Банківські інформаційні системи - це комплекси програмного та апаратного забезпечення, призначені для автоматизації банківської діяльності. Вони дозволяють банкам ефективно керувати своїми операціями, підвищувати рівень обслуговування клієнтів та забезпечувати безпеку фінансових ресурсів.

Основними компонентами БІС є:

- Апаратне забезпечення – комп’ютери, сервери, мережеве обладнання та інші пристрої, які забезпечують функціонування системи.
- Програмне забезпечення – програми, які здійснюють обробку даних та управління банківським бізнесом.
- Дані – інформація про клієнтів, фінансові операції, нормативні вимоги тощо.

Особливості банківських інформаційних систем:

- Великий обсяг даних. Банки обробляють величезний обсяг інформації про клієнтів, фінансові операції, нормативні вимоги тощо.
- Висока вимога до надійності та безпеки. БІС повинні забезпечувати надійне зберігання та обробку даних, а також захист від несанкціонованого доступу.
- Швидкість обробки даних. Банківські операції повинні виконуватися швидко та без затримок.
- Масштабованість. БІС повинні бути здатними адаптуватися до зростання обсягу операцій.

Банківські інформаційні системи можна класифікувати за різними ознаками, наприклад за масштабом (корпоративні, міжбанківські, галузеві), за призначенням (операційні, управлінські, аналітичні) та за технологічним рівнем (традиційні, еволюційні, інноваційні).

Банківські операційні системи – системи, які автоматизують виконання основних банківських операцій, таких як відкриття та обслуговування рахунків, здійснення платежів, надання кредитів тощо. Наприклад:

- Oracle Banking – система, розроблена компанією Oracle. Вона забезпечує автоматизацію основних банківських операцій, таких як відкриття та обслуговування рахунків, здійснення платежів, надання кредитів тощо.
- SAP Banking – система, розроблена компанією SAP. Вона має широкий спектр функціональних можливостей, включаючи управління ризиками, управління активами та пасивами, управління клієнтськими відносинами тощо.
- Temenos T24 – система, розроблена компанією Temenos. Вона є однією з найпопулярніших банківських операційних системи у світі.

Банківські управлінські системи – системи, які забезпечують управління банком на рівні стратегії, планування, контролю та аналізу. Наприклад:

- Oracle Hyperion Financial Management – система, розроблена компанією Oracle. Вона забезпечує управління фінансовими процесами банку, включаючи бюджетування, прогнозування, звітність тощо.
- SAP BusinessObjects BI – система, розроблена компанією SAP. Вона забезпечує управління бізнес-інформацією, включаючи аналітику продажів, маркетингу, ризиків тощо.
- Microsoft Dynamics AX – система, розроблена компанією Microsoft. Вона забезпечує управління всіма аспектами банківської діяльності, включаючи операції, управління, ризики тощо.

Банківські аналітичні системи – системи, які дозволяють банкам аналізувати фінансову інформацію та отримувати нові знання про свою діяльність. Наприклад:

- SAS Enterprise Miner – система, розроблена компанією SAS. Вона дозволяє банкам аналізувати великі обсяги даних, щоб отримувати нові знання про свою діяльність.
- IBM SPSS Modeler – система, розроблена компанією IBM. Вона також дозволяє банкам аналізувати великі обсяги даних, щоб прогнозувати тенденції, виявляти ризики тощо.
- TIBCO Spotfire – система, розроблена компанією TIBCO. Вона забезпечує візуалізацію даних, що дозволяє банкам краще розуміти свою діяльність.

Традиційні БІС – це системи, які використовують традиційні технології та методи обробки інформації. Вони мають ряд обмежень, наприклад, низьку масштабованість, недостатню надійність та безпеку.

Еволюційні БІС – це системи, які використовують сучасні технології, але зберігають у своїй основі традиційні підходи до проектування та реалізації. Вони дозволяють підвищити ефективність банківської діяльності, але не вирішують всі проблеми, пов’язані з використанням традиційних технологій.

Інноваційні БІС – це системи, які використовують передові технології та методи обробки інформації. Вони дозволяють банкам отримувати нові конкурентні переваги, наприклад, підвищувати якість обслуговування клієнтів, забезпечувати безперебійну роботу банківської системи, мінімізувати ризики.

Українські банки використовують такі БІС, як Temenos T24, SAP HANA, Oracle Flexcube, IBM Sterling тощо.

Ці приклади є лише деякими з багатьох існуючих БІС. Банки постійно вдосконалюють свої інформаційні системи, щоб відповідати сучасним вимогам та конкурентному середовищу.

Банки використовують передові технології, такі як штучний інтелект, машинне навчання та блокчейн, щоб підвищити ефективність своєї діяльності та надавати клієнтам якісніші послуги.

1.2 Аналіз бізнес-процесів предметної області

Виділимо бізнес-процеси, що протікають у системі та занесемо їх до таблиці 1.1

Таблиця 1.1 Бізнес-процеси предметної області

№ з/п	Бізнес-процес	Виконавець	Вхідні дані		Вихідні дані	
			Постачальник	Зміст	Споживач	Зміст
1	Створити заявку	Клієнтський додаток	Клієнт	Інформація про нову заявку	База даних заявок	Інформація про нову заявку
2	Отримати список нових заявок	Клієнтський додаток	База даних заявок	Список нових заявок	Користувач	Список нових заявок
3	Обробити заявку на кредитування	Клієнтський додаток	Користувач	Нова інформація про кредит	База даних заявок	Нова інформація про кредит
4	Обробити заявку на вклад	Клієнтський додаток	Користувач	Нова інформація про вклад	База даних заявок	Нова інформація про вклад
5	Змінити оброблену заявку	Клієнтський додаток	Користувач	Нова інформація про заявку	База даних заявок	Нова інформація про заявку
6	Додати нового клієнта	Клієнтський додаток	Користувач	Додавання інформації про нового клієнта	База даних заявок	Інформація про нового клієнта
7	Змінити клієнта	Клієнтський додаток	Користувач	Зміна інформації про клієнта	База даних заявок	Нова інформація про клієнта
8	Отримати список клієнті	Клієнтський додаток	База даних заявок	Список клієнтів	Користувач	Список клієнтів
9	Отримати список оброблених заявок	Клієнтський додаток	База даних заявок	Список оброблених заявок	Користувач	список оброблених заявок
10	Додати користувача	Клієнтський додаток	Адміністратор	Нова інформація про нового користувача	База даних заявок	Нова інформація про нового користувача

11	Видалити користувача	Клієнтський додаток	Адміністратор	Інформація про користувача, якого видаляють	База даних заявок	Список користувачів
12	Змінити користувача	Клієнтський додаток	Адміністратор	Змінена інформація про користувача	База даних заявок	Змінена інформація про користувача
13	Отримати список користувачів	Клієнтський додаток	База даних заявок	Список користувачів	Адміністратор	Список користувачів
14	Проглянути статистику діяльності	Клієнтський додаток	Адміністратор	Період часу	База даних заявок	Статистичні дані

1.2.1 Побудова вихідної концептуальної моделі даних

Концептуальне проектування бази даних полягає в розробці концептуальної моделі бази даних, яка не враховує обрану модель даних і особливості цільової СУБД. Концептуальна модель, як правило, представляється як сукупність типів сутностей і зв’язків між ними.

Виділимо типи сутностей і перерахуємо результати в таблиці 1.2.

Таблиця 1.2 Типи сутностей основного домену

№ з\п	Назва	Опис
1	Новий додаток	Містить інформацію про новий додаток
2	Тип нової заявки	Містить опис типів нових заявок
3	Опрацьована заявка кредитування	Містить інформацію про оброблену заявку на кредит
4	Опрацьована заявка вклади	Містить інформацію про оброблену заявку на вклад
5	Опрацьована заявка перекази	Містить інформацію про оброблену заявку на переказ

6	Клієнти	Містить інформацію про клієнтів
7	Користувачі	Містить інформацію про користувачів
8	Фізична особа	Містить інформацію про фізичну особу
9	Юридична особа	Містить інформацію про юридичну особу

У ході подальшої роботи буде здійснено перехід від концептуальної моделі до логічної.

1.3 Розрахунок економічного ефекту від впровадження системи

Для розрахунку економічного ефекту від впровадження інформаційної системи, призначеної для підвищення ефективності функціонування банку, необхідно визначити наступні показники:

Вартість впровадження інформаційної системи. Вона включає в себе витрати на проектування, розробку, тестування, впровадження та навчання персоналу.

Економічний ефект від використання інформаційної системи. Він може включати в себе наступні складові:

Зменшення витрат. Наприклад, інформаційна система може дозволити банку зменшити витрати на персонал, операційні витрати, витрати на обслуговування клієнтів тощо.

Збільшення доходів. Наприклад, інформаційна система може дозволити банку збільшити кількість клієнтів, обсяги операцій, комісійний дохід тощо.

Розрахунок економічної доцільності впровадження інформаційної системи проводиться за наступною формулою:

Економічна доцільність = Економічний ефект / Вартість впровадження

Якщо економічна доцільність більше 1, то впровадження інформаційної системи є економічно доцільним.

Приклад розрахунків

Припустимо, що банк планує впровадити інформаційну систему, яка дозволить йому автоматизувати процес відкриття та обслуговування рахунків. Вартість впровадження системи оцінюється в 10 мільйонів гривень.

Використання інформаційної системи дозволить банку зменшити витрати на персонал на 20%. В даний час банк витрачає на обслуговування рахунків 100 співробітників, які отримують середню зарплату в 20 тисяч гривень. Таким чином, щорічні витрати банку на персонал на обслуговування рахунків становлять 20 мільйонів гривень (100 співробітників * 20 тисяч гривень).

Також використання інформаційної системи дозволить банку збільшити кількість відкритих рахунків на 10%. В даний час банк відкриває 10 тисяч рахунків на рік. Таким чином, додатковий дохід від відкриття нових рахунків становитиме 2 мільйони гривень (10 тисяч рахунків * 200 гривень).

Отже, економічний ефект від використання інформаційної системи становить:

Зменшення витрат на персонал = 20 мільйонів гривень

Збільшення доходів = 2 мільйони гривень

Разом: 22 мільйони гривень

Економічна доцільність впровадження інформаційної системи дорівнює:

Економічна доцільність = 22 мільйони гривень / 10 мільйонів гривень = 2,2

Таким чином, впровадження інформаційної системи є економічно доцільним рішенням.

Конкретні значення показників вартості впровадження інформаційної системи та економічного ефекту від її використання залежать від конкретних умов банку. Однак, в цілому, впровадження інформаційної системи, призначеної для підвищення ефективності функціонування банку, є економічно доцільним рішенням.

Ось деякі приклади економічного ефекту від впровадження банківських інформаційних систем:

Зменшення витрат на персонал. За даними дослідження, проведеного компанією Capgemini, впровадження банківських інформаційних систем дозволяє банкам зменшити витрати на персонал на 20-30%.

Збільшення доходів. За даними дослідження, проведеного компанією IBM, впровадження банківських інформаційних систем дозволяє банкам збільшити доходи на 10-15%.

Покращення якості обслуговування клієнтів. За даними дослідження, проведеного компанією Gartner, впровадження банківських інформаційних систем дозволяє банкам покращити якість обслуговування клієнтів на 15-20%.

Таким чином, впровадження інформаційної системи, призначеної для підвищення ефективності функціонування банку, може принести банку суттєві економічні вигоди.

Однак, слід зазначити, що економічний ефект від впровадження інформаційної системи може бути не відразу очевидним. В деяких випадках він може бути досягнутий лише через кілька років.

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ НА ПРОЄКТУВАННЯ

1. Загальні положення.

1.1. Найменування системи: «Інформаційна система, призначена для підвищення ефективності функціонування банку»

1.2 Призначення системи

Система повинна отримувати інформацію про вибрану послугу та формувати заявку для операторів. Кожна заявка потрапляє на сервер і потім передається оператору для подальшої обробки. Кожен оператор може додати клієнта до бази даних, змінити інформацію або видалити. Обробка заявки полягає в наступному: у відповідність до своїх обов'язків, оператор заповнює відповідні бланки, зберігає їх на сервері та роздруковує відповідні форми. Адміністратор цієї системи може додавати, змінювати чи видаляти операторів системи та переглядати статистику діяльності всього відділення.

1.3 Межі проекту

Цей проект призначений для підтримки функціонування банківського відділення, а саме автоматизації дій із заявками, такими як отримання заявки, створення нової заявки, перегляд лога заявок, додавання нових клієнтів, перегляд списку клієнтів. У клієнтському додатку має бути передбачена можливість пошуку та фільтрації клієнтів та оброблених заявок.

1.4 Передбачувана аудиторія

Цей проект використовуватиметься співробітниками банківського відділення.

1.5 Загальний погляд на систему

Система, що проектується, використовує архітектуру клієнт-сервер. У цьому випадку клієнтська програма для роботи із заявками розгортається на робочих станціях користувачів, а сервер зберігає базу даних, що містить відомості про заявки та клієнтів. Роботу системи можна описати так: Клієнт створює нову заявку

на конкретний вид консультації, система шукає вільного користувача, який може обробити дану заявку, якщо є вільний користувач, то заявка переправляється йому, інакше встає в чергу. Вільному користувачеві надходить повідомлення про нову заявку, користувач у відповідність із заявкою заповнює необхідну інформацію та відправляє на сервер. Адміністратор стежить за рухом заявок шляхом аналізу, складеного сервером.

1.6 Особливості системи

Як особливості програмного забезпечення, що розробляється, можна виділити наступне:

- розмежування доступної функціональності шляхом авторизації користувачів;
- автоматичний пошук нових заявок на сервері;
- повідомлення користувача про нову заявку.

2. Вимоги до системи.

Вимогу до інформаційної системи можна визначити як «докладний опис того, що має бути реалізовано». Існує два основних типи вимог:

- функціональні вимоги – яку поведінку має пропонувати система;
- нефункціональні вимоги –властивості чи обмеження, що накладаються на систему.

2.1 Функціональні вимоги

2.1.1 Класи та характеристики користувачів

Як користувачів розроблюваного клієнтського додатку, виділяються користувач та адміністратор. Класи та характеристики користувачів представлені у таблиці 2.1.

Таблиця 2.1 Класи та характеристики користувачів

Клас користувачів	Характеристика користувачів
Оператор	Користувач має можливість переглядати оброблені заявки, додавати нові, редагувати наявні, переглядати список клієнтів, додавати нових або редагувати наявних.
Клієнт	Клієнт має можливість створювати заявки
Адміністратор	Адміністратор має можливість редагувати інформацію про користувачів та переглядати статистику роботи відділення

Відповідно до поставленого завдання було побудовано модель прецедентів предметної області. Діаграми варіантів використання представлені рисунках 2.1-2.3.

