**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Тернопільський національний економічний університет**
**Факультет комп'ютерних інформаційних технологій**
Кафедра комп'ютерних наук

**ТИХОВИЧ Лілія Олександрівна**

# Програма для розрахунку оптимальних параметрів споживацького кредиту/ Software for computing an optimal parameter set of consumer credit in bank

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконала студентка групи
ПЗС-41
Л. О. Тихович
_____

Науковий керівник:
к.е.н., доцент АВГУСТИН Р.Р.
_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

## РЕЗЮМЕ

**Дипломна робота** містить 66 сторінку, 11 таблиць, 37 рисунків, список використаних джерел із 21 найменувань.

**Метою дипломної роботи** є розробка програмної системи для розрахунку оптимальних параметрів споживацького кредиту.

**Об'єктом дослідження** є процес видачі кредиту.

**Предметом дослідження** є програмний продукт для визначення оптимальних параметрів кредиту.

Методи розробки базуються на технології .NET. та мові програмування C#.

**Одержані результати** полягають в розробці програмної системи для визначення параметрів оптимального кредиту.

**Ключові слова**: кредит, термін, сума, заборгованість, оптимальний.

## RESUME

Thesis contains 66 page, 11 tables, 37 figures, list of references with 21 titles.

The aim of the thesis is to develop a software system for calculating the optimal parameters of consumer credit.

The object of the research is the process of credit.

The subject of the study is the software to determine optimal parameters of credit.

The methods of making technology based on .NET. and the C # programming language.

The results are in development of a software system to determine the optimum parameters of credit.

Keywords: loan term, the amount payable, the best.

ЗМІСТ

# ВСТУП

***Актуальність теми***

Розвиток економіки будь якої держави в великій мірі залежить від рівня фінансування тих чи інших галузей народного господарства, або його громадян. З точки зору будь який кредит повинен приносити позитивний ефект, як для установ які видають кредит, так і для установ які отримують кредити. Банківські установи пропонують вже визначені ними параметри кредитів, на які вплинути не можливо, тому програма, яка буде визначати уму та термін кредиту для користувача кредитом є актуальною.

***Мета і задачі розробки***

Метою розробки є створення програмної засобу для підтримки прийняття рішень при отриманні кредиту. В процесі досягнення мети необхідно буде вирішити наступні задачі:

- вивчити існуючі системи визначення оптимальних параметрів кредиту;
- встановити функції програми встановлення оптимальних параметрів кредиту;
- спроектувати інформаційну систему для підтримки прийняття рішень при отриманні кредиту;
- реалізувати систему для визначення оптимальних параметрів кредиту;
- провести тестування та почати етап дослідної експлуатації програми

***Методи, засоби та технології розробки***

У даному випадку було вибрано методи аналізу та синтезу. Аналіз — це метод пізнання, який дає змогу поділити предмет на частини. Синтез, навпаки, є наслідком з'єднання окремих частин чи рис предмета в єдине ціле.

Аналіз та синтез взаємопов'язані, вони являють собою єдність протилежностей. Залежно від рівня пізнання об'єкта та глибини проникнення в його сутність застосовуються аналіз і синтез різного роду.

***Практичне значення одержаних результатів***

Практична значущість отриманих результатів полягає у програми, яка допоможе встановити оптимальні параметри кредиту.

# РОЗДІЛ І
# РОЗДІЛ АНАЛІЗУ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СПЕЦИФІКАЦІЇ ВИМОГ ДО ПРОГРАМИ РОЗРАХУНКУ ОПТИМАЛЬНИХ ПАРАМЕТРІВ КРЕДИТУ

## 1.1 Коротка характеристика об’єкту управління

В банках щодня приймаються рішення щодо залучення коштів та видачі кредитів. В зв’язку з цим ліквідність банку по балансу може бути задовільною, але реальна ліквідність, виходячи з грошових потоків, - зовсім ні. В процесі аналізу банківської звітності основним об’єктом дослідження виступає вся його сукупна комерційна діяльність. При цьому суб’єктами аналізу виступають самі комерційні банки, їх контрагенти, включаючи НБУ, інші кредитні установи, аудиторські фірми, владні структури, реальні і потенційні клієнти і кореспонденти, засновники і акціонери. Оскільки кожен із суб’єктів має власні цілі, то різні будуть напрямки і критерії аналізу.

Комерційні банки з допомогою аналізу своїх балансових даних перевіряють ступінь реалізації основних цільових установок у своїй діяльності: фактори їх доходності, збалансованість структури активних і пасивних операцій з метою підтримання ліквідності, дотримання економічних нормативів, встановлених НБУ, мінімізацію всіх видів банківських ризиків...

Комерційні банки також зацікавлені аналізувати дані про стан інших банків, однак вони не мають (і не можуть мати) всієї необхідної при цьому інформації. За кордоном банки володіють великими можливостями аналізувати інформаційні потоки, маючи при цьому єдину форму звітності банків, налагоджений облік і практику обміну інформацією, яка складається десятиліттями. Тепер банкіри розуміють, що за деяким винятком всі виграють від опублікування банківської інформації. Це накладає певну ринкову дисципліну на банки. В різних країнах при цьому є певні особливості.

Клієнти і кореспонденти банку визначають стійкість фінансового стану банку і його надійність, перспективи розвитку.

Аудиторські служби в процесі аналізу балансу перевіряють достовірність аналітичного і синтетичного обліку, звітності банку, правильність відображення діяльності банку в його балансі.

НБУ цікавить в першу чергу стан і стійкість банківської системи. Він аналізує дотримання банками економічних нормативів, відрахувань в централізовані фонди і визначає ефективність регулювання державою банківської діяльності. Важливо відзначити, що НБУ, маючи всю необхідну для аналізу інформацію, складає свої рейтинги, але не публікує їх. НБУ широко публікує зведені, порівняльні та інші дані, необхідні для аналізу стійкості і надійності комерційних банків.

НБУ може переводити комерційні банки на режим фінансового оздоровлення, підставою для чого може бути невиконання протягом трьох місяців загальновстановлених норм та нормативів ведення банківської справи, в обов’язковому порядку визначених НБУ. Невиконанням норм і нормативів слід вважати:

- неякісну оцінку капіталу;
- збиткову діяльність, яка характеризується наявністю збитків минулого та поточного років і неможливістю їх реального погашення протягом трьох місяців;
- порушення встановлених НБУ економічних нормативів та оціночних показників діяльності банку;
- неякісну структуру активів, в тому числі кредитного портфеля.

Тоді приймається рішення про надання стабілізаційної позики - це позика, яка надається Нацбанком комерційному банку для оперативного забезпечення його платоспроможності і ліквідності та підтримки виконання заходів фінансового оздоровлення.

НБУ здійснює нагляд за діяльністю комерційних банків, їх відділень, філій, представництв на території України. Він спрямований на забезпечення стабільності банківської системи, захист інтересів вкладників шляхом зменшення ризиків в діяльності комерційних банків. Зміст нагляду визначається повноваженнями, встановленими законом України «Про банки і банківську діяльність». Система нагляду спрямована на скорочення внутрішніх і зовнішніх ризиків. Серед зовнішніх виділяють ризик ліквідності, валютний ризик, ризик облікової ставки та ризик по цінних паперах. До внутрішніх відносять «комерційні ризики», пов'язані з людським фактором (кваліфікація персоналу і ділові якості керівників, виконавська дисципліна та якість аудиторської служби і ін.), а також «операційно-технічні ризики», які відображають ступінь працездатності систем, які забезпечують зовнішню роботу банку: системи безпеки, бух обліку, матеріально-технічних засобів, засобів зв'язку і т. п. На зниження внутрішніх ризиків спрямовані процедури реєстрації банків, ліцензування, внутрішніх перевірок, інспекції діяльності комерційних банків Нацбанком. При цьому для забезпечення фінансової стійкості банку вважливим є не тільки визначення загальної концепції його розвитку, але й розробка раціональних схем формування пасивів і розподіл його ресурсів за основними категоріями активів.

У спробах розв'язати дилему надійність, ліквідність - прибутковість у світовій банківській практиці визначились три основних підходи до управління активами:

1) метод загального фонду коштів (об'єднання джерел), в основі якого лежить ідея об'єднати всі ресурси банку з наступним їх розміщенням відповідно до визначених пріоритетів, призначення котрих - допомогти керівництву оперативних відділів розв'язати проблему поєднання надійності, ліквідності і прибутковості;
2) метод розподілу активів або конверсія коштів. Головною перевагою цього методу є зменшення частки ліквідних активів та вкладень

додаткових коштів у позики та інвестиції, що веде до збільшення норми прибутку, і, відповідно, до його фінансової стійкості;

збалансований науковий підхід до розв'язання управлінських проблем регулювання фінансової стійкості банку з використанням прогресивних методів та ЕОМ для вивчення елементів у складних модулях, наприклад, лінійного програмування.

## 1.2 Огляд і аналіз існуючих аналогів, що реалізують функції предметної області

### 1.2.1. Програмна система RIVC-SYSTEM

На підставі проведеного аналізу методів управління і оцінки кредитного ризику, розроблених проектів, створена система підтримки прийняття рішень RIVC-SYSTEM, що реалізує основні ключові етапи в управлінні кредитним ризиком (малюнок 3): ідентифікація, оцінка, контроль.

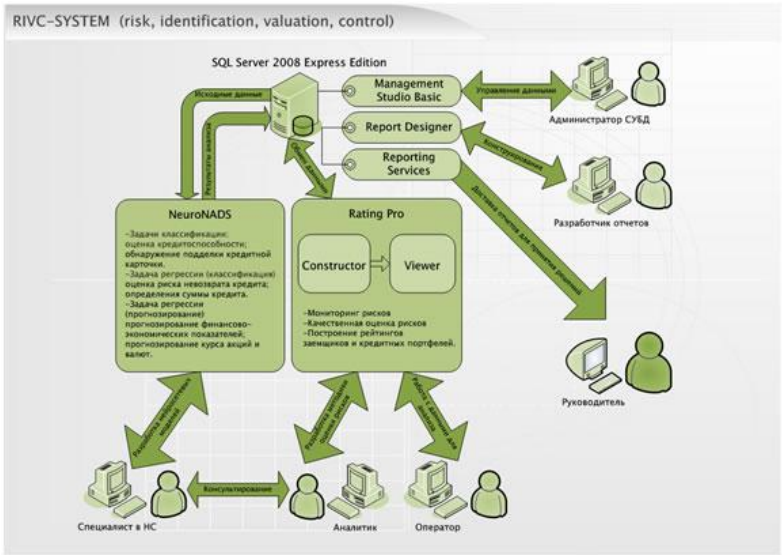


Рисунок1.1- Структура системи підтримки прийняття управлінських рішень RIVC-SYSTEM

Розроблена система моніторингу і побудови рейтингів, під назвою Rating PRO дозволяє:

1) в зручній формі здійснювати введення вихідної інформації;

2) виконувати розрахунок підсумкових показників, необхідних для якісної оцінки ризику;

3) значно економити інформаційні ресурси на технічних засобах за рахунок виключення дублювання даних ;

4) забезпечити санкціонований доступ до необхідної інформації;

5) скоротити терміни оцінки ризику;

6) проводити постійний моніторинг ризику.

Основною перевагою системи є можливість конструювання методики оцінки ризику. Труднощами використання системи є вибір коефіцієнтів і суб'єктивність значущості інтегрального показника.

Розроблена система інтелектуального аналізу даних NeuroNADS 4 реалізує наступний необхідний функціонал:

можливість введення даних з різних джерел, включення обчислюваних стовпців;

створення архітектури НС (обмеження тільки по обчислювальних ресурсів);

нормалізація входів і виходів НС;

виділення якого навчають та тестової множин;

вибір функції активації нейрона;

необхідний набір параметрів навчання;

візуалізація навчання;

опис нейросистеми;

графічне і числове представлення отриманих результатів;

збереження конфігурації нейросистеми і використання її в подальшому.

Нейронні мережі є дуже потужним інструментом, застосовуваним в рішення різних завдань. Плюсом нейромереж є об'єктивність при прийнятті рішення. Недоліком є складність використання.

Безкоштовне застосування в якості СУБД SQL Server 2008 Express дозволило побудувати клієнт-серверну архітектуру з усіма її перевагами, а включені можливості Advanced Services - конструювати і доставляти звіти кінцевим користувачам.

RIVC-SYSTEM може бути розгорнута як внутрішня система управління кредитним ризиком відповідного департаменту, так і як складова частина єдиної системи підтримки прийняття управлінських рішень банку.

Введення в експлуатацію системи RIVC-SYSTEM схильний певним ризикам, серед яких можуть бути: орієнтація на MS Excel, неприйняття системи користувачами, переоцінка можливостей системи, переоцінка можливостей користувачів.

### 1.2.2. Програмне забезпечення АБС «BARS-Millennium»

Загальні відомості

Цей документ містить першу з трьох частин керівництва користувача автоматизованої банківської системи "БАРС-Millenium" (АБС). В даному документі наведено класифікацію користувачів АБС, дано загальний опис користувальницького інтерфейсу, стисло описані роботи користувачів с АБС в розрізі функцій, об'єднаних в рамках банківського модуля.

У цьому документі використовуються наступні скорочення і позначення:

АБС - автоматизована банківська система;

АРМ - автоматизоване робоче місце;

НБУ - Національний банк України;

НДІ - нормативно-довідкова інформація;

СЕП - система електронних платежів НБУ;

ОДБ - операційний день банку;

ОС - операційна система;

БД - база даних.

Класифікація користувачів АБС

Користувач - співробітник банку, який експлуатує АБС відповідно до нада повноваженнями. У документі "Автоматизована Банківська Система 'БАРС-Millenium'. Загальний опис "наведено загальне визначення користувачів і загальна класифікація з точки зору повноважень. У документі "Автоматизована Банківська Система 'БАРС-Millenium'. Керівництво по початковому впровадження "описані настройки по організації доступу користувачів до елементам АБС.

Передбачаються наступні очевидні категорії користувачів АБС:

· Адміністратор АБС;

· Керівник;

· Головний бухгалтер;

· Технолог;

· Операціоніст;

· Контролер;

· Скарбник.

Крім того, в залежності від потреб банку, можуть вводитися інші категорії користувачів. При початковому впровадженні, для кожної категорії користувачів і кожного співробітника банку, який буде експлуатувати АБС, передбачається індивідуальне побудова АРМа. При експлуатації доступ до елементів АБС здійснюється за допомогою виклику певних пунктів меню АРМов. Нижче наведено опис роботи користувачів с АБС в розрізі функцій, об'єднаних в рамках банківського модуля.

Інтерфейс в АБС

АБС "БАРС-Millenium" має стандартний віконний інтерфейс, побудований на використанні звичних для Windows діалогових вікон, меню і методів роботи з ними. У цьому посібнику використовуються наступні угоди, а також визначення керуючих елементів діалогових вікон:

Текстове поле (поле редагування) - місце для введення будь-якої текстової інформації.

Список - набір пропонованих на вибір значень, з яких можна вибрати тільки одне. Якщо список допускає одночасний вибір кількох значень - всі необхідні елементи потрібно общелкать мишкою, утримуючи клавішу CTRL. Якщо список містить більше елементів, ніж можна зобразити одночасно у вікні - необхідно використовувати смуги прокрутки.

Список, що розкривається - текстове поле, забезпечене кнопкою з спрямованої вниз стрілкою. При натисканні на кнопку зі стрілочкою на екрані з'явиться список.

Перемикачі (кнопки вибору) використовуються для вибору одного з взаємовиключних варіантів і бувають круглими або ромбічними.

Прапорці - кнопки, зображувані квадратиками. Можна використовувати групу прапорців, один або жодного. Для вибору прапорця необхідно клацнути по квадратику або відповідної текстовому рядку. Для скасування прапорця необхідно повторити щелчек.

Текстові поля, кнопки вибору (перемикачі), списки, що розкриваються списки, прапорці забезпечені текстами, що відображають їх призначення.

