

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему

КОМП'ЮТЕРНА СИСТЕМА БАНКУ

НА ОСНОВІ ПЛАТФОРМИ .NET

Виконав: студент 2 курсу, групи КНТ-522м
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

«Комп'ютерні системи та мережі»

ЧЕРЕВКО А.І

(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С. Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЄФИМЕНКО М.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) «Комп'ютерна системи та мережі»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ЧЕРЕВКО Антон Ігорович

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Комп'ютерна система банку на основі платформи .NET

керівник проєкту (роботи) к. т. н., доцент, СКРУПСЬКИЙ Степан Юрійович

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) Кількість комп'ютерів у філіальній мережі
банку - понад 120; кількість виділених серверів для проведення мікротранзакцій - два;
пропускна спроможність внутрішнього каналу зв'язку - 300 Мбіт/с; пропускна
спроможність зовнішнього каналу - 100 Мбіт/с; мова програмування додатка - C#;
технології Windows Forms, ADO.NET і база даних - MS SQL Server.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Проектування системної інфраструктури банку;
2) Розробка програмних компонентів системи банку;
3) Дослідження розробленої системи банку.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	СКРУПСЬКИЙ С.Ю., доцент		
нормоконтроль	ПОЛЬСЬКА О.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз наявних методів та алгоритмів розробки мережі банку, перегляд отриманих результатів, виділення недоліків та переваг	12.10.2023 р.	
2	Встановлення та налаштування програмного забезпечення	15.10.2023 р.	
3	Аналіз технології ADO.NET	20.10.2023 р.	
4	Розробка програмних компонентів системи	12.11.2023 р.	
5	Дослідження розробленої системи	24.11.2023 р.	
6	Оформлення отриманих результатів у ПЗ	01.12.2023 р.	
7	Оформлення графічного матеріалу	05.12.2023 р.	
8	Оформлення допоміжного матеріалу	08.10.2023 р.	
1	Аналіз наявних методів та алгоритмів розробки	12.10.2023 р.	

Студент(ка)

Керівник проєкту (роботи)


(підпис)

Антон ЧЕРЕВКО
(Ім'я ПРИЗВИЩЕ)

Степан СКРУПСЬКИЙ
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 79 с., 20 рис., 10 табл., 38 лістингів, 16 джерел.

C#, WINDOWS FORMS, ADO.NET, КОМП'ЮТЕРНА СИСТЕМА,
РЕГРЕСІЙНА МОДЕЛЬ, GIGABIT ETHERNET.

Метою магістерської роботи є створення та дослідження комп'ютерної системи на основі технології доступу до баз даних ADO.NET, для підвищення ефективності процесу обслуговування банківських мікротранзакцій.

У даній роботі було спроектовано комп'ютерну мережу філії банку, в якій за допомогою восьми комутаторів і маршрутизаторів об'єднано 120 комп'ютерів, контролер домену і два сервери системи обслуговування мікротранзакцій банку.

Для підвищення ефективності процесу обслуговування банківських мікротранзакцій розглядається технологія доступу до баз даних ADO.NET. ADO.NET дає змогу програмувати доступ до СУБД на дуже високому рівні.

Під час дослідження було встановлено, що робота системи в ADO.NET у від'єднаному режимі значно знижує навантаження на сервер порівняно з підключеним режимом і відводить частину його на клієнта.

У роботі запропоновано регресійна поліноміальна модель, яка може передбачити навантаження на сервери системи.

ABSTRACT

Explanatory note to the master's work: 79 p., 20 figures, 10 tables, 38 listings, 16 sources.

C#, WINDOWS FORMS, ADO.NET, COMPUTER SYSTEM, REGRESSION MODEL, GIGABIT ETHERNET.

The purpose of the master's thesis is to create and study a computer system based on ADO.NET database access technology to improve the efficiency of the banking microtransaction service process.

In this work, a computer network of a bank branch was designed, in which 120 computers, a domain controller and two servers of the bank's microtransaction service system are combined using eight switches and routers.

To improve the efficiency of the bank's microtransaction servicing process, the ADO.NET database access technology is being considered. ADO.NET allows programming access to the DBMS at a very high level.

The study found that the operation of the system in ADO.NET in the disconnected mode significantly reduces the load on the server compared to the connected mode and allocates part of it to the client.

The paper proposes a regression polynomial model that can predict the load on the system's servers.

ЗМІСТ

Вступ.....	8
1 Проектування системної інфраструктури банку.....	10
1.1 Вибір мережевого обладнання.....	10
1.1.1 Комп'ютери	10
1.1.2 Сервер.....	10
1.1.3 Комутатори	11
1.1.4 Маршрутизатор.....	11
1.2 Організація комп'ютерних мереж.....	13
1.3 Вибір мережевого середовища передачі даних.....	14
1.4 Створення структурної схеми мережі.....	16
1.5 Розробка монтажною схеми мережі.....	18
1.6 Вибір і встановлення мережевого програмного забезпечення.....	20
1.6.1 Вибір і встановлення клієнтської операційної системи.....	20
1.6.2 Вибір і встановлення серверної операційної системи.....	25
1.7 Політика використання IP-адрес для мережевих та інтернет-з'єднань	27
1.8 Забезпечення захисту даних у мережі.....	29
1.9 Проектування апаратного забезпечення серверів обслуговування мікротранзакцій	33
1.9.1. Вибір джерела безперебійного живлення.....	34
1.9.2 Вибір системи кондиціонування	35
2 Розробка програмних компонентів системи банку	36
2.1 Встановлення та налаштування MS SQL Server	36
2.2 Проектування банківської бази даних	41
2.3 Розробка системних інтерфейсів	45
2.4 Аналіз технологій доступу до баз даних ADO.NET	50
2.5 Розробка програмного забезпечення для доступу до системи в підключеному режимі.....	55

2.6 Розробка програмного забезпечення для доступу до систем у від'єднаному режимі.....	62
3 Дослідження розробленої системи банку	68
3.1 Постановка задачі.....	68
3.2 Проведення експериментів.....	68
3.3 Розробка регресійних моделей для прогнозування навантаження на сервер системи	71
3.4 Економічна ефективність проєкту.....	74
Висновки	77
Перелік джерел посилання	78

ВСТУП

Метою магістерської роботи є створення та дослідження комп'ютерних систем на основі технології доступу до баз даних ADO.NET для підвищення ефективності процесу обслуговування банківських мікротранзакцій.

Система включає в себе серверне обладнання, розташоване в мережі відділень банку та в мережі Інтернет; на сервері встановлено програмне забезпечення, розроблене на мові C# з використанням фреймворку Windows Forms.

Це буде досягнуто шляхом вирішення наступних завдань:

- проектування інфраструктури системи, включаючи комп'ютерну мережу відділення банку;
- розробка системного програмного забезпечення
- розробка баз даних відділення банку;
- аналіз технології доступу до баз даних ADO.NET
- експериментальне дослідження розробленої системи для створення моделі оцінки навантаження на сервери банку.

Основна частина дослідження складається з анотації, вступу, трьох розділів, списку використаних джерел та висновків. У першому розділі обґрунтовано вибір системного обладнання та комп'ютерної мережі. Розроблено схему мережі, розглянуто встановлення та налаштування інфраструктурного програмного забезпечення та підключення до мережі Інтернет, а також захист даних у мережі.

Розділ 2 охоплює встановлення та налаштування СУБД MS SQL Server, проектування бази даних системи та розробку інтерфейсу для доступу до системи. Детально проаналізовано технологію доступу до бази даних ADO.NET, на основі якої розроблено програмне забезпечення системи. Доступ до бази даних реалізовано у підключеному та від'єднаному режимах.

У розділі 3 проведено експериментальне дослідження розробленої системи. Розроблено постановку задачі, оцінено навантаження на сервер у режимах доступу з прив'язкою до бази та без прив'язки, а також розроблено регресійну модель для оцінки навантаження на сервер при роботі з мікротранзакціями.

1 ПРОЕКТУВАННЯ СИСТЕМНОЇ ІНФРАСТРУКТУРИ БАНКУ

1.1 Вибір мережевого обладнання

Обґрунтувати вибір апаратного забезпечення для всіх компонентів мережі.

1.1.1 Комп'ютери

Материнська плата, сумісна з 14 ядерним процесором, процесор з тактовою частотою не менше 3,5 ГГц, 32 ГБ оперативної пам'яті та жорсткий диск об'ємом не менше 2 Тб. Конфігурація комп'ютера наведена в таблиці 1.1.

Таблиця 1.1 - Обладнання мережевих комп'ютерів

Материнська плата	Asus ROG Maximus Z790 Hero
Процесор	Intel Core i5-13600KF z , 14 ядер по 3.5GH
Відеоадаптер	Інтегрований на материнській платі
Оперативна пам'ять	32 ГБ DDR5 5600 МГц
Дисковий накопичувач	HDD 2 Тб Seagate BarraCuda HDD 2TB 7200rpm 256MB ST2000DM008 3.5 SATA III
Блок живлення	Straight Power 1200 Вт
Клавіатура/миша	Клавіатура PS/2, миша USB
Мережевий адаптер	Інтегрований адаптер Gigabit Ethernet на материнській платі
Монітор	«21» Samsung S21A300N LED, 1920x1080 (16:9)

1.1.2 Сервер

Для організації мережі банківських філій необхідно використовувати сервер контролера домену. Серверне обладнання має бути продуктивнішим, ніж обладнання робочих станцій, оскільки сервери несуть навантаження з обслуговування всієї мережі. Конфігурування серверного обладнання здійснюється з урахуванням продуктивної платформи Intel (табл. 1.2).

Таблиця 1.2 - Обладнання мережевих серверів

Материнська плата	Asus ROG Maximus Z790 Hero
Процесор	Intel Core i5-13600K, Box, 14 ядер по 3.5GHz
Відеоадаптер	Інтегрований на материнській платі
Оперативна пам'ять	32 Гб DDR5 5600 МГц
Дисковий накопичувач	HDD 10 Тб Western Digital Ultrastar DC HC330 10TB 7200rpm 256MB
Блок живлення	Straight Power 1200 Вт
Привід	DVD±R/RW USB 2.0
Клавіатура/миша	Клавіатура PS/2, USB миша
Мережевий адаптер	Інтегрований адаптер Gigabit Ethernet на материнській платі
Монітор	«21» Samsung S21A300N LED, 1920x1080 (16:9)

1.1.3 Комутатори

Мережі, побудовані за топологією «зірка», можливі тільки за допомогою комутаторів. Для підтримки технології Gigabit Ethernet його інтерфейс і комутаційна матриця повинні передавати дані по кручений парі з пропускною спроможністю не менше 1 Гбіт/с. Мережа побудована на базі 24-інтерфейсного комутатора. Інтелектуальні можливості комутаторів вимагають підтримки алгоритму Spanning Tree Algorithm (STA), який забезпечує відключення каналів зв'язку, що призводять до утворення петель. Їм також необхідна підтримка автоматичного визначення і зберігання MAC-адрес, що дає змогу знизити трафік протоколу ARP- під час зіставлення IP-адрес з MAC-адресами. Скористаємося комутаторами TP-LINK TL-SG1024.

1.1.4 Маршрутизатор

Головний і єдиний маршрутизатор у мережі повинен не тільки виконувати свою основну функцію - маршрутизацію вхідного і вихідного трафіку, а й виступати в ролі між мережевого екрана. Це забезпечує захист даних на апаратному рівні. Мережа складається з 120 комп'ютерів, контролера домену та

двох серверів системи обслуговування клієнтських запитів. Всі ці пристрої генерують велику кількість трафіку, який необхідно відправляти в Інтернет відповідно, трафік повинен прийматися. Тому пропускна здатність маршрутизатора повинна бути не менше 150 Мбіт/с (запас стандарту Fast Ethernet - 50 Мбіт/с). Звичайні недорогі маршрутизатори не можуть підтримувати таку функціональність, тому для корпоративних додатків використовуються спеціальні маршрутизатори. Аналіз моделей маршрутизаторів представлено в таблиці 1.3.

На основі аналізу, наведеного в таблиці 1.3, можна зробити висновок, що маршрутизатор TP-LINK Omada ER605 є найбільш прийнятним вибором з точки зору співвідношення вартості та пропускної здатності.

Таблиця 1.3 - Аналіз корпоративних маршрутизаторів

Модель маршрутизатора	Інтерфейси	Між мережевий екран	Додаткові можливості	Вартість, грн
Dlink DFL-800	2 порти Fast Ethernet, 1 порт DMZ	150 Мбіт/с, NAT, DOS, фільтр пакетів	—	16 364
Dlink DFL-1600	6 портів Gigabit Ethernet	320 Мбіт/с, NAT, DOS. фільтр пакетів	Виявлення відмови пристрою, обриву каналу	45 000
TP-LINK Omada ER605	5 порти Gigabit Ethernet, 1 порт DMZ	350 Мбіт/с, NAT, MAC фільтр	—	3149

Продовження таблиці 1.3

Модель маршрутизатора	Інтерфейси	Між мережевий екран	Додаткові можливості	Вартість, грн
Cisco 2801	2 порти Fast Ethernet, 1 порт DMZ	150 Мбіт/с, NAT, SPI, фільтр пакетів, виявлення вторгнень	Обмежена підтримка MPLS	18 000

1.2 Організація комп'ютерних мереж

Топологія мережі «зірка» є очевидним вибором, оскільки це єдина топологія, яка може забезпечити максимальну пропускну спроможність мережі за умови помірних витрат на прокладання кабелів. Розглянемо докладніше вибір технології канального рівня мережі. Технологія канального рівня визначає спосіб передавання даних на канальному рівні моделі OSI, спосіб доступу до середовища, розмір кадру та пропускну спроможність, що використовується середовищем передавання даних. Сучасні технології канального рівня, що використовуються в корпоративних мережах, включають Fast Ethernet, Gigabit Ethernet і 10G Ethernet.

Fast Ethernet - це фізичне середовище, що застосовується в мережах із топологією «зірка» і використовує віту пару UTP категорії 5 або вище завдовжки не більше 100 метрів. Це дає змогу обладнанню працювати як у напівдуплексному, так і в дуплексному режимі. При цьому дуплексний режим забезпечує максимально можливу для цього стандарту швидкість передачі 100 Мбіт/с. Для збільшення діаметра мережі необхідно використовувати два або більше ретранслятори, радіус мережі має становити не більше 205 метрів, а довжина кабелю між ретрансляторами має перевищувати 5 метрів.

Gigabit Ethernet - технологія каналного рівня, що забезпечує швидкість передавання даних до 1 Гбіт/с і застосовується в мережах, побудованих за топологією «зірка» з використанням виті пари категорії 5 і вище. Для передачі даних використовуються чотири пари провідників виті пари, а самі дані передаються на більш високій частоті, ніж у Fast Ethernet. Це забезпечує певний запас за рівнем сигналу, який використовується для виправлення помилок, що виникають. Для передачі даних використовуються кабелі категорії 5, 6 або 7 з неекранованою виті парою, довжина яких не перевищує 100 метрів. Для збільшення діаметра мережі використовуються два або більше ретрансляторів. Усі робочі станції підключаються до центрального пристрою управління - концентратора, комутатора або маршрутизатора. Максимальна кількість з'єднань становить 1024. Затримка доступу становить 180 мікросекунд.

У 10G Ethernet номінальна швидкість передачі даних збільшується до 10 Гбіт/с у повно дуплексному режимі (напівдуплекс не підтримується). Як середовище передачі даних використовуються оптоволоконні та мідні кабелі. Максимальна довжина кабелю становить 300 метрів, повторювачі між сегментами не використовуються. Затримка доступу становить 200 мікросекунд.

Таким чином, аналіз сучасних технологій каналного рівня комп'ютерної мережі та технічних завдань організації мережі філій банку привів до використання технології Gigabit Ethernet.

1.3 Вибір мережевого середовища передачі даних

На сьогодні вита пара є найпоширенішими кабелями. Це пов'язано насамперед з їхніми швидкісними характеристиками і простотою монтажу. Використання коаксіальних кабелів накладало обмеження на топологію мережі, що позначалося на можливостях модернізації та швидкості передавання даних, тому їхня поява була цілком передбачуваною.

Назва походить від його внутрішньої будови. Усередині кабелю міститься від 1 до 25 пар провідників, скручених разом і певного кольору. Зовнішній вигляд витвої пари залежить від кількості провідників всередині, типу обплетення, що використовується для захисту кабелю і пар, а також від наявності або відсутності додаткових заземлюючих провідників.

Кабелі можуть бути екранованими або неекранованими. Кабелі також поділяються за категоріями; чим вища категорія, тим кращі характеристики (в тому числі швидкість) кабелю. Наприклад, існує сім категорій кабелів витвої пари, які зараз використовуються в локальних мережах. Наприклад, кабелі категорії 5 можуть передавати дані зі швидкістю 100 Мбіт/с, а кабелі категорії 6 можуть передавати дані зі швидкістю не менше 1 Гбіт/с. Кабелі категорії 7 теоретично можуть передавати дані зі швидкістю 100 Гбіт/с [3].

Кабелі на основі витвої пари - найпоширеніший спосіб з'єднання комп'ютерів у мережу. Вартість кабелю дуже низька, але швидкість передачі даних дуже висока.

Основна відмінність між оптоволоконними кабелями та кабельними варіантами полягає в способі передачі електричних сигналів. Це означає, що оптоволоконні кабелі не схильні до впливу електромеханічних перешкод і мають дуже низьке загасання сигналу. Це забезпечує вищу швидкість передачі даних на великі відстані [4].

Оптоволоконні кабелі розрізняються залежно від діаметра серцевини, тобто товщини волокна. Існує два типи оптичного волокна, які мають виражений вплив на властивості кабелю. Тому розрізняють одномодове і багатомодове волокно.

Основна частина волоконно-оптичного кабелю, звана також оптоволоконном або світловодом, здійснює безпосередню передачу оптичних сигналів. Для запобігання витоку сигналу з оптоволокна навколо нього встановлюється відбивна оболонка товщиною 125 мкм. Ще одним елементом є оболонка, яка захищає кабель від зовнішніх впливів, таких як волога і сонячне світло. Додаткову міцність волоконно-оптичним кабелям зазвичай надають різні лакові покриття,

додаткова оболонка (буферний матеріал) і підсилювальні троси. Крім того, широко використовуються кабелі з декількома світловодами, що дає змогу значно збільшити пропускну здатність кабелю.