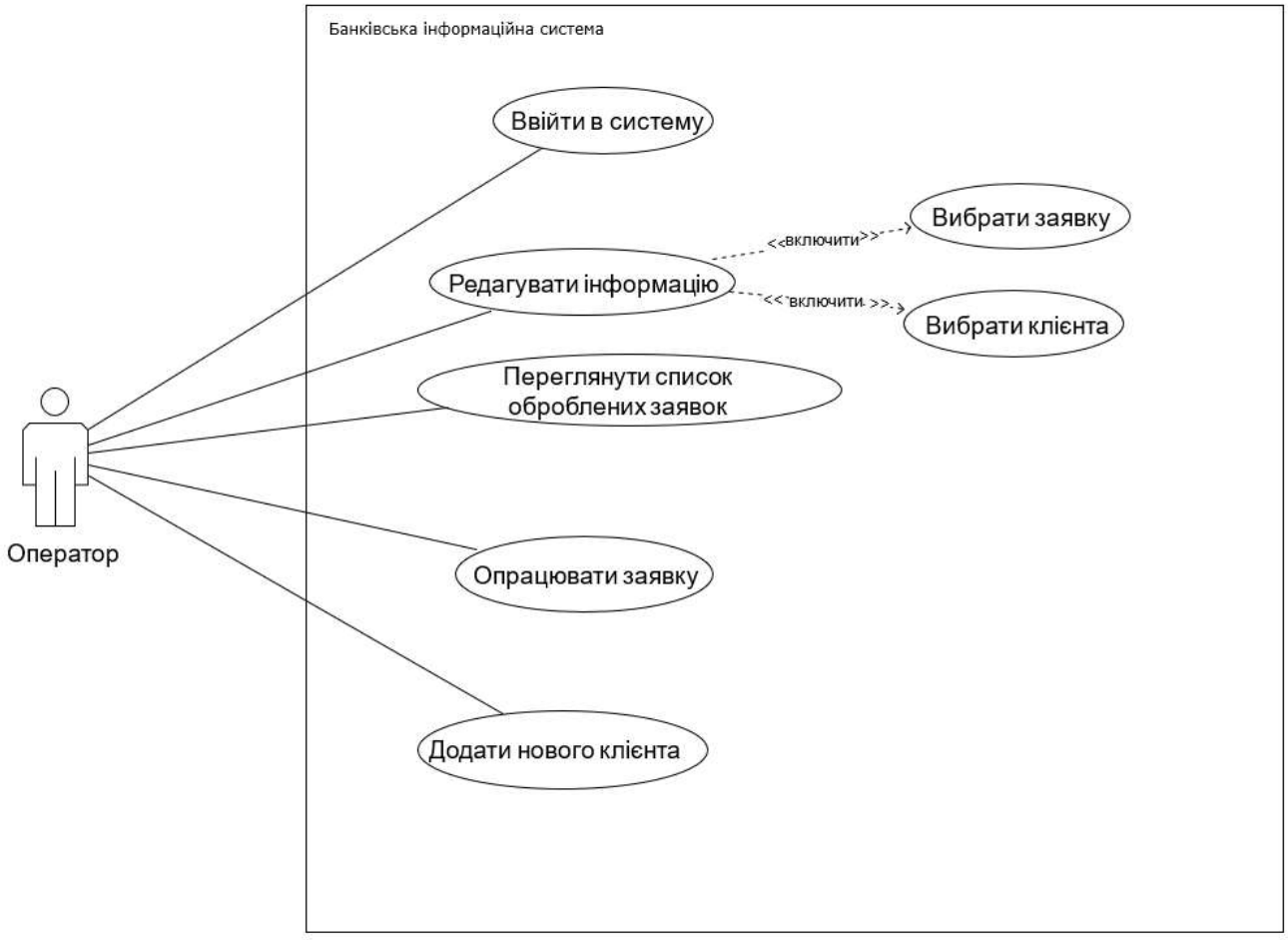

Рисунок 2.1. Діаграма варіантів використання для користувача «Оператор»

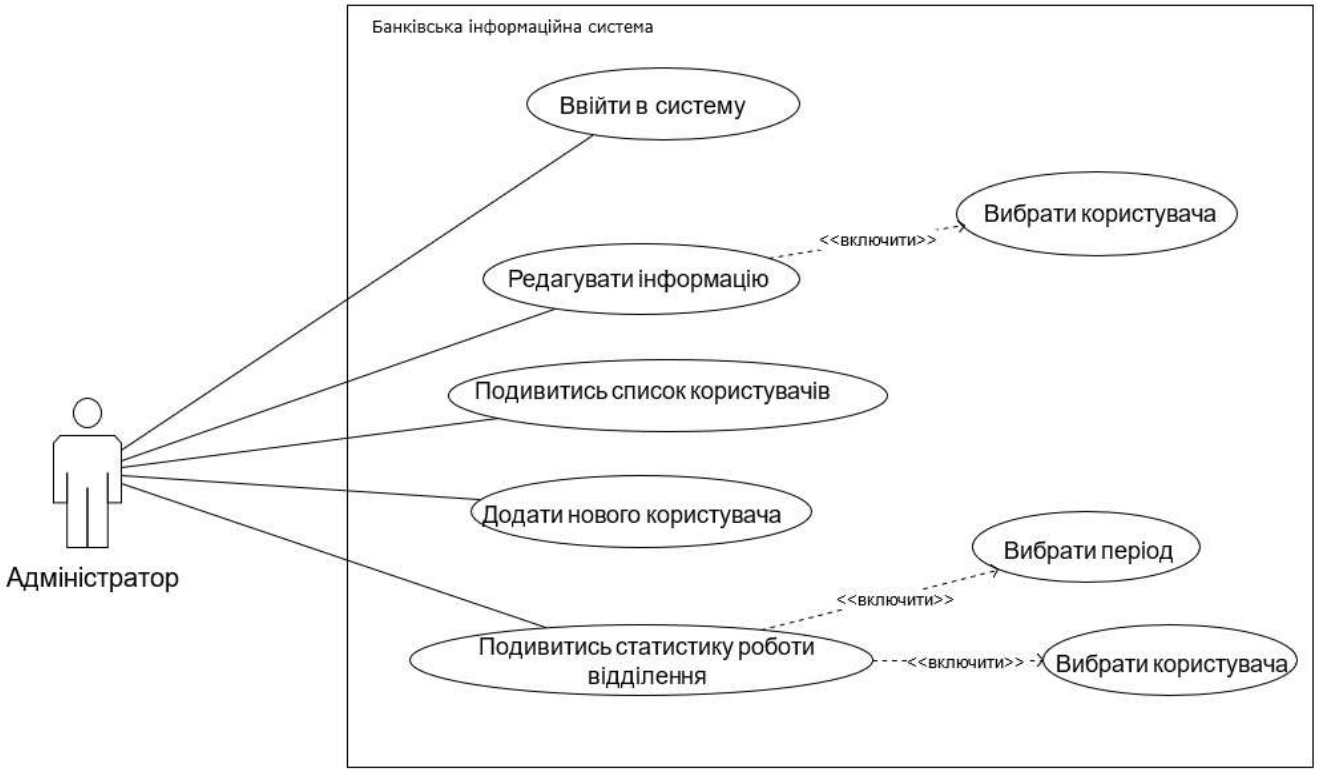

Рисунок 2.2. Діаграма варіантів використання для користувача «Адміністратор»

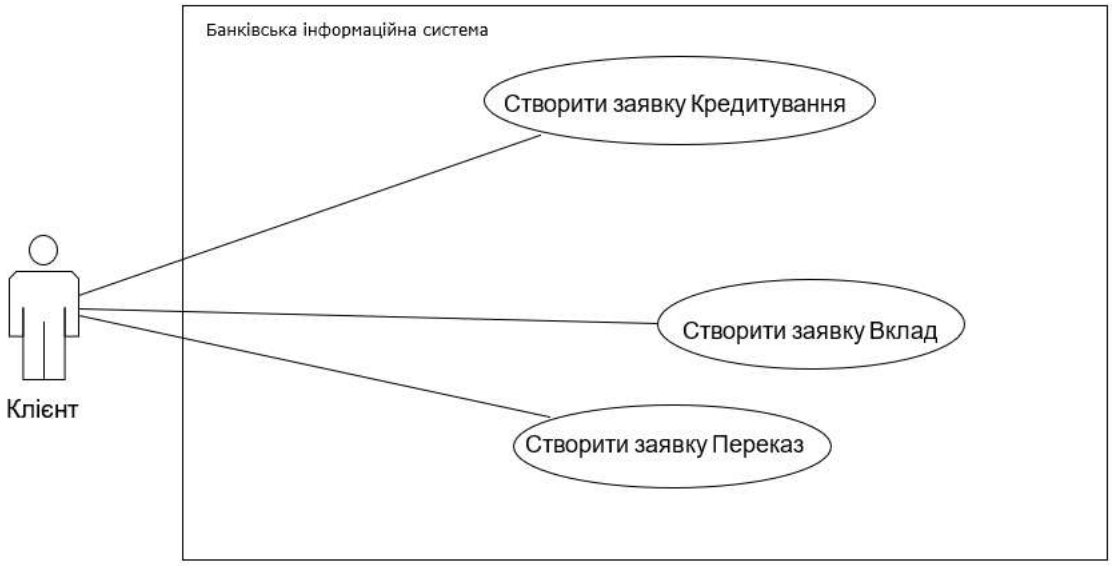

Рисунок 2.3. Діаграма варіантів використання для користувача «Клієнт»

Для наочного уявлення поведінки прецедентів зручно застосовувати діаграму діяльності. Діаграма діяльності прецеденту «Переглянути список оброблених

заявок» (рисунок 2.4), діаграма діяльності прецеденту «Створити заявку» (рисунок 2.5).

Рисунок 2.4 Діаграма діяльності прецеденту «Переглянути список оброблених заявок»

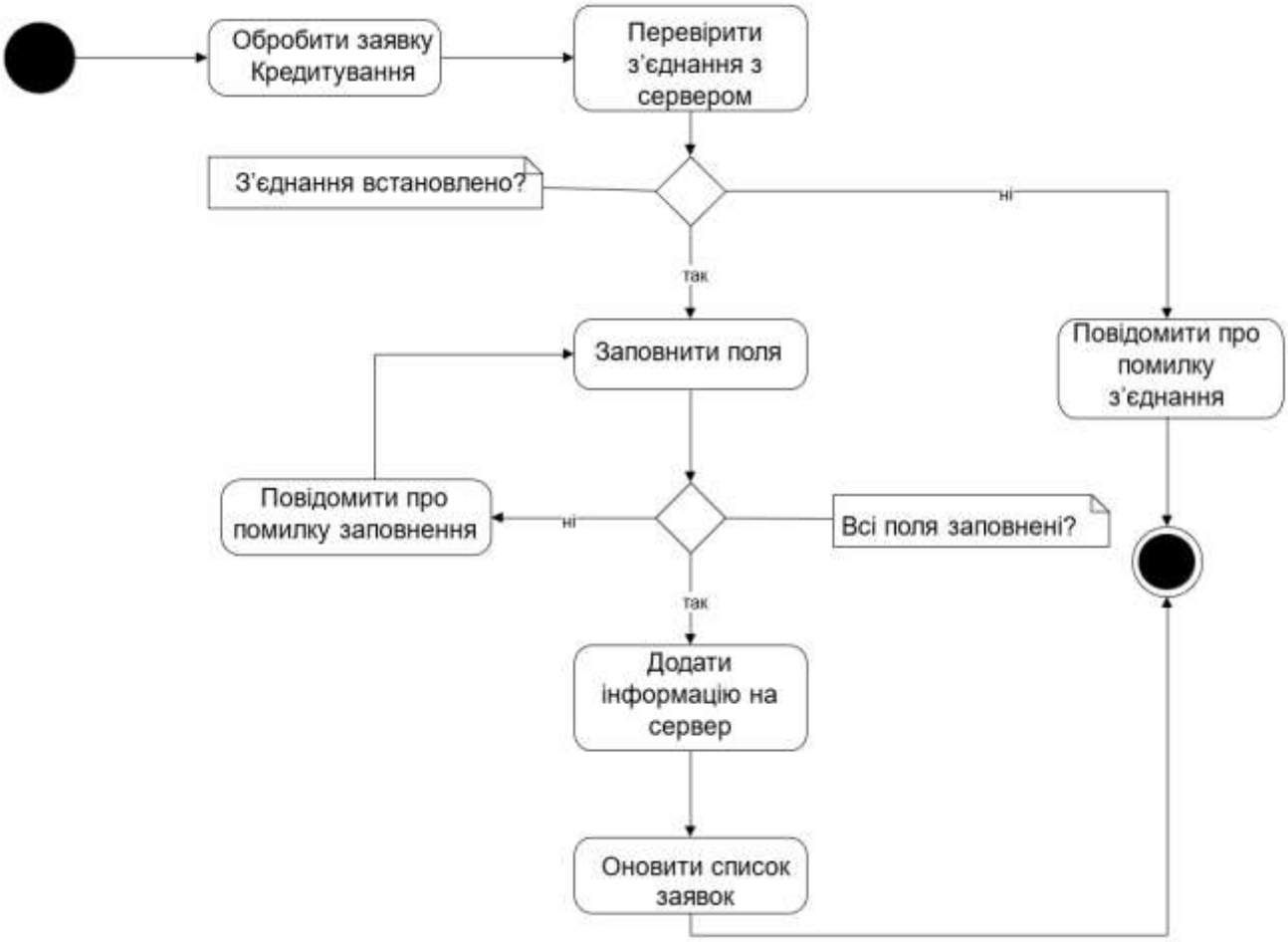

Рисунок 2.5 Діаграма діяльності прецеденту «Обробити заявку Кредитування»

2.2 Основні нефункціональні вимоги

2.2.1 Інтерфейси користувачів

Автоматизована інформаційна система банківського сервісу має надавати зручний та інтуїтивно зрозумілий інтерфейс користувачеві, що дозволяє використовувати основні функції програми.

2.2.2 Інтерфейси програмного забезпечення

При запуску користувачеві необхідно пройти аутентифікацію, після цього в базу даних відправляється SQL запит на авторизацію користувача і якщо запит виконано успішно, відкривається відповідне користувачеві вікно, при запуску якого система запитує дані, що зберігаються в базі даних за допомогою SQL – запиту на

видачу списку оброблених заявок та списку клієнтів. Отримані дані відображаються клієнту як детального списку.

2.3 Інші нефункціональні вимоги

2.3.1 Вимоги до продуктивності

Програмна система має забезпечувати стабільну роботу за наявності стабільного з'єднання. Швидкість завантаження не повинна перевищувати 20 секунд.

2.3.2 Вимоги до безпеки

Доступ до клієнтської програми повинен здійснюватися за допомогою автентифікації користувача. Користувач не повинен мати можливість редагувати користувачів та переглядати статистику роботи відділення.

2.3.3 Атрибути якості ПЗ

Список нових заявок автоматично оновлюється, коли клієнт створює нову заявку, і при появі нової заявки користувач отримує відповідне повідомлення.

РОЗДІЛ 3. ОПИС КОМПЛЕКСУ ЗАДАЧ АВТОМАТИЗАЦІЇ

3.1 Інформаційне забезпечення системи

Rational Unified Process (RUP) – це основа технології розробки програмного забезпечення, що розроблена та продається компанією Rational Software. Він забезпечує дисциплінарний підхід до розподілу та управління завданнями та областями відповідальності в організації, що займається розробкою програмного забезпечення. RUP використовує ітеративний підхід, тобто послідовність наростаючих кроків чи ітерацій. Кожна ітерація включає деякі або більшу частину дисциплін розробки (виявлення вимог, аналіз, проектування, реалізацію тощо). Кожна ітерація має чітко визначений набір цілей, і вона створює частково працюючу реалізацію кінцевої системи. Кожна наступна ітерація будується на результатах попередніх, розвиває та вдосконалює систему доти, доки не буде створено кінцевий продукт.

Особливість технології полягає в тому, що ступінь формалізації може змінюватись в залежності від потреб проекту. Можна після закінчення кожного етапу і кожної ітерації створювати всі необхідні документи і досягти максимального рівня формалізації, а можна створювати лише необхідні для роботи документи, аж до повної їх відсутності. За рахунок такого підходу до формалізації процесів технологія є досить гнучкою та широко популярною. Ця перевага дає можливість використовувати одну і ту ж команду розробників для реалізації різних за обсягом та вимогами.

Модель життєвого циклу RUP є досить складною, детально опрацьованою ітеративно-інкрементною моделлю з елементами каскаду. У моделі RUP виділяються 4 основні фази, 9 видів діяльності (процесів).

Основними фазами RUP є:

- фаза початку проекту (Inception), на якій визначаються основні цілі проекту, бюджет проекту, основні засоби його виконання – технології, інструменти,

ключовий персонал, складаються попередні плани проекту, а основна мета цієї фази – досягти компромісу між усіма зацікавленими особами щодо завдань проекту ;

• фаза опрацювання (Elaboration), основна мета якої – на базі основних, найбільш суттєвих вимог розробити стабільну базову архітектуру продукту, яка дозволяє вирішувати поставлені перед системою завдання та надалі використовується як основа розробки системи;

• фаза побудови (Construction), основна мета якої – детальне прояснення вимог та розробка системи, яка їм задовольняє, на основі спроектованої раніше архітектури;

• фаза передачі (Transition), мета якої – зробити систему повністю доступною кінцевим користувачам і де відбувається остаточне розгортання системи у її робочому середовищі, припасування дрібних деталей під потреби користувачів.

У межах кожної фази можливе проведення кількох ітерацій, кількість яких визначається складністю проекту.

Діяльності (основні процеси) RUP діляться на п'ять робітників і чотири підтримуючі. До робочих діяльностей відносяться:

- - моделювання предметної області (бізнес-моделювання, Business Modeling);
- - визначення вимог (Requirements);
- - аналіз та проектування (Analysis and Design);
- - реалізація (Implementation);
- - Тестування (Test).

Підтримуючими діяльностями є:

- - розгортання (Deployment);
- - управління конфігураціями та змінами (Configuration and Change Management);
- - управління проектом (Project Management);
- - управління середовищем проекту (Environment) [10].

