Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента *Дубовенко Єгор Андрійович*
(ПІБ)

академічної групи *121М-22-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *«Інженерія програмного забезпечення»*
(назва освітньої програми)

на тему: *Розробка автоматизованої інформаційної системи для туристичної компанії з використанням інтелектуальної системи прогнозування*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи				
розділів:				
спеціальний	*Доц. Кабак Л. В.*			
Рецензент	*Доц. Каштан В. Ю.*			
Нормоконтролер	*Проф. Лактіонов І.С.*			

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)
_____ Алексєєв М.О.
(підпис) (прізвище, ініціали)
« » _____ 20 23 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121м-22-1* *Дубовенко Є.А.*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка автоматизованої інформаційної системи для туристичної компанії з використанням інтелектуальної системи прогнозування*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень є автоматизована інформаційна система туристичної компанії.

Предмет досліджень – математичні моделі та програмне забезпечення прогнозування часових рядів.

Мета роботи – розробка системи прогнозування та обробки даних туристичної фірми.

Методи дослідження – базуються на принципах інтелектуального аналізу даних, регресійного аналізу, теорії баз даних. Також був використаний метод найменших квадратів для мінімізації відхилення прогнозу цільової функції.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна – полягає у набутті подальшого розвитку методів інтелектуального аналізу даних які поєднанні у додатку для обробки даних туристичної фірми і додатку для аналізу і прогнозування попиту на послуги.

Практична цінність полягає в тому, що розроблений програмний продукт можна буде використовувати у повсякденній роботі компанії для оптимізації роботи компа-

нії та прийняття відповідних менеджерських рішень на підставі розробленого прогнозу.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безпосереднє використання запропонованих методів. В результаті роботи повинен бути розроблений клієнт-серверний додаток на основі фреймворку .NET Framework, що працює під управлінням операційної системи Windows.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі.	15.09.2023-29.09.2023
Дослідження прогнозування часових рядів та вивчення існуючих рішень	30.09.2023-20.10.2023
Розробка та дослідження ефективності впровадження інформаційної системи для прогнозування попиту на туристичні напрямки компанії	21.11.2023-08.12.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивний завдяки пришвидшенню та автоматизації процесу прогнозування попиту на послуги туристичної компанії.

Соціальний ефект від реалізації результатів роботи очікується позитивним, завдяки поліпшенню сервісу туристичних компаній та умов роботи співробітників таких компаній.

Завдання видав _____ *Кабак Л. В.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Дубовенко Є.А.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *15.09.2023 р.*

Термін подання дипломного проекту до ЕК *12.12.2023*

РЕФЕРАТ

Пояснювальна записка: 105 стр., 52 рис., 3 додатки, 21 джерело.

Об'єкт дослідження: процес розробки та впровадження інформаційної системи для прогнозування попиту на туристичні напрямки туристичної фірми.

Предмет досліджень: математичні моделі та програмне забезпечення прогнозування попиту на товари або послуги.

Мета магістерської роботи: прогнозування попиту на туристичні напрямки компанії на підставі даних раніше введених в базу даних.

Методи дослідження: алгоритми та методи прогнозування часових рядів.

Наукова новизна даної роботи знаходиться у поєднанні додатку для ведення даних підприємства і додатку для аналізу та прогнозування часових рядів.

Практична цінність полягає у розробці програмного продукту, який можна буде використовувати у повсякденній роботі туристичних компаній для оптимізації ведення даних компаній і прогнозування попиту на їхні послуги.

Область застосування. Розроблена інформаційна система може застосовуватися для вирішення повсякденних задач з прогнозування попиту на послуги підприємства, ведення даних про продажі підприємства, а також ведення даних про співробітників та клієнтів.

Значення роботи та висновки. Алгоритм прогнозування попиту на туристичні напрямки компанії дозволяє прогнозувати попит на послуги компанії, оптимізувати та покращувати сервіс компанії, робити менеджерські рішення на підставі прогнозу та спрощувати процес аналізу наявних даних в базі даних інформаційної системи що позитивно впливає на підприємницьку діяльність та знижує рівень бізнес ризиків.

Прогнози щодо розвитку досліджень. Покращити інформаційну систему додавши до розрахунку систему знижок.

Ключові слова: регресійний аналіз, попит, клієнт серверний додаток, Windows Forms, C#, прогнозування, ковзна середня.

ABSTRACT

Explanatory note: 105 pages, 52 figures, 3 appendices, 21 sources.

The object of the study: the process of developing and implementing an information system for forecasting the demand for tourist destinations of a travel firm.

Research subject: mathematical models and software for forecasting demand for goods or services.

The purpose of the master's work: forecasting the demand for the company's tourist destinations based on the data previously entered into the database.

Research methods: algorithms and methods of time series forecasting.

The scientific novelty of this work lies in the combination of an application for managing enterprise data and an application for analyzing and forecasting time series.

The practical value lies in the development of a software product that can be used in the day-to-day work of travel companies to optimize company data management and forecast demand for their services.

Field of application. The developed information system can be used to solve everyday tasks of forecasting the demand for the company's services, keeping data on the company's sales, as well as keeping data on employees and customers.

Value of work and conclusions. The demand forecasting algorithm for the company's tourist destinations allows forecasting the demand for the company's services, optimizing and improving the company's service, making managerial decisions based on the forecast, and simplifying the process of analyzing available data in the database of the information system, which positively affects business activity and reduces the level of business risks.

Forecasts regarding the development of research. Improve the information system by adding a system of discounts to the calculation.

Keywords: regression analysis, demand, client server application, Windows Forms, C#, forecasting, moving average.

ЗМІСТ

ЗМІСТ 6

ВСТУП 8

РОЗДІЛ 1 АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ 10

1.1. Вступ 10

1.2. Поняття прогнозування 12

1.3. Існуючі рішення для прогнозування 19

1.4. Дослідження вимог до програмної системи 22

1.5. Постановка задачі 23

РОЗДІЛ 2 ОПИС МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ 24

2.1. Побудова Use-case діаграми 24

2.2. Побудова ER-моделі 28

2.3. Ковзна середня 36

2.4. Регресійний аналіз 37

2.5. Висновки 43

РОЗДІЛ 3 СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 45

3.1. Вибір мови програмування та середовища розробки 45

3.2. Структура проекту 47

3.3. Опис роботи розробленого програмного забезпечення 49

ВИСНОВКИ 77

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79

ДОДАТОК А 82

ДОДАТОК Б 85

ДОДАТОК В 107

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ 107

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UML – Unified Modeling Language;

ОС – операційна система;

БД – база даних;

SQL – Structured Query Language.

ВСТУП

Актуальність теми: незважаючи на виклики з якими зіштовхується туристичний бізнес сьогодні туристичні компанії по всьому світу активно вдосконалюють свої стратегії для забезпечення стабільності та привабливості для потенційних клієнтів. Але для того щоб дійсно ефективно керувати будь-яким бізнесом потрібні відповідні інструменти які дозволили б автоматизувати певні процеси і оптимізувати їх. Але для управління туристичною компанією потрібно багато різноманітного програмного забезпечення яке не завжди поєднує в собі всі необхідні функції, основні функції які необхідні туристичній компанії - це ведення даних підприємства і прогнозування попиту на туристичні напрямки які пропонує компанія. Адже від ефективності і точності виконання цих двох функцій залежить весь прибуток підприємства. Тому завдання коректного прогнозування попиту на туристичні напрямки компанії і введення даних компанії є актуальним.

Об'єкт досліджень: автоматизована інформаційна система туристичної компанії.

Предмет досліджень: методи прогнозування та аналізу попиту на послуги які надають туристичні компанії.

Мета дослідження: дослідження автоматизації управління туристичною компанією.

Методи дослідження: для розв'язання поставлених задач використані архітектура клієнт-сервер, математичний метод дослідження часових рядів ковзна середня та регресійний аналіз даних.

Наукова новизна даної роботи: полягає у набутті подальшого розвитку методів інтелектуального аналізу даних які поєднанні у додатку для обробки даних туристичної фірми і додатку для аналізу і прогнозування попиту на послуги.

Практичне значення: розроблено інформаційну систему що базується на клієнт-серверній архітектурі та математичних методах прогнозування

часових рядів що надає можливість оптимізації процесів роботи туристичної компанії і прогнозування її прибутків.

Особистий внесок автора: полягає в розробці інформаційної системи для туристичної компанії на основі клієнт серверної архітектури з можливістю прогнозування попиту на туристичні напрямки на підставі раніше внесених даних.

РОЗДІЛ 1

АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Вступ

Прогнозування – це техніка, яка використовує історичні дані як вхідні дані для створення обґрунтованих оцінок, які є прогнозними для визначення напрямку майбутніх тенденцій.

Компанії використовують прогнозування, щоб визначити, як розподілити свої бюджети або спланувати очікувані витрати на майбутній період часу. Зазвичай це базується на прогнозованому попиті на пропоновані товари та послуги.

Інвестори використовують прогнози, щоб визначити, чи події, що впливають на компанію, такі як очікування продажів, підвищать або зменшать ціну акцій цієї компанії. Прогнозування також забезпечує важливий орієнтир для фірм, яким потрібна довгострокова перспектива діяльності.

Аналітики акцій використовують прогнози, щоб екстраполювати тенденції, такі як валовий внутрішній продукт (ВВП) або безробіття , змінюватися в наступному кварталі чи році. Нарешті, статистики можуть використовувати прогнозування для аналізу потенційного впливу змін у бізнес-операціях. Наприклад, можуть збиратися дані щодо впливу зміни робочого часу на задоволеність клієнтів або продуктивності працівників у разі зміни певних умов праці. Потім ці аналітики пропонують оцінки прибутків, які часто об’єднуються в консенсусну цифру. Якщо фактичні оголошення про прибутки не відповідають оцінкам, це може мати великий вплив на курс акцій компанії.

Прогнозування стосується проблеми або набору даних. Економісти роблять припущення щодо ситуації, що аналізується, які повинні бути встановлені до того, як будуть визначені змінні прогнозу. На основі визначених елементів вибирається відповідний набір даних і використовується для

маніпулювання інформацією. Дані аналізуються, і визначається прогноз. Нарешті, період перевірки відбувається, коли прогноз порівнюється з фактичними результатами, щоб створити більш точну модель для прогнозування в майбутньому.

Правильний метод прогнозування залежатиме від типу та обсягу прогнозу. Якісні методи потребують більше часу та витрат, але можуть зробити дуже точні прогнози за обмеженого обсягу. Наприклад, вони можуть бути використані для прогнозування того, наскільки добре буде сприйнято громадськістю запуск нового продукту компанії.

Для швидшого аналізу, який може охоплювати більший обсяг, кількісні методи часто є більш корисними. Дивлячись на великі масиви даних, пакети статистичного програмного забезпечення сьогодні можуть обчислити цифри за лічені хвилини чи секунди. Однак чим більший набір даних і складніший аналіз, тим дорожчим він може бути.

Таким чином, прогнозисти часто проводять свого роду аналіз витрат і вигод, щоб визначити, який метод максимізує шанси на точний прогноз найбільш ефективним способом. Крім того, поєднання методів може бути синергічним і підвищити надійність прогнозу.

Найбільшим обмеженням прогнозування є те, що воно стосується майбутнього, яке сьогодні є принципово непізнаваним. У результаті прогнози можуть бути лише найкращими припущеннями. Хоча існує кілька методів підвищення надійності прогнозів, припущення, які використовуються в моделях, або дані, що вводяться в них, мають бути правильними. Інакше результатом буде сміття всередину, сміття назовні. Навіть якщо дані хороші, прогнозування часто спирається на історичні дані, які не гарантовано будуть дійсними в майбутньому, оскільки все може змінюватися і змінюється з часом. Також неможливо правильно врахувати незвичайні або одноразові події, такі як криза чи катастрофа.

1.2. Поняття прогнозування

Прогнозування – це передбачення, яке роблять на підставі вивчення статистичних даних а також враховуються минулі закономірності. Зазвичай компанії програмне забезпечення та системи для того щоб проаналізувати великі обсягі даних, які були зібрані за тривалий період часу. Після цього за допомогою програмних систем прогнозується майбутній попит та тенденції, це допомагає компаніям прийняти більш точні рішення.

Прогнозування відіграє роль інструменту планування, що дозволяє компаніям підготуватися до невизначеності, що може виникнути у майбутньому. Таким чином, менеджери можуть впевнено реагувати на зміни, контролювати бізнес-операції та ухвалювати стратегічні рішення, які сприяють майбутньому зростанню. Приклади цілей прогнозування:

– більше ефективне використання ресурсів;

– візуалізація бізнес-продуктивності;

– встановити час запуску нових товарів чи послуг;

– оцінювання поточних витрат;

– прогнозування майбутніх подій, наприклад обсягів продажу та заробітку;

– перегляд рішень менеджменту.

Методи прогнозування можуть бути якісними та кількісними.

Якісне прогнозування ґрунтується на короткострокових прогнозах фахівців з маркетингу. Якщо статистичних даних недостатньо, можна використати якісні методи. Два стандартні приклади використання наведені нижче:

– методи дослідження ринку, такі як анкетування та опитування, визначають споживчий попит;

– дельфійський метод передбачає опитування експертів у певній галузі, щоб зібрати їхні думки та передбачити тенденції у цій галузі.

Моделі кількісного прогнозування використовують змістовну статистику та статистичні дані для прогнозування довгострокових майбутніх тенденцій. Стандартні приклади використання кількісних методів:

– економетричне моделювання дозволяє аналізувати набори фінансових даних, таких як дані про кредити та інвестиції, для прогнозування значних економічних зрушень та їхнього впливу на компанію;

– підхід індикатора дозволяє порівнювати точки даних, щоб визначити відносини між, начебто, незв'язаними даними. Наприклад, зміни ВВП можна використовуватиме прогнозування рівня безробіття;

– у цьому сценарії дані про ВВП називаються випереджаючим індикатором, а рівень безробіття – індикатором, що запізнюється;

– прогнозування часових рядів дозволяє аналізувати дані, зібрані різні проміжки часу, для прогнозування майбутніх тенденцій.

Данні часового ряду:

Перехресні дані охоплюють окремі юридичні та фізичні особи за той самий період часу. З іншого боку, дані часових рядів – це будь-який набір даних, що охоплює інформацію за проміжки часу. Ці дані відрізняються одна від одної, тому що вони впорядковують точки даних за часом. У результаті є можливість кореляції між спостереженнями у сусідніх інтервалах.

Дані часового ряду можна нанести на графік з інкрементальними інтервалами (або часовими шкалами) на осі x і значеннями вибіркових даних на осі y. Такі графіки часових рядів є цінними інструментами візуалізації даних. Фахівець із роботи з даними використовують їх визначення характеристик даних прогнозування. Декілька прикладів характеристик даних часових рядів:

– дані про тенденції у часі. У даних про тенденції значення у збільшуються чи зменшуються з часом, що робить графік лінійним. Наприклад, дані про населення можуть збільшуватись або зменшуватись лінійно з часом;

– сезонність. Сезонні шаблони виникають, коли дані часових рядів показують регулярні та передбачувані шаблони з часовими інтервалами менше року. Цей шаблон даних може виявлятися у вигляді різких підйомів чи інших

аномалій на лінійному графіці. Наприклад, роздрібні продажі магазину можуть збільшитися у святкові дні у грудні та квітні;

– структурні розриви. Іноді модель поведінки даних часових рядів раптово змінюється у певний час. Графік часового ряду може раптово зміщуватися вгору чи вниз, створюючи структурний розрив чи нелінійність. Наприклад, багато економічних показників різко змінилися у 2008 р. після початку світової фінансової кризи.

Прогнозування за допомогою часових рядів:

Прогнозування за допомогою часових рядів – це наукова методика роботи з даними на основі машинного навчання та інших комп'ютерних технологій для вивчення спостережень у минулому та прогнозування майбутніх значень даних часових рядів. Приклади прогнозування за допомогою часових рядів:

– астрономічні дані складаються з відомостей про рух планет, що повторюються, протягом століть. Такі дані можна використовувати для прогнозування астрономічних подій, таких як затемнення та комети;

– прогноз погоди ґрунтується на моделі вітру та температури для прогнозування змін погоди;

– вчені можуть використовувати дані про народжуваність та міграцію для прогнозування зростання населення.

Аналіз часових рядів спрямований на дослідження основних причин у будь-яких даних часових рядів. Ця сфера дослідження спрямована на те, щоб зрозуміти причину, яка стоїть за набором даних часових рядів. Нерідко аналітикам доводиться робити припущення та декомпозувати чи розбивати дані, щоб отримати значні статистичні дані та інші характеристики.

Аналіз часових рядів – це розуміння набору даних, тоді як прогнозування – це його прогноз. Три кроки прогностичного моделювання:

– поставте запитання та зберіть приблизний набір даних часового ряду, який відповідає на це питання за минулий період часу;

– навчіть комп'ютерне програмне забезпечення або алгоритм прогнозування, використовуючи попередні значення;

– використовуйте алгоритм прогнозування майбутніх спостережень.

Як працює прогнозування за допомогою часових рядів:

Фахівці роботи з даними використовують моделі прогнозування часових рядів, щоб робити більш точні прогнози. Спочатку вони проводять дослідницький аналіз даних, щоб вибрати найкращі алгоритми прогнозування, а потім використовують моделі машинного навчання для прогнозування. Деякі моделі прогнозування, що широко використовуються, наведені нижче.

Моделі декомпозиції розкладають або розбивають дані часових рядів на три компоненти:

– спрямований компонент;

– сезонний компонент;

– компонент шуму, що не належить до жодної з двох вищезгаданих груп.

Інший метод аналізу даних часових рядів полягає в тому, щоб розбити їх на два компоненти: передбачувані та непередбачувані компоненти даних.

Моделі на основі згладжування. Згладжування даних – це статистичний метод, який включає видалення викидів або точок даних, які значно відрізняються від решти набору даних. Ці моделі прогнозування роблять основну категорію шаблонів наочнішою, усуваючи випадкові варіації даних.

Моделі з урахуванням регресії. Авторегресія – це модель прогнозування, яка використовує спостереження за попередніми часовими кроками для визначення математичного взаємозв'язку між двома точками даних. Потім застосовується математичне співвідношення з метою оцінки невідомої майбутньої вартості. Залежно від використовуваної регресійної моделі математичне рівняння враховує попередні помилки прогнозу та попередні сезонні значення, покращуючи прогноз із плином часу.

Часові ряди:

Часовий ряд – це послідовність точок даних, які виникають у послідовному порядку протягом певного періоду часу. Це можна порівняти з перехресними даними , які фіксують момент часу.

Наприклад під час інвестування часовий ряд відстежує рух вибраних точок даних, таких як ціна цінного паперу, протягом визначеного періоду часу з точками даних, що записуються через регулярні проміжки часу. Немає мінімального чи максимального часу, який необхідно включити, щоб дані могли бути зібрані таким чином, щоб надати інформацію, яку шукає інвестор або аналітик, який вивчає часовий ряд.[2]

Часовий ряд можна взяти для будь-якої змінної, яка змінюється з часом. В інвестуванні зазвичай використовують часові ряди для відстеження ціни цінних паперів у часі. Це можна відстежувати в короткостроковій перспективі, як-от ціна цінного паперу за годину протягом робочого дня, або в довгостроковій перспективі, наприклад ціна цінного паперу на момент закриття в останній день кожного місяця протягом курс п'ять років.

Аналіз часових рядів може бути корисним, щоб побачити, як певний актив, цінний папір або економічна змінна змінюється з часом. Його також можна використовувати, щоб перевірити, як зміни, пов’язані з вибраною точкою даних, порівнюються зі змінами в інших змінних за той самий період часу.

Часові ряди також використовуються в кількох нефінансових контекстах, таких як вимірювання зміни чисельності населення з часом. На малюнку нижче зображено такий часовий ряд для зростання населення США протягом століття з 1900 по 2000 рік.

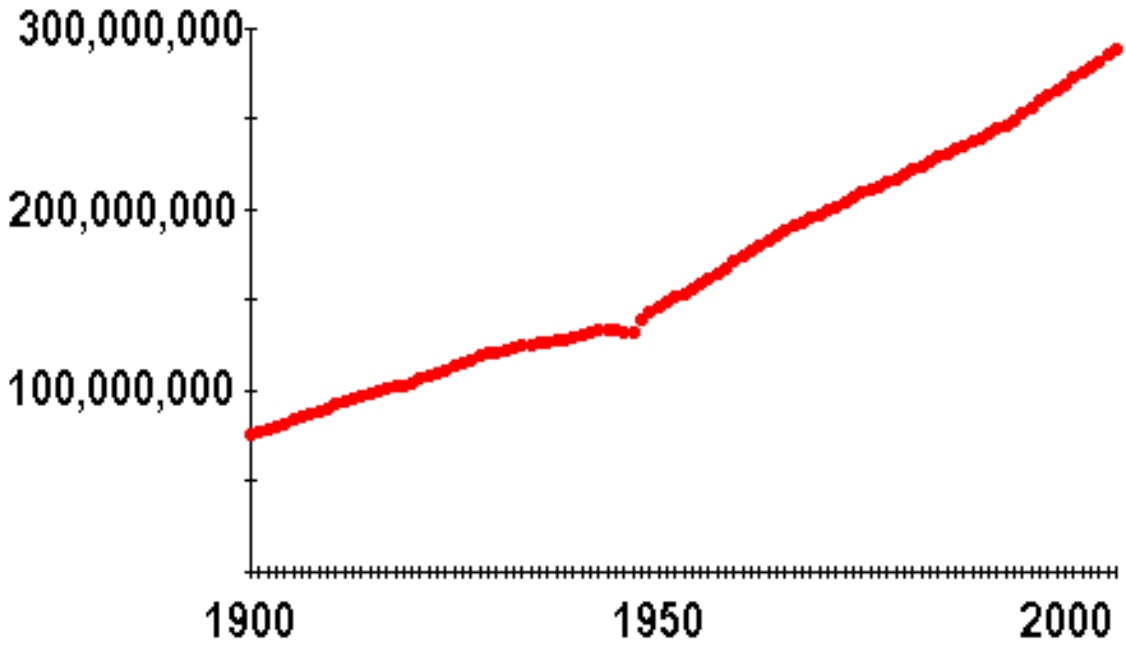


Рис. 1.1. Часовий ряд зростання населення США

Часовий ряд можна побудувати за допомогою будь-яких даних, які вимірюються протягом часу через рівномірні інтервали. Історичні курси акцій, прибутки, валовий внутрішній продукт (ВВП) або інші послідовності фінансових чи економічних даних можна аналізувати як часові ряди.[3]

Статистичні методи можна використовувати для аналізу даних часових рядів двома ключовими способами: для створення висновків про те, як одна чи більше змінних впливають на певну змінну, що цікавить, з часом, або для прогнозування майбутніх тенденцій. На відміну від перехресних даних, які, по суті, є одним зрізом часового ряду, стріла часу дозволяє аналітику робити більш правдоподібні причинно-наслідкові твердження.

Прогнозування часових рядів використовує інформацію про історичні значення та пов’язані закономірності для прогнозування майбутньої діяльності. Найчастіше це стосується аналізу трендів, аналізу циклічних коливань і питань сезонності. Як і з усіма методами прогнозування, успіх не гарантований.

Модель Бокса-Дженкінса, наприклад, є технікою, розробленою для прогнозування діапазонів даних на основі вхідних даних із визначеного часового ряду. Він прогнозує дані за трьома принципами: авторегресії , диференціації та ковзних середніх . Ці три принципи відомі як p, d і q

відповідно. Кожен принцип використовується в аналізі Бокса-Дженкінса, і разом вони представлені як авторегресійне інтегроване ковзне середнє або ARIMA (p, d, q). ARIMA можна використовувати, наприклад, для прогнозування цін на акції або зростання прибутків.

Інший метод, відомий як аналіз зміненого діапазону, можна використовувати для виявлення та оцінки ступеня стійкості, випадковості або повернення середнього значення в даних часових рядів. Перемасштабований діапазон можна використовувати для екстраполяції майбутнього значення або середнього значення для даних, щоб побачити, чи тенденція є стабільною чи ймовірно зміниться.

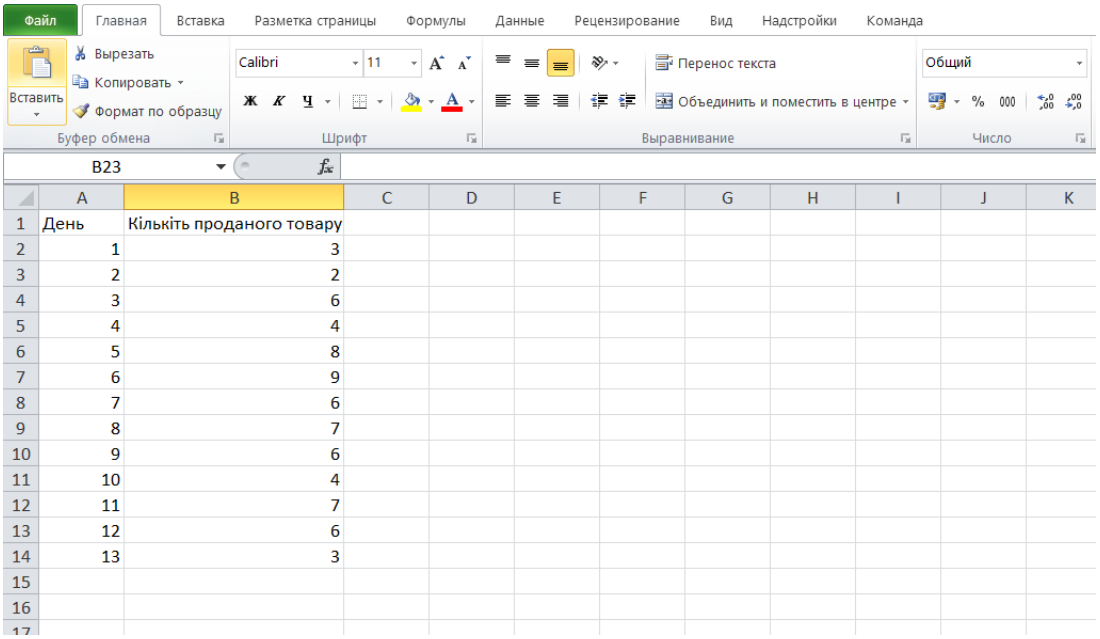
Перехресний аналіз:

Перехресний аналіз є одним із двох основних методів порівняння для аналізу запасів. Перехресний аналіз розглядає дані, зібрані в один момент часу, а не протягом певного періоду часу. Аналіз починається з встановлення цілей дослідження та визначення змінних, які аналітик хоче виміряти. Наступним кроком є визначення перерізу, наприклад, групи однолітків або галузі, і встановлення конкретного моменту часу, який оцінюється. Останній крок полягає в проведенні аналізу на основі перерізу та змінних і приході до висновку щодо ефективності компанії чи організації. По суті, перехресний аналіз показує інвестору, яка компанія найкраще підійде з урахуванням показників, які їх цікавлять.

Інтелектуальний аналіз даних – це процес, за допомогою якого сукупності не оброблених даних перетворюють на корисну інформацію. Великі компанії дізнаються більше про своїх клієнтів використовуючи програмне забезпечення для пошуку закономірностей у великих об’ємах даних, підприємства використовують цей процес для того щоб розробити ефективніші маркетингові стратегії, збільшити продажі та зменшити витрати. Часові ряди, такі як історичні записи корпоративних документів або фінансові звіти, особливо корисні тут для визначення тенденцій і закономірностей, які можна спрогнозувати в майбутньому.[4]

1.3. Існуючі рішення для прогнозування

На даний момент існує програмне забезпечення Excel, що дає можливість за допомогою регресійного аналізу створювати моделі прогнозу попиту на певні товари або послуги.