Найпростіші одномодові кабелі досягають швидкості передачі сигналу 2,5 Гбіт/с на довжині сегмента 20 км. Однак широкому поширенню волоконно-оптичних кабелів перешкоджає низка факторів. Основними з них є висока вартість кабелів і обладнання, що підключається до них, а також необхідність відповідної підготовки під час роботи з кабелями.

Згідно з технічними умовами, пропускну здатність внутрішнього сегмента мережі має становити 300 Мбіт/с. Технічні характеристики крученої пари UTP-5e повністю відповідають цим технічним умовам, тому немає необхідності використовувати дороге оптоволокно.

1.4 Створення структурної схеми мережі.

На структурній схемі показано компоненти мережі та зв'язки між ними. У відділенні банку є 120 комп'ютерів. Для їхнього підключення до мережі використовуються сім комутаторів (K1-K7) з приблизно однаковою кількістю з'єднань. Усі комутатори під'єднані до центрального комутатора (K8), до якого також під'єднані маршрутизатор, контролер домену і два сервери для системи обслуговування клієнтів. З'єднання між комп'ютерами та комутаторами показано в табл. 1.4.

Таблиця 1.4 - З'єднання комп'ютера з комутатором

Комутатор	Кількість входних підключень	Діапазон підключених комп'ютерів.
K1	19	ПК1–ПК19

Продовження таблиці 1.4

Комутатор	Кількість вхідних підключень	Діапазон підключених комп'ютерів.
K2	19	ПК20–ПК38
K3	18	ПК39–ПК56
K4	18	ПК57–ПК74
K5	19	ПК75–ПК83
K6	16	ПК84–ПК99
K7	21	ПК100–ПК120
K8	11	K1–K7, Маршрутизатор, Контролер домену, Сервер системи 1, Сервер системи 2.

Структурна схема філіальної мережі банку представлена на рис. 1.1.

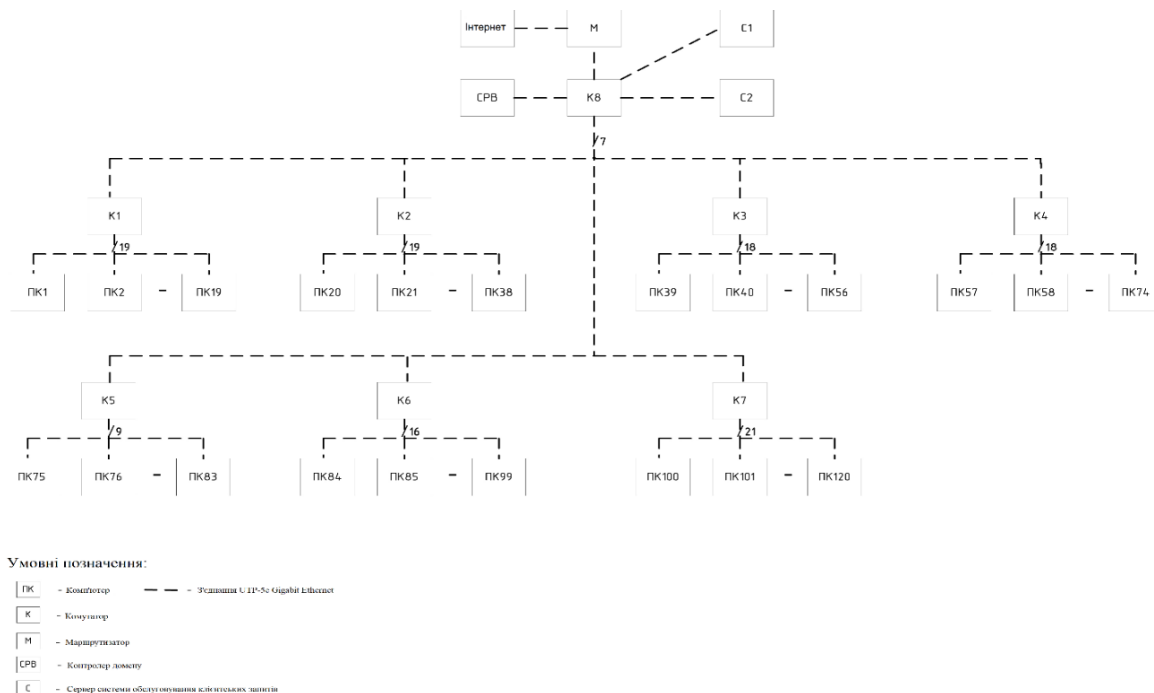


Рисунок 1.1 - Структурна схема мережі відділень банку

1.5 Розробка монтажної схеми мережі

Для створення монтажної схеми мережі необхідний план будівлі відділення банку (рис. 1.2). Поступово в кожному приміщенні розміщуються комп'ютери і розробляємо монтажну схему мережі – рис. 1.3.

Середня довжина кабелю (L_{cp}) розраховується за емпіричною формулою:

$$L_{cp} = \frac{(L_{max} + L_{min})}{2}, \quad (1.1)$$

де L_{min} і L_{max} - довжини кабельної траси від місця встановлення кросового обладнання до найближчого робочого місця.

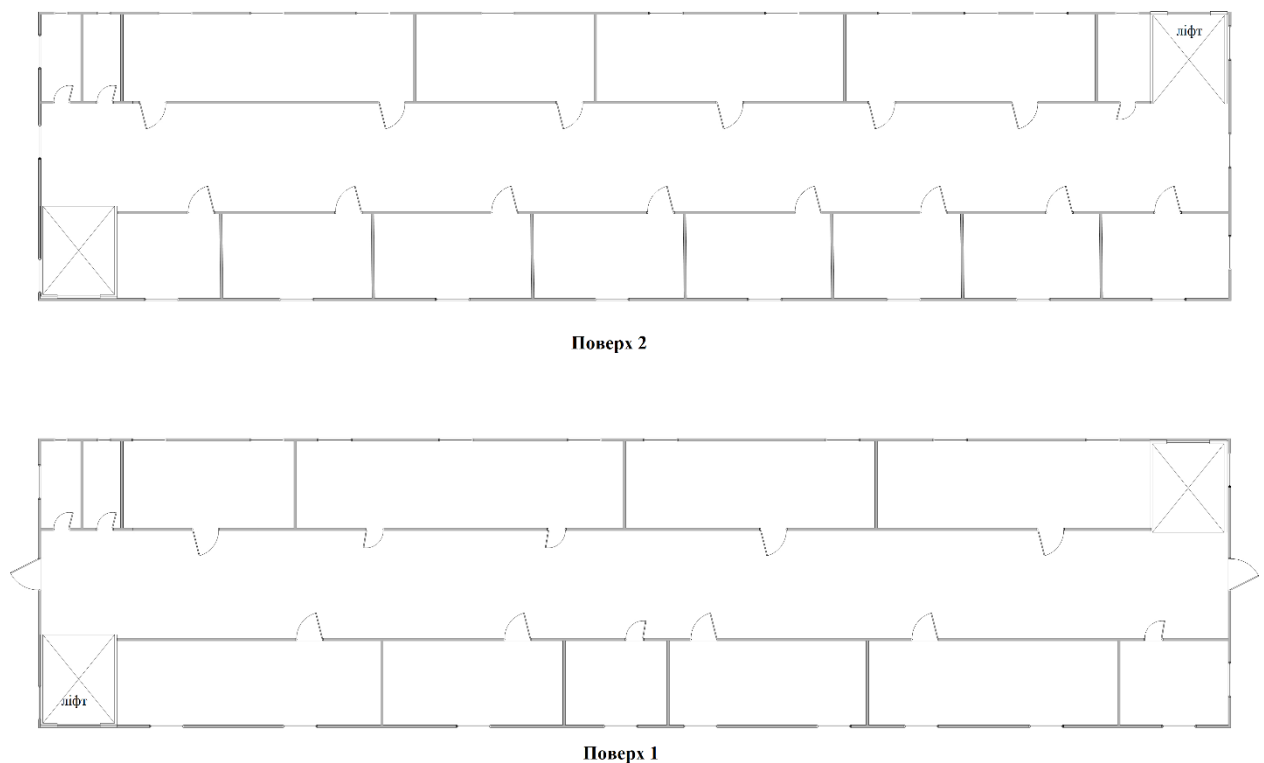


Рисунок 1.2 - План будівлі відділення банку

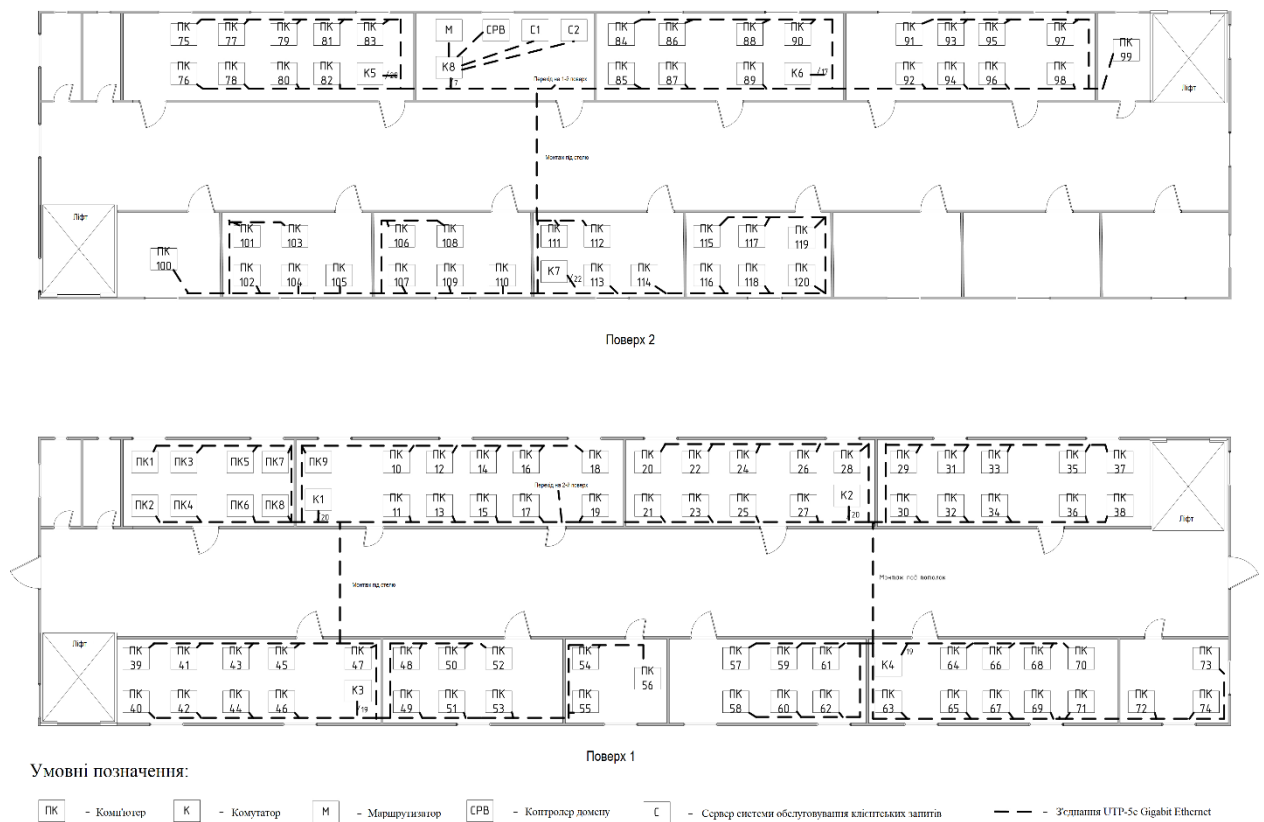


Рисунок 1.3 - Монтажна схема мережі

Під час визначення довжини траси необхідно додати технічний запас у розмірі 10% від L_{cp} і запас X від процедури прокладання кабелів для розподільних вузлів та інформаційних роз'ємів. Таким чином, довжина траси L визначається таким чином:

$$L = (L_{cp} \cdot 1,1 + X) \cdot N, \quad (1.2)$$

де N – кількість активних мережевих пристроїв у мережі.

Розрахуємо необхідну довжину кабелю. На підставі схеми підключення мережі: 120 комп'ютерів, 8 комутаторів, 1 маршрутизатор, 3 сервери

Таким чином, $N = 132$, $L_{min} = 4$ м, $L_{max} = 28$ м. Отже,

$L_{cp} = \frac{4+28}{2} = 16$ м. Тоді необхідна довжина кабелю становить:

$$L = (16 \cdot 1,1 + 2,5) \cdot 132 = 2653,2\text{м}. \quad (1.3)$$

1.6 Вибір і встановлення мережевого програмного забезпечення

1.6.1 Вибір і встановлення клієнтської операційної системи

Найпоширенішими сучасними клієнтськими операційними системами: Windows 10, Windows 11 і Linux; коротко проаналізуємо кожну з них.

Windows 10 - операційна система для персональних комп'ютерів і робочих станцій, розроблена компанією Microsoft у рамках сімейства Windows NT. Відмінною особливістю передніх версій є її здатність допомагати у розв'язанні найважливіших технічних і бізнес-завдань:

- універсальні додатки Office у Windows 10. Компанія Microsoft створює універсальні додатки Office, які підтримують сенсорне управління і працюють на всіх пристроях;
- DirectAccess. Можливість підключення віддалених користувачів до корпоративних ресурсів без створення віртуальної приватної мережі;
- BranchCache. Прискорене відкриття файлів та веб-сторінки у віддалених офісах;
- BitLocker. Захист даних на знімних пристроях;
- AppLocker. Просте встановлення обмежень для додатків і підвищити рівень безпеки;
- компанія Microsoft активно випускає оновлення системи безпеки для захисту ПК від потенційних загроз. Оновлення завантажуються і встановлюються автоматично, що дає змогу захистити комп'ютер одразу ж після появи нової загрози;
- значно швидше й ефективніше за попередні версії операційної системи. Удосконалено систему запуску та переходу в сплячий режим, що дає змогу швидше перемикатися між додатками та підвищує загальну продуктивність;
- у Windows 10 порівняні з попередніми версіями операційної системи покращено сумісність додатків.

Переваги Windows 10 для підприємств:

- локальний пошук Windows. Дає змогу шукати файли на локальному комп'ютері або в мережі;
- підключення до домену. Просте і безпечне підключення до серверів через мережу;
- друк за місцем розташування. Легко знайти потрібний принтер;
- підвищена безпека в Internet Explorer. Фільтри захищають конфіденційність і запобігають фішинговим атакам;
- покращено функціональність резервного копіювання. Тепер мобільні користувачі можуть створювати резервні копії файлів у мережі, а не на локальному диску;
- було додано низку технологій, що дають змогу створювати 3D-зображення і проєктувати голограми в додатках Windows 10.
- Windows 10 забезпечує високу продуктивність кінцевого користувача, де б він не знаходився і де б не зберігалися потрібні йому дані;
- спрощене управління ПК. Windows 10 – полегшує ІТ-фахівцям розгортання та керування настільними комп'ютерами, ноутбуками та віртуальними середовищами [3].

Windows 11 - це остання версія операційної системи Microsoft, яка має такі переваги перед Windows 10:

- у Windows 11 міститься більша кількість типів програм за межами універсальної платформи Windows (UWP);
- Windows 11 здійснює багато зусиль контролю пам'яті, щоб оптимізувати вікна відкритих програм. Це передбачає великі витрати потужності центрального процесора, якщо порівнювати з іншими системними інструментами. Windows 11 призупиняє окремі дані в ОЗУ, поки ваш ПК перебуває в режимі сну. А вихід із сплячого режиму відбувається на 25% швидше, ніж у десятій версії Windows;
- стартове меню 11 значно спрощено. У ньому відображається статичний список додатків, а найбільш часто використовувані документи

включені в нижню частину. У Windows 11 усунуто "живі плитки", що надає меню "Пуск" чистішого вигляду;

- новий спосіб доступу до додатків з Windows Store. Цей спосіб дає користувачам низку переваг, як-от безпека та ефективніше використання завантажених додатків;

- у Windows 11 спрощується робота з вікнами в режимі планшета. Стане простіше переміщати і змінювати розміри вікон на екрані;

- оптимізація багатозадачності в Windows 11 - унікальне оновлення порівняно з Windows 10: в останній версії ОС для підвищення багатозадачності та продуктивності використовуються макети прив'язки, що дають змогу групувати вікна і зберігати їх на панелі завдань. Ця функція була доступна і в попередніх версіях Windows, але в Windows 11 доступ до Snap Layouts став простішим, а можливостей налаштування стало більше. Microsoft також оновила Snap Layouts для поліпшення функціональності [5].

Операційна система Unix є найстарішою мережевою операційною системою, яка використовується і сьогодні; існує багато клонів Unix і майже нерозрізнених операційних систем від різних виробників: FreeBSD, BSD Unix (Berkley University), SunOS, Solaris (Sun Microsystems), AIX (IBM), HP-UX (Hewlett Packard) і SCO (SCO). Найпопулярнішим клоном Unix є FreeBSD, оскільки її вихідний код вільно розповсюджується і дозволяє людям довільно модифікувати операційну систему «під себе» та тестувати її на наявність помилок і лазівок. У цьому відношенні FreeBSD містить набагато менше помилок, ніж комерційна версія Unix.

Linux, клон Unix, нещодавно став незалежною операційною системою і продовжує стрімко розвиватися. Багато дистрибутивів Linux (інсталяційних пакетів) доступні від різних компаній. Найпопулярнішими є Red Hat Linux (США) та Mandrake (Європа). Існують також Slackware Linux, Corel Linux, Caldera OpenLinux, Debian Linux, SuSE Linux, Black Cat Linux і Connectiva Linux. Linux та Unix мають схожі структури файлових систем, системи контролю доступу та базові команди. З точки зору користувача, основною відмінністю Linux від

попередніх версій Unix є зручний графічний інтерфейс, який багато в чому нагадує інтерфейс Windows (особливо графічне робоче середовище Gnome), а основними перевагами - більша надійність і швидкість, більша безпека файлової системи (включаючи захист від вірусів), а також спеціалізовані інструменти для роботи в локальних мережах та Інтернеті. Для Linux існує і розробляється ціла низка програм, починаючи від офісного пакету Star Office і графічного редактора Corel Draw і закінчуючи потужною СУБД (DB2 від IBM), мовами C++, Perl, Java та іншими системами розробки програмного забезпечення, що перебувають у стадії розробки.

