Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення застосунку для обліку клієнтів інтернет-провайдера «SpeedLink» засобами C#

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Сербан Б.Р.
(підпис) (прізвище та ініціали)

Керівник _____ Млинко Б.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Марценко С.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Оробчук О.Р.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_ __ » _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Сербану Богдану Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення застосунку для обліку клієнтів інтернет-провайдера «SpeedLink» засобами C#.

Керівник роботи Млинко Богдана Богданівна, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 2024 року № *4/7-472*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Літературні та інтернет джерела інформації щодо розробки застосунку для обліку клієнтів інтернет-провайдера «SpeedLink» засобами C#

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання розробки застосунку «SpeedLink» 1.1 Аналіз предметної області. 1.2 Аналіз відомих застосунків обліку клієнтів. 1.3 Визначення цільової аудиторії застосунку «SpeedLink»1.4 Вимоги до функціональності розроблюваного застосунку. 1.5 Висновок до першого розділу. 2. Проєктування застосунку «SpeedLink» 2.1 Обгрунтування вибору засобів розробки застосунку «SpeedLink». 2.2 Вибір архітектурного підходу. 2.3 Проєктування бази даних для застосунку «SpeedLink». 2.4 Розробка концептуальної моделі бази даних. 2.5 Висновок до другого розділу. 3. Розробка та тестування застосунку «SpeedLink». 3.1 Розробка застосунку «SpeedLink» 3.1.1 Розробка інтерфейсу користувача 3.1.2 Реалізація функціональних елементів застосунку 3.2 Тестування створеного застосунку «SpeedLink». 3.3 Висновок до третього розділу 4. Безпека життєдіяльності, основи охорони праці 4.1 Надзвичайні ситуації: визначення причини, класифікація. 4.2 Вимоги до профілактичних медичних оглядів для працівників ПК. 4.3 Висновок до четвертого розділу. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульна сторінка. 2. Тема та об'єкт дослідження. 3. Актуальність розроблюваного додатку 4. Аналіз предметної області. 5. Функціональність додатку. 6. Архітектурний підхід розробки застосунку «SpeedLink». 7. Атрибути таблиць бази даних. 8. Концептуальна модель бази даних. 9. Інтерфейс користувача застосунку «SpeedLink». 10. Порівняння відомих застосунків з застосунком «SpeedLink». 11. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., доцент кафедри МТ		

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи		
2.	Підбір джерел про розробку веб-застосунків засобами C#		
3.	Опрацювання джерел щодо розробки застосунку для обліку клієнтів інтернет-провайдера «SpeedLink» засобами C#		
4.	Виконання дослідження щодо розробки застосунку для обліку клієнтів інтернет-провайдера «SpeedLink» засобами C#. Розроблення застосунку для обліку клієнтів інтернет-провайдера «SpeedLink» засобами C#.		
5.	Оформлення розділу «Аналіз предметної області та постановка завдання розробки застосунку «SpeedLink»»		
6.	Оформлення розділу «Проєктування застосунку «SpeedLink»»		
7.	Оформлення розділу «Розробка та тестування застосунку «SpeedLink»»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»		
9.	Виконання завдання до підрозділу «Основи охорони праці»		
10.	Оформлення кваліфікаційної роботи		
11.	Нормоконтроль		
12.	Перевірка на плагіат		
13.	Попередній захист кваліфікаційної роботи		
14.	Захист кваліфікаційної роботи		

Студент _____ Сербан Б.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Млинко Б.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення застосунку для обліку клієнтів інтернет-провайдера «SpeedLink» засобами C# // Кваліфікаційна робота освітнього рівня «Бакалавр» // Сербан Богдан Романович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-42 // Тернопіль, 2024 // С. 53, рис. – 18, табл. – 6, кресл. – 0, додат. – 2, бібліогр. – 44.

Ключові слова: застосунок, бази даних, облік клієнтів, мова C#, Visual Studio, Microsoft SQL Server.

Кваліфікаційна робота присвячена розробці програмного забезпечення для обліку клієнтів інтернет-провайдера «SpeedLink» засобами мови програмування C#, інтегрованого середовища розробки Visual Studio та системи управління базами даних Microsoft SQL Server.

В першому розділі кваліфікаційної роботи проведено аналіз предметної області, розглянуто існуючі застосунки для обліку клієнтів, визначено цільову аудиторію застосунку «SpeedLink» та сформульовано вимоги до його функціональності.

Другий розділ присвячений обґрунтуванню вибору засобів розробки застосунку «SpeedLink», вибору архітектурного підходу, проектуванню бази даних та розробці концептуальної моделі бази даних.

В третьому розділі описано процес розробки та тестування застосунку «SpeedLink». Детально розглянуто створення інтерфейсу користувача, реалізацію функціональних елементів та проведення тестування застосунку для забезпечення його стабільної роботи.

Об'єкт дослідження: процес розробки програмного забезпечення для обліку клієнтів інтернет-провайдера «SpeedLink».

Предмет дослідження: засоби і методи розробки програмного забезпечення для обліку клієнтів із використанням C#, Visual Studio та Microsoft SQL Server.

ANNOTATION

Application Design for Customers Accounting of the Internet Provider "SpeedLink" Using C# // Qualification work for the educational level "Bachelor" / Serban Bohdan // Ternopil National Technical University named after Ivan Pului, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, group SNs-42 // Ternopil, 2024 // P. 53, fig. - 18, tables - 6, drawings - 0, supplementary material - 2, bibliography - 44.

Keywords: application, databases, customer accounting, C#, Visual Studio, Microsoft SQL Server.

The qualification work is devoted to the development of software for customer accounting of the Internet provider "SpeedLink" using the C# programming language, the integrated development environment Visual Studio and the database management system Microsoft SQL Server.

The first chapter of the qualification work analyses the subject area, considers existing applications for customer accounting, identifies the target audience of the SpeedLink application and formulates requirements for its functionality.

The second section is devoted to the justification of the choice of development tools for the SpeedLink application, the choice of an architectural approach, the design of the database and the development of a conceptual database model.

Chapter 3 describes the process of developing and testing the SpeedLink application. The creation of the user interface, implementation of functional elements and testing of the application to ensure its stable operation are considered in detail.

Object of research: the process of developing software for accounting for customers of the Internet provider SpeedLink.

Subject of research: tools and methods for developing software for customer accounting using C#, Visual Studio and Microsoft SQL Server.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

СУБД - Система управління базами даних.

API - Інтерфейс програмування додатків (Application Programming Interface).

UI - Інтерфейс користувача (User Interface).

IDE - Інтегроване середовище розробки (Integrated Development Environment).

ER-діаграма - Діаграма сутність-зв'язок (Entity-Relationship Diagram).

SQL - Мова структурованих запитів (Structured Query Language).

SSMS – SQL Server Management Studio (студія управління SQL Server).

CRM - Застосунок для управління відносинами з клієнтами.

Команда (Command) - поведінковий патерн проєктування.

.NET — крос-платформова технологія.

ООП - Об'єктно-орієнтоване програмування.

Форма (Form) - Основний компонент графічного інтерфейсу. Вікно в Visual Studio яке слугує контейнером для різних елементів керування.

ЗМІСТ

ВСТУП

В сучасному світі інформаційні технології відіграють ключову роль у забезпеченні ефективної діяльності компаній. Вони дозволяють автоматизувати рутинні процеси, підвищити продуктивність праці, покращити якість обслуговування клієнтів та знизити витрати. Одним із важливих аспектів ефективного управління є облік клієнтів, що є основою для розвитку будь-якого бізнесу. Автоматизація процесів обліку клієнтів дозволяє компаніям централізувати зберігання інформації, забезпечити її доступність та безпеку, а також покращити взаємодію з клієнтами [1].

Тому розробка застосунку для обліку клієнтів інтернет-провайдера "SpeedLink" є актуальним напрямком досліджень. В умовах конкуренції на ринку телекомунікаційних послуг, компаніям необхідно ефективно управляти інформацією про своїх клієнтів, своєчасно реагувати на їхні запити та пропонувати персоналізовані послуги [2].

Мета і задачі дослідження:

Метою даної кваліфікаційної роботи є розробка програмного забезпечення для обліку клієнтів інтернет-провайдера "SpeedLink" з використанням мови програмування C# та бази даних SQL Server. Для досягнення поставленої мети потрібно виконати наступні завдання:

– Проаналізувати стан досліджень в області автоматизації обліку клієнтів.

– Розробити структуру бази даних для зберігання інформації про клієнтів та їхні тарифи.

– Реалізувати інтерфейс користувача для введення, редагування та видалення інформації.

– Забезпечити функціональність аутентифікації та авторизації користувачів.

– Провести тестування розробленого застосунку для виявлення та виправлення помилок [3].

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ РОЗРОБКИ ЗАСТОСУНКУ «SPEEDLINK»

1.1 Аналіз предметної області

Застосунки для обліку клієнтів є важливими інструментами для компаній, які прагнуть ефективно управляти взаємовідносинами з клієнтами. Такі програми дозволяють централізовано організовувати інформацію про клієнтів, зберігати історію взаємодій, планувати та контролювати маркетингові кампанії, а також покращувати якість обслуговування клієнтів [4].

Основними функціями цих додатків є управління контактами, що включає зберігання інформації про клієнтів, їх контактні дані, історію взаємодій та уподобання. Це допомагає персоналізувати взаємодію з клієнтами та надавати їм більш якісне обслуговування [5].

Управління продажами в таких системах дозволяє відстежувати процес продажів, включаючи управління клієнтськими запитами, прогнозування продажів та аналіз результатів. Це сприяє підвищенню продуктивності команди продажів і збільшенню доходів компанії [6].

Системи обслуговування клієнтів надають інструменти для підтримки клієнтів, включаючи управління обліком клієнтів, відстеження проблем, бази знань та інші ресурси, які допомагають швидко вирішувати питання клієнтів та покращувати їхній досвід.

Для інтернет-провайдера "SpeedLink" розробка застосунку для обліку клієнтів передбачає врахування специфічних потреб компанії, забезпечення високої функціональності та зручності використання. Інформаційна система повинна надавати повну інформацію про клієнтів, включаючи їх контактні дані, історію взаємодій, інформацію про тарифи та платежі. Також система повинна підтримувати управління тарифами, включаючи створення, редагування та видалення інформації про тарифи, а також відстеження змін у тарифних планах [7]. Автоматизація обліку та звітності є важливим аспектом, який включає автоматичне генерування звітів про клієнтів, платежі та інші показники

діяльності компанії. Інтерфейс користувача має бути інтуїтивно зрозумілим і зручним, дозволяючи легко керувати всіма функціями системи [8].

У сучасному світі інформаційні технології стали незамінними інструментами для вирішення складних управлінських завдань. Це особливо актуально в умовах, коли навіть незначні помилки можуть мати серйозні наслідки. Тому компанії прагнуть автоматизувати свої процеси на всіх етапах діяльності, щоб забезпечити високу ефективність та безпеку. Інформаційна база клієнтів є складною в управлінні, оскільки клієнти можуть підключатися та відключатися, змінювати свої контактні дані та тарифні плани [9].

1.2 Аналіз відомих застосунків обліку клієнтів

На сучасному ринку існує багато відомих рішень для обліку клієнтів, які допомагають компаніям централізовано організовувати інформацію про своїх клієнтів, зберігати історію взаємодій, покращувати обслуговування та підвищувати ефективність маркетингових і продажних кампаній. Кожна система має свої унікальні переваги та недоліки, що робить їх більш або менш підходящими для різних типів бізнесу [10].

Zoho CRM (див. рисунок 1.1) є популярною системою серед малого та середнього бізнесу завдяки своїй доступній ціні та широкому набору функцій, включаючи управління продажами, маркетингом і підтримкою клієнтів [11].

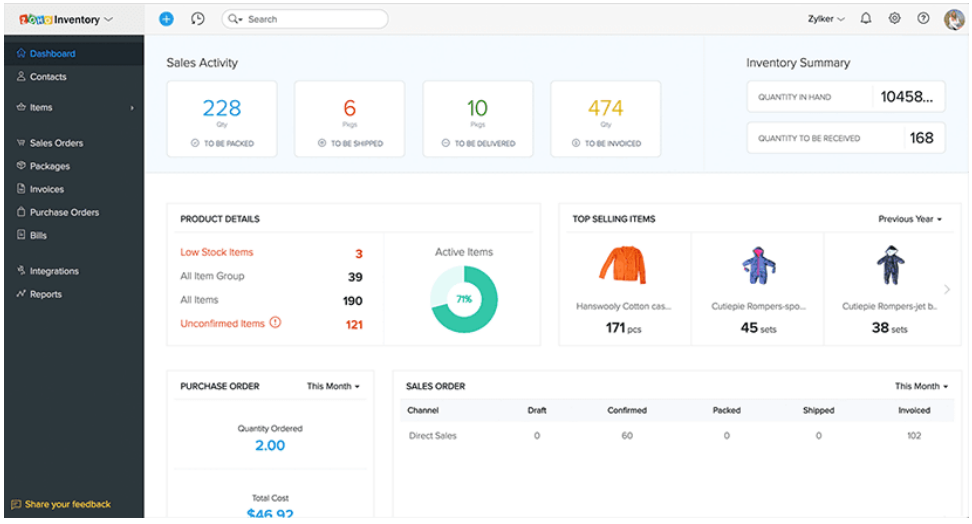

Рисунок 1.1 – Знімок екрану основної сторінки Zoho CRM

Основними перевагами Zoho CRM є доступна ціна, широкий функціонал та можливості налаштування, що дозволяють адаптувати систему під конкретні потреби бізнесу. Однак, інтерфейс може бути менш інтуїтивно зрозумілим для нових користувачів, а підтримка може бути менш оперативною порівняно з іншими CRM-системами [12].

HubSpot CRM (див. рисунок 1.2) пропонує безкоштовну версію своєї CRM-системи, що робить її привабливою для стартапів та малого бізнесу. Система включає інструменти для управління контактами, угодами, продажами та маркетингом. Основними перевагами HubSpot CRM є простота використання, безкоштовний доступ та інтеграція з іншими продуктами HubSpot [13].

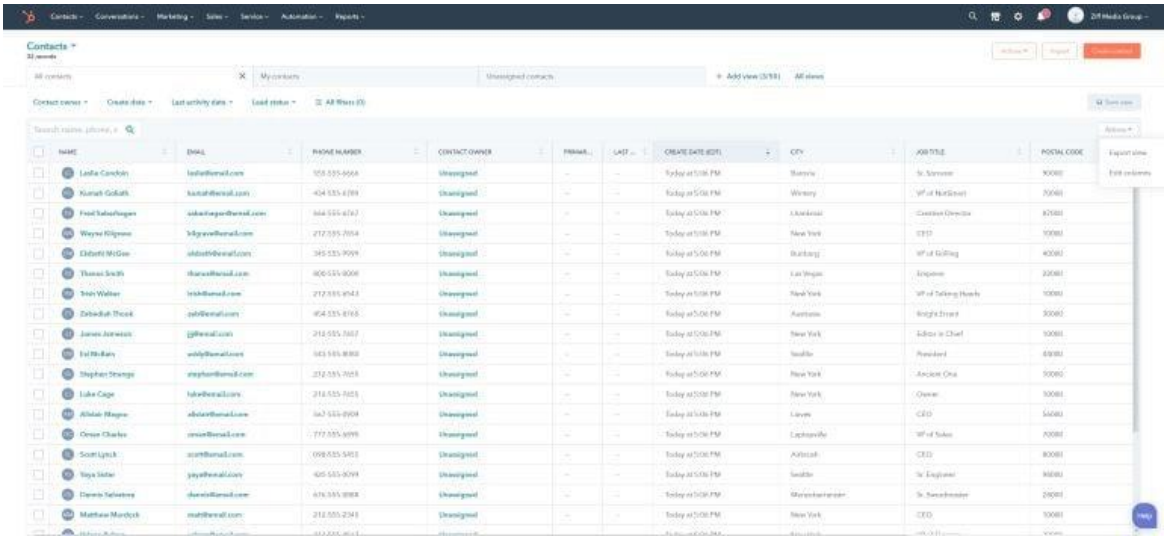

Рисунок 1.2 – Зображення сторінки обліку клієнтів в HubSpot CRM

Основним недоліком HubSpot CRM є обмежена функціональність безкоштовної версії, що може вимагати придбання додаткових платних модулів для розширення можливостей [14].

Для компаній, які не бажають використовувати хмарні рішення та віддають перевагу локальним системам, відмінним вибором може бути затосунок Act! CRM (див. рисунок 1.3). Act! CRM є популярною системою управління взаємовідносинами з клієнтами, яка може працювати без підключення до інтернету [15].