	A	B
1	День	Кількіть проданого товару
2	1	3
3	2	2
4	3	6
5	4	4
6	5	8
7	6	9
8	7	6
9	8	7
10	9	6
11	10	4
12	11	7
13	12	6
14	13	3

Рис. 1.1. Головне вікно Excel

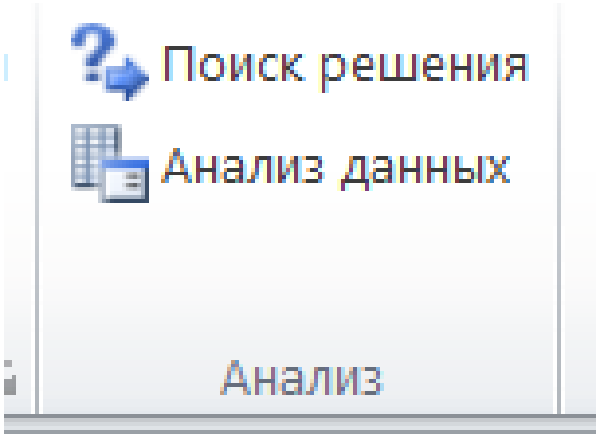


Рис. 1.2. Панель данні

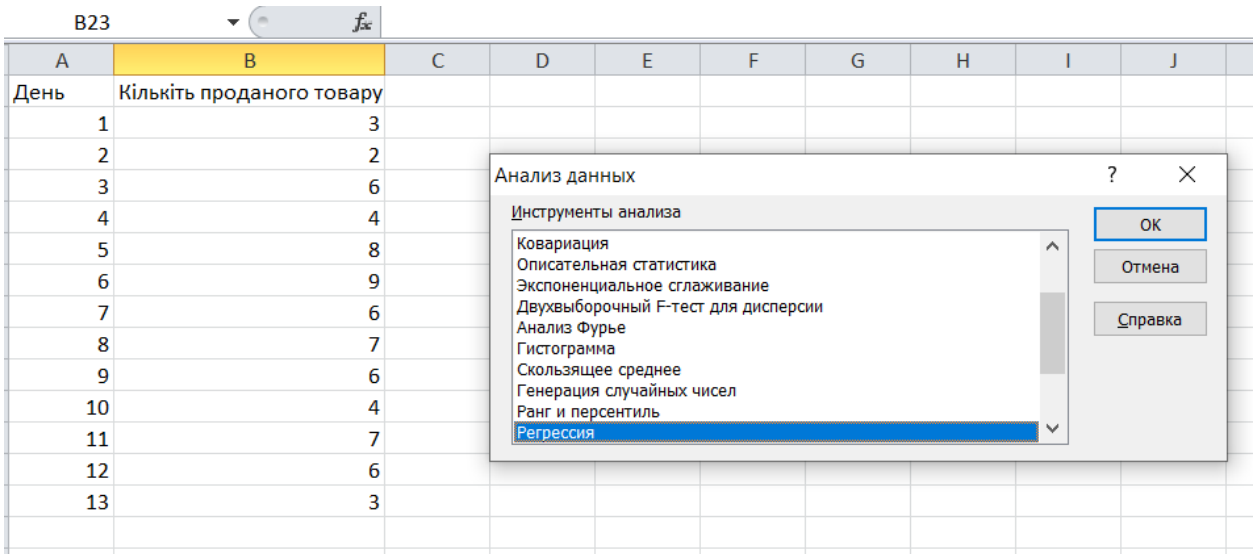


Рис. 1.3. Вікно аналізу даних

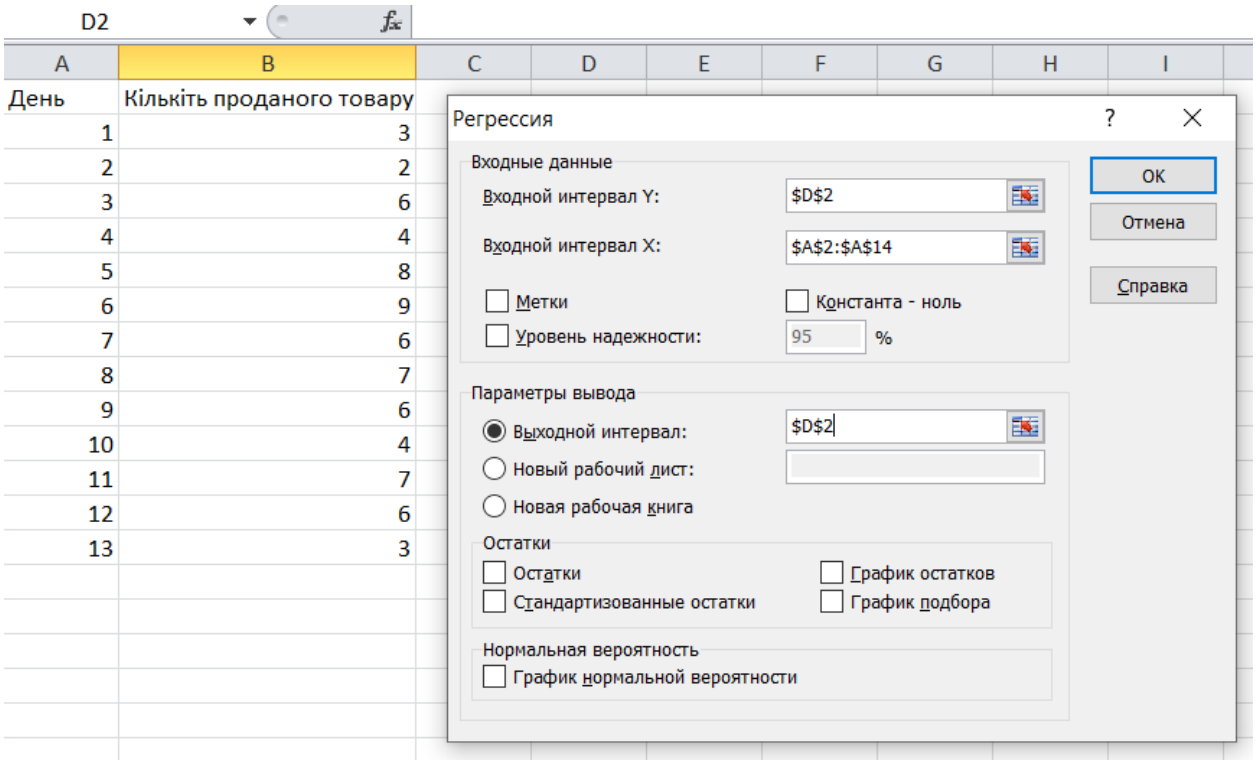


Рис. 1.4. Вікно регресії

M1

День	Кількіть проданого товару
1	3
2	2
3	6
4	4
5	8
6	9
7	6
8	7
9	6
10	4
11	7
12	6
13	3

ВЫВОД ИТОГОВ

Регрессионная статистика	
Множественный R	0,182875464
R-квадрат	0,033443435
Нормированный R-квадра	-0,054425343
Стандартная ошибка	2,16271188
Наблюдения	13

Дисперсионный анализ

	df	*SS*	*MS*	*F*	*Значимость F*
Регрессия	1	1,78021978	1,78021978	0,38060658	0,549834786
Остаток	11	51,45054945	4,677322677		
Итого	12	53,23076923			

	Коэффициенты	*Стандартная ошибка*	*t-статистика*	*P-Значение*	*Нижние 95%*	*Верхние 95%*	*Нижние 95,0%*	*Верхние 95,0%*
Y-пересечение	4,769230769	1,272428086	3,74813384	0,00322034	1,968635434	7,569826104	1,968635434	7,569826104
Переменная X 1	0,098901099	0,16031087	0,616933204	0,54983479	-0,253940748	0,451742946	-0,253940748	0,451742946

Рис. 1.5. Результат розрахунків

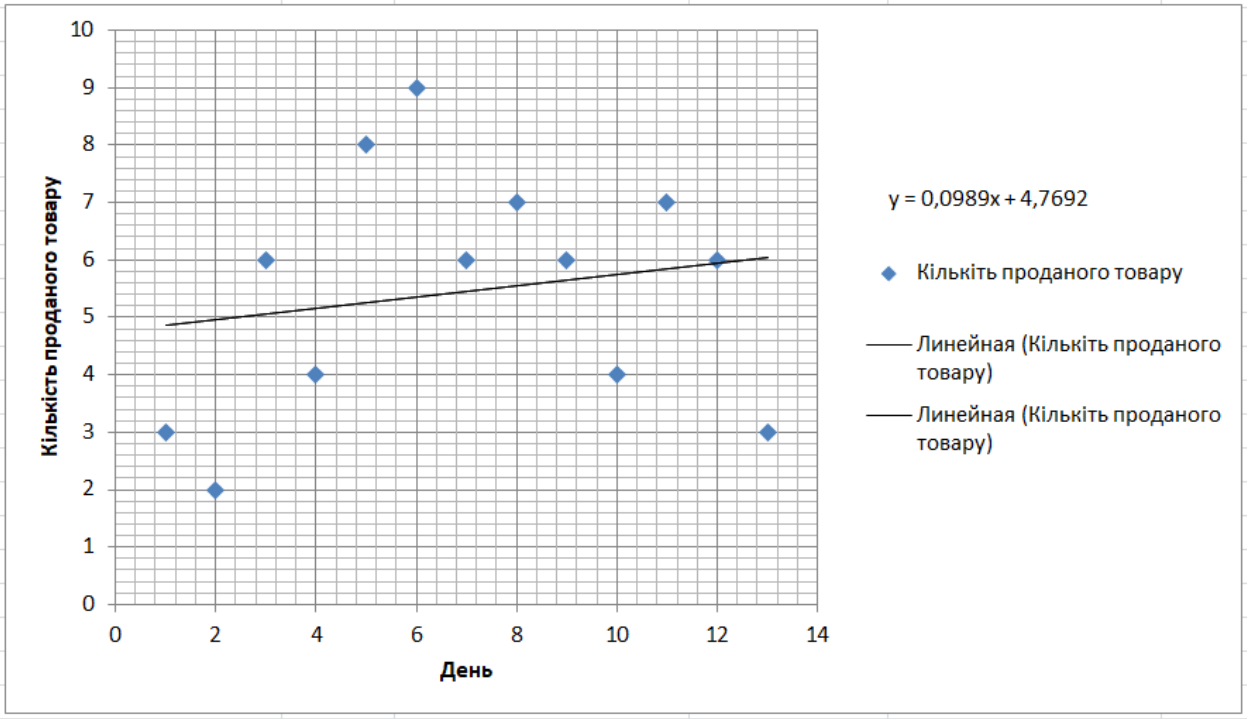


Рис. 1.6. Результат розрахунків у вигляді лінійної діаграми

Результати такого аналізу доволі точні але додаток Excel обирає данні для аналізу не з бази даних наприклад туристичної компанії а з таблиці, данні до якої потрібно вводити руками, а отже такий спосіб підходить для невеликого об’єму даних які не мають складної реляційної структури.

1.4. Дослідження вимог до програмної системи

Програмний додаток для прогнозування попиту на туристичні напрямки має за введеними раніше даними користувачів виконувати зберігання цих даних, аналіз та прогнозування.

Таким чином його функціонал має задовольняти наступні вимоги:

1. Авторизація користувачів та наділення їх певними правами доступу до системи.

2. Редагування облікових даних користувачів.

3. Ведення даних всіх співробітників компанії.

4. Ведення даних про туристичні напрямки компанії.

5. Ведення даних про готелі.

6. Ведення даних про авіакомпанії.

7. Ведення даних про авіарейси.

8. Ведення даних про укладені договори.

9. Ведення даних про клієнтів компанії.

10. Ведення даних про авіаквитки.

11. Відображення графіку попиту на обрані туристичні напрямки.

12. Аналіз та прогнозування значень попиту на обрані туристичні напрямки.

Програмна система має мати дві ролі користувачів:

– співробітника, укладає договори, замовляє авіаквитки та готелі;

– адміністратора, що контролює співробітників які працюють у програмній системі.

Програмний додаток має бути представлений у вигляді клієнт-серверного додатку та мати зрозумілий, зручний інтерфейс. Виконання обчислень має виконуватися на боці клієнту з введенням та виведенням результатів.

1.5. Постановка задачі

Отже в результаті розгляду різних методів проектування інформаційних систем і проблем які виникають при прогнозуванні часових рядів було обрано архітектуру клієнт-сервер і регресійний аналіз в якості методу прогнозування часових рядів. У даній кваліфікаційній роботі необхідно розв'язати наступні задачі:

1. Розглянути клієнт серверну архітектуру;
2. Розглянути регресійний аналіз та метод ковзної середньої;
3. Створити програмне забезпечення для реалізації обраних методів;

Забезпечити візуалізацію отриманих інформаційною системою даних за допомогою графіків.

РОЗДІЛ 2

ОПИС МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

2.1. Побудова Use-case діаграми

В процесі проектування інформаційної системи була використана одна з діаграм мови UML, а саме Use Case діаграма призначена для проектування користувацького сценарію.

UML розшифровується як Unified Modeling Language. Це загальна мова моделювання розвитку, яка використовується для аналізу, проектування та реалізації програмних систем. Метою UML є надання простого та поширеного методу візуалізації притаманних архітектурних властивостей програмної системи. З часом UML став стандартом створення об’єктно-орієнтованого програмного забезпечення де-факто. Проекти UML використовуються бізнес-користувачами, розробниками та всіма, хто потребує моделювання даних. UML не є методом розробки чи мовою програмування.

В мові UML використовується багато видів діаграм, але однією з найпоширеніших із них є Use-case діаграма.[5]

Діаграма варіантів використання UML (Unified Modeling Language) – це візуальне представлення взаємодії між акторами (користувачами або зовнішніми системами) і системою, що розглядається. Віна описує функціональність або поведінку системи з точки зору користувача. Діаграми варіантів використання відображають функціональні вимоги системи та допомагають визначити, як різні учасники взаємодіють із системою для досягнення конкретних цілей або завдань.

Use-case часто використовується на етапі проектування систем.

Отже в розроблюваній інформаційній системі можливо виділити наступні групи користувачів системи:

– адміністратор;

– співробітник.

Кожна група користувачів може використовувати інформаційну систему по різному.

Адміністратор має можливість:

– перегляду прогнозу попиту на туристичні напрямки;

– перегляду і зберігання звітів;

– ведення облікові записи співробітників;

– ведення облікові записи адміністраторів;

– ведення інформації про наявні тури;

– ведення інформації про готелі;

– ведення інформації про авіа компанії;

– ведення інформації про рейси.

Співробітник має можливість:

– перегляду прогнозу попиту на туристичні напрямки;

– перегляду і зберігання звітів;

– укладання договорів;

– замовлення квитків;

– бронювання готелів.

Тепер, коли виділили виділені групи користувачів та функціональність системи, почнемо будувати діаграму, щоб зафіксувати та структурувати отримані дані.

Кожна група користувачів на діаграмі варіантів використання позначається чоловічком, під яким записується ім'я групи людей, яку він позначає. Ім'я групи записується в однині. Символ чоловічка вже означає групу користувачів, тому не має потреби додатково відображати це в імені.

Рис. 2.1. Групи користувачів

У термінології UML, цей чоловічок називається актором. У загальному випадку, актор позначає будь-які сутності, які використовують систему. Цими сутностями можуть бути люди, технічні пристрої або інші системи.[6]

Кожна група користувачів використовує певні функції системи. На діаграмі варіантів використання функція системи зображується еліпсом, у якому записується ім'я функції у вигляді дієслова з пояснювальними словами. Як зображено на рисунку 2.2.

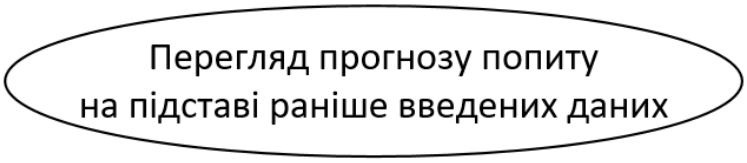


Рис. 2.2. Функція перегляду прогнозу попиту

У термінології UML цей еліпс називається варіантом використання. У загальному випадку, варіант використання – набір дій, який можна використовувати актором для взаємодії з системою.

Діаграмми UML використовують різні сполучні лінії для зв’язування елементів, такі лінії називаються відношеннями. Використовують такі відношення для досягнення певної мети, кожне зних має власну назву. В use-case діаграмах використовуються відношення які зображені на рисунках 2.3 – 2.6

Рис. 2.3. Відношення асоціації

Відношення асоціації призначене лише для з'єднання акторів та варіантів використання. Немає жодного сенсу поєднувати відношенням асоціації двох акторів або два варіанти використання.

Якщо на діаграмі варіантів використання актор з'єднаний з варіантом використання за допомогою відношення асоціації, це означає, що актор може виконувати дії, описані варіантом використання.

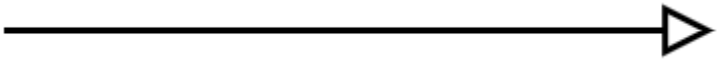


Рис. 2.4. Відношення узагальнення

Відношення узагальнення означає, що деякий актор (варіант використання) може бути узагальнений до іншого актора (варіанта використання). Стрілка спрямована від окремого випадку (спеціалізації) до загального випадку.

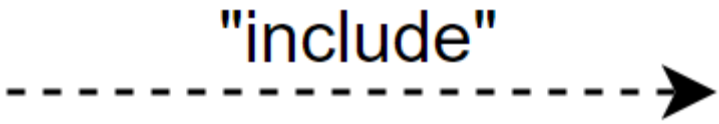


Рис. 2.5. Відношення включення

У загальному випадку, відношення включення використовується, щоб показати, що деякий варіант використання включає інший варіант використання в якості складової частини.

Коли ми використовуємо відношення включення, ми маємо на увазі, що складові варіанти використання обов’язково входять до загального варіанта використання.

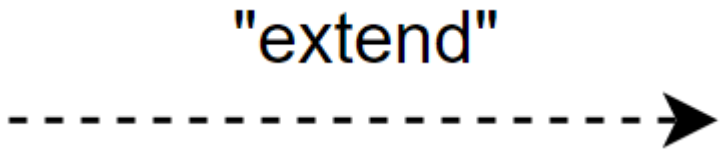


Рис. 2.6. Відношення розширення

Можна сказати, що відношення розширення – це вибіркове ставлення включення. Якщо відношення включення означає, що елемент обов'язково включається до складу іншого елемента, то у разі відношення розширення це включення необов'язково.[7]

Результат проектування use-case діаграми відображений на рисунку 2.7.

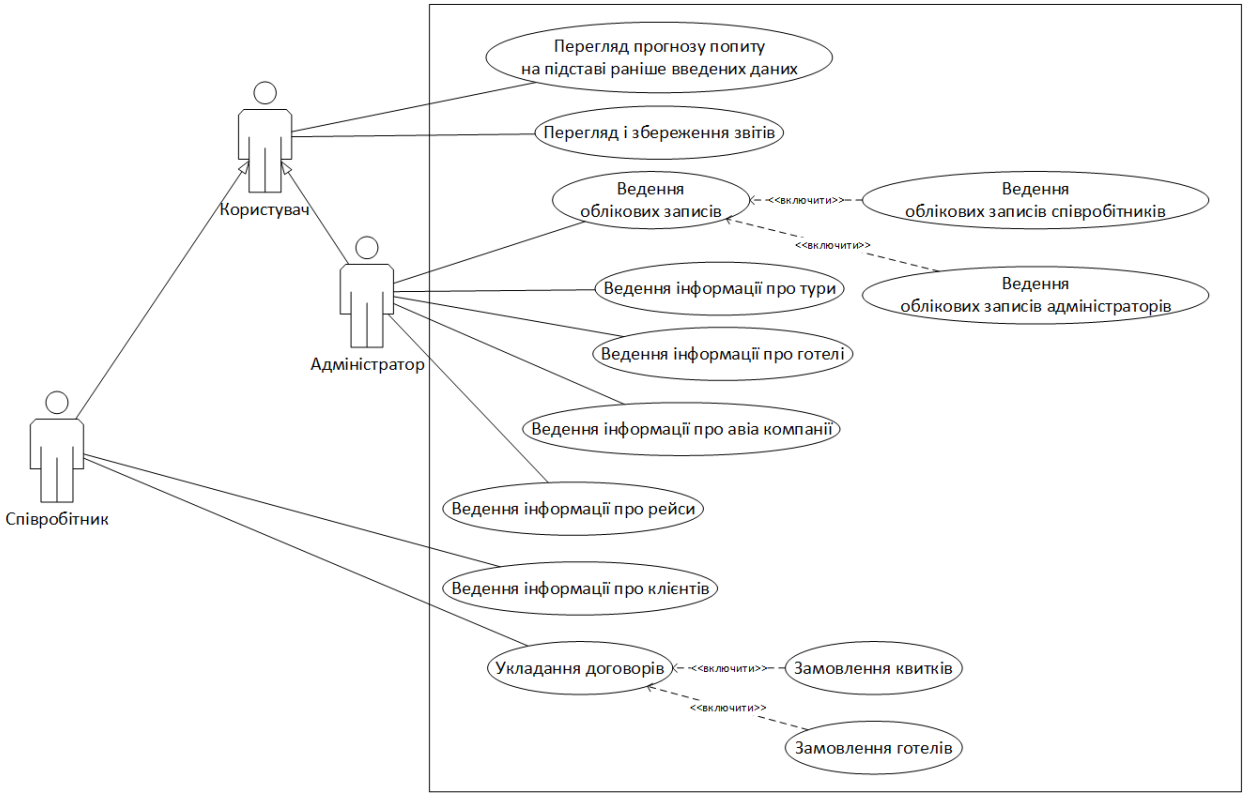


Рис. 2.7. Use-case діаграма інформаційної системи

2.2. Побудова ER-моделі

У світі проектування баз даних діаграми зв’язків сутностей (ER) можуть швидко перетворитися на складні мережі взаємопов’язаних сутностей,

атрибутів і зв’язків. Розуміння їхньої складності забезпечує ефективне керування базами даних і спрощені процеси моделювання даних.

ER-діаграма – це візуальне представлення моделі даних, яка описує, як різні сутності пов’язані одна з одною в базі даних. Ці діаграми служать інструментами для фахівців з баз даних, аналітиків і архітекторів, що дозволяє їм зрозуміти структуру бази даних.

Ключовими характеристиками діаграм ER є сутності, які є об’єктами або поняттями, і зв’язки, які визначають, як ці сутності взаємодіють. Атрибути, властивості сутностей, надають детальну інформацію, підвищуючи деталізацію моделі.

ER-діаграми служать кресленнями для проектування бази даних, що дозволяє професіоналам візуалізувати модель даних і зрозуміти складність реальних сценаріїв. Вони сприяють ефективній комунікації між зацікавленими сторонами та розробниками баз даних, гарантуючи, що всі на одній сторінці щодо структури бази даних.[8]

В нашому випадку на ER-діаграмі основними сховищами даних будуть сутності адміністратор, співробітник, тур, готель, авіа компанія, рейс, квиток, клієнт, договір. Вони зберігатимуть всі необхідні записи для отримання різноманітної інформації щодо туристичної компанії.

Далі можемо розглянути зв’язки сутностей. Зв’язок представляє собою асоціацію між двома сутностями, яка має значення для розгляду даної предметної області.

В данному випадку доцільним буде визначення типу зв’зку як один-багатьох. Це визначає як один до багатьох зв’язок екземпляру першої сутності із кількома екземплярами другої сутності. Такий тип зв’язку використовується найчастіше. Але для відношень договір, готель ми повинні провести зв’язок багато до багатьох оскільки в одному договорі може бути згадані декілька готелів, так само один готель може бути пов’язаний з декількома договорами. Для цього створимо проміжне відношення договір_готель, як зображено на рисунку 2.8.

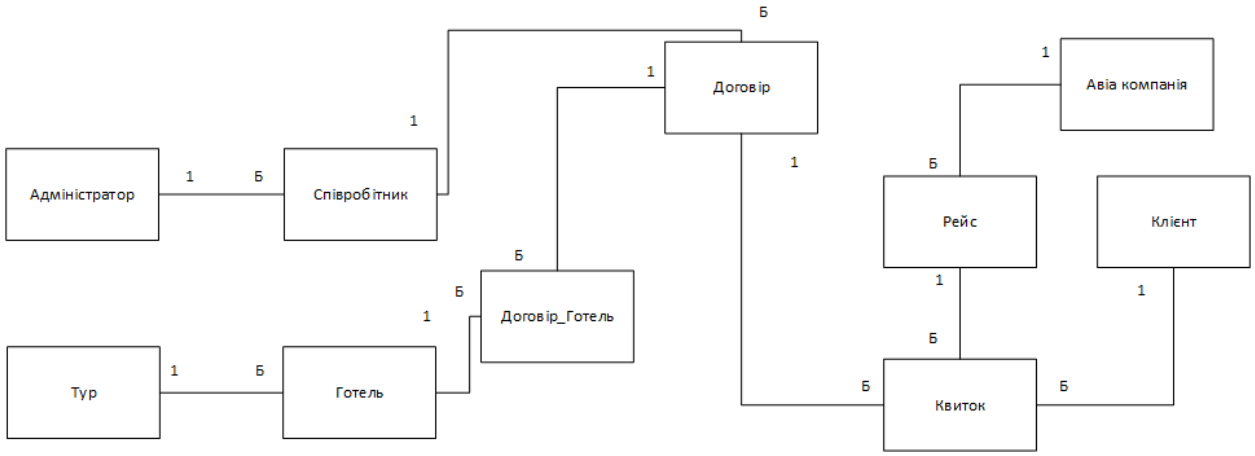


Рис. 2.8. Початкова ER-діаграма

Тпер можемо розглянути докладніше всі вище наведені сутності. По перше нам потрібно визначитися з атрибутами які будуть в них розташовані. Будь-яка характеристика сутності, яку розглядають в даній предметній області якщо вона призначена для кваліфікації, ідентифікації, класифікації, кількісної характеристики або для того щоб виражати стан сутності є атрибутом. Атрибут належить до типу характеристик або властивостей, які асоціюються з безліччю реальних або абстрактних об'єктів.

Розглянемо атрибути сутності «Адміністратор»

1) ID – унікальний ідентифікатор адміністратора;
2) ПІБ – прізвище, ім’я, по-батькові;
3) Логін – логін від облікового запису;
4) Пароль – пароль від облікового запису;
5) Телефон – номер телефону;
6) Серія, номер паспорту – паспортні данні;
7) Адреса – адреса проживання.

Розглянемо атрибути сутності «Співробітник»

1) ID – унікальний ідентифікатор співробітника;
2) ПІБ – прізвище, ім’я, по-батькові;
3) Логін – логін від облікового запису;
4) Пароль – пароль від облікового запису;

5) Телефон – номер телефону;

6) Серія, номер паспорту – паспортні данні;

7) Адреса – адреса проживання.

8) Адміністратор – унікальний порядковий номер адміністратора який зареєстрував співробітника.

Розглянемо атрибути сутності «Клієнт»

1) ID – унікальний порядковий номер клієнта;

2) ПІБ – прізвище, ім’я, по-батькові;

3) Телефон – номер телефону;

4) Серія, номер паспорту – паспортні данні;

5) ІНН – індивідуальний податковий номер.

Розглянемо атрибути сутності «Авіа компанія»

1) ID – унікальний порядковий номер авіа компанії;

2) Назва – найменування;

3) Телефон – номер телефону;

4) Логотип – файл зображення.

Розглянемо атрибути сутності «Рейс»

1) ID – унікальний порядковий номер рейсу;

2) Авіа компанія – унікальний порядковий номер авіа компанії;

3) Країна відправлення – країна відправлення авіа рейсу;

4) Місто відправлення – місто відправлення авіа рейсу;

5) Аеропорт відправлення – аеропорт відправлення авіа рейсу;

6) Час відправлення – час відправлення авіа рейсу;

7) Країна прибуття – країна прибуття авіа рейсу;

8) Місто прибуття – місто прибуття авіа рейсу;

9) Аеропорт прибуття – аеропорт прибуття авіа рейсу;

10) Час прибуття – час прибуття авіа рейсу.