Елемент діалогового вікна, зображений сірим кольором знаходиться в неактивному стані (недоступний).

Виділення елементів списків, кнопок, установка прапорців здійснюється стандартним для Windows чином (клацанням миші або з використанням клавіатури). Вибір виділеного пункту активного меню здійснюється аналогічно.

Поля діалогових вікон, що містять списки рахунків, клієнтів, банківських документів і іншу банківську інформацію, як правило представлені в табличномувигляді.

При виконанні важливих технологічних маніпуляцій (відкриття / закриття рахунку і т.д.) потрібно підтвердження на відповідний запит.

Загальне керівництво здійснюється за допомогою команд, об'єднаних в систему меню. Вибір Пукто меню відповідає вибору деякої функції, реалізованої в АБС.

Для швидкого переходу до певної функції АБС використовуються наступні " швидкі клавіші " (можуть бути використані в будь-якій точці діалогу):

Ctrl + A - перегляд рахунків

Ctrl + D - перегляд операцій (документи користувача)

Ctrl + До -виклик фінансового калькулятора.

F2 - перехід між вікнами меню АРМ / Функції

При роботі програми доступні наступні " швидкі клавіші " Windows:

Ctrl + Esc - перейти в Головне меню

Alt + Tab - переключитися останнім використовувалося вікно

Alt + F4 - завершити програму

При роботі в діалогових вікнах, як правило, в лівому нижньому кутку з'являється підказка про призначення інструменту.

## 1.3 Специфікація вимог до системи

Система, що проектується має розташовуватися на робочих місцях працівників відділення банку, яка дозволяє працювати з клієнтами банку, які хочуть отримати кредит.

Ця система дозволяє здійснювати наступні операції:

- запустити програму тобто “відкрити день”;
- ініціалізувати кредитного інспектора за допомогою індивідуальних засобів аутентифікації (так звана «таблетка»);
- підтвердити дату і час початку роботи;
- для роботи з програмою необхідно відкрити з головного меню «Кредитний калькулятор»;

- у відповідні поля внести вартість товару (сума кредиту), авансовий платіж (або залишити його нульовим);
- вибрати кредитний продукт натиснувши кнопку «Вибрати»;
- після вибору кредитного продукту з’явиться можливість роздрукувати розрахунок орієнтованих щомісячних платежів натиснувши кнопку «друкувати»;
- після прийняття рішення можна приступати до оформлення заявки вибравши команду «Нова заявка»;
- після чого система автоматично здійснить контроль наявності необхідних документів Позичальника для оформлення кредиту;
- після перевірки відкривається вікно введення даних Позичальника. Вікно складається з десяти закладок, в кожній з яких існують обов’язкові поля.

У вікні введення даних про Позичальника присутні наступні закладки:

1. дані про кредит;
2. ідентифікаційні дані;
3. контакти;
4. освіта та соціальні дані;
5. робота;
6. сімейний стан;
7. доходи та витрати;
8. майно;
9. інше;
10. поручитель;

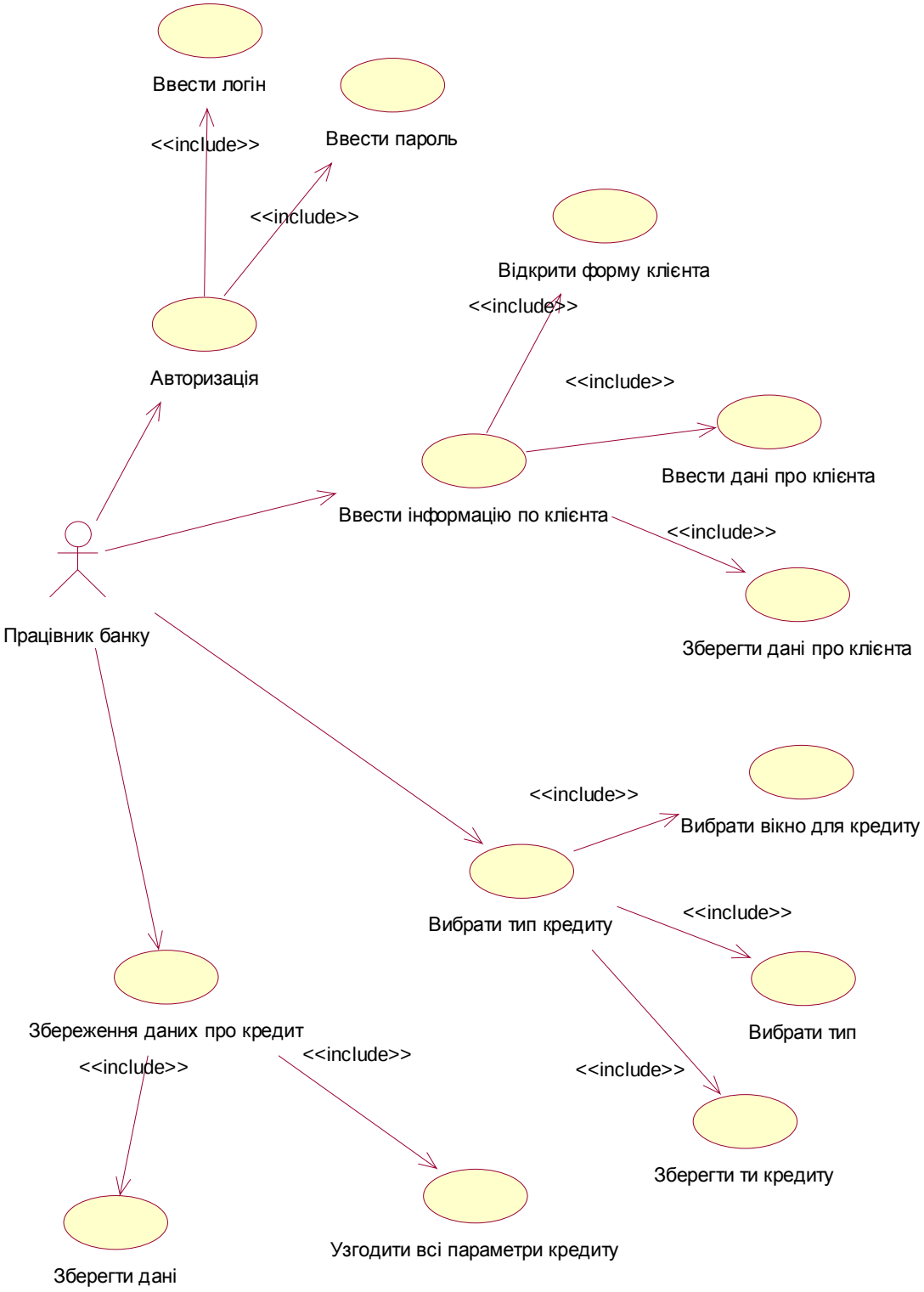


Рисунок 1.2- Загальна діаграма варіантів використання

Користувачем системи є працівник банку, який використовує систему для визначення параметрів кредиту та оформлення кредитної заявки.

На рисунку 1.2 показано діаграму варіантів використання для працівника банку, що приймає рішення про надання кредиту.

Таблиця 1.1

Глосарій проекту

| Термін | Опис терміну |
|---|---|
| 1. Основні поняття та категорії предметної області та проекту | |
| Банківська система | Сукупність різних видів національних банків і кредитних установ, що діють в рамках загального грошово-кредитного механізму. |
| Мета | Означає стан в майбутньому, котрий можливо змінити відносно теперішнього та варто, бажано або необхідно досягнути. |
| Кредит | ( лат. Creditum - позика від лат. Credere - довіряти) або кредитні відносини - суспільні відносини , що виникають між суб'єктами економічних відносин з приводу руху вартості |
| Прогноз | Являє собою спеціальне наукове дослідження конкретних перспектив розвитку якого-небудь явища |
| 2. Користувачі системи | |
| Користувач (працівник банку) | Менеджер банку з видачі кредитів |
| 3. Вхідні та вихідні документи | |
| Інформація про кредитоотримувача | Документ в якому прописано інформація про особу, що отримує кредит |
| Інформація про кредит | Документ де описано всі параметри кредиту по відношенні до його виду |
| Договір про отримання кредиту | Документ де зафіксовано наміри та відповідальність сторін при взятті кредиту |

На рисунку 1.2 зображено загальну діаграму варіантів використання проектованого програмного забезпечення.

Далі наводимо опис варіантів використання, що реалізують основну функціональність у вигляді таблиць.

Таблиця 1.2

Варіант використання «Авторизація»

| | |
|---|---|
| Контекст використання | Вхід в систему |
| Дійові особи | Працівник банку |
| Передумови | Початок роботи з системою |
| Триггер | Запуск програми на виконання |
| Сценарій | Ввести логін<br>Ввести пароль<br>Натиснути кнопку «Вхід» |
| Постумова | Отримання доступу до елементів системи<br>Стартове вікно «Панель керування» |

Таблиця 1.3

Варіант використання «Ввести інформацію про клієнта»

| | |
|---|---|
| Контекст використання | Зібрати первинні дані про клієнта |
| Дійові особи | Працівник банку |
| Передумови | Оформлення кредитного договору |
| Триггер | Вхід в систему працівником банку |
| Сценарій | Ввести дані про клієнта<br>Зберегти дані |
| Постумова | Перехід до визначення параметрів кредиту |

Таблиця 1.4

Варіант використання «Вибрати тип кредиту»

| | |
|---|---|
| Контекст використання | Необхідність вибрати тип кредиту |
| Дійові особи | Працівник банку |
| Передумови | Вибір параметрів кредиту |
| Триггер | Вибір характеристик договору по кредиту |
| Сценарій | Відкрити модуль для вибору параметрів кредиту<br>Ввести суму кредиту<br>Вибрати тип кредиту<br>Зберегти дані |
| Постумова | Оформлення заявки на кредит |

Таблиця 1.5

Варіант використання «Збереження даних про кредит»

| | |
|---|---|
| Контекст використання | Отримано згоду про отримання кредиту |
| Дійові особи | Працівник банку |
| Передумови | Записані дані про клієнта та вибрано тип кредиту |
| Триггер | Завершення роботи з клієнтм |
| Сценарій | Вибрати команду зберегти<br>Вибрати місце збереження даних<br>Вибрати команду зберегти |
| Постумова | Сформовано кредитний договір |

Далі проводимо розкадровку варіантів використання для моделювання необхідних екранних форм, які будуть використовуватися для реалізації функцій системи вибору оптимального кредиту.

Розкадровка варіанту використання «Автоизація» показана на рисунку 1.3.

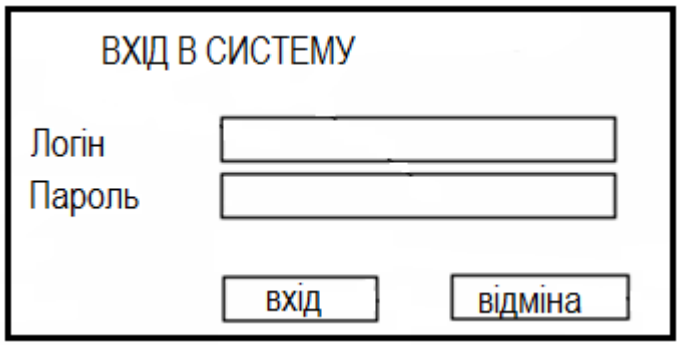


Рисунок 1.3.- Розкадровка варіанту використання «Авторизація»

Розкадровка варіанту використання «Ввести інформацію про клієнта» показана на рисунку 1.4.

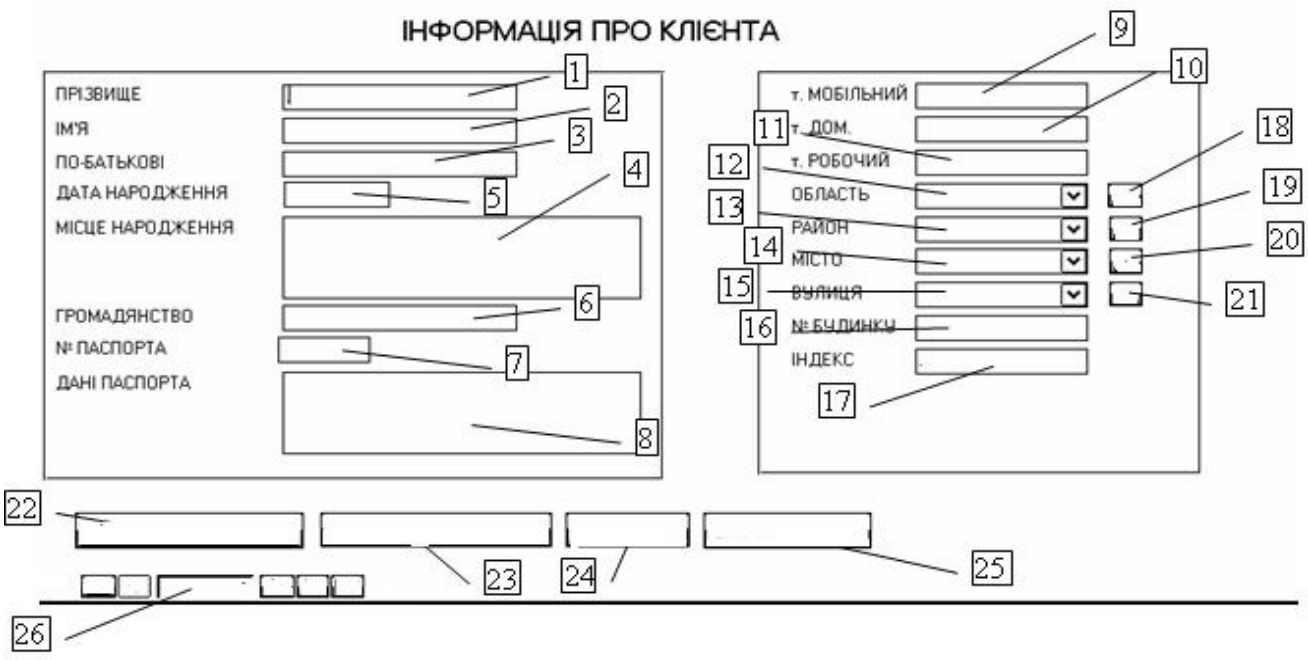


Рисунок 1.4.- Розкадровка варіанту використання «Ввести інформацію про клієнта»

Графічним елементам діалогового вікна (рисунок 1.4) пропонується надати наступні функції:

1. –введення прізвища клієнта;
2. - введення імені клієнта;
3. -введення по-бітькові клієнта;
4. - введення інформації про місце народження клієнта;

5. -введення дати народження клієнта;
6. -введення дати народження клієнта;
7. -введення номера паспорта клієнта;
8. -введення дати про паспорт (дата, місце видачі, реєстрація ) клієнта;
9. -введення номера мобільного телефону клієнта;
10. -введення номера домашнього телефону клієнта;
11. -введення номера робочого телефону клієнта;
12. –вибір із списку області проживання клієнта;
13. –вибір із списку району проживання клієнта;
14. –вибір із списку міста проживання клієнта;
15. –вибір із списку вулиці проживання клієнта;
16. -введення номера будинку проживання клієнта;
17. -введення номера поштового індексу місця проживання клієнта;
18. –відкриття ділового вікна для додавання нового запису назви області;
19. – відкриття ділового вікна для додавання нового запису назви району;
20. – відкриття ділового вікна для додавання нового запису назви міста;
21. – відкриття ділового вікна для додавання нового запису назви вулиці;
22. -відкриття ділового вікна для вибору кредиту;
23. -відкриття ділового вікна для фіксації кредиту;
24. -відкриття ділового вікна для пошуку клієнта;
25. -закриття ділового вікна;
26. –група кнопок для додавання, видалення, редагування записів в базі даних;

Розкадровка варіанту використання «Вибрати тип кредиту» показана на рисунку 1.5.

