МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних систем

«До захисту в ЕК»		**«До захисту допущено»**	
Директор інституту(декан факультету)		Завідувач кафедри	
_____	Андрій Форсюк	_____	Сергій Чумаченко
(підпис)	(ім'я та прізвище)	(підпис)	(ім'я та прізвище)
«___» _____ 2022 р.		«___» _____ 2022 р.	

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)
освітньо-професійної програми Комп'ютерні науки

на тему: Розроблення Інформаційної системи діяльності туристичного агентства

Виконав: здобувач 4 курсу, групи 2

Івахненко Андрій Олегович
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Мазуренко Ольга Олександрівна
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти _____ _____
(ім'я та прізвище) (підпис)
_____ _____
(ім'я та прізвище) (підпис)
_____ _____
(ім'я та прізвище) (підпис)

Рецензент *Ярослав Смітюх* _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач _____
(підпис)

Київ – 2022 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) __Автоматизації і комп'ютерних систем_____
Кафедра __Інформаційних систем_____
Освітній ступінь__бакалавр_____
Спеціальність__122 «Комп'ютерні науки»_____
(код і назва)
Освітньо-професійна програма __Комп'ютерні науки_____
_____
(назва)

ЗАТВЕРДЖУЮ
Завідувач
кафедри Інформаційних систем

_____Чумаченко С.М.
"____" _____2022року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

_____Івахненко Андрій Олегович _____
(прізвище, ім'я, по батькові)

1. Тема роботи ___Розроблення Інформаційної системи діяльності туристичного агентства
_____
керівник роботи_____Мазуренко Ольга Олександрівна _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від "31" березня 2022року №163-КС
2. Строк подання здобувачем роботи_6 червня 2022 року_____
3. Вихідні дані до роботи *Дані про роботу туристичної галузі, структура туристичної фірми, бізнес-процес туристичної фірми*_____
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)__*Аналіз поточного стану інформаційної системи в діяльності туристичного агентства, постановка задачі, розробка, тестування та реалізація прграмного продукту*_____
_____
5. Перелік графічного матеріалу
6-таблиць, 30-ілюстрацій, 10-літературних джерел
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1.	Мазуренко О.О	*04.04.2022*	*06.06.2022*
2.	Мазуренко О.О	*04.04.2022*	*06.06.2022*
3.	Мазуренко О.О	*04.04.2022*	*06.06.2022*

7. Дата видачі завдання____31 березня 2022 року_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Вибір теми, подання заяви в деканат	01.04.2022	Виконане
2.	Підготовка першого розділу роботи та подання його керівнику.	18.04.2022	Виконане
3.	Підготовка другого розділу роботи та подання його керівнику.	15.05.2022	Виконане
4.	Підготовка вступу та висновків	20.05.2022	Виконане
5.	Доопрацювання роботи з урахуванням зауважень керівника.	30.05.2022	Виконане
6.	Подача електронного варіанту роботи для перевірки на плагіат.	01.06.2022	Виконане
7.	Створення Презентації	06.06.2022	Виконане

Здобувач _____ ____Івахненко А.О._____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____

АНОТАЦІЯ

Розроблено програмне забезпечення для організації роботи туристичного агентства.

Об’єкт дослідження: діяльність туристичного агентства.

Предмет дослідження: методи та засоби підтримки діяльність туристичного агентства.

Мета дослідження полягає у створенні інформаційної системи діяльності туристичного агентства. Ця система призначена для ведення обліку даних туристичного агентства. Відповідно до мети в було поставлено такі завдання:

- зробити аналіз поточного стану інформаційної системи діяльності туристичного агентства;
- визначити функціональні вимоги до інформаційної системи;
- визначити технологія, яку необхідно застосувати для розробки програмного забезпечення;
- провести опис основних функціональних можливостей системи;
- провести тестування системи;
- розробити інструкцію користувача програми.

Результати дослідження: розроблено програмне забезпечення для ведення обліку діяльності туристичного агентства. Програмне забезпечення дозволяє не тільки вести облік діяльності туристичного агентства, але й здійснювати формування звітності. Також даний продукт є універсальним і його можна використовувати для будь-яких агенцій, що займаються туризмом. Основна особливість продукту є безкоштовність та масовість у використанні, так як даний продукт можна легко переносити на будь-які операційні системи сімейства Windows.

Ключові слова: ЗВІТ, БАЗА ДАНИХ, ПРЕДМЕТНА ОБЛАСТЬ, ПРОЕКТУВАННЯ, ДІАГРАМА, ПРОГРАМА, ІНТЕРФЕЙС, ТУРАГЕНСТВО, СЕРЕДОВИЩЕ РОЗРОБКИ, ПРОДАЖ, КЛІЄНТ, МЕНЕДЖЕР.

ANNOTATION

Developed software for organizing the work of a travel agency.

Object of research: activity of a travel agency.

Subject of research: methods and means of supporting the activities of a travel agency.

The purpose of the study is to create an information system of the travel agency. This system is designed to keep records of travel agency data. In accordance with the purpose of the following tasks were set:

• make an analysis of the current state of the information system of the travel agency;

• determine the functional requirements for the information system;

• identify the technology that needs to be used for software development;

• describe the main functionalities of the system;

• test the system;

• develop user instructions for the program.

The results of the study: developed software for accounting for the activities of a travel agency. The software allows not only to keep track of the activities of the travel agency, but also to carry out reporting. This product is also universal and can be used for any tourism agency. The main feature of the product is free and massive in use, as this product can be easily transferred to any operating system of the Windows family.

Keywords: REPORT, DATABASE, SUBJECT AREA, DESIGN, DIAGRAM, PROGRAM, INTERFACE, TRAVEL AGENCY, DEVELOPMENT ENVIRONMENT, SALES, CLIENT

ЗМІСТ

ВСТУП

У нашому суспільстві, у зв'язку з тим, що подорожі світом стали доступні і легко здійсненні, з'явилася велика кількість туристичних компаній, які допомагають здійснити подорожі. З метою візиту подорожі діляться на бізнес-трип, подорож з метою відпочинку, виїзні семінари тощо.

Завдяки такому різноманіттю та різновидам подорожей, у туристичних компаніях з'явилася маса проблем з тим, як вести облік людей, які тільки збираються подорожувати, як облікувати людей, які вже здійснили свої подорожі, як вести облік турів, які є в агенції та ін. проблеми.

У той час, коли туристичний вид діяльності був не так розвинений, дані проблеми ускладнювали роботу багатьох компаній, але через низьку завантаженість, була можливість проводити багато операцій у ручному вигляді, що не було складно для компаній. Зараз, коли велика кількість людей здатна дозволити собі подорожувати, подібна хаотичність призводить до незручності роботи з документами, викликає труднощі з урахуванням та прогнозування дій агентства.

Зважаючи на наявні проблеми, основною метою роботи є розробка автоматизованої інформаційної системи, яка служить для спрощення роботи менеджерів туристичного агентства.

Для успішної діяльності туристичного агентства в сучасних умовах велике значення мають рішення у галузі автоматизованого обліку клієнтів та бронювання турів. Для цього кола завдань існують спеціалізовані системи, здатні забезпечити керівництво компанії інформацією, для прийняття оперативних та стратегічних рішень у галузі покращення роботи компанії, а також для персоналу оперативного обслуговування клієнта.

В даний час на ринку програмних засобів існує безліч різних продуктів, призначених для вирішення подібних питань.

Поруч із програмами, призначеними для автоматизації бізнес процесів у компанії загалом, розроблено ряд програм окремих відділів компанії. Зараз легко

можна знайти організацію чи приватну особа, що займається розробкою інформаційних систем.

Досвід створення ІС дозволяє умовно виділити такі основні етапи їх життєвого циклу:

– аналіз – визначення того, що має робити система;

– проектування - визначення того, як система робитиме те, що вона повинна робити.;

– розробка - створення функціональних компонентів та підсистем, з'єднання підсистем у єдине ціле;

– тестування – перевірка функціональної та параметричної відповідності системи показникам, визначеним на етапі аналізу;

– використання - встановлення та введення системи в дію;

– супровід – забезпечення штатного процесу експлуатації системи;

– діагностика необхідності модернізації;

– модернізація.

Актуальність цієї роботи пов'язані з виникненням потреби туристичних агентств у розробці програмного забезпечення, що дозволяє автоматизувати процес введення, виведення, зберігання, обробки, реєстрації необхідної інформації.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- Провести дослідження предметної галузі.
- Проаналізувати аналогічні програмні продукти.
- Розробити моделі інформаційної системи.

Розробити програмний продукт для автоматизації процесу обліку діяльності туристичного агентства допомоги на платформі .NET.

1 АНАЛІЗ ПОТОЧНОГО СТАНУ ІНФОРМАЦІЙНОЇ СИСТЕМИ В ДІЯЛЬНОСТІ ТУРИСТИЧНОГО АГЕНТСТВА

1.1 Особливості діяльності сфери туристичного агентства

1.1.1 Туристична індустрія та її устрій

Як відомо основу туристичної індустрії складають фірми туроператори та турагенти, що займаються туристичними поїздками, продажем їх у вигляді путівок та турів; що надають послуги з розміщення та харчування туристів (готелі, кемпінги та ін.), їх пересування країною, а також органи управління, інформації, реклами з дослідження туризму та підготовки для нього кадрів, підприємства з виробництва та продажу товарів туристського попиту [1]. На туризм працюють інші галузі, котрим обслуговування туристів перестав бути основним видом діяльності (підприємства культури, торгівлі та інших.). Туризм – інформаційно насичена діяльність. Існує трохи інших галузей, у яких збирання, обробка, застосування та передача інформації були б настільки ж важливі для щоденного функціонування, як у туристичній індустрії.

Послуга в туризмі не може бути виставлена та розглянута у пункті продажу як споживчі або виробничі товари. Її зазвичай купують заздалегідь і далеко від місця споживання. Таким чином, туризм на ринку майже повністю залежить від зображень, описів, засобів комунікацій та передачі інформації.

Влаштування туристичної галузі дуже схоже на організацію будь-якої іншої економічної сфери діяльності (рис. 1.1). Турагент - фізична чи юридична особа, яка виступає посередником із продажу сформованих туроператором турів. Туроператор – туристична організація, що займається комплектацією турів.

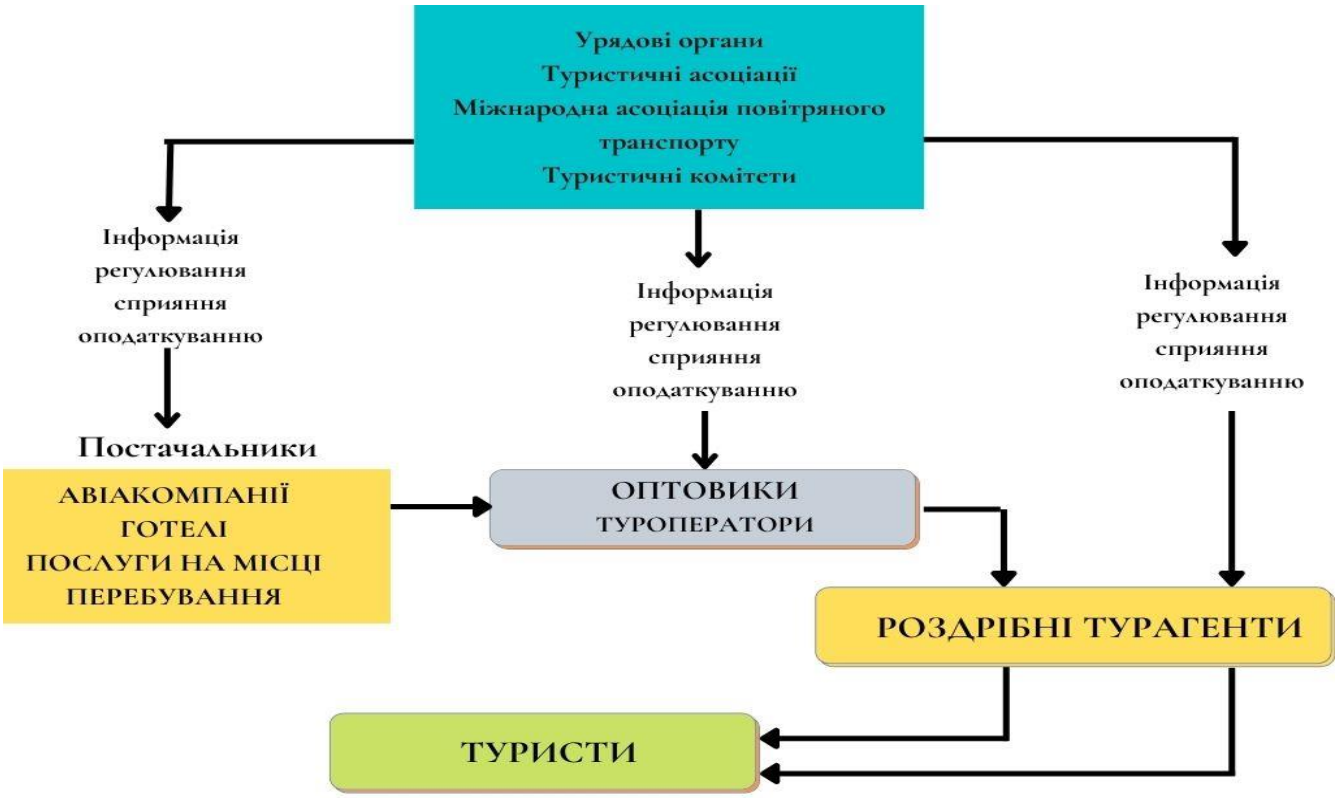


Рисунок 1.1 – Робота туристичної галузі

Проте виділяється особливість – сполучним центром, що утримує різних виробників у рамках туристичної галузі, є інформація. Саме інформаційні потоки, а не товари забезпечують зв'язок між виробниками туристичних послуг. Вони йдуть у вигляді потоків даних, але виступають у формі послуг і платежів. Послуги, наприклад (ночівля в готелі; оренда автомобіля; комплексні тури та місця в літаках) не пересилаються турагентам, які в свою чергу не зберігають їх доти, доки не продадуть споживачам. Передається та використовується інформація про наявність, вартість та якості цих послуг [2]. Так само реальні платежі не переводяться від турагентів турпостачальникам а комісійні - від турпостачальників. Насправді перекладається інформація про платежі та надходження (рис. 1.2).

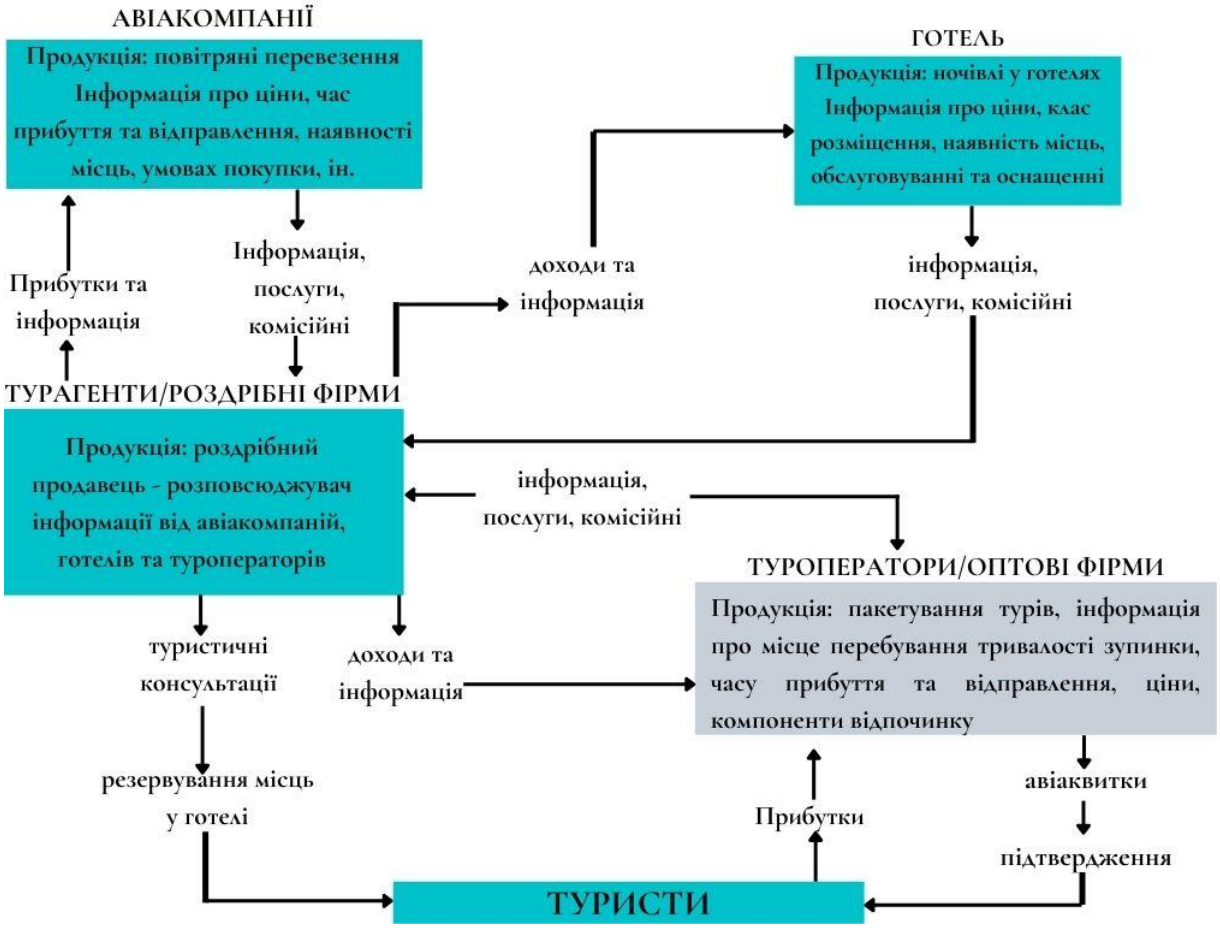


Рисунок 1.2 – Сполучний матеріал туристичного бізнесу

Виділяють три характерні риси туризму. По-перше, це – різноманітна та інтегрована торгівля послугами. По-друге, це комплексна послуга, як з погляду виробника, так і споживача. Зрештою, туризм це – інформаційно насичена послуга. Тому туризм - як міжнародний, і внутрішній, - сфера зростаючого застосування інформаційних технологій. Система інформаційних технологій, що використовуються у туризмі, складається з комп'ютерної системи резервування, системи проведення телеконференцій, відеосистем, комп'ютерів, інформаційних систем управління, електронних інформаційних систем авіаліній електронного пересилання грошей, телефонних мереж, рухомих засобів сполучення тощо. При цьому необхідно відзначити, що ця система технологій розгортається не туристичними агенціями, готелями чи авіакомпаніями кожним окремо, а всіма ними. Понад те, використання кожним сегментом туризму системи інформаційних

технологій має значення всім інших елементів. Через це, системи внутрішнього управління готелем можуть бути пов'язані з комп'ютерними глобальними мережами, які забезпечують, у свою чергу, основу для зв'язку з готельними системами резервування, які вже у зворотному напрямку можуть бути доступні туристичним агентствам через їх комп'ютери. Таким чином тур. сфера використовує інтегровану систему інформаційних технологій, яка розповсюджується у туризмі.