Розглянемо атрибути сутності «Квиток»

1) ID – унікальний порядковий номер квитка;

2) Рейс – унікальний порядковий номер рейсу;

3) Клієнт – унікальний порядковий номер Клієнта;

4) Договір – унікальний порядковий номер Договору;

5) Номер місця – ідентифікаційний номер місця в літаку.

Розглянемо атрибути сутності «Тур»

1) ID – унікальний порядковий номер туру;

2) Країна – країна призначення;

3) Місто – місто призначення;

4) Фото – фото туристичного місця.

Розглянемо атрибути сутності «Готель»

1) ID – унікальний порядковий номер готелю;

2) Назва – назва готелю;

3) Кількість зірок – кількість зірок;

4) Телефон – номер телефона;

5) Тур – унікальний номер туру;

6) Фото готелю – фото готелю.

Розглянемо атрибути сутності «Договір»

1) ID – унікальний порядковий номер договору;

2) Номер договору – номер договору;

3) Ціна – ціна за надання послуг компанії;

4) Дата – дата укладання договору;

5) Файл – файл договору в форматі doc або pdf;

6) Співробітник – унікальний ідентифікатор співробітника який оформив договір.

Розглянемо атрибути сутності «Договір_Готель»

1) Договір – унікальний порядковий номер договору;

2) Готель – унікальний порядковий номер готелю.

Після розгляду атрибутів всіх сутностей отримаємо ER-модель зображену на рисунку 2.9.

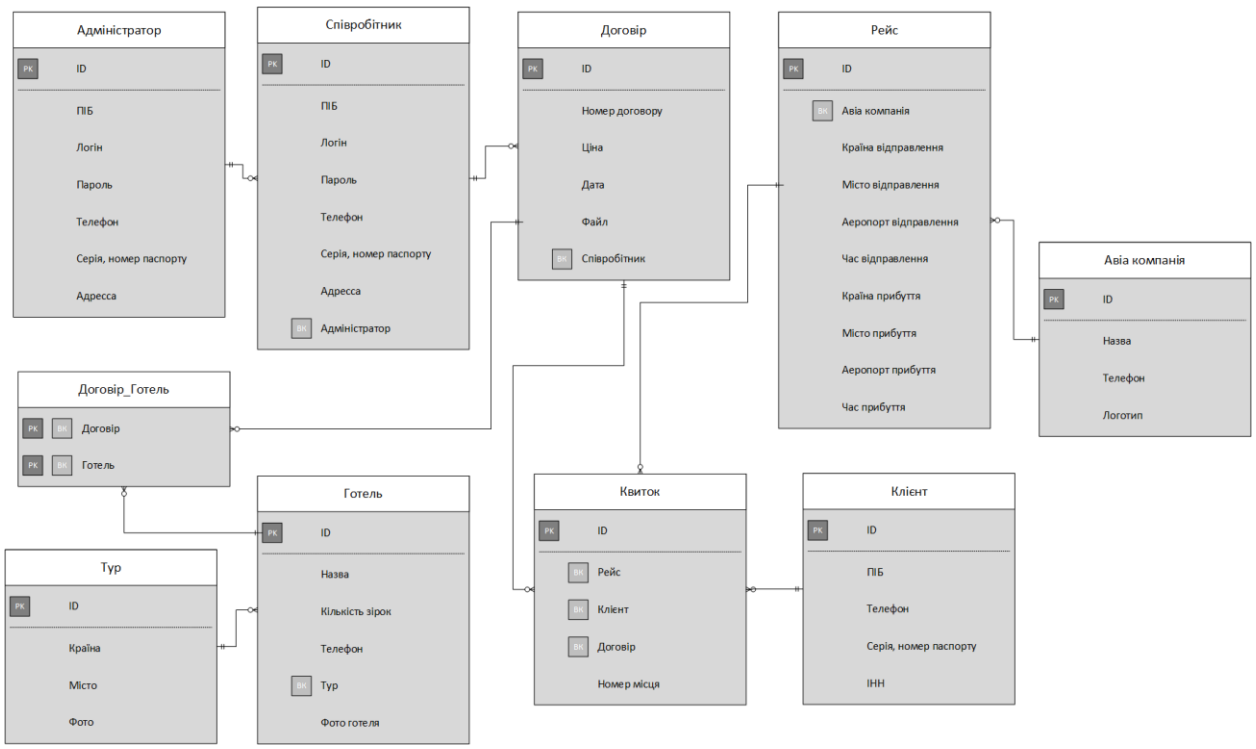


Рис. 2.9. Уточнена ER-діаграма

Поряд із методом "ER-діаграм" у проектуванні баз даних застосовується метод нормальних форм. Процес проектування баз даних з використанням методу нормальних форм є ітераційним і полягає в послідовному перекладі відношення з першої нормальної форми в нормальну форму вищого порядку за певними правилами. Кожна наступна нормальна форма обмежується певним типом функціональних залежностей та усуненням відповідних аномалій при виконанні операцій над відношеннями бази даних, а також збереження властивостей попередніх нормальних форм[9].

Теоретично для реляційних баз даних зазвичай виділяється наступна послідовність нормальних форм:

– перша нормальна форма або 1НФ;
– друга нормальна форма або 2НФ;
– третя нормальна форма або 3НФ;
– нормальна форма BCNF або НФБК;
– четверта нормальна форма або 4НФ;
– п'ята нормальна форма або 5НФ.

Таким чином, нормалізація – це процес ефективної організації даних у базі даних. Існує дві цілі процесу нормалізації: усунення зайвих даних (наприклад, збереження тих самих даних у кількох таблицях) і забезпечення того, щоб залежності даних мали сенс (зберігання лише пов’язаних даних у таблиці). Обидва вони є гідними цілями, оскільки вони зменшують обсяг простору, який споживає база даних, і забезпечують логічне зберігання даних. При проектуванні досить простих баз даних достатній рівень нормалізації до 3НФ.[10]

Після визначення нормалізації можна перейти безпосередньо до процесу нормалізації ER-моделі:

Щоб таблиця була в першій нормальній формі, вона повинна відповідати таким критеріям:

– одна клітинка не повинна містити більше одного значення (атомарність);

– повинен бути первинний ключ для ідентифікації;

– відсутність дубльованих рядків або стовпців;

– кожен стовпець повинен мати лише одне значення для кожного рядка таблиці.[11]

Наприклад, якщо у атрибуті Прізвище, ім'я, по батькові міститься прізвище, ім'я та по батькові користувача, вимога неподільності не виконується. Тут необхідно виділити в окремі атрибути ім'я та по батькові. Аналогічно слід вчинити з атрибутом Адреса, в результаті отримаємо ER-модель зображену на рисунку 2.10.

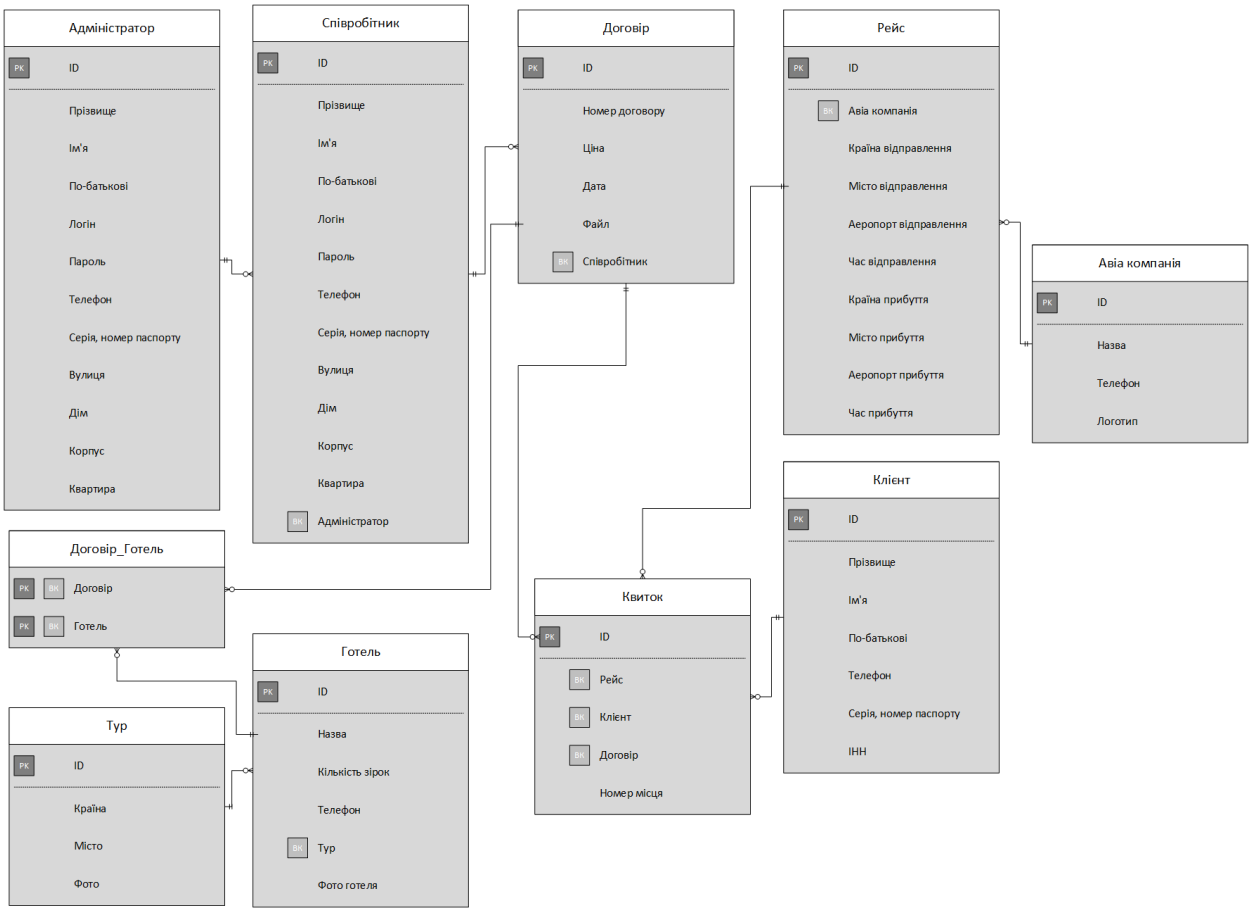


Рис. 2.10. ER-діаграма приведена до 3НФ

Перша нормальна форма усуває лише повторювані групи, а не надмірність. Тому є 2НФ.

Кажуть, що таблиця знаходиться в 2НФ, якщо вона відповідає таким критеріям:

– вона вже в 1 НФ;

– не має часткової залежності. Тобто всі неключові атрибути повністю залежать від первинного ключа.[12]

В нашому випадку первинні ключі кожного відношення однозначно визначають не ключові атрибути а отже ER-діаграма перебуває у 2НФ.

Коли таблиця знаходиться в 2НФ, це усуває повторювані групи та надмірність, але не усуває транзитивну часткову залежність.

Це означає, що непростий атрибут (атрибут, який не є частиною ключа) залежить від іншого непростого атрибута. Це те, що усуває третя нормальна форма (3НФ).

Отже, щоб таблиця була в 3НФ, вона повинна:

– бути в 2 НФ;

– не мати транзитивної часткової залежності.

В нашому випадку такі залежності відсутні а отже ER-діаграма перебуває у 3НФ.[13]

2.3. Ковзна середня

Ковзне середнє – це технічний індикатор, який використовують інвестори та трейдери, який визначає цінові дані шляхом створення постійно оновлюваної середньої ціни. Ковзні середні широко використовуються в технічному аналізі для визначення тенденцій, рівнів підтримки та опору, а також потенційних розворотів цін. Ковзне середнє є запізнілим індикатором, оскільки воно базується на минулих цінах і використовується для згладжування коливань цін і визначення тенденцій.

Ковзне середнє обчислюється шляхом взяття суми певної кількості цін і подальшого ділення цієї суми на кількість цін у розрахунку. Наприклад, 10-денне ковзне середнє буде розраховано шляхом додавання цін за останні 10 днів, а потім ділення цієї суми на 10. Потім цей процес повторюється для кожного дня, використовуючи останню ціну закриття в розрахунку.

Найпоширенішим типом ковзної середньої є проста ковзна середня (SMA), яка обчислюється шляхом додавання цін певного періоду та ділення на кількість періодів.[14]

Ковзне середнє використовується для згладжування коливань цін і спотових моделей. Його можна використовувати для точного визначення ймовірних рівнів підтримки та опору, а також точок входу та виходу з торгівлі. Трейдери можуть використовувати ковзні середні, щоб точно визначити

напрямок тренду та визначити, чи є він висхідним, низхідним або бічним трендом.

Три основні категорії ковзних середніх: прості ковзні середні (SMA), експоненціальні ковзні середні (EMA) і зважені ковзні середні (WMA). Обчислення простого ковзного середнього є простим. Він обчислюється шляхом множення загальної ціни за всі періоди на кількість періодів. Щоб побудувати 10-денну SMA, наприклад, об’єднайте ціни за останні 10 днів, а потім розділіть отриману суму на 10.[15]

2.4. Регресійний аналіз

Регресія – це статистичний метод, що використовується у фінансах, інвестиціях та інших дисциплінах, який намагається визначити силу та характер зв’язку між однією залежною змінною (зазвичай позначається Y) та рядом інших змінних (відомих як незалежні змінні).

Лінійна регресія, яка також називається простою регресією або звичайним методом найменших квадратів (МНК), є найпоширенішою формою цієї методики. Лінійна регресія встановлює лінійний зв'язок між двома змінними на основі лінії найкращого підходу . Таким чином, лінійна регресія графічно зображена за допомогою прямої лінії з нахилом, який визначає, як зміна однієї змінної впливає на зміну іншої. Відрізок Y лінійної регресії представляє значення однієї змінної, коли значення іншої дорівнює нулю. Також існують моделі нелінійної регресії , але вони набагато складніші.

Регресійний аналіз є потужним інструментом для виявлення зв’язків між змінними, що спостерігаються в даних, але не може легко вказати на причинний зв’язок. Він використовується в кількох контекстах у бізнесі, фінансах та економіці. Наприклад, він використовується, щоб допомогти інвестиційним менеджерам оцінити активи та зрозуміти взаємозв’язок між такими факторами, як ціни на сировинні товари та акціями компаній, що займаються цими товарами.

Регресія може допомогти фахівцям у сфері фінансів та інвестицій, а також фахівцям в інших сферах діяльності. Регресія також може допомогти передбачити продажі для компанії на основі погоди, попередніх продажів, зростання ВВП або інших умов. Модель оцінки капітальних активів (CAPM) – це часто використовувана регресійна модель у фінансах для визначення ціни активів і визначення вартості капіталу.

Моделі лінійної регресії часто використовують підхід найменших квадратів для визначення лінії найкращого підходу. Техніка найменших квадратів визначається шляхом мінімізації суми квадратів, створених математичною функцією. Квадрат, у свою чергу, визначається зведенням у квадрат відстані між точкою даних і лінією регресії або середнім значенням набору даних.[16]

Після завершення цього процесу (зазвичай це виконується сьогодні за допомогою програмного забезпечення) будується регресійна модель.

Предметом регресійного аналізу є дослідження залежності випадкової величини від сукупності випадкових і невипадкових величин.

У загальному вигляді залежність результативної ознаки Y від спільного і одночасного впливу факторних ознак $X_1, X_2, \dots, X_k$ (k – кількість факторних ознак) має вигляд:

$$Y = f(X_1, X_2, \dots, X_k, a_0, a_1, \dots, a_m) + e$$

де $f(X_1, X_2, \dots, X_k, a_0, a_1, \dots, a_m)$ – функція регресії, яка виражає об'єктивну закономірну залежність результативної ознаки від спільного впливу факторних ознак, $a_0, a_1, \dots, a_m$ – коефіцієнти регресії, e – випадкова величина, що виражає вплив неконтрольованих і неврахованих факторних ознак, а також помилок вимірювання.

З останнього виразу випливає, що:

$$e = Y - f(X_1, X_2, \dots, X_k, a_0, a_1, \dots, a_m)$$

тобто e – відхилення значень результативної ознаки від значень, обчислених за функцією регресії. Оцінкою функції регресії є рівняння регресії:

$$Y(X) = f(X_1, X_2, \dots, X_k, a_0, a_1, \dots, a_m)$$

Оцінка параметрів рівнянь парної регресії:

Вхідними даними для визначення величин коефіцієнтів регресії є вибірка, кожна варіанта якої представлена певним значенням результативної ознаки і відповідними їй значеннями факторних ознак $\{Y_i, X_{1i}, X_{2i}, \dots, X_{ki}\}_{i=1,\dots,n}$. При визначенні найбільш застосовуваним є метод найменших квадратів, який мінімізує суму квадратів відхилення значень результативної ознаки за даними вибірки від значень, обчислених за функцією регресії.[17]

$$E(a_0, a_1, \dots, a_m) = \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} [Y_i - f(X_1, X_2, \dots, X_k, a_0, a_1, \dots, a_m)]^2$$

Система рівнянь для визначення величин коефіцієнтів регресії є системою з $m+1$ рівняння з $m+1$ невідомим, носить назву системи нормальних рівнянь і має вигляд:

$$\begin{cases} \dfrac{\partial E(a_0, a_1, \dots, a_m)}{\partial a_0} = 0 \\ \dfrac{\partial E(a_0, a_1, \dots, a_m)}{\partial a_1} = 0 \\ \dfrac{\partial E(a_0, a_1, \dots, a_m)}{\partial a_m} = 0 \end{cases}$$

Рівняння парної лінійної регресії має вигляд:

$$Y(X) = a_1 X + a_0$$

Відповідна система нормальних рівнянь:

$$\begin{cases} \dfrac{\partial E(a_0, a_1)}{\partial a_0} = 2\sum_{i=1}^{n}(a_1 X_i + a_0 - Y_i) = 0 \\ \dfrac{\partial E(a_0, a_1)}{\partial a_0} = 2\sum_{i=1}^{n}(a_1 X_i + a_0 - Y_i)X_i = 0 \end{cases}$$

$$\begin{cases} a_1 \sum_{i=1}^{n} X_i + n a_0 = \sum_{i=1}^{n} Y_i \\ a_1 \sum_{i=1}^{n} X_i^2 + a_0 \sum_{i=1}^{n} X_i = \sum_{i=1}^{n} Y_i X_i \end{cases}$$

Система має просте аналітичне розв’язання, а саме:

$$a_0 = \frac{1}{n}\sum_{i=1}^{n}(Y_i - a_1 X_i) = \overline{Y} - a_1 \overline{X};$$

$$\sum_{i=1}^{n}\left(a_1 X_i^2 + a_0 X_i - Y_i X_i\right) = \sum_{i=1}^{n}(a_1 X_i^2 + \overline{Y} X_i - a_1 \overline{X} X_i - Y_i X_i)$$

$$a_1\left(\overline{X^2} - \overline{X}^2\right) = \overline{X * Y} - \overline{X} * \overline{Y};\ a_1 \widetilde{D}_X = \tilde{C}_{XY}$$

Тобто остаточно для значень коефіцієнтів регресії маємо:

$$a_1 = \frac{\tilde{C}_{XY}}{\widetilde{D}_X};\ \ a_0 = \overline{Y} - \frac{\tilde{C}_{XY}}{\widetilde{D}_X}\overline{X}$$

Тут $\widetilde{D}_X$ – статистична дисперсія змінної X, $\overline{X}$, $\overline{Y}$ – середні значення змінних X та Y, $\tilde{C}_{XY}$ – їх статистична коваріація.

Критерії оцінки якості моделей парної регресії:

Критерієм оцінки якості отриманої моделі регресії є оцінка статистичної значущості рівняння регресії в цілому і окремих його параметрів. Оцінка статистичної значущості рівняння регресії в цілому проводиться за допомогою F-критерію Фішера, а оцінка статистичної значущості її параметрів, в разі значущості рівняння регресії в цілому, проводиться за t-розподілом Стьюдента.

Для перевірки нульової гіпотези про незначущості рівняння регресії обчислюють контрольне значення величини:

$$F = \frac{Q_{reg}}{Q_{res}} * \frac{k_2}{k_1}$$

яка підпорядковується статистиці Фішера з $k_1 = 1, \quad k_2 = n - 2$ ступенями волі.

$$Q_{tot} = \sum_{i=1}^{n} \left(\overline{Y} - Y_i\right)^2$$

$$Q_{res} = \sum_{i=1}^{n} [Y(X_i) - Y_i]^2 = \sum_{i=1}^{n} (a_0 + a_1 X_i - Y_i)^2$$

$$Q_{reg} = Q_{tot} - Q_{res}$$

Тут $Q_{tot}, Q_{reg}, Q_{res}$ – відповідно повна, регресійна і залишкова суми квадратів відхилення результативної ознаки, $\overline{Y}$ – його середнє значення. Регресійна сума квадратів відхилень викликається розсіюванням значень результативної ознаки під впливом факторної ознаки включеної в модель, а залишкова характеризується впливом неврахованих факторних ознак. Чим менше залишкове розсіювання, тим менше вплив неврахованих факторних ознак і тим краще математична модель регресії відповідає даним спостережень,

так як зміни результативної ознаки в цьому випадку в основному пояснюється впливом розглянутих факторних ознак. Величина

$$R^2 = \frac{Q_{reg}}{Q_{tot}}$$

носить назву коефіцієнту детермінації. Він являє собою частину повного розсіювання значень результативної ознаки під впливом розглянутих факторних ознак. Чим більше коефіцієнт детермінації, тим краще обрана модель регресії відповідає даним спостережень. Якщо коефіцієнт детермінації дорівнює одиниці, то всі дані спостережень розташовані на лінії регресії. У випадку парної регресії коефіцієнт детермінації дорівнює квадрату коефіцієнта кореляції між результативною та факторною ознаками.

У разі, якщо контрольне значення перевищує критичне, яке визначається статистикою Фішера $F \geq F_{kp}(a, k_1, k_2)$, рівняння регресії вважається значущим.

Для оцінки статистичної значущості коефіцієнтів моделі регресії обчислюються значення контрольних величин

$$t = \left| \frac{a}{\tilde{S}_a} \right|$$

яка підпорядковується статистиці Стьюдента з $k = n - 2$ ступенями волі. Тут $a, \tilde{s}_a$ – відповідно оцінки значення коефіцієнта і його стандартної похибки. Останні розраховуються за формулами:

$$\tilde{S}_{a_0} = \tilde{\sigma}_{res} \sqrt{\frac{\sum_{i=1}^{n} X_i^2}{n \sum_{i=1}^{n} (X_i - \overline{X})^2}}$$

$$\tilde{S}_{a_1} = \frac{\tilde{\sigma}_{res}}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}}$$

$$\tilde{\sigma}_{res} = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - Y(X_i))^2}{n-2}}$$

Де $X_i, \overline{X}$ – відповідно i – е та середнє значення факторної ознаки, $D_{res} = \tilde{\sigma}_{res}^2$ – залишкова дисперсія.

У разі, якщо контрольне значення перевищує критичне, яке визначається статистикою Стьюдента $t \geq t_{kp}(a,k)$, відповідний коефіцієнт регресії вважається значущим. Величина довірчого інтервалу для значущого коефіцієнта визначається як:

$$\Delta_a = t(a,k) * \tilde{s}_a$$

При оцінці якості моделі регресії можливі наступні випадки:

– рівняння регресії на основі перевірки за F-критерієм Фішера в цілому статистично значимо і його параметри статистично значущі. Така модель може бути використана для прийняття рішень і прогнозування;

– рівняння регресії за критерієм Фішера статистично значуще, але хоча б один його параметр статистично незначущий. У цьому випадку модель придатна для прийняття деяких рішень, але не для прогнозування;

– рівняння регресії за критерієм Фішера статистично незначуще. У цьому випадку модель регресії вважається ненадійною і непридатною для використання в практиці.[18]

2.5. Висновки

Після дослідження та порівняння методів прогнозування часових рядів (ковзна середня, регресійний аналіз) було обрано регресійний аналіз для прогнозування часових рядів і ковзна середня для аналізу тенденції часового ряду. Також була розроблена ER-модель бази даних інформаційної системи і проведено нормалізацію бази даних.

РОЗДІЛ 3

СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Вибір мови програмування та середовища розробки

Для реалізації даної інформаційної системи було обрано мову програмування C# та платформу Windows Forms.

На сьогоднішній момент мова програмування C# одна з найпотужніших мов, що швидко розвиваються в IT-галузі.

C# – це сучасна, об’єктно-орієнтована мова програмування загального призначення. Він був розроблений Microsoft під керівництвом Андерса Хейлсберга та його команди в рамках ініціативи .Net і був схвалений Європейською асоціацією виробників комп’ютерів (ECMA) і Міжнародною організацією стандартів (ISO). C# є однією з мов для спільної мовної інфраструктури. Синтаксично C# дуже схожий на Java і простий для користувачів, які мають знання C, C++ або Java. Програми .Net є мультиплатформенними програмами, і фреймворк можна використовувати з таких мов, як C++, C#, Visual Basic, COBOL тощо. Він розроблений таким чином, щоб його могли використовувати інші мови. C# має багато інших причин бути популярним і затребуваним. Нижче наведені основні переваги C#:

– c# дуже ефективний в управлінні системою. Все сміття автоматично збирається.

– у C# немає проблеми з витоком пам’яті через високу резервну копію пам’яті.

– вартість обслуговування менша, і її використання безпечніше порівняно з іншими мовами.

– код C# компілюється в проміжну мову (Common Intermediate Language), яка є стандартною мовою, незалежно від цільової операційної системи та архітектури.

– простий і легкий у вивченні. C# розроблено як мову, яку легко вивчати, особливо для програмістів, які знайомі з такими мовами, як Java і C++. Він має чіткий синтаксис, що полегшує читання та написання коду.

– об’єктно-орієнтоване програмування. C# – це повністю об’єктно-орієнтована мова, яка дозволяє розробникам створювати багаторазовий код і з легкістю створювати складні програми.

– велика стандартна бібліотека. C# має велику стандартну бібліотеку, яка включає широкий спектр попередньо створених класів і функцій. Це полегшує розробникам виконання типових завдань без необхідності писати багато спеціального коду.

– підтримка між платформами. C# можна використовувати для розробки додатків для Windows, Linux і macOS, а також для розробки мобільних і веб-додатків.

– строго типізований. C# – це строго типізована мова, що означає, що типи даних перевіряються під час компіляції. Це допомагає зменшити кількість помилок і підвищити надійність коду.

– інтеграція з технологіями Microsoft. C# розроблено Microsoft і добре інтегрується з іншими технологіями Microsoft, такими як .NET, Azure та Visual Studio.[19]

Недоліки C#:

– c# є менш гнучким, оскільки він сильно залежить від .Net framework.

– c# працює повільно, і програму потрібно компілювати щоразу, коли вносяться будь-які зміни.

– обмежено платформами Microsoft. Незважаючи на те, що C# можна використовувати для розробки кросплатформних програм, він все ще пов’язаний переважно з платформами Microsoft, що обмежує його використання в деяких контекстах.

– збирання сміття. C# використовує автоматичне збирання сміття для керування пам’яттю, що в деяких випадках може призвести до проблем із продуктивністю.[20]

Як середовище розробки було обрано Microsoft Visual Studio.