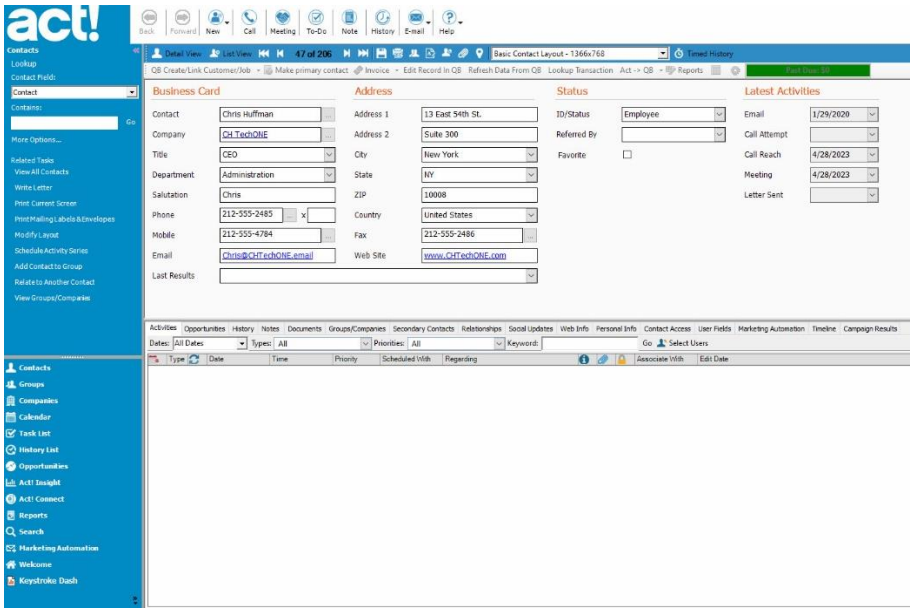

Рисунок 1.3 – Приклад системи Act! CRM

Цей застосунок дозволяє зберігати всі дані локально на сервері компанії, що забезпечує високу безпеку даних і незалежність від зовнішніх сервісів. Основними перевагами Act! CRM є її незалежність від інтернету, висока безпека даних та можливість налаштування під конкретні потреби бізнесу. Однак, для великих компаній із складними потребами ця система може виявитися недостатньо гнучкою, а вартість ліцензій може бути високою для малого бізнесу [16].

Trello (див. рисунок 1.4) є популярним інструментом для управління проєктами та завданнями.

Рисунок 1.4 – Зображення застосунку Trello

Він дозволяє компаніям організовувати роботу своїх команд за допомогою візуальних дошок, карток та списків. Основними перевагами Trello є його простота у використанні, гнучкість та можливість спільної роботи в режимі реального часу. Недоліками Trello є обмежена функціональність для великих проєктів та відсутність деяких розширених функцій, які можуть бути необхідні для складних завдань [17]. Для кращого розуміння порівняльних характеристик різних застосунків, наведено приклад у вигляді таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз застосунків обліку клієнтів

Назва застосунку	**Основні переваги**	**Основні недоліки**	**Основні функції**
Zoho CRM	Широкий функціонал	Важко зрозумілий інтерфейс	Управління маркетингом, клієнтами
HubSpot CRM	Простота використання, безкоштовний доступ	Необхідність придбання платних модулів	Управління контактами, угодами, продажами та маркетингом
Act! CRM	Незалежність від інтернету, висока безпека даних	Недостатня гнучкість, висока вартість ліцензій для малого бізнесу	Управління взаємовідносинами з клієнтами
Trello	Простота у використанні, гнучкість	Обмежена функціональність	Управління проєктами та завданнями за допомогою карток та списків

Наведена таблиця допомагає швидко оцінити основні функції, переваги та недоліки різних застосунків. Це дозволяє обрати найбільш відповідне рішення для конкретних потреб бізнесу.

1.3 Визначення цільової аудиторії застосунку «SpeedLink»

Визначення цільової аудиторії є критично важливим етапом розробки будь-якого програмного забезпечення, оскільки саме від цього залежить, наскільки успішно продукт задовольнить потреби своїх користувачів [18]. Для застосунку SpeedLink, розробленого для інтернет-провайдера, основними категоріями цільової аудиторії є адміністратори системи та працівники провайдера.

Адміністратори є ключовими користувачами застосунку, відповідальними за управління всіма аспектами роботи з клієнтами та працівниками [19]. Їхня роль включає моніторинг і координацію діяльності провайдера, управління базами даних клієнтів, а також забезпечення безпеки та конфіденційності інформації. Адміністратори потребують доступу до широкого спектру функцій для ефективного управління всіма процесами [20]. Це включає можливість додавати нових клієнтів, редагувати інформацію про існуючих клієнтів, управляти тарифами та налаштовувати доступ для інших працівників. Вони також повинні мати можливість змінювати паролі та забезпечувати безпеку даних у системі. Зручний і інтуїтивно зрозумілий інтерфейс є критично важливим для адміністраторів, оскільки він сприяє швидкому реагуванню на запити клієнтів та працівників, а також забезпечує ефективну роботу системи [21].

Працівники провайдера, зокрема ті з розширеними можливостями, складають іншу важливу категорію користувачів. Вони займаються безпосереднім обслуговуванням клієнтів, тому для них необхідний доступ до детальної інформації про клієнтів та їхні послуги. Працівники повинні мати можливість швидко знаходити та переглядати необхідну інформацію для надання якісного обслуговування. Також важливо, щоб працівники з розширеними можливостями могли змінювати свої паролі для забезпечення безпеки облікових записів. Вони повинні мати інструменти для оперативного вирішення запитів клієнтів та надання їм актуальної інформації про тарифи та

послуги. Зручність і швидкість доступу до інформації є ключовими факторами для цієї категорії користувачів.

Таким чином, цільова аудиторія застосунку SpeedLink включає адміністраторів системи та працівників провайдера. Кожна з цих категорій має свої специфічні потреби та очікування, які необхідно враховувати при розробці функціональності та інтерфейсу застосунку. Адміністратори потребують доступу до всіх функцій управління, працівники – до швидкого перегляду та редагування даних клієнтів. Відповідно, система повинна бути зручною, надійною та ефективною, щоб задовольнити всі ці потреби та забезпечити високий рівень обслуговування.

1.4 Вимоги до функціональності розроблюваного застосунку

Застосунок SpeedLink має надавати широкий спектр функцій для адміністраторів та працівників різних рівнів доступу, що відображено на діаграмі прецедентів. Основні функції охоплюють управління клієнтами, тарифами, працівниками, а також забезпечення безпеки через систему автентифікації та управління доступом.

Однією з головних функцій застосунку є управління інформацією про клієнтів. Адміністратори мають можливість додавати нових клієнтів, редагувати їхні дані та видаляти клієнтів разом з усією пов'язаною інформацією. Це дозволяє, наприклад, при реєстрації нового клієнта легко внести всі необхідні дані, такі як ім'я, контактну інформацію та номер договору [22]. Якщо клієнт змінює адресу або обирає інший тарифний план, адміністратор може швидко оновити ці дані в системі. У випадку, якщо клієнт вирішує припинити співпрацю, адміністратор може видалити його запис разом з усією історією послуг.

Ще однією важливою функцією є управління працівниками. Адміністратори можуть додавати нових працівників до системи, редагувати їхні дані, змінювати паролі та налаштовувати доступи. Це дозволяє, наприклад, при прийомі на роботу нового працівника, швидко створити його профіль з усіма необхідними правами доступу. Якщо працівник переходить на нову посаду або

змінюються його обов'язки, адміністратор може легко оновити відповідну інформацію. У випадку звільнення працівника його профіль можна видалити, щоб забезпечити конфіденційність даних.

Адміністратори також мають можливість управляти тарифами. Це включає додавання нових тарифних планів, редагування існуючих та видалення застарілих. Наприклад, якщо з'являється новий тарифний план з кращими умовами, адміністратор може додати його до системи та оновити інформацію для клієнтів, які ним користуються. Видалення старих або неактуальних тарифів також доступне, що дозволяє підтримувати систему актуальною.

Безпека даних є ключовим аспектом роботи застосунку. Система автентифікації передбачає захищений вхід з використанням логіну та пароля [23]. Адміністратори можуть змінювати паролі як для себе, так і для працівників. Кожен працівник, зокрема й ті з розширеними можливостями, може змінювати свій особистий пароль для забезпечення безпеки облікового запису. Важливо зазначити, що система повинна мати чіткий розподіл ролей та прав доступу, щоб кожен працівник мав доступ лише до тієї інформації та функцій, які йому необхідні для виконання своїх обов'язків.

Працівники з розширеними можливостями, а також адміністратори, мають доступ до детальної інформації про клієнтів. Це включає перегляд історії послуг, тарифних планів та контактної інформації. Така функція дозволяє швидко знаходити необхідну інформацію про клієнта та ефективно вирішувати його запити чи проблеми.

Таким чином, застосунок SpeedLink повинен забезпечувати комплексне управління клієнтською базою, тарифами та працівниками, забезпечуючи високу продуктивність, безпеку та зручність у використанні. Система має бути гнучкою та масштабованою, щоб відповідати потребам інтернет-провайдера та його клієнтів. Усі ці функції та можливості чітко відображені на діаграмі прецедентів (див. рисунок 1.5), де можна побачити, як різні ролі в системі взаємодіють з різними частинами застосунку. Адміністратори мають доступ до додавання та редагування працівників, тарифів та клієнтів, а також до налаштування доступу

та зміни паролів. Працівники з розширеними можливостями можуть змінювати свої паролі та переглядати детальну інформацію про клієнтів.

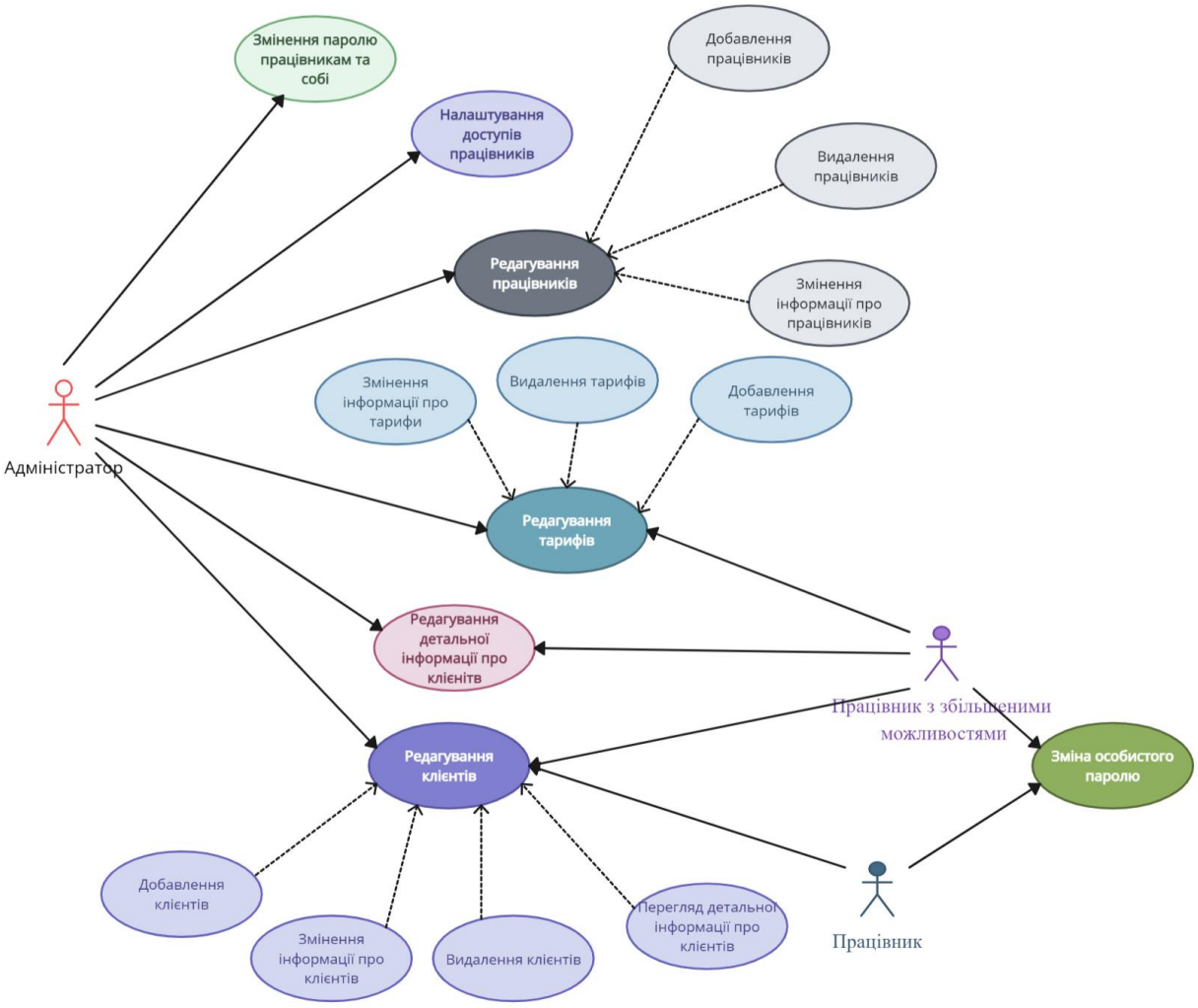

Рисунок 1.5 – Діаграма прецедентів

Діаграма прецедентів (варіантів використання) наочно демонструє, як функції застосунку розподілені між різними користувачами і як вони забезпечують ефективну роботу системи.

1.5 Висновок до першого розділу

Аналіз систем обліку клієнтів показав, що такі додатки є ключовими для ефективного управління взаємовідносинами з клієнтами, забезпечуючи централізоване зберігання даних, підтримку маркетингових кампаній та покращення обслуговування клієнтів. Основні функції включають управління

контактами, продажами та обслуговуванням клієнтів, а також забезпечення простоти використання.

Визначення цільової аудиторії для застосунку SpeedLink є критично важливим етапом, що впливає на успішність розробки програмного забезпечення. Цільова аудиторія включає адміністраторів системи, працівників провайдера та кінцевих клієнтів. Кожна з цих категорій має свої унікальні потреби, які необхідно враховувати при розробці функціональності та інтерфейсу застосунку.

Вимоги до функціональності застосунку SpeedLink включають широкий спектр функцій для різних рівнів доступу. Адміністратори повинні мати можливість додавати, редагувати та видаляти клієнтів, управляти тарифами та працівниками, а також забезпечувати безпеку через систему автентифікації. Працівники з розширеними можливостями мають доступ до детальної інформації про клієнтів та можуть змінювати свої паролі. Усі ці функції наочно відображені на діаграмі прецедентів, що демонструє розподіл функцій між різними користувачами.

Таким чином, затосунок SpeedLink повинен бути зручним, надійним та ефективним, щоб задовольнити потреби всіх категорій користувачів і забезпечити високий рівень обслуговування.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ЗАТОСУНКУ «SPEEDLINK»

2.1 Обгрунтування вибору засобів розробки застосунку «SpeedLink»

Розробка застосунку SpeedLink потребувала ретельного вибору технологічного стеку для забезпечення високої продуктивності, надійності, масштабованості та зручності розробки. Після детального аналізу було обрано мову програмування C#, інтегроване середовище розробки Visual Studio та систему управління базами даних Microsoft SQL Server.

Мова програмування C# була обрана через кілька важливих причин.

По-перше, C# підтримує принципи об'єктно-орієнтованого програмування (ООП), що дозволяє створювати модульний, легко підтримуваний і розширюваний код. Це означає, що можна ефективно організувати код, розділяючи його на класи та об'єкти, що відображають реальні сутності [24].

По-друге, C# працює в середовищі .NET, що забезпечує велику кількість готових бібліотек і фреймворків для різних завдань, включаючи роботу з базами даних, обробку XML, побудову веб-додатків і багато іншого. Це значно прискорює процес розробки, дозволяючи використовувати готові рішення замість створення власних з нуля [25].

По-третє, мова C# забезпечує автоматичне управління пам'яттю та вбудовані механізми для обробки виключень, що підвищує безпеку і стабільність додатків. Це допомагає запобігати помилкам, пов'язаним з ручним управлінням пам'яттю, та забезпечує надійність роботи застосунку. Також, C# дозволяє створювати додатки, які легко масштабуються, що є важливим для підтримки зростаючої кількості користувачів і обсягів даних. Це особливо критично для застосунку SpeedLink, який повинен обробляти великий потік даних від численних користувачів [26].

