**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І**

**ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Кафедра комп’ютерних наук

**КУРСОВИЙ ПРОЄКТ**

з дисципліни «Організація баз даних» на тему:

**«Автоматизація робочого процесу документообігу банківських працівників»**

Виконав:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Юхименко Єгор Ігорович, ІПЗ-22007б

підпис ПІБ, група

**Робота захищена:**

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_\_ р.

Оцінка за національною шкалою: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали) (підпис) (прізвище та ініціали)

КИЇВ – 2024

ЗМІСТ

[ВСТУП 3](#_Toc170471598)

[1. ПОСТАНОВКА ЗАДАЧІ 5](#_Toc170471599)

[2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 7](#_Toc170471600)

[2.1. Логічна модель даних у вигляді ЕR-діаграми 7](#_Toc170471601)

[2.2. Створення бази даних 9](#_Toc170471602)

[3. ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 14](#_Toc170471603)

[3.1. Організаційна структура програмного забезпечення 14](#_Toc170471604)

[3.2. Вибір інструментарію для створення прикладного програмного забезпечення 16](#_Toc170471605)

[3.3. Алгоритмізація та програмування програмних модулів 17](#_Toc170471606)

[4. РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ 23](#_Toc170471607)

[4.1. Тестування системи 23](#_Toc170471608)

[4.2. Вимоги до апаратного та програмного забезпечення 28](#_Toc170471609)

[ВИСНОВКИ 30](#_Toc170471610)

[СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ 32](#_Toc170471611)

[ДОДАТКИ 34](#_Toc170471612)

# **ВСТУП**

В умовах сучасного світу, де інформація є важливим ресурсом, автоматизація робочих процесів стає необхідною умовою ефективної роботи будь-якої організації. Особливо це стосується банківських установ, де документообіг є однією з основних складових операційної діяльності. Величезні обсяги інформації, які щодня обробляються банківськими працівниками, вимагають впровадження ефективних систем автоматизації, що дозволяють зменшити час на виконання рутинних операцій, підвищити точність та надійність обробки даних, а також забезпечити безпеку інформації.

Метою курсового проекту є розробка системи автоматизації робочого процесу документообігу для банківських працівників. Така система покликана спростити та прискорити процес обробки документів, зменшити кількість помилок при введенні даних, а також забезпечити зручне зберігання та пошук інформації.

Виконання проекту має велике значення для банківської сфери, оскільки забезпечує ряд переваг, таких як підвищення продуктивності праці, зниження операційних витрат та покращення якості обслуговування клієнтів. Автоматизація процесів дозволяє банківським працівникам зосередитись на виконанні більш складних та творчих завдань, що вимагають аналітичного підходу та професійних знань, замість витрачання часу на рутинні операції з обробки документів.

Актуальність теми дослідження обумовлена необхідністю постійного вдосконалення технологій управління інформаційними потоками в умовах стрімкого розвитку банківської індустрії. Впровадження сучасних програмних рішень для автоматизації документообігу є важливим кроком на шляху до створення ефективних та конкурентоспроможних банківських структур.

У курсовому проекті буде проведено аналіз існуючих методів автоматизації документообігу, розроблено логічну модель даних у вигляді ER-діаграми, створено базу даних, а також розроблено прикладне програмне забезпечення для автоматизації робочого процесу документообігу банківських працівників. На завершення будуть надані рекомендації щодо впровадження та експлуатації розробленої системи.

Таким чином, результати даного дослідження сприятимуть підвищенню ефективності банківських установ, зменшенню навантаження на працівників та покращенню якості обслуговування клієнтів, що в кінцевому підсумку позитивно вплине на розвиток банківської галузі в цілому.

# **ПОСТАНОВКА ЗАДАЧІ**

Банківська установа є складною організаційною структурою, де щодня обробляється великий обсяг документів, пов'язаних із фінансовими операціями, клієнтськими запитами, внутрішнім та зовнішнім листуванням тощо. Ефективне управління документообігом є важливим для забезпечення безперебійної роботи банку, дотримання нормативних вимог і підвищення якості обслуговування клієнтів.

У системі автоматизації документообігу використовуються такі основні типи даних:

- Клієнтська інформація: особисті дані клієнтів, інформація про банківські рахунки, історія операцій.

- Фінансові операції: деталі транзакцій, переказів, платежів, депозити та кредити.

- Внутрішня документація: службові записки, внутрішні звіти, накази та розпорядження.

- Зовнішнє листування: кореспонденція з іншими банками, організаціями та клієнтами.

- Регуляторна інформація: звіти для контролюючих органів, нормативні документи.

Основні операції, що виконуються в системі автоматизації документообігу:

- Введення даних: створення нових записів про клієнтів, фінансові операції, внутрішні та зовнішні документи.

- Корегування даних: редагування існуючих записів для актуалізації інформації.

- Пошук даних: швидкий пошук необхідних документів та записів за різними критеріями (номер документа, дата, тип документа, клієнт).

- Видалення даних: видалення застарілих або помилкових записів з бази даних.

- Генерація звітів: створення звітів для внутрішнього користування та для зовнішніх органів контролю.

Документи для введення, корегування та пошуку даних:

- Форми введення клієнтських даних: анкети нових клієнтів, договори на обслуговування.

- Форми для фінансових операцій: заяви на переказ коштів, платіжні доручення, заявки на кредити.

- Внутрішні форми: службові записки, розпорядження керівництва.

- Зовнішні форми: листи до інших банків, звіти для регуляторів.

Зразки звітів:

1. Звіт про фінансові операції:

- Період: січень 2024 р.

- Тип операцій: перекази, депозити, кредити.

- Загальна кількість операцій: 500

- Загальна сума операцій: 10 000 000 грн.

2. Звіт про клієнтську активність:

- Період: січень 2024 р.

- Кількість нових клієнтів: 150

- Кількість активних клієнтів: 1200

- Кількість закритих рахунків: 50

3. Внутрішній звіт про виконання службових завдань:

- Період: січень 2024 р.

- Кількість виконаних завдань: 200

- Кількість невиконаних завдань: 20

- Кількість запитів на додаткову інформацію: 30

Таким чином, даний проект спрямований на створення ефективної системи автоматизації робочого процесу документообігу, яка забезпечить підвищення продуктивності праці, зменшення операційних витрат та покращення якості обслуговування клієнтів банку.

**2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ**

## 2.1. Логічна модель даних у вигляді ЕR-діаграми

Для розробки системи автоматизації робочого процесу документообігу банківських працівників обрана СУБД Microsoft SQL Server. Це реляційна система управління базами даних, розроблена компанією Microsoft. Вона забезпечує надійність, масштабованість та високу продуктивність для корпоративних додатків. SQL Server підтримує транзакції, процедури, тригери та інші механізми для забезпечення цілісності та безпеки даних.

SQL (Structured Query Language) — це стандартна мова для управління та маніпуляцій даними у реляційних базах даних. SQL використовується для створення, модифікації та видалення баз даних та таблиць, а також для виконання запитів до бази даних. Основні компоненти SQL включають:

- DDL (Data Definition Language): визначення структури даних (створення, зміна та видалення таблиць).

- DML (Data Manipulation Language): маніпуляції даними (вставка, оновлення, видалення даних).

- DCL (Data Control Language): управління доступом до даних (надання та відкликання привілеїв).

ER-діаграма (Entity-Relationship Diagram) відображає логічну структуру бази даних та взаємозв'язки між її сутностями. Нижче представлена ER-діаграма для нашої системи автоматизації робочого процесу документообігу банківських працівників.

ER-діаграма для нашої бази даних зображена на рисунку 2.1.

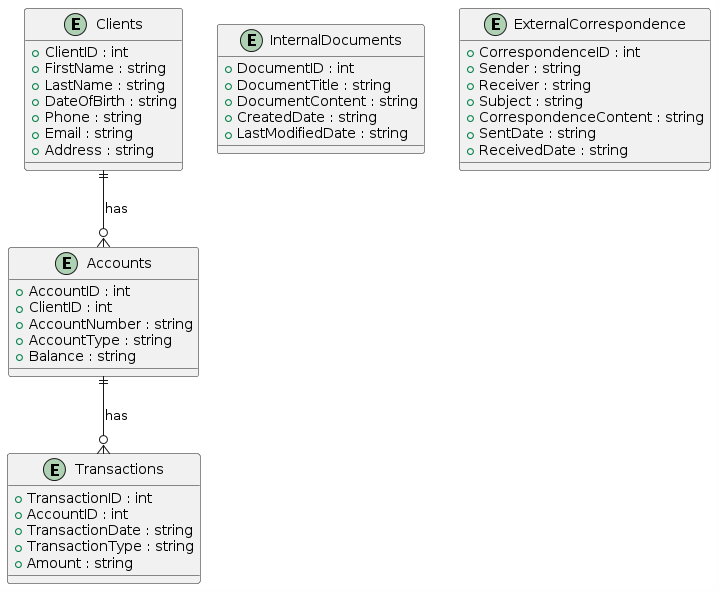


Рис. 2.1 - ER-діаграма для нашої бази даних

ER-діаграма відповідає третій нормальній формі (3NF), оскільки:

1. Всі атрибути таблиць мають атомарні значення.