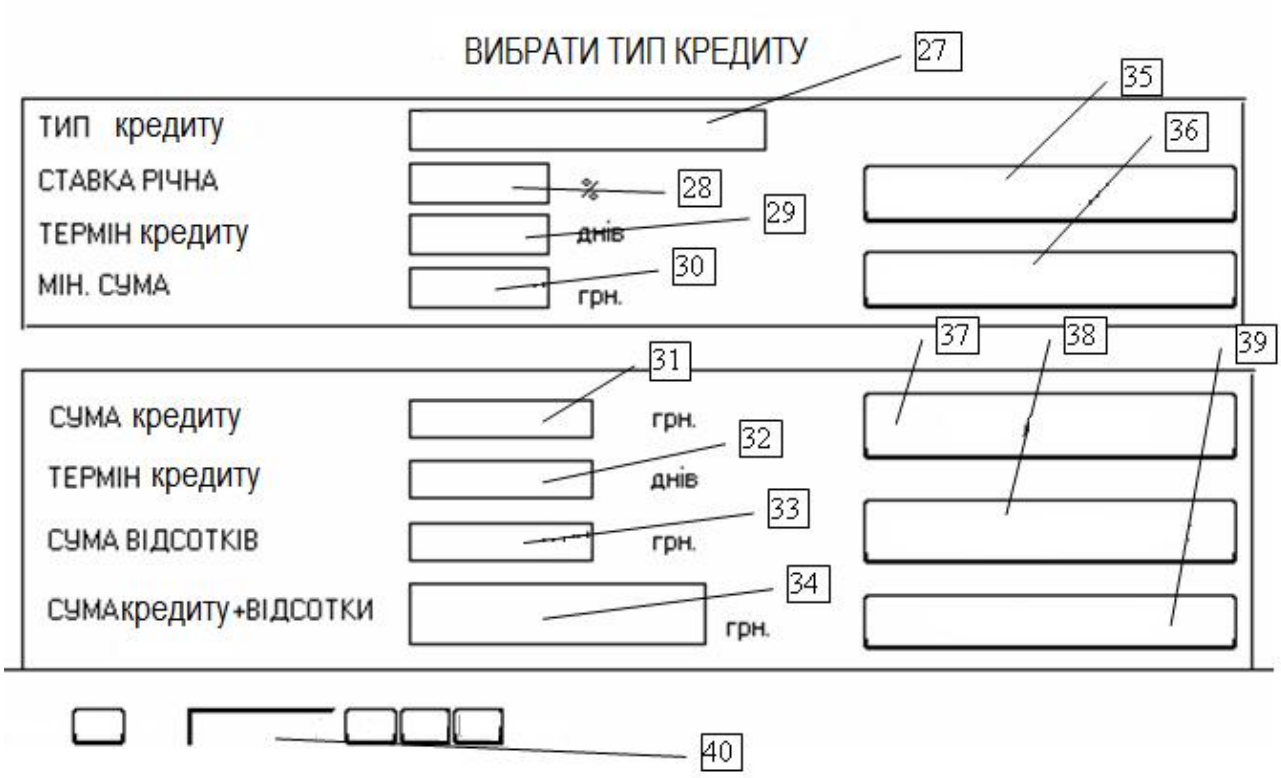


Рисунок 1.5.- Розкадровка варіанту використання «Вибрати тип кредиту»

Графічним елементам діалогового вікна (рисунок 1.5) пропонується надати наступні функції:

27. введення та вибір кредитного рахунку;
28. - введення та відображення річної ставки вибраного кредитного рахунку;
29. - введення та відображення терміну кредиту;
30. – мінімальна грошова сума кредиту;
31. –сума кредиту;
32. -відображення терміну кредиту;
33. -розраховується та відображається сума за відсотками вибраного кредиту;
34. - розраховується та відображається сума за відсотками плюс сума вкладу вибраного кредиту;
35. – навігація по списку кредитних рахунків;
36. – навігація по списку кредитів назад;

37.– розрахунок суми за відсотками та загальної суми;

38. – перехід до форми для введення інформації про клієнта;

39.-закриття ділового вікна;

40.– група кнопок для додавання, видалення, редагування записів в базі даних;

Розкадровка варіанту використання «Збереження даних про кредит» показана на рисунку 1.6.

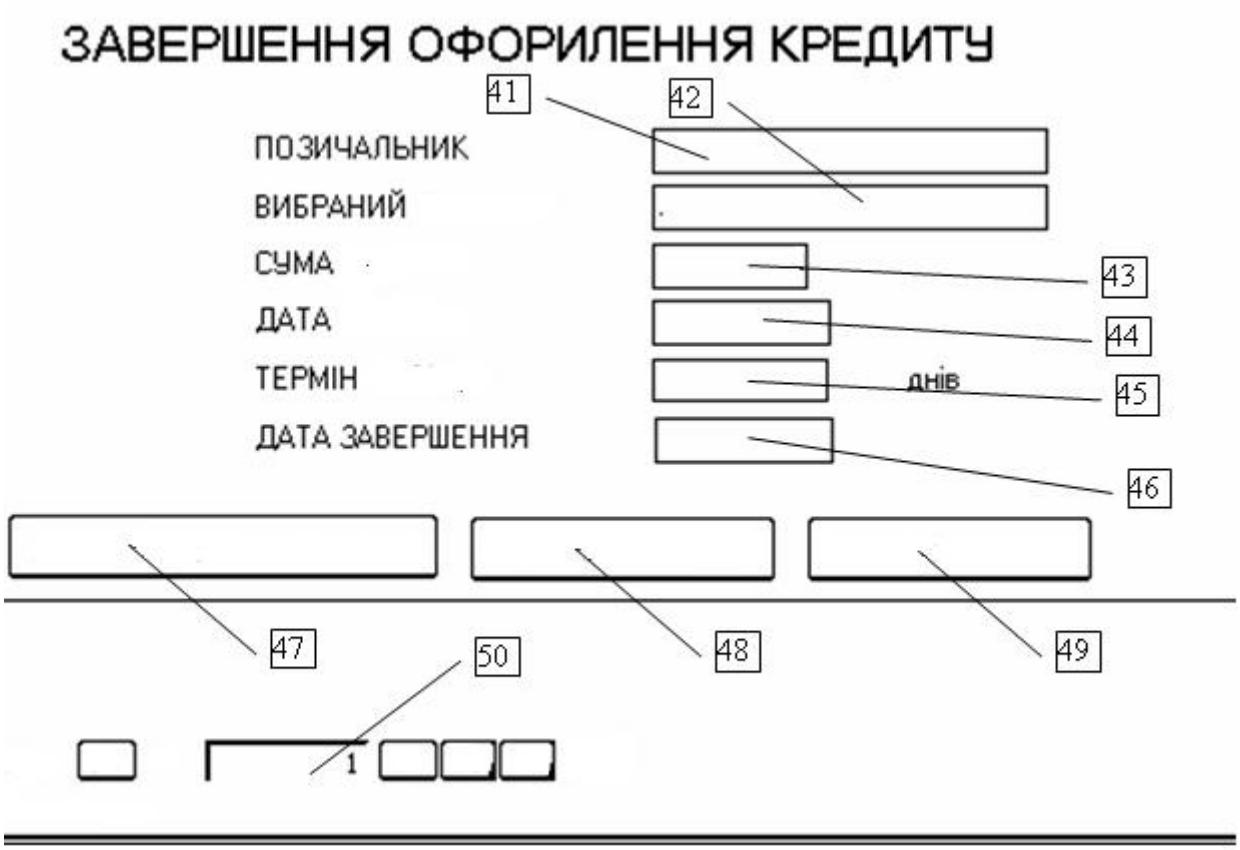


Рисунок 1.6.- Розкадровка варіанту використання «Збереження даних про кредит»

41.- відображення прізвища клієнта;

42.- відображення вибраного кредитного рахунку;

43.- відображення суми кредитного вкладу;

44.- введення та відображення дати початку кредитного договору;

45.- відображення терміну дії кредитного договору;

46.-розрахунок та відображення дати закінчення кредитного договору;

47.–повернення до діалогового вікна для введення інформації про клієнта;

48.–збереження інформації про кредит;

49.-закриття ділового вікна;

50.–група кнопок для додавання, видалення, редагування записів в базі даних;

В результаті опису отримуємо специфікацію функціональних та нефункціональних вимог відповідно таблиця 1.6 та таблиця 1.7.

Таблиця 1.6

Специфікацію функціональних вимог

| Іденти-фікатор вимог | Назва вимоги | Атрибути вимоги | | |
|---|---|---|---|---|
| | | Пріоритет | Складність | Контакт |
| 1 | Авторизація | Обов’язкова | Середня | |
| 2 | Ввести інформацію про клієнта | Обов’язкова | Середня | |
| 3 | Вибрати тип кредиту | Обов’язкова | Висока | |
| 4 | Збереження даних про кредит | Обов’язкова | Висока | |

Таблиця 1.7

Специфікацію нефункціональних вимог

| Іденти-фікатор вимог | Назва вимоги | Атрибути вимоги | | |
|---|---|---|---|---|
| | | Пріоритет | Складність | Контакт |
| Застосовність | | | | |
| 1 | Авторизація | Обов’язкова | 5 | |
| 2 | Ввести інформацію про клієнта | Обов’язкова | 5 | |
| 3 | Вибрати тип кредиту | Обов’язкова | 30 | |
| 4 | Збереження даних про кредит | Обов’язкова | 45 | |
| Надійність | | | | |
| 1 | Авторизація | Обов’язкова | 99 | |
| 2 | Ввести інформацію про клієнта | Обов’язкова | 98 | |
| 3 | Вибрати тип кредиту | Обов’язкова | 98 | |
| 4 | Збереження даних про кредит | Обов’язкова | 98 | |
| Експлуатаційна придатність | | | | |
| 1 | Авторизація | Обов’язкова | 100 | |
| 2 | Ввести інформацію про клієнта | Обов’язкова | 100 | |
| 3 | Вибрати тип кредиту | Обов’язкова | 100 | |
| 4 | Збереження даних про кредит | Обов’язкова | 100 | |

На підставі проведеного аналізу методів управління і оцінки кредитного ризику, розроблених проектів, необхідно створити систему підтримки прийняття рішень, що реалізує основні ключові етапи в управлінні кредитним ризикомідентифікація, оцінка.

Проектована система повинна виконувати наступні функції:

1) в зручній формі здійснювати введення вихідної інформації;

2) виконувати розрахунок підсумкових показників, необхідних для якісної оцінки ризику;

3) значно економити інформаційні ресурси на технічних засобах за рахунок виключення дублювання даних ;

4) забезпечити санкціонований доступ до необхідної інформації;

6) проводити постійний моніторинг.

## Висновки до першого розділу

Проведено огляд існуючих напрацювань в області вибору оптимальних варіантів кредитного договору. Базуючись на цьому було поставлено завдання для створення власного програмного засобу, визначено варіати використання програми та висунуто функціональні та нефункціональні вимого.

# РОЗДІЛ ІІ
# ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ВИЗНАЧЕННЯ ОПТИМАЛЬНИХ ПАРАМЕТРІВ КРЕДИТУ

## 2.1. Розроблення архітектури модуля програмної системи

На основі запроектованих діалогових форм створюємо наступні діалогові форми (рисунки 2.1-2.8).