Visual Studio було обрано як основне інтегроване середовище розробки (IDE) завдяки своїм численним перевагам. Перш за все, Visual Studio пропонує інтуїтивно зрозумілий інтерфейс користувача, що спрощує процес розробки навіть для новачків. Інтерфейс IDE організований так, що всі необхідні

інструменти та функції доступні під рукою, що прискорює процес розробки. Крім того, це середовище має потужні інструменти для налагодження, включаючи покрокове виконання коду, перевірку змінних і стеження за виконанням програм. Ці інструменти дозволяють швидко знаходити та виправляти помилки, забезпечуючи високу якість кінцевого продукту. Глибока інтеграція з .NET дозволяє швидко створювати, тестувати та розгортати додатки на цій платформі. Це забезпечує безперебійну роботу з усіма компонентами .NET екосистеми [27]. До того ж, Visual Studio підтримує велику кількість розширень і плагінів, що дозволяє розробникам налаштовувати середовище під свої потреби. Це робить процес розробки більш гнучким і адаптивним до конкретних вимог проєкту.

Microsoft SQL Server був обраний як основна система управління базами даних з кількох причин. Перш за все, SQL Server забезпечує високу продуктивність і надійність роботи з великими обсягами даних, що є критично важливим для застосунку SpeedLink. Висока швидкість обробки запитів і стабільність роботи бази даних забезпечують ефективне управління даними навіть при високих навантаженнях. Відмінна інтеграція з .NET спрощує розробку і управління базою даних безпосередньо з коду застосунку. Це забезпечує безперебійну взаємодію між застосунком і базою даних. Крім того, SQL Server Management Studio (SSMS) надає потужні інструменти для управління базою даних, виконання запитів, створення резервних копій і моніторингу продуктивності [28]. Ці інструменти полегшують адміністрування бази даних і забезпечують її надійну роботу.

Важливо зазначити, що інтеграція між Visual Studio та Microsoft SQL Server є особливо вдалою. Visual Studio має вбудовані засоби для роботи з базами даних, що дозволяє розробникам легко підключатися до SQL Server, створювати та керувати базами даних, писати запити і виконувати їх безпосередньо з середовища розробки. Це забезпечує безшовний робочий процес, де розробка застосунку та робота з базою даних інтегровані в одному інтерфейсі. Така інтеграція дозволяє швидко вносити зміни в структуру бази даних, перевіряти їх

і бачити результати в реальному часі, що значно підвищує ефективність розробки.

Вибір C# як мови програмування, Visual Studio як інтегрованого середовища розробки та Microsoft SQL Server як системи управління базами даних забезпечує високу продуктивність, надійність і масштабованість застосунку SpeedLink. Ці інструменти та технології дозволяють створити якісний продукт, який відповідає всім вимогам користувачів і забезпечує ефективну роботу з даними. Таке поєднання технологій дозволяє не лише швидко і якісно розробляти застосунок, але й забезпечувати його подальшу підтримку та розвиток.

2.2 Вибір архітектурного підходу

Для розробки застосунку SpeedLink було обрано патерн «Команда» (Command). Цей патерн забезпечує чітку організацію коду, інкапсуляцію дій користувача та підвищення гнучкості системи. Патерн «Команда» дозволяє інкапсулювати запити у вигляді об'єктів, що спрощує збереження, чергування та передачу запитів між різними компонентами системи (див. рис. 2.1).

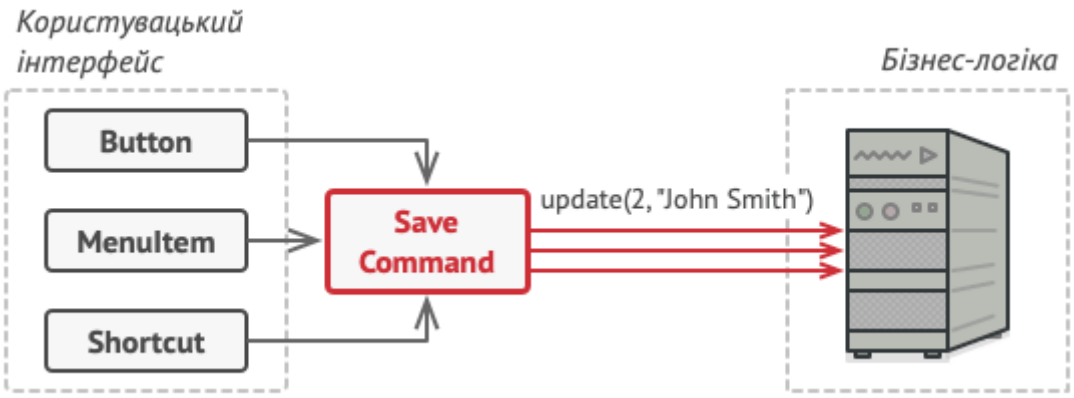

Рисунок 2.1 – Використання патерну «Команда»

По-перше, патерн «Команда» забезпечує легке розширення функціональності програми. Додавання нових команд не вимагає зміни існуючого коду, що спрощує підтримку та розвиток застосунку. Розробники

можуть просто створювати нові класи команд, реалізуючи потрібну логіку, і додавати їх до системи без необхідності змінювати вже існуючий код.

По-друге, патерн «Команда» надає можливість легко реалізувати відміну та повторення дій (undo/redo). Це підвищує зручність користувача, оскільки дозволяє виправляти помилки та повертатися до попереднього стану застосунка. Команди можуть зберігати інформацію про попередній стан, що дозволяє легко відміняти виконані дії або повторювати їх у разі потреби [29].

По-третє, команди інкапсулюють запити та дії користувача, що робить код чистішим та легшим для розуміння. Це дозволяє зберігати всю логіку виконання дії в одному місці, ізолюючи її від іншого коду програми. Завдяки цьому, розробникам легше підтримувати та змінювати код, оскільки кожна команда відповідає за свою конкретну дію [30].

Таким чином, використання патерну «Команда» значно покращує архітектуру застосунка, роблячи його більш гнучким, структурованим та зручним у використанні.

Коли працівник або адміністратор компанії SpeedLink запускає затосунок і намагається увійти в систему, він вводить свої облікові дані та натискає кнопку «Увійти». У цьому випадку він є відправником запиту на вхід. Натискання кнопки «Увійти» формує команду на вхід в систему, яка містить інформацію про введені логін та пароль. Команда передається до обробника, який знає, як обробити цю команду. Обробник використовує передані облікові дані для перевірки в базі даних. У випадку успішної перевірки користувач отримує доступ до системи. Якщо облікові дані неправильні, обробник повертає повідомлення про помилку.

2.3 Проєктування бази даних для застосунку «SpeedLink»

Проєктування бази даних для застосунку SpeedLink є ключовим етапом, що забезпечує ефективне зберігання та доступ до даних. База даних має бути структурована таким чином, щоб забезпечити швидку та надійну роботу застосунку, а також підтримувати всі необхідні функціональні можливості. В

Microsoft SQL Server створюються наступні таблиці AdminCredentials, Workers, Clients, ClientDetails, Tariffs.Таблиця «AdminCredentials» (див. таблицю 2.1) використовується для зберігання облікових даних адміністраторів, що дозволяє забезпечити безпеку та контроль доступу до системи SpeedLink. Кожен запис містить унікальний ідентифікатор адміністратора, логін та пароль, які необхідні для автентифікації при вході в систему.

Таблиця 2.1 – Атрибути таблиці AdminCredentials

Атрибут	**Тип даних**	**Значення атрибуту**
ID_адміністратора	Int, Primary Key	Унікальний ідентифікатор адміністратора
Логін	nvarchar(50)	Логін для входу адміністратора
Пароль	nvarchar(50)	Пароль для входу адміністратора

Таблиця Workers (див. таблицю 2.2) використовується для зберігання інформації про працівників компанії SpeedLink. Вона містить особисті дані працівників, контактну інформацію, дані для входу в систему, а також дані про дозволи та статуси працівників. Вона дозволяє ефективно керувати доступом працівників до застосунку та підтримувати актуальну інформацію про їхній статус в компанії.

Таблиця 2.2 – Атрибути таблиці Workers

Атрибут	**Тип даних**	**Значення атрибуту**
ID_працівника	Int, Primary Key	Унікальний ідентифікатор працівника
Прізвище	nvarchar(50)	Прізвище працівника
Імя	nvarchar(50)	Ім’я працівника
По_батькові	nvarchar(50)	По батькові працівника
Номер_телефону	nvarchar(20)	Номер телефону працівника
Пошта	nvarchar(50)	Пошта працівника

Продовження таблиці 2.2

Вхід	nvarchar(10)	Надалі зчитується чи дозволений вхід в застосунок для працівника
Пароль	nvarchar(50)	Пароль для входу працівника
Дата_реєстрації	date	Дата реєстрації працівника в програмі
Дата_звільнення	date	Дата звільнення працівника
Дозвіл_редагування	nvarchar(10)	Надалі зчитується чи надано збільшений дозвіл редагування в застосунку

Таблиця Clients (див. таблицю 2.3) використовується для зберігання інформації про клієнтів компанії SpeedLink. Вона містить особисті дані клієнтів, контактну інформацію та інформацію про підписки на тарифні плани. Це дозволяє ефективно керувати інформацією про клієнтів та їх підписки.

Таблиця 2.3 – Атрибути таблиці Clients

Атрибут	**Тип даних**	**Значення атрибуту**
ID_Договору	Int, Primary Key	Унікальний ідентифікатор договору клієнта
Прізвище	nvarchar(50)	Прізвище клієнта
Імя	nvarchar(50)	Ім’я клієнта
По_батькові	nvarchar(50)	По батькові клієнта
Назва_тарифу	nvarchar(100)	Назва тарифу, який обрав клієнт
Номер_телефону	nvarchar(15)	Номер телефону клієнта
Адреса	nvarchar(100)	Адреса клієнта

Таблиця ClientsDetails (див. таблицю 2.4) використовується для зберігання детальної інформації про підключення клієнтів до тарифних планів. Вона містить інформацію про дати підключення та сплати, вартість тарифу та суми останніх сплат. Ця таблиця дозволяє ефективно керувати інформацією про підключення клієнтів та їхні платежі.

Таблиця 2.4 – Атрибути таблиці ClientsDetails

Атрибут	Тип даних	Значення атрибуту
ID_детальної_інформації	Int, Primary Key	Унікальний ідентифікатор детальної інформації
ID_Договору	Int, Foreign Key	Ідентифікатор договору клієнта
ID_Тарифу	Int, Foreign Key	Ідентифікатор тарифу
Назва_тарифу	nvarchar(100)	Назва тарифу
Дата_підключення_тарифу	date	Дата підключення тарифу
Вартість_тарифу	nvarchar(50)	Вартість тарифу
Дата_останньої_сплати	date	Дата останньої сплати
Сума_останньої_сплати	decimal(10,2)	Сума останньої сплати

Таблиця Tariffs (див. таблицю 2.5) використовується для зберігання інформації про доступні тарифні плани компанії SpeedLink.

Таблиця 2.5 – Атрибути таблиці Tariffs

Атрибут	Тип даних	Значення атрибуту
ID_тарифу	Int, Primary Key	Унікальний ідентифікатор тарифу
Назва_тарифу	nvarchar(100)	Назва тарифу
Послуги	nvarchar(max)	Опис послуг, що надаються в рамках тарифу
Вартість_тарифу	decimal(10,2)	Вартість тарифу
Дата_введення_тарифу	date	Дата введення тарифу в дію

Таблиця містить усю інформацію про атрибути, що дозволяє ефективно керувати тарифними планами та їх параметрами.

2.4 Розробка концептуальної моделі бази даних

Розробка концептуальної моделі бази даних є важливим етапом у процесі проєктування інформаційних систем. Концептуальна модель забезпечує абстрактне уявлення про структуру даних, які будуть зберігатися та використовуватися у системі. Вона допомагає визначити основні сутності, їхні атрибути та взаємозв’язки між ними. Це дозволяє розробникам отримати чітке розуміння структури даних та уникнути помилок на етапі реалізації [31].

Концептуальна модель бази даних є важливим інструментом для комунікації, забезпечуючи спільне розуміння структури даних між усіма учасниками проєкту, включаючи розробників, аналітиків і замовників. Вона також сприяє проєктуванню, дозволяючи визначити сутності і їх атрибути, а також зв'язки між ними, що створює точний і повний опис даних. Крім того, модель сприяє аналізу, виявляючи і вирішуючи проблемні місця у структурі даних ще на етапі проєктування. Врешті-решт, концептуальна модель служить основою для створення фізичної моделі бази даних, полегшуючи процес її реалізації в системі управління базами даних (СУБД) [32].

Одним з основних інструментів для створення концептуальної моделі бази даних є ER діаграма (Entity-Relationship діаграма). ER діаграма відображає сутності (Entity), їх атрибути (Attributes) та взаємозв’язки (Relationships) між сутностями. Вона допомагає візуально уявити структуру даних та зрозуміти, як дані пов'язані між собою.

ER діаграма складається з сутностей, що представляють об'єкти або концепції, які потрібно зберігати в базі даних, атрибутів, що є характеристиками або властивостями сутностей, та зв'язків, які показують, як ці сутності взаємодіють [33]. Нижче наведена ER діаграма на рисунку 2.2, для бази даних SpeedLink, яка візуалізує сутності та їх атрибути, а також взаємозв’язки між ними.

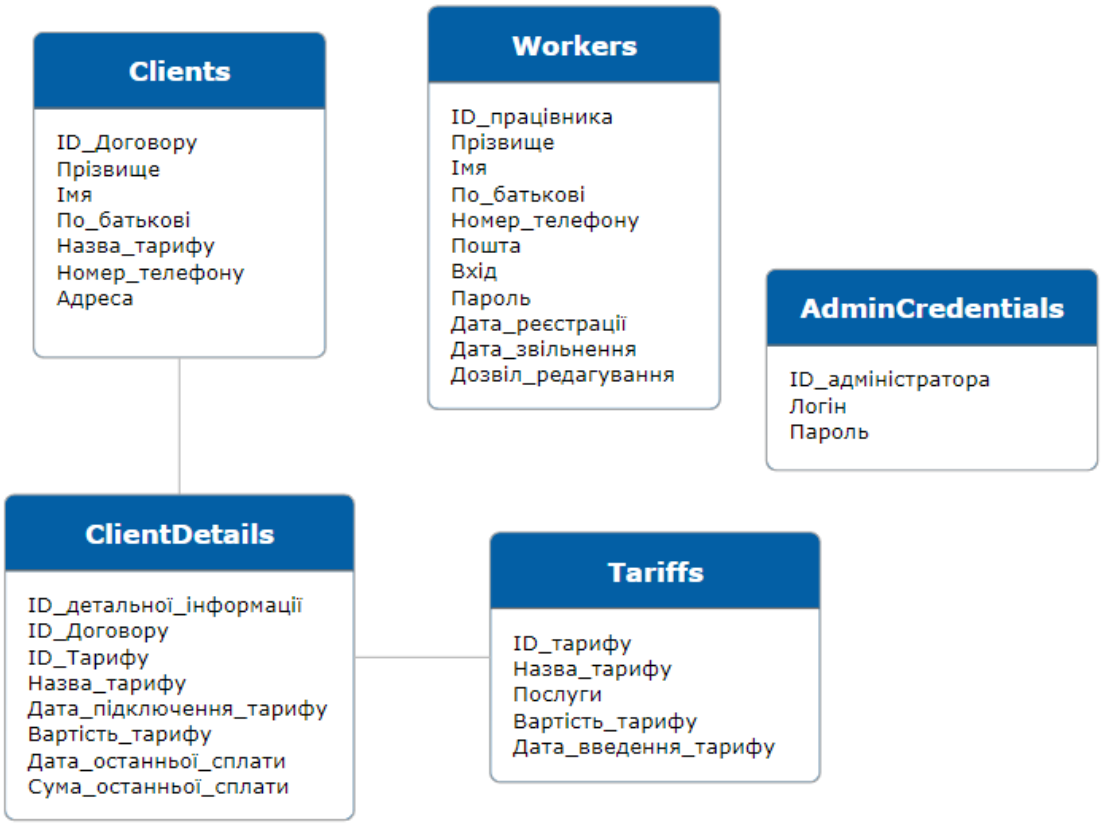

Рисунок 2.2 – Створена ER діаграма

Створена діаграма є наочним інструментом, що дозволяє краще зрозуміти структуру даних і зменшити ризик помилок на етапі реалізації.

2.5 Висновок до другого розділу

Для розробки застосунку SpeedLink було обрано відповідний технологічний стек, який включає мову програмування C#, інтегроване середовище розробки Visual Studio та систему управління базами даних Microsoft SQL Server. C# надає переваги об'єктно-орієнтованого програмування, широкий набір бібліотек і фреймворків, автоматичне управління пам'яттю та обробку виключень, що сприяє підвищенню безпеки та стабільності додатків. Visual Studio пропонує зручний інтерфейс, потужні інструменти для налагодження та можливості розширення, що полегшує процес розробки. Microsoft SQL Server забезпечує високу продуктивність при роботі з великими обсягами даних та відмінну інтеграцію з .NET.