Одним із основних стовпів, на які спирається RUP, є процес створення моделей за допомогою уніфікованої мови моделювання (UML). Уніфікована мова моделювання (Unified Modeling Language, UML) – це універсальна мова візуального моделювання систем.[14] Найчастіше UML асоціюється з моделюванням об'єктно-орієнтованих програмних систем, але має набагато ширше застосування завдяки властивій йому розширюваності. UML не прив'язаний до будь-якої конкретної методології чи життєвого циклу. Він може використовуватись з усіма існуючими методологіями. Основна ідея UML – можливість моделювати програмне забезпечення та інші системи як набори об'єктів, що взаємодіють. У UML-моделі є два аспекти:

- статична структура – визначає, які типи об'єктів важливі для моделювання системи та як вони взаємопов'язані;

- динамічна поведінка – визначає життєві цикли цих об'єктів і те, як вони взаємодіють друг з одним задля забезпечення необхідної функціональності системи [14].

Технологія RUP є найбільш універсальною, оскільки застосовується як у невеликих та швидких проектах, де за рахунок відсутності формалізації потрібно скоротити час виконання проекту та витрати, так і у великих та складних проектах, де потрібний високий рівень формалізму.

Платформа Entity Framework є набором технологій ADO.NET, що забезпечують розробку додатків, пов’язаних з обробкою даних. Архітекторам і розробникам додатків, орієнтованих на обробку даних, доводиться враховувати необхідність досягнення двох абсолютно різних цілей. [11] Вони повинні моделювати сутності, зв'язки та логіку вирішуваних бізнес-завдань, а також працювати з ядрами СУБД, які використовуються для збереження та отримання даних. Дані можуть розподілятися за декількома системами зберігання даних, у кожній з яких застосовуються свої протоколи, але навіть у додатках, що працюють з однією системою зберігання даних, необхідно підтримувати баланс між вимогами

системи зберігання даних та вимогами написання ефективного та зручного для обслуговування коду програми.

Entity Framework дозволяє працювати з даними у формі специфічних для домену об’єктів та властивостей, таких як клієнти та їх адреси, без необхідності звертатися до базових таблиць та стовпців бази даних, де зберігаються ці дані. Entity Framework дає розробникам можливість працювати з даними на вищому рівні абстракції, створювати та супроводжувати додатки, орієнтовані на дані, використовуючи менше коду, ніж у традиційних додатках. Оскільки Entity Framework є компонентом .NET Framework, програми Entity Framework можуть працювати на будь-якому комп’ютері, де інстальовано платформу .NET Framework, починаючи з версії 3.5 з пакетом оновлень 1 (SP1).

Паттерн Chain of Responsibility (Ланцюжок обов’язків) є поведінковим та призначений для організації в системі рівнів відповідальності (Рис. 3.1).

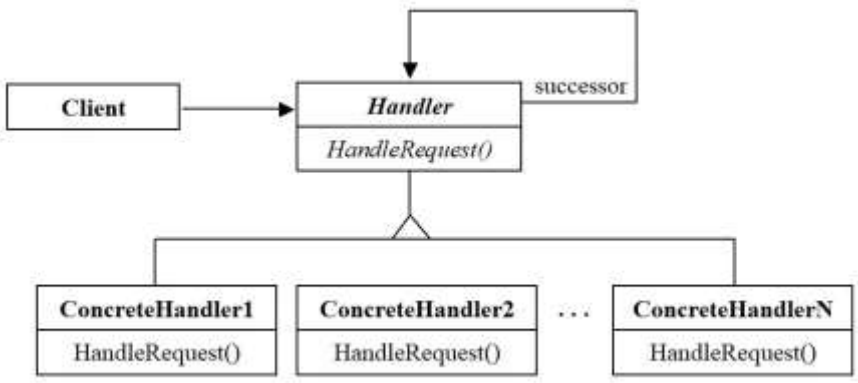

Рисунок 3.1 Структура паттерну Chain of Responsibility

Похідні класи знають, як обробляти запити клієнтів. Якщо «поточний» об’єкт неспроможна обробити запит, він делегує його базовому класу, який делегує «наступному» об'єкту тощо. Обробники можуть робити свій внесок у обробку кожного запиту. Запит може бути переданий по всій довжині ланцюжка до останньої ланки.[5]

Шаблон Model-View-ViewModel (MVVM) – застосовується при проектуванні архітектури програми (Рис. 3.2). Спочатку був представлений спільноті Джоном

Госсманом (John Gossman) в 2005 як модифікація шаблону Presentation Model. MVVM орієнтований на сучасні платформи розробки, такі як Windows Presentation Foundation .[5]

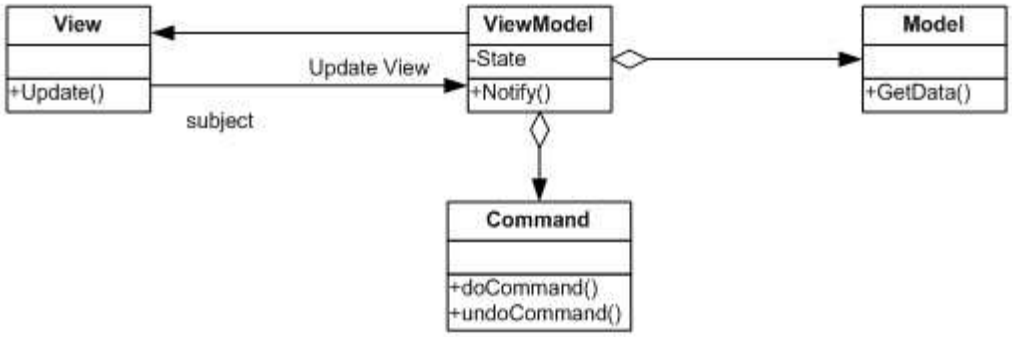

Рисунок 3.2 Діаграма класів шаблона MVVM

MVVM використовується для розділення моделі та її уявлення, що необхідно для зміни їх окремо один від одного. Наприклад, розробник задає логіку роботи з даними, а дизайнер відповідно працює з інтерфейсом користувача.

Розробка проекту буде проводитись на основі побудови моделей. Як мову моделювання вибрано UML. Мова UML є стандартом мови, що дозволяє чітко повідомляти вимоги до архітектури та дазайну.[14]

Як мову програмування вибрано C#. Вибір обумовлений тим, що ця мова має багату бібліотеку класів, що дозволяє зручно проектувати як графічний інтерфейс, так і взаємодіяти з базою даних. Також C# постійно розвивається та вдосконалюється, додаються нові функції.

Вибір інтерфейсу припав на користь віконного, оскільки для нього є безліч автоматизованих засобів проектування, а також він надає більш широкий спектр можливостей порівняно з web-інтерфейсом.

Для структурованого зберігання інформації про заявки було прийнято рішення використовувати реляційну базу даних, як СУБД – MS SQL Server 2012, а як інформаційно-логічну мову баз даних – SQL. Вибір обумовлений тим, що SQL –

стандартна мова для роботи з базами даних, яка отримала досить широке поширення. Майже всі найбільші розробники СУБД нині виробляють свої товари з використанням мови SQL чи з SQL - интерфейсом. Він став частиною архітектури додатків, є стратегічним вибором багатьох великих та впливових організацій.

Логічна та фізична модель інформаційної системи виконується за допомогою case – засобу ERWin Data Modeler. Інтегроване середовище розробки – Microsoft Visual Studio.

Відповідно до моделі предметної області були спроектовані такі класи, представлені на діаграмі класів (Рис. 3.3).

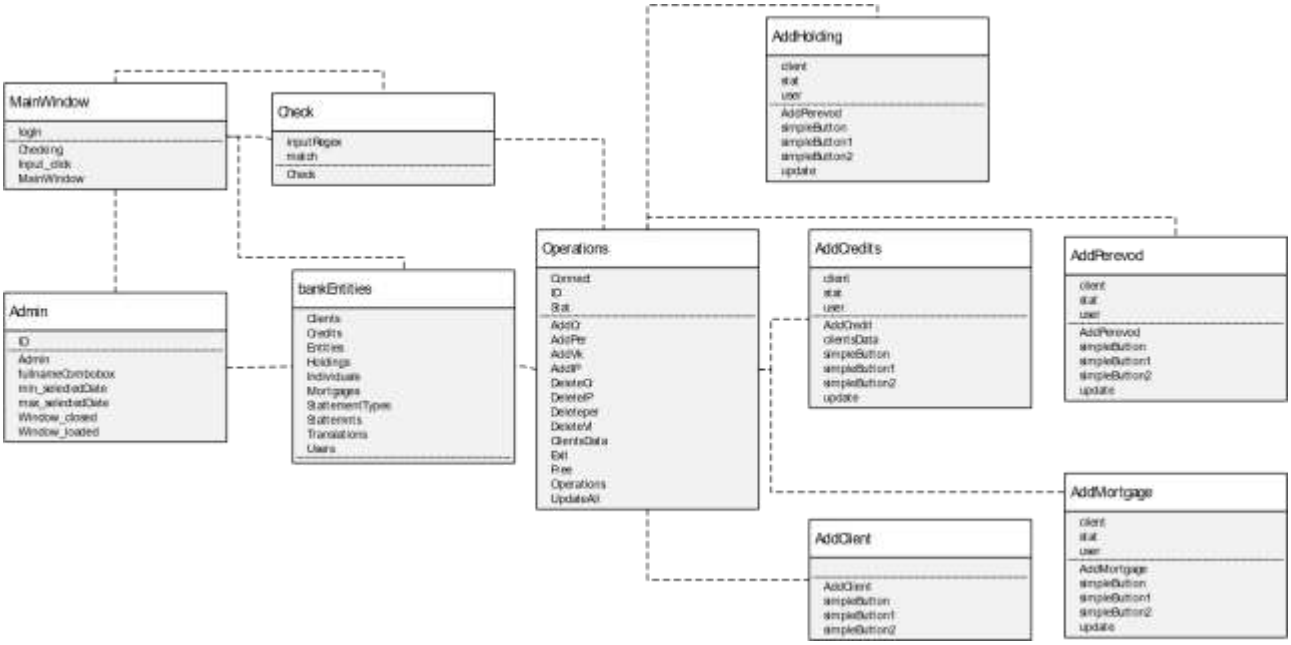

Рисунок 3.3 Діаграма класів для додатку

На основі шаблону та моделі предметної області було спроектовано 8 інтерфейсних класів та 4 класи сутностей:

Інтерфейсні класи:

MainWindow – вікно авторизації користувача;

Admin – вікно для роботи адміністратора відділення;

Operations – вікно для роботи операційніста відділення;

AddClient – вікно для додавання нових клієнтів до системи;

AddCredit – вікно для оформлення нових заявок на кредитування;

AddPerevod – вікно для оформлення нових заявок на переклад;

AddHoldings – вікно для оформлення нових заявок на вклад

Класи – сутності:

bankEntityes – клас – модель предметної області, що містить всю інформацію про таблиці та процедури БД;

Consultant – абстрактний клас, що містить необхідні методи та властивості для конкретних консультантів;

ConcreteConsultant – клас, що містить інформацію щодо конкретного консультанта;

Check – клас, що перевіряє правильність введення логіну та пароля з клавіатури

3.2 Алгоритмізація та реалізація комплексу задач автоматизації

Модель реалізації визначає, як реалізуються у вигляді компонентів елементи моделі проектування.

В Уніфікованому процесі розробки для відображення рішень реалізації найчастіше застосовуються діаграми компонентів. Діаграма компонентів для рішення (рис. 3.5).

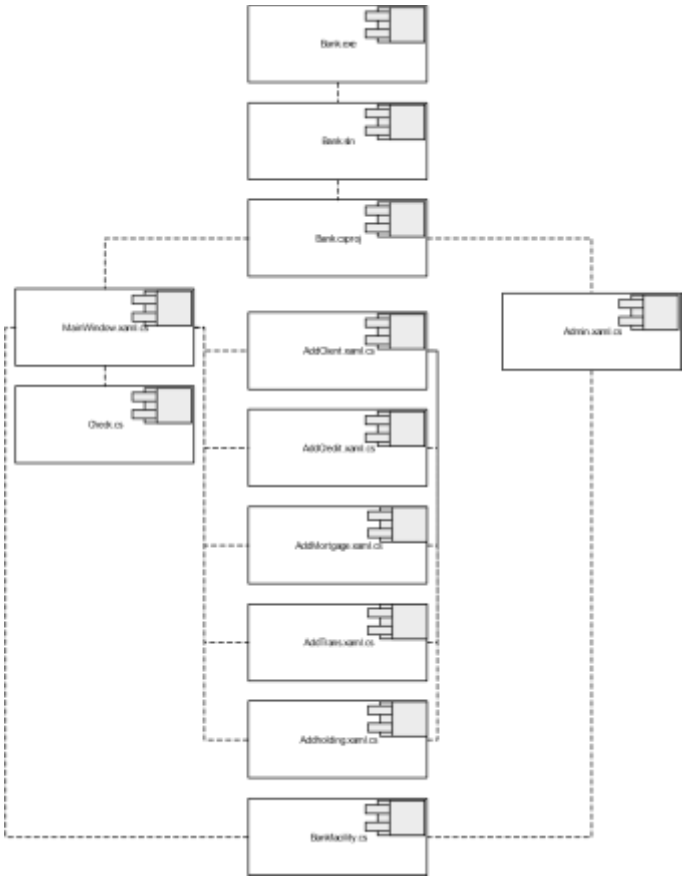

Рисунок 3.5 Діаграма компонентів

Одним із способів моделювання статичного вигляду системи з погляду розгортання є діаграма розгортання. Діаграма розгортання представлена на рисунку 3.6.

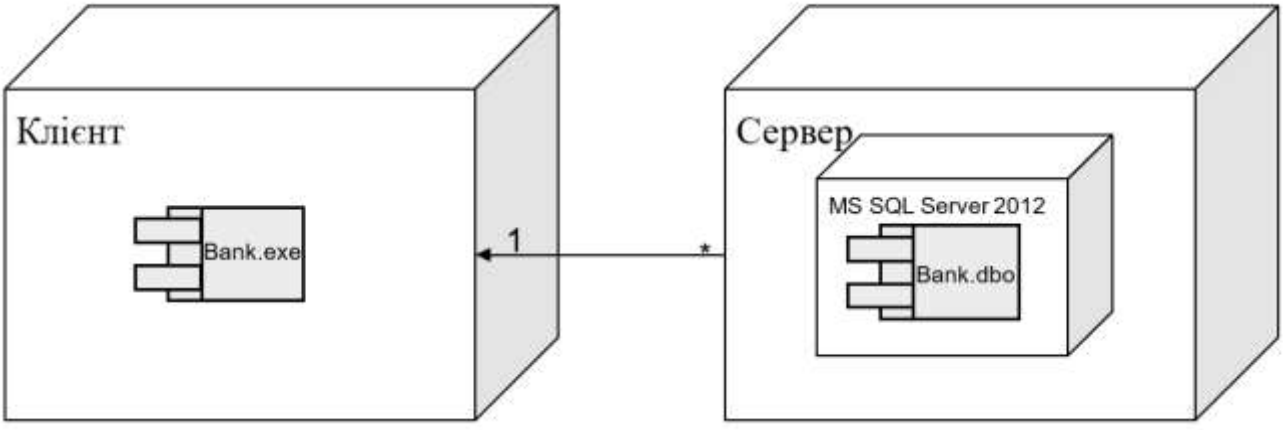

Рисунок 3.6 Діаграма розгортання

На рисунку 3.6 видно, що клієнтський додаток розгортається на робочих станціях користувача, а база даних розгортається на окремому сервері, який є одночасно сервером інтеграційної побудови.

3.2.1 Інтерфейс користувача клієнтської програми

Виходячи з аналізу вимог і моделі аналізу, був спроектований простий і інтуїтивно зрозумілий графічний інтерфейс користувача. Розглянемо докладно головне вікно інтерфейсу користувача для клієнта системи (Рис 3.7).

Рисунок 3.7 Візуальні компоненти графічного інтерфейсу головної форми програми «Клієнт»

Це вікно є основним для клієнта і дозволяє користувачеві вибрати потрібний йому вид консультації. Компоненти та їх дії описані у таблиці 3.1.