Перевагами Unix/Linux є більша надійність і стабільність, значно менша кількість вірусів та іншого шкідливого програмного забезпечення, а також нижчі системні вимоги порівняно з Windows. До недоліків можна віднести низький рівень проникнення програмного забезпечення та додаткові навички, необхідні персоналу для роботи з цією операційною системою.

Використання Linux обмежує можливості встановлення програмного забезпечення на робочих станціях відділень банку. Відгуки користувачів показують, Windows 11 все ще потребує поліпшення, тому перевага надається перевіреним Windows 10.

У проєктованій мережі 120 комп'ютерів, і встановлення операційної системи на кожен з них зайняло б дуже багато часу. Тому слід використати реплікацію образу і скопіювати його по мережі. Встановіть систему на один комп'ютер. Після завантаження з дистрибутива натисніть кнопку «Встановити». Виберіть версію Windows 10 відповідно до придбаної версії. Архітектура системи x64. Прийміть умови ліцензійної угоди, натисніть кнопку «Завершити встановлення». Створіть розділ розміром 100 ГБ у нерозміченій області жорсткого диска та відформатуйте його у файлову систему NTFS. Вибіримо цей розділ для встановлення системи. Після встановлення системи вводимо ім'я локального адміністратора «Адміністратор» і пароль. Не вводьте ключ продукту перед клонуванням жорстких дисків. Після клонування можна ввести ключ продукту на кожному комп'ютері. Вказуємо часовий пояс і встановлюємо дату і

час. Встановлюємо драйвери на апаратне забезпечення комп'ютера з диска, що додається до комп'ютера.

Після встановлення Windows 10 приступаємо до клонування системи за допомогою програми «Norton Ghost» [6]:

- запускаємо «GhostCast Server» з групи «Symantec Ghost» на робочій станції, де встановлена система;
- вибираємо опцію «Create Image», вказуємо ім'я файлу «System.gho» та місце зберігання «C:\images\»;
- у цьому вікні введіть назву сесії «session_1»;
- натискаємо «MoreOptions» і встановлюємо параметр «Client Count» «1»;
- у «Client command line options» вибираємо «disk – 1», це означає створення образу для одного диска;
- вибираємо тип компресії «High» і очікуємо завершення процесу створення образу, на цьому етапі клієнтська машина завантажується з флеш-накопичувача або диска, наданого «Symantec», запускає клієнтську версію «Ghost» і чекає на появу сервера;
- запускаємо утиліту «Symantec Ghost Console»;
- переходимо на вкладку «Tasks» і вибираємо пункт «Створення нового завдання»;
- у полі «name» вікна, що відкриється, вводимо «System.gho»;
- вибираємо пункт «Clone» і в розділі «Target machine group/machine» вибираємо групу «Default», куди за замовчуванням додаються всі комп'ютери, жорсткі диски яких мають бути клоновані;
- переходимо на вкладку «Clone» і вибираємо опцію «Додати образу»;
- у вікні в полі «Name» вводимо ім'я «System.gho», в полі «Location» вказуємо шлях до образу;
- вибираємо пункт «Partition restore» і вказуємо в пункті «Destination Drive» – 1 і «Destination Partition» - 1;
- в «Source Partition» встановлюємо розділ 1 - NTFS;

- виділяємо пункт «Use Ghostwalker to perform a SID Change on the target machine» і переходимо на вкладку «Network»;
- встановлюємо «Use WOL when executing a task» та закриваємо вікно властивостей;
- виділяємо завдання «System.gho» і виконуємо його «Execute Task».

Після завершення клонування жорсткого диска на всіх робочих станціях необхідно виконати такі дії:

- ввести ліцензійний ключ продукту;
- настроїти автоматичне отримання IP-адреси і присвоїти доменне ім'я комп'ютера;
- підключіть комп'ютер безпосередньо до домену «Bank.local» на вкладці мережна ідентифікація властивостей робочої станції;
- встановити необхідне програмне забезпечення.

1.6.2 Вибір і встановлення серверної операційної системи

Windows 2008 Server - найшвидша, найнадійніша і найбезпечніша операційна система в сімействі серверних ОС Windows. Надійність цієї системи обумовлена наявністю інтегрованої інфраструктури, що забезпечує безпеку бізнес-інформації, а також надійність, доступність і масштабованість мережевої інфраструктури. Операційна система допомагає побудувати інфраструктуру бізнес-рішень, що покращують взаємодію зі співробітниками та партнерами. Це досягається завдяки інтегрованому серверу потокового передавання даних і мультимедіа, який допомагає прискорити і спростити створення динамічних вузлів інтернету та підвищити рівень безпеки, а також інтегрованому серверу додатків, який спрощує розробку, розгортання та керування веб-службами XML. Windows 2008 Server надає інструменти для розгортання, управління та використання мережевої інфраструктури, яка максимізує продуктивність, надаючи гнучкі засоби для адаптації розробки та впровадження до потреб організації та мережі, надаючи користувачам більшу автономію та можливість зменшити витрати на підтримку.

Встановлюємо Windows 2008 Server Enterprise Edition на контролері домену:

- встановіть у BIOS комп'ютера порядок завантаження із зовнішнього носія за допомогою флеш-накопичувача USB;
- встановіть флеш-накопичувач, що містить Windows Server 2008 Datacenter Edition;
- виберіть мову встановлення та регіональний стандарт і натискаємо кнопку «Далі»;
- виберіть опцію «Встановити»;
- введіть ліцензійний ключ, що поставляється на носії з дистрибутивом;
- виберіть версію Windows Server 2008 Datacenter Edition з архітектурою x64 для встановлення і натисніть кнопку «Далі»;
- ознайомтеся з ліцензійною угодою, позначте її галочкою і натисніть кнопку «Далі»;
- виберіть «Custom» -установку;
- виберіть жорсткий диск для встановлення;
- виберіть у меню пункт «Параметри» і позначте жорсткий диск;
- створіть у нерозміченому просторі жорсткого диска, на якому буде встановлено систему, розділ «C:» об'ємом 500 Гбайт і виберіть цей розділ;
- відформатуйте обраний розділ із файловою системою NTFS;
- дочекайтеся закінчення процесу встановлення.

При першому ввімкненні системи необхідно створити обліковий запис локального адміністратора: «Адміністратор» і встановіть пароль довжиною не менше 16 символів, що містить принаймні одну велику літеру і принаймні один розділовий знак. Після входу в систему необхідно встановити IP-адресу та доменне ім'я сервера. У вікні «Менеджер сервера», яке відкриється автоматично, виберіть «Налаштування мережі». Перейдіть до «Підключення по локальній мережі» → «Властивості» → «Версія інтернет-протоколу TCP/IPv4» → «Властивості». Виберіть «Використовувати наступну IP-адресу» і налаштуйте параметри інтерфейсу:

- IP-адреса: 192.168.0.250;
- маска підмережі: 255.255.0.0;
- основний шлюз: 192.168.0.249;
- DNS-сервер: 8.8.4.4.

Після підтвердження налаштувань зніміть прапорець і вимкніть опцію «Інтернет-протоколу TCP/IPv6». Далі, в менеджері серверів виберіть пункт «Вкажіть ім'я комп'ютера і домен». Натисніть «Змінити» і вкажіть доменне ім'я сервера як «Srv».

Далі серверу необхідно призначити роль контролера домену і встановити відповідні служби. Для цього у вікні «Завдання початкове налаштування» виберіть «Налаштувати цей сервер», а потім натисніть кнопку «Додати ролі». Відкриється майстер додавання серверних ролей, у полі «Ролі сервера» виберіть «Служби Active Directory», а потім натисніть кнопку «Далі». Після того, як всі пункти буде прийнято, призначте серверу роль контролер домену (створюється ліс з деревом доменів і нашим доменом «Bank.local»).

1.7 Політика використання IP-адрес для мережевих та інтернет-з'єднань

Політики DHCP використовуються для динамічного виділення адрес у мережі. Контролер домену виступає в ролі DHCP-сервера. Усі клієнти підключаються до нього і запитують IP-адресу, після чого адреса виділяється клієнту з урахуванням його MAC-адреси відповідно до табл. 1.5.

Обидва сервери системи обслуговування клієнтських запитів мають статичні адреси: 192.168.0.251 та 192.168.0.252. Контролер домену має статичну адресу 192.168.0.250. Внутрішній інтерфейс маршрутизатора має адресу 192.168.0.249. Ця адреса сприймається всіма вузлами мережі як шлюз. Маршрутизатор налаштований на переадресацію портів 6867 і 1099 на адресу

серверів системи обслуговування клієнтських запитів.. Це дозволяє клієнтам надсилати запити на сервери системи, коли вони отримують доступ до зовнішнього інтерфейсу маршрутизатора. DNS-сервер для всіх вузлів є 8.8.4.4 (публічний DNS від Google).

Таблиця 1.5 - Таблиця зіставлення MAC-адрес і IP-адрес

Доменне ім'я вузла	MAC-адрес	IP-адрес
ПК1	00:A7:56:0C:98:07	192.168.0.1
ПК2	00:A7:56:0C:98:08	192.168.0.2
ПК3	00:A7:56:0C:98:09	192.168.0.3
...		
ПК120	00:A7:56:0C:98:7F	192.168.0.120

Для підключення мережі до Інтернету може бути використана будь-яка з таких технологій: зовнішні виділені провідні канали, лінії АТС і DSL-модеми, 4G, бездротовий доступ через WIMAX, Frame Relay.

Порівняльна характеристика сучасних технологій, що використовуються для підключення до Інтернету, наведена в табл. 1.6. Виходячи з аналізу, наведеного в табл. 1.6, дротове з'єднання використовується за певними критеріями. Для цього підписується договір з провайдером.

Таблиця 1.6 – Порівняльна характеристика технологій підключення до Інтернету

Характеристика	Провідний канал	Лінія АТС та модем DSL	4G	WIMAX	Frame relay
Пропускна здатність	до 100 Мбіт/с	до 24 Мбіт/с	до 1 Гбіт/с	до 1 Гбіт/с	до 34,368 Мбіт/с

Продовження таблиці 1.6

Характеристика	Провідний канал	Лінія АТС та модем DSL	4G	WIMAX	Frame relay
Складність реалізації	низька	середня	низька	низька	середня
Реконфігурованість	ні	ні	так	так	ні
Вартість	низька	низька	висока	середня	середня
Надійність	середня	середня	низька	низька	середня

1.8 Забезпечення захисту даних у мережі

Захист даних у мережі є необхідною умовою для належного функціонування мережі. Мережева безпека включає захист комп'ютерів та іншого активного мережевого обладнання від хакерських атак, DOS-атак, вірусів, троянів, шпигунських та інших шкідливих програм.

Мережеву безпеку можна розділити на активну і пасивну. До активних засобів захисту належать політики безпеки Active Directory, апаратні брандмауери, антивірусне програмне забезпечення та шифрування трафіку.

Пасивні заходи безпеки включають регулярний аудит безпеки та програми, що запобігають доступу скомпрометованих облікових записів до мережевих ресурсів.

На прикладному рівні моделі OSI використовуються політики безпеки Active Directory, антивірусне програмне забезпечення та шифрування трафіку. Політики безпеки Active Directory встановлюються на сервері керування (контролері домену). Паролі повинні мати довжину не менше 12 символів,

містити цифри, символи, малі та великі літери, а також змінюватись кожні два місяці.

Контролюючи окремі події в системі, можна контролювати авторизацію дій користувачів в цілому, наприклад, доступ користувачів до ресурсів, і реалізувати її таким чином. Система визначає події, які потрібно перевіряти колективно, але, перевіряючи їх індивідуально, можна контролювати дії в цілому. Якщо подій багато, вони об'єднуються в групи (списки) за функціональними ознаками (або іншими спільними характеристиками). Створюється еталонна копія атрибутів цієї групи (список авторизованих випадків). Кожен список формує окремий рівень управління. Якщо еталонна копія та оригінал не збігаються, властивість системи вважається порушеною. Події під час роботи системи, що захищається, контролюються шляхом порівняння еталонної копії списку кожного рівня з еталонним значенням. У разі невідповідності генерується стан порушення безпеки системи, який може бути усунутий шляхом відновлення цілісності перевірених об'єктів системних файлів або об'єктів реєстру операційної системи, завершення події (наприклад, шляхом завершення несанкціонованих процесів, зупинки несанкціонованих дисків), перезапуску події (наприклад, шляхом запуску обов'язкових процесів) або за іншим сценарієм, передбаченим системою.

Встановіть антивірусне програмне забезпечення на кожную робочу станцію. Та виберіть Norton Internet Security [6], який забезпечує високий рівень антивірусного, анти шпигунського та анти троянського захисту. Перевага захисту на рівні додатків полягає в тому, що несанкціонована подія завжди може бути зафіксована, незалежно від того, чому вона сталася. Основним недоліком є те, що несанкціоновані події не можна запобігти на рівні програми.

Для захисту мережі на транспортному рівні використовується функціональність, що міститься в заголовках сегментів транспортних протоколів TCP і UDP. Оскільки будь-який мережевий застосунок використовує транспортні протоколи для доставки оброблюваних даних, а кожен клас застосунків має певний номер транспортного порту, цю характеристику може використати зловмисник для атаки на конкретний мережевий сервіс або може використати

мережевими адміністраторами для захисту мережевих служб і сервісів. Реалізація політик захисту пристроями транспортного рівня здійснюється за допомогою між мережевих екранів. Спроектвані мережеві брандмауери розташовуються на маршрутизаторах.

Мережевий рівень моделі OSI дозволяє встановити захищене з'єднання між двома комп'ютерами, розташованими в довільних точках мережі (IPSec). На цьому рівні реалізується фільтрація трафіку між внутрішньою та зовнішньою мережами, захист від несанкціонованого доступу із зовнішньої мережі до внутрішньої та маскування топології внутрішньої мережі. в діапазоні зарезервованих адрес Windows Server 2008, послуги моніторингу мережі та підтримки віртуальних вузлів. Цей сервіс завжди використовує вільний адресний простір мережі і створює свій власний віртуальний вузол (новий віртуальний вузол створюють одразу ж у разі відключення від мережі реально функціонуючого надійного вузла). Таким чином, ця служба замінює відсутні робочі станції, сервери, маршрутизатори тощо.

Алгоритм визначення доступності середовища базується на тому, що всі вузли, підключені до середовища передачі, завжди його прослуховують. Ця особливість використовується зломисниками для організації різноманітних атак на комп'ютерні мережі. Завдання протоколу канального рівня вважається виконаним, якщо взаємодіючі вузли знають апаратну адресу один одного і можуть інкапсулювати мережеві пакети в кадр канального рівня, ідентифікований комутатором з MAC-адресою вузла.

Для захисту мережі на канальному рівні необхідно заборонити обмін даними «один-ко-многим». Також необхідно визначити, яким робочим станціям дозволено конфігурувати комутаційні пристрої канального рівня. Мережеві адміністратори повинні регулярно контролювати відповідність конфігурації комутатора створеній ними політиці безпеки і відстежувати мережеву активність для виявлення джерела аномальної кількості ширококомовних запитів.

Для запобігання можливості витоку інформації через фізичні шляхи (дротові телекомунікаційні лінії, міські комунікації, обладнання з вихідним

побічним випромінюванням, радіопередавачі) необхідно передбачити спеціальні фізико-технічні заходи. Для цього мережеві кабелі повинні прокладатися в прихованій кабельній каналізації або в кабельних каналах, які можуть бути опломбовані нестираємими пломбами. Кабельні сегменти, які використовуються для з'єднання всіх вузлів комп'ютерної мережі, мають бути зосереджені в одній комутаційній панелі. Конфігурацію фізичних зв'язків між вузлами має контролювати виключно комутаційна панель. Комутаційна панель має бути встановлена в комутаційну шафу, що замикається. Доступ до комутаційної шафи має бути строго обмежений.

Резервне копіювання системного розділу контролера домену на ПК 85 використовується для підвищення надійності роботи мережі. Резервне копіювання даних здійснюється автоматично на регулярній основі, раз на тиждень. Для цього використовується програма «Handy Backup Server». Створення та налаштування завдань резервного копіювання, відновлення та синхронізації здійснюється за допомогою майстра завдань. Ви можете вибирати між повним, інкрементним і диференціальним резервним копіюванням, багаторівневим стисненням ZIP і шифруванням Blowfish. Вбудований планувальник дає змогу виконувати завдання в заданий час або у зв'язку з певними подіями. Програма може працювати як служба Windows, не вимагаючи входу користувача в систему. Звіти про результати виконання кожного завдання можуть бути надіслані на зазначену адресу електронної пошти. Handy Backup Server встановлюється на машину адміністратора і дає змогу керувати операціями резервного копіювання, синхронізації та відновлення на мережевих серверах і робочих станціях, операції відновлення на мережевих серверах і робочих станціях. Крім копіювання файлів, він дає змогу створювати образи дисків сервера і мережевих комп'ютерів, резервувати дані MS Exchange і Lotus Notes на сервері й мережевих комп'ютерах, а також резервувати ODBC-сумісні бази даних на сервері Handy Клієнт резервного копіювання (Backup Network Workstation) встановлюються на кожний користувацький комп'ютер, дані якого треба резервувати; він працює в режимі

служби Windows, не має графічного інтерфейсу й дає змогу серверу збирати дані з користувацьких машин для дає змогу серверу збирати дані для копіювання.

Через певні проміжки часу всі клієнти зв'язуються з сервером, щоб переконатися, що вони готові до резервного копіювання.

1.9 Проектування апаратного забезпечення серверів обслуговування мікротранзакцій

Особливістю системи є обробка мікротранзакцій з великою кількістю клієнтів. Така обробка має бути багатопотоковою, тому в системі мають використовуватися восьмиядерні процесори. Кожне ядро потребує 2 Гбайт оперативної пам'яті. Система не потребує відеоадаптера (достатньо адаптера, вбудованого в материнську плату). Мережевий адаптер також має бути вбудований у системну плату і підтримувати стандарт Gigabit Ethernet. Обрана конфігурація системи наведена в табл. 1.7.