1.1.2 Структура туристичної агенції

Управління туристичного агентства складає основи певної організаційної структури. Структура туристичного агентства та його підрозділів визначається агенством самостійно. Під час створення організаційної структури управління необхідно забезпечити ефективне розподіл функцій управління за підрозділами. При цьому важливе виконання наступних умов:

– вирішення тих самих питань не повинно перебувати у віданні різних підрозділів;

– всі функції управління повинні входити до обов'язків керуючих підрозділів;

– на цей підрозділ не покладається вирішення питань, які ефективніше вирішувати в іншому.

Структура туристичної агенції представлена на рис. 1.3.

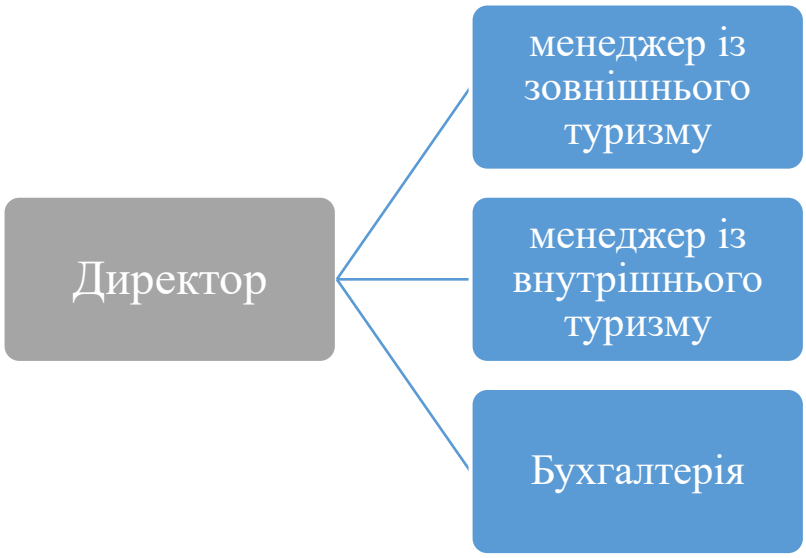


Рисунок 1.3 – Структура туристичної агенції

1.1.3 Бізнес процес туристичного агентства

Продаж туристського продукту за умов жорсткої конкуренції стає дедалі складнішим завданням. У зв'язку з цим особливого значення набуває удосконалення технології продажів.

Для досягнення цієї мети менеджер повинен проводити маркетингові дослідження туристського ринку, знати сильні та слабкі сторони своїх маршрутів, порівняно з конкурентами. Менеджер повинен регулярно брати участь у зборах фірми, що проводяться для обміну досвідом та обговорення різних скрутних ситуацій; володіти короткою інформацією про всі напрямки, про наявність "горючих" путівок і, таким чином, постійно знати поточні справи фірми.

Найважливішим аспектом технології продажів є психологія взаємодії із клієнтом. Співробітник фірми, який бере участь у процесі продажу, повинен вміти красиво представити свої маршрути, а й професійно володіти технікою спілкування з клієнтом.

Найкращим способом представленя бізнес- процесу туристичної фірми буде відображення його у вигляді схеми на рис. 1.4.

Рисунок 1.4 – Бізнес- процес туристичної фірми

1.1.4 Необхідність автоматизації бізнес-процесів у даній сфері

На сьогоднішній день на ринку представлено безліч систем для автоматизації роботи туристичного агентства.

Завдяки автоматизації відбувається:

- підвищення продуктивність праці;
- зведення до мінімуму негативного впливу людського фактору на найважливіші бізнес-процеси;
- безпечне зберігання інформації;
- підвищення якості обслуговування клієнтів;
- швидке отримання звітів;
- швидка підготовка документів;

- підвищення ефективності обслуговування.

Автоматизація необхідна в тих ділянках діяльності, де її поява збільшить швидкість виконання завдань і, відповідно, підвищить ефективність праці менеджерів туристичного агентства.

Менеджер без установленого програмного забезпечення багато працюватиме і йому знадобиться велика кількість часу, тому що кожного разу при запиті будь-якого звіту доведеться діставати всі папери, проводити розрахунки самостійно на калькуляторі тощо. Але якщо впровадити програму для автоматизації його роботи, то будь-який звіт можна буде сформувати за допомогою натискання однієї кнопки.

Автоматизація роботи менеджера тур. агентства дозволяє підвищити ефективность роботи менеджерів, підвищити якість обслуговування клієнтів, збільшити точність даних, швидкість виконання операцій тощо.

Автоматизація діяльності завжди є унікальним рішенням та робота над цим проектом завжди індивідуальна. Але робота з проектами побудована на однакових засадах.

Процес автоматизації необхідно поділити на наступні етапи:

- проектне дослідження;
- розробка і впровадження;
- супровід.

На першому етапі вивчається предметна область. Ставиться ціль, яка допоможе вирішити поставлені завдання. Також збирається інформація про те, як працює дана сфера. Після перерахованого вище визначається вартість та терміни, необхідні для реалізації проекту.

На другому етапі робота поділяється на фази:

- написання технічного завдання;
- розробка;
- тестування;
- дослідна експлуатація;
- навчання користувачів;
- введення в експлуатацію.

На третьому етапі здійснюється супровід автоматизованої системи:

- виправлення помилок;
- доопрацювання для покращення роботи програми, консультація.

1.2 Обробка інформації в тур. Агентстві

1.2.1 Опис технології обробки інформації в тур. агентстві

У туристичному агентстві немає програмного забезпечення для обліку клієнтів, турів, складання документації, виведення звітів та статистики. Запис клієнтів здійснювався ручним способом у журналі. А для обліку та складання звітів використовувалась програма Microsoft Excel. Але при великому потоці клієнтів, збільшився і час їх виконання.

Виявлено такі недоліки обслуговування клієнта:

1. Великі часові витрати на реєстрацію клієнтів.
2. Неповне консультування клієнтів через тривалий пошук або неможливість доступу до необхідної інформації.
3. Небезпечна технологія зберігання документації.
4. Великі часові витрати на пошук необхідної інформації про тури та ін.
5. Проблеми з формуванням звітів про діяльність туристичного агентства.

Провівши аналіз поточної інформаційної системи, прийнято автоматизувати робочі процеси компанії (ведення бази клієнтів, бази турів, реєстрація клієнтів, формування звітів тощо).

1.2.2 SWOT-аналіз існуючої інформаційної системи

Згідно із [3] SWOT-аналіз — це аналіз, який показує сильні та слабкі сторони, можливості та загрози. Отже, стан туристичного агентства залежить від успішного реагування на різні дії ззовні, тому при аналізі зовнішньої ситуації необхідно

враховувати суттєві фактори на даний час. Розгляд цих чинників з можливостями туристичного агентства дозволяє вирішувати проблеми .

SWOT-аналіз — це необхідний елемент досліджень, обов'язковий попередній етап при складанні будь-якого рівня стратегічних і маркетингових планів.

В результаті SWOT-аналізу отримано дані, які необхідні при розробці стратегічних цілей та завдань туристичного агентства. У таблиці 1.1 представлено результати SWOT-аналізу для туристичного агентства.

Таблиця 1.1 – SWOT-аналіз бізнес сфери туристичного агентства

Потенційні внутрішні сильні сторони (S)	***Потенційні внутрішні слабкості (W)***
1. Застосовується проста форма для формування заявок, звична для працівників (Журнал, MS Excel) 2. Не потрібно часто використовувати ПК. 3. Можливість залучення клієнтів акціями.	1. Використання паперових носіїв інформації. 2. Великі витрати ручної праці. 3. Довготривале та складне формування звітності.
Потенційні зовнішні сприятливі можливості (O)	***Потенційні зовнішні погрози (T)***
1. Знижки від туристичного агентства. 2. Можливість запровадження нового програмного забезпечення.	1. Можливість втрати інформації на паперових носіях. 2. Слабка система забезпечення безпеки інформації. 3. Відсутність актуальної інформації.

Сильні та слабкі сторони є факторами внутрішнього середовища, які впливають на стан туристичного агентства. Можливості та загрози — зовнішні фактори, що впливають на туристичне агентство ззовні та при цьому не контролюються нею.

Переваги SWOT-аналізу полягають у тому, що він достатньо просто дозволяє розглянути становище туристичного агентства, послуги у галузі, і тому він є популярним інструментом в управлінні ризиками та прийнятті управлінських рішень.

1.3 Аналіз сучасних програм із визначенням їх переваг та недоліків

1.3.1 Критерії ПО для тур. агентства

Основною метою для автоматизування варіанту вирішення завдання є створення програмного забезпечення для обліку послуг туристичного агентства .

Для реалізації проекту автоматизації мають забезпечуватися такі можливості:

1. Реєстрація клієнтів, країн, міст, послуг, подорожей

2. Виконання розрахунків та видача різних звітів та статистик.

3. Отримання засобів моніторингу замовлень та формування звітності.

Переваги автоматизованого процесу виражаються у:

- скороченні часу на процедуру прийому, реєстрації клієнтів;
- скороченні часу на процес формування різних звітів;
- оптимізації роботи менеджерів, з метою швидкого та надійного способу ведення, пошуку, обробки, безпечного зберігання інформації про результати діяльності та поточний стан туристичного агентства.
-

1.3.2 U-ON.Travel

U-ON.Travel – це CRM-система для туристичного бізнесу. Дозволяє ефективно керувати туристичною компанією та її бізнес-процесами з будь-якого комп'ютера або планшета, розташованого у будь-якій точці світу. Дані сервісу St artpack. Більше того, система дозволяє значно підвищити рівень сумлінності роботи менеджерів і, як наслідок, збільшити обсяги продажів.

Менеджери витрачатимуть мінімум часу на створення, ведення та опрацювання замовлень, а туристи відчують значне покращення сервісу. Система дозволяє розраховувати заробітну плату, створювати програми лояльності клієнтів, вести фінансове планування та багато іншого.

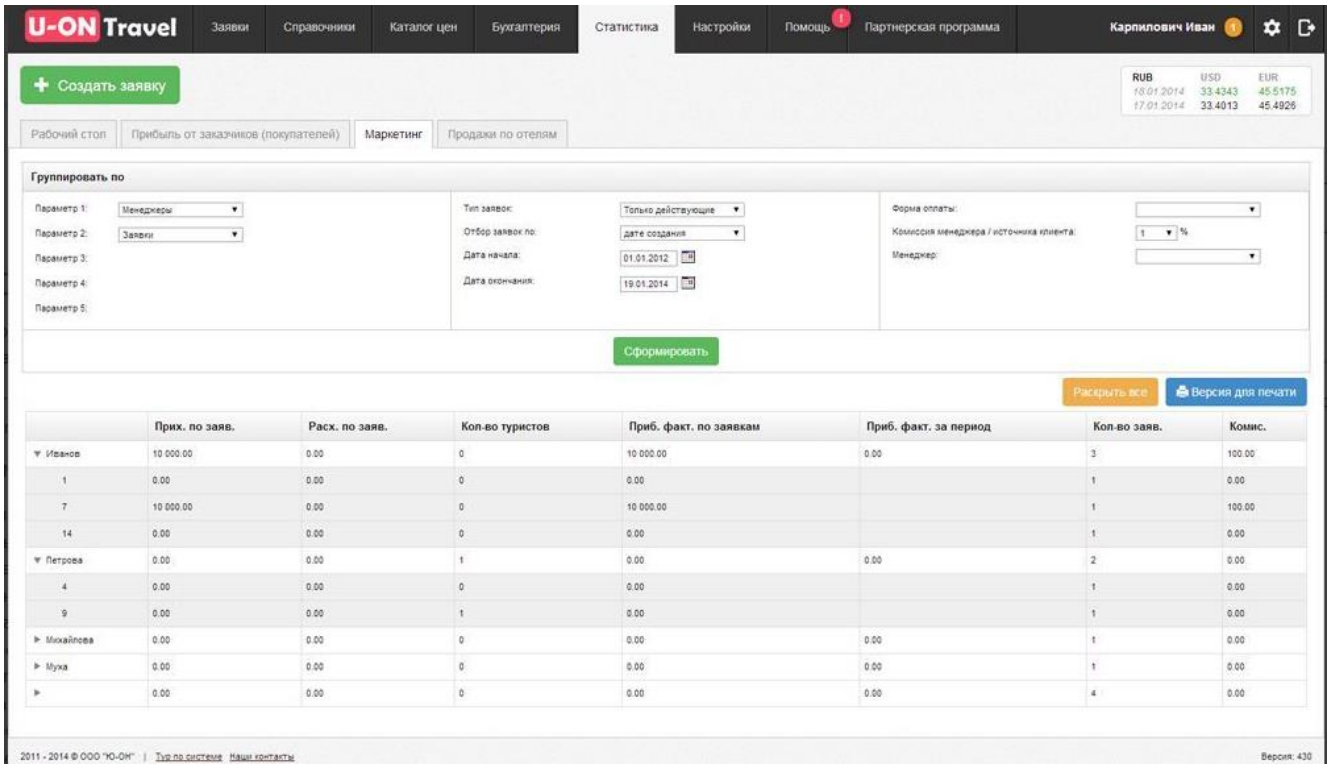


Рисунок 1.5 – Інтерфейс програми «U-ON.Travel»

Можливості U-ON:

- Єдине хмарне сховище з архівом для роботи компанії.
- Оперативний облік звернень туристів з розрахунком вирв продажу.

- Автоматичний розрахунок суми туру, управління заявками туристів з деталізацією послуг та автоматичним оформленням документів. Широка інтеграція з сайтами, IP-телефонією та іншими сервісами, що застосовуються.
- SMS та e-mail розсилки по клієнтській базі.
- Система обліку платежів, ведення бухгалтерії та фінансового планування.
- Розрахунок системи мотивації працівників.
- Формування програм лояльності, систем знижок та бонусних програм для туристів.
- Високий рівень захисту даних із застосуванням систем шифрування.
- Застосування технології API та WebHooks.

Плюси системи:

- Швидкий та зрозумілий процес додавання лідів;
- Зручне редагування заявок/звернень;
- Інтеграція з багатьма цікавими проектами, що реально спрощують життя тур компанії;
- Особистий кабінет туриста - чудова річ;
- Приємний інтерфейс;
- Постійна оперативна підтримка з будь-якого питання;
- Зручна та докладна статистика (воронка продажів, касова книга);
- Постійне додавання нових "фішок", гарний зворотний зв'язок із користувачем.

Мінуси:

- не доопрацьовано довідник по готелям;
- інтерфейс потрібно вивчати досконально, дуже складна в користуванні.

1.3.3 Платформа ERP.travel

ERP.travel – це професійна онлайн система для турагентств. Система максимально автоматизує процес роботи із замовленням туру, включаючи друк усіх документів. В системі найшвидший час оформлення замовлення туру – всього 4,5 хвилини. При цьому в системі є найбільший функціонал: путівки та заявки туроператорам; оформлення індивідуальних турів; приходи та витрати по касі та банку; непрямі витрати; управлінський облік; звітність та інтеграція із сайтами.

Найбільші переваги в системі для мереж турагентств, оскільки лише в ній можна організувати роботу всіх офісів в одній базі. З системою можна працювати з ноутбуків та планшетів, з офісу, будинку та кафе. У системі автоматизації турагентств ERP.travel реалізовано гнучку систему прав доступу турагентам, що дозволяє в одній базі вести облік навіть не однієї, а кількох мереж турагентств.

Інтерфейс програми «ERP.travel» зображений на рис. 1.6.

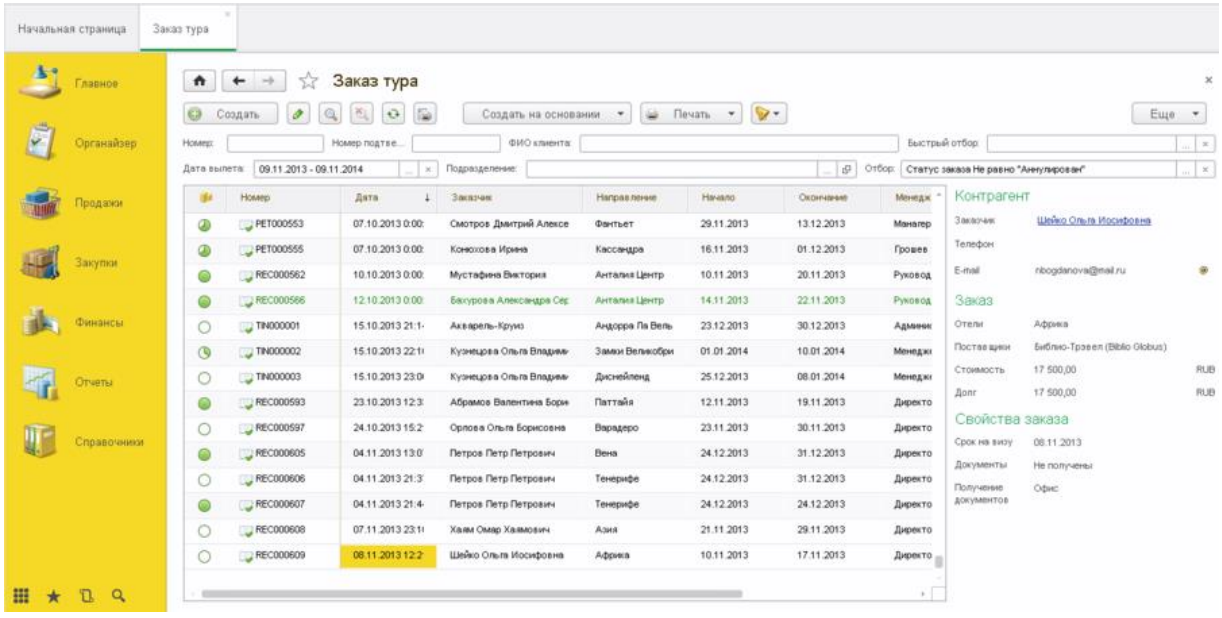


Рисунок 1.6 – Інтерфейс програми «ERP.travel»

Функціонал ERP:

- База клієнтів;

- Управління замовленнями;
- Кол-центр та телефонія;
- Історія взаємодії із клієнтом;
- Системи лояльності;
- Моніторинг ефективності персоналу;
- Тайм менеджмент;
- Управління підтримкою;
- Звіти;
- Інтеграція з поштою;
- Email-розсилки;
- Шаблони проектів;
- Сховище файлів;
- Білінг та рахунки;
- Експорт/імпорт даних;
- API для інтеграції;
- Веб-форми.