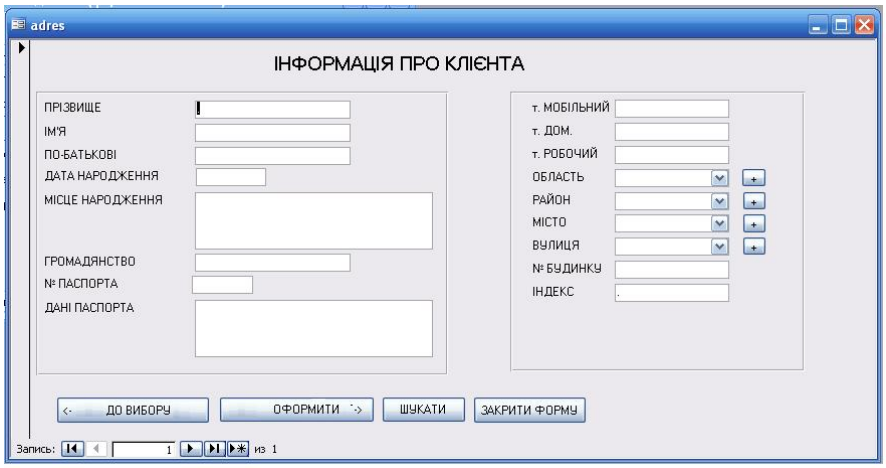


Рисунок 2.1 – Діалогове вікно для роботи з інформацією про клієнта

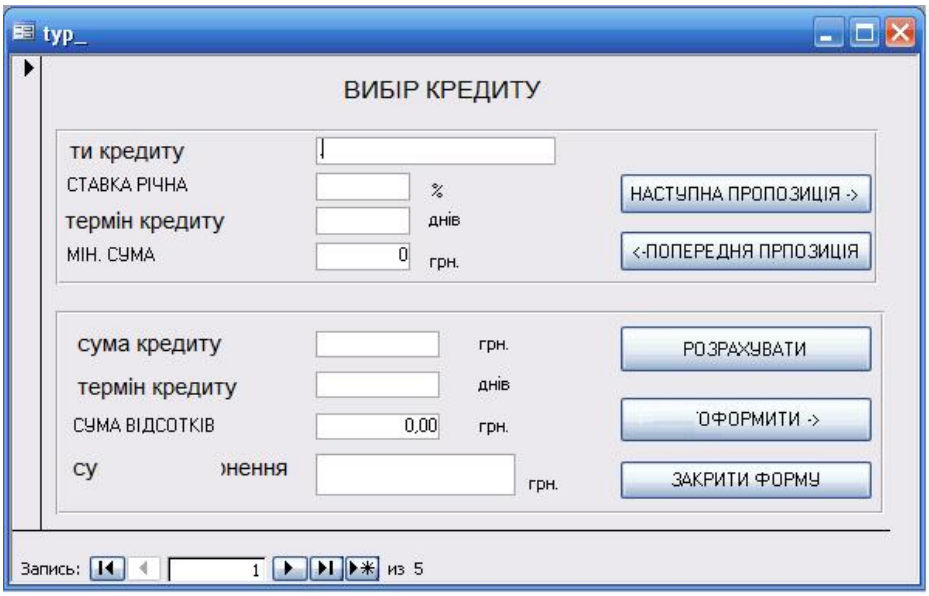


Рисунок 2.2 – Діалогове вікно для вибору кредиту

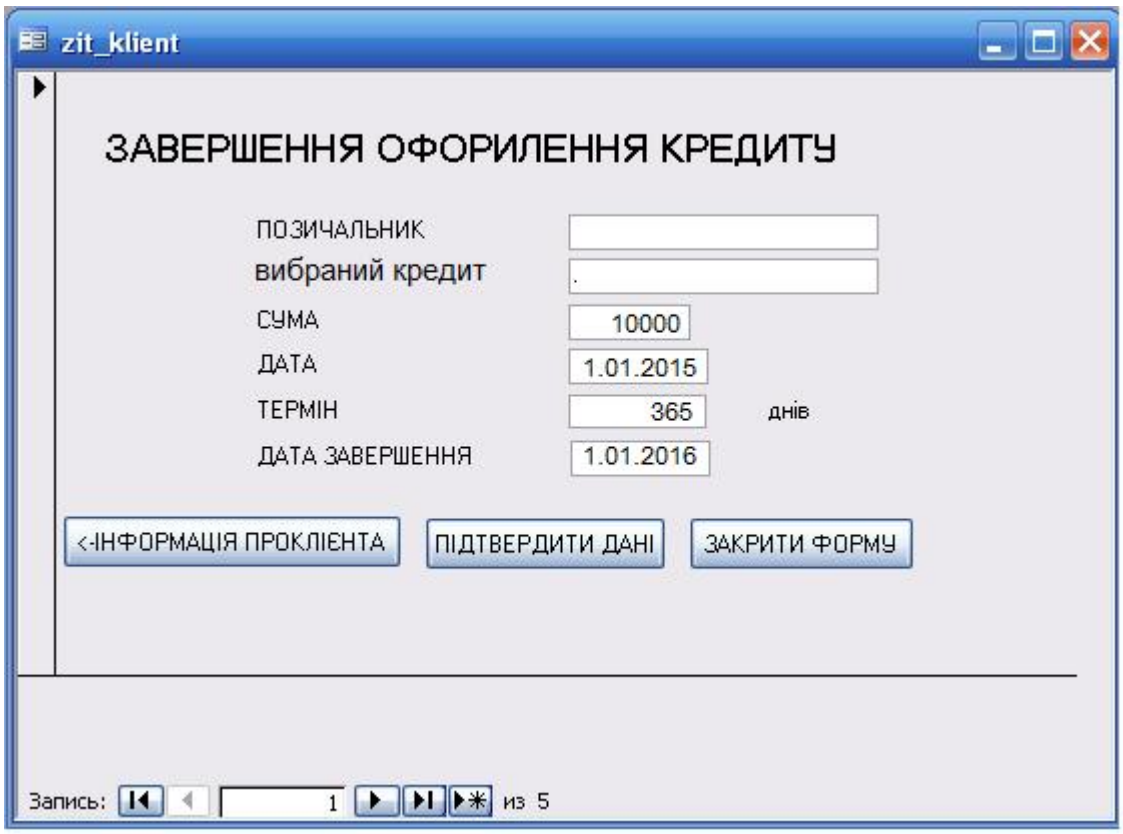


Рисунок 2.3 – Проект діалогового вікна для завершення оформлення кредиту

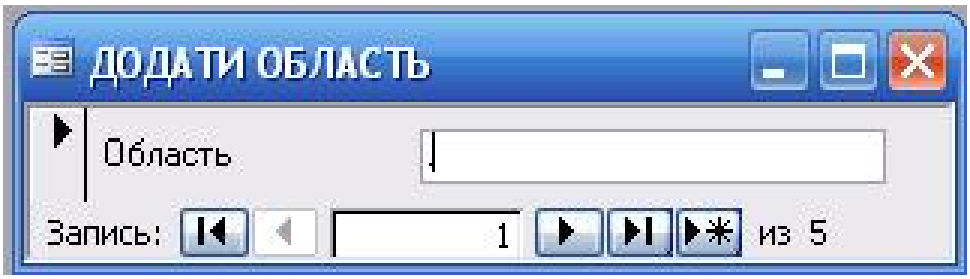


Рисунок 2.4 – Діалогове вікно для додавання нової області


Рисунок 2.5 – Діалогове вікно для додавання нового району


Рисунок 2.6 – Діалогове вікно для додавання нового району


Рисунок 2.7 – Діалогове вікно для додавання нової вулиці

Програмне забезпечення складається з окремих модулів, які прив’язуються до певних конкретних діалогових форм.

Назви модулів відповідають назвам діалогових форм дивись малюнок 1.29.

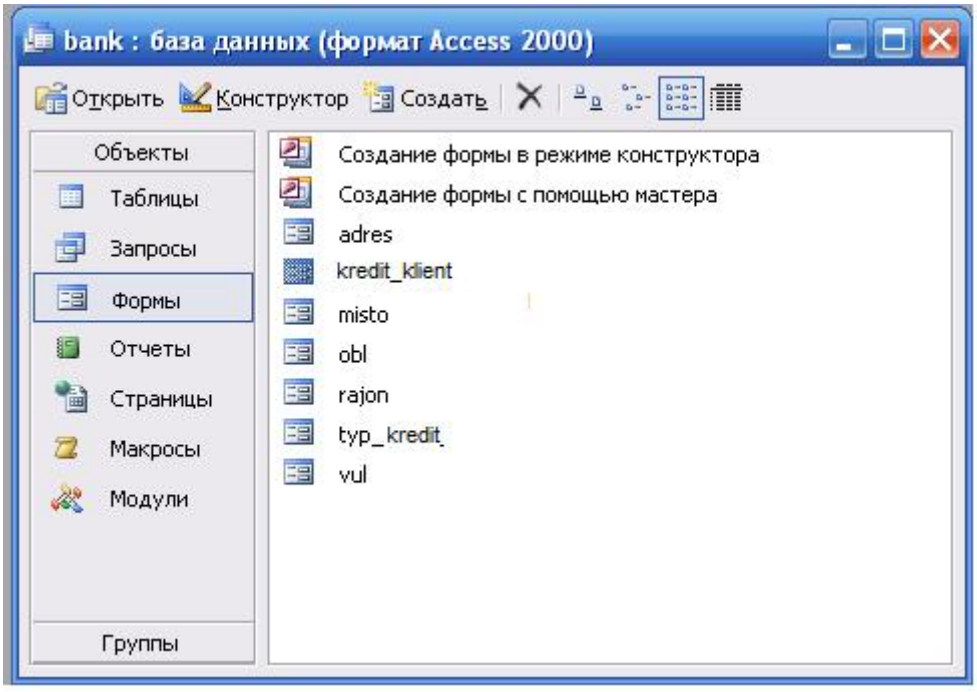


Рисунок 2.8 – Назви модулів програми

### 2.1.1 Функції частин програмного забезпечення.

Опишемо функції часин програмного забезпечення в контексті діалогових форм до яких ці модулі відносяться в таблиці 2.1.

Таблиця 2.1 – Функції модулів програми

| Номер графічного елемента | Функція графічного елемента | Подія, яка визиває реакцію елемента | Тип елемента | Примітка |
|---|---|---|---|---|
| Модуль Form_adres | | | | |
| 1. | Записує введене значення в поле prizvyche таблиці klient | Після обновлення | Поле вводу | |
| 2. | Записує введене значення в поле imja таблиці klient | Після обновлення | Поле вводу | |
| 3. | Записує введене значення в поле po_bat таблиці klient | Після обновлення | Поле вводу | |
| 4. | Записує введене значення в поле misz_narod таблиці klient | Після обновлення | Поле вводу | |
| 5. | Записує введене значення в поле data_narod таблиці klient | Після обновлення | Поле вводу | |
| 6. | Записує введене значення в поле gromad таблиці klient | Після обновлення | Поле вводу | |
| 7. | Записує введене значення в поле N_ pasport таблиці klient | Після обновлення | Поле вводу | |
| 8. | Записує введене значення в поле dani_pasport таблиці klient | Після обновлення | Поле вводу | |
| 9. | Записує введене значення в поле tel_mob таблиці klient | Після обновлення | Поле вводу | |
| 10. | Записує введене значення в | Після | Поле | |

| | | | | |
|---|---|---|---|---|
| | поле tel_dom таблиці klient | обновлення | вводу | |
| 11. | Записує введене значення в поле tel_rob таблиці klient | Після обновлення | Поле вводу | |
| 12. | Записує введене значення в поле obl таблиці adres | Після обновлення | Поле вводу | |
| 13. | Записує введене значення в поле rajon dтаблиці adres | Після обновлення | Поле вводу | |
| 14. | Записує введене значення в поле misto таблиці adres | Після обновлення | Поле вводу | |
| 15. | Записує введене значення в поле vul таблиці adres | Після обновлення | Поле вводу | |
| 16. | Записує введене значення в поле bud таблиці adres | Після обновлення | Поле вводу | |
| 17. | Записує введене значення в поле index таблиці adres | Після обновлення | Поле вводу | |
| 18. | Запускає форму obl | Натискання лівою кнопкою миші | Кнопка | |
| 19. | Запускає форму rojon | Натискання лівою кнопкою миші | Кнопка | |
| 20. | Запускає форму misto | Натискання лівою кнопкою миші | Кнопка | |
| 21. | Запускає форму vul | Натискання лівою кнопкою миші | Кнопка | |
| 22. | Запускає форму typ_kredit | Натискання лівою кнопкою | Кнопка | |

| | | | | |
|---|---|---|---|---|
| | | миші | | |
| 23. | Запускає форму kredit_klient | Натискання лівою кнопкою миші | Кнопка | |
| 24. | Запускає форму пошуку | Натискання лівою кнопкою миші | Кнопка | |
| 25. | Закриває форму | Натискання лівою кнопкою миші | Кнопка | |
| 26. | Створює, знищує, редагує записи | Натискання лівою кнопкою миші | Кнопки | |

| Модуль Form_typ_kredit | | | | |
|---|---|---|---|---|
| 27. | Записує введене значення в поле typ_kredit таблиці typ_kredit | Після обновлення | Поле вводу | |
| 28. | Записує введене значення в поле stavka таблиці typ_kredit | Після обновлення | Поле вводу | |
| 29. | Записує введене значення в поле min_term таблиці typ_kredit | Після обновлення | Поле вводу | |
| 30. | Записує введене значення в поле min_sum таблиці typ_kredit | Після обновлення | Поле вводу | |
| 31. | Записує введене значення в поле suma таблиці kredit_klient | Після обновлення | Поле вводу | |

| | | | | |
|---|---|---|---|---|
| 32. | Копіює значення з поля №29 | Автоматично | Поле вводу | |
| 33. | Підраховується сума за відсотками | Автоматично | Поле вводу | |
| 34. | Підраховується сума за відсотками плюс вклад | Автоматично | Поле вводу | |
| 35. | Переходить до попереднього запит в базі даних | Натискання лівою кнопкою миші | Кнопка | |
| 36. | Переходить до наступного запит в базі даних | Натискання лівою кнопкою миші | Кнопка | |
| 37. | Здійснює розрахунок | Натискання лівою кнопкою миші | Кнопка | |
| 38. | Запускає форму адрес | Натискання лівою кнопкою миші | Кнопка | |
| 39. | Закриває форму | Натискання лівою кнопкою миші | Кнопка | |
| 40. | Створює, знищує, редагує записи | Натискання лівою кнопкою миші | Кнопки | |
| | | | | |
| 41. | Відображається прізвище клієнта | Автоматично | Поле вводу | |
| 42. | Відображається кредит | Автоматично | Поле вводу | |

| | | | | |
|---|---|---|---|---|
| 43. | Відображається сума вкладу | Автоматично | Поле вводу | |
| 44. | Вводиться дата вкладу, яка записується поле data_p таблиці depozit_klient | Автоматично, Після обновлення | Поле вводу | |
| 45. | Відображається сума вкладу | Автоматично, | Поле вводу | |
| 46. | Розраховується дата повернення, яка записується поле data_k таблиці depozit_klient | Автоматично, Після обновлення | Поле вводу | |
| 47. | Запускає форму адрес | Натискання лівою кнопкою миші | Кнопка | |
| 48. | Зберігає інформацію про кредит | Натискання лівою кнопкою миші | Кнопка | |
| 49. | Закриває форму | Натискання лівою кнопкою миші | Кнопка | |
| 50. | Створює, знищує, редагує записи | Натискання лівою кнопкою миші | Кнопки | |
| | | | | |
| 51. | – Вводить в довідник нового елемента (область, район, місто, вулиця) і записує в таблицю | Після обновлення | Поле вводу | |

| | | | | |
|---|---|---|---|---|
| 52. | Створює, знищує, редагує записи | Натискання лівою кнопкою миші | Кнопки | |

## 2.2. Проектування структури бази даних

Проектуємо архітектуру баз даних (БД). Всі архітектури представляються нотацією UML і при потребі, доповнюються текстовими описами.

Розглянута концепція розподілених баз даних знаходить широке розповсюдження при функціонуванні розподіленої обробки інформації. При цьому організація такої обробки інформації відбувається, як правило, в рамках системної автоматизованої обробки економічної інформації як в окремих структурних ланках, так і по об'єкту управління в цілому.

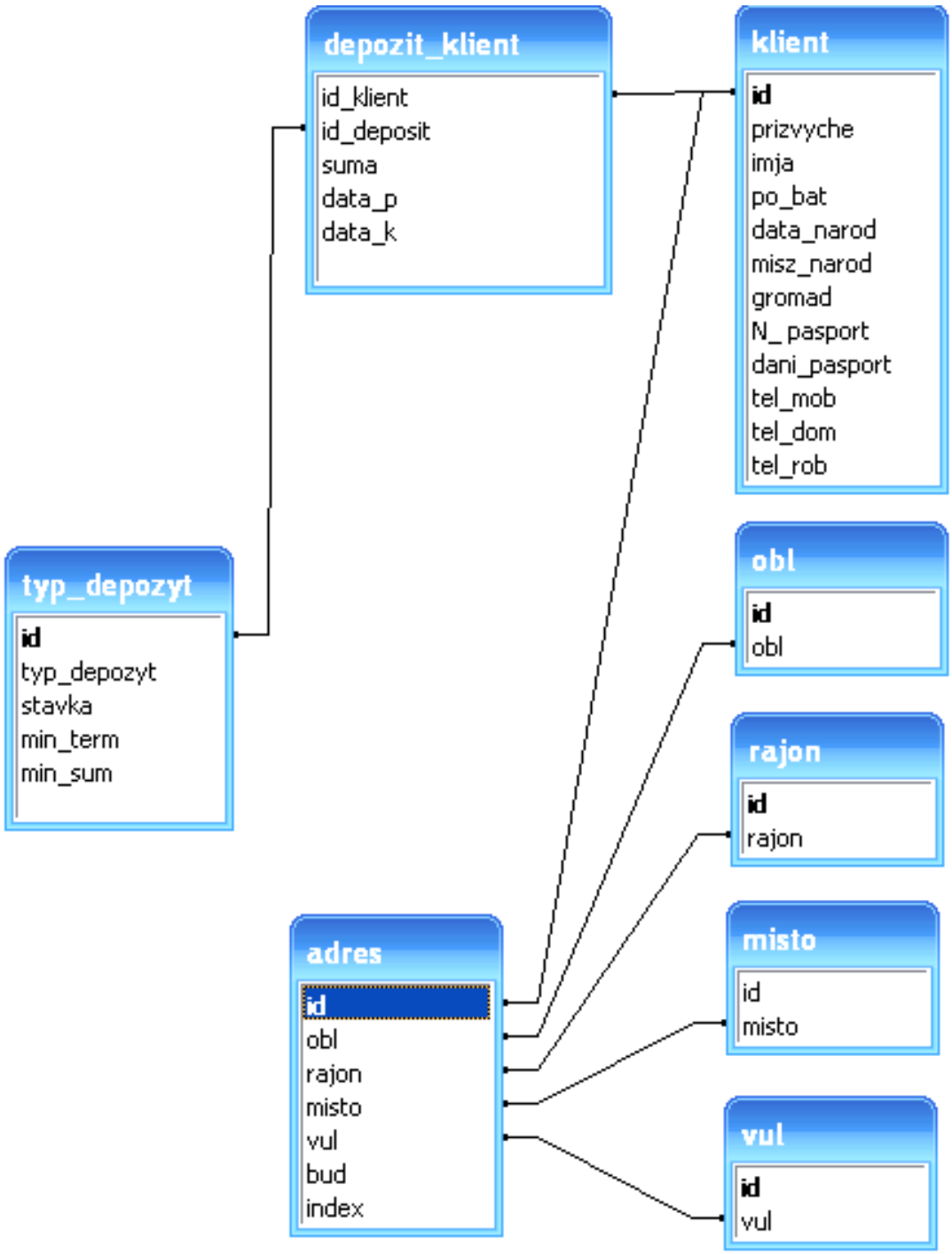


Рисунок 2.9 – Схема зв’язків таблиць БД

Часто зустрічається характеристика БД на основі певних параметрів або необхідних вимог, наприклад:

- значна кількість даних;
- незалежність даних;

- відкритий доступ до даних;
- підтримка трансакцій з гарантією відповідних властивостей;
- гарантована відсутність збоїв;
- одночасна робота з багатьма користувачами.

З подальшим розвитком БД змінюються ці вимоги та додаються нові, тому одностайності щодо повноти цієї характеристики немає.