2. Всі неосновні атрибути залежні від первинного ключа.

3. Всі транзитивні залежності усунуті, тобто неосновні атрибути залежать тільки від первинного ключа.

Ця логічна модель забезпечує цілісність та ефективність зберігання даних у базі даних, мінімізуючи надлишковість та можливі аномалії при оновленні даних.

## 2.2. Створення бази даних

Для створення бази даних та відповідних об'єктів, таких як таблиці, уявлення, індекси, збережені процедури, тригери та користувачі, наведемо SQL-запити, які реалізують створення цих об'єктів. Нижче наведено основні SQL-запити для створення таблиць та інших об'єктів бази даних.

Лістинг коду 2.1 - Створення бази даних

CREATE DATABASE BankDocumentWorkflow;

GO

USE BankDocumentWorkflow;

GO

CREATE TABLE Clients (

ClientID INT PRIMARY KEY IDENTITY(1,1),

FirstName NVARCHAR(50),

LastName NVARCHAR(50),

DateOfBirth DATE,

Phone NVARCHAR(20),

Email NVARCHAR(50),

Address NVARCHAR(100)

);

GO

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY IDENTITY(1,1),

ClientID INT,

AccountNumber NVARCHAR(20),

AccountType NVARCHAR(20),

Balance DECIMAL(18, 2),

FOREIGN KEY (ClientID) REFERENCES Clients(ClientID)

);

CREATE TABLE Transactions (

TransactionID INT PRIMARY KEY IDENTITY(1,1),

AccountID INT,

TransactionDate DATETIME,

TransactionType NVARCHAR(20),

Amount DECIMAL(18, 2),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE InternalDocuments (

DocumentID INT PRIMARY KEY IDENTITY(1,1),

DocumentTitle NVARCHAR(100),

DocumentContent NVARCHAR(MAX),

CreatedDate DATETIME,

LastModifiedDate DATETIME

);

CREATE TABLE ExternalCorrespondence (

CorrespondenceID INT PRIMARY KEY IDENTITY(1,1),

Sender NVARCHAR(50),

Receiver NVARCHAR(50),

Subject NVARCHAR(100),

CorrespondenceContent NVARCHAR(MAX),

SentDate DATETIME,

ReceivedDate DATETIME

);

Кінець лістингу 2.1

Лістинг коду 2.2 - Створення уявлень

CREATE VIEW ClientAccounts AS

SELECT

c.ClientID,

c.FirstName,

c.LastName,

a.AccountID,

a.AccountNumber,

a.AccountType,

a.Balance

FROM

Clients c

JOIN

Accounts a ON c.ClientID = a.ClientID;

GO

Кінець лістингу 2.2

Лістинг коду 2.3 - Створення індексів

CREATE INDEX IX\_AccountNumber ON Accounts (AccountNumber);

GO

Кінець лістингу 2.3

Лістинг коду 2.5 - Створення збережених процедур

CREATE PROCEDURE AddClient

@FirstName NVARCHAR(50),

@LastName NVARCHAR(50),

@DateOfBirth DATE,

@Phone NVARCHAR(20),

@Email NVARCHAR(50),

@Address NVARCHAR(100)

AS

BEGIN

INSERT INTO Clients (FirstName, LastName, DateOfBirth, Phone, Email, Address)

VALUES (@FirstName, @LastName, @DateOfBirth, @Phone, @Email, @Address);

END;

GO

Кінець лістингу 2.4

Лістинг коду 2.5 - Створення тригерів

CREATE TRIGGER UpdateBalance

ON Transactions

AFTER INSERT

AS

BEGIN

UPDATE Accounts

SET Balance = Balance + inserted.Amount

FROM Accounts

INNER JOIN inserted ON Accounts.AccountID = inserted.AccountID

WHERE inserted.TransactionType = 'Deposit';

UPDATE Accounts

SET Balance = Balance - inserted.Amount

FROM Accounts

INNER JOIN inserted ON Accounts.AccountID = inserted.AccountID

WHERE inserted.TransactionType = 'Withdrawal';

END;

GO

Кінець лістингу 2.5

Лістинг коду 2.6 - Створення користувачів

CREATE LOGIN BankUser WITH PASSWORD = 'SecurePassword123';

GO

CREATE USER BankUser FOR LOGIN BankUser;

GO

ALTER ROLE db\_datareader ADD MEMBER BankUser;

ALTER ROLE db\_datawriter ADD MEMBER BankUser;

GO

Кінець лістингу 2.6

Для створення діаграми бази даних можна використовувати графічний інструмент, такий як SQL Server Management Studio (SSMS), або інші інструменти для побудови ER-діаграм. Наведемо приклад діаграми для наших таблиць (рис. 2.2).

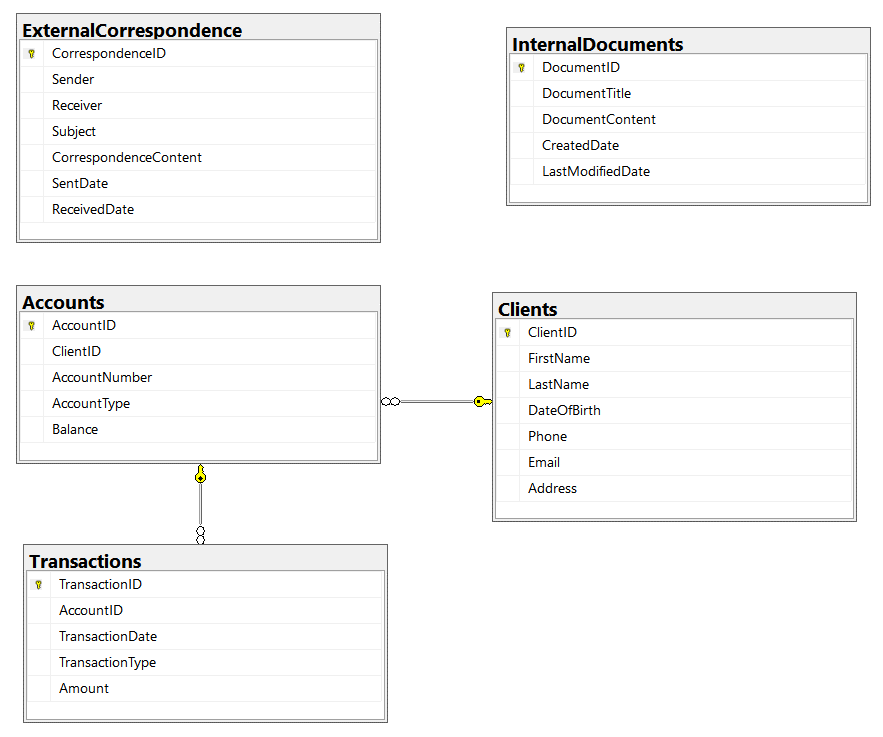


Рис. 2.2 – Діаграма, створена в SSMS

Ці SQL-запити забезпечують створення основних об'єктів бази даних, включаючи таблиці, уявлення, індекси, збережені процедури, тригери та користувачів, необхідних для функціонування системи автоматизації робочого процесу документообігу банківських працівників.

# **3. ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ**

## 3.1. Організаційна структура програмного забезпечення

Організаційна структура програмного забезпечення (ПЗ) для автоматизації робочого процесу документообігу банківських працівників повинна забезпечувати модульність, масштабованість, зручність в обслуговуванні та розширюваність. Нижче представлена структура ПЗ, яка складається з кількох основних модулів та компонентів (табл. 3.1).

Таблиця 3.1 - Основні компоненти програмного забезпечення

|  |  |  |
| --- | --- | --- |
| Компонент | Підкомпонент | Опис |
| **Клієнтська частина (Client-side application)** | Інтерфейс користувача (UI) | |
| FormClients | Форма для управління клієнтами |
| FormAccounts | Форма для управління рахунками |
| FormTransactions | Форма для управління транзакціями |
| FormInternalDocuments | Форма для управління внутрішніми документами |
| FormExternalCorrespondence | Форма для управління зовнішнім листуванням |
| Компоненти інтерфейсу | |
| TextBox, DataGridView, Button тощо | Компоненти для введення та відображення даних |
| Логіка управління формами | Методи для додавання, редагування та видалення записів |
| **Серверна частина (Server-side application)** | Бізнес-логіка | |
| Client | Клас для обробки даних клієнтів |
| Account | Клас для обробки даних рахунків |
| Transaction | Клас для обробки даних транзакцій |
| InternalDocument | Клас для обробки даних внутрішніх документів |
| ExternalCorrespondence | Клас для обробки даних зовнішнього листування |
| Operations | Класи для виконання бізнес-операцій |
| Доступ до даних (Data Access Layer) | |
| Database, SqlConnection, SqlCommand | Класи для взаємодії з базою даних |
| Методи для виконання SQL-запитів | Додавання, оновлення, видалення даних |
| **База даних (Database)** | Таблиці (Tables) | |
| Clients | Таблиця для зберігання даних про клієнтів |
| Accounts | Таблиця для зберігання даних про рахунки |
| Transactions | Таблиця для зберігання даних про транзакції |
| InternalDocuments | Таблиця для зберігання внутрішніх документів |
| ExternalCorrespondence | Таблиця для зберігання зовнішнього листування |
| Уявлення (Views) | |
| ClientAccounts | Уявлення для відображення клієнтів та їх рахунків |
| Індекси (Indexes) | |
| Індекс на таблицю Accounts | Підвищення швидкості пошуку даних у таблиці рахунків |
| Збережені процедури (Stored Procedures) | |
| AddClient | Процедура для додавання нового клієнта |
| Тригери (Triggers) | |
| UpdateBalance | Тригер для оновлення балансу рахунків після транзакцій |
| Користувачі (Users) | |
| BankUser | Банківський користувач з відповідними правами доступу |

Діаграма організаційної структури зображено на рисунку 3.1.