Плюси системи:

- Централізоване зберігання даних - агрегована оперативна інформація по всій мережі турагентств;
- Необмежені можливості звітів прямо з веб-інтерфейсу;
- Самостійне налаштування вибірок та нових варіантів звітів;
- Гнучке настроювання умов договорів, комісій та курсів;
- Контролює собівартість турів за договорами з постачальниками;
- Автоматично оновлюваний курс ЦП та розрахункові курси від ЦП;
- Вже наповнені та поповнювані довідники готелів, номерів, пансіонів, категорії готелів;

- Збереження в системі оригінальних файлів та швидкий їх друк, наприклад, рахунок постачальника або пам'ятка туристу;
- Додаткові властивості для документів та довідників.

Мінуси:

- Немає додатку, необхідне постійне підключення до інтернету.
- Перевантажений інтерфейс.

1.3.4 Порівняльний аналіз існуючих програмних продуктів

В таблиці 1.2 приведений порівняльний аналіз існуючих програмних продуктів.

Таблиця 1.2 – Аналіз існуючих програмних продуктів

Характеристики	*U-ON. Travel*	*ERP.travel*	*Розроблене програмне забезпечення*
Інтерфейс програми	Зручний	Зручний	Зручний
Функціонал	Широкий	Широкий	Необхідний для більшості агентств
Процес впровадження	Платне	Платне	Безкоштовне
Технічні вимоги	Високі	Мінімальні	Мінімальні
Вартість	1200 грн. за 1 місяць до	300 грн. за 1 місяць для 1 користувача	Безкоштовна

	10 користувачів		
Український інтерфейс	Немає	Немає	Присутній
Оцінка	2 (+)	3 (+)	5 (+)

Із проведеного аналізу існуючих програмних продуктів можна зробити висновок, що є потреба в розробці нового програмного продукту, який мав би рад переваг над конкурентами. Також розробку програмного забезпечення необхідно здійснити із врахуванням особливостей діяльності турагентств, специфіку їх функціонування, що дозволяє виконувати певні функції майбутньої системи, актуальні на даний момент, а також проводити гнучке налаштування та розширення функціоналу у разі необхідності у процесі використання програмного продукту.

1.4 Характеристика організації вирішення проекту

Розроблене програмне забезпечення передбачається використовувати у діяльності туристичного агентства для роботи менеджера з метою вдосконалення процесу реєстрації клієнта, а також вибір відповідного туру в залежності від його побажань.

Перевагою пропонованої автоматизованої системи є простота та інтерфейс. У цій програмі реалізується весь необхідний для роботи функціонал. Також програма забезпечує зручність зберігання та обробки інформації, реалізує швидке отримання інформації про наявність турів. Менеджер туристичного агентства може отримувати різні види звітів та статистику.

1.5 Обґрунтування проектних рішень

Важливим фактором, який необхідно врахувати при розробці програмного забезпечення є потреба в ресурсах, які можуть бути в туристичному агентстві.

Мінімальні вимоги для запуску та функціонування розробленого програмного забезпечення:

1. Процесор PentiumI V/Xeon 2.4 ГГц.

2. Оперативна пам'ять: 1024 Мб і вище.

3. Вільний дисковий простір не менше 120 Мб.

4. Мережева картка.

5. Миша.

6. Клавіатура.

7. Монітор.

З опису видно, що до технічного забезпечення програми потреби не є високими.

1.6 Постановка задачі

Таким чином, необхідно розробити модель для предметної області «туристичного агентства», дотримуючись заданих бізнес – правил, та вимог. Для того, щоб здійснити це необхідно провести фізичне проектування моделі за допомогою СУБД Access. Оскільки для база даних буде створена засобами Access, то доцільним буде реалізувати програму у середовищі Visual Studio на мові програмування C#.

На базі фізичної моделі розробити програму на мові програмування C#, яка буде реалізовувати графічний інтерфейс для взаємодіїї із базою даних.

Програма має реалізовувати наступні задачі:

- ✓ додавання, редагування інформації про клієнта;
- ✓ додавання, редагування інформації про менеджера;

- ✓ додавання, редагування інформації про країни;
- ✓ додавання, редагування інформації про міста;
- ✓ додавання, редагування інформації про послуги туристичної фірми;
- ✓ оформлення подорожі;
- ✓ формування звітності про поїздки клієнтів;
- ✓ формування звітності про надані послуги турагенства.

Основні можливості системи:

Клієнти:

- ✓ Можливість виведення каталогу «Клієнти»;
- ✓ Можливість додавання даних про нового клієнта;
- ✓ Можливість редагування вибраного із списку клієнта;
- ✓ Можливість видалення вибраного клієнта із системи.

Працівники:

- ✓ Можливість виведення каталогу «Працівники»;
- ✓ Можливість додавання даних про нового працівника;
- ✓ Можливість редагування вибраного із списку працівника;
- ✓ Можливість видалення вибраного працівника із системи.

Країни:

- ✓ Можливість виведення каталогу «Країни»;
- ✓ Можливість додавання даних про нову країну;
- ✓ Можливість редагування вибраної із списку країни;
- ✓ Можливість видалення даних вибраної країни.

Міста:

- ✓ Можливість виведення каталогу «Міста»;
- ✓ Можливість додавання даних про нове місто;
- ✓ Можливість редагування вибраного із списку міста;
- ✓ Можливість видалення даних вибраного міста.

Послуги:

- ✓ Можливість виведення каталогу «Послуги»;

- ✓ Можливість додавання даних про нову послугу;
- ✓ Можливість редагування вибраної із списку послуги;
- ✓ Можливість видалення даних вибраної послуги.

Подорож:

- ✓ Можливість виведення каталогу «Подорожі»;
- ✓ Можливість додавання даних про нову подорож для вибраного із списку клієнта;
- ✓ Можливість редагування вибраної із списку подорожі в разі, якщо попередні дані були введені неправильно;
- ✓ Можливість видалення даних вибраної подорожі.

Звітність:

a) за вибраною датою, де закінчується подорож виводиться інформація про всі подорожі клієнтів;
   1) номер;
   2) Інформація про клієнта;
   3) Інформація про послуги;
   4) Інформація про менеджера, що оформляв подорож;
   5) Ціна, в яку обійшлася подорож.
b) за вибраною послугою;
   1) номер;
   2) Інформація про клієнта;
   3) Дата відпустки;
   4) Інформація про менеджера, що оформляв подорож;
   5) Вартість відпустки.

1.7 Висновок

У ході аналізу поточного стану інформаційної системи в області діяльності туристичного агентства були розглянуті можливі проблеми та помилки при

організації роботи туристичного агентства, причини виникнення та способи їхнього відстеження. Було зроблено опис технологій обробки інформації. Зроблено SWOT-аналіз існуючої інформаційної системи. На основі отриманих даних було проведено постановку задачі на створення інформаційної системи, основне завдання якої полягає у автоматизації бізнес процесів діяльності туристичного агентства.

Підводячи підсумки про вище перелічені системи автоматизації діяльності туристичного агентства, треба сказати, що всі платформи є досить хорошими для організації роботи туристичного агентства, але нажаль вони спрямовані на отримання прибутку від користувачів.

Розроблену систему без проблем можна копіювати та переносити на інші комп’ютери на яких встановлена операційна система Windows та установлена бібліотека класів .NET framework 4.7 або вище.

2 РОЗРОБКА

2.1 Аналіз вимог. Use-case діаграми. Основні прецеденти

Згідно з поставленою задачею можна висунути функціональний список вимог до проекту (табл. 2.1.).

Таблиця 2.1 – Функціональні вимоги до додатку «Інформаційна система туристичного агентства»

Вимоги	**Опис**
REQ-1	Система повинна дозволяти здійснювати реєстрацію користувача в програмі
REQ-2	Система повинна вести журнал подій, що відбулися в програмі
REQ-3	Система повинна дозволяти користувачеві додати та редагувати інформацію про облікові записи користувачів
REQ-4	Система повинна дозволяти користувачеві додати та редагувати інформацію про клієнтів
REQ-5	Система повинна дозволяти користувачеві додати та редагувати інформацію про менеджерів
REQ-6	Система повинна дозволяти користувачеві додати та редагувати інформацію про країни
REQ-7	Система повинна дозволяти користувачеві додати та редагувати інформацію про міста
REQ-8	Система повинна дозволяти користувачеві додати та редагувати інформацію про послуги
REQ-9	Система повинна дозволяти користувачеві оформляти подорожі

REQ-10	Система повинна дозволяти формування звітність по клієнтах та послугах

Таблиця 2.2 – Нефункціональні вимоги до додатка «Інформаційна система туристичного агентства»

Вимоги	**Опис**
REQ-12	Додаток повинен мати простий дизайн та зручну навігації
REQ-13	Поля повинні бути не порожніми, унікальними відносно вже існуючих записів

Таблиця 2.3 – Актори та цілі додатка «Інформаційна система туристичного агентства»

Актори	**Цілі**
Користувач	Мета користувача полягає в роботі з системою.
База даних	Мета бази даних полягає у зберігання інформації

Таблиця 2.4 – Опис варіантів використання додатка «Інформаційна система туристичного агентства»

Варіант використання	**Ім’я**	**Опис**
UC1	Ідентифікація в системі	Дозволяє користувачу пройти ідентифікацію в системі
UC2	Вивід каталогу користувачів	Дозволяє користувачеві з правами системного адміністратора вивести каталог всіх користувачів системи
UC3	Додати користувача	Дозволяє користувачеві додати нового користувача системи
UC4	Редагувати користувача	Дозволяє користувачеві редагувати інформацію вибраної із списку користувача
UC5	Видалити користувача	Дозволяє користувачеві видалити вибраного із списку користувача
UC6	Вивід каталогу клієнтів	Дозволяє користувачеві вивести каталог всіх клієнтів
UC7	Додати клієнта	Дозволяє користувачеві додати нового клієнта
UC8	Редагувати клієнта	Дозволяє користувачеві редагувати інформацію вибраного із списку клієнта
UC9	Видалити клієнта	Дозволяє користувачеві видалити вибраного із списку клієнта

UC10	Вивід каталогу менеджерів	Дозволяє користувачу вивести каталог всіх зареєстрованих у системі менеджерів
UC11	Додати менеджера	Дозволяє користувачу додати інформацію про менеджера
UC12	Редагувати менеджера	Дозволяє користувачеві редагувати інформацію вибраного із списку менеджера
UC13	Видалити менеджера	Дозволяє користувачеві видалити вибраного менеджера
UC14	Вивід каталогу країн	Дозволяє користувачу вивести список всіх країн
UC15	Додати країну	Дозволяє користувачу додати нову країну
UC16	Редагувати країну	Дозволяє користувачеві редагувати інформацію вибраної із списку країни
UC17	Видалити країну	Дозволяє користувачеві видалити вибрану із списку країну
UC18	Вивід каталогу міст	Дозволяє користувачу вивести список всіх туристичних міст
UC19	Додати місто	Дозволяє користувачу додати нове місто
UC20	Редагувати місто	Дозволяє користувачеві редагувати інформацію вибраного із списку міста
UC21	Видалити місто	Дозволяє користувачеві видалити вибране із списку місто
UC22	Вивід каталогу послуг	Дозволяє користувачу вивести список всіх послуг агентства

UC23	Додати послугу	Дозволяє користувачу додати нову послугу
UC24	Редагувати послугу	Дозволяє користувачеві редагувати інформацію вибраної із списку послуги
UC25	Видалити послугу	Дозволяє користувачеві видалити вибрану із списку послугу
UC26	Вивід каталогу подорожей	Дозволяє користувачу вивести список всіх подорожей
UC27	Додати подорож	Дозволяє користувачу додати нову подорож
UC28	Редагувати подорож	Дозволяє користувачеві редагувати інформацію вибраної із списку подорожі
UC29	Видалити подорож	Дозволяє користувачеві видалити вибрану із списку подорож
UC30	Звітність по клієнтах	Дозволяє користувачеві програми формувати звітність по клієнтах за вибраний період часу
UC31	Звітність по послугах	Дозволяє користувачеві програми формувати звітність по вибраній послузі
UC32	Вивід системних подій	Дозволяє вивести всі події, які відбулися в системі

З поставлених вимог (див. розділ 1) тепер можна виставити повний опис вимог із сценаріями, що будуть вхідними даними для візуального моделювання мовою UML.

UC1 Ідентифікація в системі

Актор: користувач.

Ціль актора: пройти ідентифікацію в системі.

Задіяний актор: база даних.

Передумова: користувач ввівши нікнейм на пароль натискає кнопку «Підтвердити».

Післяумова: система відображає головне вікно програми з меню.

UC2 Вивід каталогу користувачів

Актор: користувач.

Ціль актора: вивести інформацію про всіх користувачів системи.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Користувачі».

Післяумова: система відображає екран для виведення списку всіх зареєстрованих користувачів системи.

UC3 Додати користувача

Актор: користувач.

Ціль актора: додати нового клієнта.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про користувача натискає на кнопку «Додати».

Післяумова: система додає нового користувача та відображає екран із списком всіх користувачів.

UC4 Редагувати користувача

Актор: користувач.

Ціль актора: редагувати інформацію про вибраного користувача.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку користувача.

Післяумова: система відображає екран для редагування інформації про вибраного користувача.

UC5 Видалити користувача

Актор: користувач.

Ціль актора: видалити користувача із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку користувача.

Післяумова: система відображає екран з можливістю видалення даних про користувача.

UC6 Вивід каталогу клієнтів

Актор: користувач.

Ціль актора: вивести інформацію про клієнтів.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Клієнти».

Післяумова: система відображає екран для виведення списку всіх клієнтів.

UC7 Додати клієнта

Актор: користувач.

Ціль актора: додати нового клієнта.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про клієнта натискає на кнопку «Додати».

Післяумова: система додає нового клієнта і відображає екран із списком всіх клієнтів.

UC8 Редагувати клієнта

Актор: користувач.

Ціль актора: редагувати вибраного із списку клієнта.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку клієнта.

Післяумова: система відображає екран для редагування інформації про вибраного клієнта.

UC9 Видалити клієнта

Актор: користувач.

Ціль актора: видалити вибраного клієнта із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку клієнта.

Післяумова: система відображає екран з можливістю видалення даних клієнта.

UC10 Вивід каталогу менеджерів

Актор: користувач.

Ціль актора: вивести інформацію про всіх зареєстрованих в системі менеджерів.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Менеджери».

Післяумова: система відображає екран для виведення списку всіх менеджерів.

UC11 Додати менеджера

Актор: користувач.

Ціль актора: додати інформацію про нового менеджера.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про менеджера натискає на кнопку «Додати».

Післяумова: система додає нового менеджера і відображає екран із списком всіх менеджерів.

UC12 Редагувати менеджера

Актор: користувач.

Ціль актора: редагувати вибрану із списку менеджера.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку менеджера.

Післяумова: система відображає екран для редагування інформації про вибраного менеджера.

UC13 Видалити менеджера

Актор: користувач.

Ціль актора: видалити менеджера із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку менеджера.

Післяумова: система відображає екран з можливістю видалення даних про менеджера.

UC14 Вивід каталогу країн

Актор: користувач.

Ціль актора: вивести інформацію про всі країни.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Країни».

Післяумова: система відображає екран для виведення списку всіх країну.

UC15 Додати країну

Актор: користувач.

Ціль актора: додати нову країну.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про країну натискає на кнопку «Додати».

Післяумова: система додає нову країну та відображає екран із списком всіх країн.

UC16 Редагувати країну

Актор: користувач.

Ціль актора: редагувати вибрану із списку країну.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку країну.

Післяумова: система відображає екран для редагування інформації про вибрану країну.

UC17 Видалити країну

Актор: користувач.

Ціль актора: видалити країну із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку країну.

Післяумова: система відображає екран з можливістю видалення даних про країну.

UC18 Вивід списку міст

Актор: користувач.

Ціль актора: вивести інформацію про всі міста.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Міста».

Післяумова: система відображає екран для виведення списку всіх міст.

UC19 Додати місто

Актор: користувач.

Ціль актора: додати нове місто.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про туристичне місто, та натискає на кнопку «Додати».

Післяумова: система додає нове туристичне місто та відображає екран із списком всіх міст.

UC20 Редагувати місто

Актор: користувач.

Ціль актора: редагувати вибране із списку місто.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідне із списку місто.

Післяумова: система відображає екран для редагування інформації про вибране місто.

UC21 Видалити місто

Актор: користувач.

Ціль актора: видалити місто із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідне із списку місто.

Післяумова: система відображає екран з можливістю видалення даних про місто.

UC22 Вивід каталогу послуг

Актор: користувач.

Ціль актора: вивести інформацію про всі послуги.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Послуги».

Післяумова: система відображає екран для виведення списку всіх послуг.

UC23 Додати послугу

Актор: користувач.

Ціль актора: додати нову послугу.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про послугу, та натискає на кнопку «Додати».

Післяумова: система додає нову послугу та відображає екран із списком всіх послуг.

UC24 Редагувати послугу

Актор: користувач.

Ціль актора: редагувати вибрану із списку послугу.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку послугу.

Післяумова: система відображає екран для редагування інформації про вибрану послугу.

UC25 Видалити послугу

Актор: користувач.

Ціль актора: видалити послугу із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку послугу.

Післяумова: система відображає екран з можливістю видалення даних про послугу.

UC26 Вивід каталогу подорожей

Актор: користувач.

Ціль актора: вивести інформацію про всі подорожі.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Подорожі».

Післяумова: система відображає екран для виведення списку всіх подорожей.

UC27 Додати подорож

Актор: користувач.

Ціль актора: додати нову подорож.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про подорож, та натискає на кнопку «Додати».

Післяумова: система додає нову подорож та відображає екран із списком всіх подорожей.

UC28 Редагувати подорож

Актор: користувач.

Ціль актора: редагувати вибрану із списку подорож.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку подорож.

Післяумова: система відображає екран для редагування інформації про вибрану подорож.

UC29 Видалити подорож

Актор: користувач.

Ціль актора: видалити подорож із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку подорож.

Післяумова: система відображає екран з можливістю видалення даних про подорож.

UC30 Звітність по клієнтах

Актор: користувач.

Ціль актора: вивести детальну інформацію по клієнтах за вибрану дату.

Задіяний актор: база даних.

Передумова: користувач натискає на пункт меню «По клієнтах».

Післяумова: система відображає екран з можливістю формування звітності по клієнтах.

UC31 Звітність по послугах

Актор: користувач.

Ціль актора: вивести детальну інформацію по вибраній із списку послузі.

Задіяний актор: база даних.

Передумова: користувач натискає на пункт меню «По послугах».