Таблиця 1.7 Конфігурації серверів для систем обслуговування клієнтських запитів

Материнська плата	Asus P11C-X
Процесор	Intel Core i5 Coffee Lake Refresh, 6 ядер 3.7 ГГц
Відеоадаптер	Інтегрований на материнській платі
Оперативна пам'ять	16 Гб DDR-4 2666 МГц
Дисковий накопичувач	HDD 10 Тб Western Digital Ultrastar DC HC330 10TB 7200rpm 256MB
Блок живлення	Straight Power 1200 Вт
Клавіатура/миша	Клавіатура PS/2, USB миша

Продовження таблиці 1.7

Мережевий адаптер	Інтегрований адаптер Gigabit Ethernet на материнській платі
Монітор	«19» Samsung S19A300N LED, 1366x768 (16:9)

Обидва сервери в системі будуть налаштовані таким чином.

1.9.1. Вибір джерела безперебійного живлення

Джерело безперебійного живлення має забезпечувати надійну роботу системи обслуговування клієнтських запитів упродовж щонайменше однієї години в разі вимкнення лінії електропередачі. Сервери та лінії підключення до Інтернету: комутатори та маршрутизатори в серверній кімнаті мають бути підживлені.

Сумарна пікова споживана потужність системи має такий вигляд:

$$P = P_{сервер1} + P_{сервер2} + P_{коммутатор} + P_{маршрутизатор} \text{ (Вт)}. \quad (1.4)$$

$$P = 600 + 600 + 40 + 60 = 1300 \text{ Вт}.$$

Джерело безперебійного живлення APC Smart-UPS 2000VA 230V з наступними характеристиками: номінальна вхідна напруга 230В, тип - подвійне перетворення (online), вхідна частота 50/60 Гц (автоматичне визначення), спотворення вихідної напруги <3%, сумісний з холодним стартом, діапазон вхідної напруги 180-270В під час роботи від мережі, максимальна вихідна потужність 1400Вт, номінальна вихідна напруга 230В, акумулятор 12В, 4 x 9 Ач. Необслуговувана герметична свинцево-кислотна батарея. Можливе під'єднання додаткового зовнішнього акумуляторного блоку SURT48XLBP (купується окремо); ККД за повного навантаження 90%; типовий час заряджання 2 години; забезпечує живлення під'єднаних навантажень від мережі в разі перевантаження або несправності; автоматичний запуск під'єданого обладнання після відновлення живлення; автоматичне вбудоване тестування; регулярні процедури самодіагностики батареї щодо потреби заміни; своєчасне визначення стану

батарей, регулювання частоти та напруги, захист навантажень від перенапруг; захист від перенапруг; захист від перегрівів, перебоїв у роботі, перебоїв у роботі, перебоїв у роботі.

1.9.2 Вибір системи кондиціонування

Прецизійні кондиціонери використовуються для кондиціонування серверних приміщень. Прецизійні кондиціонери є спеціалізованим кліматичним обладнанням, що складається з холодильного агрегату (компресорного блоку), системи теплообмінників, вентиляторів, фільтрів, повітропроводів, зволожувачів повітря і системи контролю та управління режимами роботи обладнання. Прецизійні кондиціонери розраховані на особливі умови експлуатації та можуть працювати 24 години на добу, цілий рік, підтримуючи задану температуру в межах $\pm 0,5^{\circ}\text{C}$ і вологість у межах $\pm 2,5\%$. Висока точність підтримання параметрів досягається завдяки оснащенню прецизійних кондиціонерів прецизійною мікропроцесорною системою керування.

У серверній кімнаті знаходяться три сервери (контролер домену і два сервери для системи обслуговування клієнтських запитів), комутатори і маршрутизатори. Сумарна споживана потужність, розрахована вище, становить 1300 Вт. Отже, таку саму потужність необхідно розсіювати.

Вимоги до систем кондиціонування повітря наведено нижче:

- відносна вологість повітря - 20-70% (75% або менше в холодну погоду, 65% або менше в теплу погоду при температурі 25°C і 70% або менше при температурі 24°C і нижче);
- кімнатна температура $20^{\circ} \pm 2^{\circ}\text{C}$ (не вище 24°C);
- запиленість повітря в приміщеннях не повинна перевищувати 0,75 мг/м³, а розмір частинок не повинен перевищувати 3 мкм (пил, сажа, дим, спори та азбест);
- оптимальна швидкість вітру - 0,2 м/с (менше 0,3 м/с в холодну погоду і менше 0,5 м/с в теплу);
- допустимі рівні шуму не повинні перевищувати 65 дБ. Допустимі рівні вібрації не повинні перевищувати амплітуду 0,1 мм і частоту 25 Гц;

- система кондиціонування серверного приміщення має бути повністю незалежною від інших систем кондиціонування;
- у холодну пору року системи кондиціонування не повинні допускати утворення конденсату;
- для врахування цих вимог використовується система Stulz CVS40 [7].

2 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТІВ СИСТЕМИ БАНКУ

2.1 Встановлення та налаштування MS SQL Server

Встановимо і налаштуємо Microsoft Sql Server 2022 на сервері системи [8]. Для цього необхідно встановити пакети Microsoft .NET Framework 4, Windows Installer 4.5 і Windows PowerShell 1.0. Процес встановлення включає такі кроки:

- запустить програму встановлення;
- у розділі «Планування» виберіть пункт «Перевірка конфігурації»;
- натисніть на кнопку «Показати подробиці», щоб переконатися в успішному проходженні всіх перевірок. Якщо виявлено помилки, виправте їх і виконайте перевірки ще раз за допомогою кнопки «Включить знову». Після цього вікно закривається кнопкою «ОК»;
- потім у вікні програми встановлення виберіть розділ «Встановлення» і виберіть пункт «Нове встановлення ізольованого SQL Server або додавання компонентів до наявного екземпляра»;
- у результаті розпочнеться процес перевірки системи перед встановленням. Як і в кроці 3, для перегляду докладного звіту про перевірку можна скористатися кнопкою «Показати подробиці» і натиснути кнопку «ОК» для завершення;
- ведіть ліцензійний ключ і натисніть кнопку «Далі»;
- прочитайте та прийміть ліцензійну угоду;

- запустити установку за допомогою кнопки «Встановити» в наступному вікні;
- виберіть компоненти, необхідні для встановлення; кількість і конфігурація компонентів можуть відрізнятися залежно від редакції SQL Server 2022 (рис. 2.1).

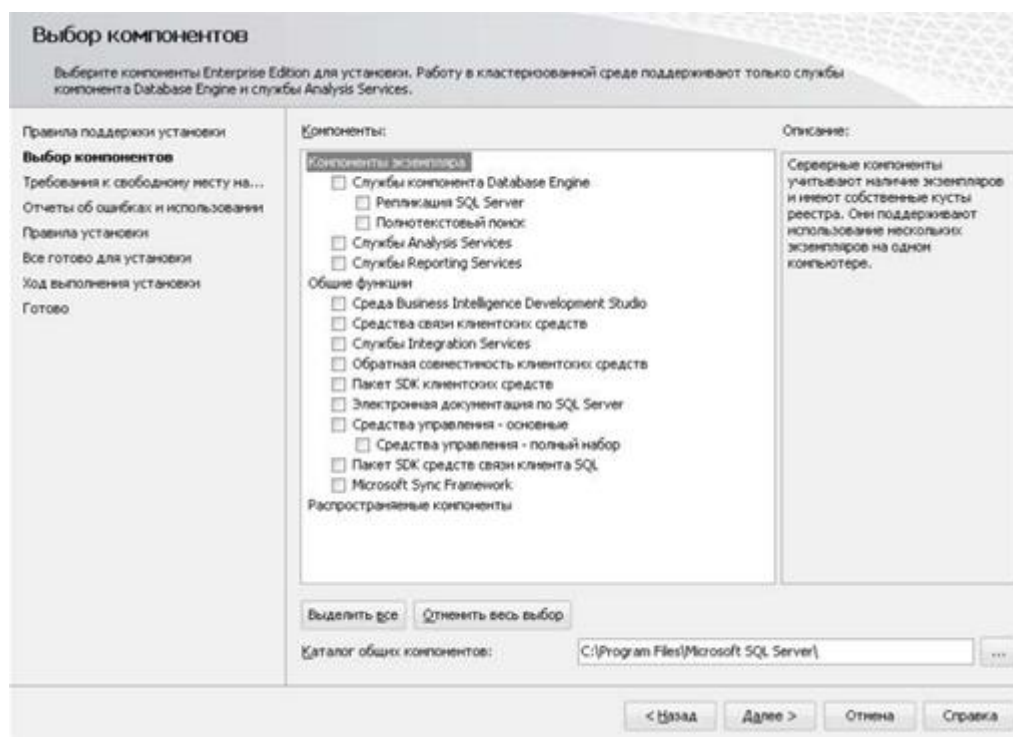


Рисунок 2.1 - Вибір компонентів MS SQL Server 2022

Проаналізуємо список компонентів.

Функція екземпляра - це набір серверних компонентів для окремих екземплярів. Кожен з цих компонентів має свою окрему гілку реєстру. До них відносяться наступні:

- Database Engine Service - містить компонент Database Engine, який є основною службою для зберігання, захисту та обробки даних. По суті, це ядро SQL Server, що використовує . Ці служби забезпечують керований доступ, швидку обробку транзакцій і підтримують високий рівень доступності. Встановлення служби Database Engine є обов'язковим;

- реплікація SQL Server - відповідає за можливість реплікації даних;
- Full-Text Search - відповідає за швидкий текстовий пошук за наявними даними. Інакше кажучи, він маніпулює індексами і зберігає значення даних і покажчики на рядки, в яких ці дані зустрічаються. Іншими словами, він відповідає за технологію індексного сервера Microsoft.

Analysis Services - це набір інструментів, що підтримують аналітичну обробку (OLAP) та інтелектуальний аналіз даних; підтримується тільки в редакції Enterprise. Встановлюємо їх.

Reporting Services - це набір компонентів, інструментів і програмних інтерфейсів для розробки та використання вбудованих звітів у контрольованому середовищі. Набір містить засоби розроблення, конфігурування та керування, а також програму перегляду звітів. Програмні інтерфейси включають простий протокол доступу до об'єктів (SOAP), кінцеві точки URL і Windows Management Instrumentation (WMI) для простої інтеграції з веб-додатками і порталами.

Загальні функції - використовуються всіма екземплярами сервера:

- Business Intelligence Development Studio - це середовище для розроблення бізнес-рішень, включно з проектами Analysis Services, Integration Services і Reporting Services;
- Connection client tools - інструменти для організації взаємодії клієнт-сервер;
- Integration Services - використовуються для перетворення, інтеграції та переміщення даних між різними типами бізнес-додатків. Не написавши жодного рядка коду, можна створювати рішення Integration Services, що дають змогу розв'язувати складні бізнес-завдання, керувати базами даних SQL Server і копіювати об'єкти SQL Server між екземплярами;
- засоби управління включають підтримку середовища Management Studio для компонентів Database Engine, утиліту командного рядка SQLCMD і провайдер SQL Server PowerShell;
- Microsoft Sync Framework - це комплексна платформа синхронізації, що підтримує спільну роботу та автономний доступ до додатків, служб і

пристроїв. Платформа використовує технології та інструменти для підтримки переміщення даних, спільного використання та автономного доступу;

– Redistributable Features - набір компонентів, що встановлюються в міру необхідності: звіти про помилки, рідний клієнт SQL Server, MSXML, SQL Server Compact Workbench, середовище ADO.NET, SQL Server Explorer.

Для переходу до наступного кроку виберіть параметри установки для екземпляра за замовчуванням і залиште всі параметри за замовчуванням. На наступному кроці відображається ємність, займана сервером бази даних, і його адреса; натисніть кнопку «Далі»; на вкладці «Параметри сортування» вкажіть необхідні значення і натисніть кнопку «Далі». Залиште все без змін. Виберіть спосіб аутентифікації на сервері за логіном і паролем. Потім перейдіть на вкладку «Каталоги даних» і задайте необхідні каталоги (наприклад, кореневий каталог даних, каталог системної бази даних, каталог користувацької бази даних). (рис. 2.2).

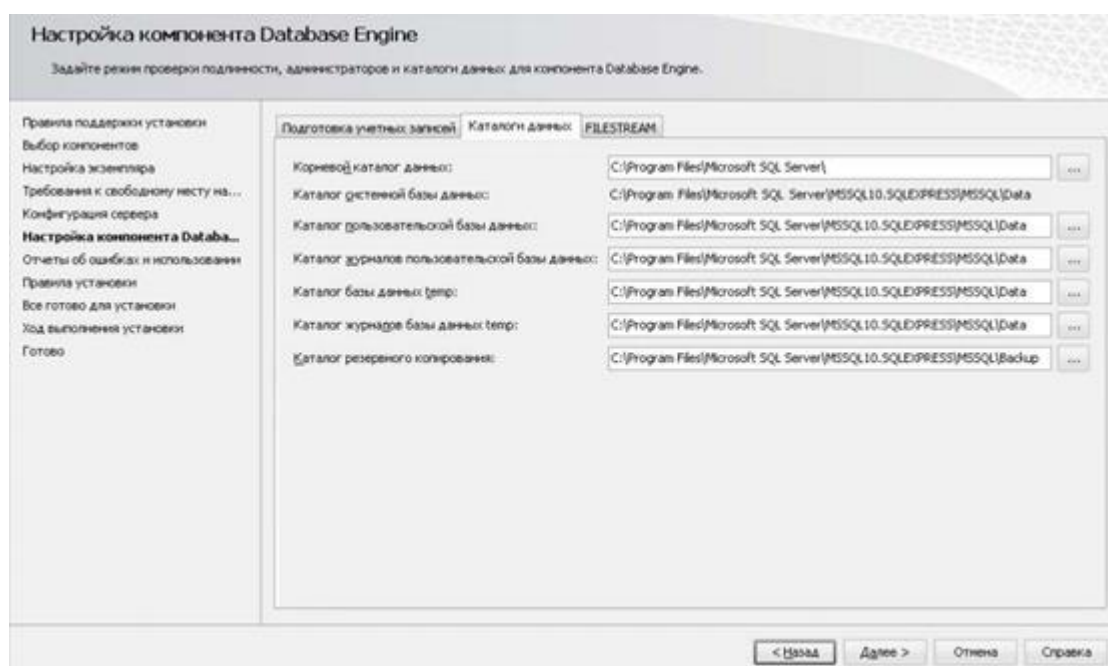


Рисунок 2.2 - Конфігурація каталогу даних MS SQL Server

Встановіть Analysis Services та Reporting Services і натисніть кнопку Next. Налаштуйте механізм надсилення помилок на MS SQL Server. Перевірте ще раз, і якщо все гаразд, переходьте до наступного кроку. Переконайтеся, що компоненти, необхідні для встановлення, обрано правильно, і натисніть кнопку «Встановити». Після завершення роботи програми інсталяції та відображення процесу інсталяції перейдіть до наступного кроку і натисніть кнопку «Закрити» для завершення роботи.

Після встановлення та налаштування сервера баз даних необхідно встановити SQL Server Management Studio [9].

Під час входу в SQL Server Management Studio з'являється вікно аутентифікації, як показано нижче (рис. 2.3).

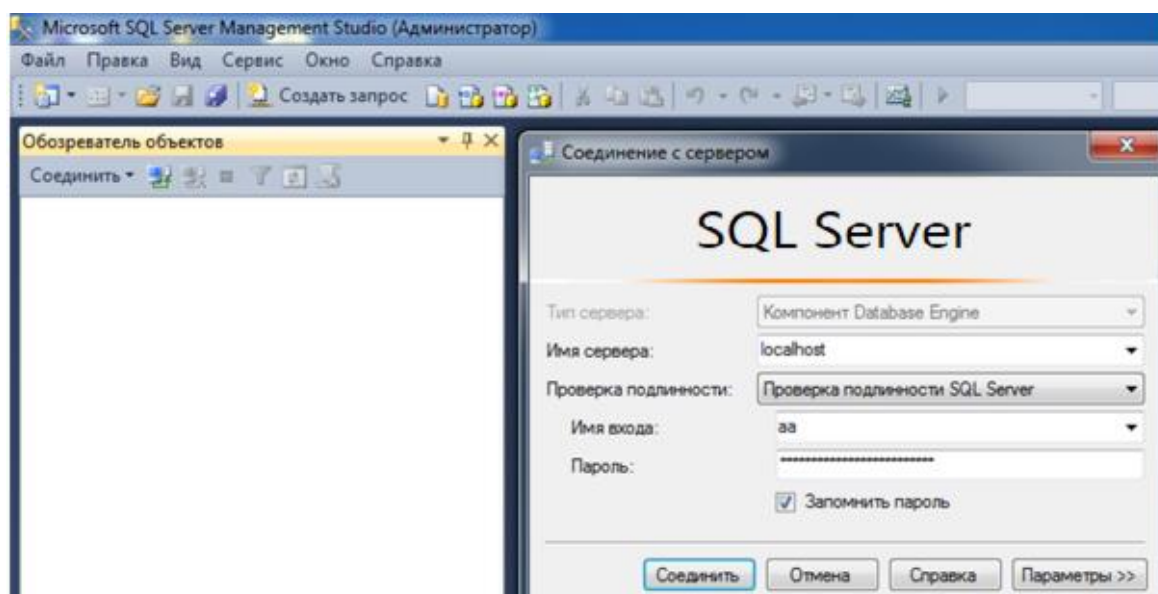


Рисунок 2.3 - Экран аутентифікації SQL Server Management Studio 2022

Спочатку виберіть тип сервера, а потім сам сервер у полі «SQL Server» (за необхідності вкажіть IP-адресу). Потім необхідно вказати режим аутентифікації SQL Server. При його виборі буде запропоновано ввести ім'я користувача і пароль. У цьому випадку «sa» - це перший обліковий запис адміністратора, за допомогою якого створюється новий обліковий запис користувача.

2.2 Проектування банківської бази даних

Після встановлення сервера системи створюється обліковий запис aa, який за замовчуванням представляє адміністратора системи: BankAdmin - обліковий запис адміністратора конструктора бази даних; BankClient - обліковий запис конструктора бази даних, який підключається до бази даних через інтерфейс програми та виконує основні операції. Це обліковий запис клієнта, який виконує операції. Процес створення облікового запису показано на рисунку 2.4.