Таблиця 3.1 Конструкції вікна створення заявки

Найменування елемента форми	Тип елемента форми	Дія користувача	Відгук системи

Кредитування	Кнопка	одинарне клацання лівою кнопкою миші	Додати до системи нову заявку на кредитування
Вклади	Кнопка	одинарне клацання лівою кнопкою миші	Додати до системи нову заявку на вклад
Перекази	Кнопка	одинарне клацання лівою кнопкою миші	Додати до системи нову заявку на переказ

Вікно входу до системи представлене на рисунку 3.8.

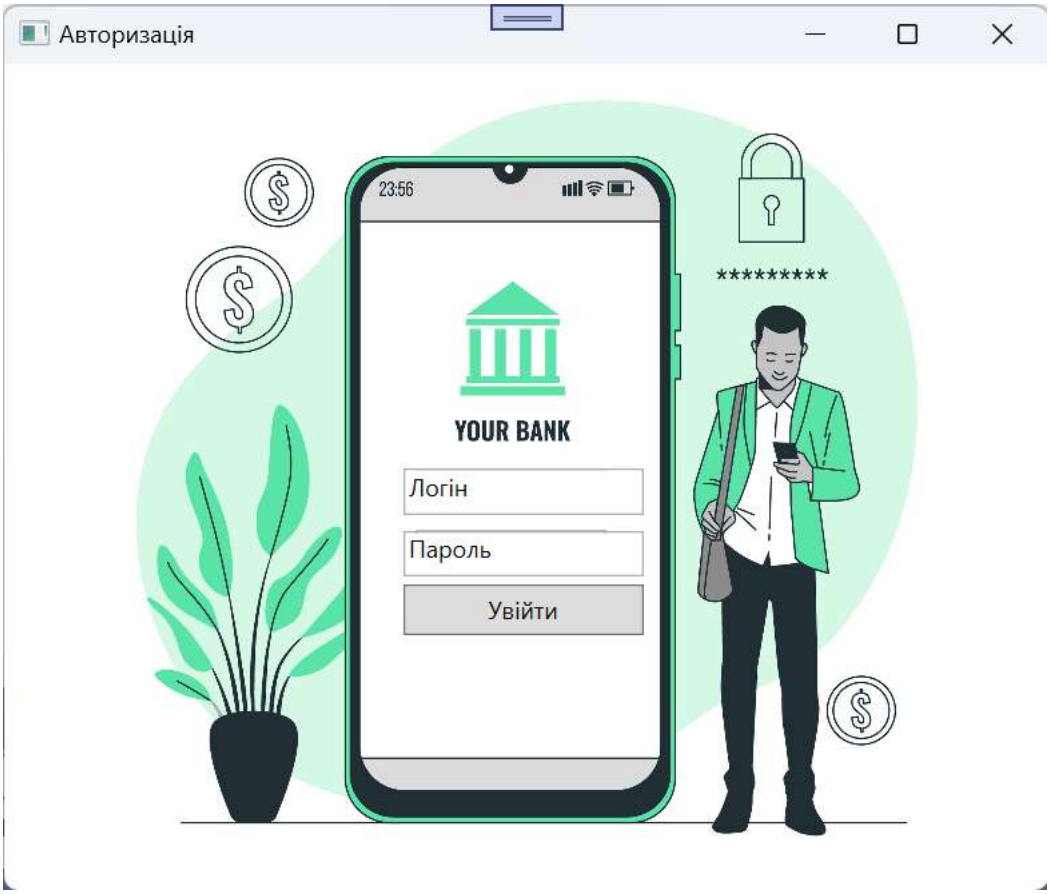

Рисунок 3.8 Візуальні компоненти графічного інтерфейсу вікна входу до системи

Це вікно призначене для авторизації користувача в системі. Кожен користувач у базі даних має свій вид додатку, і залежно від цього виду користувачеві доступний вигляд програми. Наприклад, оператор може обробляти заявки, додавати клієнтів, коли адміністратор може переглядати статистику діяльності відділення та керувати користувачами.

У таблиці 3.2 наведено компоненти форми та їх дії.

Таблиця 3.2 Компоненти форми та їх дії.

Найменування елемента форми	Тип елемента форми	Дія користувача	Відгук системи
Поле введення логіна і пароля	Текстове поле	Введення з клавіатури	–
Увійти	Кнопка	одинарне клацання лівою кнопкою миші	система перевіряє користувача з введеним логіном і паролем

Головне вікно операціоніста представлене на рисунку 3.9.

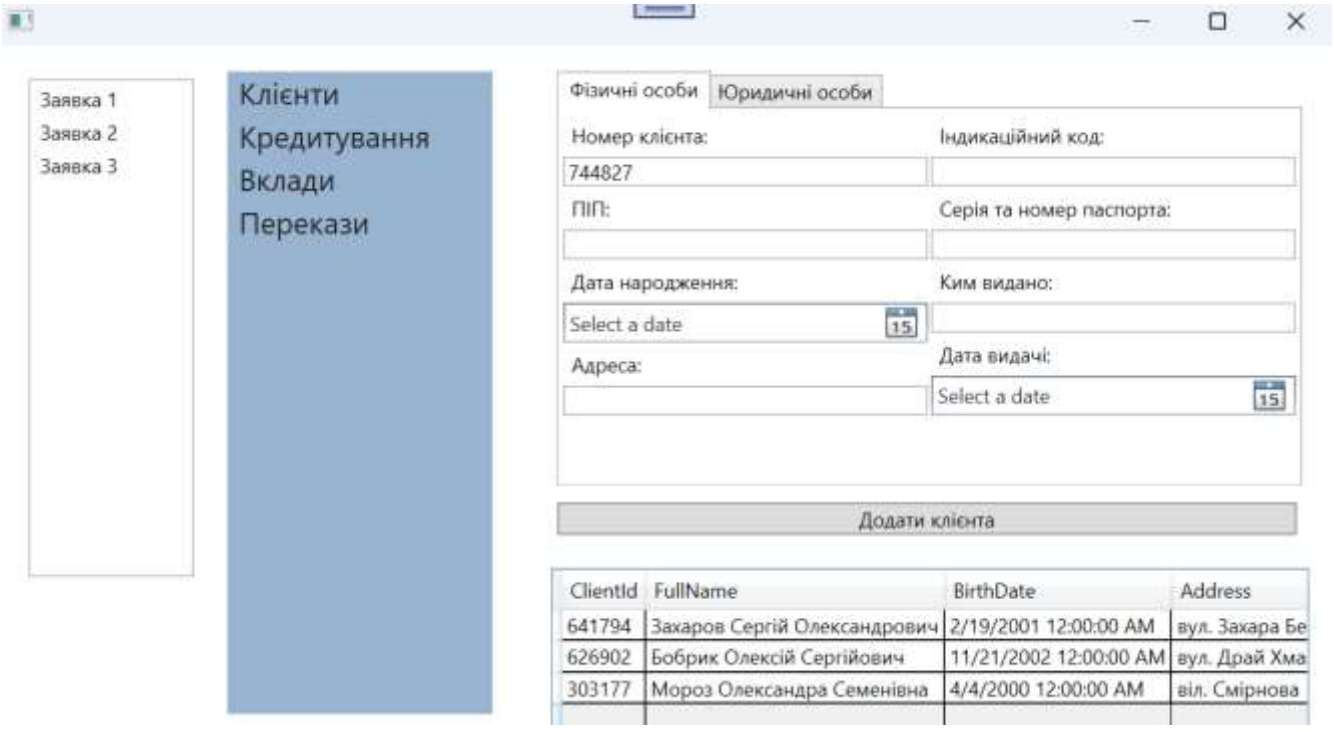

Рисунок 3.9 Головне вікно операціоніста

Вікно операціоніста призначене для обробки заявок, а саме додавання нових клієнтів, оформлення кредитів, вкладів та переказів.

Нижче в таблиці 3.3 наведено компоненти вікна головного вікна операціоніста

Таблиця 3.3 Конструкції вікна редагування інформації про складання

Найменування елемента форми	Тип елемента форми	Дія користувача	Відгук системи
Список нових заявок	Список	Клацніть правою кнопкою миші	Початок обробки заявки, відкриття потрібної форми
Меню керування	Меню	одинарне клацання лівою кнопкою миші	Система відображає певне вікно
Вкладки програми	Вкладка	одинарне клацання лівою кнопкою миші	система змінює робочу область
Текстові поля	Текст	-	-
Таблиця	Таблиця	Одинарне клацання лівою кнопкою миші	Система змінює наповнення

3.2.2 Фізична модель та реалізація бази даних системи

Фізична модель будується на основі логічної моделі даних та враховує особливості цільової СУБД. В якості цільової СУБД обрано SQL Server 2012. У ERWin можна легко перейти від логічної моделі до фізичної, при цьому:

- типи сутностей стають таблицями у фізичній базі даних;

- атрибути стають колонками у фізичній базі даних;
- унікальні ідентифікатори стають колонками, які не допускають значення NULL. У базі даних вони називаються первинними ключами (primary keys);
- зв’язки моделюються у вигляді зовнішніх ключів (foreign keys).[18]

Однією з найважливіших якостей ERWin Data Modeler є можливість проводити як пряме проектування так і зворотне. Фізична модель бази даних представлена на рисунку 3.10.

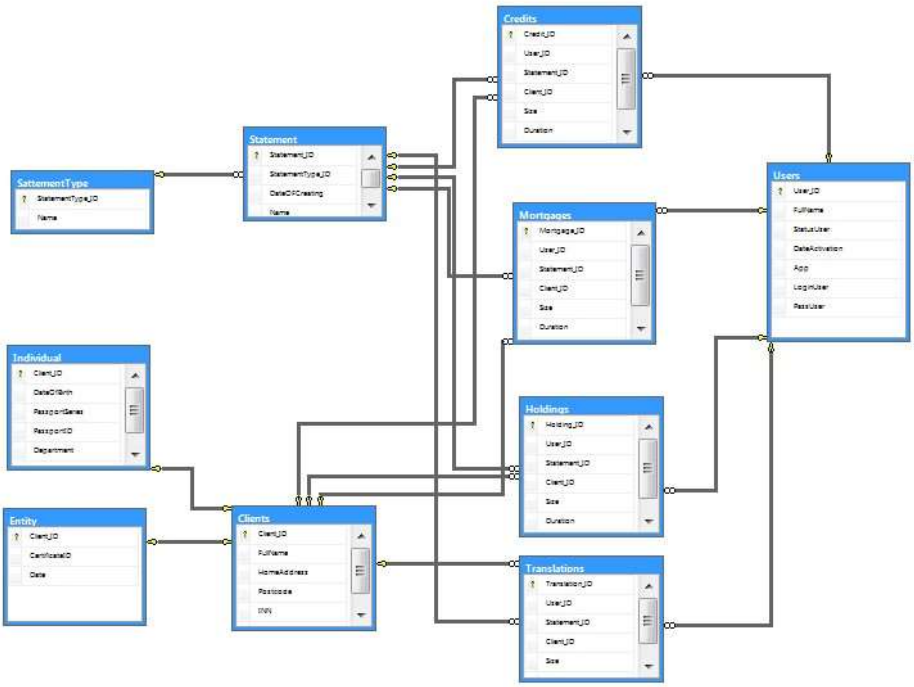

Рисунок 3.10 Фізична модель системи

Скрипт DDL згенерований ERWin (Лістинг А.1) було виконано SQL Server 2012.

У ході розробки програмних засобів підтримки безперервної інтеграції було прийнято рішення перенести логіку програми для роботи з базою даних на сервер, тому клієнтська програма не відправляє SQL-запити на сервер для отримання результатів, а викликає збережені процедури. Дане рішення обумовлено наступними перевагами збережених процедур в порівнянні з запитами.

При обробці навіть невеликих обсягів даних у зовнішньому додатку витрачається додатковий час на передачу мережі та перетворення даних у потрібний нам формат. Також відбувається значне навантаження на мережу через те, що запит необхідно сформувати на стороні клієнта, потім відправити на сервер бази даних, і отримати результат виконання запиту назад у клієнтську програму.

Зі зростанням та еволюцією програмної системи схема даних може і повинна змінюватися. Добре спроектований програмний інтерфейс на процедурах, що зберігаються, дозволить змінювати схему даних, не змінюючи код зовнішніх додатків.

При використанні SQL з боку клієнтської програми клієнтська програма передає СУБД SQL команди як рядків, попередньо формованих у коді. При використанні збережених процедур SQL код на стороні програми зазвичай статичний і виглядає як простий виклик процедури, що зберігається з подальшою передачею

Реалізуючи логіку роботи з даними в шарі ми отримуємо звичну ієрархічну модель повторного використання SQL коду.

Спрощується налагодження як клієнтської програми, так і серверної частини бази даних, у зв'язку з тим, що ми уникаємо змішування коду клієнтської програми з кодом роботи з базою даних. Семантична та синтаксична перевірка SQL проводиться на етапі компіляції.

Як процедури, були реалізовані наступні бізнес-процеси, згруповані за категоріями:

Створення заявки

```
CREATE PROCEDURE AddStatement
@Name varchar(2),
@Type int,
@output varchar(10) output
AS
Begin
Declare @id int
```

```
SET @id =((select Statement_ID from Statement where Statement_ID =
(select max(Statement_ID) from Statement)
AND StatementType_ID=@Type)+1)
SET @output =@Name + CAST(@id AS VARCHAR(5)) END
```

Отримати список опрацьованих заявок на кредитування

```
Create proc GetStatementCredits
AS
Select * from Credits
```

Отримати список опрацьованих заявок на вклади

```
Create proc GetStatementHoldings
AS
Select * from Holdings
```

Отримати список опрацьованих заявок на перекази

```
Create proc GetStatementTranslations
AS
Select * from Translations
```

Отримати список всіх клієнтів

```
Create proc GetClients
AS
Select * from Clients
```

Авторизація користувачів в системі

```
Create PROC [dbo].[GetAppUser]
@Login VARCHAR(30), @Password VARCHAR(30)
AS
SELECT Users.App
FROM Users
WHERE  Users.LoginUser = @Login AND Users.PassUser=@Password
```

Реєстрація нової Фізичної особи

```
CREATE PROC AddIndividual
@fullname varchar(20), @Home varchar(20),
@PostCode varchar(20),
@INN varchar(20),
@Account varchar(20),
@Date datetime,
@PassportSeries varchar(20),
```

```
@passport_ID varchar(20),
@department varchar(20),
@Duration datetime
AS
BEGIN
DECLARE @ID int
INSERT INTO Clients
(FullName,HomeAddress,Postcode,INN,Account) values
(@fullname,@Home,@PostCode,@INN,@Account) SET @ID =(SELECT MAX(Client_ID) FROM Clients)
INSERT INTO Individual (Client_ID,DateOFBirth,PassportSeries,PassportID,Department,Dur ation) values
(@ID,@Date,@PassportSeries,@passport_ID,@department,@Durati on)
END
```

Реєстрація нової юридичної особи

```
CREATE PROC AddEntity
@fullname varchar(20), @Home varchar(20),
@PostCode varchar(20),
@INN varchar(20),
@Account varchar(20),
@Certificate varchar(20),
@Date datetime
AS
BEGIN
DECLARE @ID int
INSERT INTO Clients
(FullName,HomeAddress,Postcode,INN,Account) values
(@fullname,@Home,@PostCode,@INN,@Account) SET @ID =(SELECT MAX(Client_ID) FROM Clients) INSERT INTO Entity (Client_ID,CertificateID,Date) values (@ID,@Certificate,@Date) END
```

3.3 Інструкції користувача

Після запуску програми, відкривається форма для авторизації користувачів (Рисунок 3.11-3.12).

У полі «Логін» потрібно ввести ім’я користувача або ідентифікатор, а у полі «Пароль» ввеести пароль. В полі «Пароль» символи будуть відображатися як крапки для захисту особистих даних.

Після введення логіна та пароля потрібно натиснути кнопку «Увійти», щоб здійснити вхід в систему.