Visual Studio це потужний засіб розробника, який можна використовувати для виконання всього циклу розробки в одному місці. Це комплексне інтегроване середовище розробки (IDE), яке можна використовувати для запису, редагування, налагодження та компіляції коду, а потім розгортання програми. Крім редагування та налагодження коду Visual Studio включає компілятори, засоби завершення коду, керування версіями, розширення та багато іншого, щоб покращити кожен етап процесу розробки програмного забезпечення.[21]

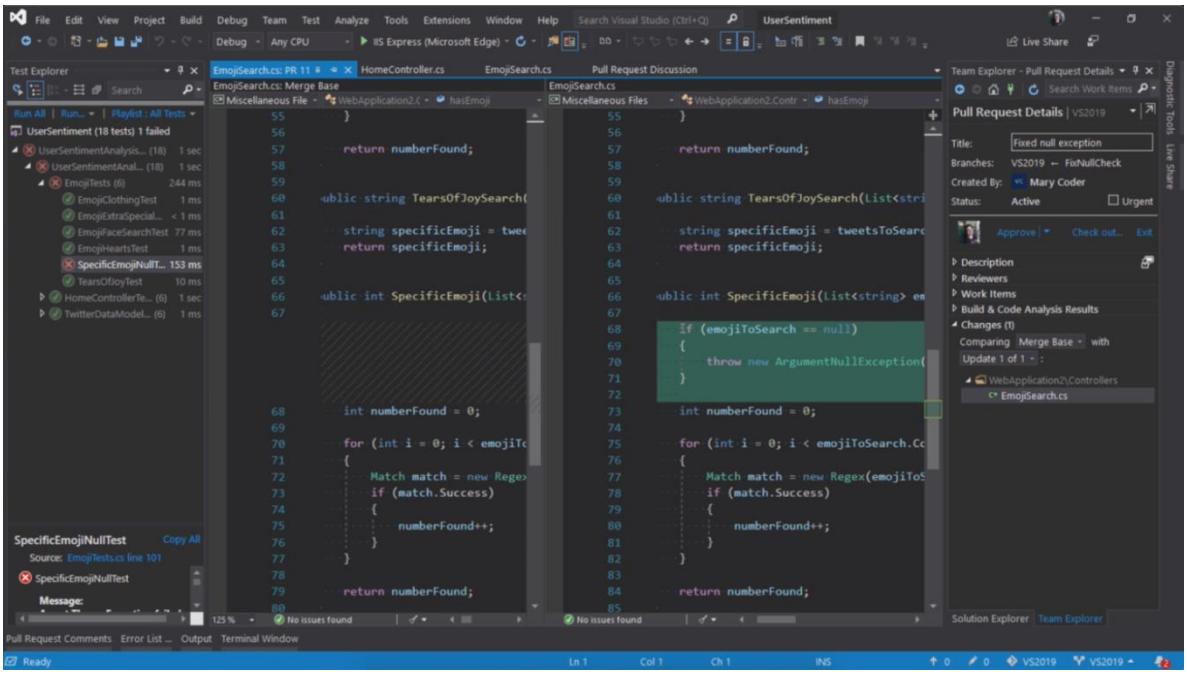


Рис. 3.1. Інтерфейс середовища розробки MS Visual Studio

3.2. Структура проекту

Проект який є результатом розробки має наступні файли:

– Program.cs – місце з якого починається програма (точка входу до програми);

– App. config – звичайний XML файл, всередині якого за умовчанням міститься рядок декларації та один кореневий елемент configuration. Сам файл

конфігурації, знову ж таки за замовчуванням, зберігається в папці поточного проекту;

– Form_autorization.cs – файл форми авторизації користувача;

– Form_main.cs – файл форми з головним меню програми;

– Form_account_admin.cs – файл форми редагування даних облікового запису адміністратора;

– Form_account_emploee.cs – файл форми редагування даних облікового запису співробітника;

– Form_emploee.cs – файл форми редагування даних про облікові записи співробітників;

– Form_emploee_add.cs – файл форми додавання даних про облікові записи співробітників;

– Form_tour.cs – файл форми редагування даних про туристичні напрямки компанії;

– Form_tour_add.cs – файл форми додавання даних про туристичні напрямки компанії;

– Form_hotel.cs – файл форми редагування даних про готелі;

– Form_hotel_add.cs – файл форми додавання даних про готелі;

– Form_airline.cs – файл форми редагування даних про авіакомпанії;

– Form_airline_add.cs – файл форми додавання даних про авіакомпанії;

– Form_flight.cs – файл форми редагування даних про авіарейси;

– Form_flight_add.cs – файл форми додавання даних про авіарейси;

– Form_document.cs – файл форми редагування даних про договори компанії;

– Form_document_add.cs – файл форми додавання даних про договори компанії;

– Form_document_hotel.cs – файл форми редагування даних про готелі які закріплені в договорах;

– Form_client.cs – файл форми редагування даних про клієнтів компанії;

– Form_client_add.cs – файл форми додавання даних про клієнтів компанії;

– Form_ticket.cs – файл форми редагування даних про авіаквитки;

– Form_future.cs – файл форми для відображення прогнозу попиту на тури;

– accountsDataSet.xsd – файл набору даних з даними про облікові записи користувачів;

– emploeesDataSet.xsd – файл набору даних з даними про облікові записи співробітників;

– tourDataSet.xsd – файл набору даних з даними про туристичні напрямки;

– hotelDataSet.xsd – файл набору даних з даними про готелі;

– airlineDataSet.xsd – файл набору даних з даними про авіакомпанії;

– flightDataSet.xsd – файл набору даних з даними про авіарейси;

– documentDataSet.xsd – файл набору даних з даними про договори;

– document_hotelDataSet.xsd – файл набору даних з даними про готелі які пов’язані з договорами;

– clientDataSet.xsd – файл набору даних з даними про клієнтів компанії;

– ticketDataSet.xsd – файл набору даних з даними про авіаквитки.

3.3. Опис роботи розробленого програмного забезпечення

Назва програми: TourAnalysis.exe

Програмне забезпечення, необхідне для функціонування програми: Microsoft SQL Server 2019, Microsoft .NET Framework 4

Мови програмування: C#

Функціональне призначення:

Програма призначена для ведення даних туристичної компанії та розрахунку прогнозу попиту на туристичні напрямки запропоновані компанією і побудови графіків прогнозу.

Опис структури:

В розробленій системі є двадцять вікон в залежності від прав доступу користувачу надається доступ до певного функціоналу інформаційної системи. Особливість полягає у тому що кожен користувач має доступ до вікна з прогнозуванням попиту на туристичні напрямки, тож коли система має достатньо даних про придбані тури кожен користувач може розрахувати прогноз попиту щодо обраного туру.

Для початку роботи з програмою користувачу необхідно спочатку пройти авторизацію, для цього після запуску програми у вікні авторизації потрібно обрати тип облікового запису, ввести логін і пароль та натиснути кнопку “Вхід”.

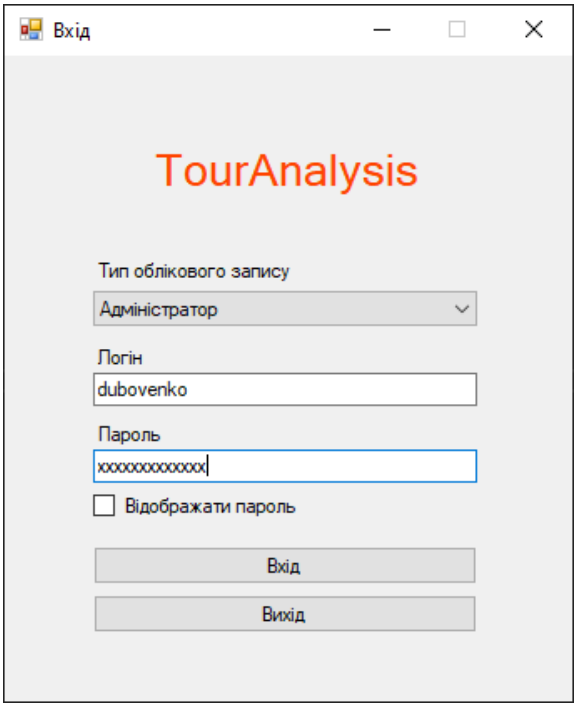


Рис. 3.2. Вікно авторизації

Після проходження авторизації з’явиться головне вікно програми в якому розташовано головне меню програми. Пункти головного меню змінюють свій стан з доступного на не доступний в залежності від типу користувача.

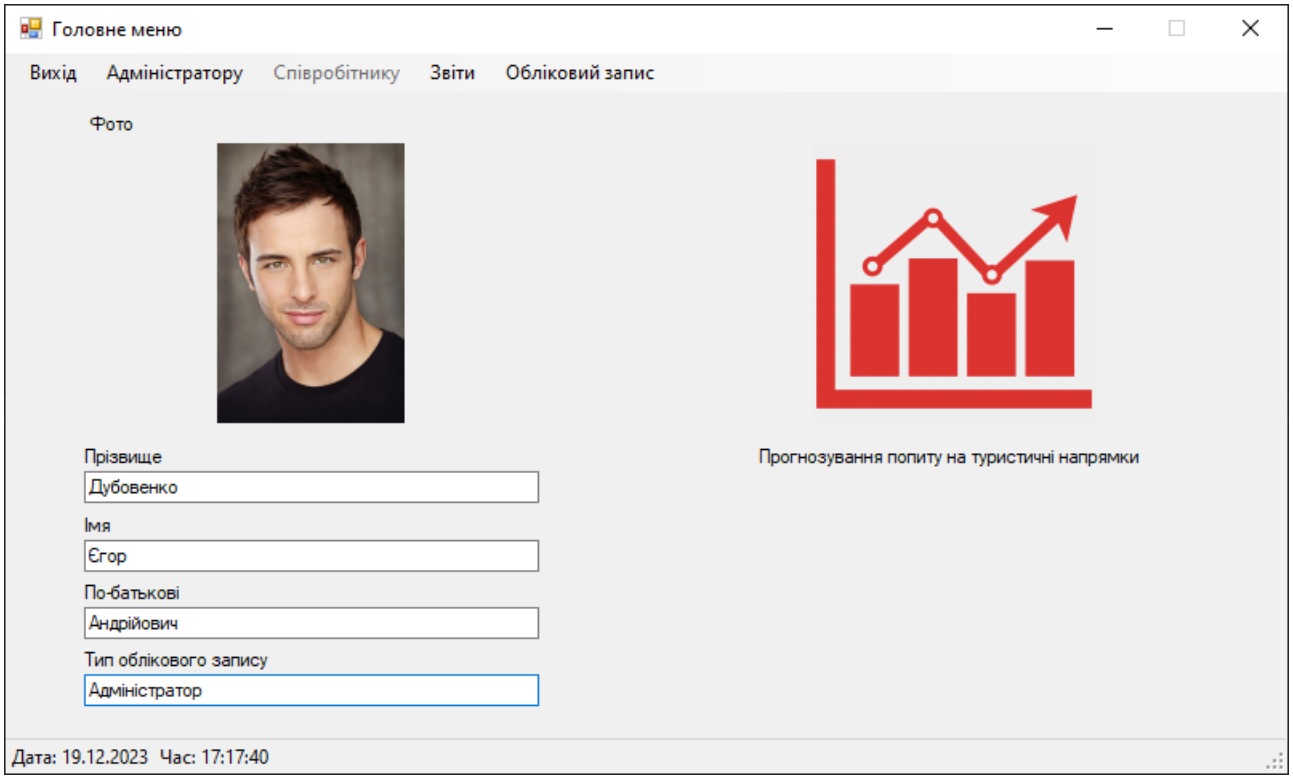


Рис. 3.3. Головне вікно програми

Для того щоб згорнути головне вікно потрібно натиснути кнопку у вигляді горизонтальної риски у верхній правій частині вікна.

Для розгортання вікна на весь екран необхідно натиснути кнопку у вигляді квадрата у верхній правій частині вікна.

Для виходу з програми потрібно натиснути кнопку у вигляді хрестика у верхній правій частині вікна.

Для редагування даних про співробітників компанії необхідно в головному вікні програми обрати пункт меню “Адміністратору”, “Співробітники”. Після цього з’явиться вікно редагування даних про співробітників. В центрі вікна розташована таблиця з даними. На панелі зліва розташовані інструменти для пошуку записів в таблиці. У верхній частині вікна розташоване меню для виконання операцій з даними.

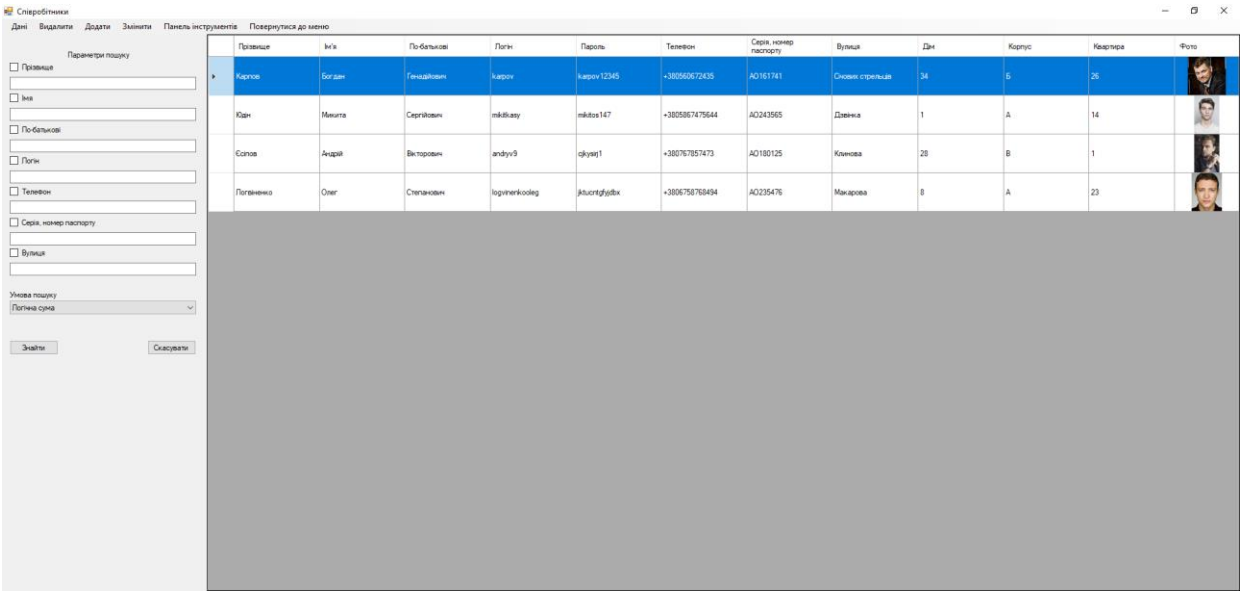


Рис. 3.4. Вікно редагування даних про співробітників

Щоб виконати пошук даних потрібно обрати поля по яких буде виконуватись пошук, обрати умову пошуку і натиснути кнопку “Знайти”, якщо потрібно скасувати результати пошуку необхідно натиснути кнопку “Скасувати”.

Щоб приховати або відкрити панель інструментів потрібно натиснути пункт меню “Панель інструментів”.

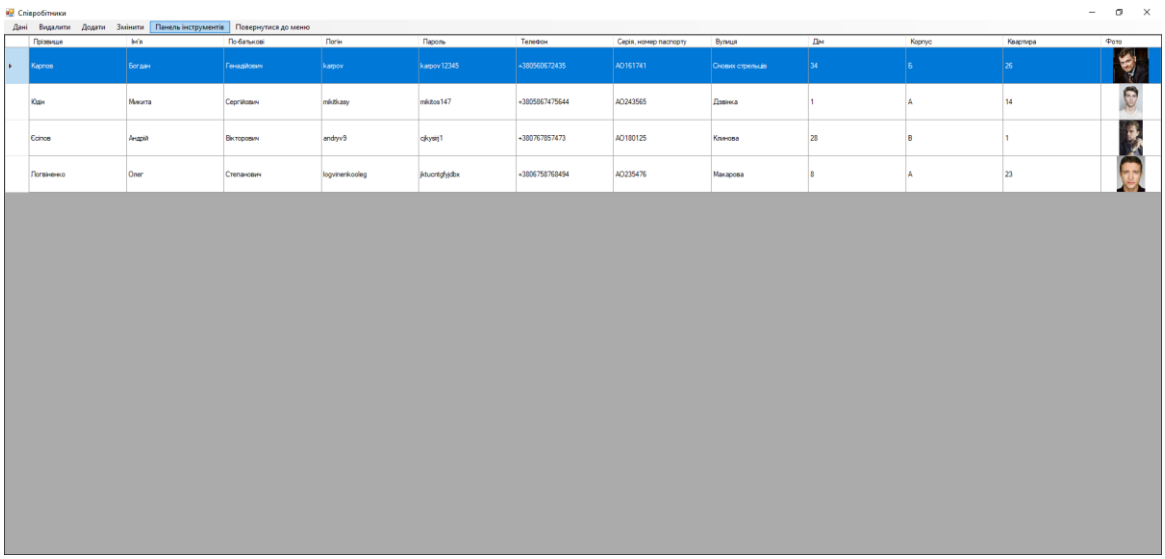


Рис. 3.5. Вікно редагування даних про співробітників з прихованою панеллю інструментів

Для збереження або оновлення даних таблиці потрібно обрати пункт меню “Данні”, “Зберегти” або “Оновити”. Щоб видалити запис з таблиці потрібно натиснути на нього та обрати пункт меню “Видалити”. Для додавання даних до таблиці потрібно натиснути пункт меню “Додати” після цього з’явиться вікно додавання даних про співробітників.

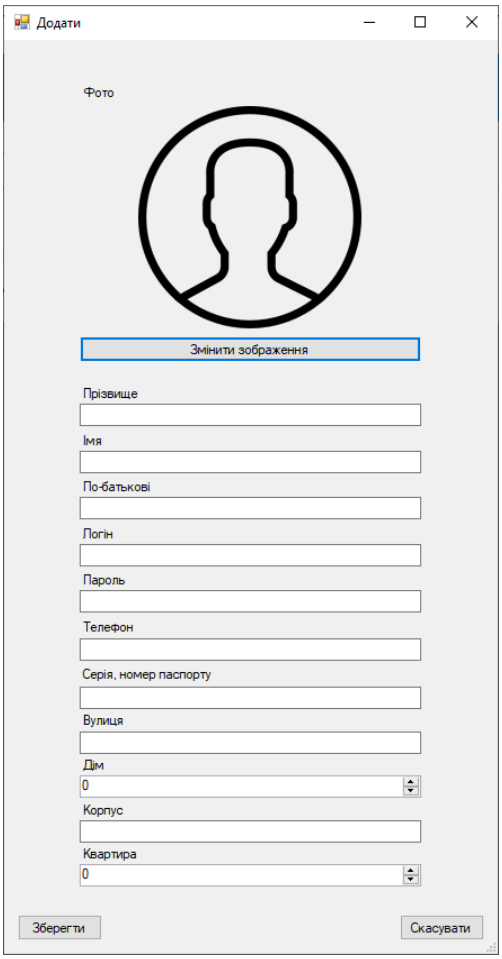


Рис. 3.6. Вікно додавання даних про співробітників

У вікні додавання даних потрібно обрати фото співробітника і заповнити всі текстові поля а потім натиснути кнопку “Зберегти”.

Щоб змінити будь-який запис в таблиці співробітників потрібно обрати цей запис і натиснути пункт меню “Змінити” після цього з’явиться вікно редагування даних співробітника.

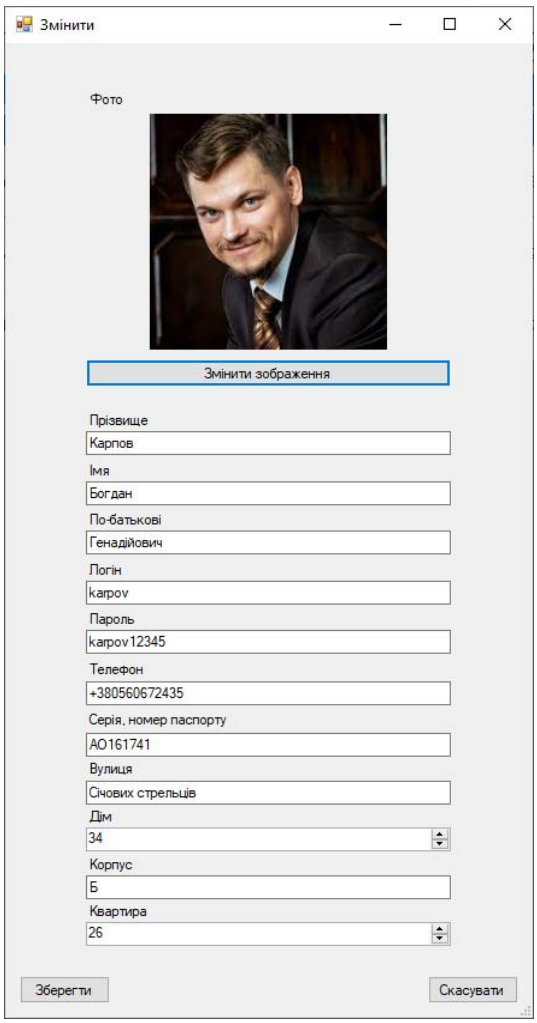


Рис. 3.7. Вікно редагування даних співробітника

У вікні редагування даних після внесення будь-яких змін потрібно натиснути кнопку “Зберегти”.

Щоб повернутися до головного меню необхідно натиснути пункт “Повернутися до меню”.

Для редагування даних про тури необхідно в головному вікні програми обрати пункт меню “Адміністратору”, “Тури”. Після цього з’явиться вікно редагування даних про тури. У верхній частині вікна розташоване меню для виконання операцій з даними. На панелі зліва розташовані інструменти для пошуку записів в таблиці. В центрі вікна розташована таблиця з даними.

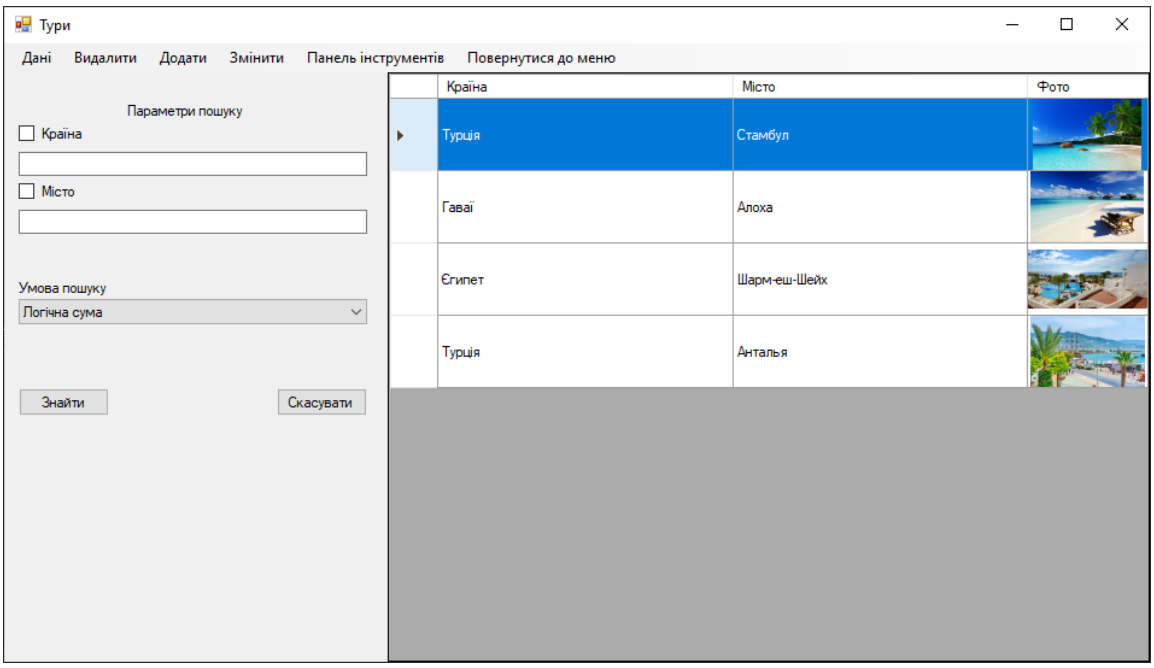


Рис. 3.8. Вікно редагування даних про тури

Щоб виконати пошук даних потрібно обрати поля по яких буде виконуватись пошук, обрати умову пошуку і натиснути кнопку “Знайти”, якщо потрібно скасувати результати пошуку необхідно натиснути кнопку “Скасувати”.

Щоб приховати або відкрити панель інструментів потрібно натиснути пункт меню “Панель інструментів”.

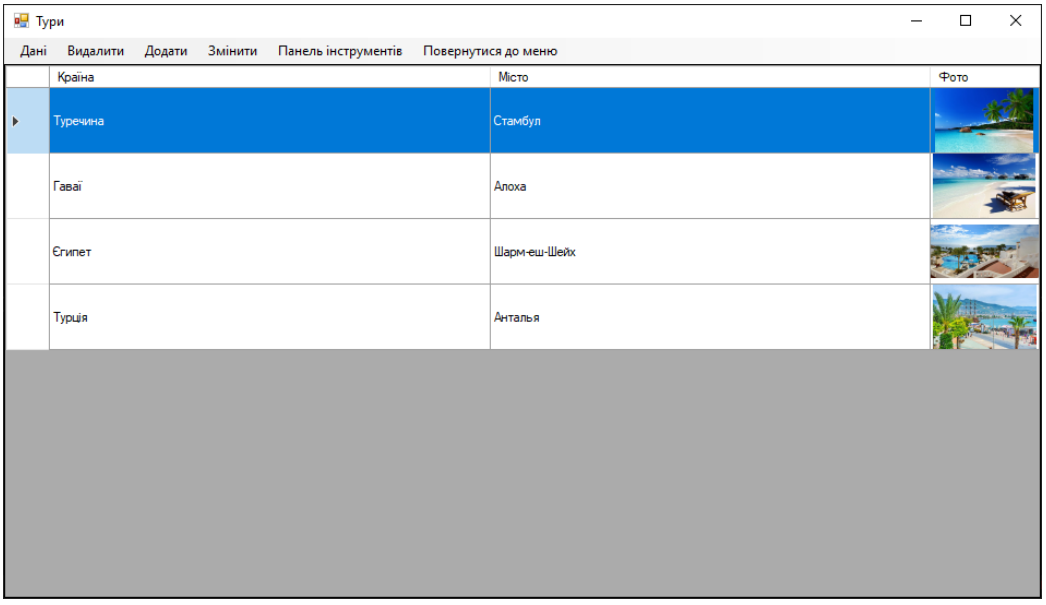


Рис. 3.9. Вікно редагування даних про тури з прихованою панеллю інструментів

Для збереження або оновлення даних таблиці потрібно обрати пункт меню “Данні”, “Зберегти” або “Оновити”. Щоб видалити запис з таблиці потрібно натиснути на нього та обрати пункт меню “Видалити”. Для додавання даних до таблиці потрібно натиснути пункт меню “Додати” після цього з’явиться вікно додавання даних про тури.

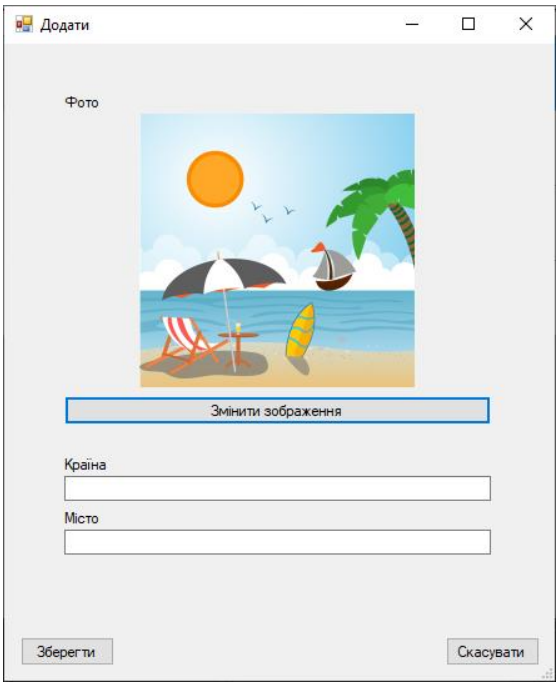


Рис. 3.10. Вікно додавання даних про тури

У вікні додавання даних потрібно обрати фото туру і заповнити всі текстові поля а потім натиснути кнопку “Зберегти”.