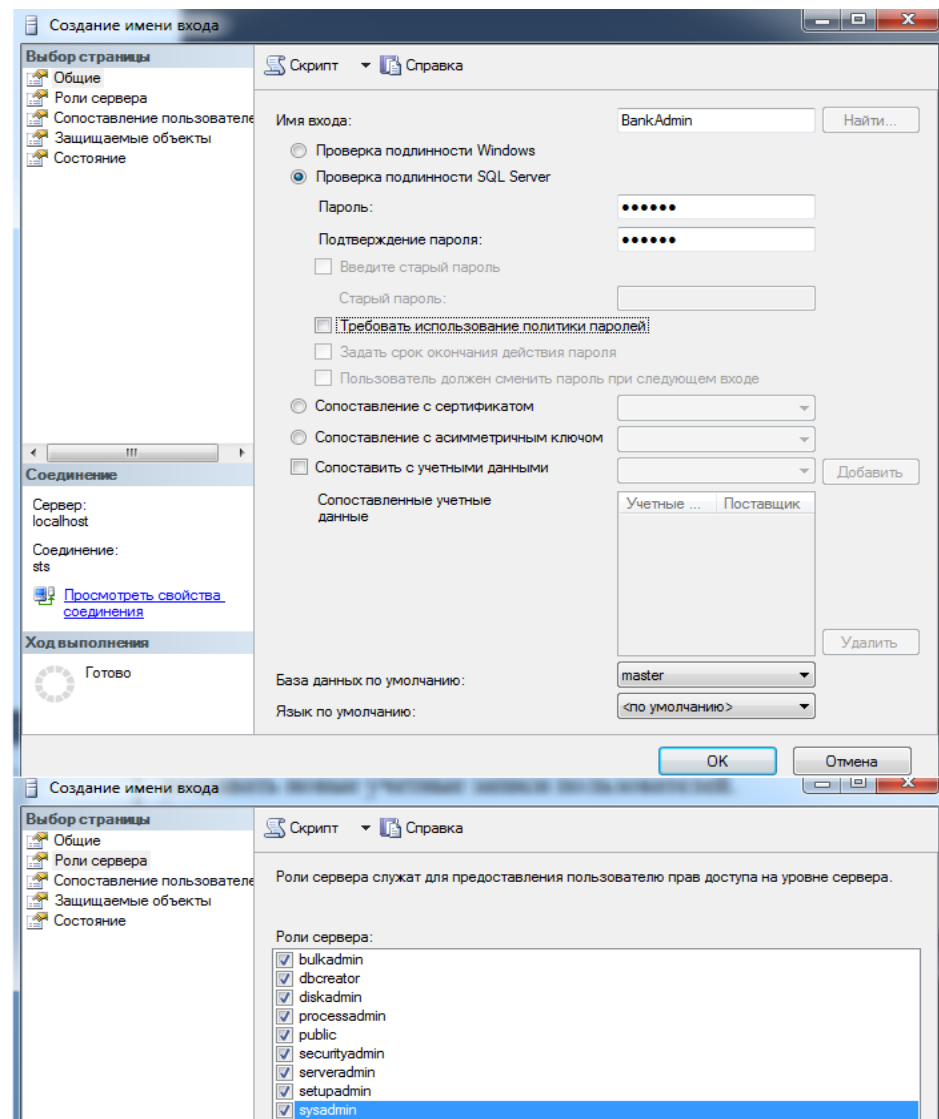


Рисунок 2.4 - Створення облікового запису в MS SQL Server

Коли обліковий запис BankAdmin має доступ до MS SQL Server, необхідно створити базу даних облікових записів клієнтів, які здійснюють мікротранзакції для таких платежів, як оплата комунальних послуг та мобільного зв'язку. Також необхідно створити базу даних комунальних підприємств, на які клієнти перераховують кошти. Для цього виконується наступний SQL-запит.

База даних платників податків. З цієї бази даних списуються кошти, перераховані на рахунки компаній, що надають послуги з електропостачання та мобільного зв'язку. База даних містить таблицю Accounts з полями id - первинний ключ, summ - сума коштів на рахунку, прізвище, ім'я, логін та відповідні паролі для доступу до платіжної системи можна побачити на лістингу 2.1.

Лістинг 2.1 – База даних платників податків

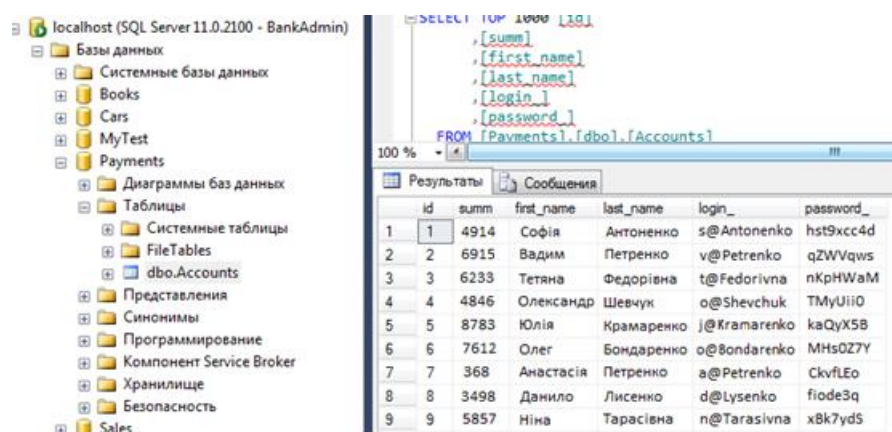
```
CREATE DATABASE Payments
use Payments
create table Accounts
(
    id int identity(1,1) primary key not null,
    summ float not null,
    first_name nvarchar(50) not null,
    last_name nvarchar(50) not null,
    login_ nvarchar(50) not null,
    password_ nvarchar(50) not null
)
Insert into Accounts(summ, first_name, last_name, login_,
password_) values
(4914, 'Софія', 'Антоненко', 's@Antonenko', 'hst9xcc4d')
Insert into Accounts(summ, first_name, last_name, login_,
password_) values
(6915, 'Вадим', 'Петренко', 'v@Petrenko', 'qZWVqws')
Insert into Accounts(summ, first_name, last_name, login_,
password_) values
(6233, 'Тетяна', 'Федорівна', 't@Fedorivna', 'nKpHWaM')
Insert into Accounts(summ, first_name, last_name, login_,
password_) values
(4846, 'Олександр', 'Шевчук', 'o@Shevchuk', 'TMyUii0')
Insert into Accounts(summ, first_name, last_name, login_,
password_) values
(8783, 'Юлія', 'Крамаренко', 'j@Kramarenko', 'kaQyX5B')
Insert into Accounts(summ, first_name, last_name, login_,
password_) values
(7612, 'Олег', 'Бондаренко', 'o@Bondarenko', 'MHs0Z7Y')
Insert into Accounts(summ, first_name, last_name, login_,
password_) values
(368, 'Анастасія', 'Петренко', 'a@Petrenko', 'CkvfLEo')
```

```

Insert into Accounts(summ, first_name, last_name, login_,
password_) values
(3498, ' Данило', ' Лисенко', 'd@Lysenko ', ' fiode3q')
Insert into Accounts(summ, first_name, last_name, login_,
password_) values
(5957, ' Ніна', ' Тарасівна', 'n@Tarasivna', ' xBk7ydS')

```

Щоб перевірити, що база даних і таблиці були успішно створені, запусимо стандартний скрипт, який вибирає перші 1000 рядків з таблиці. Результати показано на рисунку 2.5.



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Server Enterprise' tree is expanded to 'Tables' under the 'dbo' schema, showing the 'Accounts' table. On the right, a query window displays the following SQL query:

```

SELECT TOP 1000 [id],
[first_name],
[last_name],
[login_],
[password_]
FROM [Payments].[dbo].[Accounts]

```

Below the query, the 'Results' tab shows the first 10 rows of the 'Accounts' table:

	id	summ	first_name	last_name	login_	password_
1	1	4914	Софія	Антоненко	s@Antonenko	hst9xcc4d
2	2	6915	Вадим	Петренко	v@Petrenko	qZWVqws
3	3	6233	Тетяна	Федорівна	t@Fedorivna	nKpHWaM
4	4	4846	Олександр	Шевчук	o@Shevchuk	TMyUiiO
5	5	8783	Юлія	Крамаренко	j@Kramarenko	kaQyX5B
6	6	7612	Олег	Бондаренко	o@Bondarenko	MHs0Z7Y
7	7	368	Анастасія	Петренко	a@Petrenko	CkvfLEo
8	8	3498	Данило	Лисенко	d@Lysenko	fiode3q
9	9	5857	Ніна	Тарасівна	n@Tarasivna	xBk7ydS

Рисунок 2.5 – Результати вибірки за таблицею облікових записів клієнтів банку

База даних ЦП "Водоканал" являє собою таблицю з такими полями: id - первинний ключ, account - розрахунковий рахунок абонента, sum - кошти на розрахунковому рахунку, name - прізвище абонента, лістинг 2.2.

Лістинг 2.2 – База даних ЦП "Водоканал"

```

CREATE DATABASE Vodokanal
use Vodokanal
create table Accounts
(
    id int identity(1,1) primary key not null,
    account int not null,
    summ float not null,
    first_name nvarchar(50) not null,
    last_name nvarchar(50) not null
)

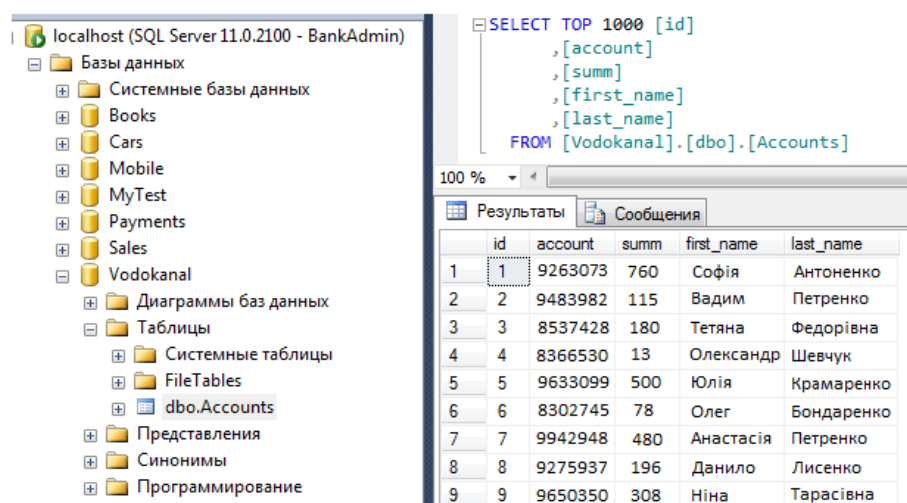
```

```

Insert into Accounts(account, summ, first_name, last_name)
values
('9263073', 760, 'Софія', 'Антоненко')
Insert into Accounts(account, summ, first_name, last_name)
values
('9483982', 115, 'Вадим', 'Петренко')
Insert into Accounts(account, summ, first_name, last_name)
values
('8537428', 180, 'Тетяна', 'Федорівна')
Insert into Accounts(account, summ, first_name, last_name)
values
('8366530', 13, 'Олександр', 'Шевчук')
Insert into Accounts(account, summ, first_name, last_name)
values
('9633099', 500, 'Юлія', 'Крамаренко')
Insert into Accounts(account, summ, first_name, last_name)
values
('8302745', 78, 'Олег', 'Бондаренко')
Insert into Accounts(account, summ, first_name, last_name)
values
('9942948', 480, 'Анастасія', 'Петренко')
Insert into Accounts(account, summ, first_name, last_name)
values
('9275937', 196, 'Данило', 'Лисенко')
Insert into Accounts(account, summ, first_name, last_name)
values
('9650350', 308, 'Ніна', 'Тарасівна')

```

Приклад абонента "Водоканалу" наведено на рис. 2.6.



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Vodokanal' database is expanded, showing the 'dbo.Accounts' table. On the right, a query window displays the following SQL query:

```

SELECT TOP 1000 [id]
      ,[account]
      ,[summ]
      ,[first_name]
      ,[last_name]
FROM [Vodokanal].[dbo].[Accounts]

```

Below the query, the 'Results' tab shows a table with 9 rows of data. The first row is highlighted.

	id	account	summ	first_name	last_name
1	1	9263073	760	Софія	Антоненко
2	2	9483982	115	Вадим	Петренко
3	3	8537428	180	Тетяна	Федорівна
4	4	8366530	13	Олександр	Шевчук
5	5	9633099	500	Юлія	Крамаренко
6	6	8302745	78	Олег	Бондаренко
7	7	9942948	480	Анастасія	Петренко
8	8	9275937	196	Данило	Лисенко
9	9	9650350	308	Ніна	Тарасівна

Рисунок 2.6 - Результати вибірки з таблиці абонентів «Водоканалу»

База даних "Мобільні Телесистеми" містить таблиці з ідентифікатором (первинний ключ), номером телефону, що передається, балансом рахунку, прізвищем та ім'ям абонента; на відміну від бази даних "Водоканалу", у цій базі ім'я та прізвище можуть бути незаповненими. Це пов'язано з тим, що абонент може використовувати передплатений зв'язок без договору і перевірки особи. Частина коду бази даних зображено на лістингу 2.3.

Лістинг 2.3 – База даних "Мобільні Телесистеми"

```
CREATE DATABASE Mobile
use Mobile
create table Operators
(
    id int identity(1,1) primary key not null,
    number int not null,
    summ float not null,
    first_name nvarchar(50),
    last_name nvarchar(50)
)

Insert into Operators(number, summ) values ('0683377286', 60)
Insert into Operators(number, summ) values ('0683377750', 30)
Insert into Operators(number, summ) values ('0680441359', 102)
Insert into Operators(number, summ) values ('0680441360', 115)
Insert into Operators(number, summ) values ('0680441349', 57)
Insert into Operators(number, summ) values ('0680441358', 458)
Insert into Operators(number, summ) values ('0680441391', 10)
Insert into Operators(number, summ) values ('0680441325', 69)
Insert into Operators(number, summ) values ('0680441376', 30)
```

2.3 Розробка системних інтерфейсів

Розробити інтерфейси платіжної системи мовою C# у середовищі Microsoft Visual Studio. Для цього проект створюється з використанням фрейм ворку Windows Forms [10], який дає змогу будувати графічні інтерфейси на основі векторної графіки.

Після активації платіжної системи клієнт має побачити вікно з інтерфейсом для переказу грошей зі свого рахунку на рахунок комунальної служби або оператора мобільного зв'язку. Інтерфейс системи показано на рис. 2.7.

Рисунок 2.7 - Інтерфейси платіжних систем

Для програмування такого інтерфейсу необхідно написати код для списків, що випадають, полів введення, статичного тексту, кнопок, прапорця, та індикаторів виконання. Програмуємо випадаючі списки (ComboBox), лістинг 2.4

Лістинг 2.4 – Випадаючі списки

```
this.cbService = new System.Windows.Forms.ComboBox();
this.cbService.DropDownStyle=
System.Windows.Forms.ComboBoxStyle.DropDownList;
this.cbService.FormattingEnabled = true;
this.cbService.Location = new System.Drawing.Point(56, 30);
this.cbService.Name = "cbService";
this.cbService.Size = new System.Drawing.Size(165, 20);
this.cbService.TabIndex = 0;
```

Створюємо об'єкт `combo box`, вказуємо його стиль (список, що випадає) і задаємо його положення на головній формі. Далі запрограмуємо поля введення (текстові поля), лістинг 2.5

Лістинг 2.5 – Поля введення

```
this.tbSum = new System.Windows.Forms.TextBox();
this.tbLogin = new System.Windows.Forms.TextBox();
this.tbPassword = new System.Windows.Forms.TextBox();
this.tbPayAccount = new System.Windows.Forms.TextBox();
this.tbAccount = new System.Windows.Forms.TextBox();
// tbAccount
this.tbAccount.Location = new System.Drawing.Point(56, 74);
this.tbAccount.Name = "tbAccount";
this.tbAccount.Size = new System.Drawing.Size(165, 20);
this.tbAccount.TabIndex = 1;
// tbSum
this.tbSum.Location = new System.Drawing.Point(56, 119);
this.tbSum.Name = "tbSum";
this.tbSum.Size = new System.Drawing.Size(165, 20);
this.tbSum.TabIndex = 2;
// tbLogin
this.tbLogin.Location = new System.Drawing.Point(167, 173);
this.tbLogin.Name = "tbLogin";
this.tbLogin.Size = new System.Drawing.Size(197, 20);
this.tbLogin.TabIndex = 3;
// tbPassword
this.tbPassword.Location = new System.Drawing.Point(167, 202);
this.tbPassword.Name = "tbPassword";
this.tbPassword.Size = new System.Drawing.Size(197, 20);
this.tbPassword.TabIndex = 4;
// tbPayAccount
this.tbPayAccount.Location = new System.Drawing.Point(167,
229);
this.tbPayAccount.Name = "tbPayAccount";
this.tbPayAccount.Size = new System.Drawing.Size(197, 20);
this.tbPayAccount.TabIndex = 5;
```

Програмування кнопки, лістинг 2.6. Особливістю є необхідність зазначення події, яка відбувається як реакція на натискання кнопки.

Лістинг 2.6 – Програмування кнопки

```
this.bPay = new System.Windows.Forms.Button();
this.bPay.Location = new System.Drawing.Point(95, 283);
this.bPay.Name = "bPay";
this.bPay.Size = new System.Drawing.Size(74, 22);
this.bPay.TabIndex = 6;
```

```
this.bPay.Text = "Pay";
this.bPay.UseVisualStyleBackColor = true;
this.bPay.Click += new System.EventHandler(this.bPay_Click);
```

Тому метод `bPay_Click` викликається щоразу, коли користувач натискає на кнопку. Програмуємо прапорець, лістинг 2.7.

Лістинг 2.7 – Програмування прапорця

```
this.cPrint = new System.Windows.Forms.CheckBox();
this.cPrint.AutoSize = true;
this.cPrint.Location = new System.Drawing.Point(234, 281);
this.cPrint.Name = "cPrint";
this.cPrint.Size = new System.Drawing.Size(93, 16);
this.cPrint.TabIndex = 7;
this.cPrint.Text = "Printing check";
this.cPrint.UseVisualStyleBackColor = true;
```

Поставте галочку, щоб відображати підтвердження (квитанцію) після оплати.

Коли мікро транзакція завершена, індикатор виконання (progress bar) заповнюється, лістинг 2.8.