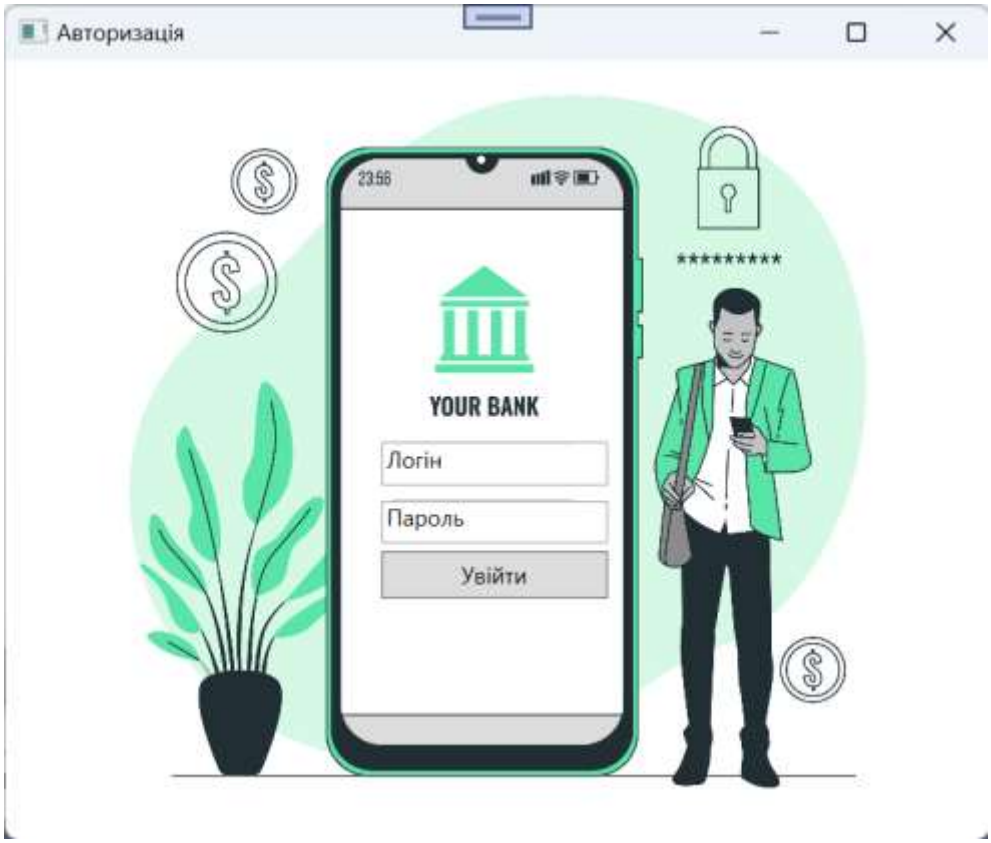

Рисунок 3.11 Авторизація клієнта

Рисунок 3.12 Авторизація клієнта

Система перевірить введені дані. Якщо вони вірні, буде повідомлено про успішний вхід (Рис. 3.13). Після натиснення кнопки «ОК», в залежності від типу користувача, буде відкрито головне вікно програми.

У випадку неправильного введення логіна чи пароля, побачите повідомлення про помилку (Рис. 3.14).

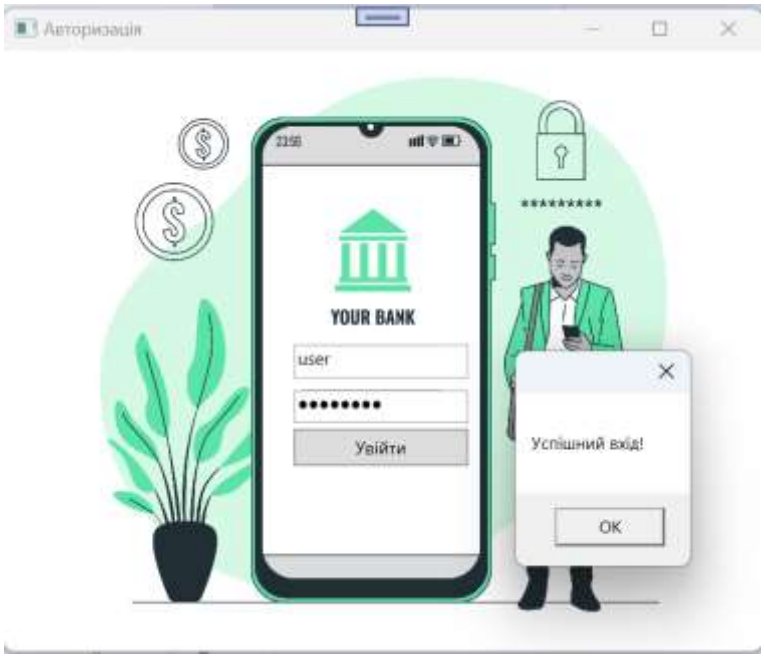

Рисунок 3.13 Успішна авторизація клієнта

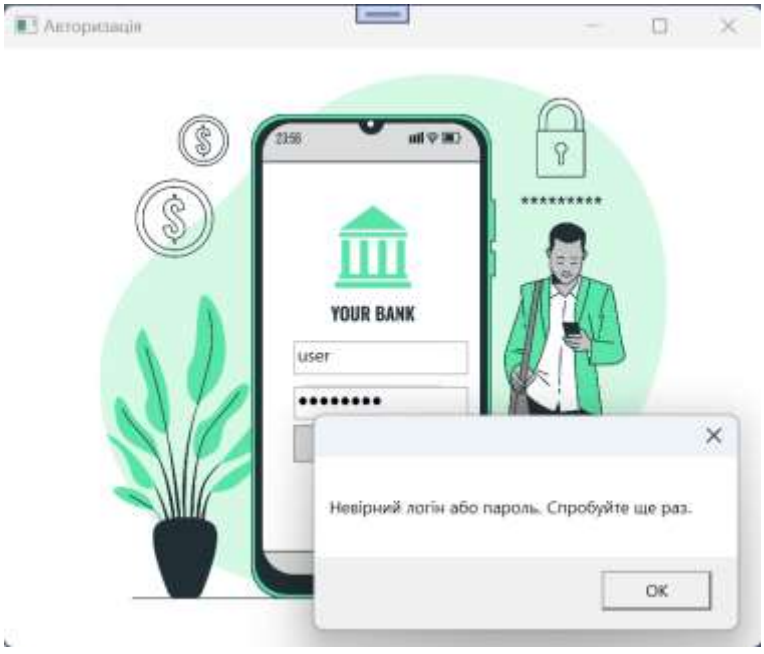

Рисунок 3.14 Повідомлення про помилку авторизації клієнта

В залежності від типу користувача, буде відкрито відповідне головне вікно програми. Для клієнтів банку головне вікно програми виглядає як на рисунку 3.15.

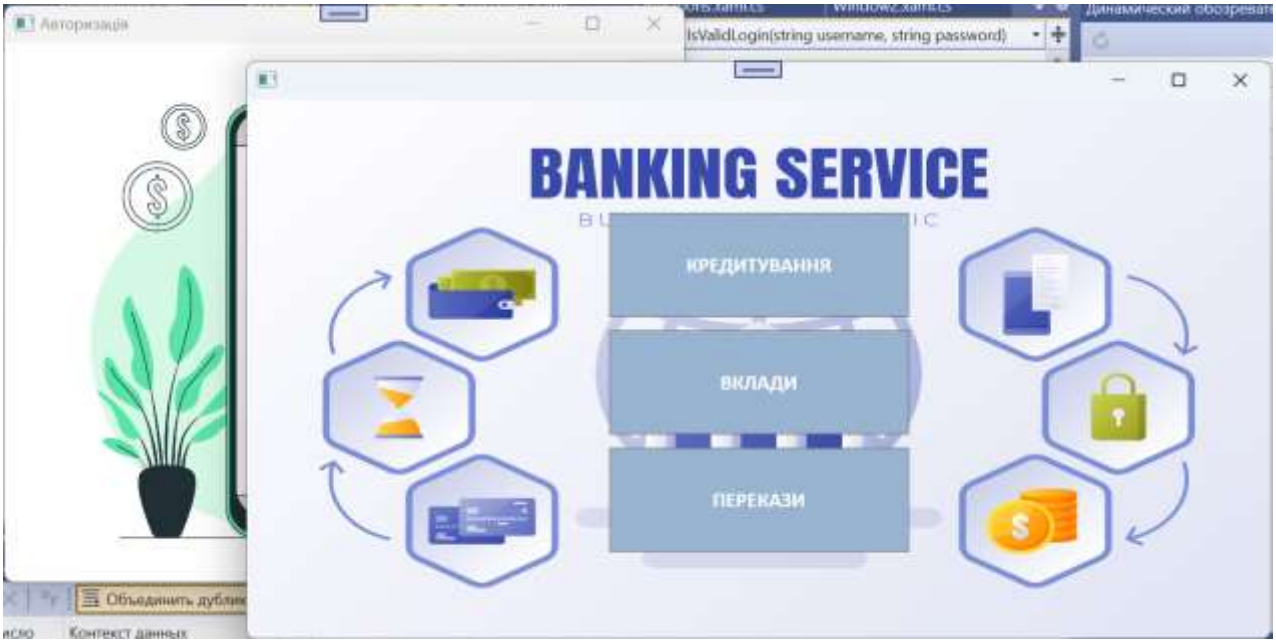

Рисунок 3.15 Головне клієнтське вікно

Тут користувач може створити заяки на кредитування\вклади\перекази (Рис. 3.16-3.18).

Рисунок 3.16 Створення заявки «Вклади»

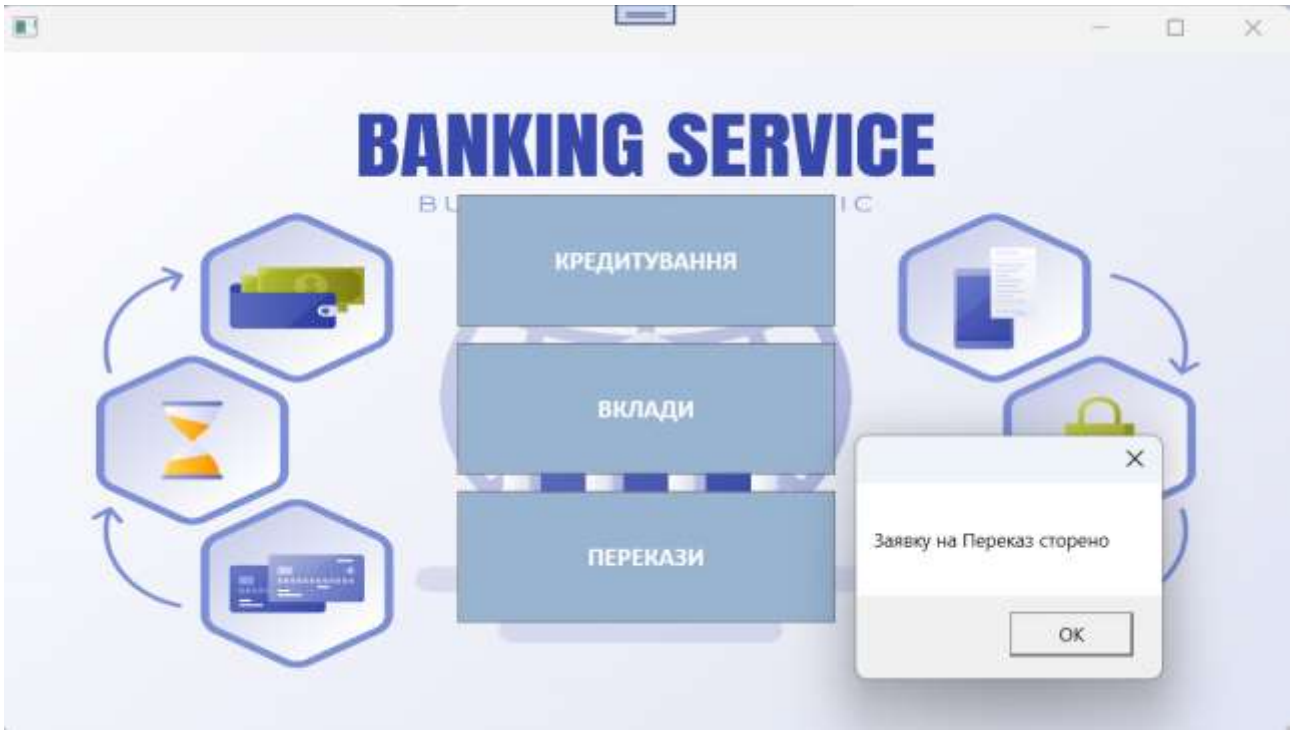

Рисунок 3.17 Створення заявки «Перекази»

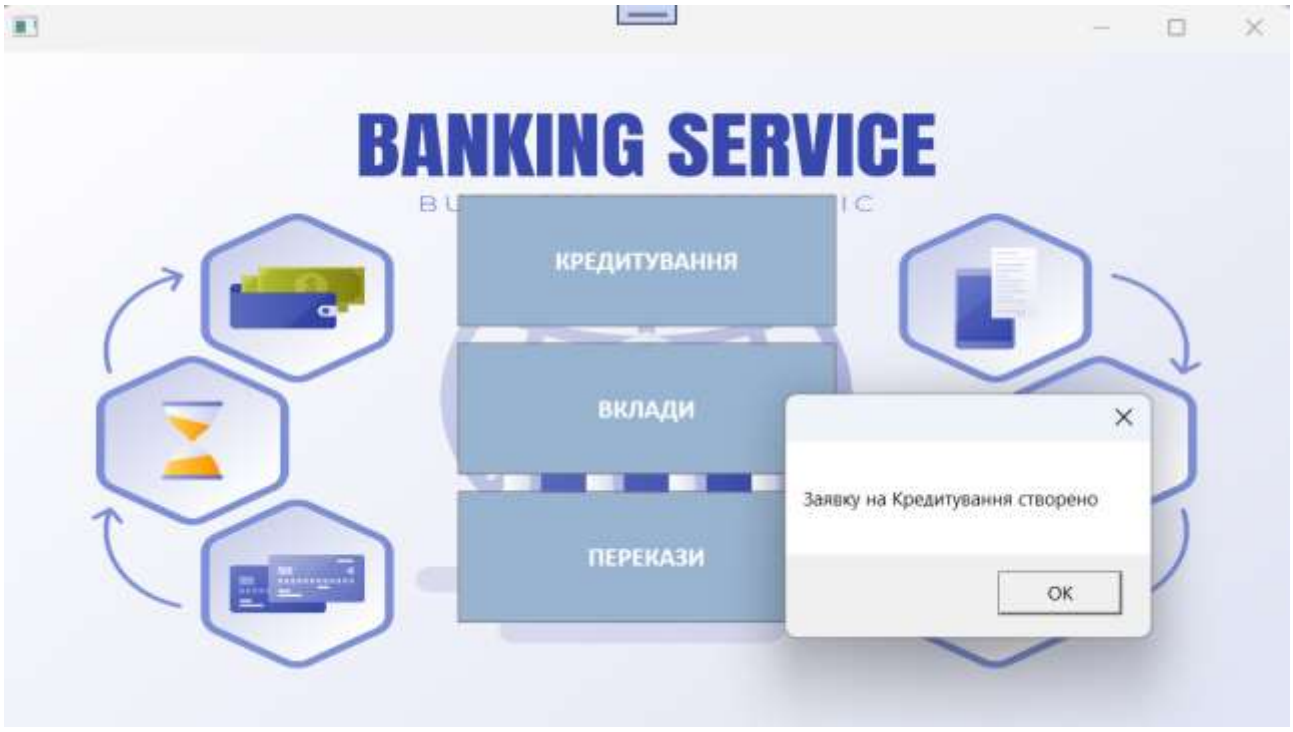

Рисунок 3.18 Створення заявки «Кредитування»

Для операторів банку головне вікно програми виглядає як на рисунку 3.20.