Післяумова: система відображає екран з можливістю формування звітності по вибраній із списку послузі.

UC32 Вивід системних подій

Актор: користувач.

Ціль актора: вивести список всіх подій в системі.

Задіяний актор: база даних.

Передумова: користувач вибирає пункт меню «Системний журнал».

Післяумова: система відображає екран із списком всіх системних подій.

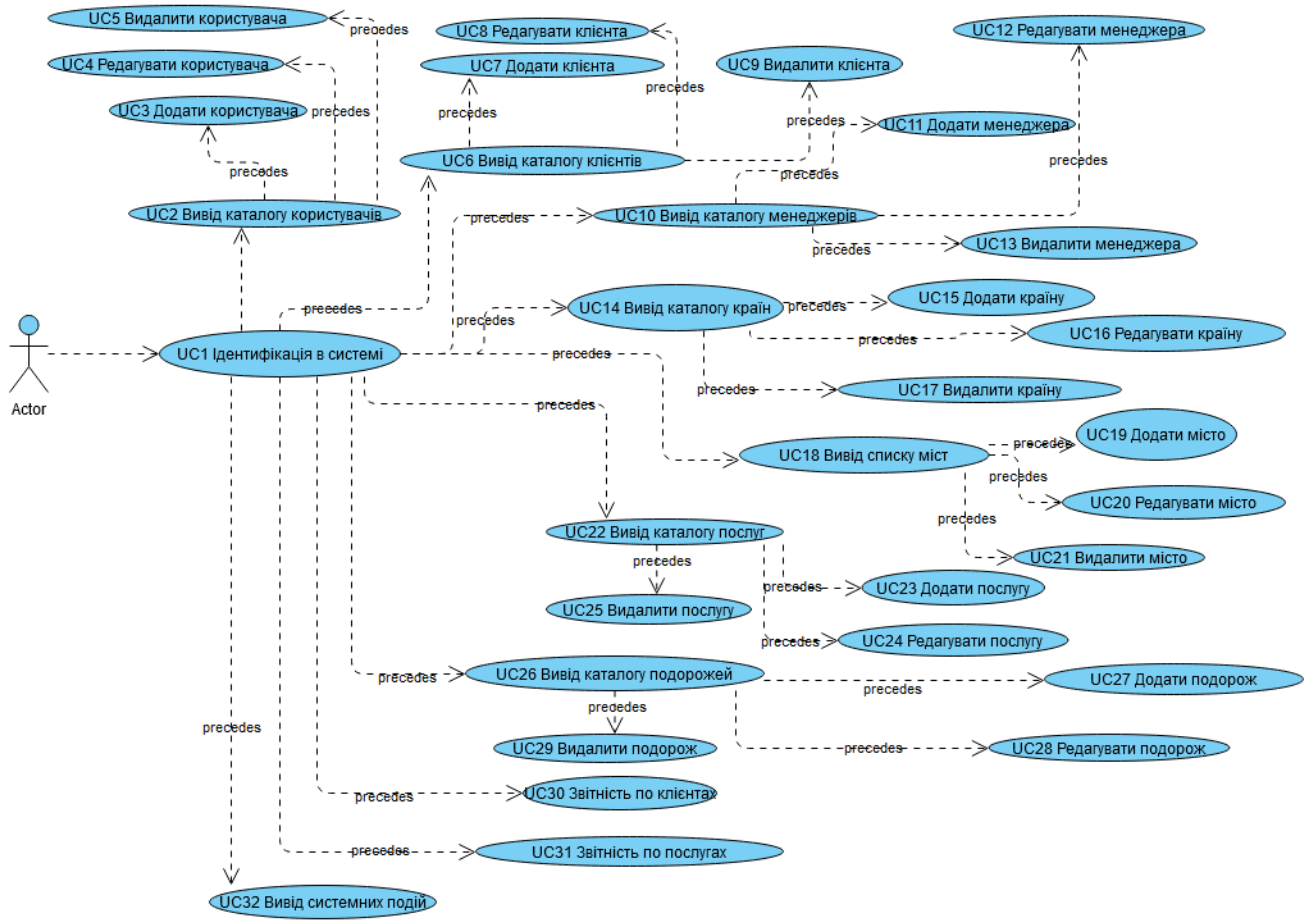


Рисунок 2.1 – Діаграма use-case

2.2 Архітектура проекту

Розроблений продукт повинен відповідати характеристикам якості, таким як: стійкість, корисність, доступність, масштабованість, відкритість, гнучкість, можливість тестування. Це вимагає від процесу розробки додаткові обмеження/правила, а саме:

- дотримання шаблонів і стилів;
- документування розробки на різних рівнях;
- тестування компонентів, окремих модулів, підсистем;
- управління проектами, процесами.

З урахуванням вимог до забезпечення стійкості та гнучкості системи при її розробці було обрано шаблон Layers, який розбиває систему на дві частини: клієнт та сервер.

Проектування системи буде покладатись на предметну область (DDD підхід) та принципи SOLID [5].

При розробці клієнта був обраний користувацький інтерфейс Windows Forms.

Сервер, в свою чергу, буде складатись з таких модулів:

1) BLL (англ. Business Logic Layer) – логіка та всі необхідні обчислення додатку на мові бізнесу;
2) DAL (англ. Data Acces Layer) – рівень доступу до даних.
3) DB (англ. Data Base) – база даних для зберігання даних.

З урахування всіх вище перерахованих шаблонів структура проекту буде виглядати наступним чином:

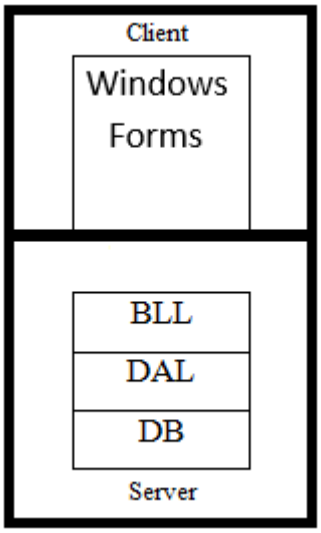


Рисунок 2.2 – Структура проекту

Кожен компонент буде реалізувати контракт (інтерфейс), який надає на гнучкість компоненту.

Оскільки бізнес постійно і відносно швидко змінюється, то кожен із компонентів повинен швидко адаптуватись під ці зміни. Для цього використаємо шаблон Dependency Injection (DI).

2.3 Особливості розробки бази даних. ERD діаграма з описанням сутностей

Схема "сутність-зв'язок" (також ERD або ER-діаграма) - це різновид блок-схеми, де показано, як різні "сутності" (люди, об'єкти, концепції і так далі) пов'язані між собою всередині системи. ER-діаграми найчастіше застосовуються для проектування та налагодження реляційних баз даних у сфері освіти, дослідження та розробки програмного забезпечення та інформаційних систем для бізнесу.

ER-діаграми (або ER-моделі) покладаються на стандартний набір символів, включаючи прямокутники, ромби, овали та сполучні лінії для відображення сутностей, їх атрибутів та зв'язків. Ці діаграми влаштовані за тим самим принципом, як і граматичні структури: сутності виконують роль іменників, а зв'язку — дієслів.

ER-діаграми - "родичі" схем структури даних (DSD), де замість зв’язків між самими сутностями відображається відношення між елементами всередині них. ER-діаграми часто використовуються у поєднанні з діаграмами DFD, що схематично показують рух потоків інформації в рамках процесу або системи.

У ER-моделях та моделях даних зазвичай виділяють до трьох рівнів деталізації:

Концептуальна модель даних – схема найвищого рівня з мінімальною кількістю подробиць. Перевага цього підходу полягає у можливості відобразити загальну структуру моделі та всю архітектуру системи. Менш масштабні системи можуть обійтися без цієї моделі. І тут можна відразу переходити до логічної моделі.

Логічна модель даних: містить більш детальну інформацію, ніж концептуальна модель. На цьому рівні визначаються докладніші операційні та транзакційні сутності. Логічна модель залежить від технології, у якій застосовуватиметься.

Фізична модель даних: на основі кожної логічної моделі даних можна становити одну або дві фізичні моделі. В останніх має бути достатньо технічних подробиць для складання та впровадження самої бази даних.

Можна звернути увагу на той факт, що схожі рівні масштабу та деталізації зустрічаються і в інших видах схем (наприклад, у діаграмах DFD), проте дана класифікація відрізняється від трьохсхемного підходу у розробці ПЗ, де розподіл інформації здійснюється за дещо іншим принципом. Щоправда, іноді розробники застосовують ER-діаграми з додатковими ієрархіями, якщо дизайн бази даних потребує більше інформаційних рівнів. Наприклад, розробник може додати нові групи за принципом розширення вгору (суперкласи) та вниз (підкласи). А саме:

- ✓ *Лише реляційні дані*. Слід чітко розуміти, що мета ER-діаграм – показати зв'язки та відносини між елементами, тому вони відображають лише реляційну структуру.
- ✓ *Лише для структурованих даних*. Дані мають бути чітко розбиті на поля, стовпці та рядки, інакше користі від ER-діаграми буде мало. Це стосується і частково структурованих даних, оскільки лише деякі з них будуть придатними для роботи.
- ✓ *Складність інтеграції з базою даних*. Застосування ER-моделей для інтеграції з існуючою базою даних — непросте завдання через різницю в архітектурі.

Згідно завдання будуємо ERD діаграму.

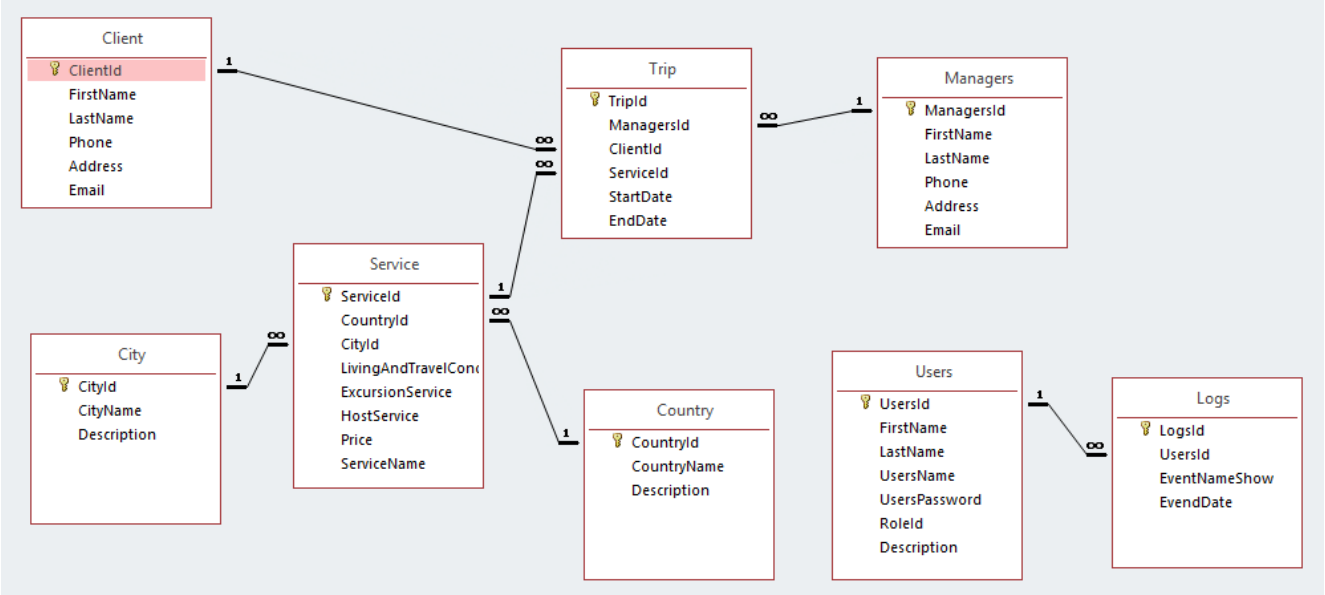


Рисунок 2.3 – ERD діаграма

Як ми бачимо на рис. 2.3 наша база даних складається із 8 сутностей. Кожна сутність має свою таблицю, а саме:

1. Таблиця «Country» – зберігає інформацію про країни.
2. Таблиця «City» – зберігає інформацію про міста.
3. Таблиця «Client» - зберігає інформацію про клієнтів.
4. Таблиця «Logs» - зберігає інформацію про активність користувачів системи і їхні дії.
5. Таблиця «Managers» - зберігає інформацію про менеджерів.
6. Таблиця «Service» - зберігає інформацію про всі послуги.
7. Таблиця «Trip» - зберігає інформацію про подорожі.
8. Таблиця «Users» - містить інформацію про всіх користувачів системи та їхні облікові дані.

2.4 Особливості розробки рівня BLL

Бізнес-логіка – у розробці інформаційних систем – сукупність правил, принципів, залежностей поведінки об'єктів предметної галузі (області людської діяльності, яку система підтримує). Інакше можна сказати, що бізнес-логіка — це реалізація правил та обмежень операцій, що автоматизуються. Є синонімом

терміну "логіка предметної області" (англ. domain logic). Бізнес-логіка визначає правила, яким підпорядковуються дані предметної області.

Простіше кажучи, бізнес-логіка – це реалізація предметної галузі в інформаційній системі. До неї відносяться: формули розрахунку щомісячних виплат з позик (у фінансовій індустрії), автоматизована відправка повідомлень електронної пошти керівнику проекту після закінчення частин завдання всіма підлеглими (у системах управління проектами), відмова від готелю при скасуванні рейсу авіакомпанією (у туристичному бізнесі)) і т.д.

У фазі бізнес-моделювання та розробки вимог бізнес-логіка описується у вигляді:

- ✓ тексту;
- ✓ концептуальних аналітичних моделей предметної галузі (онтології);
- ✓ бізнес-правил;
- ✓ різноманітних алгоритмів;
- ✓ діаграм діяльності;
- ✓ графів та діаграм переходу станів;
- ✓ моделей бізнес-процесів.

У фазі аналізу та проектування системи бізнес-логіка втілюється в різних діаграмах мови UML або подібних до неї. У фазі програмування бізнес-логіка втілюється в коді класів та їх методів, у разі використання об'єктно-орієнтованих мов програмування, або процедур та функцій у разі застосування процедурних мов.

На жаргоні розробників програмного забезпечення «бізнес-логікою» також називаються програмні модулі, що її реалізують, та рівень системи, на якому ці модулі знаходяться (англ. business logic layer).

У багаторівневих (багатошарових) інформаційних системах цей рівень взаємодіє з нижчим рівнем інфраструктурних сервісів (англ. infrastructure layer), наприклад, інтерфейсом доступу до бази даних або файлової системи (англ. data-access layer, DAL) і рівнем сервісів додатка (англ. application services layer), який, своєю чергою, взаємодіє з рівнем користувальницького інтерфейсу (англ. user interface layer) чи зовнішніми системами.

BLL буде обробляти речі, які є частиною бізнес-сфери, а не частиною бази даних і не частиною користувальницького інтерфейсу.

В даному проекті реалізовано один клас для формування звітності по клієнтах, які подорожували, а також інформацію про вибраній послузі.

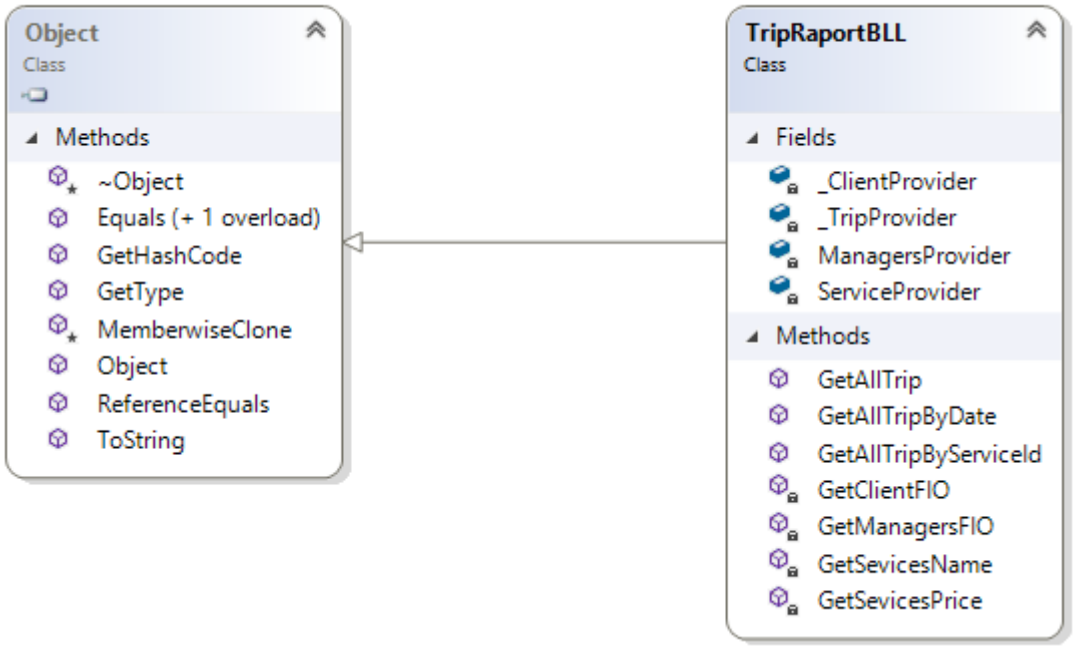


Рисунок 2.4 – Діаграма класу BLL «Інформаційна система туристичного агентства»

2.5 Особливість реалізації бізнес логіки – діаграма домена.

Будуємо діаграму класів домена. Кожен клас описує конкретну таблицю бази даних для зручного опрацювання даних.