Використання патерну «Команда» (Command) в архітектурі застосунку сприяє чіткій організації коду, інкапсуляції дій користувача та підвищенню гнучкості системи. Цей підхід спрощує збереження, чергування та передачу запитів, полегшує підтримку та розширення функціональності, а також забезпечує зручність для користувачів.

Проєктування бази даних включає створення таблиць AdminCredentials, Workers, Clients, ClientDetails та Tariffs, які зберігають інформацію про адміністраторів, працівників, клієнтів та деталі тарифних планів. Ці таблиці забезпечують ефективне управління даними користувачів та підписками, що сприяє надійній роботі застосунку.

В свою чергу розробка концептуальної моделі бази даних дозволила чітко визначити структуру даних, основні сутності, їхні атрибути та взаємозв’язки між ними. ER діаграма допомогла візуалізувати ці взаємозв’язки та забезпечити точний опис даних, що сприяє ефективному зберіганню та обробці інформації в системі. Таким чином, у другому розділі було закладено основу для подальшої успішної розробки застосунку SpeedLink.

РОЗДІЛ 3. РОЗРОБКА ТА ТЕСТУВАННЯ ЗАСТОСУНКУ SPEEDLINK

3.1 Розробка застосунку «SpeedLink»

3.1.1 Розробка інтерфейсу користувача

Інтерфейс користувача (UI) є одним з найважливіших складових застосунку SpeedLink, оскільки саме через нього користувачі взаємодіють із системою. Важливо, щоб інтерфейс був інтуїтивно зрозумілим, зручним у використанні та забезпечував швидкий доступ до всіх необхідних функцій. Розробка інтерфейсу користувача відбувається саме в Microsoft Visual Studio.

Початковим етапом розробки є створення форми авторизації (див. рисунок 3.1).

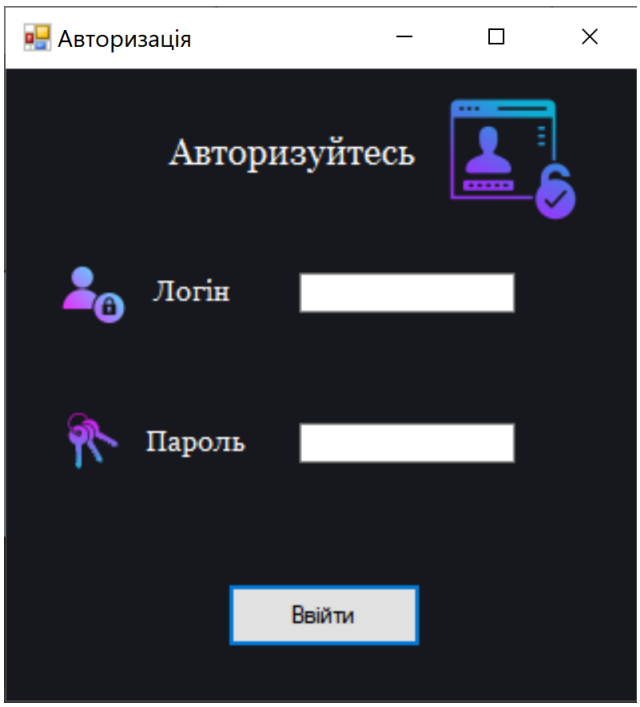

Рисунок 3.1 – Створена форма авторизації

З цієї форми є можливість авторизації, як у вигляді адміністратора, так і у вигляді працівника компанії. Після успішної авторизації застосунок відображатиме функціональні можливості відповідно до ролі користувача. Адміністратор отримує доступ до всіх можливостей системи, включаючи

управління працівниками, тарифами та клієнтами. Працівник отримує доступ до функцій, необхідних для виконання його обов'язків.

Наступним етапом є створення основної форми з меню для користувачів програми (див. рисунок 3.2).

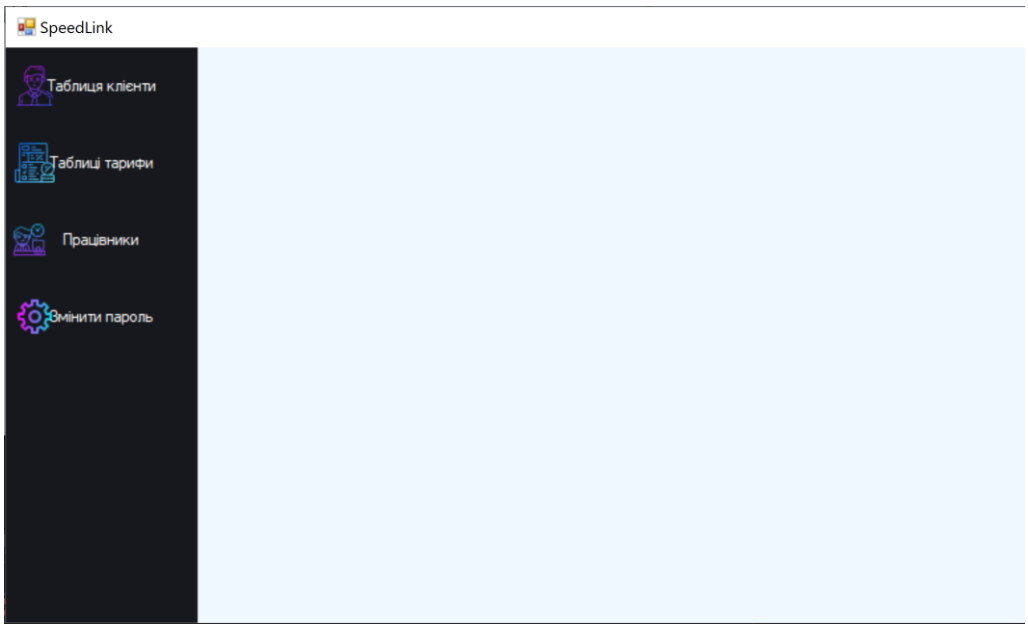

Рисунок 3.2 – Основна форма

Основна форма включає панель навігації з лівого боку, яка містить кнопки для переходу до різних розділів застосунку, таких як "Таблиця клієнтів", "Таблиця тарифів", "Працівники" та "Змінити пароль" з піктограмами, які допомагають користувачам швидко ідентифікувати відповідні розділи і полегшують навігацію.

Вкладка "Таблиця клієнтів" є одним з ключових компонентів інтерфейсу користувача. Вона дозволяє адміністраторам додавати, редагувати та видаляти клієнтів, а також переглядати детальну інформацію про них (див. рисунок 3.3). Користувачі можуть вводити дані про клієнтів, включаючи ім'я, прізвище, контактні дані, вибраний тарифний план та історію платежів. Інтерфейс забезпечує зручний доступ до цієї інформації та дозволяє легко її оновлювати.

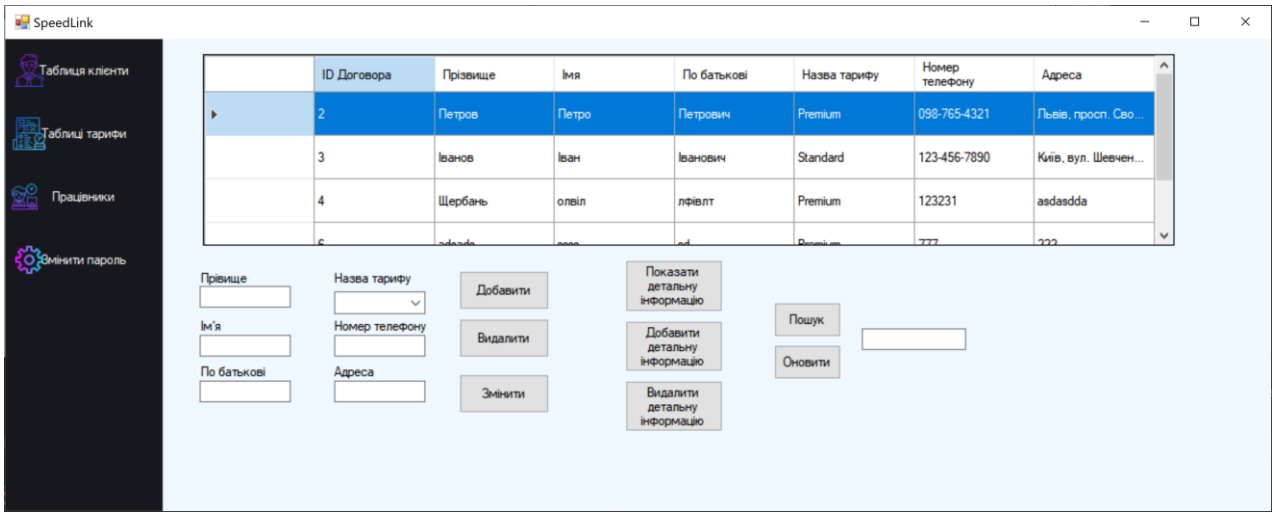

Рисунок 3.3 – Відкрите меню

Інші вкладки, такі як "Таблиця тарифів", "Працівники" та "Змінити пароль", мають подібну структуру та функціональність. Наприклад, вкладка "Таблиця тарифів" дозволяє адміністраторам додавати нові тарифні плани, редагувати існуючі та видаляти застарілі. Інформація про тарифні плани включає назву, вартість, швидкість інтернету та інші важливі характеристики.

Вкладка "Працівники" дозволяє адміністраторам додавати нових працівників, редагувати їхні дані, змінювати паролі та налаштовувати рівні доступу. Важливо, щоб кожен працівник мав доступ лише до тих функцій, які необхідні для виконання його обов'язків. Це забезпечує безпеку даних та знижує ризик несанкціонованого доступу.

Форма зміни пароля дозволяє користувачам змінювати свої паролі для підвищення безпеки облікового запису. Користувач повинен ввести поточний пароль, а також новий пароль і підтвердження нового пароля. Після підтвердження зміни пароля користувач отримує повідомлення про успішну зміну.

Розробка інтерфейсу користувача додатку «SpeedLink» включала багатоетапний процес, що охоплює проектування макетів, створення форм, налаштування елементів керування та тестування інтерфейсу на зручність використання. Використання сучасних технологій та інструментів дозволяє створити інтуїтивно зрозумілий, зручний та ефективний інтерфейс, що забезпечить високу якість обслуговування клієнтів інтернет-провайдера.

Однією з ключових переваг використання подібної уніфікованої структури для всіх вкладок є спрощення процесу навчання користувачів. Коли кожна вкладка має схожий інтерфейс та логіку, користувачам легше розуміти, як користуватися системою. Це зменшує кількість помилок та підвищує ефективність роботи. Також використання уніфікованої структури дозволяє розробникам повторно використовувати код та компоненти, що знижує кількість роботи та ризик помилок [34]. Це також спрощує підтримку системи в майбутньому, оскільки зміни в одному місці можна легко застосувати до інших розділів.

В свою чергу для адміністраторів та розробників така структура полегшує внесення змін та оновлень. Якщо потрібно додати нову функцію або змінити існуючу, це можна зробити одноразово для всіх подібних вкладок, що значно скорочує час на реалізацію змін.

Загалом, використання уніфікованої структури вкладок у додатку SpeedLink забезпечує кращий користувацький досвід, підвищує ефективність роботи користувачів та спрощує процеси розробки, тестування та підтримки системи.

3.1.2 Реалізація функціональних елементів застосунку

Для початку необхідно встановити зв’язок з базою даних Microsoft SQL Server, для цього використовується об'єкт SqlConnection

Для створення об'єкта SqlConnection використовується наступний рядок коду:

```
SqlConnection conn = new SqlConnection(@"Data Source=DESKTOP-P0CN0C3;Initial Catalog=SpeedLinkDB;Integrated Security=True");
```

Цей рядок створює новий екземпляр об'єкта SqlConnection і ініціалізує його параметрами підключення. Параметр Data Source=DESKTOP-P0CN0C3 вказує на сервер бази даних. У цьому випадку сервер бази даних знаходиться на

комп'ютері з іменем DESKTOP-P0CN0C3. Параметр Initial Catalog=SpeedLinkDB вказує на ім'я бази даних, до якої потрібно підключитися. У цьому випадку база даних має назву SpeedLinkDB. Параметр Integrated Security=True вказує на використання автентифікації Windows для підключення до бази даних. Це означає, що для підключення будуть використані облікові дані користувача, який виконує застосунок, замість вказування окремих ім'я користувача та пароля в рядку підключення.

Об'єкт SqlConnection використовується для відкриття і закриття з'єднання з базою даних, а також для виконання SQL-запитів

Щоб реалізувати форму авторизації було використано дві таблиці з бази даних, а саме Workers і AdminCredentials. В лістингу 3.1 показано метод CheckAdminCredentials, що перевіряє чи існує даний адміністратор в БД.

Лістинг 3.1 – Метод для перевірки облікових даних адміністратора

```
private bool CheckAdminCredentials(string login, string password)
{
    try
    {
        conn.Open();
        SqlCommand cmd = new SqlCommand("SELECT * FROM
AdminCredentials WHERE Логін = @login AND Пароль = @password", conn);
        cmd.Parameters.AddWithValue("@login", login);
        cmd.Parameters.AddWithValue("@password", password);
        SqlDataReader reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            reader.Close();
            conn.Close();
            return true;
        }
        else
        {
            reader.Close();
            conn.Close();
```

```
            return false;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Помилка при перевірці облікових даних: " +
ex.Message);
        conn.Close();
        return false;
    }
}
```

Цей метод працює так: створюється SQL-запит для вибору записів з таблиці AdminCredentials, де значення стовпців Логін і Пароль відповідають введеним параметрам. Параметри додаються до запиту для уникнення SQL-ін'єкцій. Запит виконується, і результати зберігаються у SqlDataReader. Якщо reader.Read() повертає true, це означає, що знайдено відповідний запис, і облікові дані правильні. Метод закриває reader і з'єднання з базою даних (conn), а потім повертає true.

Якщо reader.Read() повертає false, це означає, що жодного відповідного запису не знайдено, і облікові дані неправильні. Метод також закриває reader і з'єднання з базою даних (conn), а потім повертає false.

Якщо виникає будь-яка помилка під час виконання коду, вона обробляється в блоці catch. Відображається повідомлення про помилку, з'єднання з базою даних закривається, і метод повертає false.

Аналогічним чином відбувається й перевірка працівників. Повні лістинги форми авторизації прикладено в Додатку А.

Після авторизації в застосунок SpeedLink відображається меню, функціональність якого залежить від даних входу. Для адміністратора доступні всі можливості системи, включаючи управління користувачами, тарифами та перегляд звітів. Працівник має доступ лише до тих функцій, які необхідні для виконання його робочих обов'язків. Це організовано шляхом зміни властивостей компонентів на основній формі, зокрема їх видимості та активності:

```
form2.button11.Enabled = false;
form2.panel3.Enabled = true;
```

Процедури для добавлення інформації в таблиці використовують мову SQL для передавання інформації у Microsoft SQL Server Management Studio. Вони відкривають з'єднання з базою даних, створюють і виконують SQL-запит для вставки даних з текстових полів та «комбобокса» форми, закриває з'єднання, очищує поля форми, оновлює відображення даних і показує повідомлення про успішне додавання запису. Процедуру добавлення інформації в таблицю наведено в лістингу 3.2.

Лістинг 3.2 – Процедура для добавлення інформацію в таблицю «Clients»

```
conn.Open();
 SqlCommand cmd = conn.CreateCommand();
 cmd.CommandType = CommandType.Text;
 cmd.CommandText = "INSERT INTO Clients VALUES ('" + textBox2.Text
+ "','" + textBox3.Text + "','" + textBox4.Text + "','" +
comboBox1.Text + "', '" + textBox6.Text + "', '" + textBox7.Text +
"')";
 cmd.ExecuteNonQuery();
 conn.Close();
 textBox2.Text = "";
 textBox3.Text = "";
 textBox4.Text = "";
 comboBox1.Text = "";
 textBox6.Text = "";
 textBox7.Text = "";
 dispp_data();
 MessageBox.Show("Запис додано в таблицю");
```

В свою чергу видалення інформації з таблиці зроблено командою наведеною нижче:

```
cmd.CommandText = "DELETE FROM Clients WHERE ID_Договора ='" +
textBox1.Text + "'";
```

Для організації пошуку використана процедура яка наведена в лістингу 3.3. Ця процедура виконує пошук у таблиці Clients за значенням, введеним у текстове поле textBox8, і відображає результати у dataGridView1.