Далі приведемо структуру баз даних, що використовуватиметься в процесі розробки програмного продукту.

| | | |
|---|---|---|
| id | Счетчик | ЛІЧИЛЬНИК |
| prizvyche | Текстовый | ПРІЗВИЩЕ ПОЗИЧАЛЬНИКА |
| imja | Текстовый | ІМ'Я ПОЗИЧАЛЬНИКА |
| po_bat | Текстовый | ПО-БАТЬКОВІ ПОЗИЧАЛЬНИКА |
| data_narod | Дата/время | ДАТА НАРОДЖЕННЯ ПОЗИЧАЛЬНИКА |
| misz_narod | Поле МЕМО | МІСЦЕ НАРОДЖЕННЯ ПОЗИЧАЛЬНИКА |
| gromad | Текстовый | ГРОМАДЯНСТВО ПОЗИЧАЛЬНИКА |
| N_pasport | Числовой | НОМЕР ПАСПОРТА ПОЗИЧАЛЬНИКА |
| dani_pasport | Поле МЕМО | ДАНІ ПАСПОРТА ПОЗИЧАЛЬНИКА |
| tel_mob | Текстовый | МОБІЛЬНИЙ ТЕЛЕФОН |
| tel_dom | Текстовый | ДОМАШНІЙ ТЕЛЕФОН |
| tel_rob | Текстовый | РОБОЧИЙ ТЕЛЕФОН |

Рисунок 2.10 – Проект таблиці БД для опису клієнта банку

| Имя поля | Тип данных | |
|---|---|---|
| id | Счетчик | ЛІЧИЛЬНИК |
| obl | Числовой | ОБЛАСТЬ ПРОЖИВАННЯ ПОЗИЧАЛЬНИКА |
| rajon | Числовой | РАЙОН ПРОЖИВАННЯ ПОЗИЧАЛЬНИКА |
| misto | Числовой | МІСТО ПРОЖИВАННЯ ПОЗИЧАЛЬНИКА |
| vul | Числовой | ВІЛИЦЯ ПРОЖИВАННЯ ПОЗИЧАЛЬНИКА |
| bud | Текстовый | БУДИНОК ПРОЖИВАННЯ ПОЗИЧАЛЬНИКА |
| index | Текстовый | ІНДЕКС ПРОЖИВАННЯ ПОЗИЧАЛЬНИКА |

Рисунок 2.11 – Проект таблиці БД для опису місця проживання клієнта банку

| Имя поля | Тип данных | |
|---|---|---|
| id | Счетчик | Лічильник |
| typ_depozyt | Текстовый | Назва депозиту |
| stavka | Числовой | Річна ставка |
| min_term | Числовой | термін депозиту |
| min_sum | Числовой | мінімальна сума депозиту |
| | | |

Рисунок 2.12 – Проект таблиці БД для опису характеристик кредиту

| Имя поля | Тип данных | |
|---|---|---|
| id_klient | Текстовый | КОД КЛІЄНТА |
| id_deposit | Текстовый | КОД ДЕПОЗИТУ |
| suma | Числовой | СУМА ВКЛАДУ |
| data_p | Дата/время | ДАТА ПОЧАТКУ ВКЛАДУ |
| data_k | Дата/время | ДАТА ЗАКІНЧЕННЯ ВКЛАДУ |
| | | |
| | | |

Рисунок 2.13 – Проект таблиці БД для опису зв’язку клієнту банку з вибраним кредитом

| Имя поля | Тип данных | |
|---|---|---|
| id | Счетчик | ЛІЧИЛЬНИК |
| obl | Текстовый | НАЗВА ОБЛАСТІ |
| | | |
| | | |

Рисунок 2.14 – Проект таблиці БД (довідник) для зберігання назв областей

| Имя поля | Тип данных | |
|---|---|---|
| id | Счетчик | ЛІЧИЛЬНИК |
| rajon | Текстовый | НАЗВА РАЙОНУ |
| | | |

Рисунок 2.15 – Проект таблиці БД (довідник) для зберігання назв районів

| Имя поля | Тип данных | |
|---|---|---|
| id | Счетчик | ЛІЧИЛЬНИК |
| misto | Текстовый | НАЗВА МІСТА |
| | | |

Рисунок 2.16 – Проект таблиці БД (довідник) для зберігання назв міст

| Имя поля | Тип данных | |
|---|---|---|
| id | Счетчик | ЛІЧИЛЬНИК |
| vul | Текстовый | НАЗВА ВУЛИЦІ |
| | | |
| | | |
| | | |

Рисунок 2.17 – Проект таблиці БД (довідник) для зберігання назв вулиць

Виходячи з запроектованих таблиць за допомогою системи управління базою даних була фізично створена база даних зовнішній вигляд таблиць якої показано на рисунку 2.18-2.25.