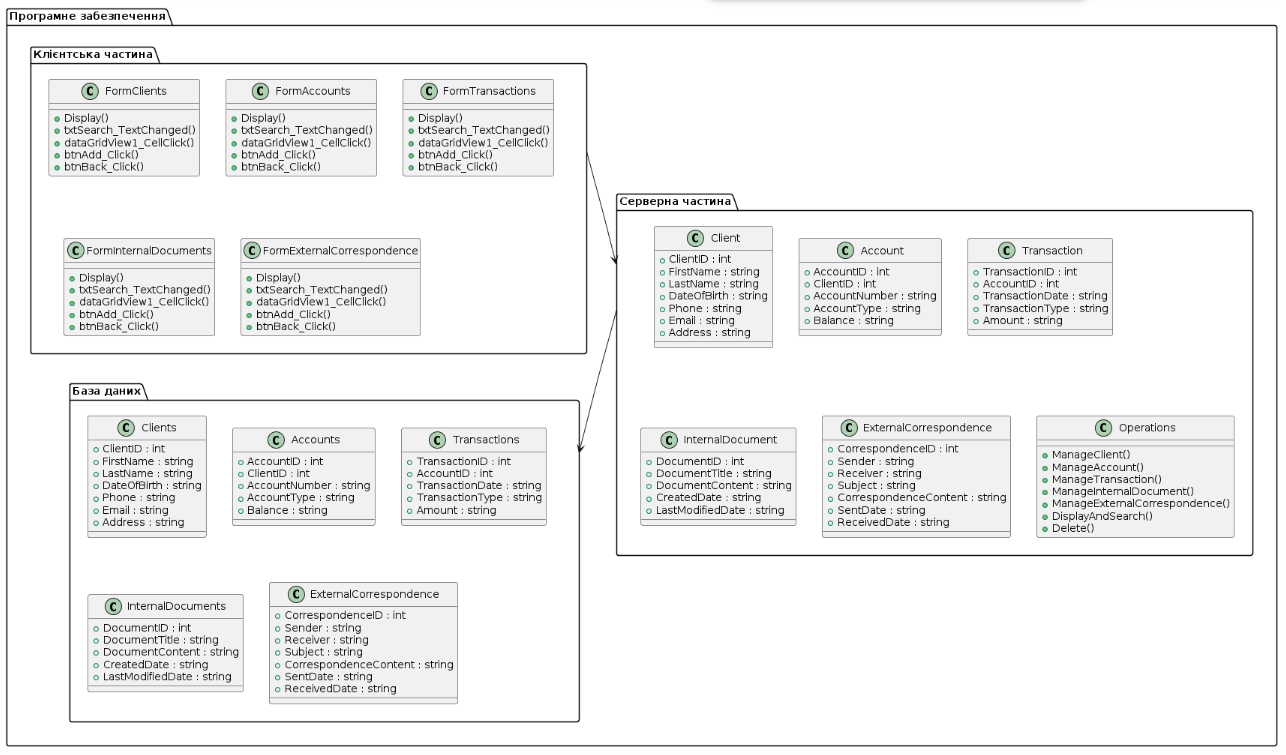


Рис. 3.1 - Діаграма організаційної структури

Ця організаційна структура дозволяє легко підтримувати та розширювати функціональність програмного забезпечення, забезпечує чітке розмежування обов'язків між клієнтською та серверною частинами, а також забезпечує безпеку та зручність доступу до даних.

## 3.2. Вибір інструментарію для створення прикладного програмного забезпечення

Вибір інструментарію для розробки системи автоматизації робочого процесу документообігу банківських працівників є важливим для забезпечення ефективності, надійності та зручності використання програмного забезпечення. Розглянемо обрані технології та інструменти, які використовуватимуться у цьому проекті, та обґрунтуємо їх вибір.

Для реалізації клієнтської та серверної частини програмного забезпечення було обрано мову програмування C#. C# є сучасною об'єктно-орієнтованою мовою, розробленою компанією Microsoft, яка забезпечує високу продуктивність і зручність для розробників. C# має потужний набір бібліотек і фреймворків, що дозволяють створювати різноманітні типи додатків, від настільних до веб-додатків. Мова підтримує об'єктно-орієнтоване програмування, що сприяє модульності, повторному використанню коду та полегшує підтримку і розширення системи [1].

Розробка програмного забезпечення здійснюватиметься у середовищі Visual Studio. Visual Studio є інтегрованим середовищем розробки (IDE), яке пропонує потужні інструменти для написання, відлагодження та тестування коду. Visual Studio забезпечує зручний інтерфейс для розробників, підтримку багатьох мов програмування, інтеграцію з системами контролю версій, такими як Git, та інші функціональні можливості, які значно підвищують продуктивність роботи. Середовище також надає інструменти для створення графічного інтерфейсу користувача, що дозволяє швидко та легко розробляти форми та інші компоненти інтерфейсу.

Для управління базою даних було обрано Microsoft SQL Server. Це потужна реляційна система управління базами даних, яка забезпечує високу продуктивність, надійність та безпеку зберігання даних. SQL Server підтримує складні запити, транзакції, збережені процедури, тригери та інші механізми для забезпечення цілісності даних і виконання бізнес-логіки на стороні сервера. Крім того, SQL Server добре інтегрується з іншими продуктами Microsoft, що спрощує розробку та адміністрування бази даних [2].

Для адміністрування бази даних та виконання запитів до неї використовується SQL Server Management Studio (SSMS). SSMS є потужним інструментом, який забезпечує зручний інтерфейс для адміністраторів баз даних та розробників. Він дозволяє виконувати SQL-запити, керувати базами даних, створювати та змінювати таблиці, уявлення, індекси, збережені процедури та тригери, а також здійснювати моніторинг продуктивності бази даних. SSMS забезпечує всі необхідні інструменти для ефективного управління базою даних, що є важливим для підтримки надійності та продуктивності системи.

Таким чином, вибір C# як основної мови програмування, Visual Studio як середовища розробки, SQL Server як системи управління базами даних та SSMS як інструменту для адміністрування бази даних є оптимальним для розробки та підтримки системи автоматизації робочого процесу документообігу банківських працівників. Ці інструменти забезпечують високу продуктивність, зручність у використанні та інтеграцію між собою, що сприяє ефективній розробці, тестуванню та підтримці програмного забезпечення.

## 3.3. Алгоритмізація та програмування програмних модулів

У процесі розробки системи автоматизації робочого процесу документообігу банківських працівників важливо чітко визначити алгоритми, що будуть використовуватися для обробки даних, взаємодії між компонентами системи та забезпечення основних функцій. Розглянемо основні алгоритми та процеси, які необхідно реалізувати у програмних модулях системи.

Основні програмні модулі та їх функції:

1. Модуль керування клієнтами

- Додавання нового клієнта

- Редагування інформації про клієнта

- Видалення клієнта

- Пошук клієнтів

2. Модуль керування рахунками

- Додавання нового рахунку

- Редагування інформації про рахунок

- Видалення рахунку

- Пошук рахунків

3. Модуль керування транзакціями

- Додавання нової транзакції

- Редагування інформації про транзакцію

- Видалення транзакції

- Пошук транзакцій

4. Модуль керування внутрішніми документами

- Додавання нового документа

- Редагування документа

- Видалення документа

- Пошук документів

5. Модуль керування зовнішнім листуванням

- Додавання нового листа

- Редагування листа

- Видалення листа

- Пошук листів

Також опишемо деякі основні алгоритми для основних функцій.

Алгоритм додавання нового клієнта:

1. Відкрити форму для введення даних клієнта.

2. Ввести дані (FirstName, LastName, DateOfBirth, Phone, Email, Address).

3. Натиснути кнопку "Зберегти".

4. Перевірити введені дані на коректність.

5. Сформувати SQL-запит для вставки даних у таблицю `Clients`.

6. Виконати SQL-запит.

7. Оновити відображення списку клієнтів.

Лістинг коду 3.1 - Додавання нового клієнта

public void AddClient(Client client)

{

string sql = "INSERT INTO Clients (FirstName, LastName, DateOfBirth, Phone, Email, Address) VALUES (@FirstName, @LastName, @DateOfBirth, @Phone, @Email, @Address)";

using (SqlConnection conn = GetConnection())

{

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@FirstName", client.FirstName);

cmd.Parameters.AddWithValue("@LastName", client.LastName);

cmd.Parameters.AddWithValue("@DateOfBirth", client.DateOfBirth);

cmd.Parameters.AddWithValue("@Phone", client.Phone);

cmd.Parameters.AddWithValue("@Email", client.Email);

cmd.Parameters.AddWithValue("@Address", client.Address);

conn.Open();

cmd.ExecuteNonQuery();

}

DisplayClients();

}

Кінець лістингу 3.1

Редагування інформації про клієнта. Алгоритм:

1. Вибрати клієнта зі списку.