Лістинг 2.8 – Індикатор виконання

```
this.progressBar = new System.Windows.Forms.ProgressBar();
this.progressBar.Location = new System.Drawing.Point(57, 330);
this.progressBar.Name = "progressBar";
this.progressBar.Size = new System.Drawing.Size(361, 22);
this.progressBar.TabIndex = 8;
```

Статичні мітки, що вказують користувачеві, куди і що вводити, програмуються таким чином, лістинг 2.9.

Лістинг 2.9 – Програмування статичних міток

```
this.label1 = new System.Windows.Forms.Label();
this.label2 = new System.Windows.Forms.Label();
this.label3 = new System.Windows.Forms.Label();
this.label4 = new System.Windows.Forms.Label();
this.label6 = new System.Windows.Forms.Label();
this.label7 = new System.Windows.Forms.Label();
this.label11.AutoSize = true;
this.label11.Location = new System.Drawing.Point(57, 5);
```

```

this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(42, 12);
this.label1.TabIndex = 9;
this.label1.Text = "Service";
// label2
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(57, 52);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(95, 12);
this.label2.TabIndex = 10;
this.label2.Text = "Checking account";
// label3
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(57, 101);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(29, 12);
this.label3.TabIndex = 11;
this.label3.Text = "Sum";
// label4
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(57, 172);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(34, 12);
this.label4.TabIndex = 12;
this.label4.Text = "Login";
// label6
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(58, 201);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(54, 12);
this.label6.TabIndex = 14;
this.label6.Text = "Password";
// label7
this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(60, 230);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(91, 12);
this.label7.TabIndex = 15;
this.label7.Text = "Personal account";

```

Далі слід запрограмувати код, що керує розмірами, зовнішнім виглядом і положенням головної форми на екрані, лістинг 2.10.

Лістинг 2.10 – Зовнішній вигляд головної форми

```

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(460, 391);
this.Controls.Add(this.label7);
this.Controls.Add(this.label6);

```

```

this.Controls.Add(this.label4);
this.Controls.Add(this.label3);
this.Controls.Add(this.label2);
this.Controls.Add(this.label1);
this.Controls.Add(this.progressBar);
this.Controls.Add(this.cPrint);
this.Controls.Add(this.bPay);
this.Controls.Add(this.tbPayAccount);
this.Controls.Add(this.tbPassword);
this.Controls.Add(this.tbLogin);
this.Controls.Add(this.tbSum);
this.Controls.Add(this.tbAccount);
this.Controls.Add(this.cbService);
this.FormBorderStyle=
System.Windows.Forms.FormBorderStyle.FixedSingle;
this.MaximizeBox = false;
this.Name = "Form";
this.Text = "Payment terminal";
this.ResumeLayout(false);
this.PerformLayout();

```

Метод `PerformLayout` малює форму і відображає на ній компоненти.

2.4 Аналіз технологій доступу до баз даних ADO.NET

Для зв'язку з базами даних NET Framework розробив технологію ADO.NET. Спочатку кожен постачальник баз даних надавав власні функції для роботи з базами даних, і не було жодного стандарту для цих функцій. Оскільки для зв'язку з кожною базою даних використовувалися різні функції, кожен розробник витрачав багато часу на знайомство і занурення в тонкощі API конкретної бази даних. Це призвело до розробки уніфікованої технології доступу до різних баз даних. В результаті активних досліджень з'явилася технологія ODBC (Open DataBase Connectivity), яка забезпечила прошарок між клієнтом і сервером бази даних, що приховує всі складнощі. ODBC є найстарішою технологією доступу до даних, вона надає широкий і гнучкий функціонал для роботи з реляційними джерелами даних. Недоліками цієї технології є складність

використання і неможливість реалізації додатків з багатьох мов. ODBC слід використовувати, коли для роботи з конкретною базою даних потрібен драйвер.

Наступною технологією доступу до даних стала DAO (Data Access Objects - об'єкти доступу до даних), заснована на процесорі баз даних Microsoft Jet для Microsoft Access. Цей процесор є першим об'єктно-орієнтованим інтерфейсом для спілкування з Microsoft Access (на відміну від технології ODBC, де доступ реалізовано процедурним способом). За допомогою DAO можна легко і швидко отримати необхідні дані та необхідні функції бази даних Jet. На жаль, DAO наразі є застарілою і замороженою технологією. Її розробка була завершена в 2001 році.

Технології ODBC і DAO призначені для взаємодії з реляційними джерелами даних. Для взаємодії з нереляційними базами даних було розроблено технологію OLE DB Object Linking and Embedding Database.

OLE DB - це технологія, яка може працювати як з реляційними, так і з нереляційними джерелами даних. Недоліками цієї технології є складність її використання та зв'язок із C та C++.

Усі ці технології були розроблені до появи платформи. Net Framework. Врахувати особливості Net. не представлялося можливим. Тому було розроблено технологію доступу до даних ADO.NET. Вона враховує найкращі досягнення попередніх технологій і гармонійно поєднує їх із платформою .Net. ADO.NET включає набір просторів і методів для доступу до технологій OLE DB і ODBC, СУБД MS SQL Server і Oracle.

На рис. 2.8 показано основні класи ADO.NET і схеми їхньої взаємодії.

Клас Connection призначений для створення з'єднання між клієнтом і сервером бази даних. У конструкторі класу задаються параметри з'єднання, показані на рисунку 2.3.

Об'єкт класу Command генерується щоразу, коли необхідно створити команду для виконання на стороні бази даних, наприклад, вибірка даних, вставка, видалення, зміна, виклик функції, створення користувача, встановлення прав доступу тощо.

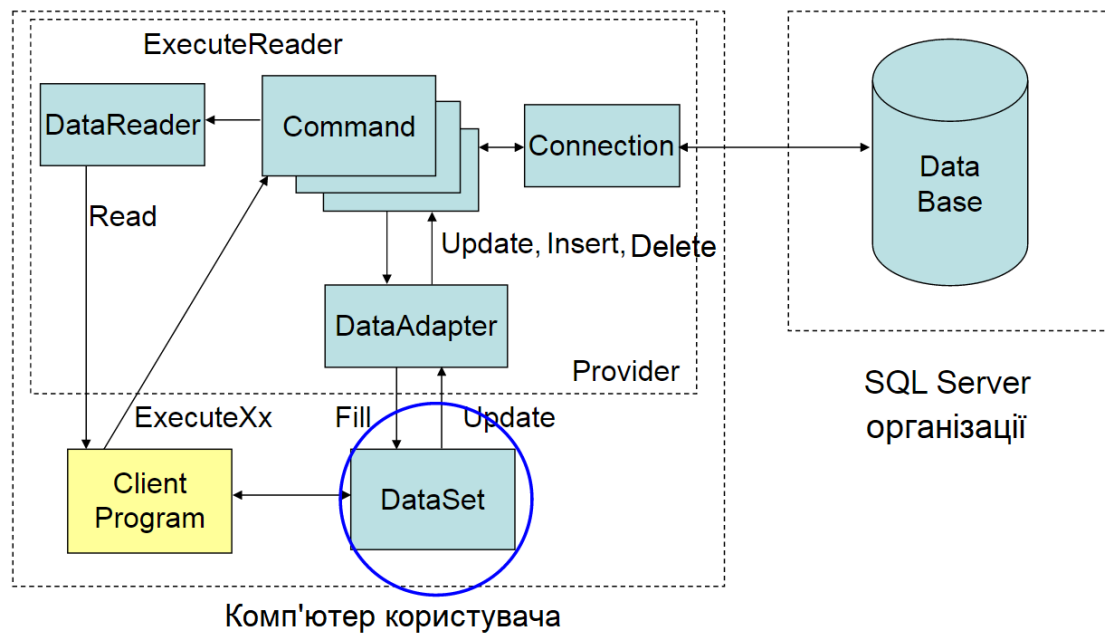


Рисунок 2.8 - Схема взаємодії основних класів ADO.NET

Клас `DataReader` призначений для читання вибірки з бази даних. Результат виконання команди повертається у вигляді об'єкта `DataReader`, який необхідно зациклити, щоб прочитати всі рядки отриманої вибірки. По суті, це курсор для вибірки даних.

Об'єкт DataSet являє собою представлення бази даних в оперативній пам'яті комп'ютера користувача і необхідний для синхронізації даних з базою. Клієнтські додатки можуть вносити зміни в DataSet і переносити їх у реальну базу даних за допомогою DataAdaper.

DataAdapter - це клас, який підтримує відповідність між уявленням клієнта про базу даних у DataSet і реальною базою даних на сервері підприємства.

В ADO.NET існує дві моделі обробки даних: модель підключення та модель від'єднання. Традиційною моделлю обробки даних є модель підключення. Робота моделі підключення полягає в тому, що додаток відкриває з'єднання з постачальником даних і залишається на зв'язку протягом усього часу виконання. Перевагами цієї концепції є швидкість і зручність обробки даних. На рисунку 2.9 схематично показано модель підключення з базою даних.

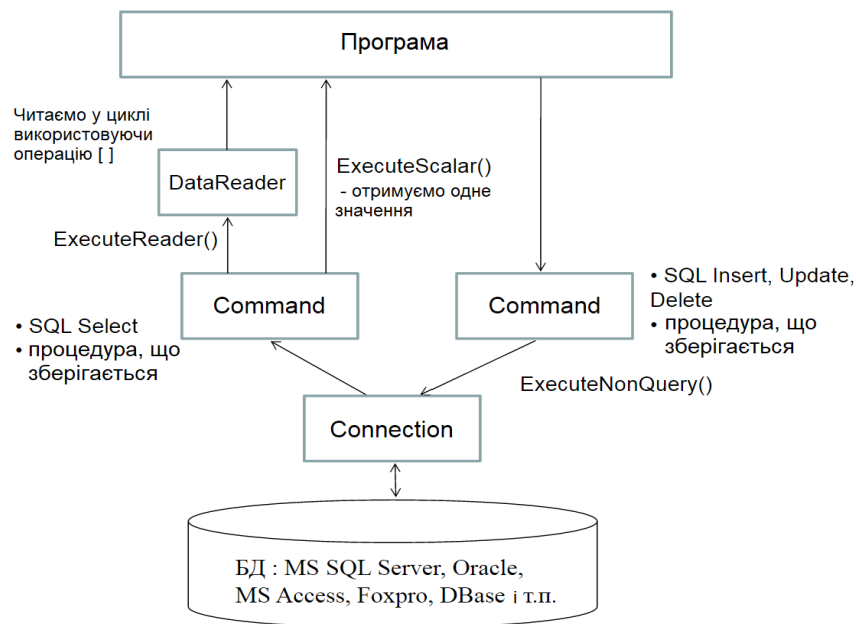


Рисунок 2.9 - Підключений режим роботи з базою даних

Щоб застосувати підключений режим в програмному коді, необхідно створити з'єднання з базою, вказавши параметри з'єднання (логін, пароль, адресу сервера і режим аутентифікації). Далі необхідно відкрити з'єднання (виклик методу `Open`). Потім можна створити команду для бази. Наприклад, команду `Select * From Accounts`. Метод `ExecuteReader` виконує команду, зазначену через `Connection`, на стороні сервера і повертає результати вибірки з бази у вигляді об'єкта `DataReader`. Він у циклі зчитує його вміст і витягує необхідні дані. Якщо необхідно виконати команду, таку як `update`, `delete` або `insert`, викличте метод `ExecuteNonQuery` з об'єкта `command`; на відміну від `ExecuteReader`, ця команда не повертає курсор, а тільки кількість рядків, порушених командою (`rows affected`).

Основним недоліком режиму `Connect with Database` є те, що для з'єднання постійно виділяються ресурси, що може сильно перевантажити провайдер даних при великій кількості з'єднань. Щоб уникнути недоліків роботи з моделлю `Attached` в `ADO.NET` передбачена інша модель роботи з даними доступна інша модель - `Disconnect` (рис. 2.10).

Спочатку створіть об'єкт `Connection`, але не відкривайте з'єднання явно. Далі в оперативній пам'яті створюється прототип `DataSet` бази даних і

заповнюється даними з бази даних; DataSet складається з DataTables, що складаються з рядка DataRow і стовпця DataColumn; створюється об'єкт DataAdapter; і створюється об'єкт DataSet, що з'єднує реальну базу даних на стороні сервера і DataSet на стороні клієнта. Створіть команди, необхідні для маніпулювання базою даних, і вкажіть з'єднання. Якщо потрібно зробити вибірку з бази даних, викликати з адаптера метод Fill; при виклику методу Fill адаптер сам відкриває з'єднання, виконує вибірку і відразу ж закриває з'єднання, звільняючи сервер. Для оновлення даних на сервері з адаптера виконується команда Update.

Таким чином, адаптер синхронізує DataSet на стороні клієнта з актуальною базою даних на стороні сервера. Адаптер сам відкриває і закриває з'єднання, максимально зменшуючи навантаження на сервер.

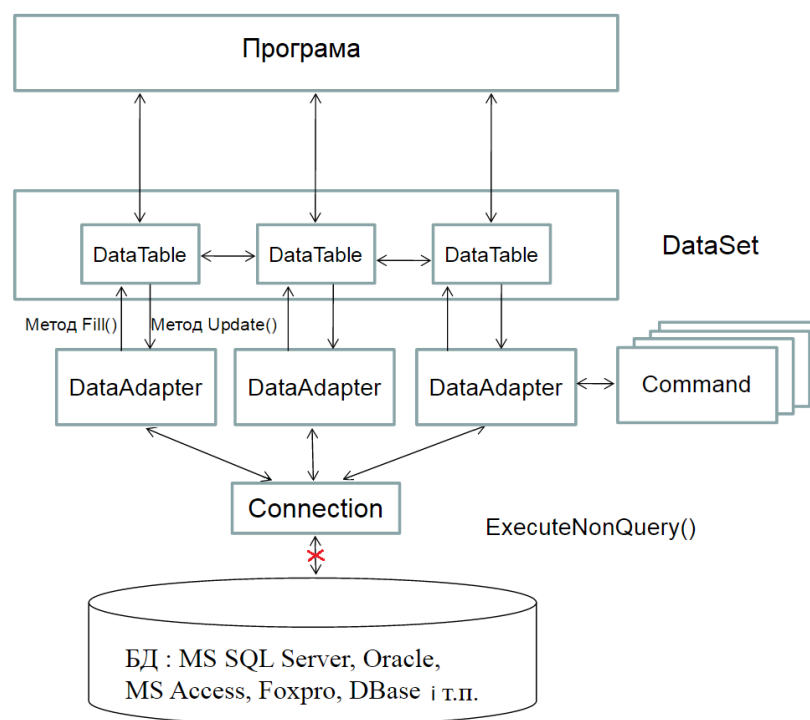


Рисунок 2.10 – Від'єднаний режим бази даних

У результаті від'єднана модель знижує навантаження на постачальника даних (сервер), однак слід пам'ятати, що дані можуть з часом застаріти.

Розробити програмне забезпечення, що працює із серверами баз даних в обох режимах, і дослідити продуктивність системи та навантаження на сервер. Це дасть змогу побудувати регресійні моделі для прогнозування навантаження на сервер.

2.5 Розробка програмного забезпечення для доступу до системи в підключеному режимі

Інтерфейс системи розроблений у 2.3, тому в нього необхідно додати код.

Так обробляється кнопка оплати на формі заявки під час її натискання. У доданому режимі код має такий вигляд, лістинг 2.11

Лістинг 2.11 – Кнопка оплати під час натискання

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

Якщо атомарна частина транзакції завершилася невдало, необхідно відкотити всю транзакцію, тому застосовується режим транзакцій, лістинг 2.12.

Лістинг 2.12 – Режим транзакцій

```
SqlTransaction tran1 = null;
SqlTransaction tran2 = null;
if (((Button) sender).Equals(bPay))
{
    try
    {
```

Якщо користувач обирає оплату послуг мобільного зв'язку, лістинг 2.13, підготуйте підключення до відповідної бази. IP-адреса сервера системи 192.168.0.251

Лістинг 2.13 – Оплата послуг мобільного зв'язку

```

if (cbService.SelectedIndex == 0)
{
    progressBar.Value = 0;
    SqlConnectionStringBuilder builderForPayments = new
SqlConnectionStringBuilder();
    builderForPayments.InitialCatalog = "Payments";
    builderForPayments.DataSource = @"192.168.0.251";
    builderForPayments.IntegratedSecurity = true;
    using (SqlConnection connection = new SqlConnection())
    {

```

Параметри доступу до облікового запису користувача зчитуються з форми, заповненої користувачем, лістинг 2.14.

Лістинг 2.14 – Доступу до облікового запису

```

connection.ConnectionString =
builderForPayments.ConnectionString;
string account = tbPayAccount.Text;
string login = tbLogin.Text;
string password = tbPassword.Text;
connection.Open();
tran1 = connection.BeginTransaction();
SqlCommand comm = connection.CreateCommand();

```

Ініціювати транзакцію, лістинг 2.15. Виберіть користувача з бази даних платників.

Лістинг 2.15 – Ініціювання транзакції

```

comm.CommandText = "Select * from Accounts Where id = " +
account;
comm.Transaction = tran1;
using (SqlDataReader reader = comm.ExecuteReader())
{
    reader.Read();
    string L = reader.GetString(reader.GetOrdinal("login_"));
    string p = reader.GetString(reader.GetOrdinal("password_"));

```

Якщо користувач вводить неправильні дані авторизації, лістинг 2.16, то на екран виводиться попередження, і користувачеві пропонується ввести нові дані.

Лістинг 2.16 – Неправильні дані авторизації

```

if (!login.Equals(L) || !password.Equals(p))
{
    MessageBox.Show("Not authorized");
    connection.Close();
}
else
{
    double summ = Convert.ToDouble(tbSum.Text);
    double s =
    Convert.ToDouble(reader.GetDouble(reader.GetOrdinal("summ")));

```

Віднімає баланс користувача з бази даних платника. Якщо залишок менший за суму, яку користувач хоче переказати на рахунок компанії, то видається відповідне повідомлення, лістинг 2.17.

Лістинг 2.17 – Недостатньо коштів

```

if (summ > s)
{
    MessageBox.Show("No money");
    connection.Close();
}
}
}
SqlConnectionStringBuilder builderForMobile = new
SqlConnectionStringBuilder();
builderForMobile.InitialCatalog = "Mobile";
builderForMobile.DataSource = @"192.168.0.251";
builderForMobile.IntegratedSecurity = true;
using (SqlConnection connectionForMobile = newSqlConnection())
{

```

У разі успішного з'єднання зазначена сума спочатку списується з балансу платника, прогрес-бар зсувається на половину, а потім зараховується на рахунок користувача в бази даних Vodafone, лістинг 2.18.