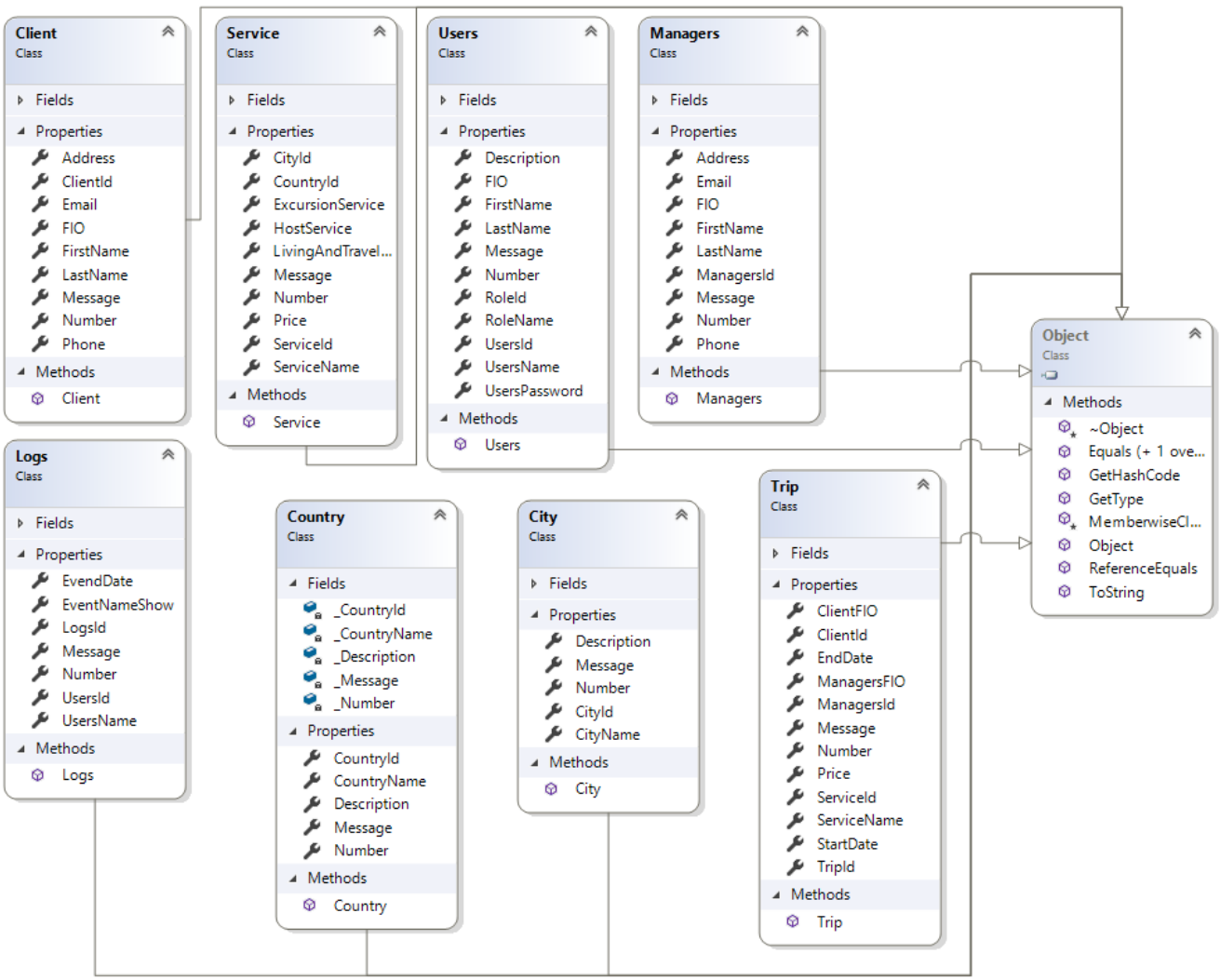


Рисунок 2.5 – Діаграма класів домена

2.6 Особливості розробки рівня UI

User interface (UI) елементи – це частини, які дизайнери використовують для створення програм або веб-сайтів. Вони додають інтерактивність в інтерфейс користувача, надаючи користувачеві точки зіткнення при навігації по них. Кнопки, смуги прокручування, пункти меню та чекбокси.

Інтерфейсу користувача (UI) використовує UI елементи для створення візуальної мови і забезпечення узгодженості продукту, що робить його зручним для користувача і простим у навігації без особливих зусиль з боку користувача.

UI елементи зазвичай поділяються на одну з наступних чотирьох країну:

- Елементи керування введенням (Input Controls) – дозволяють користувачам вводити інформацію до системи.
- Компоненти навігації (Navigation Components) – допомагають користувачам переміщатися продуктом або веб-сайтом. Загальні навігаційні компоненти включають панелі вкладок та головне меню програми.
- Інформаційні компоненти (Informational Components) – діляться інформацією з користувачем.
- Контейнери (Containers) – містять зв’язаний контент разом.

В даному проекті для розробки користувацького інтерфейсу я використав Windows Forms.

Windows Forms - це платформа користувача інтерфейсу для створення класичних додатків Windows. Вона забезпечує один з найефективніших способів створення класичних додатків за допомогою візуального конструктора в Visual Studio.

На рис. 2.6 показана діаграма програми «Інформаційна система туристичного агентства» з правами адміністратора системи.

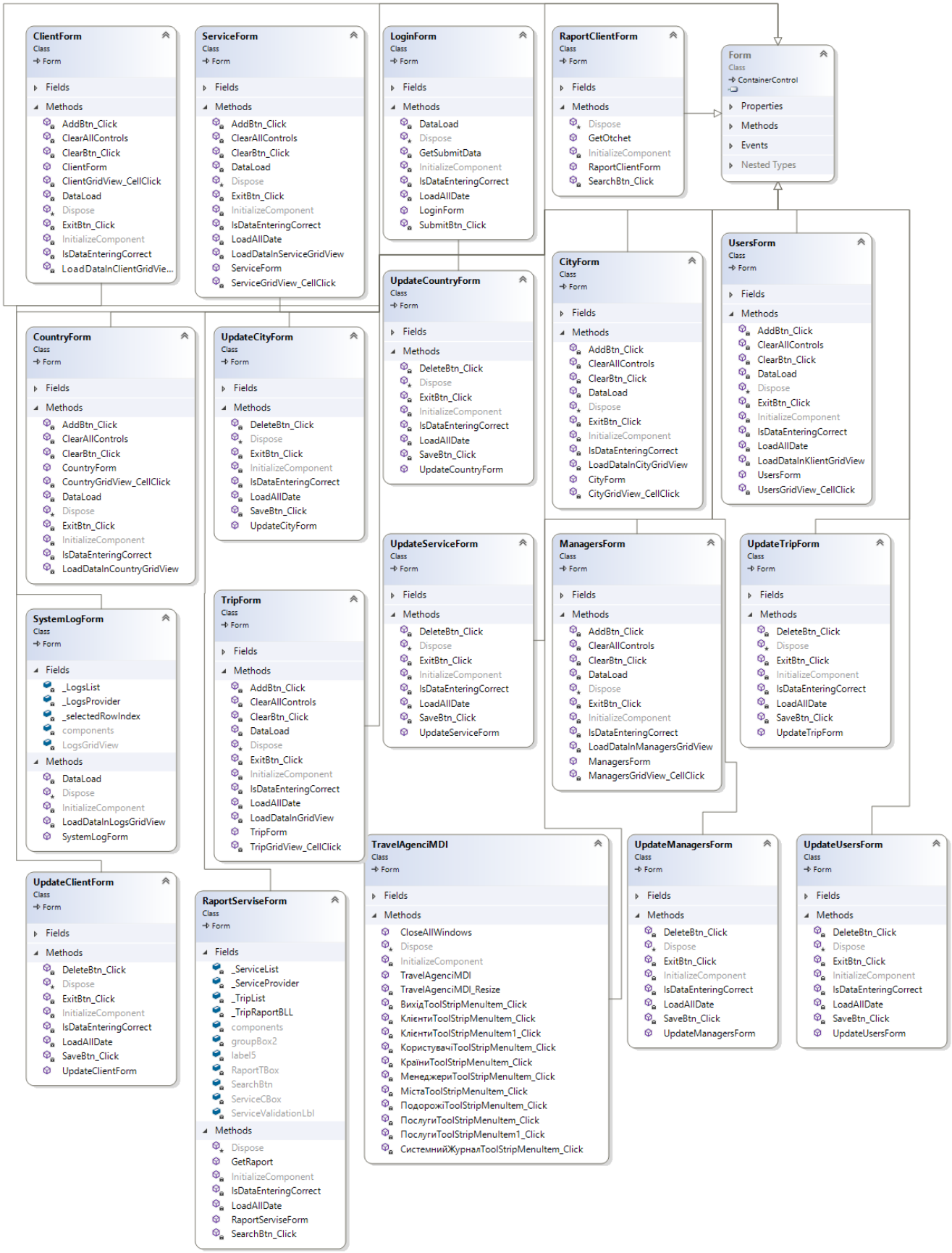


Рисунок 2.6 – Діаграма інтерфейсу програми «Інформаційна система туристичного агентства»

2.7 Особливості розробки DAL

Шар доступу до даних (Data Access Layer – DAL) у програмному забезпеченні – це шар комп'ютерної програми, який надає спрощений доступ до даних, що зберігаються у постійному сховищі якогось типу, такому як реляційна база даних. Цей акронім в основному використовується в оточенні Microsoft.NET.

DAL може повертати посилання на об'єкт (у термінах об'єктно-орієнтованого програмування) з його атрибутами замість рядків полів із таблиці бази даних. Це дозволяє створювати клієнтські (або модулі користувача) модулі з більш високим рівнем абстракції. Така модель може бути реалізована шляхом створення класу з методами доступу до даних, які безпосередньо посилаються на відповідний набір процедур бази даних. Інша реалізація може потенційно отримувати або записувати записи або з файлової системи. DAL приховує складність сховища даних, що лежить в основі даних.

Замість використання таких команд як «створити», «видалити» або «оновити» в певній таблиці в базі, клас і кілька процедур, що зберігаються, можуть бути створені в базі. Ці процедури можуть викликатися методом усередині класу, який поверне об'єкт, що містить запитані значення. Або команди створення, видалення та оновлення можуть бути виконані всередині простих функцій, що зберігаються у шарі доступу до даних.

Також методи бізнес-логіки із програми можуть бути співвіднесені до шару доступу до даних.

Для роботи з базою даних було реалізовано 8 класів. Назва кожного класу починається із відповідній їй назві таблиці в базі даних та закінчується приставкою «Provider».

На рис. 2.7 приведено діаграму класів з методами для роботи з базою даних.

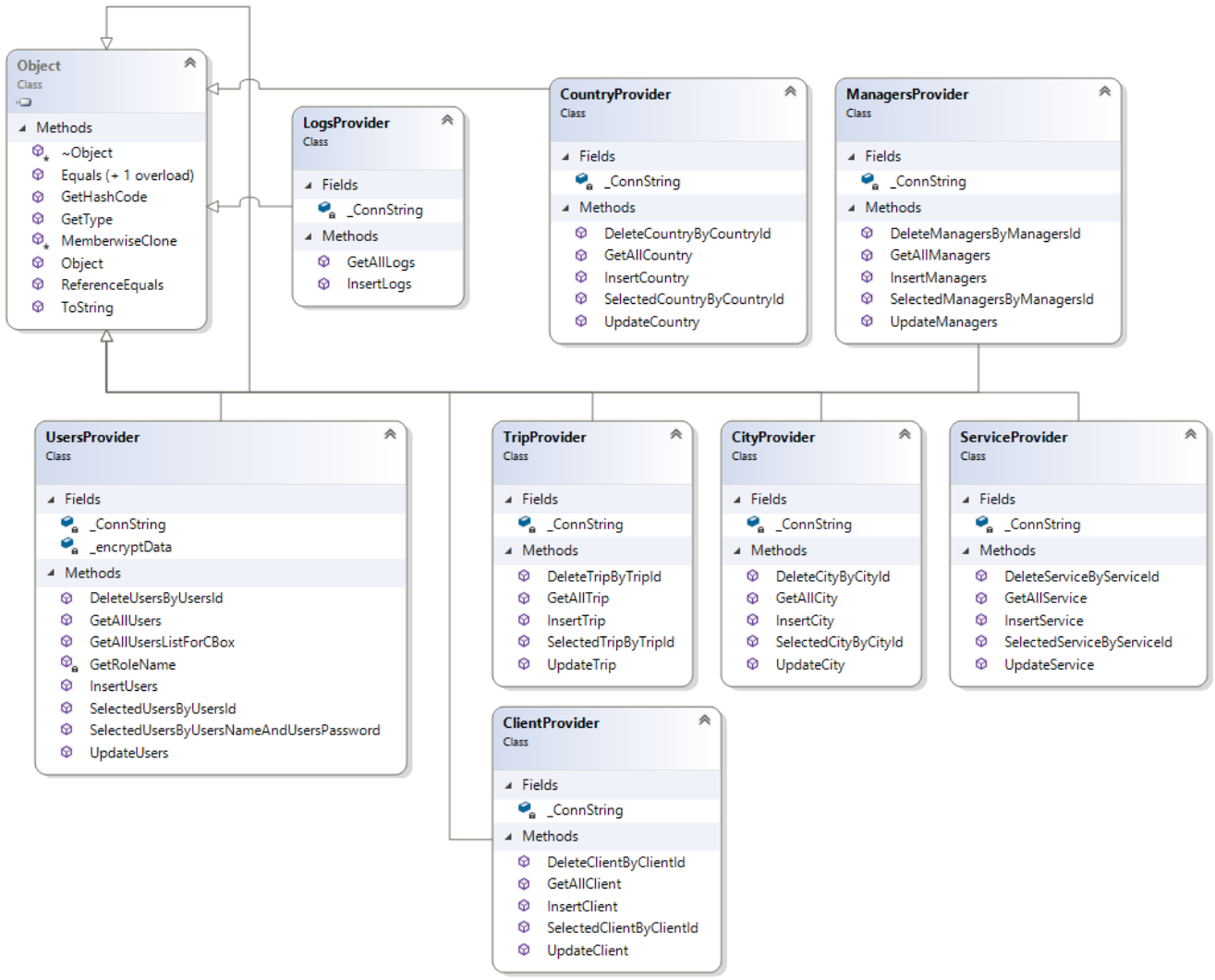


Рисунок 2.7. Діаграма класів з методами для роботи з базою даних

2.8 Висновок

Побудова сучасних інформаційних система займає дуже багато часу. Вона починається з аналізу предметної області, детального планування системи, описання вимог, моделювання її поведінки за допомогою uml діаграм. Визначається шаблони, які будуть використовуватись при розробці.

Після планування починається стадія розробки. Розробка починається зі створення користувацького інферфейсу і закінчується базою даних.

При розробці слід враховувати, що вимоги змінюються швидко і потрібно будувати систему так, щоб вона була гнучкою.

Розробка системи виконується по окремим компонентам. Кожний створений компонент потрібно детально тестувати, щоб мінімізувати помилки на наступних етапах розробки.

Якщо дотримуватися всіх вищеперерахованих вимог, то можна побудувати гнучку і стійку інформаційну систему.

2 РЕАЛІЗАЦІЯ

3.1 Вибір технологій

Для побудови користувацького інтерфейсу по шаблону будемо використовувати WinForms.

Для розробки інформаційної бази використовувався Microsoft Access. система управління базами даних, програма, що входить до складу пакету офісних програм Microsoft Office. Має широкий спектр функцій, включаючи зв'язані запити, сортування по різних полях, зв'язок із зовнішніми таблицями і базами даних.

Microsoft Access, володіючи всіма рисами класичної СУБД, надає і додаткові можливості. Access - це не тільки потужна, гнучка і проста у використанні СУБД, але і система для розробки працюючих з базами даних додатків. За допомогою Access можна створити додаток, що працює в середовищі Windows і повністю відповідає потребам по управлінню даними. Використовуючи запити, є можливість вибирати і обробляти що зберігається в таблицях інформацію.

Для роботи з Базою даних реалізовуємо класи (рис.3.1). В кожному класі є методи, за допомогою яких можна: вибирати, вставляти, редагувати та видаляти дані. Також є методи, які працюють з вибіркою даних із декількох таблиць.

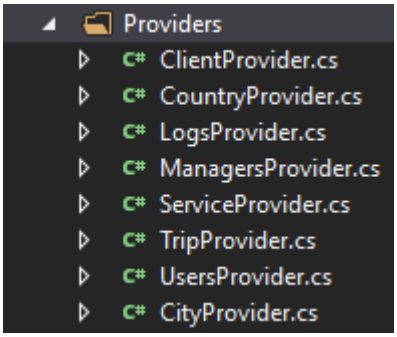


Рисунок 3.1 – Класи для роботи з базою даних

3.2 Результати функціонального тестування розробленого додатку

Для того, щоб провести тестування необхідно запустити програму із назвою «TravelAgenci». Після запуску програми буде виведено вікно, де користувачу буде запропоновано ввести ім’я та пароль (рис.3.2.).

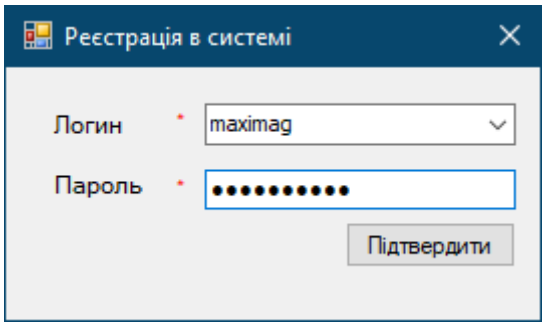


Рисунок 3.2 – Реєстрація користувача в системі.

Якщо ж дані буде введена інформація буде не коректною, програма буде попереджувати про це користувача відповідним повідомленням (рис.3.3.).

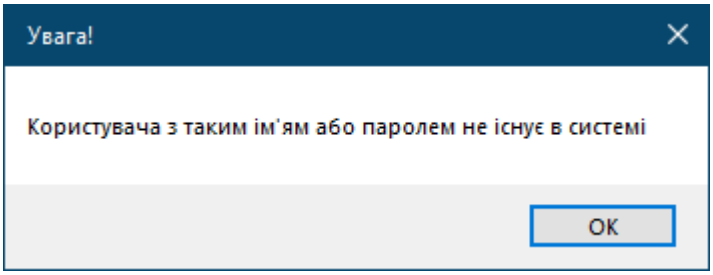


Рисунок 3.3 – Попередження про неправильне введення даних.

Для швидшого пошуку можна вибрати ім’я із випадаючого списку або під час введення даних програма поставить у верх списку те ім’я, яке буде вводитись з клавіатури.

Після успішної ідентифікації користувача системи буде відкрите головне вікно програми з основним меню (рис.3.4.).

Рисунок 3.4 – Головне вікно програми.

Так, як ми здійснили вхід з правами системного адміністратора, для нас відкриті всі можливі функції системи.

Для продовження роботи створимо країну, та введемо всі необхідні дані про неї (рис 3.5.). Також в даному вікні відображається список всіх країнт, які вже існують в системі.