2. Відкрити форму для редагування даних клієнта.

3. Внести зміни в дані.

4. Натиснути кнопку "Оновити".

5. Перевірити введені дані на коректність.

6. Сформувати SQL-запит для оновлення даних у таблиці `Clients`.

7. Виконати SQL-запит.

8. Оновити відображення списку клієнтів.

Лістинг коду 3.2 - Редагування інформації про клієнта

public void UpdateClient(Client client)

{

string sql = "UPDATE Clients SET FirstName = @FirstName, LastName = @LastName, DateOfBirth = @DateOfBirth, Phone = @Phone, Email = @Email, Address = @Address WHERE ClientID = @ClientID";

using (SqlConnection conn = GetConnection())

{

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@ClientID", client.ClientID);

cmd.Parameters.AddWithValue("@FirstName", client.FirstName);

cmd.Parameters.AddWithValue("@LastName", client.LastName);

cmd.Parameters.AddWithValue("@DateOfBirth", client.DateOfBirth);

cmd.Parameters.AddWithValue("@Phone", client.Phone);

cmd.Parameters.AddWithValue("@Email", client.Email);

cmd.Parameters.AddWithValue("@Address", client.Address);

conn.Open();

cmd.ExecuteNonQuery();

}

DisplayClients();

}

Кінець лістингу 3.2

Видалення клієнта. Алгоритм:

1. Вибрати клієнта зі списку.

2. Натиснути кнопку "Видалити".

3. Підтвердити видалення.

4. Сформувати SQL-запит для видалення даних з таблиці `Clients`.

5. Виконати SQL-запит.

6. Оновити відображення списку клієнтів.

Код:

Лістинг коду 3.3 - Видалення клієнта

public void DeleteClient(int clientId)

{

string sql = "DELETE FROM Clients WHERE ClientID = @ClientID";

using (SqlConnection conn = GetConnection())

{

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@ClientID", clientId);

conn.Open();

cmd.ExecuteNonQuery();

}

DisplayClients();

}

Кінець лістингу 3.3

Пошук клієнтів. Алгоритм:

1. Ввести пошуковий запит у текстове поле.

2. Сформувати SQL-запит для пошуку клієнтів за критеріями.

3. Виконати SQL-запит.

4. Відобразити результати пошуку у списку клієнтів.

Лістинг коду 3.1 - Пошук клієнтів

public void SearchClients(string query)

{

string sql = "SELECT FROM Clients WHERE FirstName LIKE @Query OR LastName LIKE @Query";

using (SqlConnection conn = GetConnection())

{

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@Query", "%" + query + "%");

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

List<Client> clients = new List<Client>();

while (reader.Read())

{

clients.Add(new Client

{

ClientID = reader.GetInt32(0),

FirstName = reader.GetString(1),

LastName = reader.GetString(2),

DateOfBirth = reader.GetDateTime(3).ToString("yyyy-MM-dd"),

Phone = reader.GetString(4),

Email = reader.GetString(5),

Address = reader.GetString(6)

});

}

dataGridViewClients.DataSource = clients;

}

}

Кінець лістингу 3.4

Аналогічні алгоритми можуть бути реалізовані для інших модулів системи. Загалом, алгоритмізація та програмування програмних модулів системи автоматизації робочого процесу документообігу банківських працівників включає створення алгоритмів для додавання, редагування, видалення та пошуку даних для кожної сутності. Реалізація цих алгоритмів на мові програмування C# забезпечує зручність та ефективність управління даними, а також інтеграцію з базою даних SQL Server для забезпечення надійного зберігання та доступу до даних.

# **4. РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ**

## 4.1. Тестування системи

Тестування є критичним етапом у розробці програмного забезпечення, оскільки воно дозволяє виявити та виправити помилки до того, як система буде впроваджена у виробниче середовище. Тестування системи автоматизації робочого процесу документообігу банківських працівників охоплює декілька аспектів, включаючи функціональне тестування, тестування продуктивності, тестування безпеки та тестування користувацького інтерфейсу.

Функціональне тестування перевіряє, чи правильно система виконує свої основні функції. Для нашої системи це означає перевірку коректності роботи модулів керування клієнтами, рахунками, транзакціями, внутрішніми документами та зовнішнім листуванням.

1. Тестування додавання клієнта

- Вхідні дані: коректні дані клієнта (ім'я, прізвище, дата народження, телефон, email, адреса).

- Очікуваний результат: клієнт додається до бази даних, новий запис відображається у списку клієнтів.

Лістинг коду 4.1 - Тестування додавання клієнта

[TestMethod]

public void TestAddClient()

{

Client client = new Client("Іван", "Петров", "1990-05-15", "050-123-4567", "ivan.petrov@example.com", "вул. Шевченка, 1, Київ");

bool result = Operations.AddClient(client);

Assert.IsTrue(result);

var addedClient = Operations.GetClientByName("Іван", "Петров");

Assert.IsNotNull(addedClient);

Assert.AreEqual(client.FirstName, addedClient.FirstName);

Assert.AreEqual(client.LastName, addedClient.LastName);

}

2. Тестування редагування клієнта

- Вхідні дані: Існуючий клієнт, змінені дані (наприклад, новий телефон).

- Очікуваний результат: Дані клієнта оновлюються в базі даних, зміни відображаються у списку клієнтів.

Лістинг коду 4.2 - Тестування редагування клієнта

[TestMethod]

public void TestUpdateClient()

{

Client client = Operations.GetClientByName("Іван", "Петров");

client.Phone = "050-987-6543";

bool result = Operations.UpdateClient(client);

Assert.IsTrue(result);

var updatedClient = Operations.GetClientByName("Іван", "Петров");

Assert.AreEqual("050-987-6543", updatedClient.Phone);

}

Кінець лістингу 4.2

3. Тестування видалення клієнта

- Вхідні дані: існуючий клієнт.

- Очікуваний результат: клієнт видаляється з бази даних, запис зникає зі списку клієнтів.

Лістинг коду 4.3 - Тестування видалення клієнта

[TestMethod]

public void TestDeleteClient()

{

Client client = Operations.GetClientByName("Іван", "Петров");

bool result = Operations.DeleteClient(client.ClientID);

Assert.IsTrue(result);

var deletedClient = Operations.GetClientByName("Іван", "Петров");

Assert.IsNull(deletedClient);

}

Кінець лістингу 4.3

4. Тестування пошуку клієнта

- Вхідні дані: пошуковий запит (ім'я або прізвище клієнта).

- Очікуваний результат: відображаються клієнти, які відповідають критеріям пошуку.

Лістинг коду 4.4 - Тестування пошуку клієнта

[TestMethod]

public void TestSearchClient()

{

var clients = Operations.SearchClients("Іван");

Assert.IsTrue(clients.Count > 0);

foreach (var client in clients)

{

Assert.IsTrue(client.FirstName.Contains("Іван") || client.LastName.Contains("Іван"));

}

}

Кінець лістингу 4.4

Тестування продуктивності перевіряє, як система поводиться під навантаженням. Це включає в себе перевірку часу відповіді системи при виконанні запитів, швидкість обробки даних та здатність обробляти велику кількість одночасних запитів.

1. Стрес-тестування додавання клієнтів

- Вхідні дані: велика кількість запитів на додавання клієнтів.

- Очікуваний результат: система справляється з навантаженням без значних затримок або помилок.

Лістинг коду 4.5 - Тестування продуктивності

[TestMethod]

public void TestAddClientsUnderLoad()

{

for (int i = 0; i < 1000; i++)

{

Client client = new Client($"Іван{i}", $"Петров{i}", "1990-05-15", $"050-123-{4567 + i}", $"ivan{i}.petrov@example.com", "вул. Шевченка, 1, Київ");

bool result = Operations.AddClient(client);

Assert.IsTrue(result);

}

}

Кінець лістингу 4.5

Тестування безпеки перевіряє, чи захищена система від несанкціонованого доступу та інших загроз. Це включає в себе перевірку механізмів аутентифікації та авторизації, захист від SQL-ін'єкцій та інші аспекти безпеки.

1. Тестування аутентифікації користувачів

- Вхідні дані: коректні та некоректні облікові дані користувача.

- Очікуваний результат: система дозволяє доступ лише для коректних облікових даних.

Лістинг коду 4.6 - Тестування безпеки

[TestMethod]

public void TestUserAuthentication()

{

bool result = Operations.AuthenticateUser("validUser", "validPassword");

Assert.IsTrue(result);

result = Operations.AuthenticateUser("invalidUser", "invalidPassword");

Assert.IsFalse(result);

}

Кінець лістингу 4.6

Тестування користувацького інтерфейсу перевіряє зручність використання та коректність відображення даних у формі. Це включає в себе перевірку правильності відображення даних, зручність навігації та відповідність інтерфейсу вимогам користувачів.