Лістинг 2.18 – Списання коштів з балансу та занесення їх на рахунок

```

connectionForMobile.ConnectionString =
builderForMobile.ConnectionString;
connectionForMobile.Open();
SqlCommand updateAccounts = connection.CreateCommand();
updateAccounts.CommandText = "Update Accounts set summ = summ
- " + tbSum.Text + "Where id = " + account;

```

```

        updateAccounts.Transaction = tran1;
        updateAccounts.ExecuteNonQuery();
        progressBar.Value = 50;
        tran2 = connectionForMobile.BeginTransaction();
        SqlCommand updateOperators =
connectionForMobile.CreateCommand();
        updateOperators.CommandText = "Update Operators set summ =
summ +" + tbSum.Text + " where number = " + tbAccount.Text;
        updateOperators.Transaction = tran2;
        updateOperators.ExecuteNonQuery();
        tran2.Commit();
    }

```

Якщо все пройшло успішно, заповніть індикатор виконання до кінця і підтвердіть транзакцію, лістинг 2.19.

Лістинг 2.19 – Підтвердження транзакції

```

        tran1.Commit();
    }
    progressBar.Value = 100;
    if (cPrint.Checked)
    {
        Bill b = new Bill();
        b.Message = " З рахунку" + tbPayAccount.Text +
Environment.NewLine + " знято суму " + tbSum.Text +
Environment.NewLine + "на рахунок" + tbAccount.Text;
        b.setMessage();
        b.ShowDialog();
    }
    }
    else
    {

```

Якщо користувач вибирає оплату «Водоканалу», то алгоритм аналогічний. Код показано у лістингу 2.20.

Лістинг 2.20 – Оплату «Водоканалу»

```

        progressBar.Value = 0;
        SqlConnectionStringBuilder builderForPayments = new
SqlConnectionStringBuilder();
        builderForPayments.InitialCatalog = "Payments";
        builderForPayments.DataSource = @"192.168.0.251";
        builderForPayments.IntegratedSecurity = true;
        using (SqlConnection connection = new SqlConnection())
        {

```

```

        connection.ConnectionString =
builderForPayments.ConnectionString;
        string account = tbPayAccount.Text;
        string login = tbLogin.Text;
        string password = tbPassword.Text;
        connection.Open();
        tran1 = connection.BeginTransaction();
        SqlCommand comm = connection.CreateCommand();
        comm.CommandText = "Select * from Accounts Where id = " +
account;
        comm.Transaction = tran1;
        using (SqlDataReader reader = comm.ExecuteReader())
        {
            reader.Read();
            string L = reader.GetString(reader.GetOrdinal("login_"));
            string p = reader.GetString(reader.GetOrdinal("password_"));
            if (!login.Equals(L) || !password.Equals(p))
            {
                MessageBox.Show("Not authorized");
                connection.Close();
            }
            else
            {
                double summ = Convert.ToDouble(tbSum.Text);
                double s =
Convert.ToDouble(reader.GetDouble(reader.GetOrdinal("summ")));
                if (summ > s){
                    MessageBox.Show("No money");
                    connection.Close();
                }
            }
        }
        SqlConnectionStringBuilder builderForMobile = new
SqlConnectionStringBuilder();
        builderForMobile.InitialCatalog = "Vodokanal";
        builderForMobile.DataSource = @"192.168.0.251";
        builderForMobile.IntegratedSecurity = true;
        using (SqlConnection connectionForMobile = new
SqlConnection())
        {
            connectionForMobile.ConnectionString =
builderForMobile.ConnectionString;
            connectionForMobile.Open();
            SqlCommand updateAccounts = connection.CreateCommand();
            updateAccounts.CommandText = "Update Accounts set summ =
summ - " + tbSum.Text + "Where id = " + account;
            updateAccounts.Transaction = tran1;
            updateAccounts.ExecuteNonQuery();
            progressBar.Value = 50;
            tran2 = connectionForMobile.BeginTransaction();
            SqlCommand updateOperators =
connectionForMobile.CreateCommand();

```

```

        updateOperators.CommandText = "Update Accounts set summ =
summ +" + tbSum.Text + " where account = " + tbAccount.Text;
        updateOperators.Transaction = tran2;
        updateOperators.ExecuteNonQuery();
        tran2.Commit();
    }
    tran1.Commit();
}
    progressBar.Value = 100;
    if (cPrint.Checked){
    Bill b = new Bill();
    b.Message = " З рахунку" + tbPayAccount.Text +
Environment.NewLine + " знято суму " + tbSum.Text +
Environment.NewLine + "на рахунок" + tbAccount.Text;
    b.setMessage();
    b.ShowDialog();
    }
    }
    }
    catch (Exception ex){MessageBox.Show(ex.Message); } }}

```

Перевірити працездатність розробленої системи в режимі підключення до бази даних. Для цього необхідно запустити сервер системи, запустити клієнтську частину додатка на віддаленому комп'ютері та сплатити 40 грн. на рахунок КП «Водоканал» від імені абонента Антоненко. Потім для перевірки результатів транзакції використовується MS SQL Server Management Studio. Введені у форму реквізити показано на рис. 2.11.

Рисунок 2.11 - Введення даних у форму транзакції

На рис. 2.12 показано ситуацію на рахунку до і після списання, а на рис. 2.13 - після списання. Як видно з малюнків, 40 грн. були успішно списані з рахунку. На малюнках 2.14 і 2.15 показано, що 40 грн. були успішно зараховані на рахунок Антоненко.

The screenshot shows a SQL query window with the following text:

```
SELECT TOP 1000 [id]
, [summ]
, [first_name]
, [last_name]
, [login_]
, [password_]
FROM [Payments].[dbo].[Accounts]
```

Below the query, the 'Results' tab is active, displaying a table with 7 columns: id, summ, first name, last name, login, and password. The first two rows are highlighted with a blue border.

	id	summ	first name	last name	login	password
1	1	4914	Софія	Антоненко	s@Antonenko	hst9xcc4d
2	2	6915	Вадим	Петренко	v@Petrenko	qZWVqws

Рисунок 2.12 - Стан абонентського рахунку до списання коштів

The screenshot shows the same SQL query as in Figure 2.12. The 'Results' tab displays the same table structure, but the 'summ' value for the first row (id=1) has changed from 4914 to 4874, reflecting the debit of 40 UAH.

	id	summ	first name	last name	login	password
1	1	4874	Софія	Антоненко	s@Antonenko	hst9xcc4d
2	2	6915	Вадим	Петренко	v@Petrenko	qZWVqws

Рисунок 2.13 - Стан абонентського рахунку після проведення операції

```

SELECT TOP 1000 [id]
      ,[account]
      ,[summ]
      ,[first_name]
      ,[last_name]
FROM [Vodokanal].[dbo].[Accounts]

```

100 %

Результаты Сообщения

	id	account	summ	first name	last name
1	1	9263073	760	Софія	Антоненко
2	2	9483982	115	Вадим	Петренко
3	3	8537428	180	Тетяна	Федорівна

Рисунок 2.14 - Залишок на поточному рахунку абонентів КП "Водоканал" до оплати 40 грн.

```

SELECT TOP 1000 [id]
      ,[account]
      ,[summ]
      ,[first_name]
      ,[last_name]
FROM [Vodokanal].[dbo].[Accounts]

```

100 %

Результаты Сообщения

	id	account	summ	first name	last name
1	1	9263073	800	Софія	Антоненко
2	2	9483982	115	Вадим	Петренко
3	3	8537428	180	Тетяна	Федорівна

Рисунок 2.15 - Залишок на поточному рахунку абонентів КП "Водоканал" після оплати 40 грн.

2.6 Розробка програмного забезпечення для доступу до систем у від'єднаному режимі

Створіть набір DataSet, лістинг 2.21, що відображає базу в оперативній пам'яті:

Лістинг 2.21 – Створення DataSet

```
DataSet dataSetPayments;
DataSet dataSetMobile;
DataSet dataSetVodokanal;
```

Створює з'єднання з базою даних платника, лістинг 2.22, але не відкриває його в явному вигляді.

Лістинг 2.22 – Створення з'єднання з базою даних

```
SqlConnectionStringBuilder builderForPayments = new
SqlConnectionStringBuilder();
builderForPayments.InitialCatalog = "Payments";
builderForPayments.DataSource = @"192.168.0.251";
builderForPayments.IntegratedSecurity = true;
SqlConnection disconnectedMode = new SqlConnection();
disconnectedMode.ConnectionString =
builderForPayments.ConnectionString;
```

Віднімання даних із платіжних форм. Лістинг 2.23.

Лістинг 2.23 – Віднімання даних

```
string account = tbPayAccount.Text;
string login = tbLogin.Text;
string password = tbPassword.Text;
```

Створюємо адаптер, лістинг 2.24, як міст між набором даних у пам'яті клієнта і реальною базою даних.

Лістинг 2.24 – Створення адаптера

```
SqlDataAdapter adapter = new SqlDataAdapter();
SqlCommandBuilder builder = new
SqlCommandBuilder((SqlDataAdapter)adapter);
SqlCommandselectFromAccounts=
disconnectedMode.CreateCommand();
```

Віднімання таблиці платників із бази даних, лістинг 2.25.

Лістинг 2.25 – Віднімання таблиці

```
selectFromAccounts.CommandText = "Select * from Accounts";
```

```

adapter.SelectCommand = selectFromAccounts;
adapter.InsertCommand = builder.GetInsertCommand();
adapter.DeleteCommand = builder.GetDeleteCommand();
adapter.UpdateCommand = builder.GetUpdateCommand();

```

Помістим вміст таблиці платників у DataSet клієнта. Лістинг 2.26.

Лістинг 2.26 – Переміщення вміст таблиці платників у DataSet клієнта

```

adapter.Fill(dataSetPayments, "Accounts");
DataRow[] drLoginPassword =

```

Витяг конкретних рахунків платників із набору даних, лістинг 2.27.

Лістинг 2.27 – Витяг конкретних рахунків платників

```

dataSetPayments.Tables["Accounts"].Select(@"login_ = '" +
tbLogin.Text + "' AND password_ = '" + tbPassword.Text + "'");

```

Якщо такого абонента немає, то метод завершується, лістинг 2.28.

Лістинг 2.28 – Завершування метода при невдалому результаті

```

if (drLoginPassword.Length <= 0)
    return;
DataRow[] dtResult =
dataSetPayments.Tables["Accounts"].Select("id = " +
tbPayAccount.Text);
if (dtResult.Length <= 0)
    return;

```

Якщо на рахунку абонента достатньо коштів для проведення транзакції, сума списується з рахунку, лістинг 2.29.

Лістинг 2.29 – Списання коштів з рахунку

```

foreach (DataRow r in dtResult)
{
    if (Convert.ToSingle(r["summ"]) >=
Convert.ToSingle(tbSum.Text))
        r["summ"] = Convert.ToSingle(r["summ"]) -
Convert.ToSingle(tbSum.Text);
    else
    {
        MessageBox.Show("No money");
    }
}

```

```

        return;
    }
}

```

Оновлення реальної бази даних через адаптер, лістинг 2.30.

Лістинг 2.30 – Оновлення бази даних

```

int n = adapter.Update(dataSetPayments, "Accounts");
if (n <= 0)
    return;

```

Якщо користувач вирішує оплатити послуги мобільного зв'язку, відбувається підключення до бази даних мобільного оператора, лістинг 2.31.

Лістинг 2.31 – Підключення до бази даних мобільного оператора

```

if (cbService.SelectedIndex == 0)
{
    tbAccount.Text = tp.AccountForMobile;
    SqlDataAdapter adapterMobile = new SqlDataAdapter();
    SqlCommandBuilder cb = new
SqlCommandBuilder((SqlDataAdapter)adapterMobile);
    SqlConnection connectionMobile = new SqlConnection();
    SqlConnectionStringBuilder builderForMobile = new
SqlConnectionStringBuilder();
    builderForMobile.InitialCatalog = "Mobile";
    builderForMobile.DataSource = @"192.168.0.251";
    builderForMobile.IntegratedSecurity = true;
    connectionMobile.ConnectionString =
builderForMobile.ConnectionString;
    SqlCommand selectAccounts =
connectionMobile.CreateCommand();
    selectAccounts.CommandText = "Select * from Operators";
    adapterMobile.SelectCommand = selectAccounts;
    adapterMobile.InsertCommand = cb.GetInsertCommand();
    adapterMobile.DeleteCommand = cb.GetDeleteCommand();
    adapterMobile.UpdateCommand = cb.GetUpdateCommand();
    adapterMobile.Fill(dataSetMobile, "Operators");
    DataRow[] rows =
dataSetMobile.Tables["Operators"].Select("number = " +
tbAccount.Text);
    if (rows.Length <= 0)
    {
        MessageBox.Show("Number is not found");
    }
}

```

Нараховуємо конкретному абоненту певну суму, лістинг 2.32.

Лістинг 2.32 – Нарахування певної суми

```
foreach (DataRow r in rows)
{
    r["summ"] = Convert.ToSingle(r["summ"]) +
    Convert.ToSingle(tbSum.Text);
}
```

Оновлюємо актуальні суми за допомогою адаптера, С, лістинг 2.33.

Лістинг 2.33 – Оновлення актуальні суми

```
adapterMobile.Update(dataSetMobile, "Operators");
progressBar.Value = 100;
```

Якщо обрано опцію друку чека - разом із чеком виводиться додаткова форма, лістинг 2.34.

Лістинг 2.34 – Друк чека

```
if (cPrint.Checked)
{
    Bill b = new Bill();
    b.Message = "З рахунку" + tbPayAccount.Text +
    Environment.NewLine + "знято суму" + tbSum.Text +
    Environment.NewLine + "на рахунок" + tbAccount.Text;
    b.SetMessage();
    b.ShowDialog();
}}
```

Якщо обрано оплату послуг КП «Водоканал», лістинг 2.35, то вона аналогічна описаній вище, за винятком оплати за фактом роботи.

Лістинг 2.35 – Оплата послуг КП «Водоканал»

```
else
{
    tbAccount.Text = tp.AccountForVodokanal;
    SqlDataAdapter adapterVodokanal = new SqlDataAdapter();
    SqlCommandBuilder commandBuilderVodokanal = new
    SqlCommandBuilder((SqlDataAdapter)adapterVodokanal);
    SqlConnection connectionVodokanal = new SqlConnection();
    SqlConnectionStringBuilder builderForVodokanal = new
    SqlConnectionStringBuilder();
    builderForVodokanal.InitialCatalog = "Vodokanal";
    builderForVodokanal.DataSource = @"192.168.0.251";
```

```

        builderForVodokanal.IntegratedSecurity = true;
        connectionVodokanal.ConnectionString =
builderForVodokanal.ConnectionString;
        SqlCommand selectAccounts =
connectionVodokanal.CreateCommand();
        selectAccounts.CommandText = "Select * from Accounts";
        adapterVodokanal.SelectCommand = selectAccounts;
        adapterVodokanal.InsertCommand =
commandBuilderVodokanal.GetInsertCommand();
        adapterVodokanal.DeleteCommand =
commandBuilderVodokanal.GetDeleteCommand();
        adapterVodokanal.UpdateCommand =
commandBuilderVodokanal.GetUpdateCommand();
        adapterVodokanal.Fill(dataSetVodokanal, "Accounts");
        DataRow[] rows =
dataSetVodokanal.Tables["Accounts"].Select("account = " +
tbAccount.Text);
        if (rows.Length <= 0)
            MessageBox.Show("Account is not found");
        foreach (DataRow r in rows)
        {
            r["summ"] = Convert.ToSingle(r["summ"]) +
Convert.ToSingle(tbSum.Text);
        }
        adapterVodokanal.Update(dataSetVodokanal, "Accounts");
        progressBar.Value = 100;
        if (cPrint.Checked)
        {
            Bill b = new Bill();
            b.Message = "З рахунка " + tbPayAccount.Text +
Environment.NewLine + "знято суму " + tbSum.Text +
Environment.NewLine + "на рахунок " + tbAccount.Text;
            b.setMessage();
            b.ShowDialog();
        }
    }
    dataSetPayments.Clear();
    dataSetMobile.Clear();
    dataSetVodokanal.Clear();
    }
    }
    catch (SqlException se)
    {
        MessageBox.Show(se.Message);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

```

Результати тестування системи на точність постобробки даних у від'єднаному режимі подібні до результатів тестування у підключеному режимі;

продуктивність системи та навантаження на сервер у двох режимах відрізняються. Це питання аналізується в наступному розділі.

3 ДОСЛІДЖЕННЯ РОЗРОБЛЕНОЇ СИСТЕМИ БАНКУ

3.1 Постановка задачі

Метою даної роботи є оцінка навантаження на сервер системи в режимах підключення та від'єднання з клієнтами та розробка регресійних моделей для цих режимів, що дозволяють прогнозувати навантаження на сервер системи в залежності від кількості підключених клієнтів, типу та інтенсивності запитів.

Для оцінки навантаження на сервери системи використовувався програмний компонент Microsoft Management Studio Activity Monitor. Це дає змогу в режимі реального часу відстежувати стан сервера бази даних. Основними параметрами моніторингу сервера баз даних є процесорний час, що використовується для запуску бази даних, введення та виведення даних. Параметрами, що впливають на навантаження на сервер, є кількість підключених клієнтів, кількість пакетів запитів та тип запиту (вибірка, вставка, оновлення, видалення).

Тому завданням даного дослідження є аналіз вхідних характеристик системи та визначення ступеня їх впливу на вихідні параметри системи і побудова регресійної моделі для прогнозування навантаження на сервер системи.

3.2 Проведення експериментів

Для вивчення розробленої системи спробуємо розробити програмну реалізацію клієнтських мікротранзакцій, що надходять на сервер. Для цього

створимо клас, який ідентифікує платників, лістинг 3.1. Створимо в ньому список абонентів.