Лістинг 3.3 – Процедура що виконує пошук з таблиці «Clients»

```
    conn.Open();
    SqlCommand cmd = conn.CreateCommand();
    cmd.CommandType = CommandType.Text;
    int searchID;
    if (int.TryParse(textBox8.Text, out searchID))
    {
        cmd.CommandText = "SELECT * FROM Clients WHERE ID_Договора
= @ID OR Прізвище = @SearchText OR Імя = @SearchText OR По_батькові
= @SearchText OR Назва_тарифу = @SearchText OR Номер_телефону =
@SearchText OR Адреса = @SearchText";
        cmd.Parameters.AddWithValue("@ID", searchID);
    }
    else {
        cmd.CommandText = "SELECT * FROM Clients WHERE Прізвище =
@SearchText OR Імя = @SearchText OR По_батькові = @SearchText OR
Назва_тарифу = @SearchText OR Номер_телефону = @SearchText OR Адреса
= @SearchText";
    }
    cmd.Parameters.AddWithValue("@SearchText", textBox8.Text);
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    conn.Close();
```

Спочатку відкривається з'єднання з базою даних за допомогою conn.Open(), а потім створюється об'єкт SqlCommand для виконання SQL-запиту.

Далі, процедура перевіряє, чи можна значення з textBox8 конвертувати в ціле число. Якщо це можливо, виконується пошук як за ID_Договора, так і за іншими текстовими полями (Прізвище, Ім'я, По-батькові, Назва тарифу, Номер

телефону, Адреса). Якщо ж значення не можна конвертувати в ціле число, пошук здійснюється тільки за текстовими полями.

Параметри пошуку додаються до SQL-запиту за допомогою cmd.Parameters.AddWithValue(). Після цього створюється об'єкт SqlDataAdapter, який виконує запит і заповнює результати в DataTable. Ці результати прив'язуються до dataGridView1 для відображення користувачу.

Після завершення пошуку з'єднання з базою даних закривається. У випадку виникнення помилки в процесі виконання запиту, з'являється повідомлення з деталями помилки, і з'єднання також закривається.

Лістинги коду основної форми відображенні у Додатку Б.

3.2 Тестування створеного застосунку «SpeedLink»

Для початку тестування застосунку «SpeedLink» буде проводитися від ролі адміністратора, з найбільшими можливостями ввівши логін admin та пароль admin. Першим кроком необхідно можна побачити що є доступ до всього функціоналу застосунку. Одразу ж можна перейти у меню в редагування таблиці «Працівники» та добавити декількох працівників (див. рисунок 3.4).

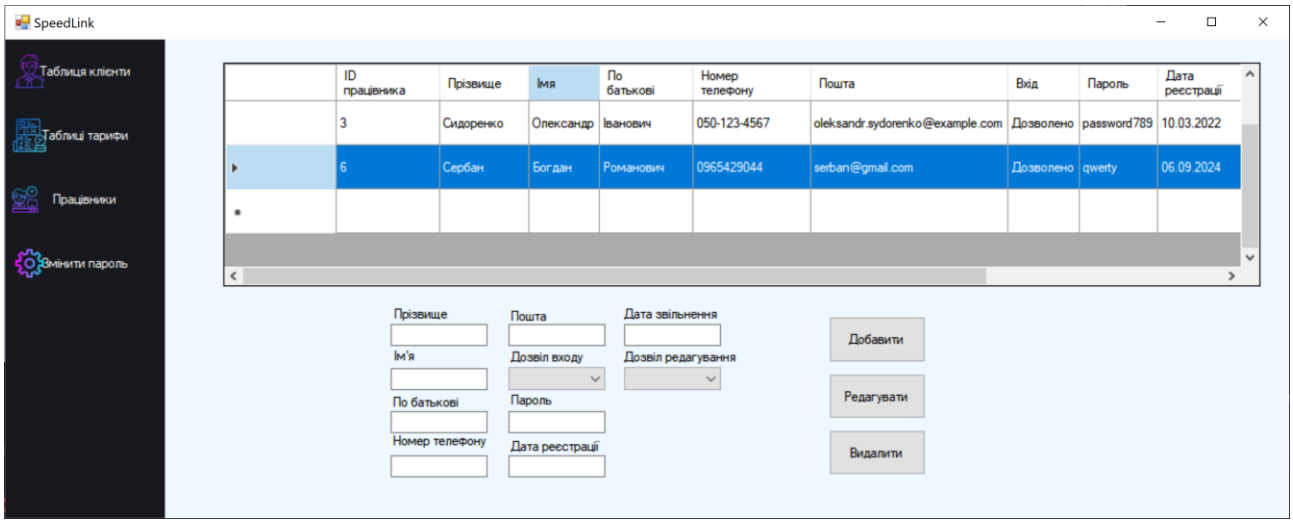

Рисунок 3.4 – Добавлення нового працівника

Ввійшовши в додаток від облікового запису працівника зі збільшеними можливостями одразу можна побачити, що вкладки «Працівники» немає, далі

перейти на вкладку «Таблиця тарифи» та добавити новий тариф під назвою «Luxe» (див. рисунок 3.5).

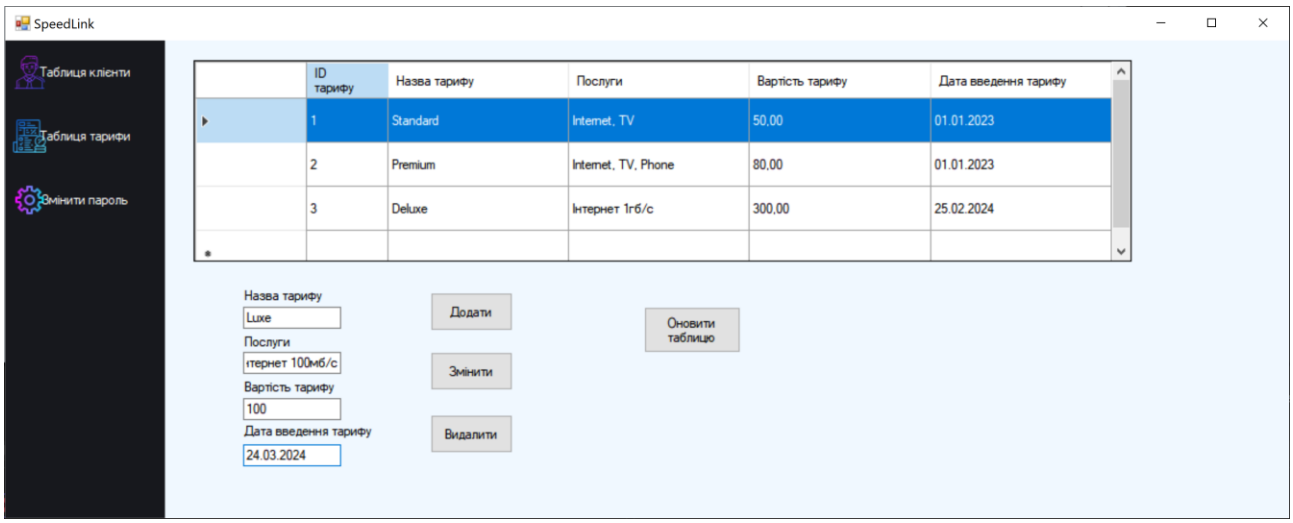

Рисунок 3.5 – Добавлення нового тарифу

Також можна перейти на «Таблиця клієнти» та добавити детальну інформацію про одного клієнта з прізвищем «Соловей» (див. рисунок 3.6).

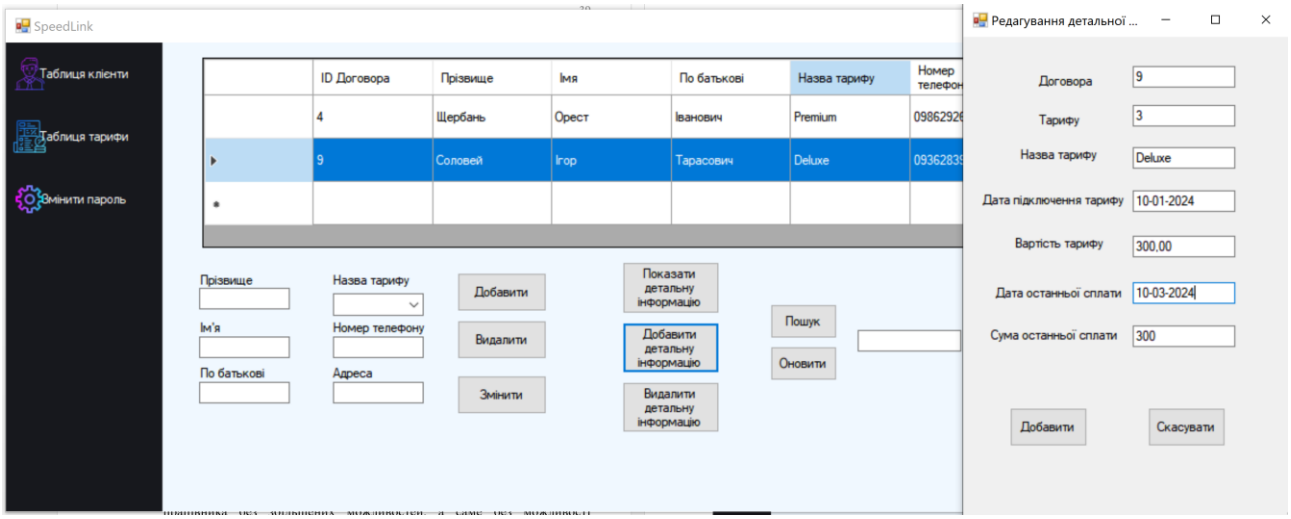

Рисунок 3.6 – Добавлення детальної інформації клієнта

Авторизувавшись працівником без збільшених можливостей, а саме без можливості добавлення та видалення детальної інформації про клієнта та можливості редагування тарифів, для цього використовуються дані для входу, добавлені раніше у базу даних, а саме логін «serban@gmail.com» та пароль

«qwerty». Після авторизації одразу ж можна спостерігати обмежений функціонал при даному варіанті входу (див. рисунок 3.7).

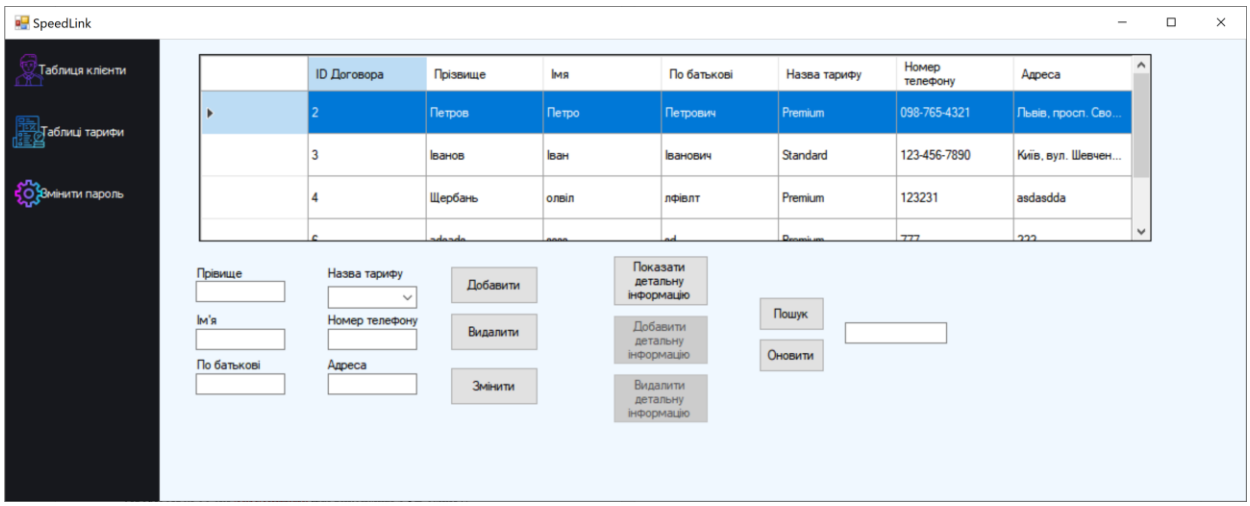

Рисунок 3.7 – Варіант входу від звичайного користувача

Можна також відкрити детальну інформацію доданого раніше для клієнта з прізвищем «Соловей» (див. рисунок 3.8).

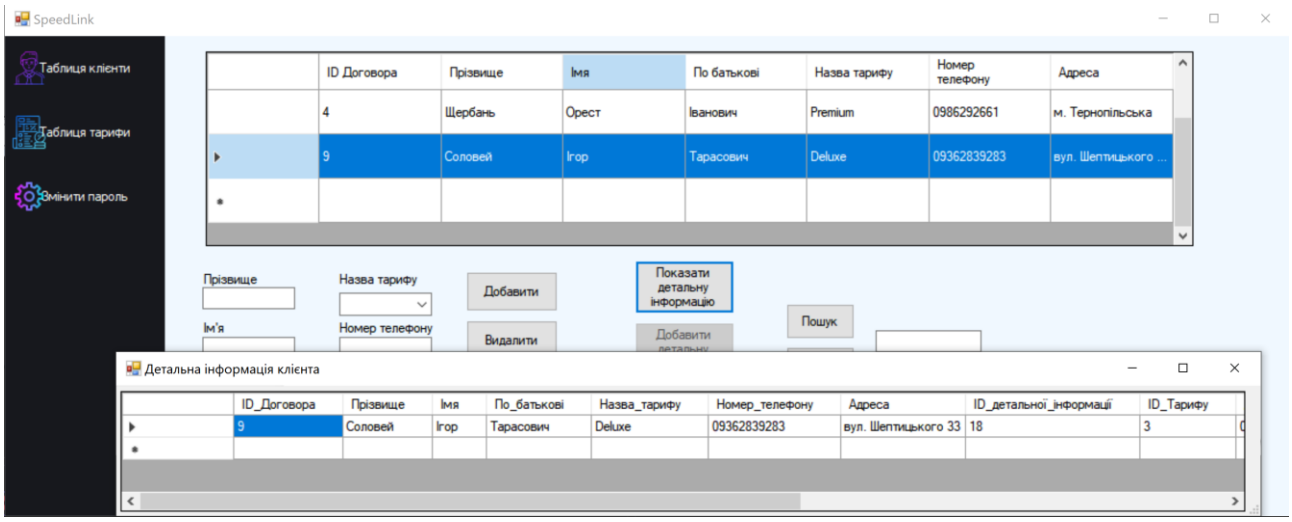

Рисунок 3.8 – Перегляд детальної інформації про клієнта

З входом звичайного працівника можна перевірити одразу ж добавлення нового клієнта в таблицю. Після введення усієї інформації у відповідні поля, та натиснувши кнопку «Добавити» можна побачити, що добавлення пройшло успішно, та одразу видно нового клієнта з ID Договору 10 (див. рисунок 3.9).

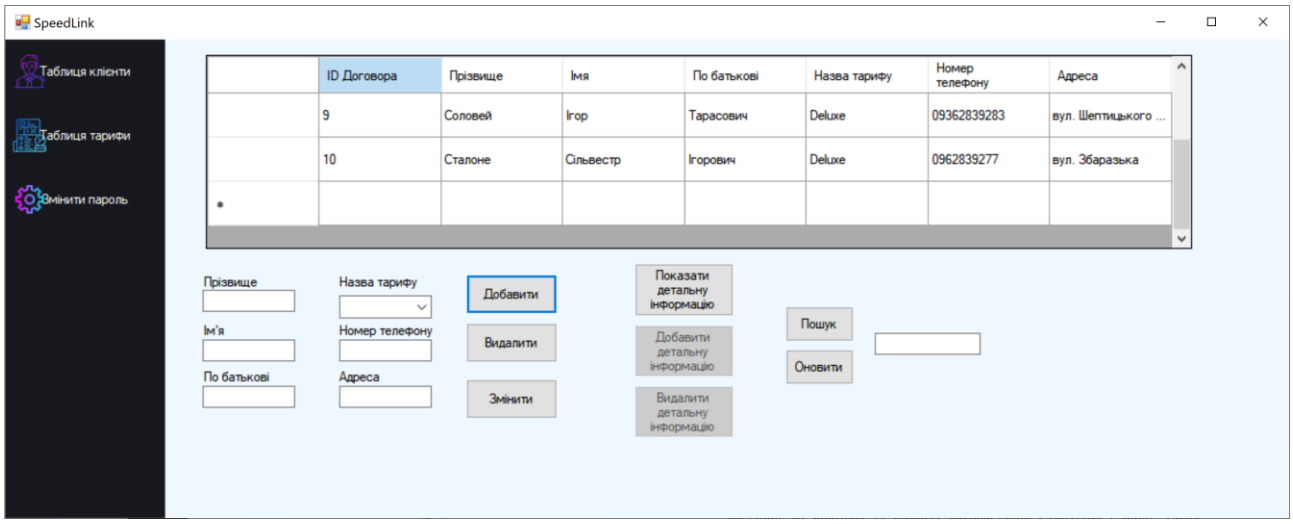

Рисунок 3.9 – Добавлення нового клієнта

Також одразу можна здійснити пошук по клієнту ввівши прізвище «Сталоне» (див. рисунок 3.10).