Щоб змінити будь-який запис в таблиці турів потрібно обрати цей запис і натиснути пункт меню “Змінити” після цього з’явиться вікно редагування даних про тур.

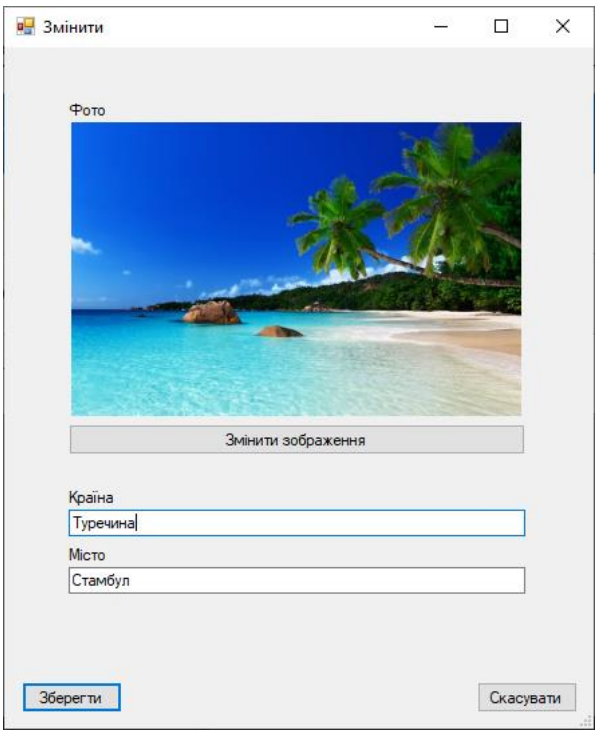


Рис. 3.11. Вікно редагування даних про тур

У вікні редагування даних після внесення будь-яких змін потрібно натиснути кнопку “Зберегти”.

Для того щоб згорнути вікно редагування даних потрібно натиснути кнопку у вигляді горизонтальної риски у верхній правій частині вікна.

Для розгортання вікна на весь екран необхідно натиснути кнопку у вигляді квадрата у верхній правій частині вікна.

Для виходу з програми потрібно натиснути кнопку у вигляді хрестика у верхній правій частині вікна.

Щоб повернутися до головного меню необхідно натиснути пункт “Повернутися до меню”.

Для редагування даних про готелі необхідно в головному вікні програми обрати пункт меню “Адміністратору”, “Готелі”. Після цього з’явиться вікно редагування даних про готелі. У верхній частині вікна розташоване меню для виконання операцій з даними. На панелі зліва розташовані інструменти для пошуку записів в таблиці. В центрі вікна розташовані таблиці з даними про тури та готелі.

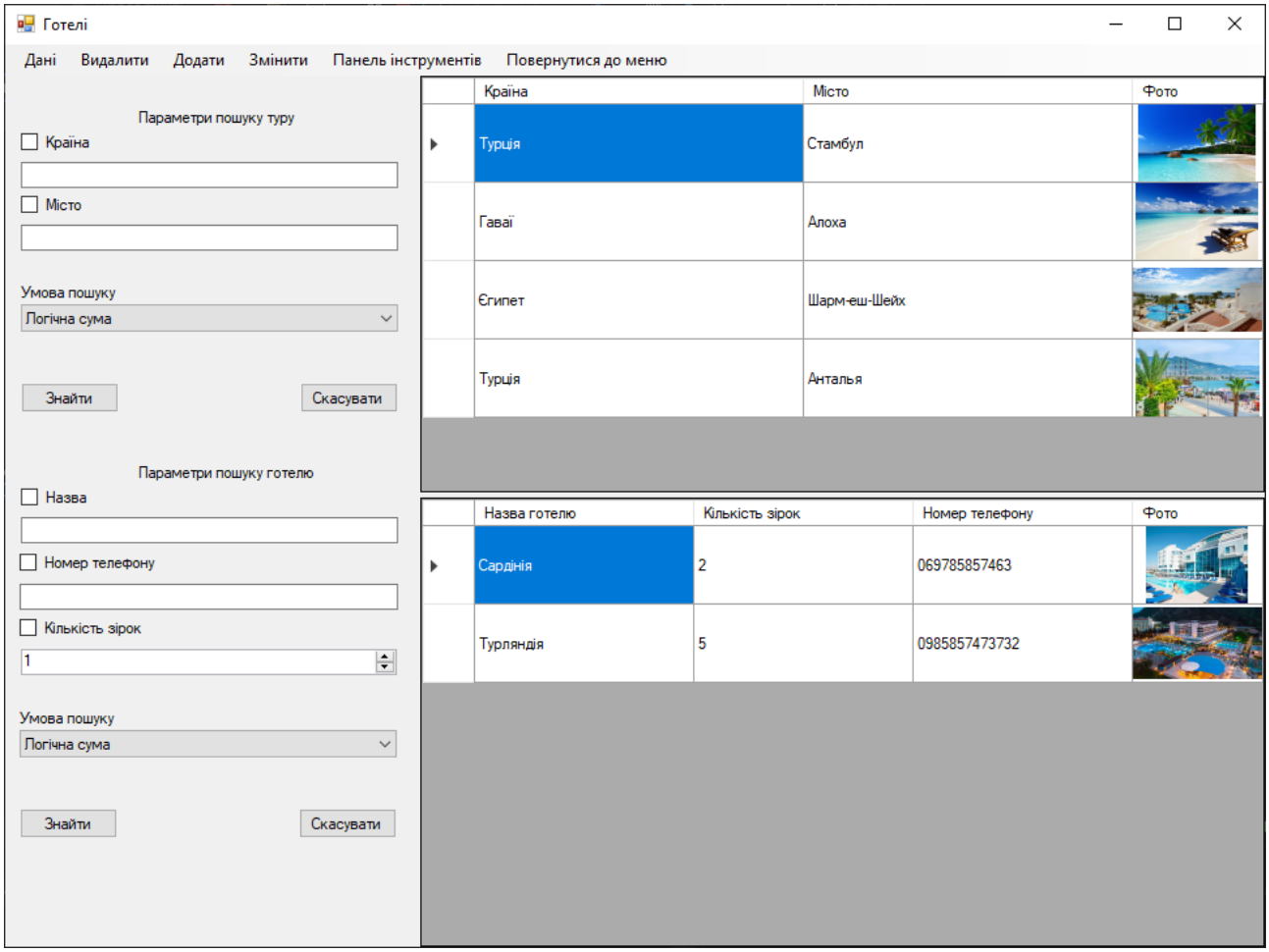


Рис. 3.12. Вікно редагування даних про готелі

Щоб виконати пошук даних потрібно обрати поля по яких буде виконуватись пошук, обрати умову пошуку і натиснути кнопку “Знайти”, якщо потрібно скасувати результати пошуку необхідно натиснути кнопку “Скасувати”.

Для того щоб згорнути вікно “Готелі” потрібно натиснути кнопку у вигляді горизонтальної риски у верхній правій частині вікна.

Для розгортання вікна на весь екран необхідно натиснути кнопку у вигляді квадрата у верхній правій частині вікна.

Для виходу з програми потрібно натиснути кнопку у вигляді хрестика у верхній правій частині вікна.

Щоб приховати або відкрити панель інструментів потрібно натиснути пункт меню “Панель інструментів”.

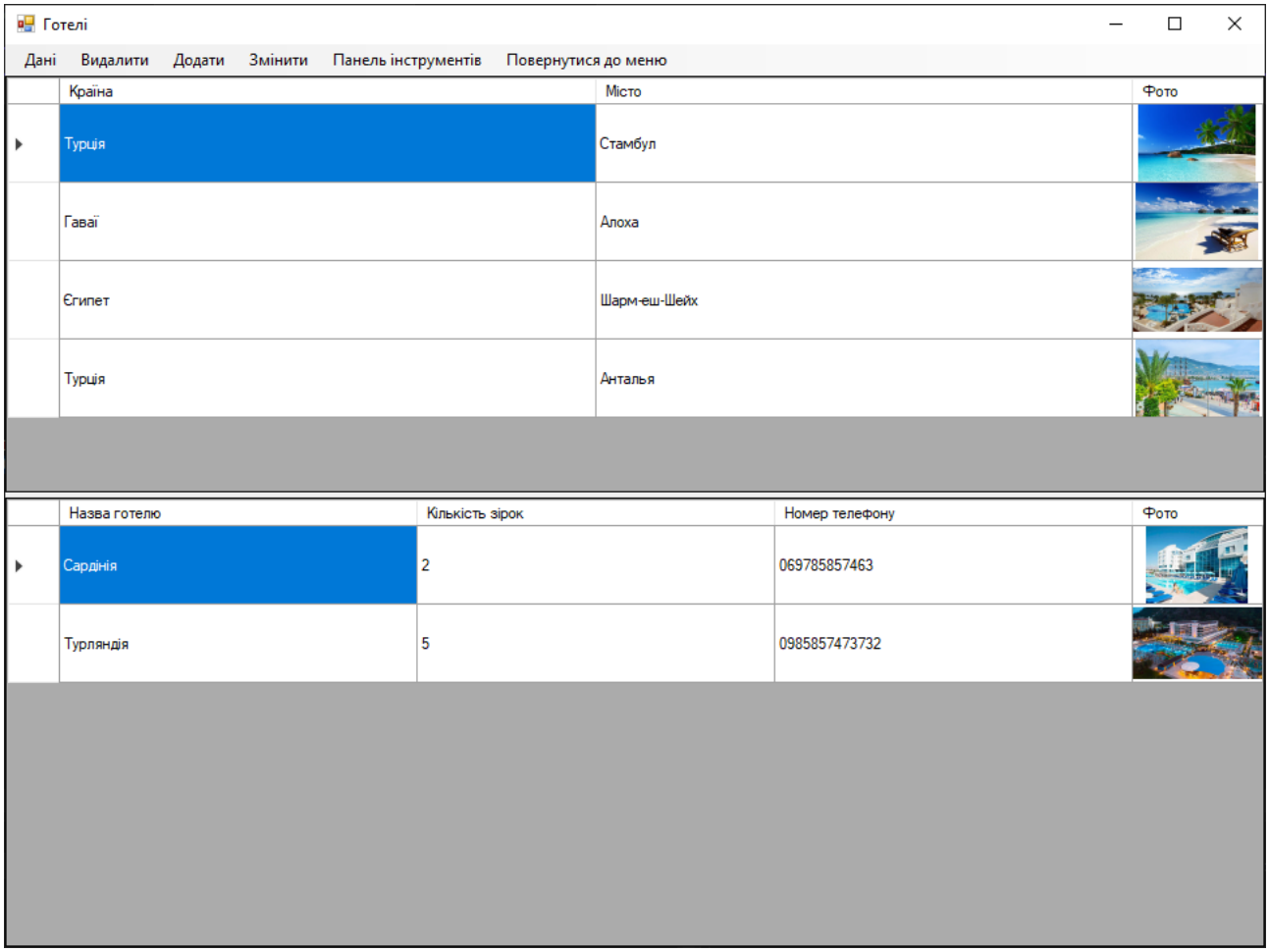


Рис. 3.13. Вікно редагування даних про готелі без панелі інструментів

Для того щоб згорнути вікно “Готелі” потрібно натиснути кнопку у вигляді горизонтальної риски у верхній правій частині вікна.

Для розгортання вікна на весь екран необхідно натиснути кнопку у вигляді квадрата у верхній правій частині вікна.

Для виходу з програми потрібно натиснути кнопку у вигляді хрестика у верхній правій частині вікна.

Для збереження або оновлення даних таблиці потрібно обрати пункт меню “Данні”, “Зберегти” або “Оновити”. Щоб видалити запис з таблиці потрібно натиснути на нього та обрати пункт меню “Видалити”. Для додавання даних до таблиці потрібно натиснути пункт меню “Додати” після цього з’явиться вікно додавання даних про готелі.

Рис. 3.14. Вікно додавання даних про готелі

У вікні додавання даних потрібно обрати фото готелю і заповнити всі текстові поля а потім натиснути кнопку “Зберегти”.

Щоб змінити будь-який запис в таблиці готелів потрібно обрати цей запис і натиснути пункт меню “Змінити” після цього з’явиться вікно редагування даних про готель.

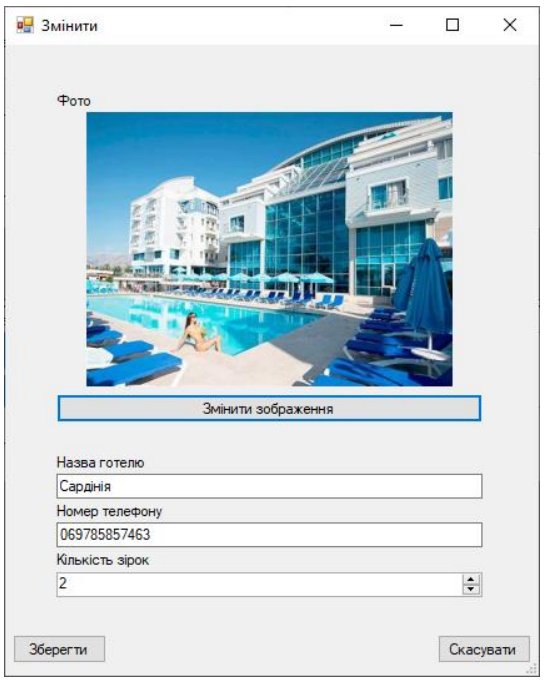


Рис. 3.15. Вікно редагування даних про готель

У вікні редагування даних після внесення будь-яких змін потрібно натиснути кнопку “Зберегти”.

Щоб повернутися до головного меню необхідно натиснути пункт “Повернутися до меню”.

Для редагування даних про авіакомпанії необхідно в головному вікні програми обрати пункт меню “Адміністратору”, “Авіакомпанії”. Після цього з’явиться вікно редагування даних про авіакомпанії. У верхній частині вікна розташоване меню для виконання операцій з даними. На панелі зліва розташовані інструменти для пошуку записів в таблиці. В центрі вікна розташована таблиця з даними.

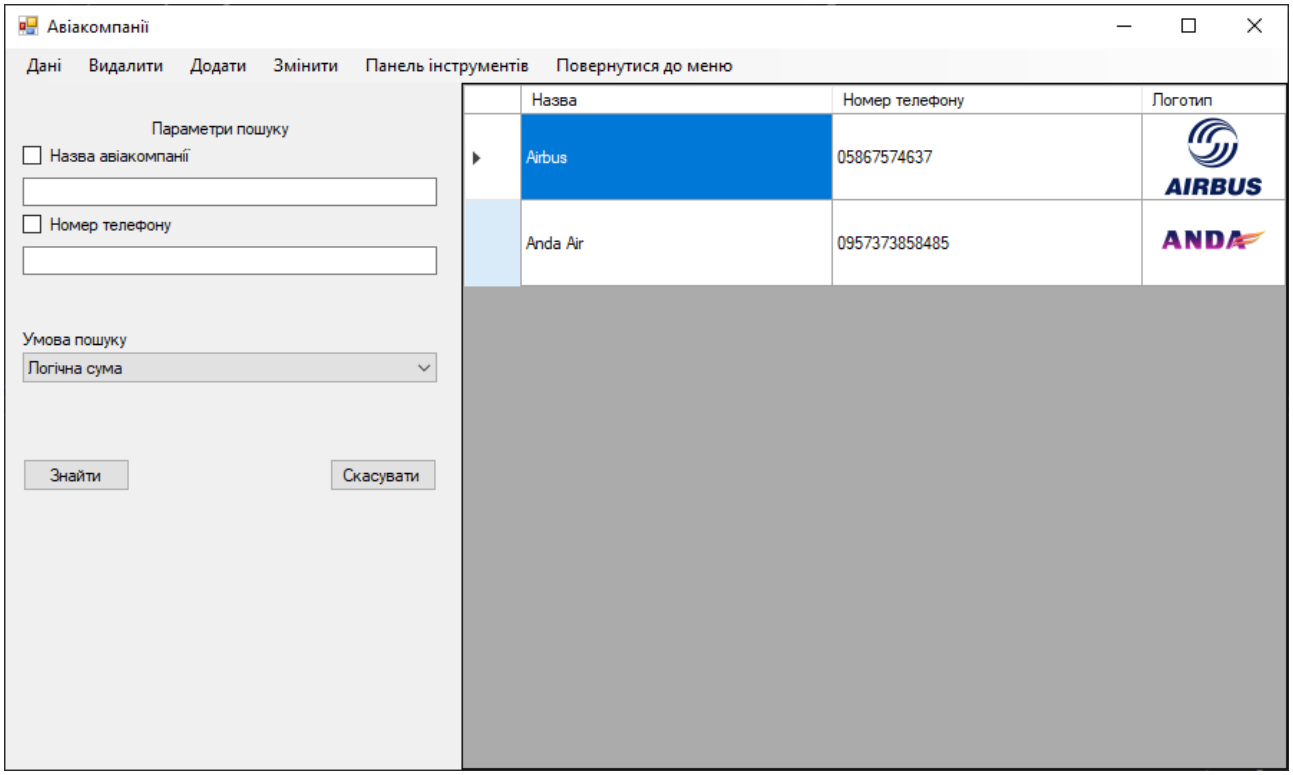


Рис. 3.16. Вікно редагування даних про авіакомпанії

Щоб виконати пошук даних потрібно обрати поля по яких буде виконуватись пошук, обрати умову пошуку і натиснути кнопку “Знайти”, якщо потрібно скасувати результати пошуку необхідно натиснути кнопку “Скасувати”.

Щоб приховати або відкрити панель інструментів потрібно натиснути пункт меню “Панель інструментів”.

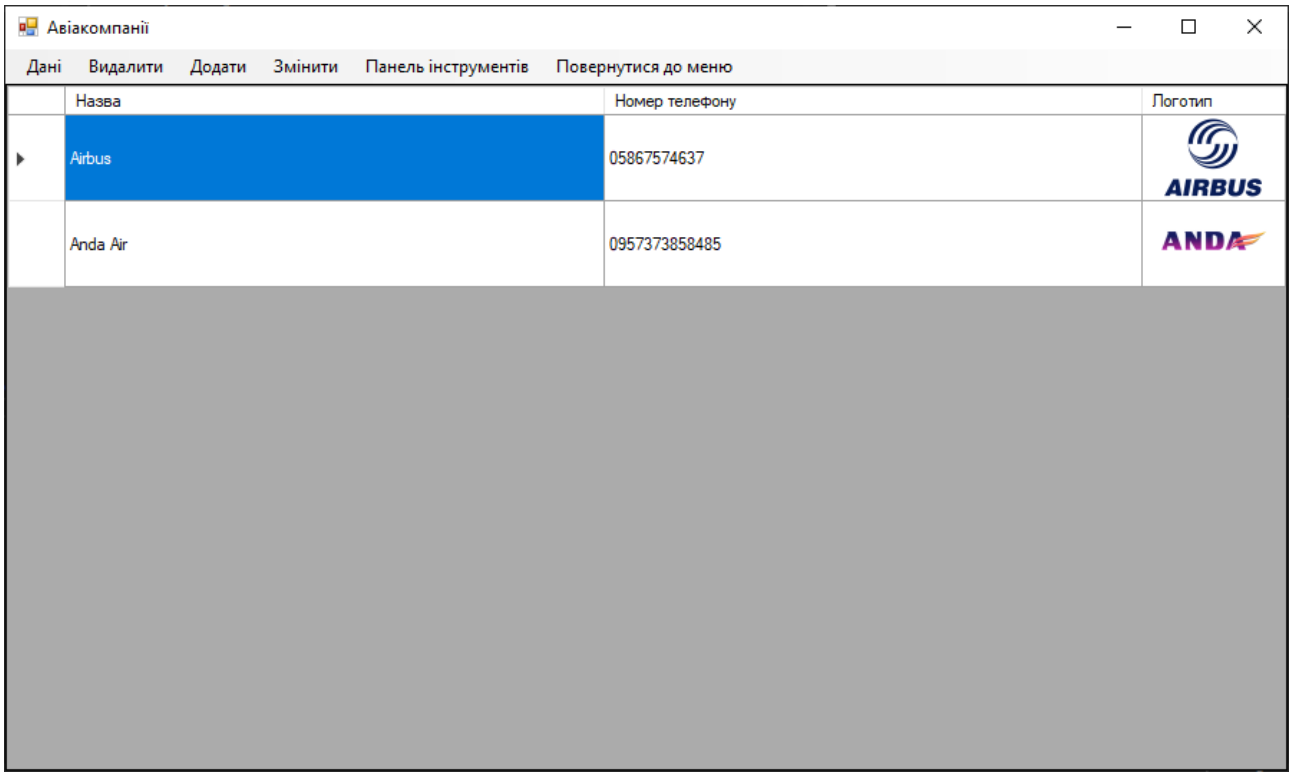


Рис. 3.17. Вікно редагування даних про авіакомпанії без панелі інструментів

Для того щоб згорнути вікно “Авіакомпанії” потрібно натиснути кнопку у вигляді горизонтальної риски у верхній правій частині вікна.

Для розгортання вікна на весь екран необхідно натиснути кнопку у вигляді квадрата у верхній правій частині вікна.

Для виходу з програми потрібно натиснути кнопку у вигляді хрестика у верхній правій частині вікна.

Для збереження або оновлення даних таблиці потрібно обрати пункт меню “Данні”, “Зберегти” або “Оновити”. Щоб видалити запис з таблиці потрібно натиснути на нього та обрати пункт меню “Видалити”. Для додавання даних до таблиці потрібно натиснути пункт меню “Додати” після цього з’явиться вікно додавання даних про авіакомпанії.

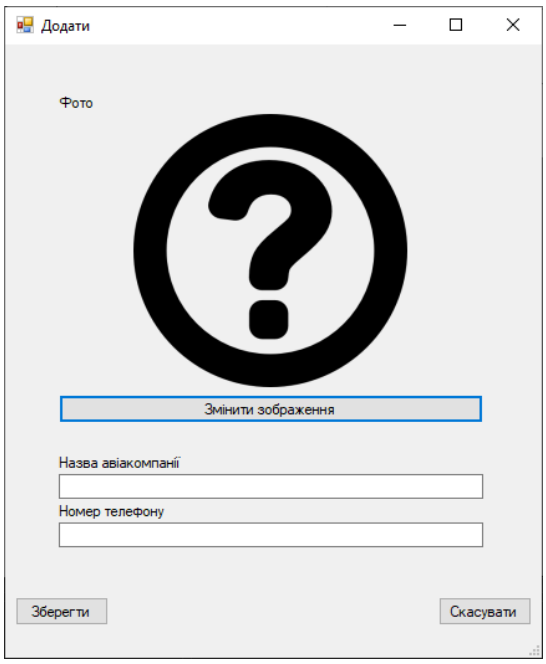


Рис. 3.18. Вікно додавання даних про авіакомпанії

У вікні додавання даних потрібно обрати фото логотипу авіакомпанії і заповнити всі текстові поля а потім натиснути кнопку “Зберегти”.

Щоб змінити будь-який запис в таблиці авіакомпаній потрібно обрати цей запис і натиснути пункт меню “Змінити” після цього з’явиться вікно редагування даних про авіакомпанію.

Рис. 3.19. Вікно редагування даних про авіакомпанію

У вікні редагування даних після внесення будь-яких змін потрібно натиснути кнопку “Зберегти”.

Щоб повернутися до головного меню необхідно натиснути пункт “Повернутися до меню”.

Для редагування даних про авіарейси необхідно в головному вікні програми обрати пункт меню “Адміністратору”, “Рейси”. Після цього з’явиться вікно редагування даних про авіарейси. У верхній частині вікна розташоване меню для виконання операцій з даними. На панелі зліва розташовані інструменти для пошуку записів в таблиці. В центрі вікна розташовані таблиці з даними про авіакомпанії та авіарейси.

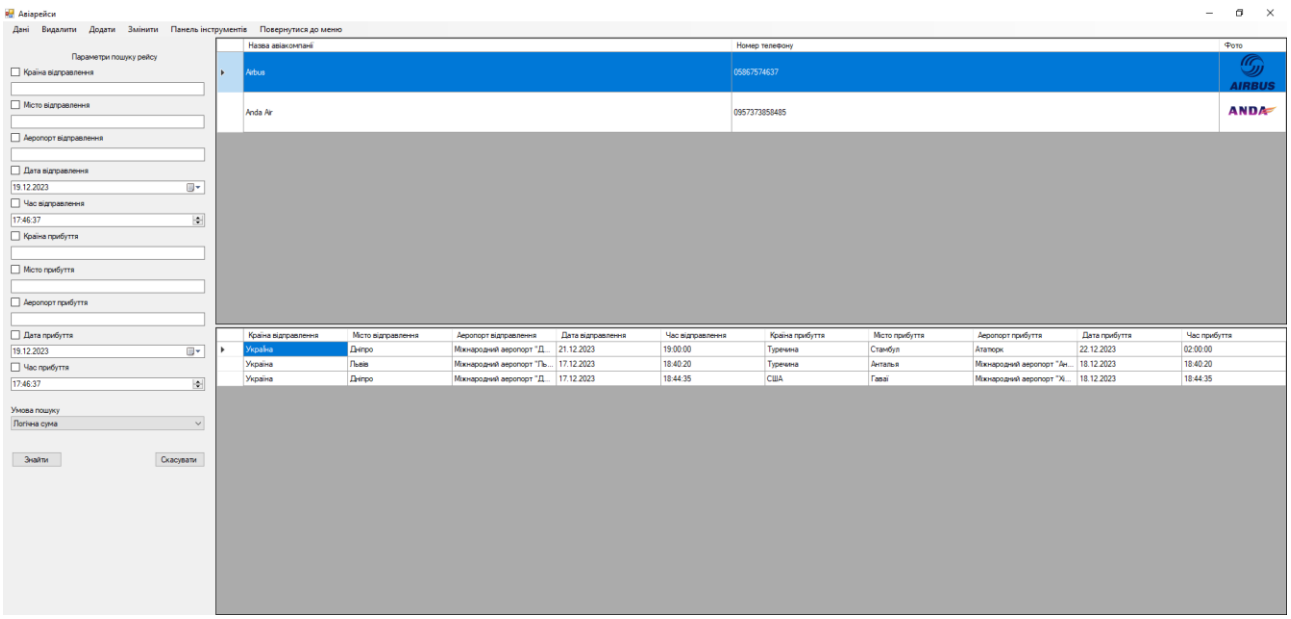


Рис. 3.20. Вікно редагування даних про авіарейси

Щоб виконати пошук даних потрібно обрати поля по яких буде виконуватись пошук, обрати умову пошуку і натиснути кнопку “Знайти”, якщо потрібно скасувати результати пошуку необхідно натиснути кнопку “Скасувати”.

Щоб приховати або відкрити панель інструментів потрібно натиснути пункт меню “Панель інструментів”.