1. Тестування відображення списку клієнтів

- Вхідні дані: дані про клієнтів у базі даних.

- Очікуваний результат: список клієнтів коректно відображається у формі, включаючи всі необхідні поля.

Лістинг коду 4.7 - Тестування відображення списку клієнтів

[TestMethod]

public void TestDisplayClients()

{

FormClients form = new FormClients();

form.Display();

Assert.IsTrue(form.dataGridViewClients.Rows.Count > 0);

}

Кінець лістингу 4.7

Загалом, тестування системи автоматизації робочого процесу документообігу банківських працівників є важливим етапом, який дозволяє виявити та виправити помилки, забезпечити надійність та безпеку системи. Воно включає функціональне тестування, тестування продуктивності, тестування безпеки та тестування користувацького інтерфейсу. Виконання цих тестів гарантує, що система буде ефективною та зручною у використанні, відповідаючи всім вимогам користувачів.

## 4.2. Вимоги до апаратного та програмного забезпечення

Для ефективного функціонування системи автоматизації робочого процесу документообігу банківських працівників необхідно забезпечити відповідні апаратні ресурси як для серверної, так і для клієнтської частини.

Серверна частина:

- Процесор: чотирьохядерний процесор з тактовою частотою не менше 2.5 ГГц.

- Оперативна пам'ять: мінімум 16 ГБ оперативної пам'яті.

- Жорсткий диск: SSD накопичувач ємністю не менше 500 ГБ.

- Мережеве підключення: гігабітне Ethernet-з'єднання.

- Резервне копіювання: система резервного копіювання для захисту даних.

Клієнтська частина:

- Процесор: двоядерний процесор з тактовою частотою не менше 2.0 ГГц.

- Оперативна пам'ять: мінімум 4 ГБ оперативної пам'яті.

- Жорсткий диск: HDD або SSD накопичувач ємністю не менше 100 ГБ.

- Мережеве підключення: стабільне інтернет-з'єднання.

Для забезпечення роботи програмного забезпечення потрібні наступні програмні компоненти.

Серверна частина:

- Операційна система: Windows Server 2016 або новіша версія.

- База даних: Microsoft SQL Server 2017 або новіша версія.

- Сервер додатків: .NET Core або .NET Framework 4.7.2 і новіші.

- Інструменти для резервного копіювання: встановлене програмне забезпечення для резервного копіювання баз даних і файлів.

Клієнтська частина:

- Операційна система: Windows 10 або новіша версія.

- Розробка і запуск додатків: .NET Framework 4.7.2 або новіша версія.

- Програмне забезпечення для розробки: Visual Studio 2019 або новіша версія.

- Браузер: сучасний веб-браузер для доступу до веб-компонентів системи.

Додаткові вимоги:

- Безпека: антивірусне програмне забезпечення та мережевий брандмауер на клієнтських і серверних машинах.

- Мережа: VPN для безпечного віддаленого доступу до серверних ресурсів.

- Розширюваність: можливість масштабування апаратних ресурсів сервера для обробки збільшених обсягів даних та користувачів.

Виконання наведених вимог до апаратного та програмного забезпечення забезпечить стабільну і ефективну роботу системи автоматизації робочого процесу документообігу банківських працівників. Система буде здатна обробляти великі обсяги даних, підтримувати багатокористувацький режим роботи та забезпечувати високу продуктивність і безпеку даних.

# **ВИСНОВКИ**

У результаті виконання курсової роботи на тему "Автоматизація робочого процесу документообігу банківських працівників" було досягнуто значних результатів. Спершу була розроблена логічна модель даних у вигляді ER-діаграми, яка відповідає третій нормальній формі, що дозволило створити структуру бази даних, включаючи таблиці для зберігання даних про клієнтів, рахунки, транзакції, внутрішні документи та зовнішнє листування. Крім того, були створені уявлення, індекси, збережені процедури та тригери, що забезпечують ефективний доступ до даних та їх обробку.

У процесі розробки програмного забезпечення була створена клієнтська частина системи з графічним інтерфейсом користувача, яка включає форми для управління клієнтами, рахунками, транзакціями, внутрішніми документами та зовнішнім листуванням. Також була реалізована серверна частина з бізнес-логікою та доступом до даних через класи для взаємодії з базою даних. Було забезпечено управління ролями та правами доступу, а також механізми аутентифікації та авторизації користувачів.

Тестування системи включало функціональне тестування основних модулів, що охоплювало додавання, редагування, видалення та пошук даних. Тестування продуктивності показало здатність системи обробляти велику кількість одночасних запитів, а тестування безпеки перевірило аутентифікацію користувачів та захист від SQL-ін'єкцій.

Вимоги до апаратного та програмного забезпечення були визначені для ефективного функціонування системи. Для серверної та клієнтської частин було забезпечено відповідні ресурси, включаючи операційні системи, бази даних, інструменти розробки та безпеки.

Загалом, розроблена система автоматизації робочого процесу документообігу банківських працівників відповідає поставленим завданням та демонструє високу техніко-економічну ефективність. Автоматизація рутинних операцій підвищує продуктивність праці, зменшує ризик помилок при введенні та обробці даних, покращує обслуговування клієнтів та забезпечує надійність зберігання та обробки інформації.

Отримані результати повністю відповідають завданню на курсовий проект. Усі визначені вимоги були виконані, включаючи розробку логічної моделі даних, створення бази даних, розробку програмного забезпечення, тестування системи та визначення вимог до апаратного та програмного забезпечення. Таким чином, розроблена система відповідає сучасним вимогам та є ефективним інструментом для автоматизації робочого процесу документообігу банківських працівників.

# **СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ**

1. Albahari J., Albahari B. C# 9.0 in a Nutshell: The Definitive Reference. O'Reilly Media, 2021. 1088 с.
2. Assaf W., West R., et al. SQL Server 2019 Administration Inside Out. Microsoft Press, 2020. 832 с.
3. Atkinson P., Vieira R. Beginning Microsoft SQL Server 2012 Programming. Wrox, 2012. 864 с.
4. C# and Windows Forms. URL: https://metanit.com/sharp/windowsforms/1.1.php
5. Delaney K., et al. Microsoft SQL Server Internals. Microsoft Press, 2012. 992 с.
6. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017. 616 с.
7. Microsoft Visual Studio. URL: https://visualstudio.microsoft.com/uk/
8. Моделювання баз даних. URL: https://book24.ua/bazi-danih-kontseptualne-modelyuvannya-karl-kppers-5081315/
9. Petkovic D. SQL Server 2019: A Beginner's Guide. McGraw-Hill Education, 2019. 800 с.
10. Sells C. Windows Forms Programming in C#. Addison-Wesley Professional, 2003. 848 с.
11. Skeet J. C# in Depth. Manning Publications, 2019. 528 с.
12. Stack Overflow. URL: https://stackoverflow.com/questions/
13. Troelsen A., Japikse P. Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming. Apress, 2021. 1376 с.
14. W3Schools tutorials. URL: <https://www.w3schools.com/>
15. Hejlsberg A., Torgersen M., Wiltamuth S., Golde, P. The C# programming language. Pearson Education, 2008. С. 784.
16. Liberty J. Programming C#: building. NET applications with C. " O'Reilly Media, Inc, 2005. С. 672
17. Sharp J. Microsoft Visual C# 2013 Step by Step. Pearson Education, 2013. 82 с.
18. Troelsen A., Olsen A. Pro C# 5.0 and the. NET 4.5 Framework (Vol. 6). New York: Apress, 2012. 554 с.
19. Albahari J. C# 10 in a Nutshell : O'Reilly Media, Inc., 2022. 1060 с.
20. Sells C., Weinhardt M. Windows forms 2.0 programming : Pearson Education, 2006. 1296 с.
21. Griffiths I., Adams M. NET Windows forms in a nutshell : O'Reilly Media, Inc., 2003. 879 с.
22. Freeman A. Windows Forms. In Introducing Visual C# 2010 (pp. 1033-1067). Berkeley, CA: Apress, 2010. С. 10.
23. Mamone M. Creating Windows Forms. Practical Mono, 2006. С. 133-178.
24. Learning C#. URL: <https://dotnet.microsoft.com/en-us/learn/csharp>
25. C# concepts. URL: <https://learn.microsoft.com/uk-ua/dotnet/csharp/programming-guide/concepts/>
26. Mamone M. Using ADO. NET. Practical Mono, 2006. С. 205-239.
27. Troelsen A. C# and the. NET Platform : Apress, 2001. 1008 с.
28. Troelsen A., Japikse P. C# 6.0 and the. NET 4.6 Framework : Apress, 2015 1625 с.

# **ДОДАТКИ**

**Додаток А – Програмний код**

using bank;

namespace WinFormsApp3

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

// To customize application configuration such as set high DPI settings or default font,

// see https://aka.ms/applicationconfiguration.