Рисунок 3.19 Успішна авторизація оператора

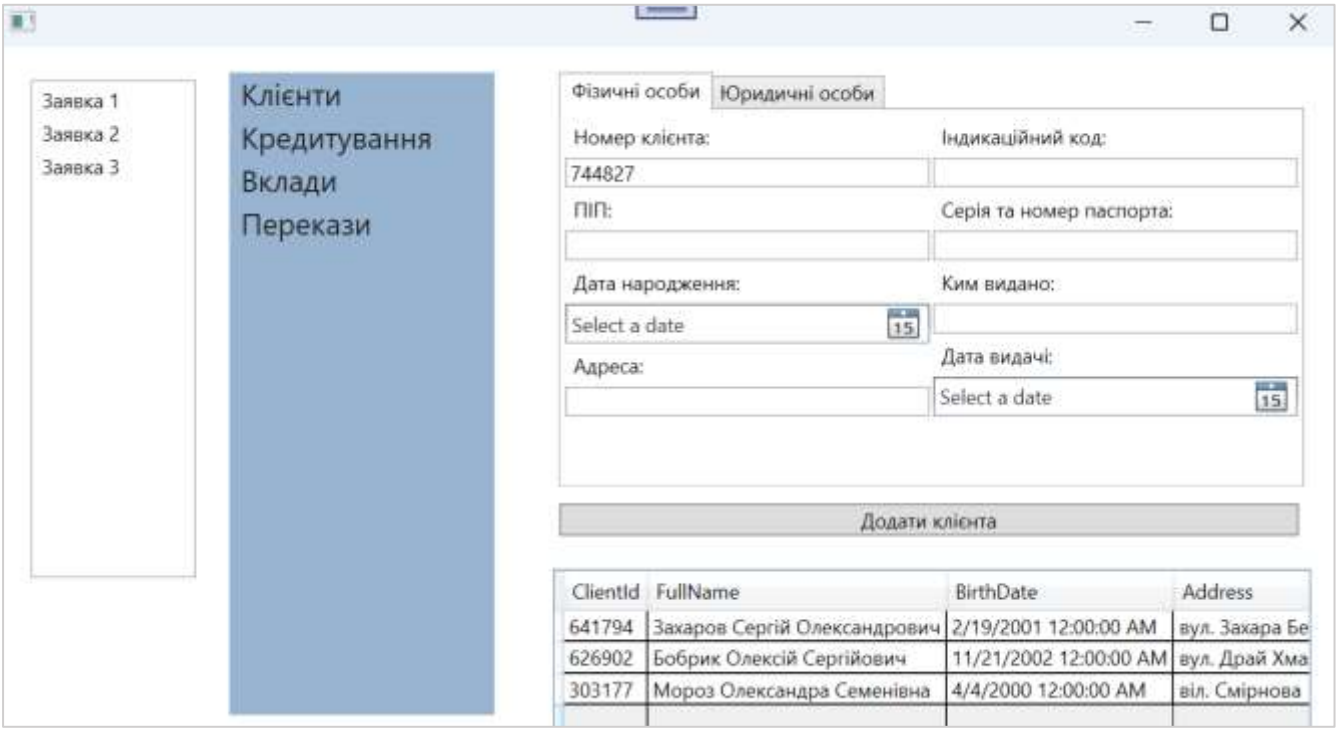

Рисунок 3.20 Головне вікно програми оператора

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА ТЕХНІКА БЕЗПЕКИ

Охорона праці при роботі з інформаційною системою, призначеною для підвищення ефективності функціонування банку, має ряд особливостей, які слід враховувати.

Однією з особливостей є те, що інформація, яка обробляється такою системою, може бути цінною та конфіденційною. Тому необхідно забезпечити її захист від несанкціонованого доступу, використання, зміни або знищення.

Для цього необхідно використовувати такі заходи безпеки, як:

- Створення систем захисту інформації, які включають в себе фізичні, технічні та програмні заходи.
- Встановлення правил доступу до інформації, які визначають, хто має право отримувати доступ до інформації та в яких обсягах.
- Проведення навчання персоналу з питань інформаційної безпеки.

Іншою особливістю є те, що робоче середовище, в якому використовується така система, може бути різним. Наприклад, система може використовуватися в офісі або вдома. У кожному випадку необхідно забезпечити відповідність робочого середовища вимогам безпеки. Для цього необхідно дотримуватися таких вимог:

- Забезпечення достатнього освітлення, щоб екран комп'ютера не стомлював очі.
- Забезпечення достатньої вентиляції, щоб повітря в робочому середовищі було свіжим.
- Забезпечення шумоізоляції, щоб шум не заважав роботі.

Особливості обладнання та програмного забезпечення, які використовуються для роботи з такою системою, можуть впливати на вимоги охорони праці та техніки безпеки. Тому необхідно уважно вивчити ці вимоги перед початком роботи з системою. Проведення навчання персоналу з питань інформаційної безпеки.

Основні вимоги охорони праці при роботі з інформаційною системою, призначеною для підвищення ефективності функціонування банку, включають:

Забезпечення безпеки обладнання та програмного забезпечення, які використовуються для роботи з системою. Обладнання повинно бути сертифіковане та відповідати вимогам безпеки. Програмне забезпечення повинно мати ліцензію та бути оновлене до останньої версії.

Забезпечення безпеки робочого середовища, в якому використовується система. Робоче середовище повинно відповідати вимогам безпеки, зокрема, повинні бути забезпечені достатнє освітлення, вентиляція та шумоізоляція.

Заборона доступу до системи сторонніх осіб без дозволу. Доступ до системи повинен бути обмежений лише авторизованими користувачами.

Виконання інструкцій з експлуатації обладнання та програмного забезпечення. Користувачі повинні уважно вивчити та виконувати інструкції з експлуатації обладнання та програмного забезпечення.

Проведення регулярних перевірок обладнання та програмного забезпечення на наявність пошкоджень та дефектів. Обладнання та програмне забезпечення необхідно регулярно перевіряти на наявність пошкоджень та дефектів.

Виконання профілактичних заходів, спрямованих на запобігання виникненню аварій та інших небезпек. Слід виконувати профілактичні заходи, спрямовані на запобігання виникненню аварій та інших небезпек.

Впровадження заходів безпеки у разі виникнення аварії або інших небезпек. У разі виникнення аварії або інших небезпек необхідно впроваджувати заходи безпеки для захисту персоналу та інформації.

Важливою умовою забезпечення охорони праці при роботі з інформаційною системою є проведення навчання персоналу з питань охорони праці та техніки безпеки. Персонал повинен бути ознайомлений з вимогами безпеки, які стосуються роботи з системою, а також повинен навчитися дотримуватися цих вимог.

Дотримання вимог охорони праці при роботі з інформаційною системою, призначеною для підвищення ефективності функціонування банку, допоможе запобігти виникненню аварій та інших небезпек, а також захистити персонал та інформацію.

Охорона праці та техніки безпеки при виконанні робіт з ПК є важливим питанням, яке необхідно враховувати на всіх етапах роботи з комп'ютером.

Безпеку обладнання. Обладнання, яке буде використовуватися для роботи з ПК, повинно бути сертифіковане та відповідати вимогам безпеки.

Безпеку робочого середовища. Робоче середовище, в якому буде використовуватися ПК, повинно відповідати вимогам безпеки, зокрема, повинні бути забезпечені достатнє освітлення, вентиляція та шумоізоляція.

На етапі впровадження ПК необхідно провести навчання персоналу з питань охорони праці та техніки безпеки. Персонал повинен бути ознайомлений з вимогами безпеки, які стосуються роботи з ПК, а також повинен навчитися дотримуватися цих вимог.

На етапі експлуатації ПК необхідно забезпечити дотримання вимог безпеки. Зокрема, слід:

Проводити регулярні перевірки обладнання. Обладнання, яке використовується для роботи з ПК, необхідно регулярно перевіряти на наявність пошкоджень та дефектів.

Виконувати профілактичні заходи. Необхідно виконувати профілактичні заходи, спрямовані на запобігання виникненню аварій та інших небезпек.

Впроваджувати заходи безпеки. У разі виникнення аварії або інших небезпек необхідно впроваджувати заходи безпеки для захисту персоналу та обладнання.

На етапі ліквідації ПК необхідно забезпечити безпечне знищення обладнання та інформації, яка зберігалася на ньому.

Основні вимоги охорони праці та техніки безпеки при виконанні робіт з ПК

До основних вимог охорони праці та техніки безпеки при виконанні робіт з ПК відносяться:

Заборона використання обладнання, яке не відповідає вимогам безпеки.

Забезпечення достатнього освітлення, вентиляції та шумоізоляції робочого середовища.

Заборона використання ПК в умовах, які можуть призвести до його пошкодження або виникнення аварії.

Заборона доступу до ПК сторонніх осіб без дозволу.

Виконання інструкцій з експлуатації обладнання та програмного забезпечення.

Проведення регулярних перевірок обладнання та програмного забезпечення на наявність пошкоджень та дефектів.

Виконання профілактичних заходів, спрямованих на запобігання виникненню аварій та інших небезпек.

Впровадження заходів безпеки у разі виникнення аварії або інших небезпек.

Особливості охорони праці та техніки безпеки при роботі з ПК

Охорона праці та техніки безпеки при роботі з ПК мають ряд особливостей, які слід враховувати:

Електромагнітне випромінювання ПК може бути шкідливим для здоров'я людини. Тому необхідно дотримуватися відстані від екрана монітора не менше 50 см.

Механічне пошкодження ПК може призвести до травми людини. Тому необхідно дотримуватися правил експлуатації ПК.

Пожежна небезпека ПК може стати причиною пожежі. Тому необхідно дотримуватися правил пожежної безпеки.

Виконання вимог охорони праці та техніки безпеки при виконанні робіт з ПК є обов'язком кожного працівника, який працює з ПК. Це допоможе запобігти виникненню аварій та інших небезпек, а також захистити персонал та обладнання.

Ось деякі конкретні рекомендації щодо охорони праці та техніки безпеки при виконанні робіт з ПК:

Перед початком роботи з ПК переконайтеся, що обладнання та програмне забезпечення відповідають вимогам безпеки.

Не працюйте з ПК в умовах, які можуть призвести до його пошкодження або виникнення аварії.

Не дозволяйте стороннім особам без дозволу отримувати доступ до ПК.

Не сидіть за ПК більш ніж 2 години без перерви.

Виконуйте регулярні профілактичні заходи для запобігання виникненню аварій та інших небезпек.

Дотримання цих рекомендацій допоможе забезпечити безпечну роботу з ПК.

ВИСНОВКИ

При виконанні кваліфікаційної роботи, було розроблено програмні засоби автоматизованої інформаційної системи банківського сервісу, що складається з клієнтської програми та бази даних.

Метою кваліфікаційної роботи було розроблення інформаційної системи, спрямованої на оптимізацію та підвищення ефективності роботи банку.

Розроблена інформаційна система включає в себе функціональність для управління клієнтськими рахунками, кредитування, вкладами, переказами та іншими банківськими операціями. Інтеграція з базою даних дозволяє забезпечити надійне зберігання та обробку інформації.

Був створений інтуїтивно зрозумілий інтерфейс користувача, що спрощує взаємодію з системою. Введення елементів керування, таких як списки, меню та вкладки, забезпечує зручність та ефективність роботи користувачів.

Інформаційна система спрямована на автоматизацію рутинних банківських операцій, що дозволяє працівникам банку витрачати менше часу на адміністративні завдання та більше часу на взаємодію з клієнтами та стратегічне планування.

Було проведено концептуальне, логічне та фізичне проектування реляційної бази даних, проектування графічного інтерфейсу. База даних була реалізована в СУБД MS SQL-Server 2012.

Клієнтська програма реалізована за допомогою мови C# в інтегрованому середовищі розробки Visual Studio 2019. Серверна частина програми: процедури і тригери реалізовані за допомогою мови SQL.

Таким чином, у результаті кваліфікаційної роботи було розроблено програмне забезпечення, яке відповідає всім вимогам технічного завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Проектування та розробка програмного забезпечення[Електронний ресурс]: Лабораторний практикум до виконання лабораторних робіт для здобувачів освітнього ступеня «Бакалавр» спеціальності 122 «Комп‘ютерні науки», ОПП «Комп’ютерні науки» і «Інформаційні системи та штучний інтелект», денної і заочної форм навчання/ Укл.: О.М. М’якшило, , О.В. Харкянен– К. НУХТ, 2022. – 102 с.

2. НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями.

3. ДСанПіН 3.3.2.007-98 Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.

4. Інструкція з охорони праці при роботі на персональному комп’ютері [Електронний ресурс] – Режим доступу до ресурсу: https://pro-op.com.ua/article/485-nstruktsya-z-ohoroni-prats-pri-robot-na-personalnomu-kompyuter

5. Yuen S. Mastering Windows Presentation Foundation: Master the art of building modern desktop applications on Windows Paperback – February 17, 2017. –568 p.

6. Stephens R. WPF 3d: Three-Dimensional Graphics with WPF and C# Paperback – February 8, 2018. – 426 p.

7. Chowdhury K. Windows Presentation Foundation Development Cookbook: 100 recipes to build rich desktop client applications on Windows Paperback –February 23, 2018. - 524 p.

8. Walkthrough: My first WPF desktop application. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/wpf/gettingstarted/walkthrough-my-first-wpf-desktop-application

9. Windows Presentation Foundation (WPF) in Visual Studio. [Електронний ресурс] – Режим доступу до ресурсу: http://windowsclient.net/wpf/default.aspx

10. Tutorial: Create your first WPF application in Visual Studio 2019. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/wpf/gettingstarted/walkthrough-my-first-wpf-desktop-application

ДОДАТКИ

Додаток А. Лістинг програми

А.1 Лістинг скрипта створення таблиць бази даних на мові SQL

```
CREATE TABLE Clients
(
Client_ID            integer IDENTITY ( 1,1 ) ,
FullName             varchar(20)  NOT NULL ,
HomeAddress              varchar(20)  NOT NULL ,
Postcode             varchar(20)  NOT NULL ,
INN                  varchar(20)  NOT NULL ,
Account              varchar(20)  NOT NULL
) go


ALTER TABLE Clients
      ADD CONSTRAINT XPKClients PRIMARY KEY  CLUSTERED (Client_ID ASC) go


CREATE TABLE Credits
(
Credit_ID            integer IDENTITY ( 1,1 ) ,
User_ID              integer  NULL ,
Statement_ID         integer  NULL ,
Client_ID            integer  NULL ,
Size                 integer  NULL ,
Duration                 datetime  NULL ,
PercentCredit              integer  NOT NULL
) go


ALTER TABLE Credits
      ADD CONSTRAINT XPKCredits PRIMARY KEY  CLUSTERED (Credit_ID ASC) go


CREATE TABLE Entity
(
Client_ID            integer  NOT NULL ,
CertificateID        varchar(20)  NOT NULL ,
Date                 datetime  NOT NULL
) go


ALTER TABLE Entity
```

```
      ADD CONSTRAINT XPKEntity PRIMARY KEY  CLUSTERED (Client_ID ASC) go


CREATE TABLE Holdings
(
Holding_ID           integer IDENTITY ( 1,1 ) ,
User_ID              integer  NOT NULL ,
Statement_ID         integer  NOT NULL ,
Client_ID            integer  NOT NULL ,
Size                 integer  NOT NULL ,
Duration                 datetime  NOT NULL ,
PercentHolding              integer  NOT NULL
) go


ALTER TABLE Holdings
      ADD CONSTRAINT XPKHoldings PRIMARY KEY  CLUSTERED (Holding_ID ASC) go


CREATE TABLE Individual
(
Client_ID            integer  NOT NULL ,
DateOfBirth          datetime  NOT NULL ,
PassportSeries       varchar(20)  NOT NULL ,
PassportID           varchar(20)  NOT NULL ,
Department           varchar(20)  NOT NULL ,
Duration                datetime  NOT NULL
) go


ALTER TABLE Individual
      ADD CONSTRAINT XPKIndividual PRIMARY KEY  CLUSTERED (Client_ID ASC) go

CREATE TABLE Mortgages
(
Mortgage_ID          integer IDENTITY ( 1,1 ) ,
User_ID              integer  NOT NULL ,
Statement_ID         integer  NOT NULL ,
Client_ID            integer  NOT NULL ,
Size                 integer  NOT NULL ,
Duration                 datetime  NOT NULL ,
PercentMortgage             integer  NOT NULL
) go


ALTER TABLE Mortgages
      ADD CONSTRAINT XPKMortgages PRIMARY KEY  CLUSTERED (Mortgage_ID ASC)
go
```

```
CREATE TABLE SattementType
(
StatementType_ID     integer IDENTITY ( 1,1 ) ,
Name                 char(18)  NULL
) go


ALTER TABLE SattementType
ADD CONSTRAINT XPKStatementTypes PRIMARY KEY  CLUSTERED (StatementType_ID
ASC) go


CREATE TABLE Statement
(
Statement_ID         integer IDENTITY ( 1,1 ) ,
StatementType_ID     integer  NULL ,
DateOFCreating                datetime  NULL ,
Name                 varchar(20)  NULL
) go


ALTER TABLE Statement
      ADD CONSTRAINT XPKStatements PRIMARY KEY  CLUSTERED (Statement_ID ASC)
go


CREATE TABLE Translations
(
Translation_ID       integer IDENTITY ( 1,1 ) ,
User_ID              integer  NOT NULL ,
Statement_ID         integer  NOT NULL ,
Client_ID            integer  NOT NULL ,
Size                 integer  NOT NULL ,
Account              varchar(20)  NOT NULL
)
go


ALTER TABLE Translations
      ADD CONSTRAINT XPKTranslations PRIMARY KEY  CLUSTERED (Translation_ID
ASC) go


CREATE TABLE Users
(
User_ID              integer IDENTITY ( 1,1 ) ,
```

```
	FullName		varchar(20)  NOT NULL ,
	StatusUser		    varchar(20)  NOT NULL ,
	DateActivation		datetime  NOT NULL
	) go


	ALTER TABLE Users
		ADD CONSTRAINT XPKUsers PRIMARY KEY  CLUSTERED (User_ID ASC) go



	ALTER TABLE Credits
	ADD CONSTRAINT R_4 FOREIGN KEY (User_ID) REFERENCES Users(User_ID)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
	go



	ALTER TABLE Credits
		ADD CONSTRAINT R_9 FOREIGN KEY (Statement_ID) REFERENCES
Statement(Statement_ID)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
	go



	ALTER TABLE Credits
	ADD CONSTRAINT R_16 FOREIGN KEY (Client_ID) REFERENCES Clients(Client_ID)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
	go



	ALTER TABLE Entity
	ADD  FOREIGN KEY (Client_ID) REFERENCES Clients(Client_ID)
		ON DELETE CASCADE
			ON UPDATE CASCADE go


	ALTER TABLE Holdings
	ADD CONSTRAINT R_6 FOREIGN KEY (User_ID) REFERENCES Users(User_ID)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
	go
```

```
	ALTER TABLE Holdings
		ADD CONSTRAINT R_11 FOREIGN KEY (Statement_ID) REFERENCES
Statement(Statement_ID)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
	go



	ALTER TABLE Holdings
	ADD CONSTRAINT R_14 FOREIGN KEY (Client_ID) REFERENCES Clients(Client_ID)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
	go



	ALTER TABLE Individual
	ADD  FOREIGN KEY (Client_ID) REFERENCES Clients(Client_ID)
		ON DELETE CASCADE
		ON UPDATE CASCADE
	go



	ALTER TABLE Mortgages
	ADD CONSTRAINT R_5 FOREIGN KEY (User_ID) REFERENCES Users(User_ID)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
	go



	ALTER TABLE Mortgages
		ADD CONSTRAINT R_10 FOREIGN KEY (Statement_ID) REFERENCES
Statement(Statement_ID)
		ON DELETE NO ACTION
			ON UPDATE NO ACTION go



	ALTER TABLE Mortgages
	ADD CONSTRAINT R_15 FOREIGN KEY (Client_ID) REFERENCES Clients(Client_ID)
```

```
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
      go  ALTER TABLE Statement
            ADD CONSTRAINT R_8 FOREIGN KEY (StatementType_ID) REFERENCES
SattementType(StatementType_ID)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
      go




      ALTER TABLE Translations
      ADD CONSTRAINT R_7 FOREIGN KEY (User_ID) REFERENCES Users(User_ID)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
      go

      ALTER TABLE Translations
            ADD CONSTRAINT R_12 FOREIGN KEY (Statement_ID) REFERENCES
Statement(Statement_ID)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
      go

      ALTER TABLE Translations
      ADD CONSTRAINT R_13 FOREIGN KEY (Client_ID) REFERENCES Clients(Client_ID)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
      go
```