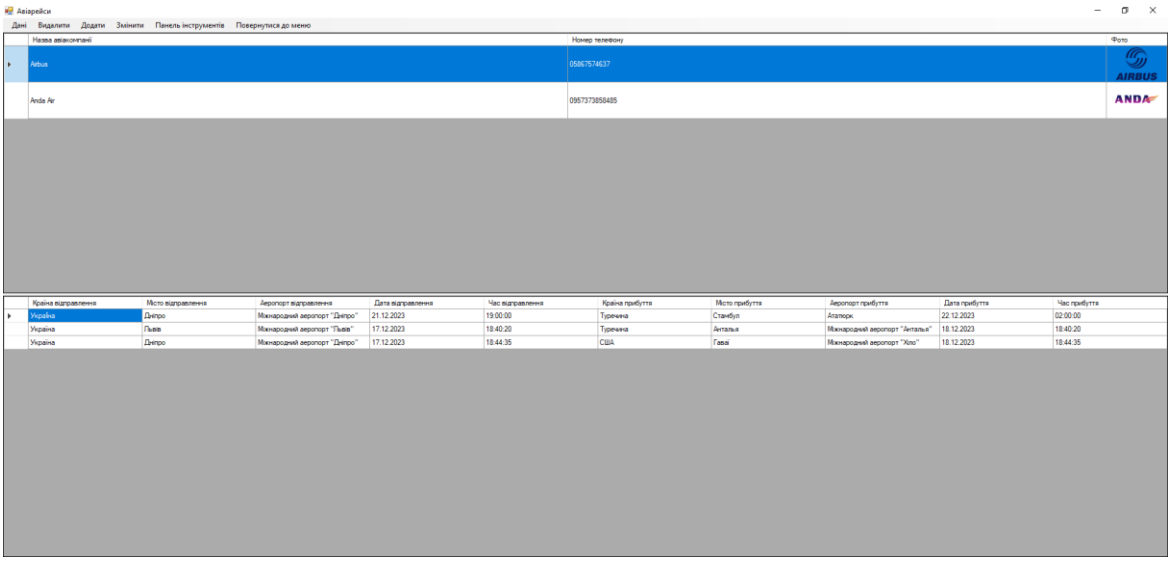


Рис. 3.21. Вікно редагування даних про авіарейси без панелі інструментів

Для збереження або оновлення даних таблиці потрібно обрати пункт меню “Данні”, “Зберегти” або “Оновити”. Щоб видалити запис з таблиці потрібно натиснути на нього та обрати пункт меню “Видалити”. Для додавання даних до таблиці потрібно натиснути пункт меню “Додати” після цього з’явиться вікно додавання даних про авіарейси.

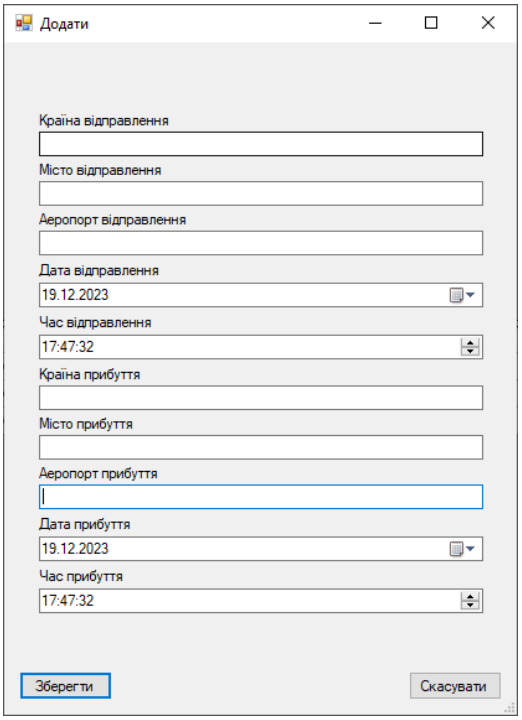


Рис. 3.22. Вікно додавання даних про авіарейси

У вікні додавання даних потрібно заповнити всі текстові поля а потім натиснути кнопку “Зберегти”.

Щоб змінити будь-який запис в таблиці авіарейсів потрібно обрати цей запис і натиснути пункт меню “Змінити” після цього з’явиться вікно редагування даних про авіарейс.

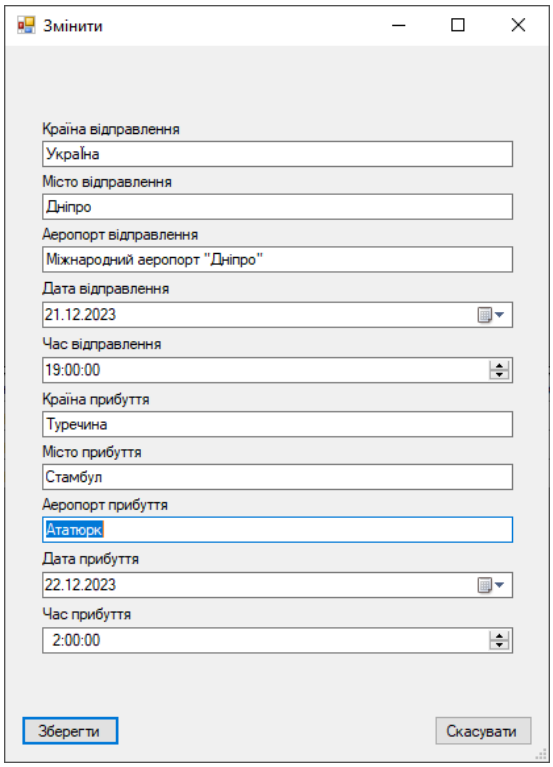


Рис. 3.23. Вікно редагування даних про авіарейс

У вікні редагування даних після внесення будь-яких змін потрібно натиснути кнопку “Зберегти”.

Щоб повернутися до головного меню необхідно натиснути пункт “Повернутися до меню”.

Для редагування даних про договори необхідно спочатку увійти до програми під обліковим записом співробітника.

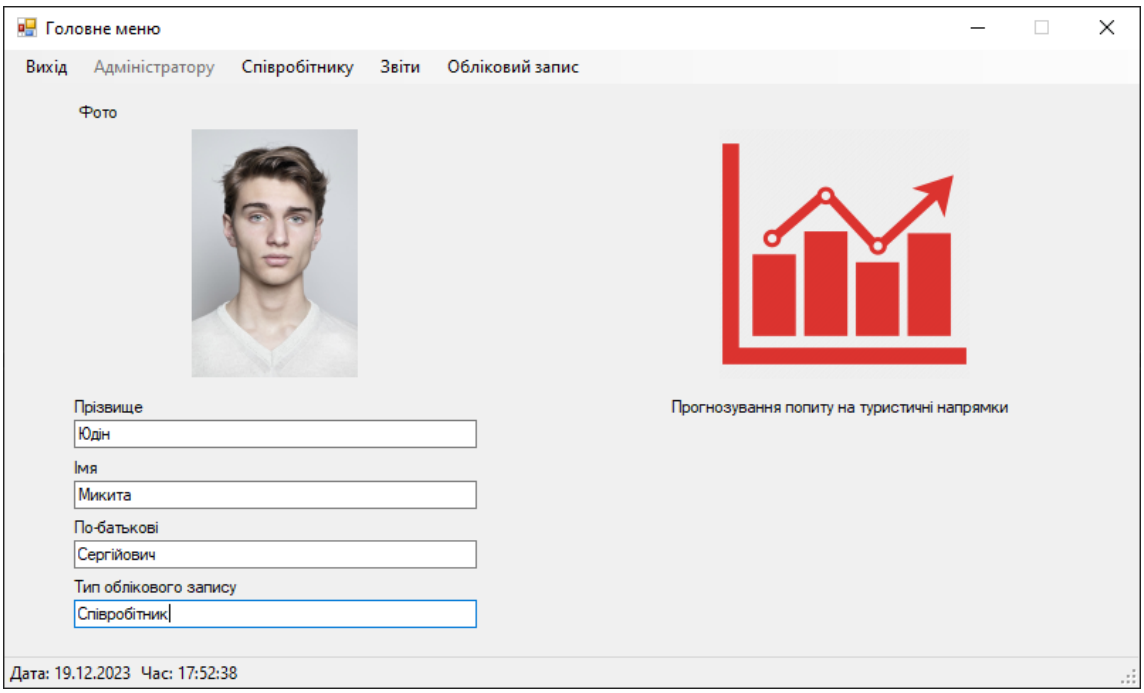


Рис. 3.24. Головне вікно програми в режимі співробітника

Далі в головному вікні програми обрати пункт меню “Співробітнику”, “Договори”. Після цього з’явиться вікно редагування даних про договори. У верхній частині вікна розташоване меню для виконання операцій з даними. На панелі зліва розташовані інструменти для пошуку записів в таблиці. В центрі вікна розташована таблиця з даними.

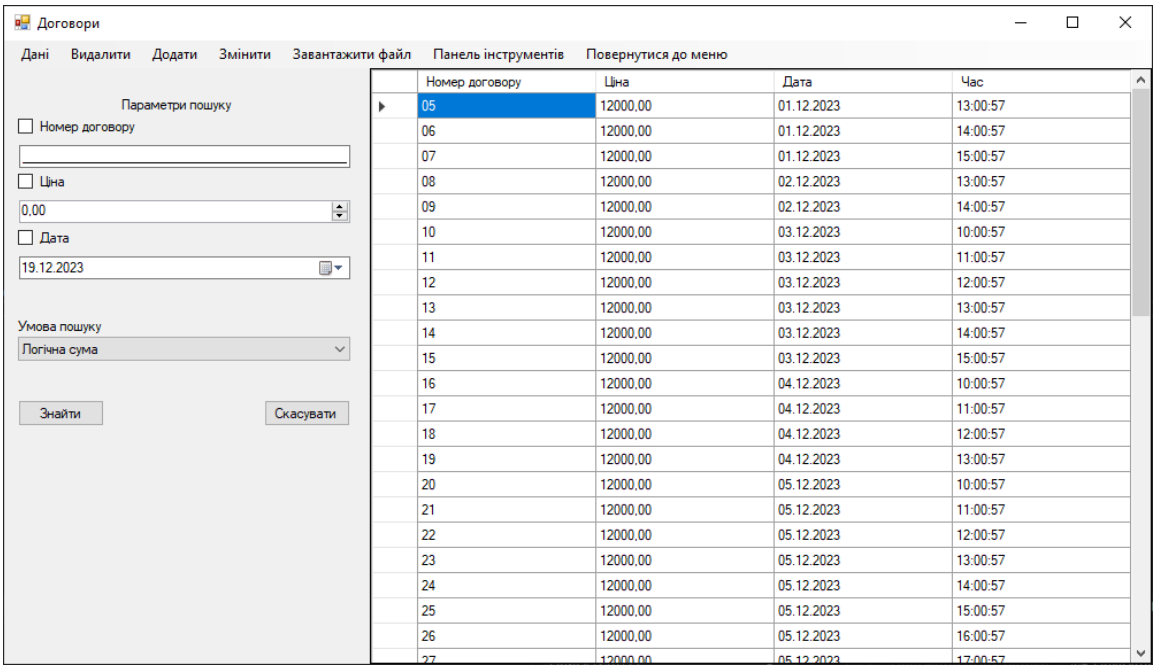
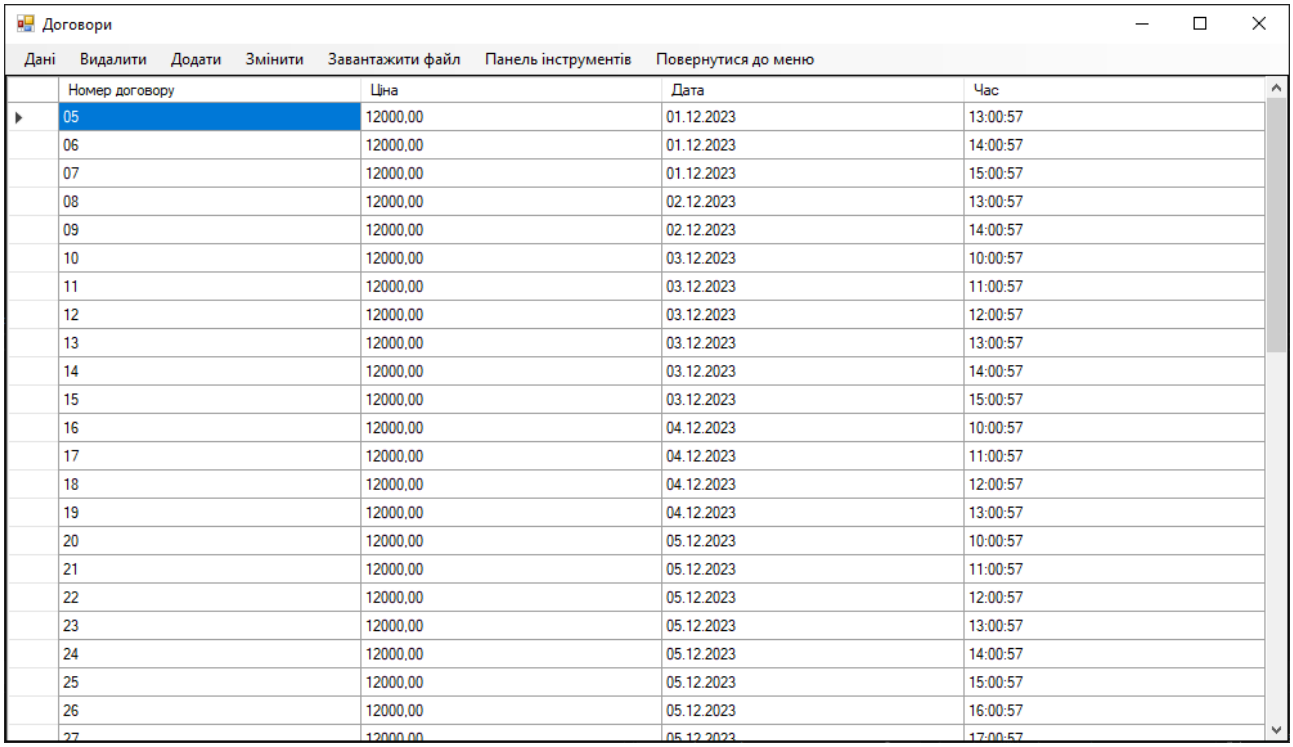


Рис. 3.25. Вікно редагування даних про договори

Щоб виконати пошук даних потрібно обрати поля по яких буде виконуватись пошук, обрати умову пошуку і натиснути кнопку “Знайти”, якщо потрібно скасувати результати пошуку необхідно натиснути кнопку “Скасувати”.

Щоб приховати або відкрити панель інструментів потрібно натиснути пункт меню “Панель інструментів”.



Номер договору	Ціна	Дата	Час
05	12000,00	01.12.2023	13:00:57
06	12000,00	01.12.2023	14:00:57
07	12000,00	01.12.2023	15:00:57
08	12000,00	02.12.2023	13:00:57
09	12000,00	02.12.2023	14:00:57
10	12000,00	03.12.2023	10:00:57
11	12000,00	03.12.2023	11:00:57
12	12000,00	03.12.2023	12:00:57
13	12000,00	03.12.2023	13:00:57
14	12000,00	03.12.2023	14:00:57
15	12000,00	03.12.2023	15:00:57
16	12000,00	04.12.2023	10:00:57
17	12000,00	04.12.2023	11:00:57
18	12000,00	04.12.2023	12:00:57
19	12000,00	04.12.2023	13:00:57
20	12000,00	05.12.2023	10:00:57
21	12000,00	05.12.2023	11:00:57
22	12000,00	05.12.2023	12:00:57
23	12000,00	05.12.2023	13:00:57
24	12000,00	05.12.2023	14:00:57
25	12000,00	05.12.2023	15:00:57
26	12000,00	05.12.2023	16:00:57
27	12000,00	05.12.2023	17:00:57

Рис. 3.26. Вікно редагування даних про договори без панелі інструментів

Для збереження або оновлення даних таблиці потрібно обрати пункт меню “Данні”, “Зберегти” або “Оновити”. Щоб видалити запис з таблиці потрібно натиснути на нього та обрати пункт меню “Видалити”. Для додавання даних до таблиці потрібно натиснути пункт меню “Додати” після цього з’явиться вікно додавання даних про договори.

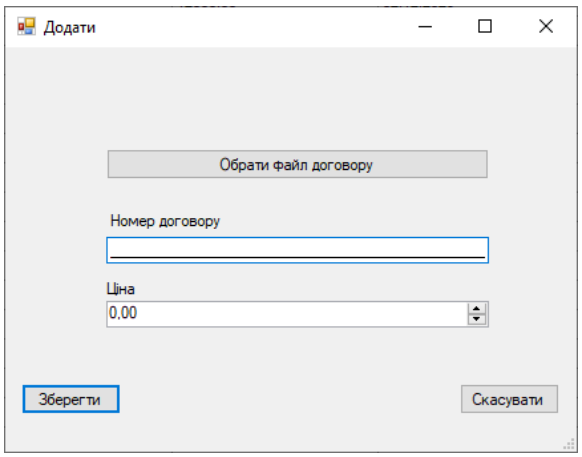


Рис. 3.27. Вікно додавання даних про договори

У вікні додавання даних потрібно заповнити всі текстові поля та обрати файл договору а потім натиснути кнопку “Зберегти”.

Щоб змінити будь-який запис в таблиці договорів потрібно обрати цей запис і натиснути пункт меню “Змінити” після цього з’явиться вікно редагування даних про договір.

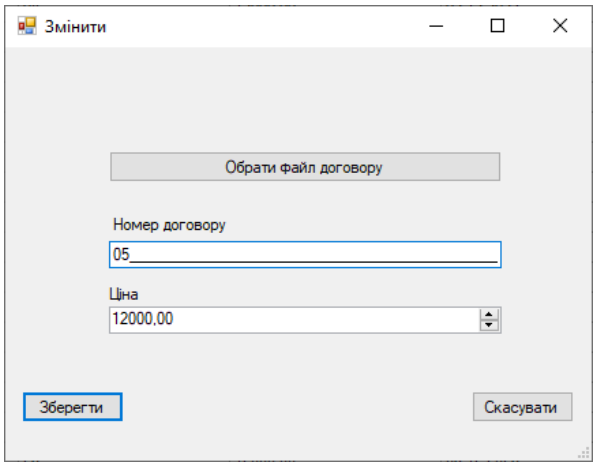


Рис. 3.28. Вікно редагування даних про договір

У вікні редагування даних після внесення будь-яких змін потрібно натиснути кнопку “Зберегти”.

Щоб повернутися до головного меню необхідно натиснути пункт “Повернутися до меню”.

Для встановлення зв’язку між договорами та готелями необхідно в головному вікні програми обрати пункт меню “Співробітнику”, “Договори-Готелі”. Після цього з’явиться вікно для встановлення зв’язку між договорами і готелями. У верхній частині вікна розташоване меню для виконання операцій з даними. На панелі зліва розташовані інструменти для пошуку записів в таблицях. В центрі вікна розташовані таблиці з даними про тури, готелі та таблиця для зв’язку між договорами та готелями, а також зліва знизу кнопки для видалення, додавання, оновлення та зберігання даних.

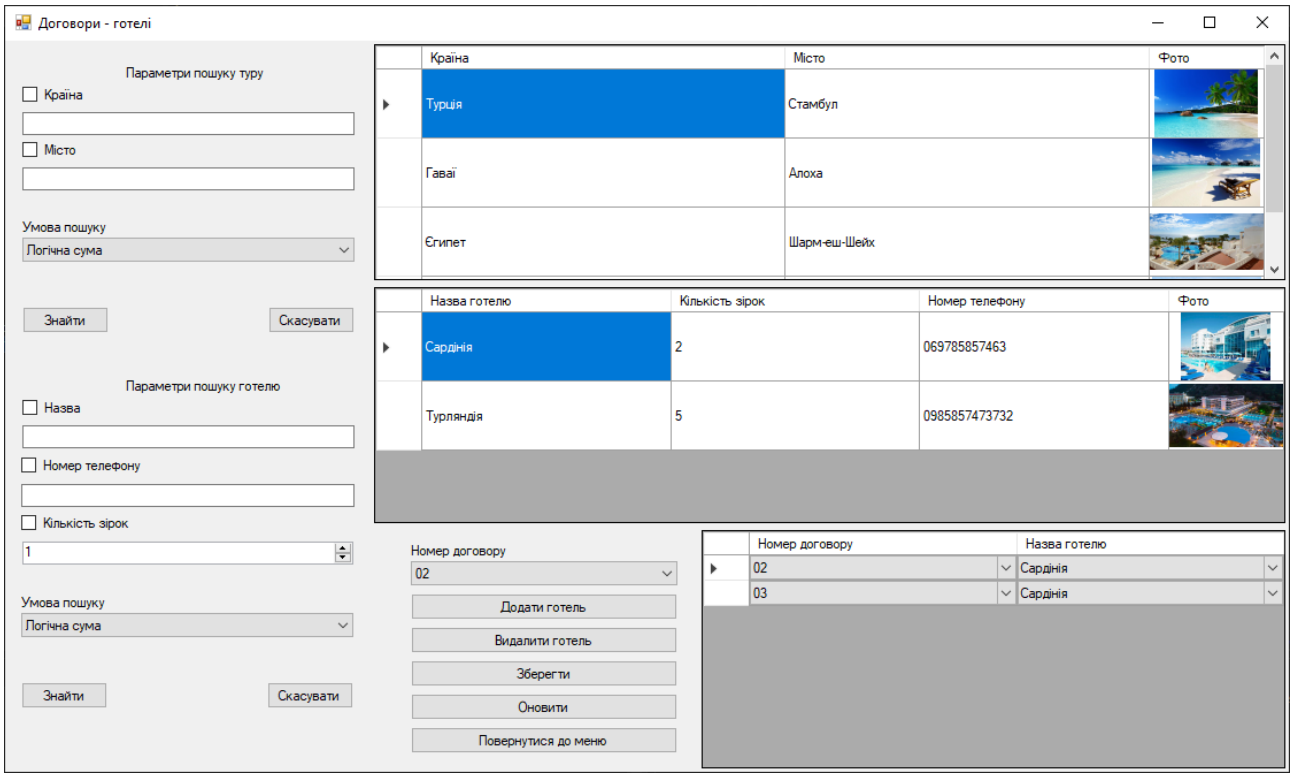


Рис. 3.29. Вікно встановлення зв’язку між договорами та готелями

Для редагування даних про клієнтів необхідно в головному вікні програми обрати пункт меню “Співробітнику”, “Клієнти”. Після цього з’явиться вікно редагування даних про клієнтів. У верхній частині вікна розташоване меню для виконання операцій з даними. На панелі зліва розташовані інструменти для пошуку записів в таблиці. В центрі вікна розташована таблиця з даними.

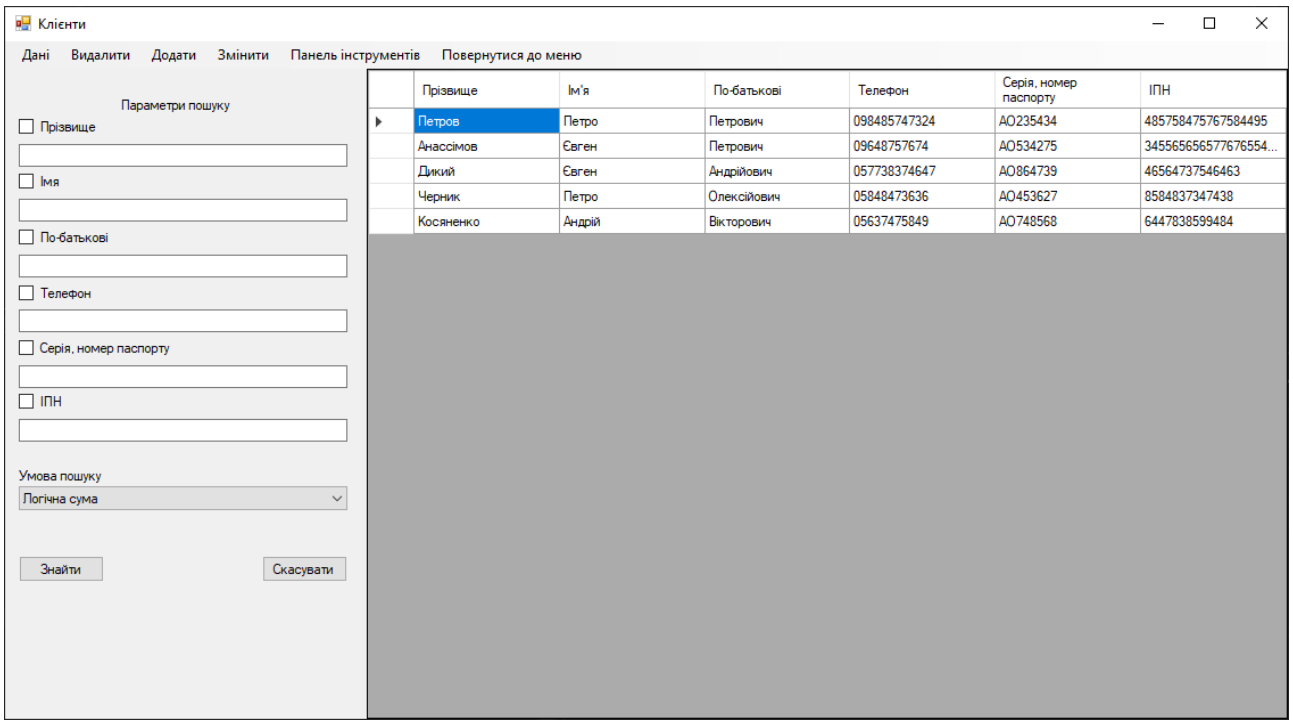


Рис. 3.30. Вікно редагування даних про клієнтів

Щоб виконати пошук даних потрібно обрати поля по яких буде виконуватись пошук, обрати умову пошуку і натиснути кнопку “Знайти”, якщо потрібно скасувати результати пошуку необхідно натиснути кнопку “Скасувати”.

Для того щоб згорнути вікно “Клієнти” потрібно натиснути кнопку у вигляді горизонтальної риски у верхній правій частині вікна.

Для розгортання вікна на весь екран необхідно натиснути кнопку у вигляді квадрата у верхній правій частині вікна.

Для виходу з програми потрібно натиснути кнопку у вигляді хрестика у верхній правій частині вікна.

Щоб приховати або відкрити панель інструментів потрібно натиснути пункт меню “Панель інструментів”.

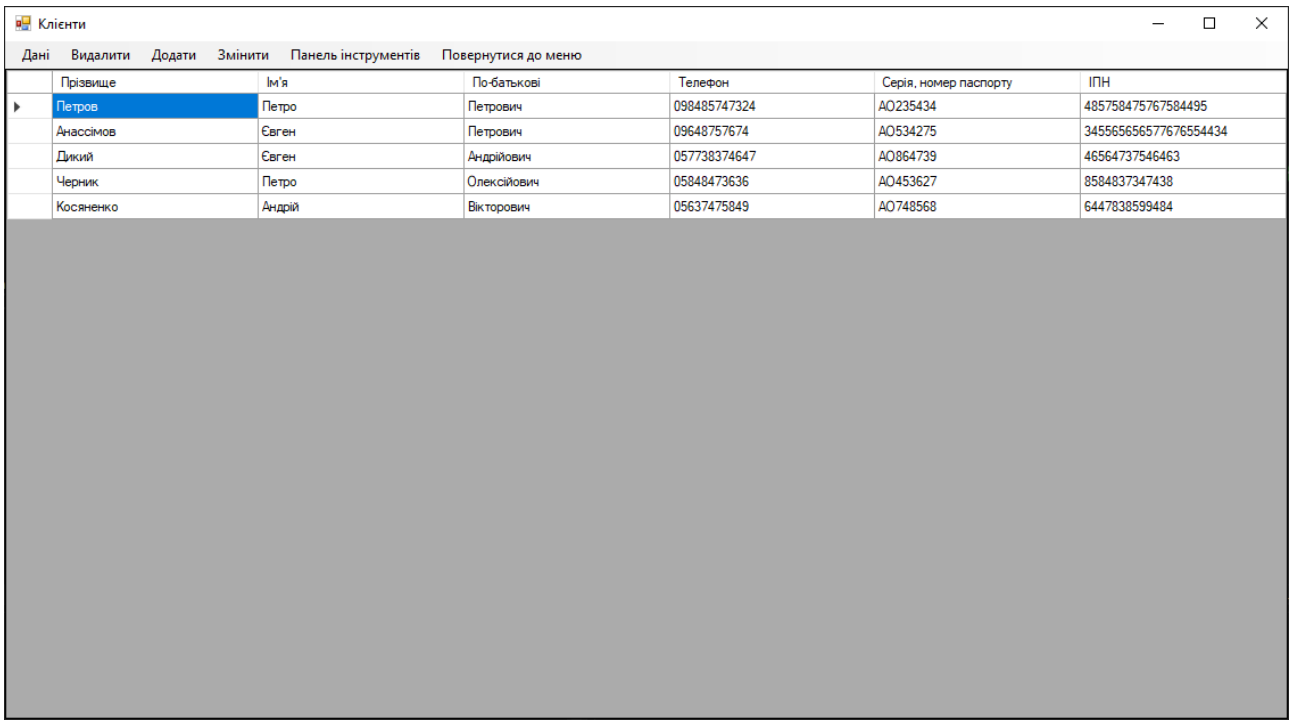


Рис. 3.31. Вікно редагування даних про клієнтів без панелі інструментів

Для збереження або оновлення даних таблиці потрібно обрати пункт меню “Данні”, “Зберегти” або “Оновити”. Щоб видалити запис з таблиці потрібно натиснути на нього та обрати пункт меню “Видалити”. Для додавання даних до таблиці потрібно натиснути пункт меню “Додати” після цього з’явиться вікно додавання даних про договори.