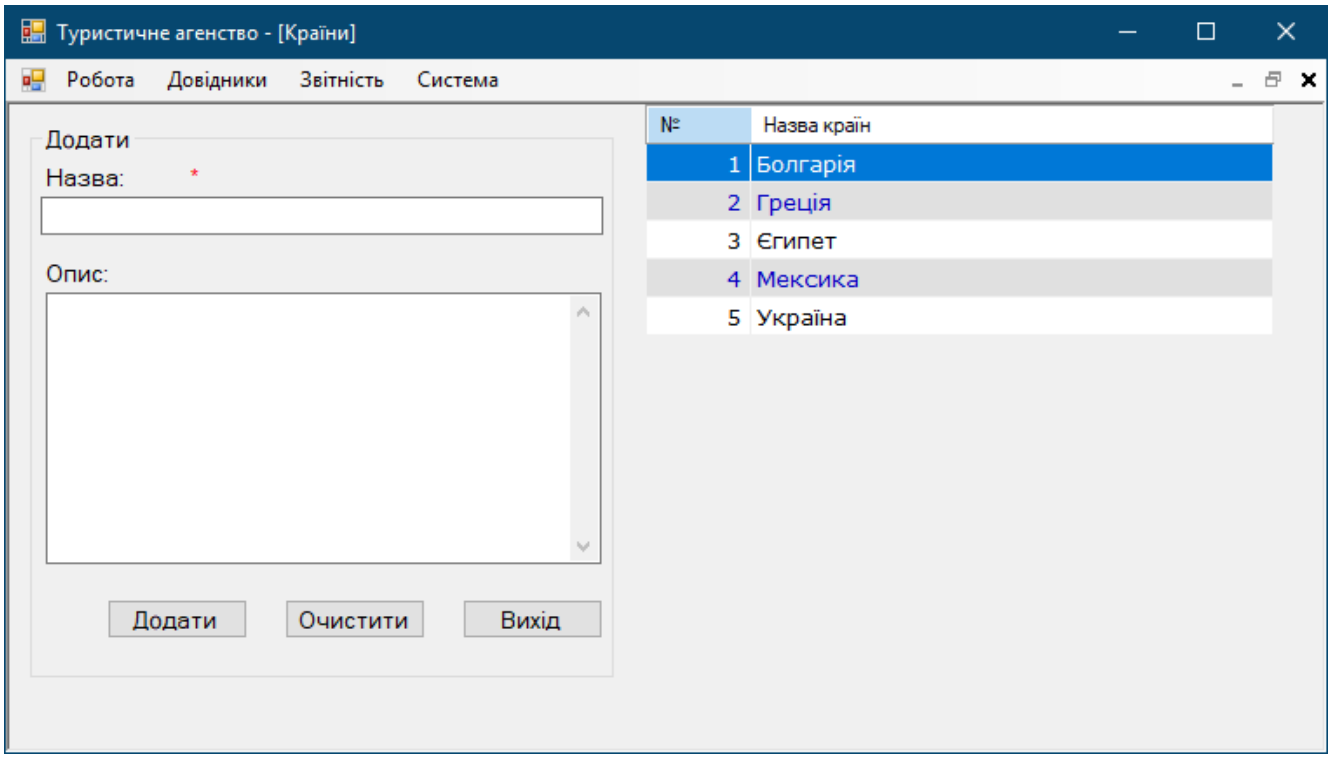


Рисунок 3.5 – Вікно для введення даних про країну

Система перевіряє всі обов'язкові поля на коректність вводу інформації. Обов'язкові поля позначаються зірочкою. В даному випадку недопустимим буде введення пустого поля в систему.

Також реалізована можливість редагування та видалення даних у випадку якщо це є необхідним. На рис 3.6. зображено вікно для редагування даних про країну.

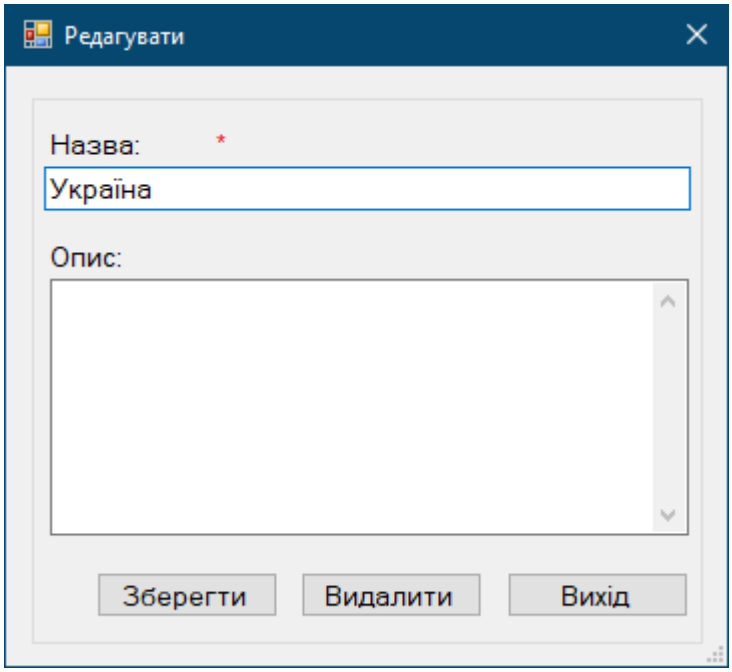


Рисунок 3.6 – Вікно для редагування даних вибраної країни

У системі є можливість додати послуги туристичного агентства. Для цього необхідно перейти по головному меню «Довідники» –> «Послуги»., після чого відкриється відповідне вікно. Також в цьому вікні є можливість додати нову послугу (рис. 3.7.).

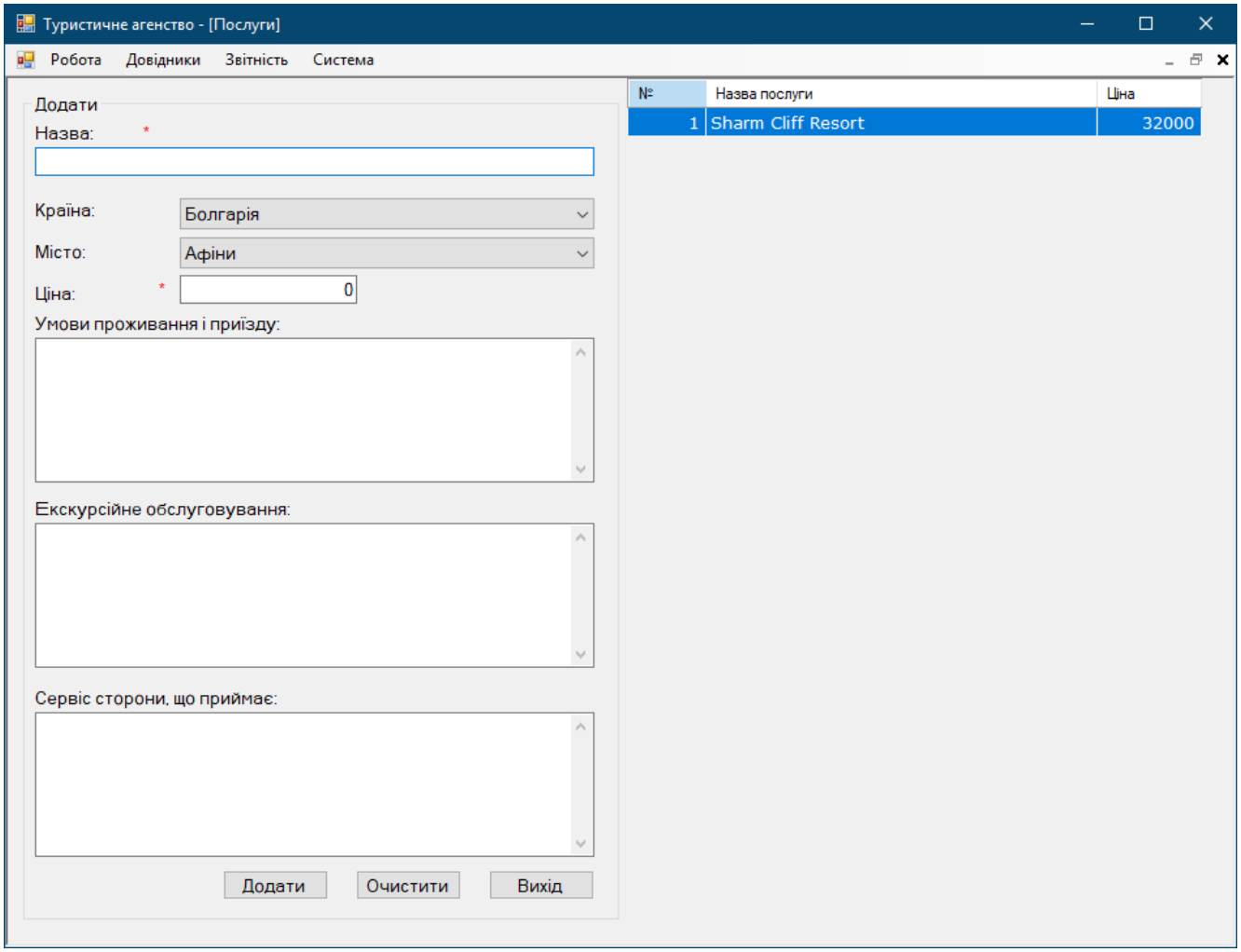


Рисунок 3.7 – Вікно для додавання нової послуги

Також у випадку помилки, дані про послуги можна редагувати, або якщо вона є не актуальною, то її можна видалити з системи (рис. 3.8).

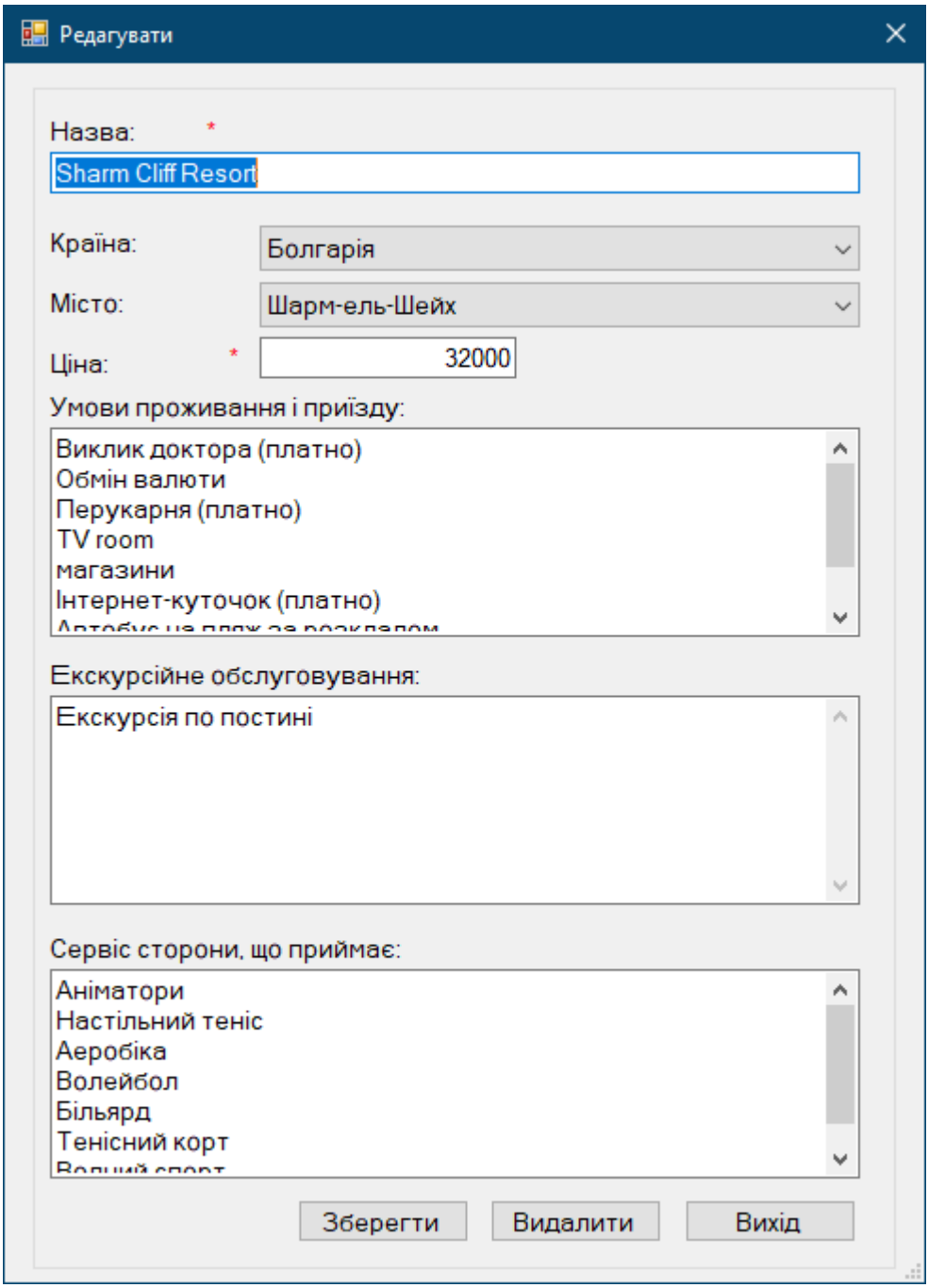


Рисунок 3.8. Вікно для редагування послуги

Для ведення інформаційної бази клієнтів реалізована можливість додавання та переглядання інформації про них. Щоб це зробити необхідно перейти по меню «Довідники» –> «Клієнти». Після цього виведеться вікно зі списком всіх зареєстрованих клієнтів (рис. 3.9).

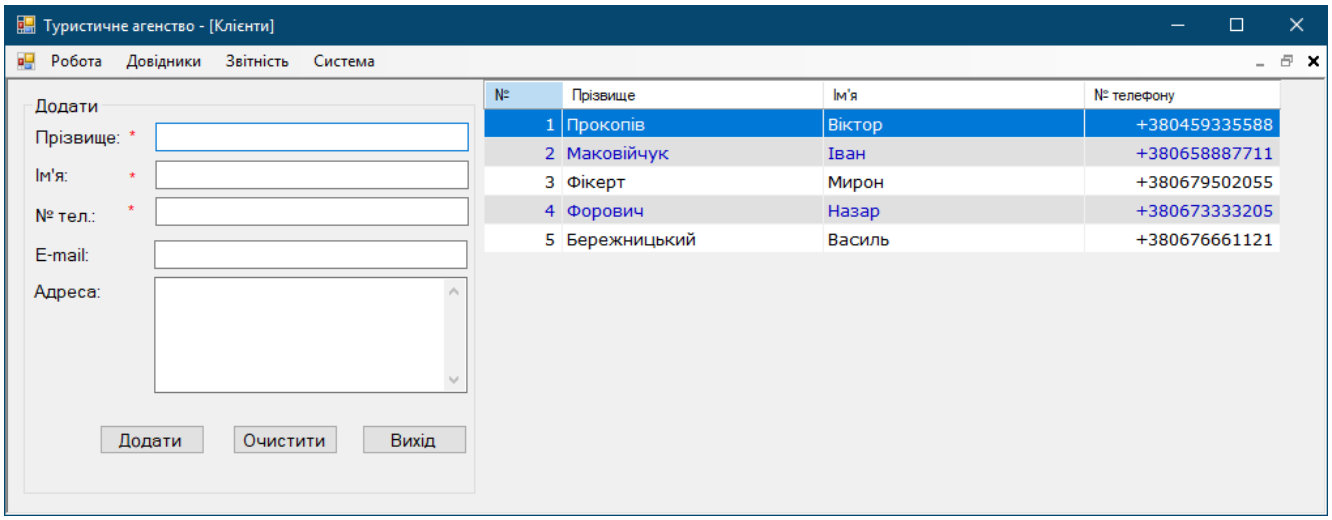


Рисунок 3.9 – Вікно для додавання інформації про клієнтів

В разі необхідності програму можна відредагувати або видалити. Для цього у списку справа потрібно натиснути лівою кнопкою на по необхідному запису, після чого з’явиться вікно з можливістю редагування/видалення інформації про клієнта (рис. 3.10).

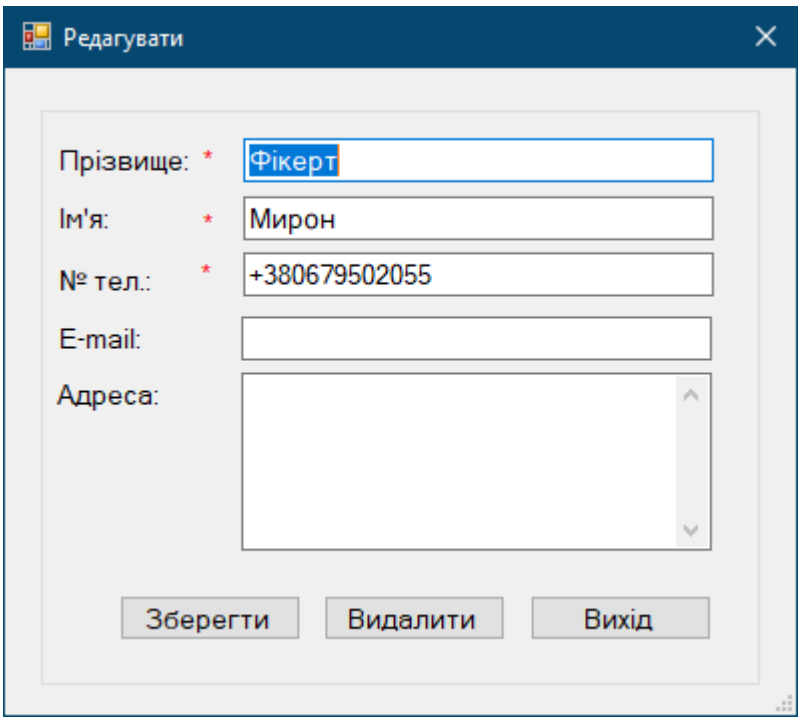


Рисунок 3.10. Вікно для редагування інформації про клієнта

Вікна для опрацювання інформації про менеджерів є схожими до вінок для опрацювання даних про клієнтів.

Далі внесення інформації про подорожі необхідно перейти по меню «Робота» –> «Подорожі», після чого відкриється вікно з можливістю додавання інформації про подорожі. (рис. 3.11).

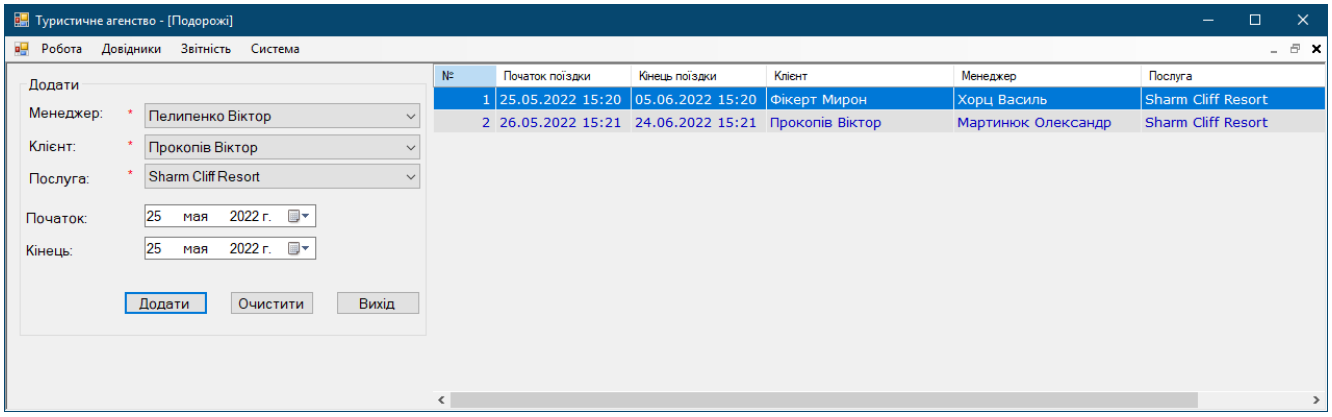


Рисунок 3.11 – Вікно для додавання інформації про подорожі

У програмі є функції, які може виконувати тільки системний адміністратор. Для цього необхідно зайти в систему з обліковим записом в якого роль системного адміністратора. Додатковою функцією системи є формування звітності по клієнтах та послугах агентства, для цього необхідно скористатись головним меню програми «Звітність» (рис. 3.12 та 3.13). Для формування звітності необхідно вказати вибрати день, на який були назначені подорожі.