А.2 Листинг Autorization.cs

```
      namespace Login
      {
          public partial class MainWindow : Window
          {
              bankEntities login;
               public MainWindow()
              {
                  InitializeComponent();
                   login = new bankEntities();
              }
              private void Checking(object sender, TextCompositionEventArgs e)
              {
                  if ((sender is TextBox))
                  {
                      if (Check.check(e.Text, (sender as TextBox).Text))
                      {
                          e.Handled = true;
                       }
                       return;
                  }
```

```
                if ((sender is PasswordBox))
                {
                    if (Check.check(e.Text, (sender as PasswordBox).Password))
                    {
                        e.Handled = true;
                    }
                }
            }
            private void Input_Click(object sender, RoutedEventArgs e)          {
                if (LoginText.Text == "" && passwordBox.Password == "")
                {
                    MessageBox.Show("Введіть логін та пароль", "Авторизація",
MessageBoxButton.OK, MessageBoxImage.Exclamation);
                     return;
                }
                foreach (GetAppUser_Result user in
login.GetAppUser(LoginText.Text, passwordBox.Password))              {
                    using (bankEntities ctx = new bankEntities())
                    {
                        var update = ctx.Users.Where(u => u.User_ID ==
user.User_ID).Update(u => new User { StatusUser = 1 });
                    }
                    switch (user.app)
                    {
                         case 1:
                            var form = new Operations(user.User_ID);
                             form.Show();
                             break;
                         case 2:
                            var form1 = new Admin(user.User_ID);
                             form1.Show();
                       break;
                    }
                this.Close();
                 }
                 this.Cursor = Cursors.Arrow;
            }
        }
    }
```

А.3 Лістинг Operations.cs

```
    namespace Login
    {
        public partial class Operations : Window
        {         int ID;
             int stat;
             bankEntities connect;
             public Operations(int id )
            {
                InitializeComponent();
                 ID = id;
                connect = new bankEntities();
            }
            private void Exit(object sender, RoutedEventArgs e)
            {
                using (bankEntities ctx = new bankEntities())
```

```
                {
                    var update = ctx.Users.Where(u => u.User_ID == ID).Update(u
=> new User { StatusUser = 0});
                }
                this.Close();
            }
            private void Window_Loaded(object sender, RoutedEventArgs e)
            {
                Login.bankDataSet bankDataSet =
((Login.bankDataSet)(this.FindResource("bankDataSet")));
                Login.bankDataSetTableAdapters.GetStatementListTableAdapter
bankDataSetGetStatementListTableAdapter = new
Login.bankDataSetTableAdapters.GetStatementListTableAdapter();
bankDataSetGetStatementListTableAdapter.Fill(bankDataSet.GetStatementList, ID);
                System.Windows.Data.CollectionViewSource
getStatementListViewSource
    =
((System.Windows.Data.CollectionViewSource)(this.FindResource("getStatementListVie
wSource")));
                getStatementListViewSource.View.MoveCurrentToFirst();



                Login.bankDataSetTableAdapters.GetIndividualClientsTableAdapter
bankDataSetGetIndividualClientsTableAdapter = new
Login.bankDataSetTableAdapters.GetIndividualClientsTableAdapter();
bankDataSetGetIndividualClientsTableAdapter.Fill(bankDataSet.GetIndividualClients)
;
                System.Windows.Data.CollectionViewSource
getIndividualClientsViewSource =
((System.Windows.Data.CollectionViewSource)(this.FindResource("getIndividualClient
sViewSource")));
            getIndividualClientsViewSource.View.MoveCurrentToFirst();


                Login.bankDataSetTableAdapters.GetEntityClientsTableAdapter
bankDataSetGetEntityClientsTableAdapter = new
Login.bankDataSetTableAdapters.GetEntityClientsTableAdapter();
bankDataSetGetEntityClientsTableAdapter.Fill(bankDataSet.GetEntityClients);
                System.Windows.Data.CollectionViewSource
getEntityClientsViewSource
    =
((System.Windows.Data.CollectionViewSource)(this.FindResource("getEntityClientsVie
wSo urce")));
                getEntityClientsViewSource.View.MoveCurrentToFirst();


                Login.bankDataSetTableAdapters.ClientsTableAdapter
bankDataSetClientsTableAdapter = new
Login.bankDataSetTableAdapters.ClientsTableAdapter();
bankDataSetClientsTableAdapter.Fill(bankDataSet.Clients);
System.Windows.Data.CollectionViewSource clientsViewSource =
((System.Windows.Data.CollectionViewSource)(this.FindResource("clientsViewSource")
));
                 clientsViewSource.View.MoveCurrentToFirst();
                UpdateAll();
            }
```

```
            private void statement_IDListBox_SelectionChanged(object sender,
System.Windows.Controls.SelectionChangedEventArgs e)
            {
                var item = statement_IDListBox.SelectedItem;
                 using (bankEntities ctx = new bankEntities())
                {
                    var update = ctx.Users.Where(u => u.User_ID == ID).Update(u
=> new User { StatusUser = 2});
                }             int id =
      (int)((DataRowView)statement_IDListBox.SelectedItem).Row["StatementType_ID"]
;             stat=
      (int)((DataRowView)statement_IDListBox.SelectedItem).Row["Statement_ID"];
                 switch (id)
                {
                     case 1:
                        Credit.IsSelected = true;
                         break;
                     case 2:
                        Vklad.IsSelected = true;
                         break;
                     case 3:
                        Perevod.IsSelected = true;
                         break;
                }
            }          private void clientsDataGrid_SelectionChanged(object
sender, System.Windows.Controls.SelectionChangedEventArgs e)         {
                Login.bankDataSet bankDataSet =
((Login.bankDataSet)(this.FindResource("bankDataSet")));             int id =
      (int)((DataRowView)getIndividualClientsDataGrid.SelectedItems[0]).Row["Clien
t_ID
      "];


                Login.bankDataSetTableAdapters.GetClientTableAdapter
           bankDataSet-
      GetClientTableAdapter = new Log-
      in.bankDataSetTableAdapters.GetClientTableAdapter();
                bankDataSetGetClientTableAdapter.Fill(bankDataSet.GetClient,
id);
                System.Windows.Data.CollectionViewSource getClientViewSource =
      ((System.Windows.Data.CollectionViewSource)(this.FindResource("getClientView
Source"))
      );
                getClientViewSource.View.MoveCurrentToFirst();
             Login.bankDataSetTableAdapters.GetIndividualListTableAdapter
bankDataSetGetIndividualListTableAdapter = new
Login.bankDataSetTableAdapters.GetIndividualListTableAdapter();

bankDataSetGetIndividualListTableAdapter.Fill(bankDataSet.GetIndividualList, id);
                System.Windows.Data.CollectionViewSource
getIndividualListViewSource
      =
((System.Windows.Data.CollectionViewSource)(this.FindResource("getIndividualListVi
ewSource")));
                getIndividualListViewSource.View.MoveCurrentToFirst();
             }
            private void MenuItem_Click(object sender, RoutedEventArgs e)
```

```
            {
                var form = new AddClient();
                 form.ShowDialog();
                UpdateAll();
            }
            void UpdateAll()
            {
                Login.bankDataSet bankDataSet =
((Login.bankDataSet)(this.FindResource("bankDataSet")));
                Login.bankDataSetTableAdapters.GetStatementCreditsTableAdapter
bankDataSetGetStatementCreditsTableAdapter = new
Login.bankDataSetTableAdapters.GetStatementCreditsTableAdapter();
                 bankDataSetGetStatementCreditsTableAdapt-
      er.Fill(bankDataSet.GetStatementCredits);
                System.Windows.Data.CollectionViewSource
getStatementCreditsViewSource = ((Sys-
      tem.Windows.Data.CollectionViewSource)(this.FindResource("getStatementCredit
sVie wSource")));
                getStatementCreditsViewSource.View.MoveCurrentToFirst();

Login.bankDataSetTableAdapters.GetStatementMortgagesTableAdapter
bankDataSetGetStatementMortgagesTableAdapter = new
Login.bankDataSetTableAdapters.GetStatementMortgagesTableAdapter();
bankDataSetGetStatementMortgagesTableAdapter.Fill(bankDataSet.GetStatementMortgage
s);
                System.Windows.Data.CollectionViewSource
getStatementMortgagesViewSource = ((Sys-
      tem.Windows.Data.CollectionViewSource)(this.FindResource("getStatementMortga
gesV iewSource")));
                getStatementMortgagesViewSource.View.MoveCurrentToFirst(); .
                Login.bankDataSetTableAdapters.GetStatementHoldingsTableAdapter
bankDataSetGetStatementHoldingsTableAdapter = new
Login.bankDataSetTableAdapters.GetStatementHoldingsTableAdapter();
bankDataSetGetStatementHoldingsTableAdapter.Fill(bankDataSet.GetStatementHoldings)
;
                System.Windows.Data.CollectionViewSource
getStatementHoldingsViewSource = ((Sys-
      tem.Windows.Data.CollectionViewSource)(this.FindResource("getStatementHoldin
gsVi ewSource")));
                getStatementHoldingsViewSource.View.MoveCurrentToFirst();
Login.bankDataSetTableAdapters.GetStatementTranslationsTableAdapter
      bankDataSetGetStatementTranslationsTableAdapter = new
Login.bankDataSetTableAdapters.GetStatementTranslationsTableAdapter();
                 bankDataSetGetStatementTranslationsTableAdapt-
      er.Fill(bankDataSet.GetStatementTranslations);
                System.Windows.Data.CollectionViewSource
getStatementTranslationsViewSource = ((Sys-
      tem.Windows.Data.CollectionViewSource)(this.FindResource("getStatementTransl
atio nsViewSource")));
                getStatementTranslationsViewSource.View.MoveCurrentToFirst();
             }
            private void biAddper_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {
                var form = new AddPerevod(ID, stat);
                form.ShowDialog();
                UpdateAll();
```

```
            }
            private void biDeleteper_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {             try             {
                    if (MessageBox.Show("Ви впевнені, що хочете видалити
запис?", "Банківський сервіс ", MessageBoxButton.YesNo,
MessageBoxImage.Information) == MessageBoxResult.Yes)
                    {
                        int id =
      (int)((DataRowView)getStatementTranslationsDataGrid.SelectedItems[0]).Row["T
rans lation_ID"];
                        using (bankEntities connect = new bankEntities())
                        {
                            connect.DeletePerevod(id);
                        }                 }
            }             catch
                {
            }
                UpdateAll();
            }
            private void biAddVk_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {
                var form = new AddHolding(ID, stat);
                 form.ShowDialog();
                UpdateAll();
            }
              private void biDeleteVk_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {
                  try
                 {
                    if (MessageBox.Show("Ви впевнені, що хочете видалити
запис?", "Банківський сервіс ", MessageBoxButton.YesNo,
MessageBoxImage.Information) == MessageBoxResult.Yes)
                    {
                        int id =
      (int)((DataRowView)getStatementHoldingsDataGrid.SelectedItems[0]).Row["Holdi
ng_I D"];
                        using (bankEntities connect = new bankEntities())
                        {
                            connect.DeleteHoldings(id);
                        }                 }
                }             catch
                {
            }
                UpdateAll();
            }
            private void biDeleteIP_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {             try             {
                    if (MessageBox.Show("Ви впевнені, що хочете видалити
запис?", "Банківський сервіс ", MessageBoxButton.YesNo,
MessageBoxImage.Information) == MessageBoxResult.Yes)
                    {
```

```
                        int id =
(int)((DataRowView)getStatementMortgagesDataGrid.SelectedItems[0]).Row["Mortgage
      _ID"];
                        using (bankEntities connect = new bankEntities())
                        {
                            connect.DeleteMortgages(id);
                        }                   }
            }             catch
                {
                }
                UpdateAll();
            }
            private void biDeleteCR_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {              try              {
                if (MessageBox.Show("Ви впевнені, що хочете видалити
запис?", "Банківський сервіс ", MessageBoxButton.YesNo,
MessageBoxImage.Information) == MessageBoxResult.Yes)
                {
                    int id =
(int)((DataRowView)getStatementCreditsDataGrid.SelectedItems[0]).Row["Credit_ID"
      ];
                    using (bankEntities connect = new bankEntities())
                    {
                        connect.DeleteCredits(id);
                    }
                }
                }             catch
                {
                }
                UpdateAll();
            }
            void free()
            {
                using (bankEntities ctx = new bankEntities())
                {
                    var update = ctx.Users.Where(u => u.User_ID == ID).Update(u
=> new User { StatusUser = 1 });
                }
            }
            private void biAddIP_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {
                var form = new AddMortgage(ID, stat);
                form.ShowDialog();
                UpdateAll();
            }
            private void biAddCR_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {
                var form = new AddCredit(ID, stat);
                form.ShowDialog();
                UpdateAll();
            }
        }
    }
```

А.4 Лістинг ConcreteConsultant.cs

```
using System;
using System.Collections.Generic; using System.Linq; using System.Text;
using System.Threading.Tasks;
 namespace Банковый_консультант
{     class ConcreteConsultant:Consultant
    {
        private int time =10 ;
        public override void ConsultantRequest(int resurce)
         {
            if (Presence)
            {
                Data.app.Add(number);
                 switch (resurce)
                {
                     case 1:
                         if (Credit && !employment)
                        {
                            employment = true;
                        }
                        else next(resurce);
                         break;
                     case 2:
                         if (Vklad && !employment)
                        {
                            employment = true;
                        }
                        else next(resurce);
                         break;
                     case 3:
                         if (Perevod && !employment)
                        {
                            employment = true;
                        }
                        else next(resurce);
                         break;
                }
                if (employment)
                {                       time--;
                }
                if (time == 0)
                {
                    employment = false;
                }             }
            else next(resurce);


         }
        private void next(int resurce)
        {
            if (Successor != null)
                Successor.ConsultantRequest(resurce);
             else
                Data.Value = resurce;
         }
```