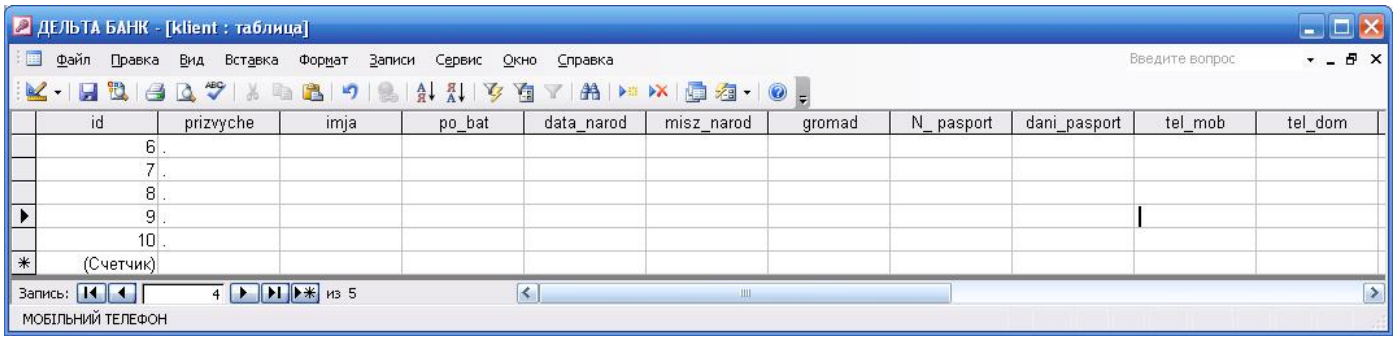


Рисунок 2.18 – Таблиця БД для опису клієнта банку

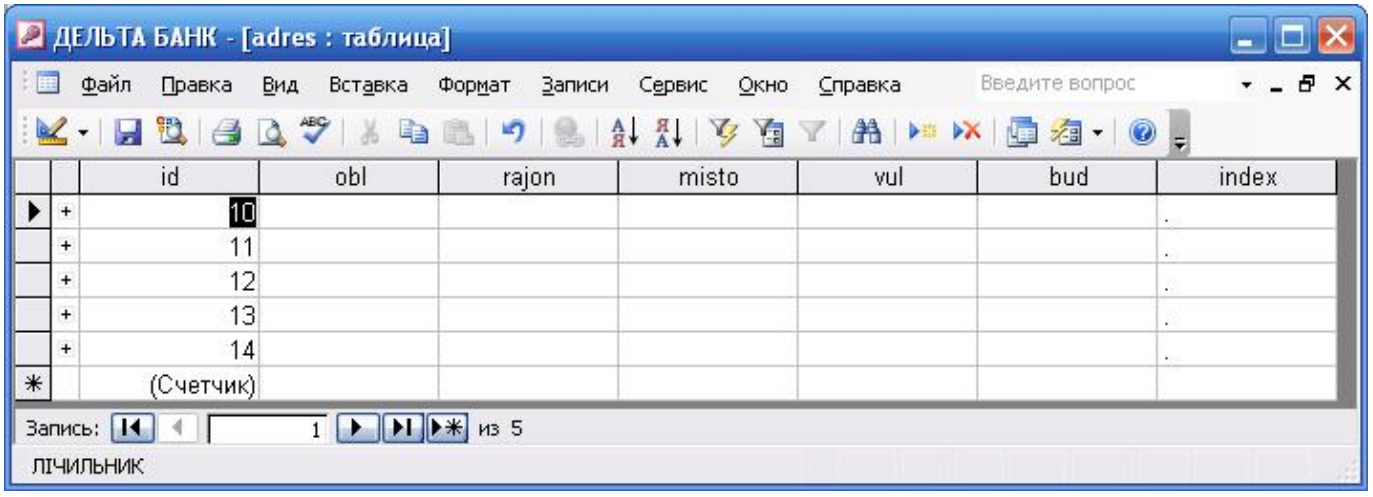


Рисунок 2.19 – Таблиця БД для опису місця проживання клієнта банку

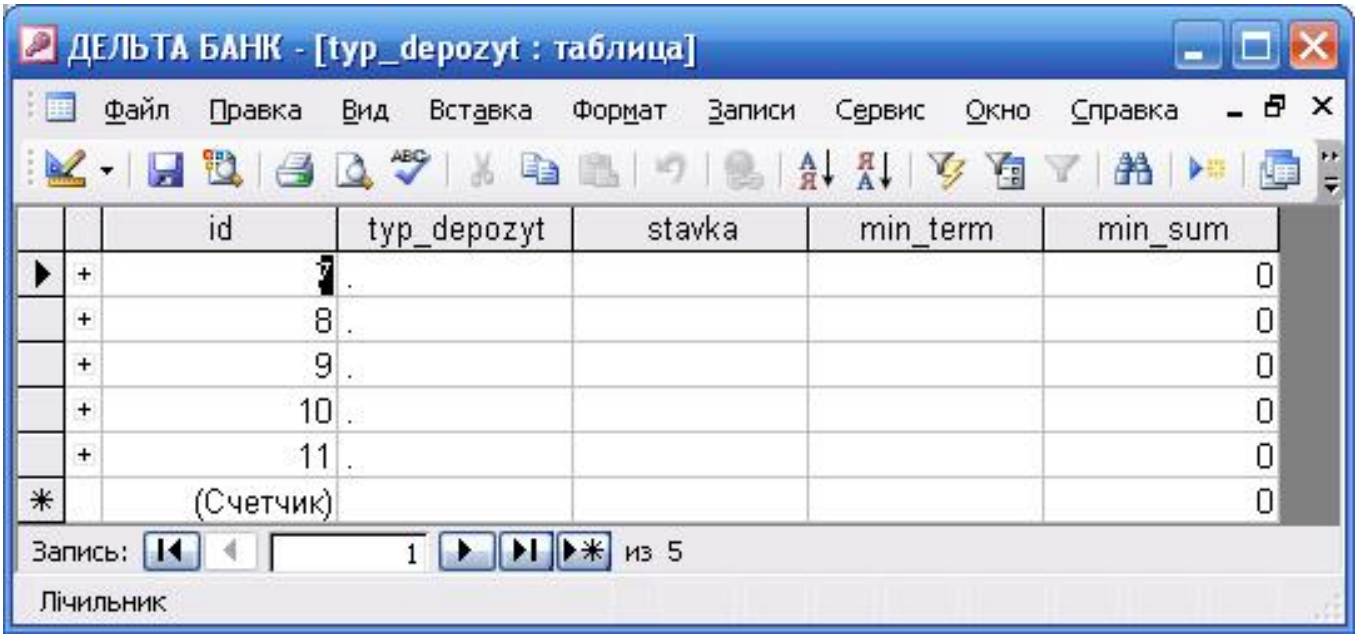


Рисунок 2.20 – Таблиця БД для опису характеристик кредиту

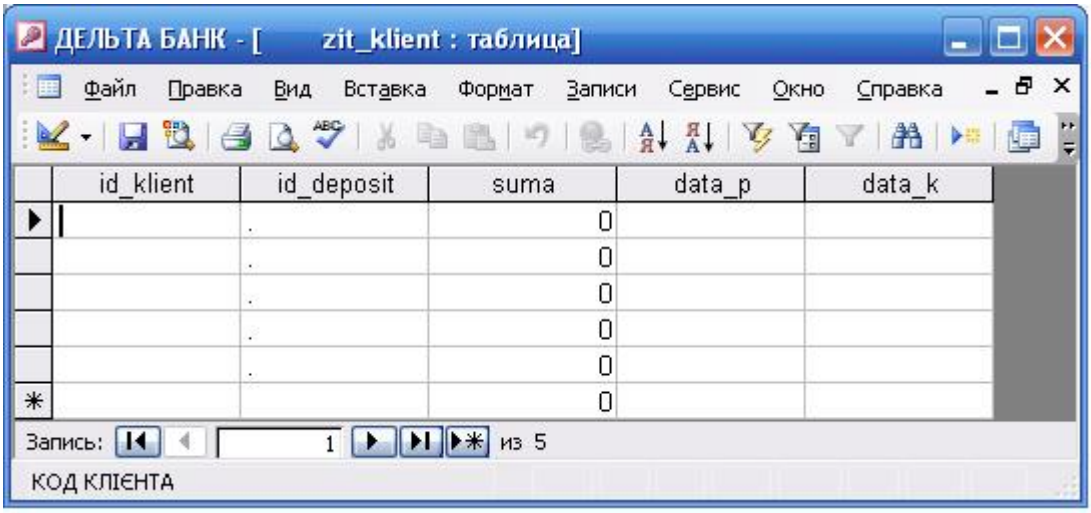


Рисунок 2.21 – Таблиця БД для опису зв’язку клієнту банку з вибраним кредитом

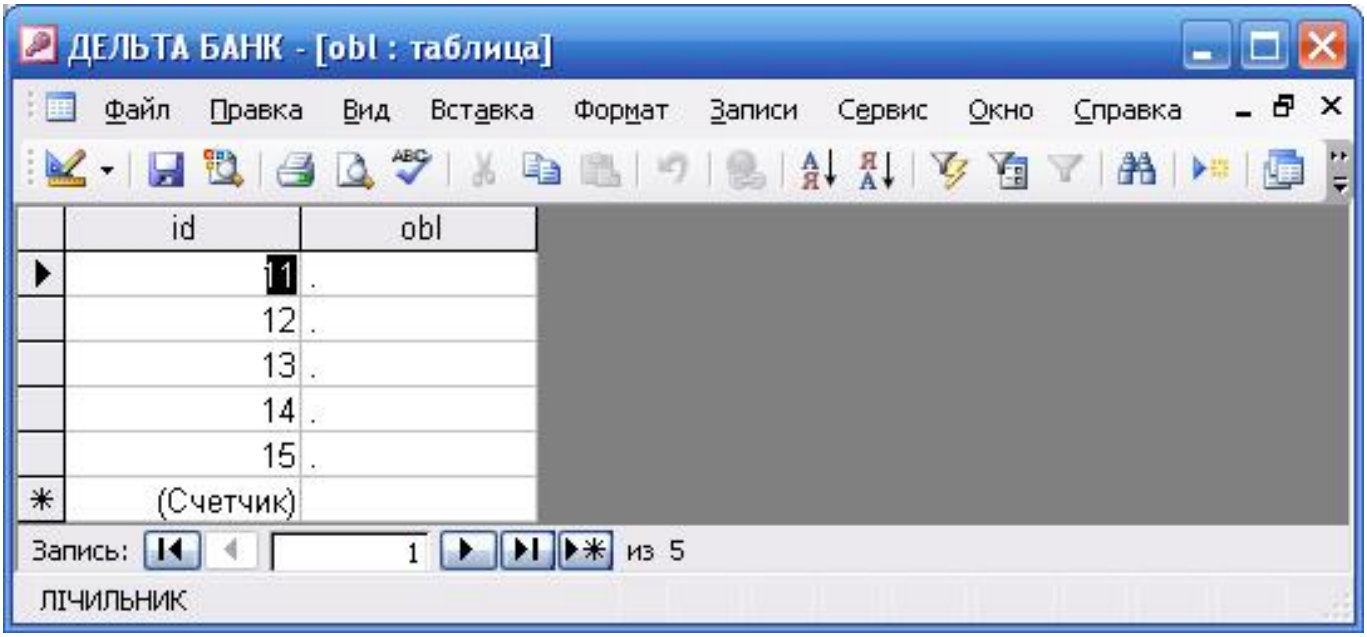


Рисунок 2.22 – Таблиця БД (довідник) для зберігання назв областей

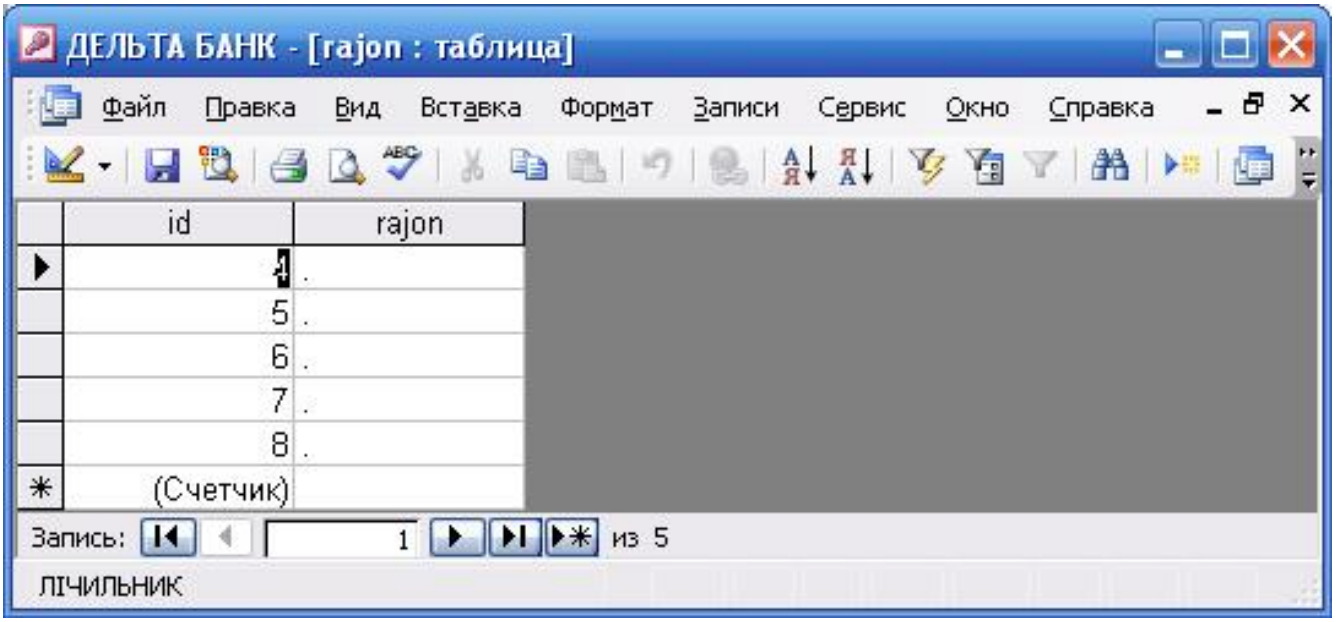


Рисунок 2.23 – Таблиця БД (довідник) для зберігання назв районів

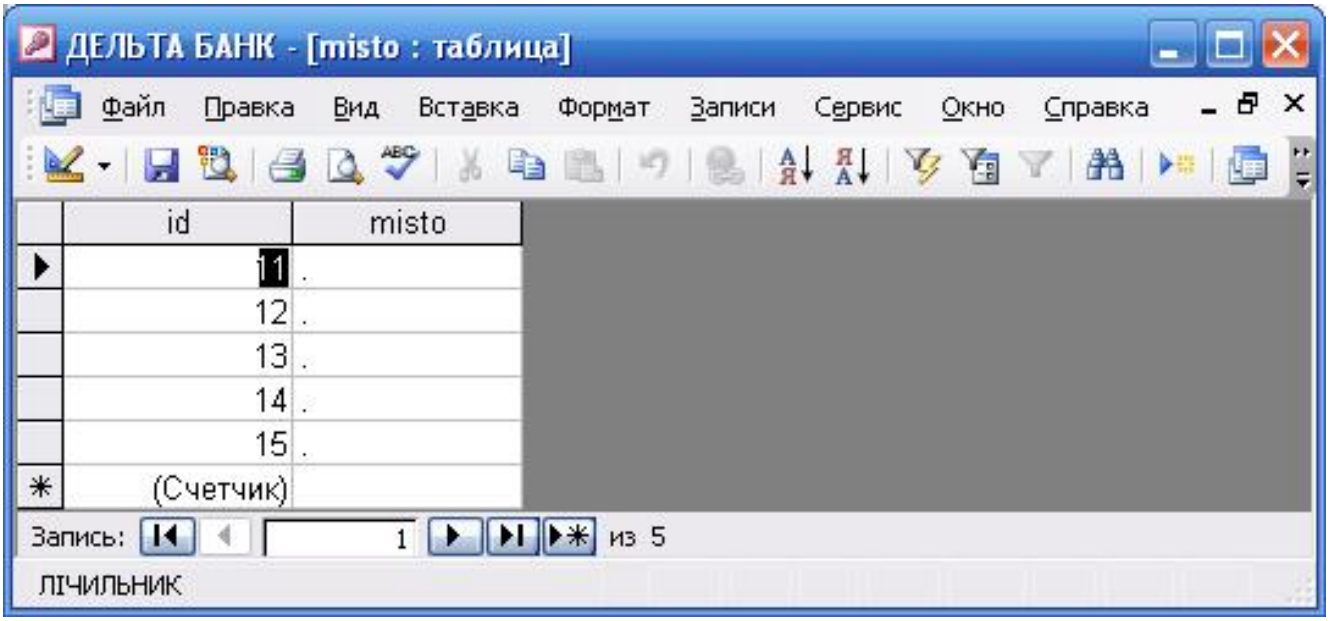


Рисунок 2.24 – Таблиця БД (довідник) для зберігання назв міст

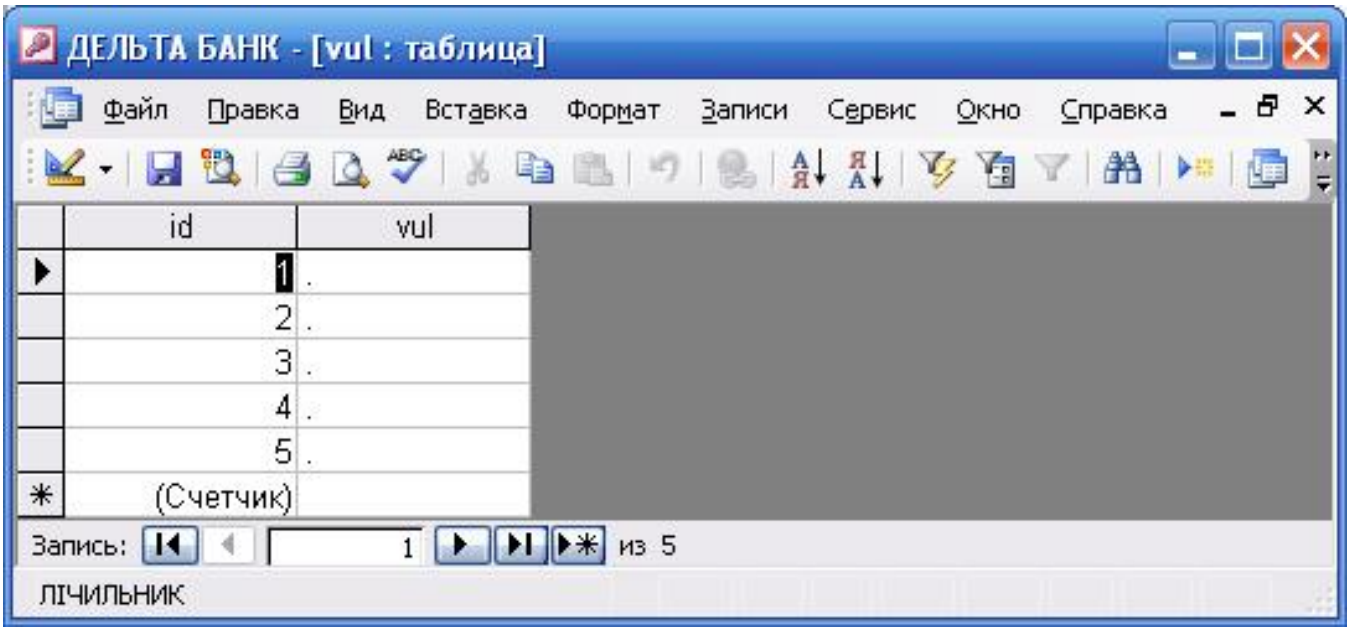


Рисунок 2.25 – Таблиця БД (довідник) для зберігання назв вулиць

## Висновки до другого розділу

В другому розділі запроектовано модулі програмної системи для підтримки прийняття рішень при виданні кредиту, також запроектовано до неї база даних. Проектування здійснювалося з використанням об’єктно-орієнтованого підходу до проектування, а також використовувалася UML діаграми для нотацій.

# РОЗДІЛ III
# ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ ВИЗНАЧЕННЯ ОПТИМАЛЬНИХ ПАРАМЕТРІВ КРЕДИТУ

## 3.1. Програмна реалізація проекту

### 3.1.1. Обґрунтування вибору мови програмування

. Для створення єдиного інформаційного простору обрані продукти Microsoft :

Microsoft SQL Server 2008 Express Edition with Advanced Services (розробка і управління базою даних);

Visual Studio 2008 Express Edition (розробка клієнтських додатків).

Microsoft SQL Server 2008 Express Edition with Advanced Services - це безкоштовна редакція, яка має ряд обмежень для великих проектів, але підтримує функції професійних СУБД:

ядро бази даних SQL Server: створення, зберігання, оновлення та вилучення даних;

Середа SQL Server Management Studio Basic: візуальний засіб для створення, редагування і управління базами даних;

повнотекстовий пошук: потужне, високошвидкісне ядро для пошуку даних у великих обсягах тексту;

служби звітів (Reporting Services): вбудоване створення звітів і середовище їх конструювання.

Вибір Visual Studio 2008 Express Edition обумовлений такими факторами:

тісна інтеграція з Microsoft SQL Server 2008;

наявність об'єктно-орієнтованої мови високого рівня (C # 3.5);

наявність ефективного візуального середовища розробки, що дозволяє в короткі терміни розробляти програмне забезпечення;

наявність сучасних технологій, що дозволяють з легкістю вирішувати поставлені завдання роботи з даними;

безкоштовне застосування середовища розробки.

Темою дипломної роботи є, «Програма для розрахунку оптимальних параметрів споживацького кредиту», тому засіб розробки має мати сучасний інтерфейс та бути розроблений з використанням сучасних засобів конструювання та програмування.

Microsoft .NET — програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Одною з ідей .NET є сумісність служб, написаних різними мовами. Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість.

Кожна бібліотека (збірка) в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

.NET — крос-платформова технологія, в цей час існує реалізація для платформи Microsoft Windows, FreeBSD(від Microsoft) і варіант технології для ОС Linux в проекті Mono (в рамках угоди між Microsoft з Novell), DotGNU[1].

Захист авторських прав відноситься до створення середовищ виконання (CLR — Common Language Runtime) для програм .NET. Компілятори для .NET випускаються багатьма фірмами для різних мов вільно.

.NET поділяється на дві основні частини — середовище виконання (по суті віртуальна машина) та інструментарій розробки.

Середовища розробки .NET-програм: Visual Studio .NET (C++, C#, J#), SharpDevelop, Borland Developer Studio (Delphi, C#) тощо. Середовище Eclipse має додаток для розробки .NET-програм. Застосовні програми також можна

розроблювати в текстовому редакторі та використовувати консольний компілятор.

Як і технологія Java, середовище розробки .NET створює байт-код, призначений для виконання віртуальною машиною. Вхідна мова цієї машини в .NET називається CIL (Common Intermediate Language), також відома як MSIL (Microsoft Intermediate Language), або просто IL. Застосування байт-кода дозволяє отримати крос-платформовість на рівні скомпільованого проекту (в термінах .NET: збірка), а не на рівні сирцевого тексту, як, наприклад, в C. Перед запуском збірки в середовищі виконання (CLR) байт-код перетворюється вбудованим в середовище JIT-компілятором (just in time, компіляція на льоту) вмашинні коди цільового процесора.

Слід зазначити, що один з перших JIT-компіляторів для Java був також розроблений фірмою Microsoft (тепер в Java використовується досконаліша багаторівнева компіляція — Sun HotSpot). Сучасна технологія динамічної компіляції дозволяє досягнути аналогічного рівня швидкодії з традиційними «статичними» компіляторами (наприклад, C++) і питання швидкодії часто залежить від якості того чи іншого компілятора.

На рисунку 3.1 відображене стартове вікно для створення аплікації за допомогою Microsoft .NET.

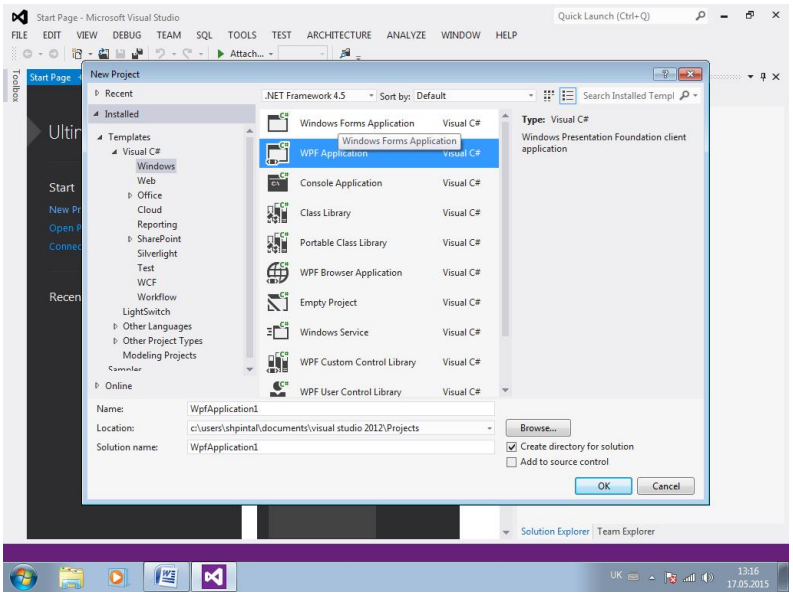


Рисунок 3.1- Вибір аплікації в діалоговому вікні нового проекту

## 3.2. . Програмна реалізація бази даних

### 3.2.1. Реалізація архітектури баз даних

Визначивши та описавши архітектуру баз даних, її було реалізовано у середовищі керування базами даних MS SQL Server 2008, дивись рисунок 3.8.

Установка Data Developer Center

Все необхідне для установки можна знайти на сторінці завантаження в Data Developer Center. Вибравши необхідну версію ви зможете без праці встановити інструменти на свій комп'ютер і описувати це не бачу сенсу.Після установки в вікні створення нового проекту у вас з'явиться новий тип проекту: Створивши новий проект ви побачите наступне (рисунок 3.1) : На панелі SQL Server Object Explorer (меню View -> SQL Server Object Explorer) ми бачимо щось дуже схоже на Object Explorer в SQL Server Management Studio , з якого прибрано все, що не має великого сенсу на етапі розробки бази даних. Підключившись до існуючої бази, можна виробляти розробку бази даних в так званому Connected режимі.