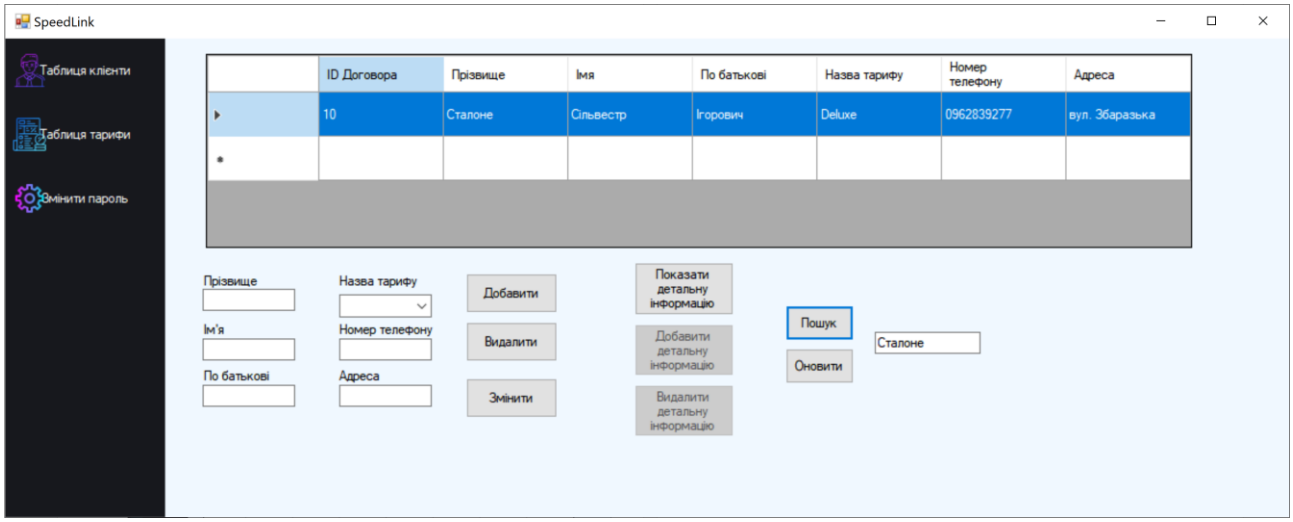

Рисунок 3.10 – Здійснення пошуку клієнта

Можна отримати результат пошуку, а саме відфільтровану таблицю з клієнтами в яких прізвище «Сталоне». Ця функція дозволяє швидко знаходити необхідні записи серед великого обсягу даних, що значно підвищує ефективність роботи.

Далі, система також підтримує функцію заміни логіну та паролю для працівників. Для цього потрібно перейти в спеціальне меню, натиснувши на кнопку «Змінити пароль» (див. рисунок 3.11). У цьому меню користувач вводить свій поточний логін і пароль, а також новий логін і пароль.

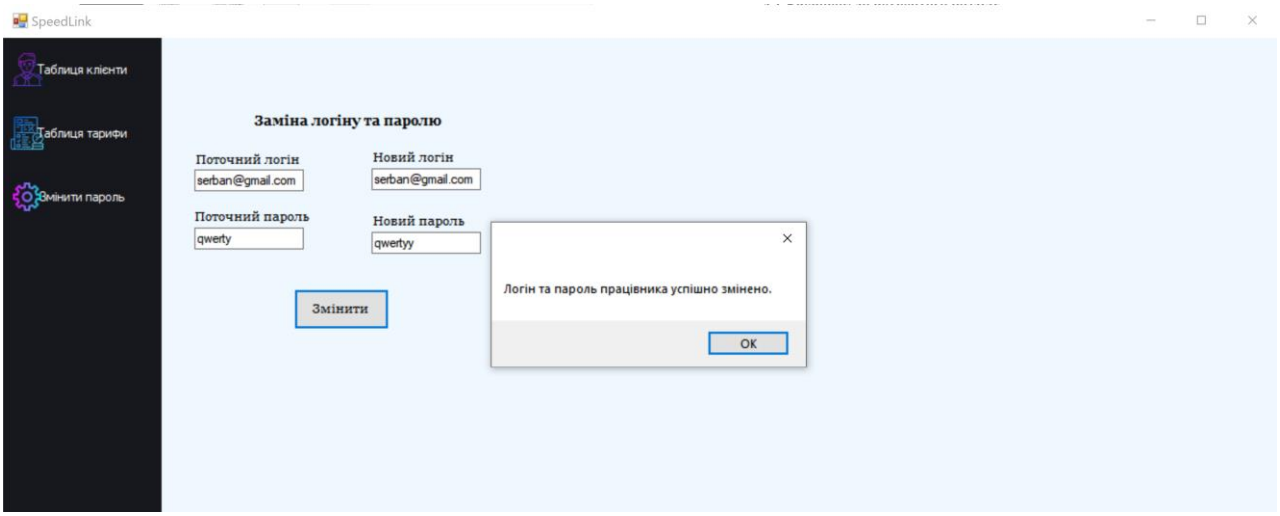

Рисунок 3.11 – Здійснення зміни паролю для працівника

При натисканні на кнопку «Змінити», відкривається спливаюче вікно, яке повідомляє про те, що логін та пароль змінено успішно. Це повідомлення підтверджує, що зміни були збережені в базі даних і нові облікові дані тепер активні. Така функція підвищує рівень безпеки та зручність використання системи.

Після проведених процедур можна сказати, що тестування програми пройшло успішно. Система правильно виконує пошук за введеними критеріями, дозволяє безпечно змінювати облікові дані користувачів, та вести облікові дані клієнтів. Це свідчить про надійність та функціональність програмного забезпечення, яке готове до експлуатації в реальних умовах.

3.3 Висновок до третього розділу

У цьому розділі було розглянуто основні етапи розробки та тестування застосунку SpeedLink. Інтерфейс користувача розроблявся в Microsoft Visual Studio і включав створення форми авторизації та основної форми з панеллю навігації. Форма авторизації дозволяє входити в систему як адміністраторам, так і працівникам, надаючи доступ до функцій відповідно до їхніх ролей.

Функціональні елементи застосунку були реалізовані через взаємодію з базою даних Microsoft SQL Server. Використання об'єкта SqlConnection забезпечило з'єднання з базою даних для виконання SQL-запитів, додавання,

редагування та видалення даних. Методи для перевірки облікових даних адміністратора та працівника гарантували безпеку доступу до системи.

Тестування показало, що система працює коректно для різних ролей користувачів. Адміністратори можуть керувати працівниками, тарифами та клієнтами, тоді як працівники мають доступ лише до необхідних для їхньої роботи функцій.

Розробка та тестування SpeedLink продемонстрували, що застосунок є зручним у використанні, надійним та безпечним. Інтерфейс користувача забезпечує легку взаємодію з системою, а функціональні можливості дозволяють ефективно керувати інформацією про клієнтів та послуги інтернет-провайдера.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Надзвичайні ситуації: визначення причини, класифікація

Надзвичайна ситуація — обстановка на окремій території, суб'єкті господарювання або водному об'єкті, яка характеризується порушенням нормальних умов життєдіяльності населення, спричинена катастрофою, аварією, пожежею, стихійним лихом, епідемією, епізоотією, епіфітотією, застосуванням засобів ураження або іншою небезпечною подією, що призвела (може призвести) до виникнення загрози життю або здоров'ю населення, великої кількості загиблих і постраждалих, завдання значних матеріальних збитків, а також до неможливості проживання населення на такій території чи об'єкті, провадження на ній господарської діяльності.

НС, які можуть виникати на території України в мирний і воєнний час негативно впливають на функціонування об'єктів економіки та життєдіяльність населення. Для організації ефективної роботи із запобігання надзвичайним ситуаціям, ліквідації їхніх наслідків, зниження масштабів втрат та збитків дуже важливо знати причини їх виникнення.

Надзвичайні ситуації, відповідно до кодексу цивільного захисту, класифікують:

- за характером походження або причиною виникнення;
- ступенем поширення;
- розміром людських утрат та матеріальних збитків.

Залежно від характеру походження подій, що можуть зумовити виникнення НС

на території України, визначаються такі види надзвичайних ситуацій:

- техногенного характеру;
- природного характеру;
- соціальні;
- воєнні

НС техногенного характеру — це транспортні аварії (катастрофи), пожежі, неспровоковані вибухи чи їх загроза, аварії з викидом небезпечних хімічних, радіоактивних, біологічних речовин, раптове руйнування споруд та будівель, аварії на інженерних мережах і спорудах життєзабезпечення, гідродинамічні аварії на греблях, дамбах тощо.

НС природного характеру, або стихійні лиха — це небезпечні геологічні, метеорологічні, гідрологічні морські та прісноводні явища, деградація ґрунтів чи надр, природні пожежі, зміна стану повітряного басейну, інфекційна захворюваність людей, сільськогосподарських тварин, масове ураження сільськогосподарських рослин хворобами чи шкідниками, зміна стану водних ресурсів та біосфери тощо.

Соціальні НС — порушення нормальних умов життя та діяльності людей на окремій території чи об’єкті на ній або на водному об’єкті, спричинене протиправними діями терористичного і антиконституційного спрямування або пов’язане із зникненням (викраденням) зброї та небезпечних речовин, нещасними випадками з людьми тощо.

Воєнні НС — порушення нормальних умов життя та діяльності людей на окремій:

– території чи об’єкті на ній або на водному об’єкті, спричинене застосуванням звичайної

– зброї або зброї масового ураження, під час якого виникають вторинні чинники ураження

– населення, визначені в окремих документах

НС техногенного характеру називають ще антропогенними, тобто ті, що виникають на об’єктах, створених людьми, і які вона використовує. Великі аварії та катастрофи на об’єктах можуть виникнути внаслідок стихійного лиха, а також порушень технології виробництва, правил експлуатації різних машин, обладнання і встановлених норм безпеки.

Виробничі, транспортні та побутові аварії та катастрофи за наслідками не поступаються стихійним лихам. Унаслідок техногенних аварій і катастроф

виникають ситуації, які призводять до значних збитків, виникає необхідність захисту людей від дії шкідливих чинників, проведення рятувальних, медичних і евакуаційних заходів [42].

4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК.

Однією із характерних особливостей сучасного розвитку суспільства є зростання сфер діяльності людини, в яких використовуються інформаційні технології. Широке розповсюдження отримали персональні комп'ютери. Однак їх використання загострило проблеми збереження власного та суспільного здоров'я, вимагає удосконалення існуючих та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку негативних наслідків впливу ПК на здоров'я користувачів.

Заходи з охорони праці користувачів ПК необхідно розглядати в трьох основних аспектах: соціальному, психологічному та медичному.

У соціальному плані розв'язання цих проблем пов'язане з оптимізацією умов життя, праці, відпочинку, харчування, побуту, розвитком культури, транспорту.

Психологічний аспект. Значне місце у профілактиці розладів здоров'я належить психології праці. Тому заходи, пов'язані з формуванням раціональних виробничих колективів, у яких відсутня психологічна несумісність, сприяють зменшенню нервово-психічного перенапруження, підвищенню працездатності та ефективності праці. Особливої значущості у користувачів відеодисплейних терміналів набуває психоемоційний стрес, який більшою або меншою мірою проявляється у кожного з них.

Медичний аспект. Значна роль у профілактиці захворювань користувачів ПК відводиться медицині. Існує перелік профілактичних заходів для користувачів ПК, що включає як складові первинної профілактики здоров'я (професійний відбір), так і вторинної, яка направлена на зниження ймовірності розвитку перевтоми та перенапруження. Ці комплексні заходи спрямовані на відновлення функціонального стану зорового та опорно-рухового апарату [43].

Також існують вимоги щодо приміщень користувачів ПК:

Природне освітлення в приміщеннях для роботи з відеотерміналами (ВДТ) повинне здійснюватися через світлові прорізи, орієнтовані переважно на північ або північний схід, і забезпечувати коефіцієнт природної освітленості (КПО) не нижче ніж 1,5%. КПО розраховується за методикою, викладеною в ДБН В.2.5-28-2006. Виробничі приміщення для роботи з ВДТ, такі як операторські та диспетчерські, не повинні межувати з приміщеннями, в яких рівні шуму та вібрації перевищують допустимі значення, відповідно до стандартів. Звукоізоляція огороджувальних конструкцій повинна відповідати вимогам СН 3223-85, ГОСТ 12.1.003-83, ГОСТ 12.1.012-90. Приміщення повинні бути обладнані системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією, параметри мікроклімату повинні відповідати стандартам.

Віконні прорізи в таких приміщеннях повинні бути оснащені регульованими пристроями, такими як жалюзі, завіски або зовнішні козирки. Для внутрішнього оздоблення приміщень слід використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття 0,7-0,8 для стелі та 0,5-0,6 для стін. Покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3-0,5, рівним, неслизьким та антистатичним. Забороняється використовувати полімерні матеріали для оздоблення інтер'єру, якщо вони виділяють шкідливі хімічні речовини. Використання полімерних матеріалів можливе лише за наявності дозволу від державної санітарно-епідеміологічної служби.

Робочі місця працівників з екранними пристроями мають бути спроектовані та мати такі розміри, щоб працівники мали простір для зміни робочого положення та рухів. Для забезпечення безпеки та захисту здоров’я працівників усе випромінювання від екранних пристроїв має бути зведене до гранично допустимого рівня. Це стосується впливу на людину таких факторів довкілля, як шум, вібрація, забруднювачі, температура тощо, які не спричиняють соматичних або психічних розладів, а також змін стану здоров’я, працездатності, поведінки, що виходять за межі пристосувальних реакцій з погляду безпеки та охорони здоров’я працівників. [44].

4.3 Висновок до четвертого розділу

В четвертому розділі кваліфікаційної роботи описано значення адаптації в трудовому процесі та загальні вимоги безпеки з охорони праці для користувачів ПК. Підкреслено важливість адаптаційних процесів, які сприяють підвищенню працездатності, зниженню рівня захворюваності та травматизму серед працівників. Розглянуто чотири основні типи адаптації: фізіологічну, психічну, соціальну та професійну. Кожен з них має свої особливості та вплив на загальний стан працівника і його здатність ефективно виконувати робочі завдання.

Детально розглянуто загальні вимоги безпеки з охорони праці для користувачів ПК. Визначено, що для забезпечення безпеки при роботі з комп'ютерами необхідно дотримуватися ряду правил, які включають правильне розташування обладнання, оптимальне освітлення робочого місця, дотримання дистанції до монітора та правила використання клавіатури і миші. Також наголошено на важливості ознайомлення працівників з діями у критичних або аварійних ситуаціях для забезпечення їхньої безпеки та зниження ризиків.

Таким чином, розділ надає комплексне уявлення про значення адаптації в трудовому процесі та вимоги з охорони праці для користувачів ПК, що сприяє покращенню умов праці та підвищенню безпеки на робочих місцях.

ВИСНОВКИ

У даній кваліфікаційній роботі було досліджено та розроблено програмне забезпечення для обліку клієнтів інтернет-провайдера "SpeedLink". Робота охоплює теоретичні та практичні аспекти, спрямовані на створення ефективної, надійної та зручної системи для управління інформацією про клієнтів.

У першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– подано детальний аналіз предметної області, зокрема систем обліку клієнтів;

– розглянуто переваги та недоліки існуючих рішень для обліку клієнтів;

– висвітлено вимоги до програмного забезпечення для інтернет-провайдерів;

– проаналізовано вибір використовуваних технологій для розробки застосунку "SpeedLink".

У другому розділі кваліфікаційної роботи:

– досліджено архітектурні підходи та обґрунтовано вибір патерну "Команда" для побудови системи;

– обґрунтовано структуру бази даних у Microsoft SQL Server для зберігання інформації про клієнтів, працівників та тарифні плани;

– сформовано концептуальну модель бази даних та ER-діаграму, яка візуалізує сутності та їх взаємозв’язки.

У третьому розділі кваліфікаційної роботи:

– розроблено інтерфейс користувача в Microsoft Visual Studio, включаючи форму авторизації та основну форму з панеллю навігації;

– запропоновано функціональні елементи для додавання, редагування, видалення та пошуку інформації в базі даних;

– спроєктовано та протестовано застосунок "SpeedLink" для різних ролей користувачів (адміністратори та працівники), що підтвердило його коректну роботу.