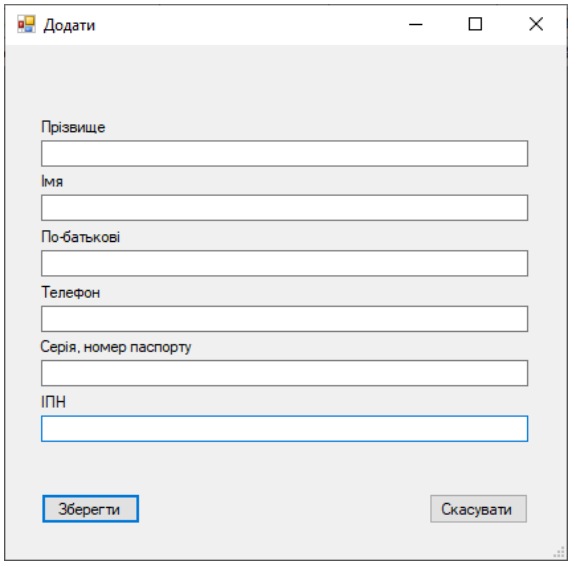


Рис. 3.32. Вікно додавання даних про клієнтів

У вікні додавання даних потрібно заповнити всі текстові поля а потім натиснути кнопку “Зберегти”.

Щоб змінити будь-який запис в таблиці клієнтів потрібно обрати цей запис і натиснути пункт меню “Змінити” після цього з’явиться вікно редагування даних про клієнта.

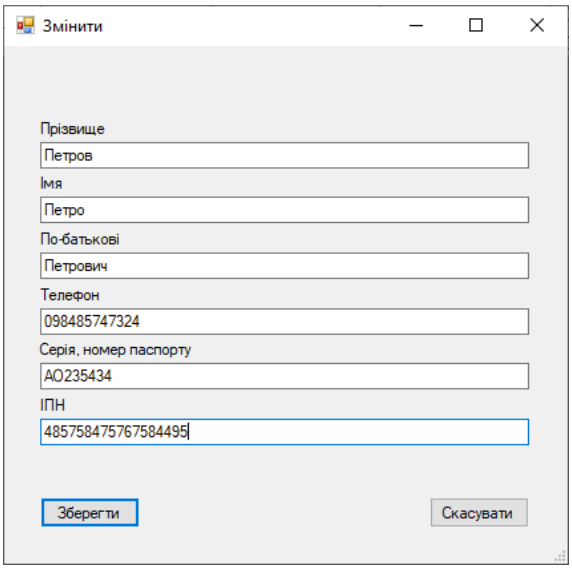


Рис. 3.33. Вікно редагування даних про клієнта

У вікні редагування даних після внесення будь-яких змін потрібно натиснути кнопку “Зберегти”.

Щоб повернутися до головного меню необхідно натиснути пункт “Повернутися до меню”.

Для редагування даних про авіаквитки необхідно в головному вікні програми обрати пункт меню “Співробітнику”, “Квитки”. Після цього з’явиться вікно редагування даних про авіаквитки. В даному вікні на панелі зліва розташовані інструменти для пошуку записів в таблиці. В центрі вікна розташовані таблиці з даними про авіарейси, клієнтів та квитки. В лівій нижній частині вікна розташовані кнопки для виконання операцій з даними.

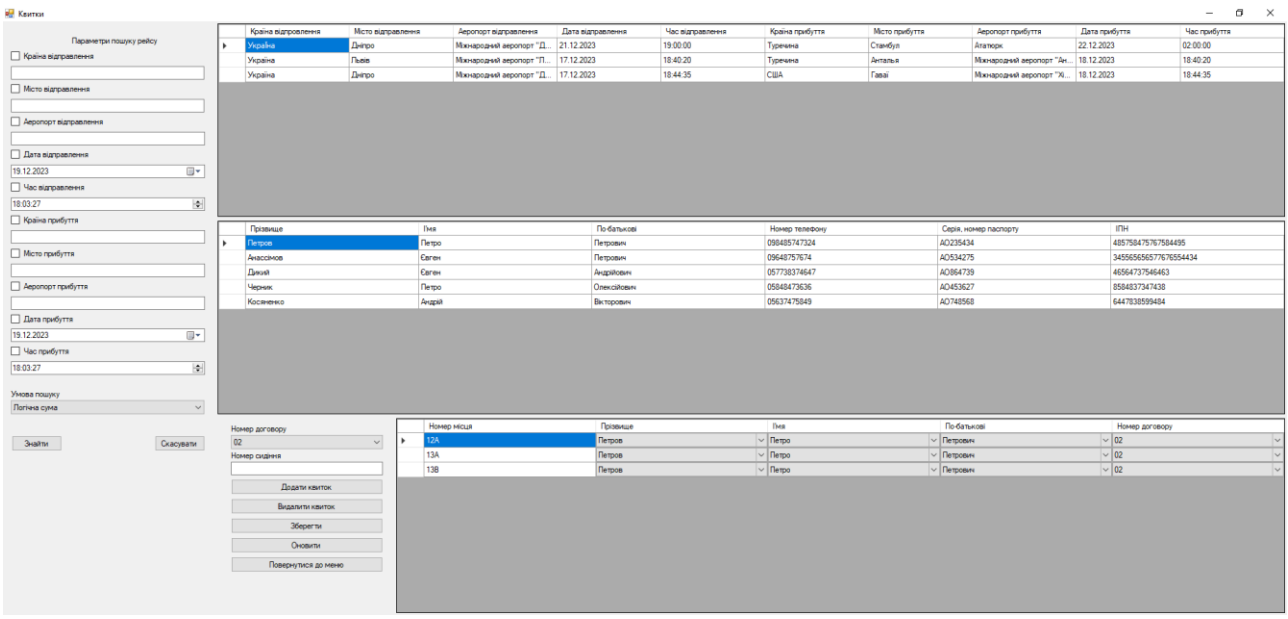


Рис. 3.34. Вікно редагування даних про авіаквитки

Для того щоб згорнути вікно “Квитки” потрібно натиснути кнопку у вигляді горизонтальної риски у верхній правій частині вікна.

Для розгортання вікна на весь екран необхідно натиснути кнопку у вигляді квадрата у верхній правій частині вікна.

Для виходу з програми потрібно натиснути кнопку у вигляді хрестика у верхній правій частині вікна.

Для переходу до вікна перегляду прогнозу попиту на туристичні напрямки необхідно в головному вікні програми натиснути на зображення з написом “Прогнозування попиту на тури”. Після цього відкриється вікно з даними про попит на тури та можливістю подальшого прогнозування попиту. У верхній частині вікна розташована таблиця з даними про тури, в середині вікна розташований графік з даними про попит на обраний тур та лінією прогнозу. В нижній частині вікна розташований список з прогнозованими значеннями. В лівій частині вікна знаходяться інструменти для пошуку турів в таблиці, зміни довжини лінії прогнозу та перевірки статистичної значущості математичної моделі прогнозу.

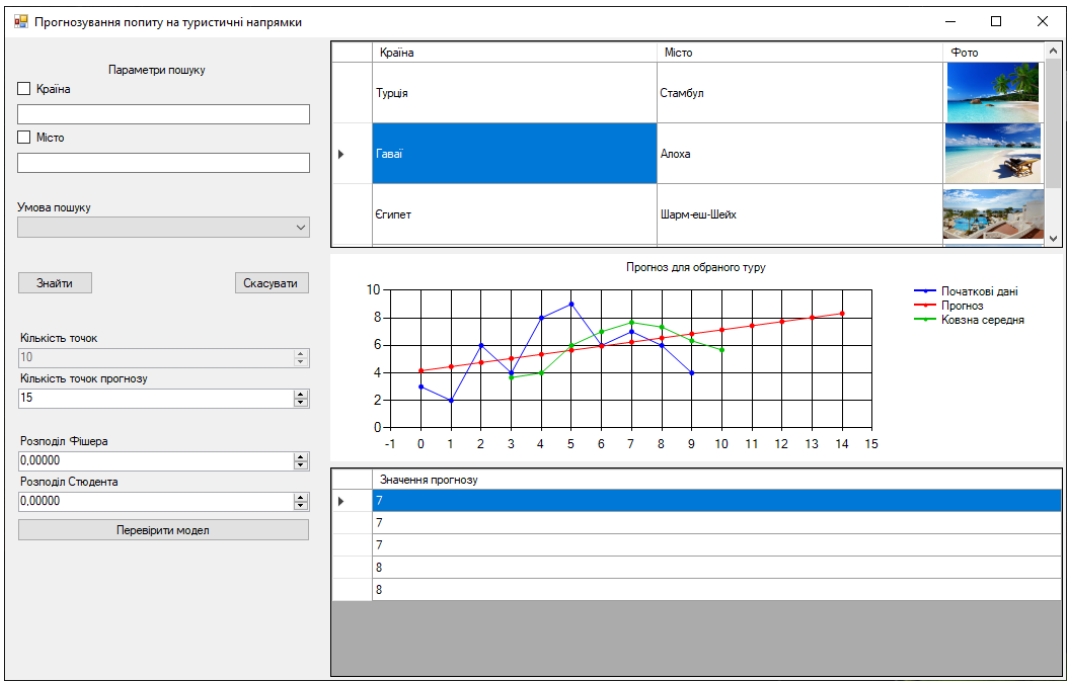


Рис. 3.35. Вікно прогнозування попиту на тури

Для переходу до вікна редагування даних про обліковий запис необхідно в головному вікні програми обрати пункт меню “Обліковий запис”. Після цього відкриється вікно редагування даних облікового запису. У верхній частині вікна розташоване зображення профілю, у нижній частині вікна розташовані поля з даними профілю.

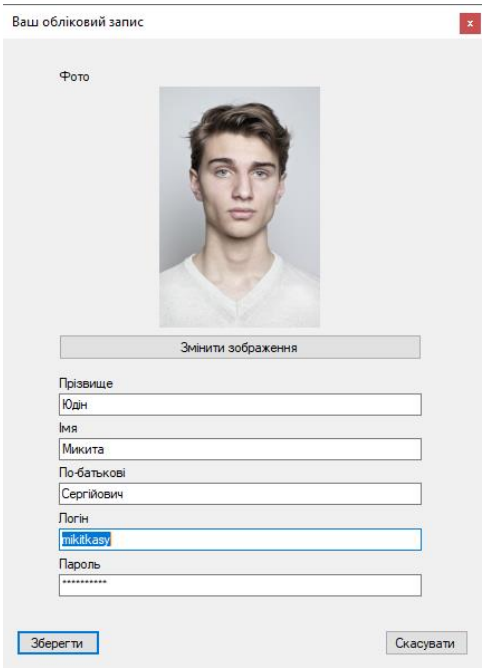


Рис. 3.36. Вікно редагування облікового запису

Після закінчення редагування даних профілю потрібно натиснути кнопку “Зберегти”.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розроблено програмне забезпечення для ведення даних туристичної компанії і розрахунку прогнозу попиту на туристичні напрямки компанії. Було проведено дослідження предметної області та рекомендацій щодо принципів прогнозування часових рядів за результатами досліджень було побудовано та реалізовано обчислення математичної моделі з використанням засобів мови c#.

Інтерфейс користувача програмного додатку створено з використанням технології Windows Forms.

Для підключення до бази даних Microsoft SQL Server використано технологію ADO.NET.

Створений програмний додаток має зручний традиційний інтерфейс користувача який дозволяє користувачеві налаштувати розміри програмних вікон. Програма має механізм реєстрації та авторизації створений з дотриманням сучасних засобів безпеки для збереження результатів обчислень та персоналізації облікового запису користувача.

В кваліфікаційній роботі для дослідження процесу аналізу часових рядів був використаний регресійний аналіз і метод ковзної середньої. Рівняння парної лінійної регресії з невідомими коефіцієнтами було розв’язано за допомогою методу найменших квадратів. Точність кінцевої математичної моделі було перевірено за допомогою коефіцієнту детермінації. Статистичну значущість моделі та коефіцієнтів моделі було перевірено за допомогою F критерію і t критерію відповідно.

Розроблена програма яка призначена для ведення даних туристичної компанії та розрахунку прогнозу попиту на тури компанії і побудови графіку прогнозу. Програма включає в себе дванадцять форм серед яких Form_future в якій знаходяться всі розрахунки. всі модулі програми написані на мові C#.

Розроблене програмне забезпечення дозволяє отримати чисельні результати прогнозування попиту а отже може бути впровадженим у діюче підприємство. Дана робота має сенс на продовження наукового дослідження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. МЕТОДИЧНІ РЕКОМЕНДАЦІЇ до виконання кваліфікаційних робіт здобувачами другого (магістерського) рівня вищої освіти спеціальностей 121 «Інженерія програмного забезпечення» / / Б.І. Мороз, О.В. Іванченко, О.В. Реута, О.С. Шевцова; М-во освіти і науки України, Нац. техн. ун-т “Дніпровська політехніка”. – Дніпро : НТУ «ДП», 2021. – 56 с.

2. esri [Електронний ресурс] //ArcGIS Enterprice. – Режим доступу: https://www.arcgis.com.ua/news-room/fact-sheets (дата звернення: 16.09.2023). – Назва з екрана..

3. Habr.com [Електронний ресурс] // Habr. – Режим доступу: https://www.habr.com/news-room/fact-sheets (дата звернення: 16.09.2023). – Назва з екрана.

4. Kibernetika [Електронний ресурс] // International Math Federation. – Режим доступу: https://kibernetika.org/about/facts-figures/ (дата звернення: 17.09.2023). – Назва з екрана.

5. MindTheGraph [Електронний ресурс] // Centers for Control and Regression. – Режим доступу: https://www. mindthegraph.com/blog/ru (дата звернення: 17.09.2023). – Назва з екрана.

6. Про регресійний аналіз [Електронний ресурс] // Все про регресію. – Режим доступу: https://pzs.dstu.dp.ua/datamining/ (дата звернення: 18.09.2023). – Назва з екрана.

7. MVC (Model View Controller) Architecture Pattern in Android - GeeksforGeeks [Електронний ресурс] // GeeksforGeeks. – Режим доступу: https://www.geeksforgeeks.org/mvvm-model-view-controller-architecture-pattern-in-android/ (дата звернення: 11.10.2023). – Назва з екрана.

8. What Is MVC Architecture? [Електронний ресурс] // Built In. – Режим доступу: https://builtin.com/software-engineering-perspectives/mvc-architecture (дата звернення: 11.10.2023). – Назва з екрана.

9. MVC: Model-View-Controller Architecture | Ramotion Branding Agency [Електронний ресурс] // Web Design, UI/UX, Branding, and App Development Blog. – Режим доступу: https://www.ramotion.com/blog/what-is-mvc/#section-advantages-of-mvc (дата звернення: 13.10.2023). – Назва з екрана.

10. Bondar S. Working with Windows Presentation Foundation | Reintech media [Електронний ресурс] / Sasha Bondar // Software Developers as a Service | Reintech. – Режим доступу: https://reintech.io/blog/tutorial-introduction-working-windows-presentation-foundation (дата звернення: 13.10.2023). – Назва з екрана.

11. WPF vs. WinForms - The complete WinForms tutorial [Електронний ресурс] // Welcome - The complete WinForms tutorial. – Режим доступу: https://winforms-tutorial.com/about-wpf/wpf-vs-winforms/ (дата звернення: 15.10.2023). – Назва з екрана.

12. Karia R. ADO.NET [Електронний ресурс] / Ravi Karia // Entity Framework Tutorial. – Режим доступу: https://www.ado.net/efcore/entity-framework-core.aspx (дата звернення: 15.10.2023). – Назва з екрана.

13. Overview of Entity Framework Core - EF Core [Електронний ресурс] // Microsoft Learn: Build skills that open doors in your career. – Режим доступу: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 15.10.2023). – Назва з екрана.

14. Language Integrated Query (LINQ) in C# - C# [Електронний ресурс] // Microsoft Learn: Build skills that open doors in your career. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/linq/ (дата звернення: 16.10.2023). – Назва з екрана.

15. SQL Server: About [Електронний ресурс] // SQL Server: The world’s most advanced open source database. – Режим доступу: https://www.sqlserver.org/about/ (дата звернення: 17.10.2023). – Назва з екрана.

16. Why use SQL Server as a Database for my Next Project in 2022 - Fulcrum [Електронний ресурс] // Fulcrum. – Режим доступу: https://fulcrum.rocks/blog/why-use-sqlserver-database#mongodb-vs-postgresql-1 (дата звернення: 17.10.2023). – Назва з екрана.

17. SQL Server Advantages: Benefits of Using SQLServer [Електронний ресурс] // Prisma’s Data Guide. – Режим доступу: https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql (дата звернення: 18.10.2023). – Назва з екрана.

18. A tour of C# - Overview - C# [Електронний ресурс] // Microsoft Learn: Build skills that open doors in your career. – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ (дата звернення: 20.10.2023). – Назва з екрана.

19. What is C# used for? [Електронний ресурс] // Stackify. – Режим доступу: https://stackify.com/what-is-c-used-for/ (дата звернення: 20.10.2023). – Назва з екрана.

20. Angella A. 15 reasons why you should learn C# in 2023 [Електронний ресурс] / Andrea Angella // LinkedIn: Log In or Sign Up. – Режим доступу: https://www.linkedin.com/pulse/15-reasons-why-you-should-learn-c-2023-andrea-angella (дата звернення: 25.10.2023). – Назва з екрана.

21. Visual Studio: IDE and Code Editor for Software Developers and Teams [Електронний ресурс] // Visual Studio. – Режим доступу: https://visualstudio.microsoft.com/ (дата звернення: 25.10.2023). – Назва з екрана.

ДОДАТОК А

КОД БАЗИ ДАНИХ

```
create database diploma

use diploma

create table admin
(
id int not null primary key identity(1,1),
surname nvarchar(100) not null,
adminName nvarchar(100) not null,
patronymic nvarchar(100) not null,
userName nvarchar(100) not null,
userPassword nvarchar(100) not null,
phone nvarchar(100) not null,
passport nvarchar(100) not null,
street nvarchar(100) not null,
house int not null,
korpuse nvarchar(100) not null,
apartment int not null,
photo varbinary(max) not null,
)

create table emploee
(
id int not null primary key identity(1,1),
surname nvarchar(100) not null,
emploeeName nvarchar(100) not null,
patronymic nvarchar(100) not null,
userName nvarchar(100) not null,
userPassword nvarchar(100) not null,
phone nvarchar(100) not null,
passport nvarchar(100) not null,
street nvarchar(100) not null,
house int not null,
korpuse nvarchar(100) not null,
apartment int not null,
photo varbinary(max) not null,

adminID int not null foreign key references admin(id) on update cascade on delete cascade,
)

create table document
(
id int not null primary key identity(1,1),
number nvarchar(100) not null,
price numeric(30,2) not null,
documentDate date default convert(date, getdate()) not null,
documentTime time(0) default convert(time, getdate()) not null,
documentFile varbinary(max) not null,

emploeeID int not null foreign key references emploee(id) on update cascade on delete cascade,
)

create table tour
(
id int not null primary key identity(1,1),
country nvarchar(100) not null,
```

```
city nvarchar(100) not null,
photo varbinary(max) not null,
)

create table hotel
(
id int not null primary key identity(1,1),
hotelName nvarchar(100) not null,
stars int not null,
phone nvarchar(100)not null,
photo varbinary(max) not null,

tourID int not null foreign key references tour(id) on update cascade on delete cascade,
)

create table document_hotel
(
documentID int not null foreign key references document(id) on update cascade on delete cascade,
hotelID int not null foreign key references hotel(id) on update cascade on delete cascade,
primary key(documentID, hotelID),
)

create table airline
(
id int not null primary key identity(1,1),
airlineName nvarchar(100) not null,
phone nvarchar(100) not null,
photo varbinary(max) not null,
)

create table flight
(
id int not null primary key identity(1,1),
countryStart nvarchar(100) not null,
cityStart nvarchar(100) not null,
airportStart nvarchar(100),
dateStart date not null,
timeStart time(0) not null,
countryFinish nvarchar(100) not null,
cityFinish nvarchar(100) not null,
airportFinish nvarchar(100),
dateFinish date not null,
timeFinish time(0) not null,

airlineID int not null foreign key references airline(id) on update cascade on delete cascade,
)

create table client
(
id int not null primary key identity(1,1),
surname nvarchar(100) not null,
clientName nvarchar(100) not null,
patronymic nvarchar(100) not null,
phone nvarchar(100) not null,
passport nvarchar(100) not null unique,
inn nvarchar(100) not null unique,
)

create table ticket
(
id int not null primary key identity(1,1),
number nvarchar(100) not null,
```

```
flightID int not null foreign key references flight(id) on update cascade on delete cascade,
clientID int not null foreign key references client(id) on update cascade on delete cascade,
documentID int not null foreign key references document(id) on update cascade on delete cascade,
constraint uc_ticket unique(flightID, clientID, documentID, number)
)
```

ДОДАТОК Б

КОД ПРОГРАМИ

Форма Form_account_admin

```
namespace TourAnalysis
{
   public partial class Form_account_admin : Form
   {
      private byte[] photo;
      public Form_account_admin()
      {
         InitializeComponent();
      }

      public string Surname
      {
         get
         {
            return this.textBox_surname.Text;
         }
         set
         {
            this.textBox_surname.Text = value;
         }
      }
      public string adminName
      {
         get
         {
            return this.textBox_name.Text;
         }
         set
         {
            this.textBox_name.Text = value;
         }
      }
      public string Patronymic
      {
         get
         {
            return this.textBox_patronymic.Text;
         }
         set
         {
            this.textBox_patronymic.Text = value;
         }
      }

      public string Phone
      {
         get
         {
            return this.maskedTextBox_phone.Text;
         }
         set
         {
            this.maskedTextBox_phone.Text = value;
         }
```

```
}

public string Passport
{
   get
   {
      return this.textBox_passport.Text;
   }
   set
   {
      this.textBox_passport.Text = value;
   }
}
public string Login
{
   get
   {
      return this.textBox_login.Text;
   }
   set
   {
      this.textBox_login.Text = value;
   }
}
public string Password
{
   get
   {
      return this.maskedTextBox_password.Text;
   }
   set
   {
      this.maskedTextBox_password.Text = value;
   }
}
public string Street
{
   get
   {
      return this.textBox_street.Text;
   }
   set
   {
      this.textBox_street.Text = value;
   }
}

public int House
{
   get
   {
      return (int)this.numericUpDown_house.Value;
   }
   set
   {
      this.numericUpDown_house.Value = value;
   }
}

public string Korpuse
{
   get
   {
```

```
            return this.textBox_korpuse.Text;
        }
        set
        {
            this.textBox_korpuse.Text = value;
        }
    }

    public int Apartment
    {
        get
        {
            return (int)this.numericUpDown_apartment.Value;
        }
        set
        {
            this.numericUpDown_apartment.Value = value;
        }
    }


    public byte[] Photo
    {
        get
        {
            if (File.Exists(this.openFileDialog_photo.FileName)) return
File.ReadAllBytes(this.openFileDialog_photo.FileName);
            return this.photo;
        }
        set
        {
            this.photo = value;
            MemoryStream stmBLOBData = new MemoryStream(this.photo);
            this.pictureBox_photo.Image = Image.FromStream(stmBLOBData);
        }
    }

    private void button_photo_change_Click(object sender, EventArgs e)
    {
        this.openFileDialog_photo.ShowDialog();
        this.pictureBox_photo.Image = Image.FromFile(this.openFileDialog_photo.FileName);
    }
  }
}
```

Форма Form_account_emploee

```
namespace TourAnalysis
{
  public partial class Form_account_emploee : Form
  {
    private byte[] photo;
    public Form_account_emploee()
    {
      InitializeComponent();
    }

    public string Surname
    {
      get
      {
        return this.textBox_surname.Text;
      }
      set
```

```
        {
            this.textBox_surname.Text = value;
        }
    }
    public string emploeeName
    {
        get
        {
            return this.textBox_name.Text;
        }
        set
        {
            this.textBox_name.Text = value;
        }
    }
    public string Patronymic
    {
        get
        {
            return this.textBox_patronymic.Text;
        }
        set
        {
            this.textBox_patronymic.Text = value;
        }
    }
    public string Login
    {
        get
        {
            return this.textBox_login.Text;
        }
        set
        {
            this.textBox_login.Text = value;
        }
    }
    public string Password
    {
        get
        {
            return this.maskedTextBox_password.Text;
        }
        set
        {
            this.maskedTextBox_password.Text = value;
        }
    }

    public byte[] Photo
    {
        get
        {
            if (File.Exists(this.openFileDialog_photo.FileName)) return
File.ReadAllBytes(this.openFileDialog_photo.FileName);
            return this.photo;
        }
        set
        {
            this.photo = value;
            MemoryStream stmBLOBData = new MemoryStream(this.photo);
            this.pictureBox_photo.Image = Image.FromStream(stmBLOBData);
        }
```

```
        }

        private void button_photo_change_Click(object sender, EventArgs e)
        {
            this.openFileDialog_photo.ShowDialog();
            this.pictureBox_photo.Image = Image.FromFile(this.openFileDialog_photo.FileName);
        }
    }
}
```