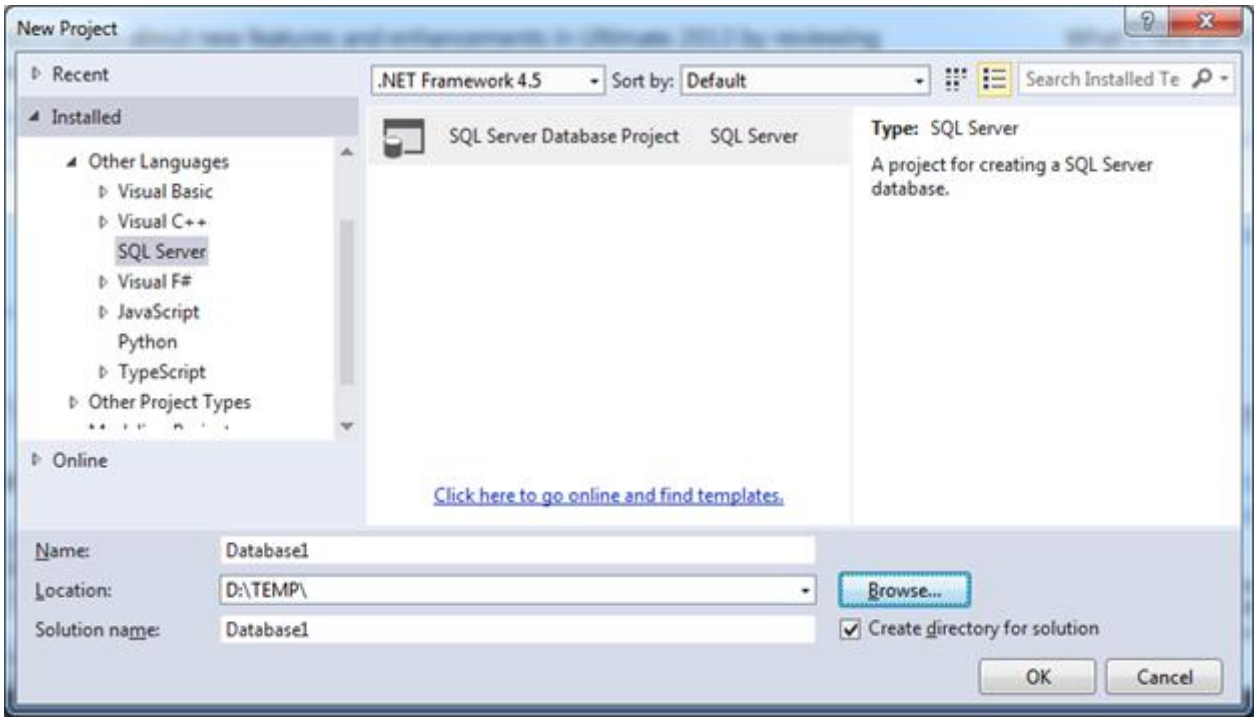


Рисунок 3.1-Створення нового проекту з використанням баз даних

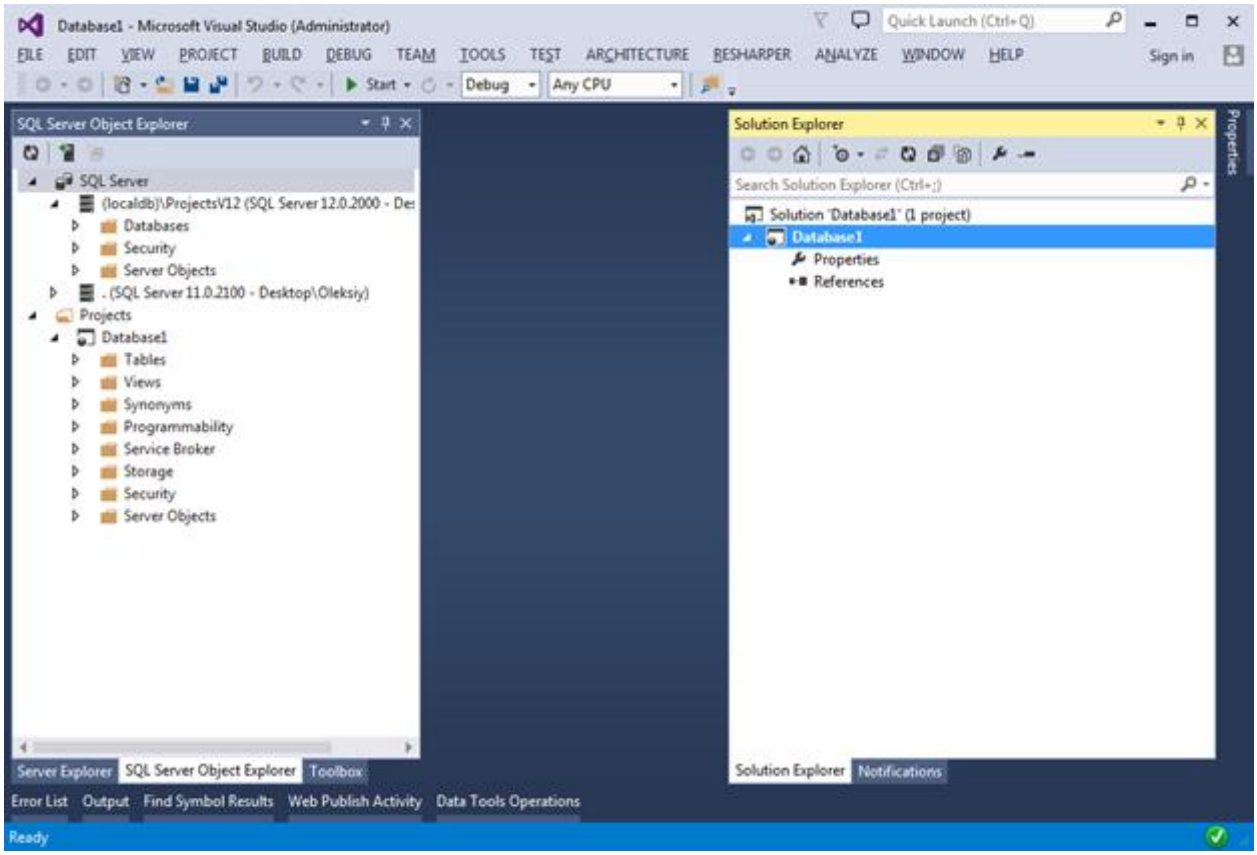


Рисунок 3.2-Система розробки у Disconnected режимі

Цей режим розробки нам найбільш цікавий, тому що саме він дозволяє отримати основні переваги використання SSDT.

В основі роботи лежить дуже проста ідея - дозволити розробникам зберігати всі скрипти створення об'єктів БД (tables, views, store procedures і т. д.) в проекті спеціального типу в складі наявного або нового рішення (solution ). На основі скриптів, Visual Studio може згенерувати DACPAC файл, який по суті є zip архів з усіма t-sql скриптами. Маючи DACPAC файл можна буде зробити публікацію (publish) на необхідному екземплярі бази даних, шляхом порівняння схеми описаної в DACPAC і схеми в цільової базі даних. В ході публікації, спеціальні механізми роблять порівняння, в результаті чого автоматично створюються міграційні скрипти для застосування змін без втрати даних.

Для того що побачити це в дії, пропоную подивитися такі приклади.

Імпорт даних (рисунок 3.3). Викликаємо контекстне меню проекту і бачимо 3 можливих варіанти:

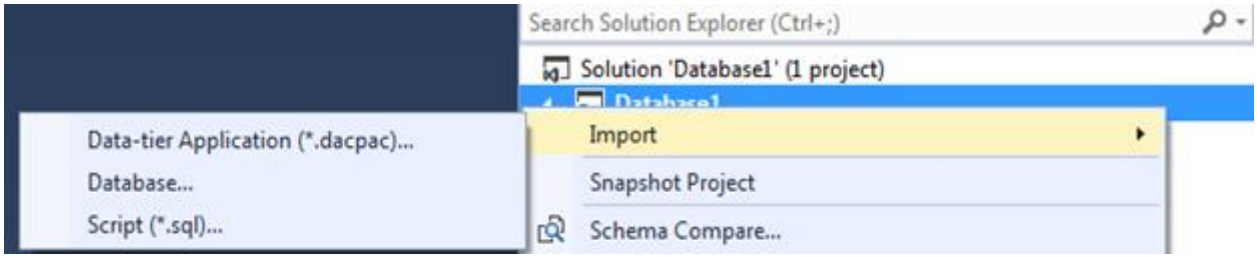


Рисунок 3.3-Імпорт даних

Так само ми завжди можемо додати нові елементи скориставшись діалоговим вікном Add New Item, (рисунок 3.4) в якому перераховані всі можливі об'єкти бази даних: Додамо таблицю TestTable. Новий файл-скрипт TestTable.sql буде додано до корінь проекту і для зручності ми його перенесемо в папку Tables.Для створення схеми таблиці ми можемо використовувати як панель дизайнера, так і панель T-SQL. Всі зміни зроблені на одній панелі будуть відразу ж відображені в інший. Так само ми можемо змінювати існуючі скрипти.Visual Studio для цього надає зручний і улюблений усіма IntelliSense. Так як ми не підключені до фізичної базі даних, Visual Studio для коректної роботи IntelliSence парсит всі скрипти в проекті, що дозволяє їй миттєво відображати останні зміни зроблені в схемі бази даних. Хочу звернути увагу на те, що ми не повинні піклуватися про інкрементних зміни нашої бази. Замість цього ми завжди створюємо скрипти так, як якщо б об'єкти створювалися заново. При публікації DACPAC пакета міграційні скрипти будуть згенеровані автоматично, шляхом порівняння DACPAC файлу і схеми в цільової базі (target Database). Як уже згадувалося, DACPAC містить не тільки схему і дані, але ще і ряд корисних налаштувань, для перегляду / редагування яких ми можемо скористатися вікном властивостей нашого проекту. Властивість Target platform дозволяє виставити версію бази даних, для якої будуть затверджувати скрипти в проекті. Мінімальна підтримувана

версія MS SQL Server 2005. Якщо наприклад задати версію бази 2005 і спробувати створити колонку типу Geography, то при компіляції ми отримаємо наступне повідомлення: На закладці Project Settings, ми можемо визначити установки бази даних, натиснувши на кнопку Database Settings. Натиснувши на неї ми побачимо діалог з настройками, аналогічні тим, що ми звикли бачити в SQL Server Management Studio: Так само хочеться згадати ятати SQLCMD Variables, на якій ми можемо задавати різні змінні для подальшого їх використання в наших скриптах.

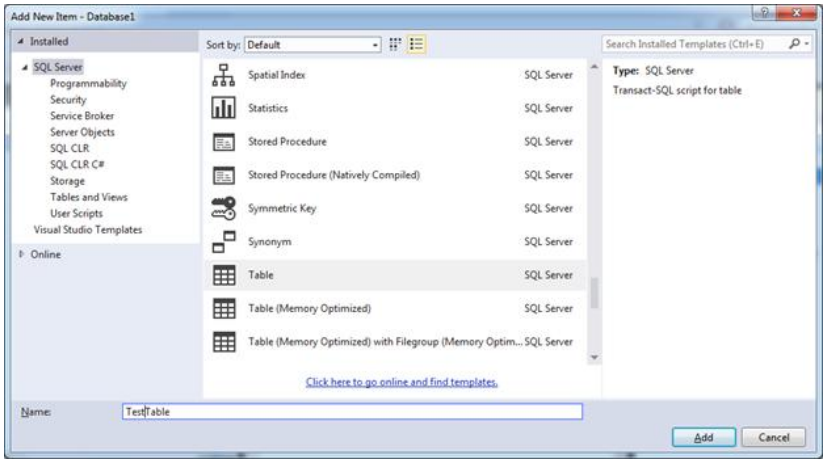


Рисунок 3.4-Діалогове вікно для додавання нових елементів

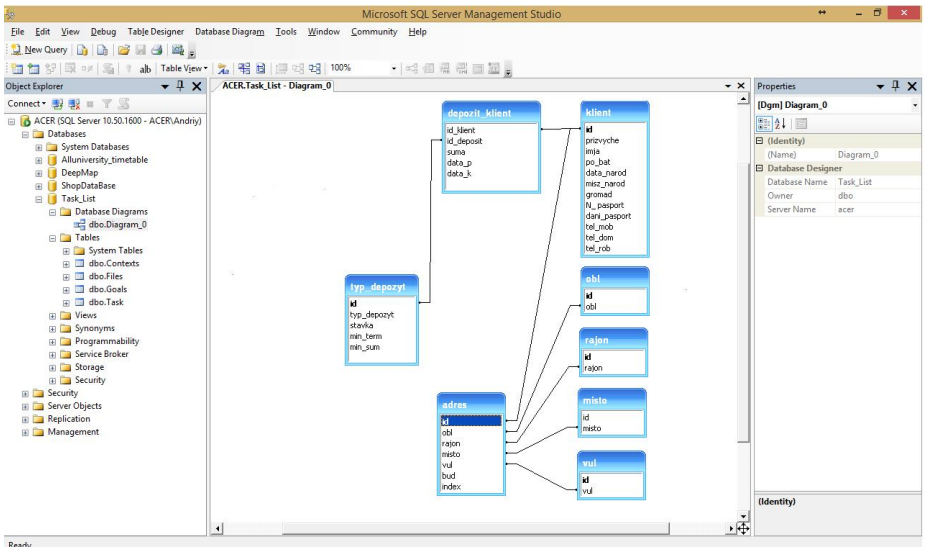


Рисунок 3.2- Реалізація бази даних

## Висновки до третього розділу

В третьому розділі описано процес кодування (створення програмного коду) програмної системи для підтримки прийняття рішень з видачі кредиту. В цьому розділі також показані діалогові вікна для вирішення конкретних необхідних задач.

# РОЗДІЛ IV
# РОЗДІЛ ТЕСТУВАННЯ ТА ДОСЛІДНОЇ ЕКСПЛУАТАЦІЇ

## 4.1.1 Поняття процесу тестування

Тестування - це процес, спрямований на виявлення помилок та їх усунення.

Процес тестування включає:

- дії, спрямовані на виявлення помилок;
- діагностику і локалізацію помилок;
- внесення виправлень у програму з метою усунення помилок.

Велика трудомісткість тестування і обмежені ресурси призводять до необхідності систематизації процесу і методів тестування. Включені методи тестування спрямовані на виявлення максимального числа помилок в найбільш важливих режимах функціонування програм при обмежених ресурсах.

## 4.1.2 Види і методи тестування

Статичний тестування - базується на правилах структурної побудови програм і обробки даних. Оператори та операнди тексту програми аналізуються в символьному вигляді.

Детерміноване тестування - вимагає багаторазового виконання програми на ЕОМ з використанням певних, спеціальним чином підібраних тестових наборів даних.

Стохастичне тестування - припускає використання в якості вихідних даних множину випадкових величин з відповідними розподілами, а для порівняння отриманих результатів використовуються також розподілу випадкових величин.

Тестування в реальному масштабі часу - в процесі тестування перевіряються результати обробки вихідних даних з урахуванням часу їх надходження, тривалості та пріоритетності обробки, динаміки використання пам'яті та взаємодії з іншими програмами.

Кожен з розглянутих методів тестування не виключає послідовного застосування іншого методу, скоріше навпаки, вимога до підвищення якості програмного виробу передбачає необхідність піддавати їх різним методам тестування.

Найбільш ефективним методом тестування є детерміноване тестування.

Детерміноване тестування грунтується на двох підходах: структурне тестування і функціональне тестування.

Структурне тестування передбачає детальне вивчення тексту програми і побудова таких вхідних наборів даних, які дозволили б при багаторазовому виконанні програми на ЕОМ забезпечити виконання максимально можливої кількості маршрутів, логічних розгалужень, циклів.

Критерії тестових наборів:

- Покриття операторів - тести підбираються так, щоб кожен оператор виконувався хоча б один раз;

- Покриття рішень (переходів) - тести повинні забезпечити перевірку кожного умови, так щоб вони приймали значення «істинно» чи «хибно»;

- Покриття умов - необхідно, щоб результат кожного умови було виконано хоча б один раз і кожній точці входу в програму має бути передано управління при виклику, принаймні, один раз;

- Покриття умов-рішень - тести повинні складатися так, щоб виконувалися результати-умови, результати кожного рішення, і кожному оператору передавалося управління хоча б один раз;

- Комбінаторне покриття умов - створюється множина тестів, щоб всі можливі комбінації результатів-умов і всі оператори виконувалися хоча б один раз.

Функціональне тестування повністю абстрагується від тексту програми, а тестові набори вибираються на підставі аналізу вхідних функціональних специфікацій.

Критерії тестових наборів:

- Метод еквівалентного роздроблення - складається з двох етапів: виділення класів еквівалентності, побудова тестів.