ApplicationConfiguration.Initialize();

Application.Run(new Main());

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace bank

{

internal class Account

{

public string ClientID { get; set; }

public string AccountNumber { get; set; }

public string AccountType { get; set; }

public string Balance { get; set; }

public Account(string clientID, string accountNumber, string accountType, string balance)

{

ClientID = clientID;

AccountNumber = accountNumber;

AccountType = accountType;

Balance = balance;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace bank

{

internal class Client

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string DateOfBirth { get; set; }

public string Phone { get; set; }

public string Email { get; set; }

public string Address { get; set; }

public Client(string firstName, string lastName, string dateOfBirth, string phone, string email, string address)

{

FirstName = firstName;

LastName = lastName;

DateOfBirth = dateOfBirth;

Phone = phone;

Email = email;

Address = address;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace bank

{

internal class ExternalCorrespondence

{

public string Sender { get; set; }

public string Receiver { get; set; }

public string Subject { get; set; }

public string CorrespondenceContent { get; set; }

public string SentDate { get; set; }

public string ReceivedDate { get; set; }

public ExternalCorrespondence(string sender, string receiver, string subject, string correspondenceContent, string sentDate, string receivedDate)

{

Sender = sender;

Receiver = receiver;

Subject = subject;

CorrespondenceContent = correspondenceContent;

SentDate = sentDate;

ReceivedDate = receivedDate;

}

}

}

using bank;

using Microsoft.Data.SqlClient;

using System;

using System.Collections.Generic;

using System.Data;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace bank

{

internal class Operations

{

public static SqlConnection GetConnection()

{

string sql = "Data Source=localhost\\MSSQLSERVER03;Initial Catalog=BankDocumentWorkflow;Integrated Security=True;TrustServerCertificate=true";

SqlConnection conn = new SqlConnection(sql);

try

{

conn.Open();

}

catch (SqlException ex)

{

MessageBox.Show("SQL з'єднання! \n" + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

return conn;

}

public static void ManageClient(Client c, string sql, string? id = null)

{

SqlConnection conn = GetConnection();

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.CommandType = CommandType.Text;

if (id != null)

{

cmd.Parameters.Add("@ClientID", SqlDbType.VarChar).Value = id;

}

cmd.Parameters.Add("@FirstName", SqlDbType.VarChar).Value = c.FirstName;

cmd.Parameters.Add("@LastName", SqlDbType.VarChar).Value = c.LastName;

cmd.Parameters.Add("@DateOfBirth", SqlDbType.VarChar).Value = c.DateOfBirth;

cmd.Parameters.Add("@Phone", SqlDbType.VarChar).Value = c.Phone;

cmd.Parameters.Add("@Email", SqlDbType.VarChar).Value = c.Email;

cmd.Parameters.Add("@Address", SqlDbType.VarChar).Value = c.Address;

try

{

cmd.ExecuteNonQuery();

}

catch (SqlException ex)

{

MessageBox.Show("Помилка SQL-з'єднання! \n" + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

conn.Close();

}

}

public static void ManageAccount(Account a, string sql, string? id = null)

{

SqlConnection conn = GetConnection();

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.CommandType = CommandType.Text;

if (id != null)

{

cmd.Parameters.Add("@AccountID", SqlDbType.VarChar).Value = id;

}

cmd.Parameters.Add("@ClientID", SqlDbType.VarChar).Value = a.ClientID;

cmd.Parameters.Add("@AccountNumber", SqlDbType.VarChar).Value = a.AccountNumber;

cmd.Parameters.Add("@AccountType", SqlDbType.VarChar).Value = a.AccountType;

cmd.Parameters.Add("@Balance", SqlDbType.VarChar).Value = a.Balance;

try

{

cmd.ExecuteNonQuery();

}

catch (SqlException ex)

{

MessageBox.Show("Помилка SQL-з'єднання! \n" + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

conn.Close();

}

}

public static void ManageTransaction(Transaction t, string sql, string? id = null)

{

SqlConnection conn = GetConnection();

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.CommandType = CommandType.Text;

if (id != null)

{

cmd.Parameters.Add("@TransactionID", SqlDbType.VarChar).Value = id;

}

cmd.Parameters.Add("@AccountID", SqlDbType.VarChar).Value = t.AccountID;

cmd.Parameters.Add("@TransactionDate", SqlDbType.VarChar).Value = t.TransactionDate;

cmd.Parameters.Add("@TransactionType", SqlDbType.VarChar).Value = t.TransactionType;

cmd.Parameters.Add("@Amount", SqlDbType.VarChar).Value = t.Amount;

try

{

cmd.ExecuteNonQuery();

}

catch (SqlException ex)

{

MessageBox.Show("Помилка SQL-з'єднання! \n" + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

conn.Close();

}

}

public static void ManageInternalDocument(InternalDocument d, string sql, string? id = null)

{

SqlConnection conn = GetConnection();

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.CommandType = CommandType.Text;

if (id != null)

{

cmd.Parameters.Add("@DocumentID", SqlDbType.VarChar).Value = id;

}

cmd.Parameters.Add("@DocumentTitle", SqlDbType.VarChar).Value = d.DocumentTitle;

cmd.Parameters.Add("@DocumentContent", SqlDbType.VarChar).Value = d.DocumentContent;

cmd.Parameters.Add("@CreatedDate", SqlDbType.VarChar).Value = d.CreatedDate;

cmd.Parameters.Add("@LastModifiedDate", SqlDbType.VarChar).Value = d.LastModifiedDate;

try

{

cmd.ExecuteNonQuery();

}

catch (SqlException ex)

{

MessageBox.Show("Помилка SQL-з'єднання! \n" + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

conn.Close();

}

}

public static void ManageExternalCorrespondence(ExternalCorrespondence e, string sql, string? id = null)

{

SqlConnection conn = GetConnection();

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.CommandType = CommandType.Text;

if (id != null)

{

cmd.Parameters.Add("@CorrespondenceID", SqlDbType.VarChar).Value = id;

}

cmd.Parameters.Add("@Sender", SqlDbType.VarChar).Value = e.Sender;

cmd.Parameters.Add("@Receiver", SqlDbType.VarChar).Value = e.Receiver;

cmd.Parameters.Add("@Subject", SqlDbType.VarChar).Value = e.Subject;

cmd.Parameters.Add("@CorrespondenceContent", SqlDbType.VarChar).Value = e.CorrespondenceContent;

cmd.Parameters.Add("@SentDate", SqlDbType.VarChar).Value = e.SentDate;

cmd.Parameters.Add("@ReceivedDate", SqlDbType.VarChar).Value = e.ReceivedDate;

try

{

cmd.ExecuteNonQuery();

}

catch (SqlException ex)

{

MessageBox.Show("Помилка SQL-з'єднання! \n" + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

conn.Close();

}

}

public static void Delete(string? id, string? cl, string? table)

{

string sql = $"DELETE FROM {table} WHERE {cl} = @Id";

SqlConnection conn = GetConnection();

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.CommandType = System.Data.CommandType.Text;

cmd.Parameters.Add("@Id", SqlDbType.VarChar).Value = id;

try

{

cmd.ExecuteNonQuery();

}

catch (SqlException ex)

{

MessageBox.Show("Не видалено. \n" + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

conn.Close();

}

public static void DisplayAndSearch(string query, DataGridView dgv)

{

string sql = query;

SqlConnection conn = GetConnection();

SqlCommand cmd = new SqlCommand(sql, conn);

SqlDataAdapter adp = new SqlDataAdapter(cmd);

DataTable tbl= new DataTable();

adp.Fill(tbl);

dgv.DataSource = tbl;

conn.Close();

}

public static void Clear(Control parent)

{

foreach (Control c in parent.Controls)

{

if (c is TextBox)

{

((TextBox)c).Text = string.Empty;

}

else if (c.HasChildren)

{

Clear(c);

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace bank

{

internal class InternalDocument

{

public string DocumentTitle { get; set; }

public string DocumentContent { get; set; }

public string CreatedDate { get; set; }

public string LastModifiedDate { get; set; }

public InternalDocument(string documentTitle, string documentContent, string createdDate, string lastModifiedDate)

{

DocumentTitle = documentTitle;

DocumentContent = documentContent;

CreatedDate = createdDate;

LastModifiedDate = lastModifiedDate;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace bank

{

internal class Transaction

{

public string AccountID { get; set; }

public string TransactionDate { get; set; }

public string TransactionType { get; set; }

public string Amount { get; set; }

public Transaction(string accountID, string transactionDate, string transactionType, string amount)

{

AccountID = accountID;

TransactionDate = transactionDate;

TransactionType = transactionType;

Amount = amount;

}

}

}

using bank;

using library;

using System;

using System.Windows.Forms;

namespace bank

{

public partial class FormAccounts : Form

{

private readonly Main \_parent;

FormNewAccount accountForm;

public string sql = "SELECT AccountID, ClientID, AccountNumber, AccountType, Balance FROM Accounts;";

public FormAccounts(Main parent)

{

InitializeComponent();

\_parent = parent;

accountForm = new FormNewAccount(this);

}

public void Display()

{

Operations.DisplayAndSearch(sql, dataGridView1);

}

private void FormAccounts\_Load(object sender, EventArgs e)

{

Display();

}

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

string searchSql = "SELECT AccountID, ClientID, AccountNumber, AccountType, Balance FROM Accounts " +