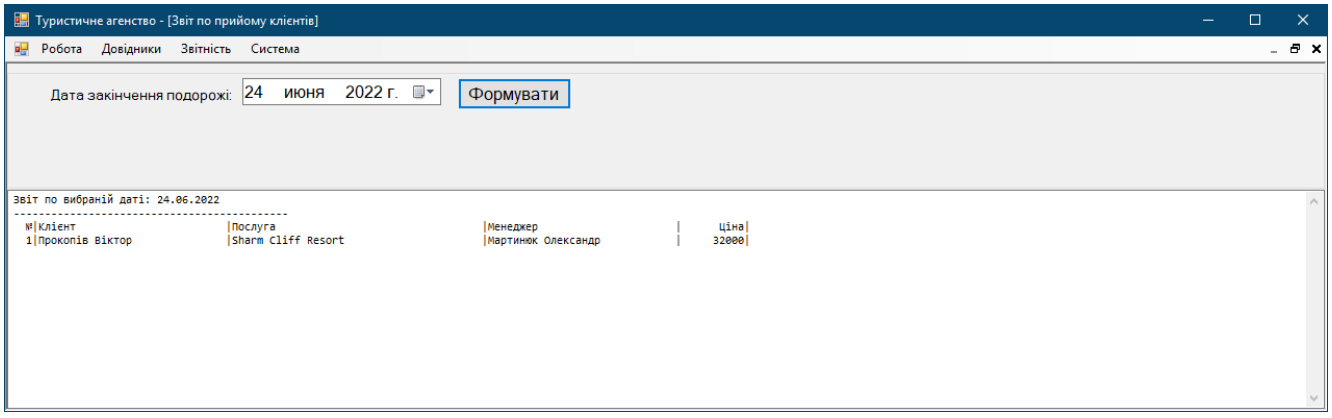


Рисунок 3.12 – Вікно для формування звітності по клієнтах

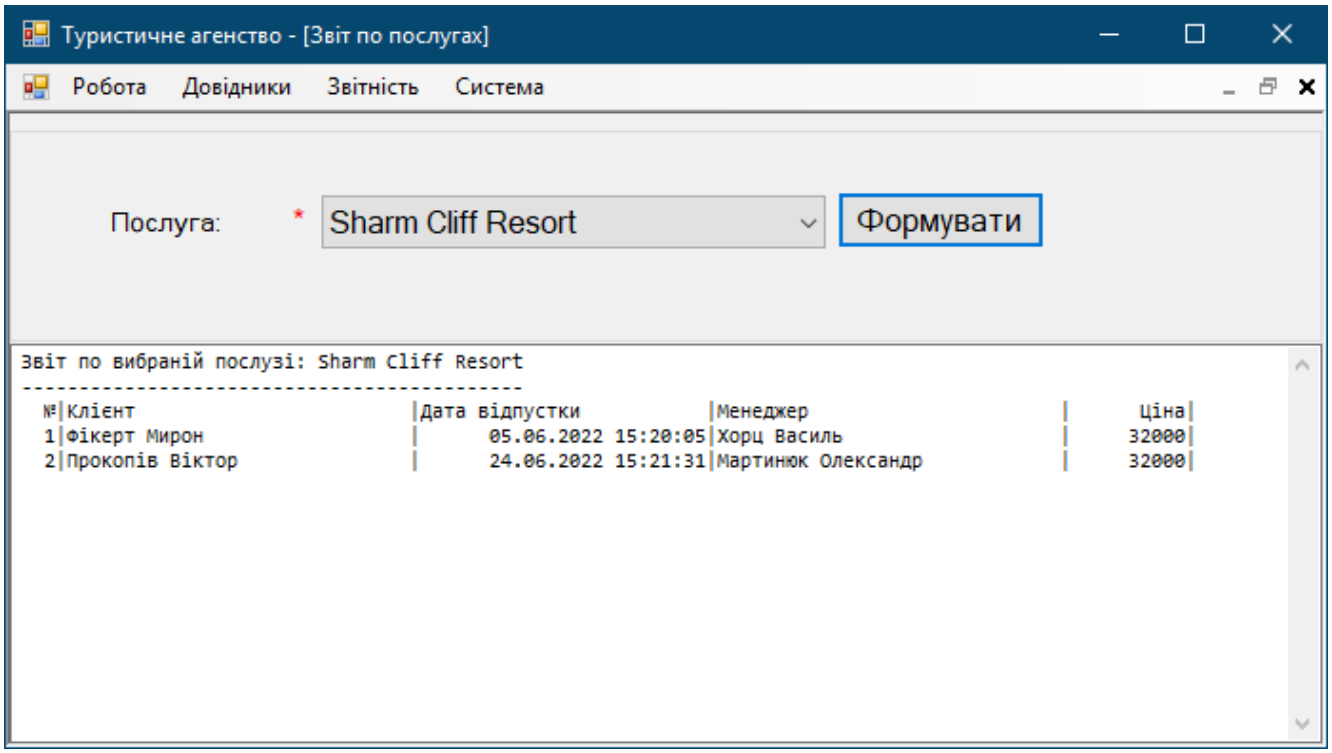
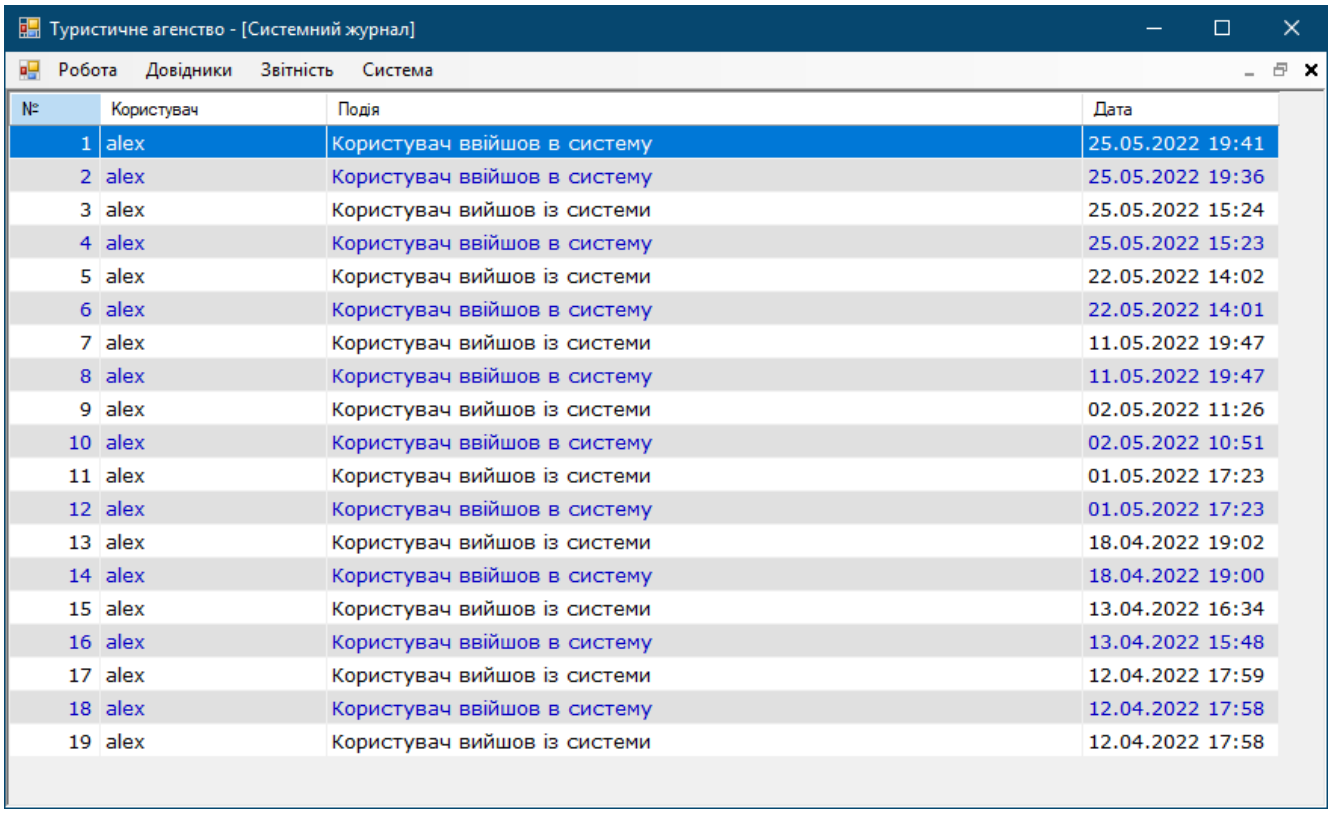


Рисунок 3.13 – Вікно для формування звітності по послугах

Також в системі можна бачити активність користувачів, та те, що вони робили в системі. Це можливо зробити за допомогою облікового запису, який має права системного адміністратора. Для цього перейдемо по меню «Система» –> «Системний журнал». В цьому вікні виводяться всі події системи (3.14).



№	Користувач	Подія	Дата
1	alex	Користувач ввійшов в систему	25.05.2022 19:41
2	alex	Користувач ввійшов в систему	25.05.2022 19:36
3	alex	Користувач вийшов із системи	25.05.2022 15:24
4	alex	Користувач ввійшов в систему	25.05.2022 15:23
5	alex	Користувач вийшов із системи	22.05.2022 14:02
6	alex	Користувач ввійшов в систему	22.05.2022 14:01
7	alex	Користувач вийшов із системи	11.05.2022 19:47
8	alex	Користувач ввійшов в систему	11.05.2022 19:47
9	alex	Користувач вийшов із системи	02.05.2022 11:26
10	alex	Користувач ввійшов в систему	02.05.2022 10:51
11	alex	Користувач вийшов із системи	01.05.2022 17:23
12	alex	Користувач ввійшов в систему	01.05.2022 17:23
13	alex	Користувач вийшов із системи	18.04.2022 19:02
14	alex	Користувач ввійшов в систему	18.04.2022 19:00
15	alex	Користувач вийшов із системи	13.04.2022 16:34
16	alex	Користувач ввійшов в систему	13.04.2022 15:48
17	alex	Користувач вийшов із системи	12.04.2022 17:59
18	alex	Користувач ввійшов в систему	12.04.2022 17:58
19	alex	Користувач вийшов із системи	12.04.2022 17:58

Рисунок 3.14 – Вікно «Системний журнал»

Треба сказати, що дана система є не сильно функціональною, але вона є досить простою в користуванні. Її інтерфейс є інтуїтивно зрозумілим для користувача.

Тестування програми успішно проведено та не було виявлено жодних помилок системи.

3.3 Інструкція користувачеві програми

3.3.1 Опис процедури розгортання програмного продукту, створеного на платформі .NET

Важливим фактором, який необхідно врахувати при розробці програмного забезпечення є потреба в ресурсах.

Мінімальні вимоги до апаратних засобів для запуску та функціонування розробленого програмного забезпечення:

– Процесор PentiumIV/Xeon2.4Ггц.

– Оперативна пам'ять: 1024 Мб вище.
– Вільний дисковий простір менше 120Мб.
– Миша.
– Клавіатура.
– Монітор.

Вимоги до програмних засобів:

– Операційна система сімейства Windows: починаючи з Windows XP – до Windows 10.
– Наявність бібліотеки класів .NET framework 4.0 або вище.

Розгортання програмного продукту на комп'ютері користувача у вигляді автономного застосунку:

1. Створіть на цільовому диску каталог для застосунку, наприклад «Тестування знань студентів»;
2. Скопіюйте каталог «Debug» із проекту.
3. Для перевірки коректності запуску програми виконайте подвійне клацання лівою кнопкою миші на файлі «TestSystem.exe» (операційна система Windows).

3.3.2 Використання програмного продукту

Запуск програми в операційній системі сімейства Windows здійснюється одним з стандартних способів:

а) подвійним клацанням лівою кнопкою миші на ярлику програми;

б) викликом контекстного меню з вибором його пункту "Відкрити";

в) натисканням кнопки "Пуск" панелі завдань із подальшим вибором пункту "Усі програми" та подвійним клацанням лівою кнопкою миші на яр-лику програми.

3.3.3 Робота з програмою

Дана програма відкриває широкі можливості для організації роботи туристичної агенції.

На початку необхідно запустити додаток «Туристичне агентство».

Для здійснення формування замовлень необхідно наповнити інформаційну систему, а саме: інформацію про клієнтів, менеджерів, країни, міста та послуги.

Програма дозволяє також редагувати та видаляти введені дані про клієнтів, менеджерів, країни, міста та послуги.

Для наповнення інформаційної бази клієнтів необхідно скористатись меню та відкрити вікно «Клієнти» (рис. 3.15).

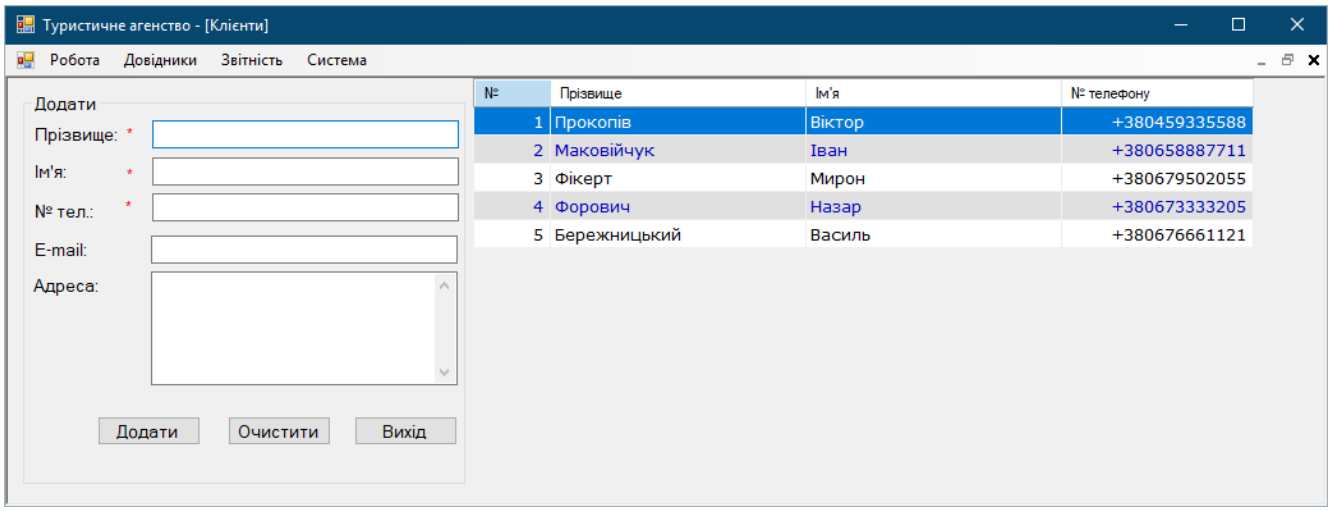


Рисунок 3.15 – Вікно для роботи з клієнтами

Щоб додати нового клієнта у лівій частині програми необхідно заповнити відповідні поля: прізвище, ім’я, № телефону, електронний адрес та адресу проживання, після чого натиснути кнопку «Додати». Інформація про клієнта буде додана у базу даних, а новий клієнт з’явиться у правій частині вікна. В правій частині відображається загальна інформація про клієнтів, а для того, щоби получити детальну інформацію необхідно натиснути лівою кнопкою мишки по необхідному запису, після чого з’явиться детальна інформація про клієнта, де можна буде її редагувати або видалити (рис. 3.16).

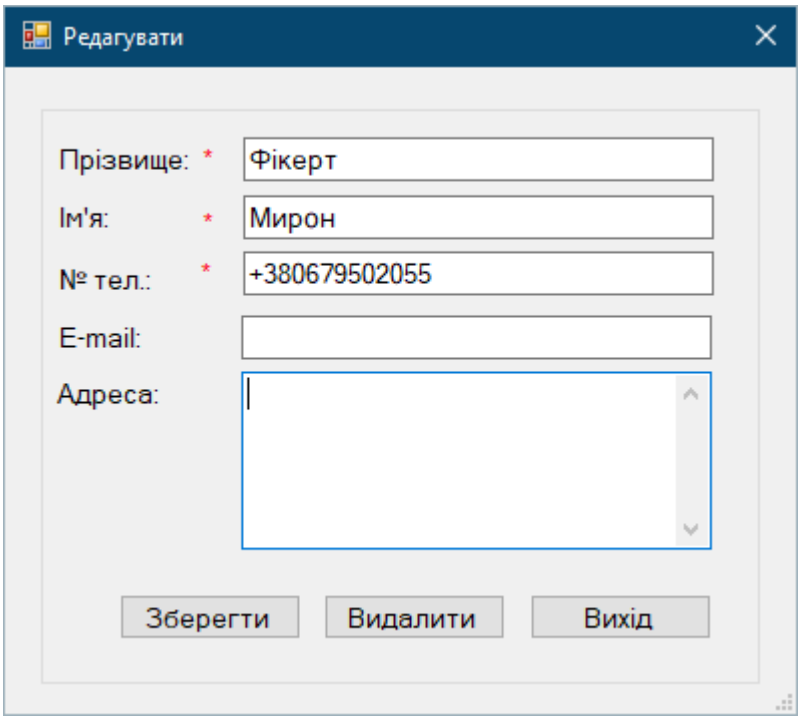


Рисунок 3.16 – Вікно для відображення детальної інформації про клієнта

Довідники з додаванням інформації про менеджерів, послуги, країни та міста додаються подібним чином. Екранні копії їх представлені у додатку А.

Щоб провести оформлення подорожі для клієнта, необхідно перейти по меню програми «Робота» - > «Подорожі», після чого виведеться дочірне вікно, де користувач програми зможе здійснити оформлення подорожі (рис. 3.17).