Класи еквівалентності виділяються шляхом аналізу вхідної умови і роздроблення його на дві або більше груп. Існують правильні і неправильні класи еквівалентності.

На основі класів еквівалентності будуються тестові набори. Причому для правильних класів еквівалентності потрібно прагнути до мінімального числа тестових наборів, для кожного неправильного класу еквівалентності будується хоча б один тестовий набір.

- Аналіз граничних значень - цей метод передбачає дослідження ситуацій, що виникають на кордонах і поблизу кордонів еквівалентних розбиття.

- Метод функціональних діаграм - полягає в перетворенні вхідної специфікації програми у функціональну діаграму за допомогою найпростіших булевих відносин.

Кожен з розглянутих методів забезпечує створення певного набору тестів, але жоден з них сам по собі не може дати вичерпний набір тестів. Тому при розробці тестових наборів слід дотримуватися стратегії розумного поєднання всіх розглянутих методів.

## 4.2. Процес і результат тестування .

Для проведення тестування було складено специфікацію тестів для функціонального тестування (дивись таблицю 4.1).

Таблиця 4.1.

Специфікація тестів для функціонального тестування

| № п/п | Функція, що тестується | Тестові випадки | Тестові дані |
|---|---|---|---|
| 1. | Внести інформацію про клієнта | 4 | 7 |
| 2. | Вибрати тип кредиту | 4 | 12 |
| 3. | Сформувати кредит (завершити оформлення) | 2 | 5 |
| 4. | Ввести дані про область | 2 | 4 |
| 5 | Ввести дані про район | 3 | 6 |
| 6. | Ввести дані про місто | 2 | 4 |
| 7. | Ввести дані про вулицю | 2 | 5 |

Підсумки тестування:

7 тестів з 7 наведених у таблиці, специфікація тестів для функціонального тестування, пройшли успішно (77 даних із 78 множини тестових даних пройшли) отже функціональне тестування вважається як частково успішне -100% тестових випадків пройшли, 98,7- наборів тестових даних пройшли.

## 4.2. Розгортання програмного продукту

Програмна система створення з використанням сучасних інструментальних засобів MS Visual Studio з використанням всіх запропонованих розширень проектування та програмування. Тут використовується система управління базою даних SQL Server. Для розгортання продукту необхідно для кожного користувача встановити на його робочу станцію його програмне забезпечення. Далі на сервер баз даних

встановити запроектовану та реалізовану базу даних і всі додатки зв’язати з сервером бази даних.

## 4.3. Інструкція користувачу

### 4.3.1. Компоненти

Пакет розроблено на мові програмування C# 3.0 у середовищі розробки Microsoft Visual Studio .NET 2012 і може експлуатуватися під управлінням сімейства операційних систем Windows. Під час проектування підсистем відбувалося поєднання об’єктно-орієнтованого підходу до програмування з процедурно-орієнтованим. Всі класи документувались інформаційно і семантично.

Для коректної роботим пакету необхідна користувацька машина з процесором не менше 1,5 ГHz, оперативною пам’яттю не менше 1 Gb. Для експлуатації пакету під управлінням сімейства операційних систем Windows необхідно встановити збірку класів .NET Framework 4.1, мати в наявності всі необхідні файли бібліотек і налаштувань.

Для роботи пакету необхідно встановити SQL Server 2005/2008. При цьому потрібно встановити повнотекстовий пошук.

1. Встановити SQL Server 2005/2008.
2. Встановити пакет ImageProcessingDB запустивши на виконання файл ipDB_setup.exe.
3. Запустити на виконання файл ipdb.exe, який створить базу даних IMGPROCDB та всі необхідні таблиці.

## Висновки до четвертого розділу

В четвертому розділі описано результати тестування програмного засобу для розрахунку оптимальних параметрів споживацького кредиту, які показали, що інформаційний ресурс є придатним до використання, а також надано інформацію, що до його розгортання.

# ВИСНОВКИ

В результаті проведеного вивчення питання стану програмного забезпечення для розрахунку оптимальних параметрів споживацького кредиту було поставлена задача створити своє аналогічне програмне забезпечення з використанням технології .NET. Це дало можливість створити програмне забезпечення для розрахунку оптимальних параметрів споживацького кредиту. Тестування показало коректну роботу системи. В четвертому розділі також наведено інструкцію до використання програмного забезпечення.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Г. Шилдт. Полный справочник по C#. – М.: Издательский дом "Вильямс", 2008.

2. Вигерс Карл Разработка требований к программному обеспечению. Пер, с англ. - М.:Издательско-торговый дом «Русская Редакция», 2004. - 576с.

3. Марка Д.А. Методология структурного анализа и проектирования – С.-Пб.: Питер, 1995. – 235 с.

4. Лешек А. Мацяшек Анализ требований и проектирование систем. Разработка информационных систем с использованием UML.: Пер. с англ. - М.: Издательский дом "Вильямс". - 2002. - 432 с.

5. Орлик С., Булуй Ю. Введение в программную инженерию и управление жизненным циклом ПЗ Программная инженерия. Программные требования. Режим доступу: http://www.sorlik.ru/swebok/3-1-software_engineering_requirements.pdf

6. Фаулер Скотт До. UML в короткому викладі. Застосування стандартної мови об'єктного моделювання: Пер. з англ. – М.:Мир, 1999. – 191 с.

7. Алістер Коберн. Сучасні методи опису функціональних вимог до систем

8. Л.Новіков. Введення в Rational Unified Process. - Режим доступу: http://www.interface.ru/rational/interface/151199/rup/main.htm

9. Фаулер М., Скотт К. UML в кратком изложении. Применение стандартного языка объектного моделирования: Пер. с англ. – М.:Мир, 1999. – 191 с.

10. Маклаков С.В. Bpwin, Erwin, Case-средства разработки информационных систем. – Москва —ДиалогМифи ‖ – 2000

11. Орлов С. Технологии разработки программного обеспечения: Учебник. – СПб.: Питер, 2002. – 464 с.

12. Каменова, Громов. Моделирование бизнеса. Методология ARIS. — М.: Весть-МетаТехнология, 2001.

13. Э. Троелсен. C# и платф орма .NET. Библиотека программиста. – СПб. : Питер, 2007.

14. Т.П. Караванова. Основи алгоритмізації та програмування. 750 задач з рекомендаціями та прикладами. – К.: Форум, 2002.

15. Э. Кингсли-Хьюджес, К. Кингсли-Хьюджес. C# 2005. Справочник программиста. – М.: ООО «ИД Вильямс», 2007.

16. Б. Керниган, Р. Пайк. Практика программирования. – СПб.: «Невский диалект», 2001.

17. В. О. Грязнова, С. В. Єфіменко. Основи методології програмування. – К.: ВПЦ «Київський університет», 2005.

18. Г. С. Иванова. Основы программирования: Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002.

19. Г. С. Иванова. Технология программирования: Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002.

20. Г. С. Иванова, Т. Н. Ничушкина, Е. К. Пугачев. Объектно-ориентированное программирование: Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2003.

21. А. Дубовцев. Microsoft .NET в подлиннике. - СПб: "БХВ-СПб", 2004.

Додаток А

Код програми

```
<VisualStudioProject>
   <CSHARP
      ProjectType = "Local"
      ProductVersion = "7.10.3077"
      SchemaVersion = "2.0"
      ProjectGuid = "{3BC1433A-8413-49D8-9716-E78ECF47EE6F}"
   >
      <Build>
         <Settings
            ApplicationIcon = "App.ico"
            AssemblyKeyContainerName = ""
            AssemblyName = "SQLLite"
            AssemblyOriginatorKeyFile = ""
            DefaultClientScript = "JScript"
            DefaultHTMLPageLayout = "Grid"
            DefaultTargetSchema = "IE50"
            DelaySign = "false"
            OutputType = "WinExe"
            PreBuildEvent = ""
            PostBuildEvent = ""
            RootNamespace = "SQLLite"
            RunPostBuildEvent = "OnBuildSuccess"
            StartupObject = ""
         >
```

```
<Config
    Name = "Debug"
    AllowUnsafeBlocks = "false"
    BaseAddress = "285212672"
    CheckForOverflowUnderflow = "false"
    ConfigurationOverrideFile = ""
    DefineConstants = "DEBUG;TRACE"
    DocumentationFile = ""
    DebugSymbols = "true"
    FileAlignment = "4096"
    IncrementalBuild = "false"
    NoStdLib = "false"
    NoWarn = ""
    Optimize = "false"
    OutputPath = "bin\Debug\"
    RegisterForComInterop = "false"
    RemoveIntegerChecks = "false"
    TreatWarningsAsErrors = "false"
    WarningLevel = "4"
/>
<Config
    Name = "Release"
    AllowUnsafeBlocks = "false"
    BaseAddress = "285212672"
    CheckForOverflowUnderflow = "false"
    ConfigurationOverrideFile = ""
    DefineConstants = "TRACE"
    DocumentationFile = ""
    DebugSymbols = "false"
```

```
                FileAlignment = "4096"
                IncrementalBuild = "false"
                NoStdLib = "false"
                NoWarn = ""
                Optimize = "true"
                OutputPath = "bin\Release\"
                RegisterForComInterop = "false"
                RemoveIntegerChecks = "false"
                TreatWarningsAsErrors = "false"
                WarningLevel = "4"
            />
        </Settings>
        <References>
            <Reference
                Name = "System"
                AssemblyName = "System"
                HintPath =
"C:\WINNT\Microsoft.NET\Framework\v1.1.4322\System.dll"
            />
            <Reference
                Name = "System.Data"
                AssemblyName = "System.Data"
                HintPath =
"C:\WINNT\Microsoft.NET\Framework\v1.1.4322\System.Data.dll"
            />
            <Reference
                Name = "System.Drawing"
                AssemblyName = "System.Drawing"
```

```
                HintPath                                                =
"C:\WINNT\Microsoft.NET\Framework\v1.1.4322\System.Drawing.dll"
              />
              <Reference
                Name = "System.Windows.Forms"
                AssemblyName = "System.Windows.Forms"
                HintPath                                                =
"C:\WINNT\Microsoft.NET\Framework\v1.1.4322\System.Windows.Forms.dll"
              />
              <Reference
                Name = "System.XML"
                AssemblyName = "System.Xml"
                HintPath                                                =
"C:\WINNT\Microsoft.NET\Framework\v1.1.4322\System.XML.dll"
              />
              <Reference
                Name = "SQLite.NET"
                AssemblyName = "SQLite.NET"
                HintPath                                                =
"..\..\..\sqllite\SQLite.NET.0.21_x68_dll\SQLite.NET.0.21_x68_dll\SQLite.NET.dll
"
              />
            </References>
          </Build>
          <Files>
            <Include>
              <File
                RelPath = "App.ico"
                BuildAction = "Content"
```

```
            />
            <File
               RelPath = "AssemblyInfo.cs"
               SubType = "Code"
               BuildAction = "Compile"
            />
            <File
               RelPath = "Form1.cs"
               SubType = "Form"
               BuildAction = "Compile"
            />
            <File
               RelPath = "Form1.resx"
               DependentUpon = "Form1.cs"
               BuildAction = "EmbeddedResource"
            />
         </Include>
      </Files>
   </CSHARP>
</VisualStudioProject>
namespace Task_List
{
   public partial class AddTaskForm : Form
   {
      int EditingTaskId;
      public AddTaskForm()
      {
         InitializeComponent();
         FillContextComboBox();
```

```
            FillAttachedFileComboBox();
        }
        public AddTaskForm(UserTask usrTask)
        {
            InitializeComponent();
            FillContextComboBox();
            FillAttachedFileComboBox();
            FillEditingFields(usrTask);
            EditingTaskId = usrTask.Id;
            updateTaskButton.Show();
            addTaskButton.Hide();
        }
        public void FillEditingFields(UserTask usrTask)
        {
            taskNameTextBox.Text = usrTask.TaskName;
            contextComboBox.Text = usrTask.Context;
            startDateDateTimePicker.Value                                    =
DateTime.Parse(usrTask.StartDate);
            dueDateDateTimePicker.Value = DateTime.Parse(usrTask.DueTime);
            priorityTrackBar.Value = usrTask.TaskPriority;
            statusComboBox.Text = usrTask.Status;
            descriptionRichTextBox.Text = usrTask.Description;
            if (usrTask.AttachedFile != null)
                addFileComboBox.Text = usrTask.AttachedFile.FileName;
            else
                addFileComboBox.Text = String.Empty;
        }
        void FillContextComboBox()
        {
```

```
            List<Context> contexts = new List<Context>();
            contexts = DataRetreiver.GetContexts();
            foreach (Context cnt in contexts)
            {
                contextComboBox.Items.Add(cnt.ContextName);
            }
        }
        void FillAttachedFileComboBox()
        {
            List<AttachedFile> files = new List<AttachedFile>();
            files = DataRetreiver.GetFiles();
            addFileComboBox.Items.Add(new AttachedFile());
            foreach (AttachedFile atf in files)
            {
                addFileComboBox.Items.Add(atf);
            }
            addFileComboBox.DisplayMember = "FileName";

        }

        private void addTaskButton_Click(object sender, EventArgs e)
        {
            /*validation*/
            UserTask usrTsk = new UserTask();
            if(taskNameTextBox.Text  !=String.Empty  ||  contextComboBox.Text
!= String.Empty || statusComboBox.Text != String.Empty)
            {
            usrTsk.TaskName = taskNameTextBox.Text;
            usrTsk.Context = contextComboBox.SelectedText;
```

```
                usrTsk.StartDate                                        =
startDateDateTimePicker.Value.Date.ToShortDateString();
                usrTsk.DueTime                                          =
dueDateDateTimePicker.Value.Date.ToShortDateString();
                usrTsk.TaskPriority = priorityTrackBar.Value;
                usrTsk.Status = statusComboBox.SelectedText;
                usrTsk.Description = descriptionRichTextBox.Text;
                usrTsk.AttachedFile                                     =
(AttachedFile)addFileComboBox.SelectedItem;
                /*send object to save to db*/
                try
                {
                   DataSaver.AddTask(usrTsk);
                }
                catch
                {

                }
                finally
                {
                   this.Close();
                }
                }
                else
                {
                MessageBox.Show("Fill all required fields!");
                }
```

```
        }

        private void cancelTaskButton_Click(object sender, EventArgs e)
        {
            updateTaskButton.Hide();
            addTaskButton.Show();
            this.Close();
        }

        private void updateTaskButton_Click(object sender, EventArgs e)
        {
            UserTask usrTsk = new UserTask();
            usrTsk.Id = EditingTaskId;
            if (taskNameTextBox.Text != String.Empty || contextComboBox.Text
!= String.Empty || statusComboBox.Text != String.Empty)
            {
                usrTsk.TaskName = taskNameTextBox.Text;
                if (contextComboBox.SelectedText != String.Empty)
                    usrTsk.Context = contextComboBox.SelectedText;
                else
                    usrTsk.Context = contextComboBox.Text;
                usrTsk.StartDate                                        =
startDateDateTimePicker.Value.Date.ToShortDateString();
                usrTsk.DueTime                                          =
dueDateDateTimePicker.Value.Date.ToShortDateString();
                usrTsk.TaskPriority = priorityTrackBar.Value;
                if (statusComboBox.SelectedText != String.Empty)
                    usrTsk.Status = statusComboBox.SelectedText;
```

```
                else
                usrTsk.Status = statusComboBox.Text;
                usrTsk.Description = descriptionRichTextBox.Text;
                if (addFileComboBox.SelectedItem != null)
                    usrTsk.AttachedFile =
(AttachedFile)addFileComboBox.SelectedItem;
                else
                {
                    AttachedFile att = new AttachedFile();
                    usrTsk.AttachedFile = att;
                }

                try
                {
                    DataUpdater.UpdateTask(usrTsk);
                }
                catch
                {

                }
                finally
                {
                    updateTaskButton.Hide();
                    addTaskButton.Show();
                    this.Close();
                }
            }
            else
            {
```

```
                MessageBox.Show("Fill all required fields!");
            }



        }
    }
}
```