"WHERE AccountNumber LIKE '%" + txtSearch.Text + "%';";

Operations.DisplayAndSearch(searchSql, dataGridView1);

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 0)

{

accountForm.id = dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString();

accountForm.clientID = dataGridView1.Rows[e.RowIndex].Cells[3].Value.ToString();

accountForm.accountNumber = dataGridView1.Rows[e.RowIndex].Cells[4].Value.ToString();

accountForm.accountType = dataGridView1.Rows[e.RowIndex].Cells[5].Value.ToString();

accountForm.balance = dataGridView1.Rows[e.RowIndex].Cells[6].Value.ToString();

accountForm.UpdateInfo();

accountForm.ShowDialog();

return;

}

if (e.ColumnIndex == 1)

{

if (MessageBox.Show("Дійсно видалити?", "Інформація", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

Operations.Delete(dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString(), "AccountID", "Accounts");

Display();

}

return;

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

accountForm.DefaultInfo();

accountForm.ShowDialog();

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using bank;

using System;

using System.Windows.Forms;

namespace library

{

public partial class FormClients : Form

{

private readonly Main \_parent;

FormNewClient clientForm;

public string sql = "SELECT ClientID, FirstName, LastName, DateOfBirth, Phone, Email, Address FROM Clients;";

public FormClients(Main parent)

{

InitializeComponent();

\_parent = parent;

clientForm = new FormNewClient(this);

}

public void Display()

{

Operations.DisplayAndSearch(sql, dataGridView1);

}

private void FormClients\_Load(object sender, EventArgs e)

{

Display();

}

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

string searchSql = "SELECT ClientID, FirstName, LastName, DateOfBirth, Phone, Email, Address FROM Clients " +

"WHERE FirstName LIKE '%" + txtSearch.Text + "%' OR LastName LIKE '%" + txtSearch.Text + "%';";

Operations.DisplayAndSearch(searchSql, dataGridView1);

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 0)

{

clientForm.id = dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString();

clientForm.firstName = dataGridView1.Rows[e.RowIndex].Cells[3].Value.ToString();

clientForm.lastName = dataGridView1.Rows[e.RowIndex].Cells[4].Value.ToString();

clientForm.dateOfBirth = dataGridView1.Rows[e.RowIndex].Cells[5].Value.ToString();

clientForm.phone = dataGridView1.Rows[e.RowIndex].Cells[6].Value.ToString();

clientForm.email = dataGridView1.Rows[e.RowIndex].Cells[7].Value.ToString();

clientForm.address = dataGridView1.Rows[e.RowIndex].Cells[8].Value.ToString();

clientForm.UpdateInfo();

clientForm.ShowDialog();

return;

}

if (e.ColumnIndex == 1)

{

if (MessageBox.Show("Дійсно видалити?", "Інформація", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

Operations.Delete(dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString(), "ClientID", "Clients");

Display();

}

return;

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

clientForm.DefaultInfo();

clientForm.ShowDialog();

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using bank;

using System;

using System.Windows.Forms;

namespace bank

{

public partial class FormExternalCorrespondence : Form

{

private readonly Main \_parent;

FormNewExternalCorrespondence correspondenceForm;

public string sql = "SELECT CorrespondenceID, Sender, Receiver, Subject, CorrespondenceContent, SentDate, ReceivedDate FROM ExternalCorrespondence;";

public FormExternalCorrespondence(Main parent)

{

InitializeComponent();

\_parent = parent;

correspondenceForm = new FormNewExternalCorrespondence(this);

}

public void Display()

{

Operations.DisplayAndSearch(sql, dataGridView1);

}

private void FormExternalCorrespondence\_Load(object sender, EventArgs e)

{

Display();

}

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

string searchSql = "SELECT CorrespondenceID, Sender, Receiver, Subject, CorrespondenceContent, SentDate, ReceivedDate FROM ExternalCorrespondence " +

"WHERE Subject LIKE '%" + txtSearch.Text + "%';";

Operations.DisplayAndSearch(searchSql, dataGridView1);

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 0)

{

correspondenceForm.id = dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString();

correspondenceForm.sender = dataGridView1.Rows[e.RowIndex].Cells[3].Value.ToString();

correspondenceForm.receiver = dataGridView1.Rows[e.RowIndex].Cells[4].Value.ToString();

correspondenceForm.subject = dataGridView1.Rows[e.RowIndex].Cells[5].Value.ToString();

correspondenceForm.correspondenceContent = dataGridView1.Rows[e.RowIndex].Cells[6].Value.ToString();

correspondenceForm.sentDate = dataGridView1.Rows[e.RowIndex].Cells[7].Value.ToString();

correspondenceForm.receivedDate = dataGridView1.Rows[e.RowIndex].Cells[8].Value.ToString();

correspondenceForm.UpdateInfo();

correspondenceForm.ShowDialog();

return;

}

if (e.ColumnIndex == 1)

{

if (MessageBox.Show("Дійсно видалити?", "Інформація", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

Operations.Delete(dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString(), "CorrespondenceID", "ExternalCorrespondence");

Display();

}

return;

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

correspondenceForm.DefaultInfo();

correspondenceForm.ShowDialog();

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using bank;

using System;

using System.Windows.Forms;

namespace bank

{

public partial class FormInternalDocuments : Form

{

private readonly Main \_parent;

FormNewInternalDocuments documentForm;

public string sql = "SELECT DocumentID, DocumentTitle, DocumentContent, CreatedDate, LastModifiedDate FROM InternalDocuments;";

public FormInternalDocuments(Main parent)

{

InitializeComponent();

\_parent = parent;

documentForm = new FormNewInternalDocuments(this);

}

public void Display()

{

Operations.DisplayAndSearch(sql, dataGridView1);

}

private void FormInternalDocuments\_Load(object sender, EventArgs e)

{

Display();

}

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

string searchSql = "SELECT DocumentID, DocumentTitle, DocumentContent, CreatedDate, LastModifiedDate FROM InternalDocuments " +

"WHERE DocumentTitle LIKE '%" + txtSearch.Text + "%';";

Operations.DisplayAndSearch(searchSql, dataGridView1);

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 0)

{

documentForm.id = dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString();

documentForm.documentTitle = dataGridView1.Rows[e.RowIndex].Cells[3].Value.ToString();

documentForm.documentContent = dataGridView1.Rows[e.RowIndex].Cells[4].Value.ToString();

documentForm.createdDate = dataGridView1.Rows[e.RowIndex].Cells[5].Value.ToString();

documentForm.lastModifiedDate = dataGridView1.Rows[e.RowIndex].Cells[6].Value.ToString();

documentForm.UpdateInfo();

documentForm.ShowDialog();

return;

}

if (e.ColumnIndex == 1)

{

if (MessageBox.Show("Дійсно видалити?", "Інформація", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

Operations.Delete(dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString(), "DocumentID", "InternalDocuments");

Display();

}

return;

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

documentForm.DefaultInfo();

documentForm.ShowDialog();

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using bank;

using System;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace library

{

public partial class FormNewAccount : Form

{

private readonly FormAccounts \_parent;

public string? id, clientID, accountNumber, accountType, balance;

public FormNewAccount(FormAccounts parent)

{

InitializeComponent();

\_parent = parent;

}

public void UpdateInfo()

{

textBox5.Text = clientID;

textBox4.Text = accountNumber;

textBox1.Text = accountType;

textBox2.Text = balance;

label1.Text = "Редагування рахунку";

btnAdd.Text = "Оновити";

}

public void DefaultInfo()

{

Operations.Clear(this);

label1.Text = "Додавання рахунку";

btnAdd.Text = "Зберегти";

}

private void btnAdd\_Click(object sender, EventArgs e)

{

Account a = new Account(textBox5.Text.Trim(), textBox4.Text.Trim(), textBox1.Text.Trim(), textBox2.Text.Trim());

string insert = "INSERT INTO Accounts (ClientID, AccountNumber, AccountType, Balance) " +

"VALUES (@ClientID, @AccountNumber, @AccountType, @Balance);";

string update = "UPDATE Accounts SET ClientID = @ClientID, AccountNumber = @AccountNumber, AccountType = @AccountType, " +

"Balance = @Balance WHERE AccountID = @AccountID;";

if (btnAdd.Text == "Зберегти")

{

Operations.ManageAccount(a, insert);

}

if (btnAdd.Text == "Оновити")

{

Operations.ManageAccount(a, update, id);

}

Operations.Clear(this);

\_parent.Display();

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using library;

using System;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace bank

{

public partial class FormNewClient : Form

{

private readonly FormClients \_parent;

public string? id, firstName, lastName, dateOfBirth, phone, email, address;

public FormNewClient(FormClients parent)

{

InitializeComponent();

\_parent = parent;

}

public void UpdateInfo()

{

textBox5.Text = firstName;

textBox4.Text = lastName;

textBox1.Text = dateOfBirth;

textBox2.Text = phone;

textBox6.Text = email;

textBox3.Text = address;

label1.Text = "Редагування клієнта";

btnAdd.Text = "Оновити";

}

public void DefaultInfo()

{

Operations.Clear(this);

label1.Text = "Додавання клієнта";

btnAdd.Text = "Зберегти";