У розділі «Безпека життєдіяльності, основи охорони праці» висвітлено ергономічні проблеми безпеки життєдіяльності, які виникають при роботі з ПК.

Також розглянуто загальні вимоги безпеки з охорони праці для користувачів ПК, що сприяють створенню безпечного та здорового робочого середовища.

Отже, виконана робота охоплює детальний аналіз, проєктування, розробку та тестування програмного забезпечення для обліку клієнтів інтернет-провайдера "SpeedLink", що забезпечує його ефективність, зручність у використанні та високий рівень безпеки.

ПЕРЕЛІК ДЖЕРЕЛ

1. Роль інформаційних технологій в роботі сучасного працівника – UA5.org. *UA5.org – Матеріали з інформаційних технологій*. URL: https://ua5.org/technol/1679-rol-informaczijnyh-tehnologij-v-roboti-suchasnogo-praczivnyka.html (дата звернення: 04.02.2024).

2. Що таке CRM-система та як вона працює? | Creatio. *No-Code Platform to Automate Workflows and CRM | Creatio*. URL: https://www.creatio.com/ua/crm/what-is-crm (дата звернення: 04.02.2024).

3. Customer Database: Definition, Importance, Steps & Tips. *QuestionPro*. URL: https://www.questionpro.com/blog/customer-database/ (дата звернення: 04.02.2024).

4. Принципи роботи та функції програми CRM: сучасний підхід до управління клієнтськими відносинами. *Muzyka.home.cx.ua*. URL: https://muzyka.home.cx.ua/ukraincyam/principi-roboti-ta-funkcii-programi-crm-suchasniy-pidkhid-do-upravlinnya-kliientskimi-vidnosinami.html (дата звернення: 04.02.2024).

5. Що таке CRM: визначення, типи та стратегії | Snov.io. *Sales automation & acceleration at scale | Snov.io*. URL: https://snov.io/glossary/ua/customer-relationship-management-crm-ua/ (дата звернення: 04.02.2024).

6. Учасники проектів Вікімедіа. Управління клієнтами. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Управління_відносинами_з_клієнтами (дата звернення: 04.02.2024).

7. Ecommerce Personalization Powered by AI | Bloomreach. *Ecommerce Personalization Powered by AI | Bloomreach*. URL: https://www.bloomreach.com/en (дата звернення: 04.05.2024).

8. Accounting Automation: The Definitive Guide - Future Firm. *Future Firm*. URL: https://futurefirm.co/accounting-automation/ (дата звернення: 04.02.2024).

9. Гончарук М. IT і бізнес: Як технології впливають на розвиток сучасних підприємств. *Lemon School*. URL: https://lemon.school/blog/it-i-biznes-yak-tehnologiyi-vplyvayut-na-rozvytok-suchasnyh-pidpryyemstv (дата звернення: 05.02.2024).

10. Як і навіщо збирати дані клієнта - Блог про email та інтернет-маркетинг. *Блог про email та інтернет-маркетинг*. URL: https://sendpulse.ua/blog/collecting-customer-data (дата звернення: 05.02.2024).

11. Zoho CRM | Die CRM-Vertriebssoftware mit der besten Kundenbewertung. *Zoho*. URL: https://www.zoho.com/crm/ (дата звернення: 15.06.2024).

12. Zoho CRM Overview. What is Zoho CRM?. *TrustRadius*. URL: https://www.trustradius.com/products/zoho-crm/reviews#pricing (дата звернення: 05.05.2024).

13. Rist O. HubSpot CRM Review. *Pc Mag*. URL: https://www.pcmag.com/reviews/hubspot-crm (дата звернення: 05.05.2024).

14. Hajric S. HubSpot CRM Review 2024: Pricing, Features, Pros & Cons. *CRM.org*. URL: https://crm.org/news/hubspot-crm-review (дата звернення: 06.02.2024).

15. Home. *Act!*. URL: https://www.act.com/ (date of access: 05.02.2024).

16. Act! CRM Review (2024): Features, Pricing, Pros & Cons. *TechRepublic*. URL: https://www.techrepublic.com/article/act-crm-review/ (дата звернення: 05.05.2024).

17. The Pros and Cons of using Trello Software. *ProjectManagers.net*. URL: https://projectmanagers.net/the-pros-and-cons-of-using-trello-software/ (дата звернення: 07.05.2024).

18. Define Target Audience for App Success: A Guide to Better User Experience - ApeScript. *ApeScript*. URL: https://www.apescript.com/mobile-app-dev/target-audience-for-app-success/ (дата звернення: 08.05.2024).

19. System administrator responsibilities: 9 critical tasks. *Opensource.com*. URL: https://opensource.com/article/19/7/sysadmin-best-practices (дата звернення: 08.05.2024).

20. BMC Software. *BMC Software*. URL: https://www.bmc.com/ (дата звернення: 08.05.2024).

21. User interface це про забезпечення користувачам зручності. *FoxmindEd*. URL: https://foxminded.ua/user-interface-tse/ (дата звернення: 08.05.2024).

22. Клієнти - розділ Perfectum CRM+ERP. Perfectum CRM+ERP. URL: https://perfectum.ua/ua/documentation/customers/customer (дата звернення: 08.05.2024).

23. The Role of Secure Authentication and Authorization in Application Security. *Bright Security*. URL: https://brightsec.com/blog/the-role-of-secure-authentication-and-authorization-in-application-security/ (дата звернення: 18.05.2024).

24. Classes, structs, and records - C#. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/ (дата звернення: 18.05.2024).

25. Introduction to .NET - .NET. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/dotnet/core/introduction (дата звернення: 18.05.2024).

26. Exception Handling in C# - Code Maze. *Code Maze*. URL: https://code-maze.com/csharp-exception-handling/ (дата звернення: 18.05.2024).

27. Опис продукту Microsoft Visual Studio Professional - ITPRO.UA. *ITPRO*. URL: https://itpro.ua/catalog/catalogarticle/view/visual-studio-professional/?tab=description (дата звернення: 25.05.2024).

28. Учасники проектів Вікімедіа. Microsoft SQL Server. *Вікіпедія*. URL : https://uk.wikipedia.org/wiki/Microsoft_SQL_Server (дата звернення: 25.05.2024).

29. Команда. *Refactoring and Design Patterns*. URL: https://refactoring.guru/uk/design-patterns/command (дата звернення: 25.05.2024).

30. Design Patterns and Refactoring. *Design Patterns & Refactoring*. URL: https://sourcemaking.com/design_patterns/command (дата звернення: 25.05.2024).

31. Diagram maker for developers. *Diagram maker for developers | Gleek*. URL: https://www.gleek.io/blog/conceptual-data-model (дата звернення: 26.04.2024).

32. Arora Y. Conceptual Database Modeling 101: A Complete Guide, Simplified. *Learn | Hevo*. URL: https://hevodata.com/learn/conceptual-database/ (дата звернення: 26.04.2024).

33. Що таке ER Modeling? Навчайтеся на прикладі. *Guru99*. URL: https://www.guru99.com/uk/er-modeling.html (дата звернення: 26.04.2024).

34. The Importance of a Unified UI/UX. *Eleveo: Optimizing Your Contact Center*. URL: https://www.eleveo.com/blog/the-importance-of-a-unified-ui-ux (дата звернення: 26.04.2024).

35. Fryz M., Mlynko B. Property Analysis of Conditional Linear Random Process as a Mathematical Model of Cyclostationary Signal // 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022). Ternopil, Ukraine: CEUR Workshop Proceedings, 2022. Vol. 3309. P. 77–82.

36. Fryz M., Mlynko B. Determination of the characteristic function of discrete-time conditional linear random process and its application // Sci. J. TNTU. 2023. Vol. 109, № 1. P. 16–23.

37. Fryz M., Mlynko B. Property analysis of multivariate conditional linear random processes in the problems of mathematical modelling of signals // Technol. Audit Prod. Reserv. 2022. Vol. 3, № 2(65). P. 29–32.

38. Фриз М.Є., Млинко Б.Б. Умовні лінійні випадкові процеси з дискретним часом та їх властивості // Вісник Хмельницького національного університету. Серія: Технічні науки. 2022 (309), № 3. С. 7–12.

39. Fryz M., Mlynko B. Properties of Stationarity and Cyclostationarity of Conditional Linear Random Processes // 2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET). Lviv-Slavske, Ukraine: IEEE, 2020. P. 166–170.

40. Fryz M., Scherbak L., Mlynko B., Mykhailovych T. Linear Random Process Model-Based EEG Classification Using Machine Learning Techniques // Proceedings of the 1st International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2023). Ternopil, Ukraine: CEUR Workshop Proceedings, 2023. Vol. 3468. P. 126–132.

41. Бабак В. П., Марченко Б. Г., Фриз М. Є. Теорія ймовірностей, випадкові процеси та математична статистика. К.: Техніка, 2004. 288 с.

42. Захист Вітчизни : підручник для 10 класу закладів загальної середньої освіти. Рівень стандарту. // І. М. Гарасимів, К. О. Пашко, М. М. Фука, Ю. П. Щирба. — Тернопіль : Астон, 2018. — 256 с

43. Пойда В. Ю. Курс лекцій з «Охорони праці в галузі». Ужгород : Ужгор. Нац. Ун-т, 2017. 89 с.

44. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями : Постанова Каб. Міністрів України від 14.02.2018 р. № 207

ДОДАТКИ

Додаток А

Програмний код до форми авторизації

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace SpeedLinkV2ap
{
    public partial class Form1 : Form
    {
        SqlConnection    conn    =    new    SqlConnection(@"Data
Source=DESKTOP-P0CN0C3;Initial       Catalog=SpeedLinkDB;Integrated
Security=True");
        public Form1()
        {
            InitializeComponent();
            this.FormClosing                 +=                 new
FormClosingEventHandler(Form1_FormClosing);
        }
        private     void     Form1_FormClosing(object     sender,
FormClosingEventArgs e)
        {
            Application.Exit();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string login = textBox1.Text;
            string password = textBox2.Text;

            if (CheckAdminCredentials(login, password))
            {
                OpenAdminForm();
            }
            else
            {
                CheckWorkerCredentials(login, password);
            }
        }
        private  bool  CheckAdminCredentials(string  login,  string
password)
        {
            try
            {
                conn.Open();
```

```
SqlCommand cmd = new SqlCommand("SELECT * FROM AdminCredentials
WHERE Логін = @login AND Пароль = @password", conn);
                cmd.Parameters.AddWithValue("@login", login);
                cmd.Parameters.AddWithValue("@password", password);
                SqlDataReader reader = cmd.ExecuteReader();
                if (reader.Read())
                {
                    reader.Close();
                    conn.Close();
                    return true;
                }
                else
                {
                    reader.Close();
                    conn.Close();
                    return false;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при перевірці облікових
даних: " + ex.Message);
                conn.Close();
                return false;
            }
        }
        private void OpenAdminForm()
        {
            Form2 form2 = new Form2();
            form2.button10.Enabled = true;
            form2.button11.Enabled = true;
            form2.panel3.Enabled = true;
            form2.button3.Visible = true;
            form2.Show();
            this.Hide();
        }
        private void CheckWorkerCredentials(string login, string
password)
        {
            try
            {
                conn.Open();
                SqlCommand cmd = new SqlCommand("SELECT * FROM
Workers WHERE Пошта = @login AND Пароль = @password", conn);
                cmd.Parameters.AddWithValue("@login", login);
                cmd.Parameters.AddWithValue("@password", password);

                SqlDataReader reader = cmd.ExecuteReader();

                if (reader.Read())
                {
                    string access = reader["Вхід"].ToString();
```

```
      string                    editPermission                    =
reader["Дозвіл_редагування"].ToString();

                    if (access == "Дозволено")
                    {
                        OpenWorkerForm(editPermission);
                    }
                    else
                    {
                        MessageBox.Show("Вхід заборонено");
                    }
                }
                else
                {
                    MessageBox.Show("Невірний логін або пароль");
                }