Форма Form_airline

```
namespace TourAnalysis
{
    public partial class Form_airline : Form
    {
        public Form_airline()
        {
            InitializeComponent();

            DataTable TableSort = new DataTable("TableSearch");
            DataColumn value = new DataColumn("value", typeof(System.String));
            DataColumn text = new DataColumn("text", typeof(System.String));
            TableSort.Columns.AddRange(new DataColumn[] { value, text });
            TableSort.PrimaryKey = new DataColumn[] { value };

            DataRow up = TableSort.NewRow();
            up[0] = "or";
            up[1] = "Логічна сума";
            TableSort.Rows.Add(up);

            DataRow down = TableSort.NewRow();
            down[0] = "and";
            down[1] = "Логічне множення";
            TableSort.Rows.Add(down);

            this.comboBox_if_airline.DataSource = TableSort;
            this.comboBox_if_airline.ValueMember = "value";
            this.comboBox_if_airline.DisplayMember = "text";
            this.comboBox_if_airline.SelectedIndex = 0;
        }

        private void airlineDelete()
        {
            if (this.airlineBindingSource.Current != null)
            {
                ((DataRowView)this.airlineBindingSource.Current).Row.Delete();
                this.airlineBindingSource.EndEdit();
            }
        }

        private void airlineAdd()
        {
            Form_airline_add form = new Form_airline_add("Додати");
            form.Photo = File.ReadAllBytes("question.png");
            if (form.ShowDialog() == DialogResult.OK)
            {
                airlineDataSet.airlineRow                                    row                                    =
(airlineDataSet.airlineRow)((DataRowView)this.airlineBindingSource.AddNew()).Row;

                row.airlineName = form.airlineName;
                row.phone = form.Phone;
                row.photo = form.Photo;
```

```
            this.airlineBindingSource.EndEdit();
        }
    }

    private void airlineChange()
    {
        if (this.airlineBindingSource.Current != null)
        {
            airlineDataSet.airlineRow row =
(airlineDataSet.airlineRow)((DataRowView)this.airlineBindingSource.Current).Row;
            Form_airline_add form = new Form_airline_add("Змінити");

            form.airlineName = row.airlineName;
            form.Phone = row.phone;
            form.Photo = row.photo;

            if (form.ShowDialog() == DialogResult.OK)
            {
                row.airlineName = form.airlineName;
                row.phone = form.Phone;
                row.photo = form.Photo;

                this.airlineBindingSource.EndEdit();
            }
        }

    }


    private void Form_airline_Load(object sender, EventArgs e)
    {
        this.airlineTableAdapter.Fill(this.airlineDataSet.airline);
    }

    private void оновитиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.airlineTableAdapter.Fill(this.airlineDataSet.airline);
    }

    private void зберегтиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.airlineTableAdapter.Update(this.airlineDataSet.airline);
    }

    private void видалитиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.airlineDelete();
    }

    private void додатиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.airlineAdd();
    }

    private void змінитиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.airlineChange();
    }

    private void панельІнструментівToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.splitContainer1.Panel1Collapsed = !this.splitContainer1.Panel1Collapsed;
```

```
        }

        private void airlineBindingSource_ListChanged(object sender, ListChangedEventArgs e)
        {
            foreach (DataGridViewRow row in this.dataGridView1.Rows)
            {
                row.Height = 60;
            }
        }

        private void button_search_Click(object sender, EventArgs e)
        {
            string strSeparator = string.Format(" {0} ", this.comboBox_if_airline.SelectedValue.ToString());
            bool[] ArrayChecked = new bool[] { this.checkBox_airlineName.Checked, this.checkBox_phone.Checked };
            string[] ArrayValues = new string[2];
            ArrayValues[0] = string.Format("airlineName='{0}'", this.textBox_airlineName.Text);
            ArrayValues[1] = string.Format("phone='{0}'", this.textBox_phone.Text);

            List<string> ArrayFilters = new List<string>();

            for (int i = 0; i < ArrayChecked.Length; i++)
            {
                if (ArrayChecked[i]) ArrayFilters.Add(ArrayValues[i]);
            }

            StringBuilder sb = new StringBuilder();
            int Count = ArrayFilters.Count;
            for (int i = 0; i < Count; i++)
            {
                sb.Append(ArrayFilters[i]);
                if (i < Count - 1) sb.Append(strSeparator);
            }

            this.airlineBindingSource.Filter = sb.ToString();
        }

        private void button_cancel_Click(object sender, EventArgs e)
        {
            this.airlineBindingSource.Filter = string.Empty;
        }

        private void повернутисяДоМенюToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.OK;
        }
    }
}
```

Форма Form_airline_add

```
namespace TourAnalysis
{
    public partial class Form_airline_add : Form
    {
        private byte[] photo;
        public Form_airline_add(string title)
        {
            InitializeComponent();
            this.Text = title;
        }


        public string airlineName
```

```
        {
            get
            {
                return this.textBox_airlineName.Text;
            }
            set
            {
                this.textBox_airlineName.Text = value;
            }
        }
        public string Phone
        {
            get
            {
                return this.textBox_phone.Text;
            }
            set
            {
                this.textBox_phone.Text = value;
            }
        }
        public byte[] Photo
        {
            get
            {
                if (File.Exists(this.openFileDialog_photo.FileName)) return
File.ReadAllBytes(this.openFileDialog_photo.FileName);
                return this.photo;
            }
            set
            {
                this.photo = value;
                MemoryStream stmBLOBData = new MemoryStream(this.photo);
                this.pictureBox_photo.Image = Image.FromStream(stmBLOBData);
            }
        }

        private void button_photo_change_Click(object sender, EventArgs e)
        {
            if (this.openFileDialog_photo.ShowDialog() == DialogResult.OK &&
File.Exists(this.openFileDialog_photo.FileName))
                this.pictureBox_photo.Image = Image.FromFile(this.openFileDialog_photo.FileName);
        }
    }
}
```

Форма Form_authorization

```
namespace TourAnalysis
{
    public partial class Form_authorization : Form
    {
        DataTable accounts;
        public Form_authorization()
        {
            InitializeComponent();

            DataColumn name = new DataColumn("name", typeof(int));
            DataColumn value = new DataColumn("value", typeof(string));

            accounts = new DataTable();
            accounts.Columns.AddRange(new DataColumn[] { name, value });
```

```
            DataRow admin = accounts.NewRow();
            admin["name"] = 1;
            admin["value"] = "Адміністратор";
            accounts.Rows.Add(admin);

            DataRow emploee = accounts.NewRow();
            emploee["name"] = 2;
            emploee["value"] = "Співробітник";
            accounts.Rows.Add(emploee);

            this.StartPosition = FormStartPosition.CenterScreen;
            this.Text = "Вхід";

        }

        private void Form_authorization_Load(object sender, EventArgs e)
        {
            this.comboBox_accounts.DataSource = accounts;
            this.comboBox_accounts.ValueMember = "name";
            this.comboBox_accounts.DisplayMember = "value";
            this.comboBox_accounts.SelectedIndex = 0;
            this.comboBox_accounts.DropDownStyle = ComboBoxStyle.DropDownList;

        }

        private void button_authorization_Click(object sender, EventArgs e)
        {
            string diplomaConnectionString = ConfigurationManager.ConnectionStrings["TourAnalysis.Properties.Settings.diplomaConnectionString"].ConnectionString;
            string queryString;

            if ((int)comboBox_accounts.SelectedValue == 1)
            {
                queryString = String.Format("select * from {0} where userName = '{1}' and userPassword = '{2}'", "admin", textBox_login.Text, textBox_password.Text);
            }
            else
            {
                queryString = String.Format("select * from {0} where userName = '{1}' and userPassword = '{2}'", "emploee", textBox_login.Text, textBox_password.Text);
            }


            using (SqlConnection connection = new SqlConnection(
              diplomaConnectionString))
            {
                SqlCommand command = new SqlCommand(queryString, connection);
                command.Connection.Open();

                SqlDataReader resalt = command.ExecuteReader();
                DataRowView SelectedRow = (DataRowView)this.comboBox_accounts.SelectedItem;
                if (!resalt.HasRows) MessageBox.Show("Не вірний логін або пароль!", "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                else
                {
                    resalt.Read();
                    int UserID = resalt.GetInt32(0);
                    resalt.Close();
                    this.Hide();
                    Form_main form = new Form_main(UserID, (int)comboBox_accounts.SelectedValue);
                    form.Show();
                }
```

```
        }
    }

    private void button_exit_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void checkBox_show_password_CheckedChanged(object sender, EventArgs e)
    {
        CheckBox tmp = (CheckBox)sender;
        if (tmp.Checked) this.textBox_password.PasswordChar = '\0';
        else this.textBox_password.PasswordChar = 'x';
    }
  }
}
```

Форма Form_client

```
namespace TourAnalysis
{
  public partial class Form_client : Form
  {
    public Form_client()
    {
      InitializeComponent();

      DataTable TableSort = new DataTable("TableSearch");
      DataColumn value = new DataColumn("value", typeof(System.String));
      DataColumn text = new DataColumn("text", typeof(System.String));
      TableSort.Columns.AddRange(new DataColumn[] { value, text });
      TableSort.PrimaryKey = new DataColumn[] { value };

      DataRow up = TableSort.NewRow();
      up[0] = "or";
      up[1] = "Логічна сума";
      TableSort.Rows.Add(up);

      DataRow down = TableSort.NewRow();
      down[0] = "and";
      down[1] = "Логічне множення";
      TableSort.Rows.Add(down);

      this.comboBox_if.DataSource = TableSort;
      this.comboBox_if.ValueMember = "value";
      this.comboBox_if.DisplayMember = "text";
      this.comboBox_if.SelectedIndex = 0;
    }

    private void Form_client_Load(object sender, EventArgs e)
    {
      this.clientTableAdapter.Fill(this.clientDataSet.client);
    }

    private void clientDelete()
    {
      if (this.clientBindingSource.Current != null)
      {
        ((DataRowView)this.clientBindingSource.Current).Row.Delete();
        this.clientBindingSource.EndEdit();
      }
    }
```

```
        private void clientAdd()
        {
            Form_client_add form = new Form_client_add("Додати");
            if (form.ShowDialog() == DialogResult.OK)
            {
                clientDataSet.clientRow row =
(clientDataSet.clientRow)((DataRowView)this.clientBindingSource.AddNew()).Row;

                row.surname = form.Surname;
                row.clientName = form.clientName;
                row.patronymic = form.Patronymic;
                row.phone = form.Phone;
                row.passport = form.Passport;
                row.inn = form.INN;

                this.clientBindingSource.EndEdit();
            }
        }

        private void clientChange()
        {
            if (this.clientBindingSource.Current != null)
            {
                clientDataSet.clientRow row =
(clientDataSet.clientRow)((DataRowView)this.clientBindingSource.Current).Row;
                Form_client_add form = new Form_client_add("Змінити");

                form.Surname = row.surname;
                form.clientName = row.clientName;
                form.Patronymic = row.patronymic;
                form.Phone = row.phone;
                form.Passport = row.passport;
                form.INN = row.inn;

                if (form.ShowDialog() == DialogResult.OK)
                {
                    row.surname = form.Surname;
                    row.clientName = form.clientName;
                    row.patronymic = form.Patronymic;
                    row.phone = form.Phone;
                    row.passport = form.Passport;
                    row.inn = form.INN;

                    this.clientBindingSource.EndEdit();
                }
            }

        }

        private void зберегтиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.clientTableAdapter.Update(this.clientDataSet.client);
        }

        private void оновитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.clientTableAdapter.Fill(this.clientDataSet.client);
        }

        private void видалитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.clientDelete();
        }
```

```
        private void додатиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.clientAdd();
        }

        private void змінитиToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.clientChange();
        }

        private void панельІнструментівToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.splitContainer1.Panel1Collapsed = !this.splitContainer1.Panel1Collapsed;
        }

        private void button_search_Click(object sender, EventArgs e)
        {
            string strSeparator = string.Format(" {0} ", this.comboBox_if.SelectedValue.ToString());
            bool[] ArrayChecked = new bool[] { this.checkBox_surname.Checked, this.checkBox_name.Checked,
this.checkBox_patronymic.Checked, this.checkBox_phone.Checked, this.checkBox_passport.Checked,
this.checkBox_inn.Checked };
            string[] ArrayValues = new string[7];
            ArrayValues[0] = string.Format("surname='{0}'", this.textBox_surname.Text);
            ArrayValues[1] = string.Format("clientName='{0}'", this.textBox_name.Text);
            ArrayValues[2] = string.Format("patronymic='{0}'", this.textBox_patronymic.Text);
            ArrayValues[3] = string.Format("phone='{0}'", this.maskedTextBox_phone.Text);
            ArrayValues[4] = string.Format("passport='{0}'", this.textBox_passport.Text);
            ArrayValues[5] = string.Format("inn={0}", this.textBox_inn.Text);

            List<string> ArrayFilters = new List<string>();

            for (int i = 0; i < ArrayChecked.Length; i++)
            {
                if (ArrayChecked[i]) ArrayFilters.Add(ArrayValues[i]);
            }

            StringBuilder sb = new StringBuilder();
            int Count = ArrayFilters.Count;
            for (int i = 0; i < Count; i++)
            {
                sb.Append(ArrayFilters[i]);
                if (i < Count - 1) sb.Append(strSeparator);
            }

            this.clientBindingSource.Filter = sb.ToString();

        }

        private void button_cancel_Click(object sender, EventArgs e)
        {
            this.clientBindingSource.Filter = string.Empty;
        }

        private void повернутисяДоМенюToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.OK;
        }
    }
}
```

Форма Form_client_add

```
namespace TourAnalysis
{
   public partial class Form_client_add : Form
   {
      public Form_client_add(string title)
      {
         InitializeComponent();
         this.Text = title;
      }

      public string Surname
      {
         get { return this.textBox_surname.Text; }
         set { this.textBox_surname.Text = value; }
      }
      public string clientName
      {
         get {return this.textBox_name.Text; }
         set { this.textBox_name.Text = value; }
      }
      public string Patronymic
      {
         get { return this.textBox_patronymic.Text; }
         set { this.textBox_patronymic.Text = value; }
      }
      public string Phone
      {
         get { return this.textBox_phone.Text; }
         set { this.textBox_phone.Text = value; }
      }
      public string Passport
      {
         get { return this.textBox_passport.Text; }
         set { this.textBox_passport.Text = value; }
      }
      public string INN
      {
         get { return this.textBox_inn.Text; }
         set { this.textBox_inn.Text = value; }
      }
   }
}
```

Форма Form_document

```
namespace TourAnalysis
{
   public partial class Form_document : Form
   {
      int emploeeID;
      public Form_document(int emploeeID)
      {
         InitializeComponent();
         this.emploeeID = emploeeID;

         DataTable TableSort = new DataTable("TableSearch");
         DataColumn value = new DataColumn("value", typeof(System.String));
         DataColumn text = new DataColumn("text", typeof(System.String));
         TableSort.Columns.AddRange(new DataColumn[] { value, text });
         TableSort.PrimaryKey = new DataColumn[] { value };

         DataRow up = TableSort.NewRow();
         up[0] = "or";
         up[1] = "Логічна сума";
```

```
            TableSort.Rows.Add(up);

            DataRow down = TableSort.NewRow();
            down[0] = "and";
            down[1] = "Логічне множення";
            TableSort.Rows.Add(down);

            this.comboBox_if.DataSource = TableSort;
            this.comboBox_if.ValueMember = "value";
            this.comboBox_if.DisplayMember = "text";
            this.comboBox_if.SelectedIndex = 0;
        }

        private void DocumentDelete()
        {
            if (this.documentBindingSource.Current != null)
            {
                ((DataRowView)this.documentBindingSource.Current).Row.Delete();
                this.documentBindingSource.EndEdit();
            }
        }
        private void DocumentChange()
        {
            if (this.documentBindingSource.Current != null)
            {
                documentDataSet.documentRow row = (documentDataSet.documentRow)((DataRowView)this.documentBindingSource.Current).Row;
                Form_document_add form = new Form_document_add("Змінити");

                form.documentFile = row.documentFile;
                form.Number = row.number;
                form.Price = row.price;

                if (form.ShowDialog() == DialogResult.OK)
                {
                    row.documentFile = form.documentFile;
                    row.number = form.Number;
                    row.price = form.Price;
                    this.documentBindingSource.EndEdit();
                }
            }
        }
        private void DocumentAdd()
        {
            Form_document_add form = new Form_document_add("Додати");

            if (form.ShowDialog() == DialogResult.OK)
            {
                documentDataSet.documentRow row = (documentDataSet.documentRow)((DataRowView)this.documentBindingSource.AddNew()).Row;

                row.documentFile = form.documentFile;
                row.number = form.Number;
                row.price = form.Price;
                row.documentDate = DateTime.Now.Date;
                row.documentTime = DateTime.Now.TimeOfDay;

                row.emploeeID = this.emploeeID;
                this.documentBindingSource.EndEdit();
            }
        }
```

```
    private void Form_document_Load(object sender, EventArgs e)
    {
        this.documentTableAdapter.FillBy(this.documentDataSet.document, this.emploeeID);

    }

    private void зберегтиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.documentTableAdapter.Update(this.documentDataSet.document);
    }

    private void оновитиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.documentTableAdapter.FillBy(this.documentDataSet.document, this.emploeeID);
    }

    private void видалитиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.DocumentDelete();
    }

    private void додатиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.DocumentAdd();
    }

    private void змінитиToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.DocumentChange();
    }

    private void завантажитиФайлToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (this.documentBindingSource.Current != null)
        {
            documentDataSet.documentRow row =
(documentDataSet.documentRow)((DataRowView)this.documentBindingSource.Current).Row;
            if (this.saveFileDialog_file.ShowDialog() == DialogResult.OK)
            {
                File.WriteAllBytes(this.saveFileDialog_file.FileName, row.documentFile);
            }
        }
    }

    private void панельІнструментівToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.splitContainer1.Panel1Collapsed = !this.splitContainer1.Panel1Collapsed;
    }

    private void button_search_Click(object sender, EventArgs e)
    {
        string strSeparator = string.Format(" {0} ", this.comboBox_if.SelectedValue.ToString());
        bool[] ArrayChecked = new bool[] { this.checkBox_number.Checked, this.checkBox_price.Checked,
this.checkBox_date.Checked };
        string[] ArrayValues = new string[3];
        ArrayValues[0] = string.Format("number='{0}'", this.maskedTextBox_number.Text);
        ArrayValues[1] = string.Format("price='{0}'", this.numericUpDown_price.Value);
        ArrayValues[2] = string.Format("documentDate='{0}'", this.dateTimePicker_date.Value.Date);

        List<string> ArrayFilters = new List<string>();

        for (int i = 0; i < ArrayChecked.Length; i++)
```

```
        {
            if (ArrayChecked[i]) ArrayFilters.Add(ArrayValues[i]);
        }

        StringBuilder sb = new StringBuilder();
        int Count = ArrayFilters.Count;
        for (int i = 0; i < Count; i++)
        {
            sb.Append(ArrayFilters[i]);
            if (i < Count - 1) sb.Append(strSeparator);
        }

        this.documentBindingSource.Filter = sb.ToString();
    }

    private void button_cancel_Click(object sender, EventArgs e)
    {
        this.documentBindingSource.Filter = string.Empty;
    }

    private void повернутисяДоМенюToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.DialogResult = DialogResult.OK;
    }
  }
}
```

Форма Form_document_add

```
namespace TourAnalysis
{
  public partial class Form_document_add : Form
  {
    private byte[] documentfile;
    public Form_document_add(string title)
    {
      InitializeComponent();
      this.Text = title;
    }
    public byte[] documentFile
    {
      set { this.documentfile = value; }
      get
      {
        if (File.Exists(this.openFileDialog_file.FileName)) return
File.ReadAllBytes(this.openFileDialog_file.FileName);
        return this.documentfile;
      }
    }
    public string Number
    {
      get
      {
        return this.maskedTextBox_number.Text;
      }
      set
      {
        this.maskedTextBox_number.Text = value;
      }
    }
    public decimal Price
    {
      get
```

```
        {
            return this.numericUpDown_price.Value;
        }
        set
        {
            this.numericUpDown_price.Value = value;
        }
    }

    private void button_file_choose_Click(object sender, EventArgs e)
    {
        this.openFileDialog_file.ShowDialog();
    }
  }
}
```

Форма Form_document_hotel

```
namespace TourAnalysis
{
  public partial class Form_document_hotel : Form
  {
    int emploeeID;
    public Form_document_hotel(int emploeeID)
    {
      InitializeComponent();
      this.emploeeID = emploeeID;
    }

    void SetRowsHeight()
    {
      foreach (DataGridViewRow row in this.dataGridView_tour.Rows)
      {
        row.Height = 60;
      }
      foreach (DataGridViewRow row in this.dataGridView_hotel.Rows)
      {
        row.Height = 60;
      }
    }

    void comboBoxTableCreater(object sender)
    {
      DataTable TableSort = new DataTable("TableSearch");
      DataColumn value = new DataColumn("value", typeof(System.String));
      DataColumn text = new DataColumn("text", typeof(System.String));
      TableSort.Columns.AddRange(new DataColumn[] { value, text });
      TableSort.PrimaryKey = new DataColumn[] { value };

      DataRow up = TableSort.NewRow();
      up[0] = "or";
      up[1] = "Логічна сума";
      TableSort.Rows.Add(up);

      DataRow down = TableSort.NewRow();
      down[0] = "and";
      down[1] = "Логічне множення";
      TableSort.Rows.Add(down);

      ComboBox comboBox_if = (ComboBox)sender;
      comboBox_if.DataSource = TableSort;
      comboBox_if.ValueMember = "value";
      comboBox_if.DisplayMember = "text";
```

```
            comboBox_if.SelectedIndex = 0;
        }

        private void Form_document_hotel_Load(object sender, EventArgs e)
        {
            this.documentTableAdapter.Fill(this.document_hotelDataSet.document);
            this.document_hotelTableAdapter.Fill(this.document_hotelDataSet.document_hotel);
            this.hotelTableAdapter.Fill(this.document_hotelDataSet.hotel);
            this.tourTableAdapter.Fill(this.document_hotelDataSet.tour);
            this.comboBoxTableCreater(this.comboBox_if_tour);
            this.comboBoxTableCreater(this.comboBox_if_hotel);

        }

        private void tourBindingSource_ListChanged(object sender, ListChangedEventArgs e)
        {
            this.SetRowsHeight();
        }

        private void fKhoteltourID4316F928BindingSource_ListChanged(object sender, ListChangedEventArgs e)
        {
            this.SetRowsHeight();
        }

        private void button_hotel_add_Click(object sender, EventArgs e)
        {
            if (this.fKhoteltourID4316F928BindingSource.Current != null)
            {
                document_hotelDataSet.hotelRow parentRow = (document_hotelDataSet.hotelRow)((DataRowView)this.fKhoteltourID4316F928BindingSource.Current).Row;
                document_hotelDataSet.document_hotelRow[] childRows = parentRow.Getdocument_hotelRows();
                bool tmp = true;
                int documentID = (int)this.comboBox_document.SelectedValue;
                foreach (document_hotelDataSet.document_hotelRow item in childRows) if(item.documentID == documentID) tmp = false;

                if (tmp)
                {
                    document_hotelDataSet.document_hotelRow row = (document_hotelDataSet.document_hotelRow)((DataRowView)this.fKdocumenthotel46E78A0CBindingSource.AddNew()).Row;
                    row.documentID = documentID;
                    this.fKdocumenthotel46E78A0CBindingSource.EndEdit();
                }
            }
        }

        private void button_hotel_delete_Click(object sender, EventArgs e)
        {
            if (this.fKdocumenthotel46E78A0CBindingSource.Current != null)
            {
                ((DataRowView)this.fKdocumenthotel46E78A0CBindingSource.Current).Delete();
                this.fKdocumenthotel46E78A0CBindingSource.EndEdit();
            }
        }

        private void button_save_Click(object sender, EventArgs e)
        {
            this.document_hotelTableAdapter.Update(this.document_hotelDataSet.document_hotel);
        }

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

```
            string strSeparator = string.Format(" {0} ", this.comboBox_if_tour.SelectedValue.ToString());
            bool[] ArrayChecked = new bool[] { this.checkBox_country.Checked, this.checkBox_city.Checked };
            string[] ArrayValues = new string[2];
            ArrayValues[0] = string.Format("country='{0}'", this.textBox_country.Text);
            ArrayValues[1] = string.Format("city='{0}'", this.textBox_city.Text);

            List<string> ArrayFilters = new List<string>();

            for (int i = 0; i < ArrayChecked.Length; i++)
            {
                if (ArrayChecked[i]) ArrayFilters.Add(ArrayValues[i]);
            }

            StringBuilder sb = new StringBuilder();
            int Count = ArrayFilters.Count;
            for (int i = 0; i < Count; i++)
            {
                sb.Append(ArrayFilters[i]);
                if (i < Count - 1) sb.Append(strSeparator);
            }

            this.tourBindingSource.Filter = sb.ToString();
        }

        private void button_cancel_tour_Click(object sender, EventArgs e)
        {
            this.tourBindingSource.Filter = string.Empty;
        }

        private void button_search_hotel_Click(object sender, EventArgs e)
        {
            string strSeparator = string.Format(" {0} ", this.comboBox_if_hotel.SelectedValue.ToString());
            bool[] ArrayChecked = new bool[] { this.checkBox_hotelName.Checked, this.checkBox_phone.Checked,
this.checkBox_stars.Checked };
            string[] ArrayValues = new string[3];
            ArrayValues[0] = string.Format("hotelName='{0}'", this.textBox_hotelName.Text);
            ArrayValues[1] = string.Format("phone='{0}'", this.textBox_phone.Text);
            ArrayValues[2] = string.Format("stars='{0}'", this.numericUpDown_stars.Value);

            List<string> ArrayFilters = new List<string>();

            for (int i = 0; i < ArrayChecked.Length; i++)
            {
                if (ArrayChecked[i]) ArrayFilters.Add(ArrayValues[i]);
            }

            StringBuilder sb = new StringBuilder();
            int Count = ArrayFilters.Count;
            for (int i = 0; i < Count; i++)
            {
                sb.Append(ArrayFilters[i]);
                if (i < Count - 1) sb.Append(strSeparator);
            }

            this.fKhoteltourID4316F928BindingSource.Filter = sb.ToString();
        }

        private void button_cancel_hotel_Click(object sender, EventArgs e)
        {
            this.fKhoteltourID4316F928BindingSource.Filter = string.Empty;
        }

        private void button_update_Click(object sender, EventArgs e)
```

```
        {
            this.documentTableAdapter.Fill(this.document_hotelDataSet.document);
            this.document_hotelTableAdapter.Fill(this.document_hotelDataSet.document_hotel);
            this.hotelTableAdapter.Fill(this.document_hotelDataSet.hotel);
            this.tourTableAdapter.Fill(this.document_hotelDataSet.tour);
        }

        private void button_back_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.OK;
        }
    }
}
```

Форма Form_emploee_add

```
namespace TourAnalysis
{
    public partial class Form_emploee_add : Form
    {
        private byte[] photo;
        public Form_emploee_add(string title)
        {
            InitializeComponent();
            this.Text = title;
        }

        public string Surname
        {
            get
            {
                return this.textBox_surname.Text;
            }
            set
            {
                this.textBox_surname.Text = value;
            }
        }
        public string emploeeName
        {
            get
            {
                return this.textBox_name.Text;
            }
            set
            {
                this.textBox_name.Text = value;
            }
        }
        public string Patronymic
        {
            get
            {
                return this.textBox_patronymic.Text;
            }
            set
            {
                this.textBox_patronymic.Text = value;
            }
        }
        public string Login
        {
            get
```

```
    {
        return this.textBox_login.Text;
    }
    set
    {
        this.textBox_login.Text = value;
    }
}
public string Password
{
    get
    {
        return this.textBox_password.Text;
    }
    set
    {
        this.textBox_password.Text = value;
    }
}
public string Phone
{
    get
    {
        return this.maskedTextBox_phone.Text;
    }
    set
    {
        this.maskedTextBox_phone.Text = value;
    }
}
public string Passport
{
    get
    {
        return this.textBox_passport.Text;
    }
    set
    {
        this.textBox_passport.Text = value;
    }
}
public string Street
{
    get
    {
        return this.textBox_street.Text;
    }
    set
    {
        this.textBox_street.Text = value;
    }
}
public int House
{
    get
    {
        return (int)this.numericUpDown_house.Value;
    }
    set
    {
        this.numericUpDown_house.Value = value;
    }
}
```

```
        public string Korpuse
        {
            get
            {
                return this.textBox_korpuse.Text;
            }
            set
            {
                this.textBox_korpuse.Text = value;
            }
        }
        public int Apartment
        {
            get
            {
                return (int)this.numericUpDown_apartment.Value;
            }
            set
            {
                this.numericUpDown_apartment.Value = value;
            }
        }
        public byte[] Photo
        {
            get
            {
                if (File.Exists(this.openFileDialog_photo.FileName)) return
File.ReadAllBytes(this.openFileDialog_photo.FileName);
                return this.photo;
            }
            set
            {
                this.photo = value;
                MemoryStream stmBLOBData = new MemoryStream(this.photo);
                this.pictureBox_photo.Image = Image.FromStream(stmBLOBData);
            }
        }

        private void button_photo_change_Click(object sender, EventArgs e)
        {
            if(this.openFileDialog_photo.ShowDialog() == DialogResult.OK &&
File.Exists(this.openFileDialog_photo.FileName))
            this.pictureBox_photo.Image = Image.FromFile(this.openFileDialog_photo.FileName);
        }
    }
}
```

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота_ Dubovenko_Y_A.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота_ Dubovenko_Y_A.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Dubovenko _Products_client.rar	Архів. Містить коди програми клієнту та її залежості
Презентація	
Dubovenko_Y_A.ppt	Презентація кваліфікаційної роботи