}

private void btnAdd\_Click(object sender, EventArgs e)

{

Client c = new Client(textBox5.Text.Trim(), textBox4.Text.Trim(), textBox1.Text.Trim(), textBox2.Text.Trim(), textBox6.Text.Trim(), textBox3.Text.Trim());

string insert = "INSERT INTO Clients (FirstName, LastName, DateOfBirth, Phone, Email, Address) " +

"VALUES (@FirstName, @LastName, @DateOfBirth, @Phone, @Email, @Address);";

string update = "UPDATE Clients SET FirstName = @FirstName, LastName = @LastName, DateOfBirth = @DateOfBirth, " +

"Phone = @Phone, Email = @Email, Address = @Address WHERE ClientID = @ClientID;";

if (btnAdd.Text == "Зберегти")

{

Operations.ManageClient(c, insert);

}

if (btnAdd.Text == "Оновити")

{

Operations.ManageClient(c, update, id);

}

Operations.Clear(this);

\_parent.Display();

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using bank;

using System;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace bank

{

public partial class FormNewExternalCorrespondence : Form

{

private readonly FormExternalCorrespondence \_parent;

public string? id, sender, receiver, subject, correspondenceContent, sentDate, receivedDate;

public FormNewExternalCorrespondence(FormExternalCorrespondence parent)

{

InitializeComponent();

\_parent = parent;

}

public void UpdateInfo()

{

textBox5.Text = sender;

textBox4.Text = receiver;

textBox1.Text = subject;

textBox2.Text = correspondenceContent;

textBox3.Text = sentDate;

textBox6.Text = receivedDate;

label1.Text = "Редагування кореспонденції";

btnAdd.Text = "Оновити";

}

public void DefaultInfo()

{

Operations.Clear(this);

label1.Text = "Додавання кореспонденції";

btnAdd.Text = "Зберегти";

}

private void btnAdd\_Click(object sender, EventArgs e)

{

ExternalCorrespondence ec = new ExternalCorrespondence(textBox5.Text.Trim(), textBox4.Text.Trim(), textBox1.Text.Trim(), textBox2.Text.Trim(), textBox3.Text.Trim(), textBox6.Text.Trim());

string insert = "INSERT INTO ExternalCorrespondence (Sender, Receiver, Subject, CorrespondenceContent, SentDate, ReceivedDate) " +

"VALUES (@Sender, @Receiver, @Subject, @CorrespondenceContent, @SentDate, @ReceivedDate);";

string update = "UPDATE ExternalCorrespondence SET Sender = @Sender, Receiver = @Receiver, Subject = @Subject, CorrespondenceContent = @CorrespondenceContent, " +

"SentDate = @SentDate, ReceivedDate = @ReceivedDate WHERE CorrespondenceID = @CorrespondenceID;";

if (btnAdd.Text == "Зберегти")

{

Operations.ManageExternalCorrespondence(ec, insert);

}

if (btnAdd.Text == "Оновити")

{

Operations.ManageExternalCorrespondence(ec, update, id);

}

Operations.Clear(this);

\_parent.Display();

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using bank;

using System;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace bank

{

public partial class FormNewInternalDocuments : Form

{

private readonly FormInternalDocuments \_parent;

public string? id, documentTitle, documentContent, createdDate, lastModifiedDate;

public FormNewInternalDocuments(FormInternalDocuments parent)

{

InitializeComponent();

\_parent = parent;

}

public void UpdateInfo()

{

textBox5.Text = documentTitle;

textBox4.Text = documentContent;

textBox1.Text = createdDate;

textBox2.Text = lastModifiedDate;

label1.Text = "Редагування документа";

btnAdd.Text = "Оновити";

}

public void DefaultInfo()

{

Operations.Clear(this);

label1.Text = "Додавання документа";

btnAdd.Text = "Зберегти";

}

private void btnAdd\_Click(object sender, EventArgs e)

{

InternalDocument d = new InternalDocument(textBox5.Text.Trim(), textBox4.Text.Trim(), textBox1.Text.Trim(), textBox2.Text.Trim());

string insert = "INSERT INTO InternalDocuments (DocumentTitle, DocumentContent, CreatedDate, LastModifiedDate) " +

"VALUES (@DocumentTitle, @DocumentContent, @CreatedDate, @LastModifiedDate);";

string update = "UPDATE InternalDocuments SET DocumentTitle = @DocumentTitle, DocumentContent = @DocumentContent, CreatedDate = @CreatedDate, " +

"LastModifiedDate = @LastModifiedDate WHERE DocumentID = @DocumentID;";

if (btnAdd.Text == "Зберегти")

{

Operations.ManageInternalDocument(d, insert);

}

if (btnAdd.Text == "Оновити")

{

Operations.ManageInternalDocument(d, update, id);

}

Operations.Clear(this);

\_parent.Display();

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using bank;

using System;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace library

{

public partial class FormNewTransaction : Form

{

private readonly FormTransactions \_parent;

public string? id, accountID, transactionDate, transactionType, amount;

public FormNewTransaction(FormTransactions parent)

{

InitializeComponent();

\_parent = parent;

}

public void UpdateInfo()

{

textBox1.Text = accountID;

textBox2.Text = transactionDate;

textBox3.Text = transactionType;

textBox4.Text = amount;

label1.Text = "Редагування транзакції";

btnAdd.Text = "Оновити";

}

public void DefaultInfo()

{

Operations.Clear(this);

label1.Text = "Додавання транзакції";

btnAdd.Text = "Зберегти";

}

private void btnAdd\_Click(object sender, EventArgs e)

{

Transaction t = new Transaction(textBox1.Text.Trim(), textBox2.Text.Trim(), textBox3.Text.Trim(), textBox4.Text.Trim());

string insert = "INSERT INTO Transactions (AccountID, TransactionDate, TransactionType, Amount) " +

"VALUES (@AccountID, @TransactionDate, @TransactionType, @Amount);";

string update = "UPDATE Transactions SET AccountID = @AccountID, TransactionDate = @TransactionDate, TransactionType = @TransactionType, " +

"Amount = @Amount WHERE TransactionID = @TransactionID;";

if (btnAdd.Text == "Зберегти")

{

Operations.ManageTransaction(t, insert);

}

if (btnAdd.Text == "Оновити")

{

Operations.ManageTransaction(t, update, id);

}

Operations.Clear(this);

\_parent.Display();

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using bank;

using library;

using System;

using System.Windows.Forms;

namespace library

{

public partial class FormTransactions : Form

{

private readonly Main \_parent;

FormNewTransaction transactionForm;

public string sql = "SELECT TransactionID, AccountID, TransactionDate, TransactionType, Amount FROM Transactions;";

public FormTransactions(Main parent)

{

InitializeComponent();

\_parent = parent;

transactionForm = new FormNewTransaction(this);

}

public void Display()

{

Operations.DisplayAndSearch(sql, dataGridView1);

}

private void FormTransactions\_Load(object sender, EventArgs e)

{

Display();

}

private void txtSearch\_TextChanged(object sender, EventArgs e)

{

string searchSql = "SELECT TransactionID, AccountID, TransactionDate, TransactionType, Amount FROM Transactions " +

"WHERE TransactionType LIKE '%" + txtSearch.Text + "%';";

Operations.DisplayAndSearch(searchSql, dataGridView1);

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 0)

{

transactionForm.id = dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString();

transactionForm.accountID = dataGridView1.Rows[e.RowIndex].Cells[3].Value.ToString();

transactionForm.transactionDate = dataGridView1.Rows[e.RowIndex].Cells[4].Value.ToString();

transactionForm.transactionType = dataGridView1.Rows[e.RowIndex].Cells[5].Value.ToString();

transactionForm.amount = dataGridView1.Rows[e.RowIndex].Cells[6].Value.ToString();

transactionForm.UpdateInfo();

transactionForm.ShowDialog();

return;

}

if (e.ColumnIndex == 1)

{

if (MessageBox.Show("Дійсно видалити?", "Інформація", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

Operations.Delete(dataGridView1.Rows[e.RowIndex].Cells[2].Value.ToString(), "TransactionID", "Transactions");

Display();

}

return;

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

transactionForm.DefaultInfo();

transactionForm.ShowDialog();

}

private void btnBack\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using library;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace bank

{

public partial class Main : Form

{

FormTransactions transactions;

FormClients client;

FormAccounts accounts;

FormExternalCorrespondence correspondence;

FormInternalDocuments documents;

public Main()

{

InitializeComponent();

transactions = new FormTransactions(this);

client = new FormClients(this);

accounts = new FormAccounts(this);

correspondence = new FormExternalCorrespondence(this);

documents = new FormInternalDocuments(this);

}

private void button1\_Click(object sender, EventArgs e)

{

client.ShowDialog();

}

private void button2\_Click(object sender, EventArgs e)

{

accounts.ShowDialog();

}

private void button3\_Click(object sender, EventArgs e)

{

transactions.ShowDialog();

}

private void button4\_Click(object sender, EventArgs e)

{

documents.ShowDialog();

}

private void button5\_Click(object sender, EventArgs e)

{

correspondence.ShowDialog();

}

private void button6\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}