А.5 Лістинг Client.cs

```
namespace WpfApplication1
{
    using System;
    using System.Collections.Generic;
   public partial class Client
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
         public Client()
        {
            this.Translations = new HashSet<Translation>();
             this.Holdings = new HashSet<Holding>();
             this.Mortgages = new HashSet<Mortgage>();
             this.Credits = new HashSet<Credit>();
        }
        public int Client_ID { get; set; }
         public string FullName { get; set; }
         public string HomeAddress { get; set; }
         public string Postcode { get; set; }
         public string INN { get; set; }
         public string Account { get; set; }
        public virtual Entity Entity { get; set; }
         public virtual Individual Individual { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Translation> Translations { get; set; }
[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Holding> Holdings { get; set; }
[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Mortgage> Mortgages { get; set; }
[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Credit> Credits { get; set; }
    }
}
```

А.6 Лістинг Admin.cs

```
namespace Login
{
    public partial class Admin : Window
    {         int id;
        public Admin(int ID)
        {
            InitializeComponent();             id = ID;
        }
          private void Window_Loaded(object sender, RoutedEventArgs e)
         {
            Login.bankDataSet bankDataSet =
((Login.bankDataSet)(this.FindResource("bankDataSet")));
            Login.bankDataSetTableAdapters.UsersTableAdapter
bankDataSetUsersTableAdapter = new
```

```
Login.bankDataSetTableAdapters.UsersTableAdapter();
bankDataSetUsersTableAdapter.Fill(bankDataSet.Users);
                    System.Windows.Data.CollectionViewSource usersViewSource =
((System.Windows.Data.CollectionViewSource)(this.FindResource("usersViewSource")))
;
                     usersViewSource.View.MoveCurrentToFirst();
                     min.SelectedDate = new DateTime(2012, 01, 01);
                     max.SelectedDate = new DateTime(2017, 01, 01);
                }
                private void Window_Closed(object sender, EventArgs e)
                {
                    using (bankEntities ctx = new bankEntities())
                    {
                        var update = ctx.Users.Where(u => u.User_ID == id).Update(u
=> new User { StatusUser = 0 });
                    }
                    this.Close();
                }
                private void fullNameComboBox_SelectionChanged_1(object sender,
SelectionChangedEventArgs e)
                {
                    int user =
      (int)((DataRowView)fullNameComboBox.SelectedItem).Row["User_ID"];
                    Login.bankDataSet bankDataSet =
((Login.bankDataSet)(this.FindResource("bankDataSet")));
                    Login.bankDataSetTableAdapters.AvgTimeCreditTableAdapter
bankDataSetAvgTimeCreditTableAdapter = new
Login.bankDataSetTableAdapters.AvgTimeCreditTableAdapter();
bankDataSetAvgTimeCreditTableAdapter.Fill(bankDataSet.AvgTimeCredit,user);
            System.Windows.Data.CollectionViewSource avgTimeCreditViewSource =
((System.Windows.Data.CollectionViewSource)(this.FindResource("avgTimeCreditViewSo
urc e")));
                    avgTimeCreditViewSource.View.MoveCurrentToFirst();
                    Login.bankDataSetTableAdapters.AvgTimeHoldingTableAdapter
bankDataSetAvgTimeHoldingTableAdapter = new
Login.bankDataSetTableAdapters.AvgTimeHoldingTableAdapter();
bankDataSetAvgTimeHoldingTableAdapter.Fill(bankDataSet.AvgTimeHolding,user);
                    System.Windows.Data.CollectionViewSource
avgTimeHoldingViewSource =
      ((System.Windows.Data.CollectionViewSource)(this.FindResource("avgTimeHoldin
gViewSource")));
                    avgTimeHoldingViewSource.View.MoveCurrentToFirst();
                     Login.bankDataSetTableAdapters.AvgTimeMortgageTableAdapter
bankDataSetAvgTimeMortgageTableAdapter = new
Login.bankDataSetTableAdapters.AvgTimeMortgageTableAdapter();
      bankDataSetAvgTimeMortgageTableAdapter.Fill(bankDataSet.AvgTimeMortgage,user
);
                    System.Windows.Data.CollectionViewSource
avgTimeMortgageViewSource =
      ((System.Windows.Data.CollectionViewSource)(this.FindResource("avgTimeMortga
geViewSource")));
                    avgTimeMortgageViewSource.View.MoveCurrentToFirst();
                     using (bankEntities connect = new bankEntities())
                    {
                        DevExpress.Xpf.Charts.SeriesPoint point = new
DevExpress.Xpf.Charts.SeriesPoint();
                        point.Argument = "Кредитування";
```

```
                    foreach (int a in connect.StaticCredit(user))
                    {
                        point.Value = a;
                    }
                    gist.Points.Add(point);
                     point = new DevExpress.Xpf.Charts.SeriesPoint();
                    foreach (int a in connect.StaticMortgage(user))
                    {
                        point.Value = a;
                    }
                    gist.Points.Add(point);
                    point = new DevExpress.Xpf.Charts.SeriesPoint();
                     point.Argument = "Вклади";
                    foreach (int a in connect.StaticHoldings(user))
                    {
                        point.Value = a;
                    }
                    gist.Points.Add(point);
                point = new DevExpress.Xpf.Charts.SeriesPoint();
                point.Argument = "Перекази";
                    foreach (int a in connect.StaticTrans(user))
                    {
                        point.Value = a;
                    }
                    gist.Points.Add(point);
                 }
            }
            private void min_SelectedDateChanged(object sender, Selection-
      ChangedEventArgs e)         {
                 using (bankEntities connect = new bankEntities())
                 {
                    DevExpress.Xpf.Charts.SeriesPoint point = new
DevExpress.Xpf.Charts.SeriesPoint();
                    point.Argument = "Кредитування";
                     foreach (int a in
connect.ValueCredits(min.SelectedDate,max.SelectedDate))
                    {
                        point.Value = a;
                    }
                    gist1.Points.Add(point);
                point = new DevExpress.Xpf.Charts.SeriesPoint();
                point.Argument =;
                foreach (int a in
connect.ValueMortgages(min.SelectedDate,max.SelectedDate))
                    {
                        point.Value = a;
                    }
                    gist1.Points.Add(point);
                    point = new DevExpress.Xpf.Charts.SeriesPoint();
                     point.Argument = "Вклади";
                    foreach (int a in connect.ValueHoldings(min.SelectedDate,
max.SelectedDate))
                     {
                        point.Value = a;
                    }
                    gist1.Points.Add(point);
                point = new DevExpress.Xpf.Charts.SeriesPoint();
```

```
                point.Argument = "Перекази";
               foreach (int a in connect.ValueTrans(min.SelectedDate,
max.SelectedDate))
                {
                        point.Value = a;
                    }
                    gist1.Points.Add(point);
                }
            }
        }}
```

А.7 Лістинг AddPerevod.cs

```
      using System;
      using System.Collections.Generic;
      using System.Linq; using System.Text; using System.Windows; using
System.Windows.Controls; using System.Windows.Data; using
System.Windows.Documents; using System.Windows.Input; using System.Windows.Media;
using System.Windows.Media.Imaging; using System.Windows.Shapes; using
System.Data; namespace Login
      {
          public partial class AddPerevod : Window
          {         int user;         int stat;
              public AddPerevod(int User,int Stat)
              {
                  InitializeComponent();
                   user = User;
                   stat = Stat;
              }
              private void Window_Loaded(object sender, RoutedEventArgs e)
{
             Login.bankDataSet bankDataSet =
((Login.bankDataSet)(this.FindResource("bankDataSet")));
                  Login.bankDataSetTableAdapters.ClientsTableAdapter
bankDataSetClientsTableAdapter = new
Login.bankDataSetTableAdapters.ClientsTableAdapter();
                   bankDataSetClientsTableAdapter.Fill(bankDataSet.Clients);
                   System.Windows.Data.CollectionViewSource clientsViewSource =
((System.Windows.Data.CollectionViewSource)(this.FindResource("clientsViewSource")
));             clientsViewSource.View.MoveCurrentToFirst();
               }
              private void simpleButton1_Click(object sender, RoutedEventArgs e)
              {
                  this.Close();
              }         int client;
              private void simpleButton2_Click(object sender, RoutedEventArgs e)
              {
                  DateTime date = DateTime.Now;


                  using (bankEntities connect = new bankEntities())
                  {
                      connect.AddPerevod(user, stat, client,
Convert.ToInt32(textBox1.Text), textBox.Text);
                  }
                      this.Close();
              }
```

```
        private void simpleButton_Click(object sender, RoutedEventArgs e)
              {
                  var form = new AddClient();
                   form.ShowDialog();
              }
              private void clientsDataGrid_SelectionChanged(object sender,
Selection-
      ChangedEventArgs e)
              {             try
                   {
                      client =
(int)((DataRowView)clientsDataGrid.SelectedItems[0]).Row["Client_ID"];
                   }
                   catch { }
              }
          }
      }
```

A.8 Лістинг AddHoldings.cs

```
      using System;
      using System.Collections.Generic; using System.Linq; using System.Text;
using System.Windows; using System.Windows.Controls; using System.Windows.Data;
using System.Windows.Documents; using System.Windows.Input; using
System.Windows.Media; using System.Windows.Media.Imaging; using
System.Windows.Shapes; using System.Data;
      namespace Login
      {
          public partial class AddHolding : Window
          {          int user;          int stat;
              public AddHolding(int User, int Stat)
              {
                  InitializeComponent();
                   user = User;
                   stat = Stat;
              }
                private void Window_Loaded(object sender, RoutedEventArgs e)
               {
                    update();
              }
              void update()
              {
                  Login.bankDataSet bankDataSet =
((Login.bankDataSet)(this.FindResource("bankDataSet")));
                   Login.bankDataSetTableAdapters.ClientsTableAdapter
bankDataSetClientsTableAdapter = new
Login.bankDataSetTableAdapters.ClientsTableAdapter();
                   bankDataSetClientsTableAdapter.Fill(bankDataSet.Clients);
                   System.Windows.Data.CollectionViewSource clientsViewSource =
((System.Windows.Data.CollectionViewSource)(this.FindResource("clientsViewSource")
));
                   clientsViewSource.View.MoveCurrentToFirst();
              }
              private void simpleButton_Click(object sender, RoutedEventArgs e)
              {
                  var form = new AddClient();
```

```
                form.ShowDialog();
                update();
            }
            private void simpleButton1_Click(object sender, RoutedEventArgs e)
            {
                DateTime date = DateTime.Now;
                int size = Convert.ToInt32(textBox.Text);
                int proc = Convert.ToInt32(textBox1.Text);
                using (bankEntities connect = new bankEntities())
                {
                    connect.AddHoldings(user,stat,client,size,date,proc);
                }
                this.Close();
            }         int client;
            private void clientsDataGrid_SelectionChanged(object sender,
SelectionChangedEventArgs e)
            {             try
                {
                    client =
(int)((DataRowView)clientsDataGrid.SelectedItems[0]).Row["Client_ID"];
                }
                catch { }
            }
            private void simpleButton2_Click(object sender, RoutedEventArgs e)
            {
                this.Close();
            }
        }
    }
```

A.9 Лістинг AddMortgage.cs

```
    namespace Login
    {     public partial class AddMortgage : Window
        {         int user;         int stat;
            public AddMortgage(int User, int Stat)
            {
                InitializeComponent();
                user = User;
                stat = Stat;
            }
            private void Window_Loaded(object sender, RoutedEventArgs e)
{
                update();
            }
            void update()
            {
                Login.bankDataSet bankDataSet =
((Login.bankDataSet)(this.FindResource("bankDataSet")));
                Login.bankDataSetTableAdapters.ClientsTableAdapter
bankDataSetClientsTableAdapter = new
Login.bankDataSetTableAdapters.ClientsTableAdapter();
                bankDataSetClientsTableAdapter.Fill(bankDataSet.Clients);
                System.Windows.Data.CollectionViewSource clientsViewSource =
((System.Windows.Data.CollectionViewSource)(this.FindResource("clientsViewSource")
));
```

```
                clientsViewSource.View.MoveCurrentToFirst();
            }
            private void simpleButton_Click(object sender, RoutedEventArgs e)
            {
                var form = new AddClient();
                form.ShowDialog();
                update();
            }
            private void simpleButton1_Click(object sender, RoutedEventArgs e)
            {
                DateTime date = DateTime.Now;
                int size = Convert.ToInt32(textBox.Text);
                int proc = Convert.ToInt32(textBox1.Text);
                using (bankEntities connect = new bankEntities())
                {
                    connect.AddMortgages(user,stat,client,size,date,proc);
                }
                this.Close();
            }        int client;
            private void clientsDataGrid_SelectionChanged(object sender,
Selection-
      ChangedEventArgs e)
            {
                try
                {
                    client =
(int)((DataRowView)clientsDataGrid.SelectedItems[0]).Row["Client_ID"];
                }            catch { }
            }
              private void simpleButton2_Click(object sender, RoutedEventArgs e)
            {
                this.Close();
            }
        } }
```

A.10 Лістинг AddCredit.cs

```
      namespace Login
      {     public partial class AddCredit : Window
          {         int user;         int stat;
              public AddCredit(int User, int Stat)
              {
                  InitializeComponent();
                  user = User;
                  stat = Stat;
              }
         private void Window_Loaded(object sender, RoutedEventArgs e)
         {
                  update();
              }
              void update()
              {
                  Login.bankDataSet bankDataSet =
((Login.bankDataSet)(this.FindResource("bankDataSet")));
```

```
                Login.bankDataSetTableAdapters.ClientsTableAdapter
bankDataSetClientsTableAdapter = new
Login.bankDataSetTableAdapters.ClientsTableAdapter();
                 bankDataSetClientsTableAdapter.Fill(bankDataSet.Clients);
                 System.Windows.Data.CollectionViewSource clientsViewSource =
((System.Windows.Data.CollectionViewSource)(this.FindResource("clientsViewSource")
));             clientsViewSource.View.MoveCurrentToFirst();
            }
            private void simpleButton_Click(object sender, RoutedEventArgs e)
            {
                var form = new AddClient();
                 form.ShowDialog();
                 update();
            }
            private void simpleButton1_Click(object sender, RoutedEventArgs e)
            {
                DateTime date = DateTime.Now;
                 int size = Convert.ToInt32(textBox.Text);
                 int proc = Convert.ToInt32(textBox1.Text);
                 using (bankEntities connect = new bankEntities())
                {
                    connect.AddCredit(user,stat,client,size,date,proc);
                }
                this.Close();
            }         int client;
            private void clientsDataGrid_SelectionChanged(object sender,
Selection-
      ChangedEventArgs e)
            {             try             {
                    client =
      (int)((DataRowView)clientsDataGrid.SelectedItems[0]).Row["Client_ID"];
                }             catch { }
            }
            private void simpleButton2_Click(object sender, RoutedEventArgs e)
            {
                this.Close();
            }
        }
      }
```

A.11Лістинг AddClient.cs

```
      namespace Login
      {
          public partial class AddClient : Window
          {
              public AddClient()
              {
                  InitializeComponent();
              }
              private void simpleButton_Click(object sender, RoutedEventArgs e)
              {
                  this.Close();
              }
```

```
            private void simpleButton1_Click(object sender, RoutedEventArgs e)
            {
                DateTime? d1 = dr.SelectedDate;
                DateTime? d2 = dv.SelectedDate;
                using (bankEntities connect = new bankEntities())
                {
                    connect.AddIndividual(textBox.Text, textBox1.Text, text-
      Box2.Text, textBox3.Text, textBox4.Text, d1, textBox5.Text, textBox6.Text,
text-
      Box7.Text, d2);
                 }
                this.Close();
            }
            private void simpleButton3_Click(object sender, RoutedEventArgs e)
             {
                DateTime? d2 = dv1.SelectedDate;
                using (bankEntities connect = new bankEntities())
                {
                    connect.AddEntity(textBox8.Text, textBox9.Text,
textBox10.Text, textBox11.Text, textBox12.Text,textBox13.Text,d2);
                }
                this.Close();
            }
        }
      }
```