                reader.Close();
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка  при  перевірці  облікових
даних: " + ex.Message);
                conn.Close();
            }
        }
        private void OpenWorkerForm(string editPermission)
        {
            Form2 form2 = new Form2();
            if (editPermission == "Дозволено")
            {
                form2.button10.Enabled = true;
                form2.button11.Enabled = true;
                form2.panel3.Enabled = true;
                form2.button3.Visible = false;
            }
            else
            {
                form2.button10.Enabled = false;
                form2.button11.Enabled = false;
                form2.panel3.Enabled = false;
                form2.button3.Visible = false;
            }
            form2.Show();
            this.Hide();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
        }
    }
}
```

Додаток Б

Програмний код до основної форми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SpeedLinkV2ap
{
    public partial class Form2 : Form
    {
        SqlConnection conn = new SqlConnection(@"Data
Source=DESKTOP-P0CN0C3;Initial Catalog=SpeedLinkDB;Integrated
Security=True");
        private string adminLogin = "Admin";
        private string adminPassword = "Admin";
        public Form2()
        {
            InitializeComponent();
            this.FormClosed += new
FormClosedEventHandler(Form2_FormClosed);
        }
        private void Form2_FormClosed(object sender,
FormClosedEventArgs e)
        {
            Form1 form1 = new Form1();
            form1.Show();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            panel1.Visible = true;
            panel2.Visible = false;
            panel4.Visible = false;
            panel5.Visible = false;
        }
        private void button2_Click(object sender, EventArgs e)
        {
            panel1.Visible = false;
            panel2.Visible = true;
            panel4.Visible = false;
            panel5.Visible = false;
        }
        private void button3_Click(object sender, EventArgs e)
        {
```

```
panel1.Visible = false;
        panel2.Visible = false;
        panel4.Visible = true;
        panel5.Visible = false;
    }
    private void button4_Click(object sender, EventArgs e)
    {
        panel1.Visible = false;
        panel2.Visible = false;
        panel4.Visible = false;
        panel5.Visible = true;
    }
    public void dispp_data()
    {
        conn.Open();
        SqlCommand cmd = conn.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT * FROM Clients";
        cmd.ExecuteNonQuery();
        DataTable dt = new DataTable();
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        da.Fill(dt);
        dataGridView1.DataSource = dt;
        conn.Close();
    }
    public void disp_data()
    {
        conn.Open();
        SqlCommand cmd = conn.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT * FROM Tariffs";
        cmd.ExecuteNonQuery();
        DataTable dt = new DataTable();
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        da.Fill(dt);
        dataGridView2.DataSource = dt;
        conn.Close();
    }
    public void displ_data()
    {
        conn.Open();
        SqlCommand cmd = conn.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT * FROM Workers";
        cmd.ExecuteNonQuery();
        DataTable dt = new DataTable();
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        da.Fill(dt);
        dataGridView3.DataSource = dt;
        conn.Close();
    }
    private void Clien_Load(object sender, EventArgs e)
```

```
   {
            dispp_data();
        }
        private void Form2_Load(object sender, EventArgs e)
        {

this.workersTableAdapter.Fill(this.speedLinkDBDataSet.Workers)
;

this.tariffsTableAdapter.Fill(this.speedLinkDBDataSet.Tariffs)
;

this.clientsTableAdapter.Fill(this.speedLinkDBDataSet.Clients)
;
            this.dataGridView1.SelectionChanged     +=      new
System.EventHandler(this.dataGridView1_SelectionChanged);
            this.dataGridView2.SelectionChanged     +=      new
System.EventHandler(this.dataGridView2_SelectionChanged);
            this.dataGridView3.SelectionChanged     +=      new
System.EventHandler(this.dataGridView3_SelectionChanged);
            dispp_data();
            disp_data();
            displ_data();
            LoadTariffsIntoComboBox();
        }
        private   void   dataGridView1_CellContentClick(object
sender, DataGridViewCellEventArgs e)
        {
        }
        private   void   dataGridView1_SelectionChanged(object
sender, EventArgs e)
        {
            if (dataGridView1.SelectedRows.Count > 0)
            {
                int             selectedRowIndex             =
dataGridView1.SelectedRows[0].Index;
                int                 selectedID                =
Convert.ToInt32(dataGridView1.Rows[selectedRowIndex].Cells[0].
Value);
                textBox1.Text = selectedID.ToString();
            }
            if (dataGridView1.SelectedRows.Count > 0)
            {
                int             selectedRowIndex             =
dataGridView1.SelectedRows[0].Index;
                string           selectedTariffName          =
dataGridView1.Rows[selectedRowIndex].Cells[4].Value.ToString()
;
                textBox5.Text = selectedTariffName;
            }
        }
```

```
  private void dataGridView2_SelectionChanged(object sender,
EventArgs e)
        {
            if (dataGridView2.SelectedRows.Count > 0)
            {
                int              selectedRowIndex              =
dataGridView2.SelectedRows[0].Index;
                int                  selectedID                =
Convert.ToInt32(dataGridView2.Rows[selectedRowIndex].Cells[0].
Value);
                textBox13.Text = selectedID.ToString();
            }
        }
        private   void   dataGridView3_SelectionChanged(object
sender, EventArgs e)
        {
            if (dataGridView3.SelectedRows.Count > 0)
            {
                int              selectedRowIndex              =
dataGridView3.SelectedRows[0].Index;
                int                  selectedID                =
Convert.ToInt32(dataGridView3.Rows[selectedRowIndex].Cells[0].
Value);
                textBox19.Text = selectedID.ToString();
            }
        }
        private void button5_Click(object sender, EventArgs e)
        {
            Form3 fm3 = new Form3();
            fm3.textBox1.Text = this.textBox1.Text;
            fm3.ShowDialog();
        }
        private void button6_Click(object sender, EventArgs e)
        {
            conn.Open();
            SqlCommand cmd = conn.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "INSERT INTO Clients VALUES ('" +
textBox2.Text + "','" + textBox3.Text + "','" + textBox4.Text +
"','" + comboBox1.Text + "', '" + textBox6.Text + "', '" +
textBox7.Text + "')";
            cmd.ExecuteNonQuery();
            conn.Close();
            textBox2.Text = "";
            textBox3.Text = "";
            textBox4.Text = "";
            comboBox1.Text = "";
            textBox6.Text = "";
            textBox7.Text = "";
            dispp_data();
            MessageBox.Show("Запис додано в таблицю");
        }
```

```
  private void button7_Click(object sender, EventArgs e)
        {
            conn.Open();
            SqlCommand cmd = conn.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText  =  "DELETE  FROM  Clients  WHERE
ID_Договора ='" + textBox1.Text + "'";
            cmd.ExecuteNonQuery();
            conn.Close();
            dispp_data();
            MessageBox.Show("Запис видалено успішно");
        }
        private void button8_Click(object sender, EventArgs e)
        {
            conn.Open();
            SqlCommand cmd = conn.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "UPDATE Clients SET Прізвище='" +
textBox2.Text + "', Імя='" + textBox3.Text + "', По_батькові='"
+ textBox4.Text + "', Назва_тарифу='" + comboBox1.Text + "',
Номер_телефону='"  +  textBox6.Text  +  "',  Адреса='"  +
textBox7.Text + "' WHERE ID_Договора ='" + textBox1.Text + "'";
            cmd.ExecuteNonQuery();
            conn.Close();
            dispp_data();
            MessageBox.Show("Запис змінено успішно");
        }
        public int GetTariffIDByName(string tariffName)
        {
            int tariffID = -1;
            try
            {
                conn.Open();
                SqlCommand  cmd  =  new  SqlCommand("SELECT
ID_Тарифу FROM Tariffs WHERE Назва_тарифу = @TariffName", conn);
                cmd.Parameters.AddWithValue("@TariffName",
tariffName);
                SqlDataReader reader = cmd.ExecuteReader();
                if (reader.Read())
                {
                    tariffID                                =
Convert.ToInt32(reader["ID_Тарифу"]);
                }
                reader.Close();
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка  при  пошуку  ID_Тарифу:
" + ex.Message);
                conn.Close();
            }
```

```
    return tariffID;
        }
        public decimal GetTariffCostByName(string tariffName)
        {
            decimal tariffCost = -1;
            try
            {
                conn.Open();
                SqlCommand cmd = new SqlCommand("SELECT
Вартість_тарифу FROM Tariffs WHERE Назва_тарифу = @TariffName",
conn);
                cmd.Parameters.AddWithValue("@TariffName",
tariffName);
                SqlDataReader reader = cmd.ExecuteReader();
                if (reader.Read())
                {
                    tariffCost =
Convert.ToDecimal(reader["Вартість_тарифу"]);
                }
                reader.Close();
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при пошуку
Вартість_тарифу: " + ex.Message);
                conn.Close();
            }
            return tariffCost;
        }
        private void button9_Click(object sender, EventArgs e)
        {
            try
            {
                conn.Open();
                SqlCommand cmd = conn.CreateCommand();
                cmd.CommandType = CommandType.Text;
                int searchID;
                if (int.TryParse(textBox8.Text, out searchID))
                {
                    cmd.CommandText = "SELECT * FROM Clients
WHERE ID_Договора = @ID OR Прізвище = @SearchText OR Імя =
@SearchText OR По_батькові = @SearchText OR Назва_тарифу =
@SearchText OR Номер_телефону = @SearchText OR Адреса =
@SearchText";
                    cmd.Parameters.AddWithValue("@ID",
searchID);
                }
                else
                {
                    cmd.CommandText = "SELECT * FROM Clients
WHERE Прізвище = @SearchText OR Імя = @SearchText OR По_батькові
```

```
= @SearchText OR Назва_тарифу = @SearchText OR Номер_телефону =
@SearchText OR Адреса = @SearchText";
                }
                cmd.Parameters.AddWithValue("@SearchText",
textBox8.Text);
                SqlDataAdapter da = new SqlDataAdapter(cmd);
                DataTable dt = new DataTable();
                da.Fill(dt);
                dataGridView1.DataSource = dt;
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при виконанні пошуку:
" + ex.Message);
                conn.Close();
            }
        }
        private void button10_Click(object sender, EventArgs e)
        {
            try
            {
                string tariffName = textBox5.Text;
                int tariffID = GetTariffIDByName(tariffName);
                decimal              tariffCost              =
GetTariffCostByName(tariffName);
                if (tariffID != -1 && tariffCost != -1)
                {
                    Form4 form4 = new Form4();
                    form4.textBox1.Text = this.textBox1.Text;
                    form4.textBox3.Text = this.textBox5.Text;
                    form4.SetTariffID(tariffID);
                    form4.SetTariffCost(tariffCost);
                    form4.ShowDialog();
                }
                else
                {
                    MessageBox.Show("Тариф не знайдено.");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка    при    відображенні
тарифу: " + ex.Message);
            }
        }
        private   void   comboBox1_SelectedIndexChanged(object
sender, EventArgs e)
        {
        }
        private void LoadTariffsIntoComboBox()
        {
```

```
        try
            {
                conn.Open();
                SqlCommand cmd = new SqlCommand("SELECT
Назва_тарифу FROM Tariffs", conn);
                SqlDataReader reader = cmd.ExecuteReader();
                while (reader.Read())
                {

comboBox1.Items.Add(reader["Назва_тарифу"].ToString());
                }
                reader.Close();
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при завантаженні
тарифів: " + ex.Message);
                conn.Close();
            }
        }
        private void textBox8_TextChanged(object sender,
EventArgs e)
        {
        }
        private void textBox1_TextChanged(object sender,
EventArgs e)
        {
        }
        private void textBox5_TextChanged(object sender,
EventArgs e)
        {
        }
        private void panel1_Paint(object sender, PaintEventArgs
e)
        {
        }
        private void button11_Click(object sender, EventArgs e)
        {
            conn.Open();
            SqlCommand cmd = conn.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "DELETE FROM ClientDetails WHERE
ID_Договора ='" + textBox1.Text + "'";
            cmd.ExecuteNonQuery();
            conn.Close();
            dispp_data();
            MessageBox.Show("Запис видалено успішно");
        }
        private void textBox9_TextChanged(object sender,
EventArgs e)
        {
```

```
}
        private void button12_Click(object sender, EventArgs e)
        {
            conn.Open();
            SqlCommand cmd = conn.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "INSERT INTO Tariffs VALUES ('" +
textBox9.Text + "','" + textBox10.Text + "','" + textBox11.Text
+ "', '" + textBox12.Text + "')";
            cmd.ExecuteNonQuery();
            conn.Close();
            textBox9.Text = "";
            textBox10.Text = "";
            textBox11.Text = "";
            textBox12.Text = "";
            disp_data();
            MessageBox.Show("Запис додано в таблицю");
        }
        private void button13_Click(object sender, EventArgs e)
        {
            conn.Open();
            SqlCommand cmd = conn.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText    =    "UPDATE    Tariffs    SET
Назва_тарифу='"   +   textBox9.Text   +   "',   Послуги='"   +
textBox10.Text + "', Вартість_тарифу='" + textBox11.Text + "',
Дата_введення_тарифу='" + textBox12.Text + "' WHERE ID_Тарифу
='" + textBox13.Text + "'";
            cmd.ExecuteNonQuery();
            conn.Close();
            disp_data();
            MessageBox.Show("Запис змінено успішно");
        }
        private void button14_Click(object sender, EventArgs e)
        {
            conn.Open();
            SqlCommand cmd = conn.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText  =  "DELETE  FROM  Tariffs  WHERE
ID_тарифу ='" + textBox13.Text + "'";
            cmd.ExecuteNonQuery();
            conn.Close();
            disp_data();
            MessageBox.Show("Запис видалено успішно");
        }
        private   void   dataGridView2_CellContentClick(object
sender, DataGridViewCellEventArgs e)
        {
        }
        private void button15_Click(object sender, EventArgs e)
        {
            disp_data();
```

```
  }
        private void button16_Click(object sender, EventArgs e)
        {
            try
            {
                conn.Open();
                SqlCommand cmd = conn.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText   =   "INSERT   INTO   Workers
(Прізвище,  Ім'я,  По_батькові,  Номер_телефону,  Пошта,  Вхід,
Пароль,  Дата_реєстрації,  Дата_звільнення,  Дозвіл_редагування)
VALUES (@Прізвище, @Ім'я, @По_батькові, @Номер_телефону, @Пошта,
@Вхід,     @Пароль,     @Дата_реєстрації,     @Дата_звільнення,
@Дозвіл_редагування)";
                cmd.Parameters.AddWithValue("@Прізвище",
textBox14.Text);
                cmd.Parameters.AddWithValue("@Ім'я",
textBox15.Text);
                cmd.Parameters.AddWithValue("@По_батькові",
textBox16.Text);
                cmd.Parameters.AddWithValue("@Номер_телефону",
textBox17.Text);
                cmd.Parameters.AddWithValue("@Пошта",
textBox18.Text);
                cmd.Parameters.AddWithValue("@Вхід",
comboBox2.Text);
                cmd.Parameters.AddWithValue("@Пароль",
textBox20.Text);

cmd.Parameters.AddWithValue("@Дата_реєстрації",
textBox21.Text);

cmd.Parameters.AddWithValue("@Дозвіл_редагування",
comboBox3.Text);
                if (string.IsNullOrEmpty(textBox22.Text))
                {

cmd.Parameters.AddWithValue("@Дата_звільнення", DBNull.Value);
                }
                else
                {

cmd.Parameters.AddWithValue("@Дата_звільнення",
textBox22.Text);
                }
                cmd.ExecuteNonQuery();
                conn.Close();
                textBox14.Text = "";
                textBox15.Text = "";
                textBox16.Text = "";
                textBox17.Text = "";
                textBox18.Text = "";
```

```
  comboBox2.Text = "";
                textBox20.Text = "";
                textBox21.Text = "";
                textBox22.Text = "";
                comboBox3.Text = "";
                displ_data();
                MessageBox.Show("Запис додано в таблицю");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при додаванні запису:
" + ex.Message);
                conn.Close();
            }
        }
        private void button17_Click(object sender, EventArgs e)
        {
            try
            {
                conn.Open();
                SqlCommand cmd = conn.CreateCommand();
                cmd.CommandType = CommandType.Text;
                cmd.CommandText    =    "UPDATE    Workers    SET
Прізвище=@Прізвище,     Імя=@Імя,     По_батькові=@По_батькові,
Номер_телефону=@Номер_телефону,    Пошта=@Пошта,    Вхід=@Вхід,
Пароль=@Пароль,               Дата_реєстрації=@Дата_реєстрації,
Дата_звільнення=@Дата_звільнення,
Дозвіл_редагування=@Дозвіл_редагування                    WHERE
ID_працівника=@ID_працівника";
                cmd.Parameters.AddWithValue("@Прізвище",
textBox14.Text);
                cmd.Parameters.AddWithValue("@Імя",
textBox15.Text);
                cmd.Parameters.AddWithValue("@По_батькові",
textBox16.Text);
                cmd.Parameters.AddWithValue("@Номер_телефону",
textBox17.Text);
                cmd.Parameters.AddWithValue("@Пошта",
textBox18.Text);
                cmd.Parameters.AddWithValue("@Вхід",
comboBox2.Text);
                cmd.Parameters.AddWithValue("@Пароль",
textBox20.Text);

cmd.Parameters.AddWithValue("@Дата_реєстрації",
textBox21.Text);

cmd.Parameters.AddWithValue("@Дозвіл_редагування",
comboBox3.Text);
                if (string.IsNullOrEmpty(textBox22.Text))
                {
```

```
cmd.Parameters.AddWithValue("@Дата_звільнення", DBNull.Value);
                }
                else
                {

cmd.Parameters.AddWithValue("@Дата_звільнення",
textBox22.Text);
                }
                cmd.Parameters.AddWithValue("@ID_працівника",
textBox19.Text);
                cmd.ExecuteNonQuery();
                conn.Close();
                displ_data();
                MessageBox.Show("Запис змінено успішно");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Помилка при зміненні запису: "
+ ex.Message);
                conn.Close();
            }
        }
        private void button18_Click(object sender, EventArgs e)
        {
            conn.Open();
            SqlCommand cmd = conn.CreateCommand();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText  =  "DELETE  FROM  Workers  WHERE
ID_працівника ='" + textBox19.Text + "'";
            cmd.ExecuteNonQuery();
            conn.Close();
            displ_data();
            MessageBox.Show("Запис видалено успішно");
        }
        private void button19_Click(object sender, EventArgs e)
        {
            string currentLogin = textBox23.Text;
            string currentPassword = textBox24.Text;
            string newLogin = textBox25.Text;
            string newPassword = textBox26.Text;
            try
            {
                conn.Open();
                SqlCommand cmd = new SqlCommand("SELECT COUNT(*)
FROM AdminCredentials WHERE Логін = @currentLogin AND Пароль =
@currentPassword", conn);
                cmd.Parameters.AddWithValue("@currentLogin",
currentLogin);

cmd.Parameters.AddWithValue("@currentPassword",
currentPassword);
                int count = (int)cmd.ExecuteScalar();
```

```
if (count > 0)
                {
                    SqlCommand updateCmd = new
SqlCommand("UPDATE AdminCredentials SET Логін = @newLogin,
Пароль = @newPassword WHERE Логін = @currentLogin AND Пароль =
@currentPassword", conn);

updateCmd.Parameters.AddWithValue("@newLogin", newLogin);

updateCmd.Parameters.AddWithValue("@newPassword", newPassword);

updateCmd.Parameters.AddWithValue("@currentLogin",
currentLogin);

updateCmd.Parameters.AddWithValue("@currentPassword",
currentPassword);
                    updateCmd.ExecuteNonQuery();
                    MessageBox.Show("Логін та пароль
адміністратора успішно змінено.");
                }
                else
                {
                    SqlCommand workerCmd = new
SqlCommand("SELECT COUNT(*) FROM Workers WHERE Пошта =
@currentLogin AND Пароль = @currentPassword", conn);

workerCmd.Parameters.AddWithValue("@currentLogin",
currentLogin);

workerCmd.Parameters.AddWithValue("@currentPassword",
currentPassword);
                    int workerCount =
(int)workerCmd.ExecuteScalar();
                    if (workerCount > 0)
                    {
                        SqlCommand updateWorkerCmd = new
SqlCommand("UPDATE Workers SET Пошта = @newLogin, Пароль =
@newPassword WHERE Пошта = @currentLogin AND Пароль =
@currentPassword", conn);

updateWorkerCmd.Parameters.AddWithValue("@newLogin", newLogin);

updateWorkerCmd.Parameters.AddWithValue("@newPassword",
newPassword);

updateWorkerCmd.Parameters.AddWithValue("@currentLogin",
currentLogin);

updateWorkerCmd.Parameters.AddWithValue("@currentPassword",
currentPassword);
                        updateWorkerCmd.ExecuteNonQuery();
```

```
                    MessageBox.Show("Логін та пароль працівника
успішно змінено.");
                        }
                        else
                        {
                            MessageBox.Show("Невірний поточний
логін або пароль.");
                        }
                    }
                    textBox23.Text = "";
                    textBox24.Text = "";
                    textBox25.Text = "";
                    textBox26.Text = "";
                    conn.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Помилка при зміні облікових
даних: " + ex.Message);
                    conn.Close();
                }
            }
            private void button20_Click(object sender, EventArgs e)
            {
                dispp_data();
            }
        }
}
```