Лістинг 3.1 – Створення класа який ідентифікує платників

```
class Payers
{
public List<Payer> list;
public Payers ()
{
    list = new List<Payer>();
    list.Add(new Payer("0683377286", "9263073", "s@Antonenko",
"hst9xcc4d", "1"));
    list.Add(new Payer("0683377750", "9483982", " v@Petrenko",
"qZWVqws", "2"));
    list.Add(new Payer("0680441359", "8537428", "t@Fedorivna",
"nKpHWaM", "3"));
    list.Add(new Payer("0680441360", "8366530", "o@Shevchuk",
"TMyUii0", "4"));
    list.Add(new Payer("0680441349", "9633099",
"j@Kramarenko", "kaQyX5B", "5"));
    list.Add(new Payer("0680441358", "8302745",
"o@Bondarenko", "MHs0Z7Y", "6"));
    list.Add(new Payer("0680441391", "9942948", "a@Petrenko",
"CkvfLEo", "7"));
    list.Add(new Payer("0680441325", "9275937", "d@Lysenko", "
fiode3q ", "8"));
    list.Add(new Payer("0680441376", "9650350", "n@Tarasivna",
"xBk7ydS", "9"));
}
}
```

Класи, що визначають абонентів комп'ютерної системи, мають наступний вигляд, лістинг 3.2.

Лістинг 3.2 – Класи що визначають абонентів

```
class Payer
{
    public string AccountForMobile {get; set;}
    public string AccountForVodokanal {get; set;}
    public string Login {get; set;}
    public string Password {get; set;}
    public string PayAccount {get; set;}
    public Payer (string AccountForMobile, string
AccountForVodokanal, string Login, string Password, string
PayAccount)
    {
        this.AccountForMobile = AccountForMobile;
```

```

        this.AccountForVodokanal = AccountForVodokanal;
        this.Login = Login;
        this.Password = Password;
        this.PayAccount = PayAccount;
    }
}

```

Потім код у клієнтській частині системи створює абонента та організовує цикл, який імітує запит на транзакцію до бази. Кількість ітерацій циклу є змінним параметром зображено у лістину 3.3.

Лістинг 3.3 – Кількість ітерацій циклу

```

for (int i = 0; i < 1000; i++)
{
    Payer tp = payers.list[rand.Next(0, 10)];
    tbLogin.Text = tp.Login;
    tbPassword.Text = tp.Password;
    tbPayAccount.Text = tp.PayAccount;
    tbSum.Text = rand.Next(1, 11).ToString();
    cbService.SelectedIndex = rand.Next(0, 2);
}

```

Тип платежу (водопостачання чи телефонний зв'язок) та сума переказу є випадковими параметрами. Результати експерименту з 1000 абонентів показано на рисунку 3.1.

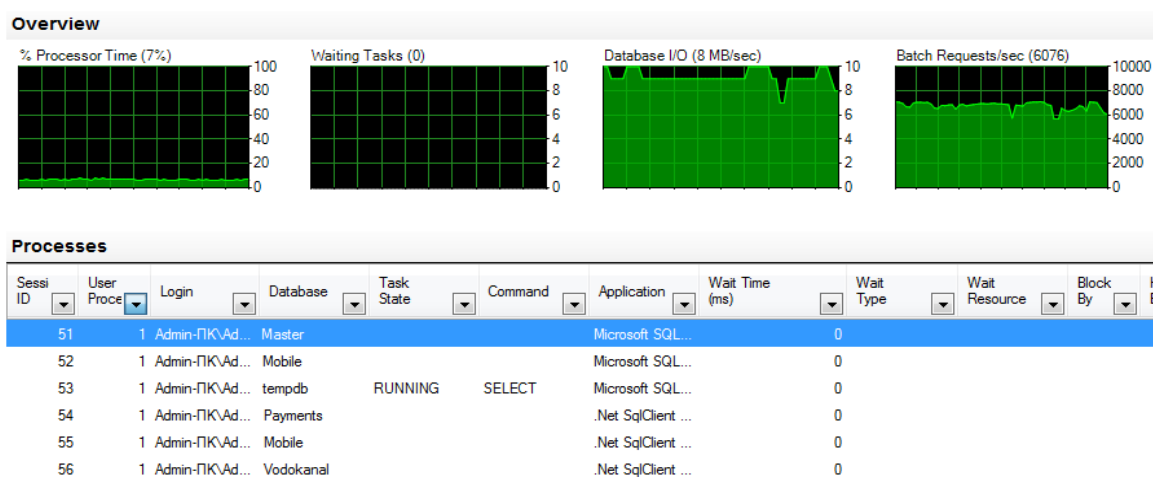


Рисунок 3.1 - Результати експерименту для 1000 абонентів у підключеному режимі

Як видно з рисунку, завантаження процесора на сервері становило 7%, середня швидкість вводу-виводу при роботі з базою даних - 8 МБ/с, а кількість запитів - 6076.

Як видно з рисунку 3.2, навантаження на сервер значно зменшилося: процесорний час на 6%, введення/виведення даних при маніпуляціях з базою даних в середньому становило 5 МБ/с, а кількість запитів, оброблених за одиницю часу, становила 2992 (6076 у попередньому експерименті).

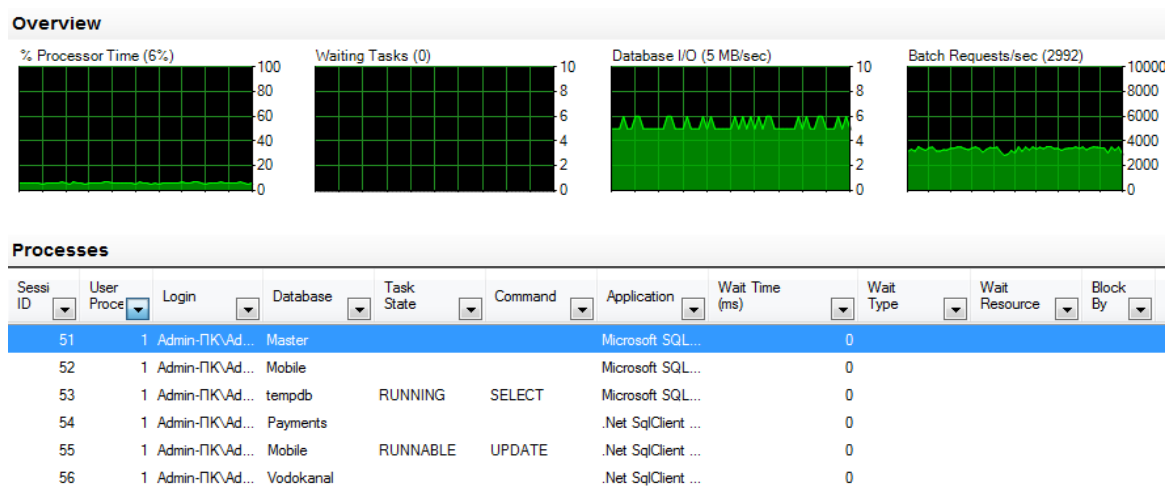


Рисунок 3.2 - Результати експерименту для 1000 абонентів для системи, що працює в від'єднаному режимі

Таким чином, робота системи у від'єднаному режимі значно знижує навантаження на сервер, частина якого перекладається на клієнта.

3.3 Розробка регресійних моделей для прогнозування навантаження на сервер системи

Як уже згадувалося в розділі 3.2, основними параметрами моніторингу сервера бази даних є процесорний час, що використовується під час роботи бази

даних, і час введення/виведення даних. Найважливішим з цих двох параметрів є час використання процесорного часу, і тому він використовується як вихідний параметр у регресійній моделі. Для нормалізації результатів було використано шкалу від 0 до 1, де 0 і 1 означають 0% і 100% використання центрального процесора системного сервера відповідно. Вхідними параметрами є кількість підключених клієнтів (N), кількість пакетів у запиті (Packets) і тип запиту - вибір, додавання, оновлення або видалення (Type). Кількість клієнтів також нормується між 0 і 1, де 0 означає 0 клієнтів, а 1 - 106 клієнтів підключено одночасно. Кількість пакетів у запиті також нормалізується між 0 і 1, де 1 означає 104. Для нормалізації типу запиту вводиться шкала коефіцієнтів, що відповідають певним типам: update - 1, delete - 0.8, select - 0.7, insert - 0.5. Коефіцієнти вводяться відповідно до складності виконання запиту на сервері бази даних, запити на оновлення є найскладнішими, оскільки потребують спочатку відбору та модифікації даних.

На основі прийнятих нормованих параметрів вхід-вихід було побудовано дві таблиці, що описують взаємозв'язки між параметрами вхід-вихід. Всього було проведено 100 експериментів для суміщених і не суміщених режимів. Під час експериментів використовувалася тільки база.

Таблиця 3.1 - Фрагментарна залежність вихідних параметрів від вхідних у режимі сполучення

N	Packets	Type	CPU
0,1	0,2	0,5	0,02
0,1	0,3	0,7	0,03
0,1	0,4	0,8	0,03
0,2	0,2	0,5	0,04
0,2	0,3	0,7	0,04
0,2	0,4	0,8	0,09
0,3	0,2	0,5	0,08

Продовження таблиці 3.1

N	Packets	Type	CPU
0,3	0,3	0,7	0,09
0,3	0,4	0,8	0,11
...	...	...	...
1	1	1	1

Отримані експериментальні дані вводяться в програму Statistica для побудови поліноміальної регресійної моделі другого порядку:

$$CPU = 0,27N + 0,76N^2 - 1,24Packets + 0,66Packets^2 + 0,09Type + 0,45Type^2 \quad (3.1)$$

При цьому коефіцієнт детермінації для адекватності моделі становить 0,9918, що свідчить про високу валідність запропонованої регресійної моделі.

Далі будується регресійна модель для випадку, коли клієнт працює у від'єднаному від бази режимі. Деякі нормовані експериментальні дані подано в табл. 3.2.

Таблиця 3.2 - Фрагментарна залежність вихідних параметрів від вхідних у переривчастому режимі

N	Packets	Type	CPU
0,1	0,2	0,5	0,15
0,1	0,3	0,7	0,02
0,1	0,4	0,8	0,02
0,2	0,2	0,5	0,026
0,2	0,3	0,7	0,031
0,2	0,4	0,8	0,04
0,3	0,2	0,5	0,037
0,3	0,3	0,7	0,071

Продовження таблиці 3.3

N	Packets	Type	CPU
0,3	0,4	0,8	0,095
...	...	...	...
1	1	1	0,9

Регресійна поліноміальна модель другого порядку для від'єданого режиму підключення споживачів до системи виглядає наступним чином:

$$CPU = -0,285N + 1,16N^2 - 2,08Packets + 1,23Packets^2 + 0,78Type + 0,1Type^2 \quad (3.2)$$

Коефіцієнт детермінації 0,981 вказує на те, що запропонована регресійна модель має прийнятний ступінь валідності.

Таким чином, запропонована регресійна модель може прогнозувати навантаження на сервер комп'ютерної системи для обробки банківських мікротранзакцій, задаючи вхідні параметри та обчислюючи вихідні параметри (відповідний процесорний час). Крім того, модель може оцінити вплив кожного вхідного параметра на збільшення навантаження на процесор.

3.4 Економічна ефективність проєкту

Економічна ефективність КС визначається співвідношенням між величиною результатів (економії), отриманих від експлуатації системи, і сумарними витратами на її будівництво. Визначення економічної ефективності КС передбачає розрахунок таких показників:

- річна економія (Р);
- річний економічний ефект (Ер);

- коефіцієнт ефективності капітальних вкладень на створення і впровадження КС (Е);
- термін окупності капітальних витрат (Ток).

Таблиця 3.3 - Розрахунок річних експлуатаційних витрат на експлуатацію

Стаття витрат	Сума, грн.	Порядок розрахунку
1. Заробітна плата обслуговуючого персоналу (метод нарахування)	$ЗП = 4 * 14700 * 12 * 1,2 * 2,2 = 1\ 862\ 784$	$ЗП = Р * О * \Phi_{\text{еф}} * K1 * K2$, де Р - кількість обслуговуючого персоналу, чол.; О – оклад фахівця; Феф – кількість відпрацьованих місяців на рік; K1 – коефіцієнт премій та надбавок; K2 – коефіцієнт відрахування єдиного соціального внеску
2. Витрати на спожиту КТС електроенергію Зел	$Зел = 2,64 * 55 * 1800 * 0,8 = 209\ 088$	$Зел = С_{\text{е}} * М * \Phi * K_{\text{в}}$, де $C_{\text{е}}$ – ціна за 1кВт; М - потужність споживана обладнанням, кВт; Ф – ефективний фонд робочого дня устаткування; Kв – коефіцієнт використання потужності.
3. Амортизаційні відрахування Агод.	$А_{\text{год}} = 7\ 926\ 100,2 * 50 / 100 = 3\ 963\ 050$	$А_{\text{год}} = С_{\text{б}} * N_{\text{а}} / 100$, де $C_{\text{б}}$ – балансова вартість ОФ, грн.; Nа – норма амортизації, 50%.
4. Витрати на поточний ремонт Ртек	$Р_{\text{тек}} = 7\ 926\ 100,2 * 3 / 100 = 237\ 783$	$Р_{\text{тек}} = С_{\text{б}} * 3 / 100$, де $C_{\text{б}}$ – балансова вартість ОФ, грн.;
5. Інші витрати	$6\ 272\ 705 * 0,03 = 188\ 181,15$	Становлять 2-3% від загальної суми попередніх витрат
Разом	6 460 886,15	

Річний економічний ефект E_p , одержуваний у результаті впровадження КС, визначається за таким рівнянням:

$$E_p = P - E_n * K \quad (3.3)$$

де Р - річна економія (результат) від впровадження КС, грн. / Рік; Ен - нормативний коефіцієнт ефективності капітальних вкладень (0,33);

К – обсяг додаткових вкладень, необхідних створення КС, грн.

Коефіцієнт ефективності та термін окупності капітальних витрат у розробленій КС розраховуються за формулою:

$$E = P / K; \quad \text{Ток} = K / P \quad (3.4)$$

Створення та впровадження КС дає змогу умовно вивільнити 60 співробітників із фондом заробітної плати:

22 239 360 грн. (= 60*11700*1,2*2,2*12) на рік.;

Економія складе: Р=26 991 360-14540367,8=7 698 992,2;

Річний економічний ефект: Ер=7698992,2 - 0,33*14540367,8=2 900 670,8 (грн.).

Коефіцієнт ефективності капітальних вкладень у КС:

$E=7\,698\,992,2/14540367,8=0,52$;

Термін окупності: Ток=14540367,8/7 698 992,2=1,8 (рік).

Таким чином, створення і впровадження комп'ютерної системи для обслуговування банківських мікротранзакцій на базі технології ADO.NET є економічно ефективною, термін окупності капітальних вкладень становить 1,8 року, що нижче нормативного терміну в три роки.

На базі технології ADO.NET річний економічний ефект від впровадження комп'ютерної системи для обслуговування банківських мікротранзакцій становить 2 900 670,8 грн.

ВИСНОВКИ

У магістерської роботи створюється та досліджуються комп'ютерні системи на основі технології доступу до баз даних ADO.NET для підвищення ефективності процесів обслуговування банківських мікротранзакцій.

Поставленої мети досягнуто шляхом вирішення наступних завдань:

- інфраструктура системи побудована на мережі відділень банку з 120 комп'ютерами, контролером домену і двома серверами системи, які відповідають на запити клієнтів за допомогою восьми комутаторів і маршрутизаторів;
- програмне забезпечення системи розроблено на мові C# з використанням Windows Forms;
- розробка бази даних відділень банку, що охоплює абонентську базу послуг КП "Водоканал" та послуг мобільного зв'язку;
- аналіз середовища доступу до баз даних ADO.NET;
- проведено експериментальні дослідження розробленої системи та створено регресійну поліноміальну модель другого порядку, яка дозволяє прогнозувати навантаження на сервер системи.

В ході експериментального дослідження системи було виявлено, що запуск системи у від'єднаному режимі значно зменшує навантаження на сервер і перекладає частину навантаження на клієнтів. Розроблена модель (рівняння 3.1 та 3.2) дозволяє прогнозувати навантаження на сервер, змінюючи вхідні параметри та оцінюючи вихідні значення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hogan C., Limoncelli T. A. The practice of system and network administration. Addison-Wesley Professional, 2001. 776 p.
2. Olifer N., Olifer V. Computer networks: principles, technologies and protocols for network design. Wiley, 2006. 1000 p.
3. Концепція мережевих технологій. Реферат: мережеві технології. Комп'ютерний портал osblog. URL: <https://osblog.ru/uk/ponyatie-setevyih-tehnologii-referat-setevye-tehnologii/>.
4. Книга Адміністрування комп'ютерних мереж та систем. Навч. кн. - Богдан, 2010. 196 с.
5. Софліст. URL: <https://ua.softlist.com.ua/articles/sravnenie-windows-11-i-windows-10/>.
6. Norton™ internet security. Norton. URL: <https://uk.norton.com/>.
7. Комплектні системи прецизійного керування внутрішнім кліматом фірми STULZ. Farex. URL: https://farex.ua/files/materials/system_stulz.pdf.
8. Microsoft® SQL server® 2022 SP2. Microsoft Store - Download Center. <https://www.microsoft.com/en-us/sql-server/sql-server-2022>.
9. SQL server management studio (SSMS). Microsoft Learn. URL: <https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>.
10. Рихтер Дж. CLR via C#. програмування на платформі microsoft .NET framework 4.5 на мові C#. 4-е изд. 4-те вид. Print2print, 2020. 896 с.
11. Сеппа Д. Програмування Microsoft ADO.NET 2.0. Майстер клас. 2006. 784 с.
12. Geeks.news. URL: <https://geeks.news/statti/422-chym-windows-11-vidrizniaietsia-vid-windows-10-microsoft-nazvala-osnovnividminnosti.html>.
13. Dickinson P. Professional ADO. NET. Birmingham, UK : Wrox Press Ltd, 2001. 719 p.

14. ADO.NET in a nutshell. Google Books. URL: https://books.google.com.ua/books?hl=ru&lr=&id=1hXuG_w_QmYC&oi=fnd&pg=PR14&dq=ADO.NET&ots=xv3LUtgt5o&sig=kHvONEPCymLkmkytFdGoTrHGFHE&redir_esc=y#v=onepage&q=ADO.NET&f=false.

15. Lin Y. J. ADO.NET database access technology. Applied mechanics and materials. 2015. Vol. 713-715. P. 2217–2220. URL: <https://doi.org/10.4028/www.scientific.net/amm.713-715.2217>.

16. Vaughn W. R. Working with ADO. ADO.NET and ADO examples and best practices for VB programmers. Berkeley, CA, 2002. P. 3–14.