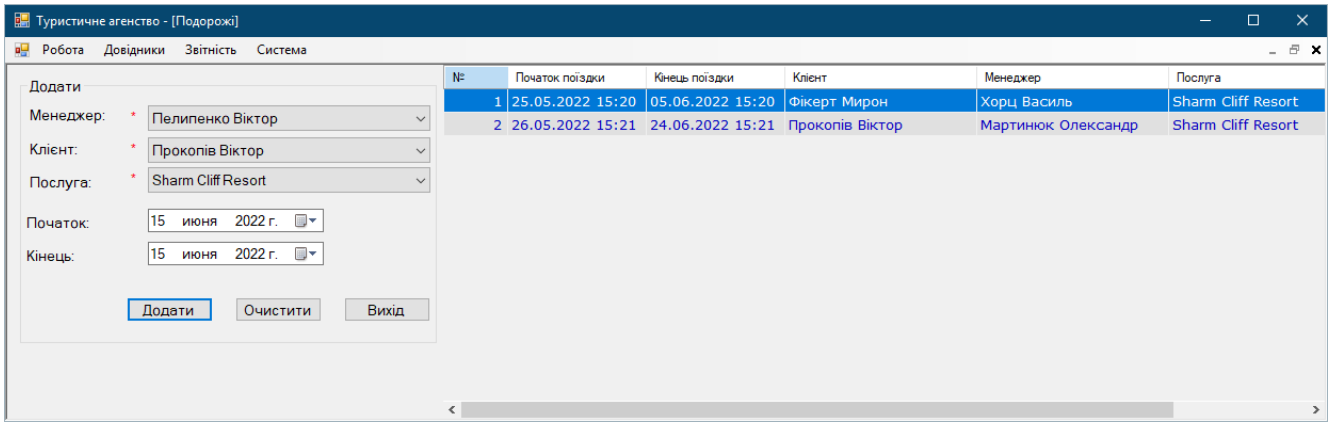


Рисунок 3.17 – Вікно для реєстрації подорожі

Так, як всі довідники заповнені можна додати інформацію про подорож. Для цього необхідно вибрати із випадаючого списку менеджера, клієнта, послугу тур.

агентства та вказати дату початку та кінця поїздки. Після цього програма додасть запис про поїздку у базу даних.

Однією із важливих функцій програми є можливість створення звітів, а саме (додаток А):

1. За вибраною датою виводиться наступна інформація по проведених подорожах:

- номер;
- Інформація про клієнта;
- Інформація про послуги;
- Інформація про менеджера, що оформляв подорож;
- Ціна, в яку обійшлася подорож.

2. За вибраною послугою виводиться інформація:

- номер;
- Інформація про клієнта;
- Дата відпустки;
- Інформація про менеджера, що оформляв подорож;
- Вартість відпустки.

Програма дозволяє також редагувати та видаляти будь-які дані системи. Всі

3.4 Висновок

В ході виконання роботи мовою C# в середовищі Visual Studio 2019 реалізовано систему для організації діяльності туристичного агентства.

Реалізовано такі функціональні вимоги:

- можливість додавати, редагувати та видаляти облікові записи користувачів;
- можливість додавати, редагувати та видаляти дані про користувачів;
- можливість додавати, редагувати та видаляти дані про менеджерів;
- можливість додавати, редагувати та видаляти дані про країни;

- можливість додавати, редагувати та видаляти дані про туристичні міста;
- можливість додавати, редагувати та видаляти дані про послуги туристичного агентства;
- можливість оформлення подорожі для клієнтів туристичного агентства;
- фіксування активності користувачів системи. Ведення логів.
- можливість формування звітності по клієнтах та послуги туристичного агентства.

ВИСНОВКИ

У ході виконання роботи було вивчено процес роботи туристичного агентства. Були розглянуті можливі проблеми та помилки при роботі туристичного агентства, причини виникнення та способи їхнього відстеження.

Розроблену програму можна використовувати для туристичного агентства, що значно полегшить їхню роботу.

У першій частині роботи було проведено поточного стану інформаційних систем в області туристичного агентства. Було зроблено опис технологій обробки інформації. Зроблено SWOT-аналіз існуючої інформаційної системи. На основі отриманих даних було проведено постановку задачі на створення інформаційної системи.

У другій частині роботи розглянуто дослідження системи та описано архітектуру системи туристичного агентства. Зроблено аналіз вимог до програмного забезпечення та побудовано use-case діаграми основних прецедентів.

У третій частині проведено тестування розробленого продукту та аналіз отриманих результатів. Також приведений загальний опис програми розробленої системи. Результати тестування були успішними, помилок не було виявлено. Закінчується розділ інструкцією користувачеві та висновками, що вдалося розробити в даній системі.

В результаті проведеної роботи вирішено актуальне науково-технічне завдання для організації роботи туристичного агентства. У процесі вирішення завдання розроблено інженерну методику автоматизованої процедури роботи туристичного агентства, і таким чином поставленої мети досягнуто. У ході досліджень отримано такі основні наукові та практичні результати:

1. Розроблено гнучку систему, призначену для організації роботи туристичного агентства.

2. Дана система поки що ніде не використовується. Розроблені методи в даній роботі можуть бути використані для широкого класу завдань, тому можливий подальший розвиток розробленого програмного забезпечення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Основні форми туризму: [Електронний ресурс]. – Режим доступу: https://pidru4niki.com/1263111343760/turizm/osnovni_formi_turizmu.
2. Схема розвитку туризму: [Електронний ресурс]. – Режим доступу: http://www.fa.ru/org/chair/mtgbism/Documents/Publications/Turizm_Basics.pdf.
3. SWOT-аналіз: [Електронний ресурс]. – Режим доступу: http://www.goodstudents.ru/swot/112-swot-analiz-primery.html.
4. Supported Platforms. [Електронний ресурс] – Режим доступу: http://www.asp.net/signalr/overview/getting-started/supported-platforms
5. Клієнт-серверна архітектура. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура
6. Особенности архитектуры клиент/сервер. [Електронний ресурс] – Режим доступу: http://citforum.ru/programming/application/builder_cs3.shtml
7. Компоненты сетевого приложения. Клиент-серверное взаимодействие и роли серверов. [Електронний ресурс] – Режим доступу: http://www.4stud.info/networking/lecture5.html
8. Introduction to SignalR. [Електронний ресурс] – Режим доступу: http://www.asp.net/signalr/overview/getting-started/introduction-to-signalr
9. Self-Hosting SignalR in a Windows Service. [Електронний ресурс] – Режим доступу: http://weblog.west-wind.com/posts/2013/Sep/04/SelfHosting-SignalR-in-a-Windows-Service#Self-HostingSignalR
10. Шпаргалка по NUnit: Основы unit тестирования [Електронний ресурс]. Режим доступу: http://iqhouse.blogspot.ru/2011/07/nunit-unit.html

ДОДАТКИ

ДОДАТОК А Електронні копії програми

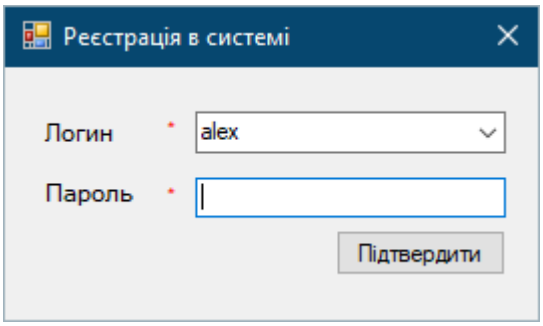


Рисунок 1– Вікно авторизації користувача

Рисунок 2– Головне вікно програми

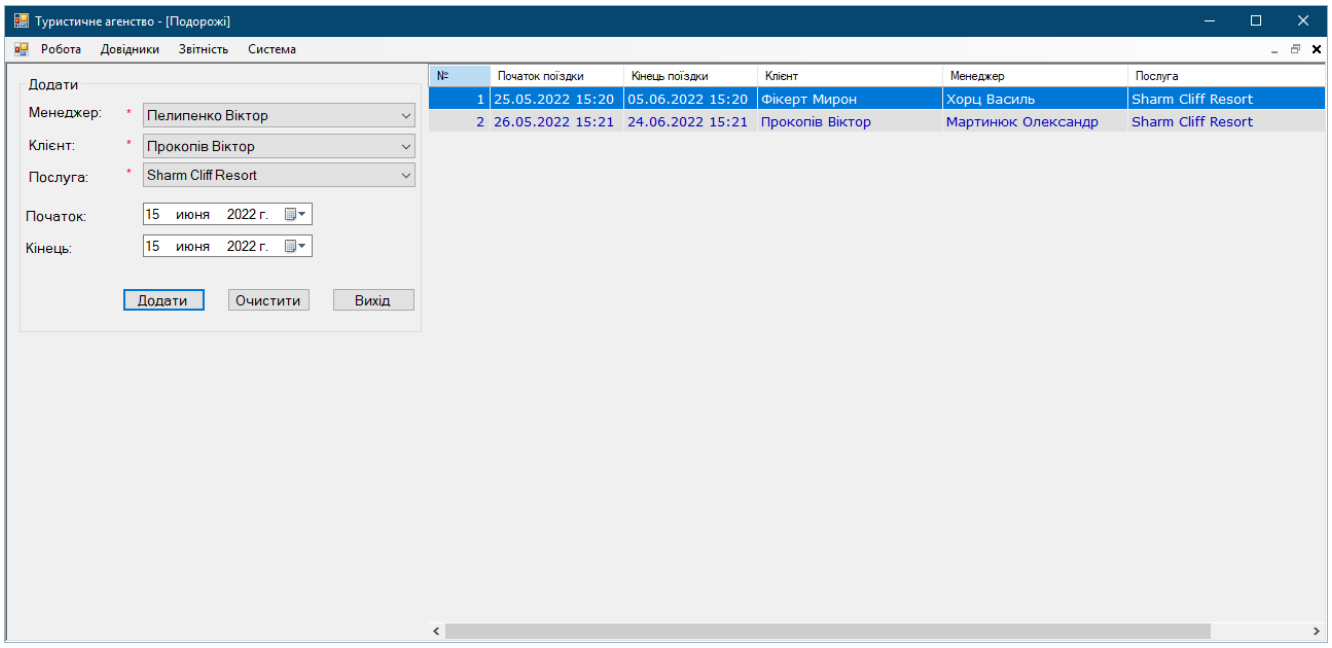


Рисунок 3– Вікно для реєстрації подорожі

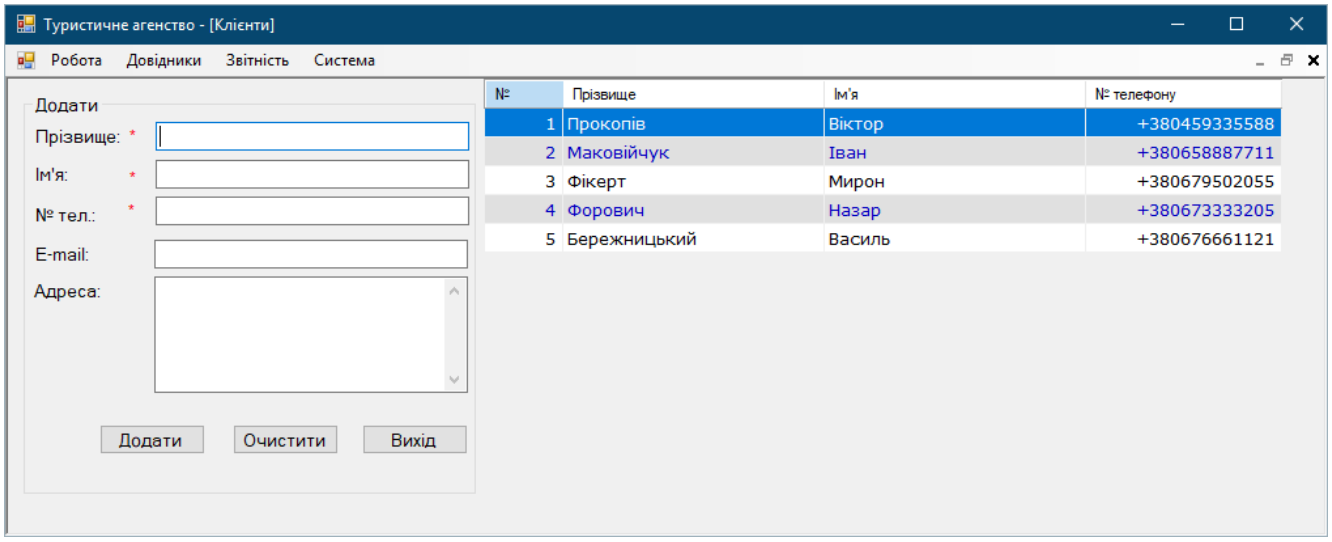


Рисунок 4– Вікно для опрацювання інформації по клієнтах

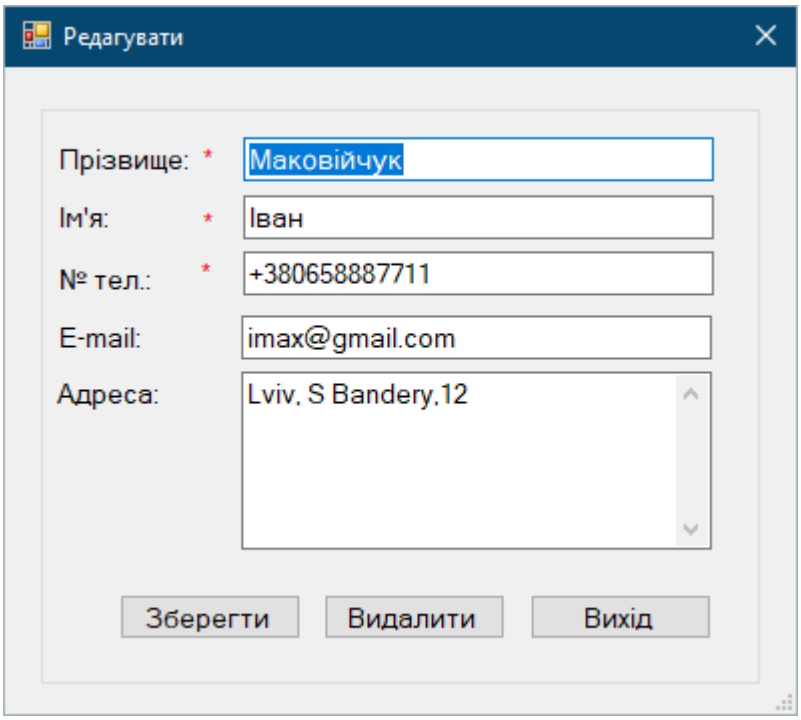


Рисунок 5– Вікно для редагування інформації по клієнтах

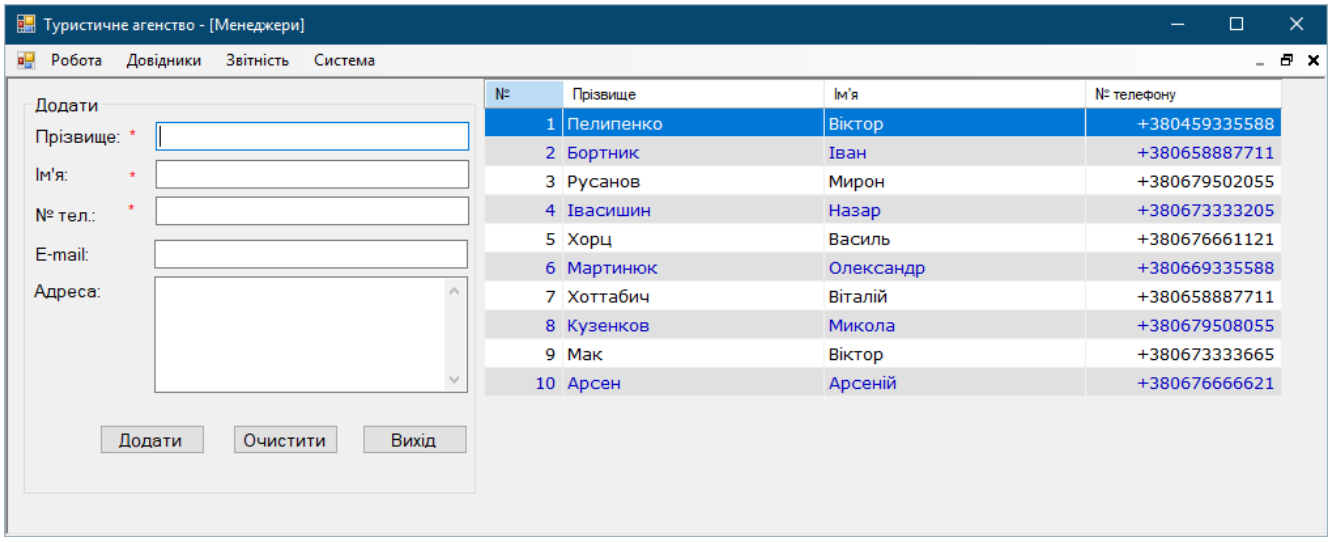


Рисунок 6– Вікно для опрацювання інформації по менеджерах

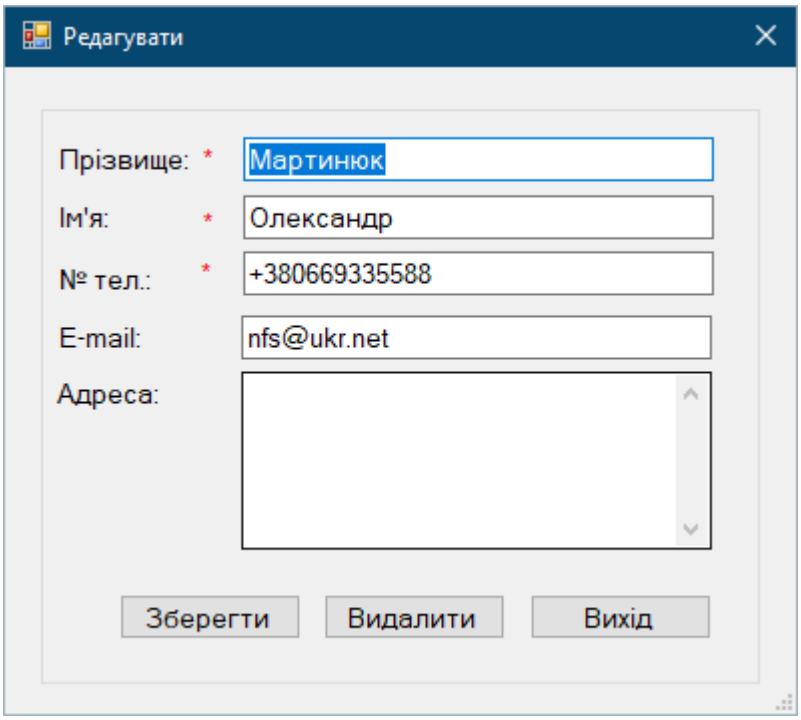


Рисунок 6– Вікно для опрацювання інформації по менеджерах

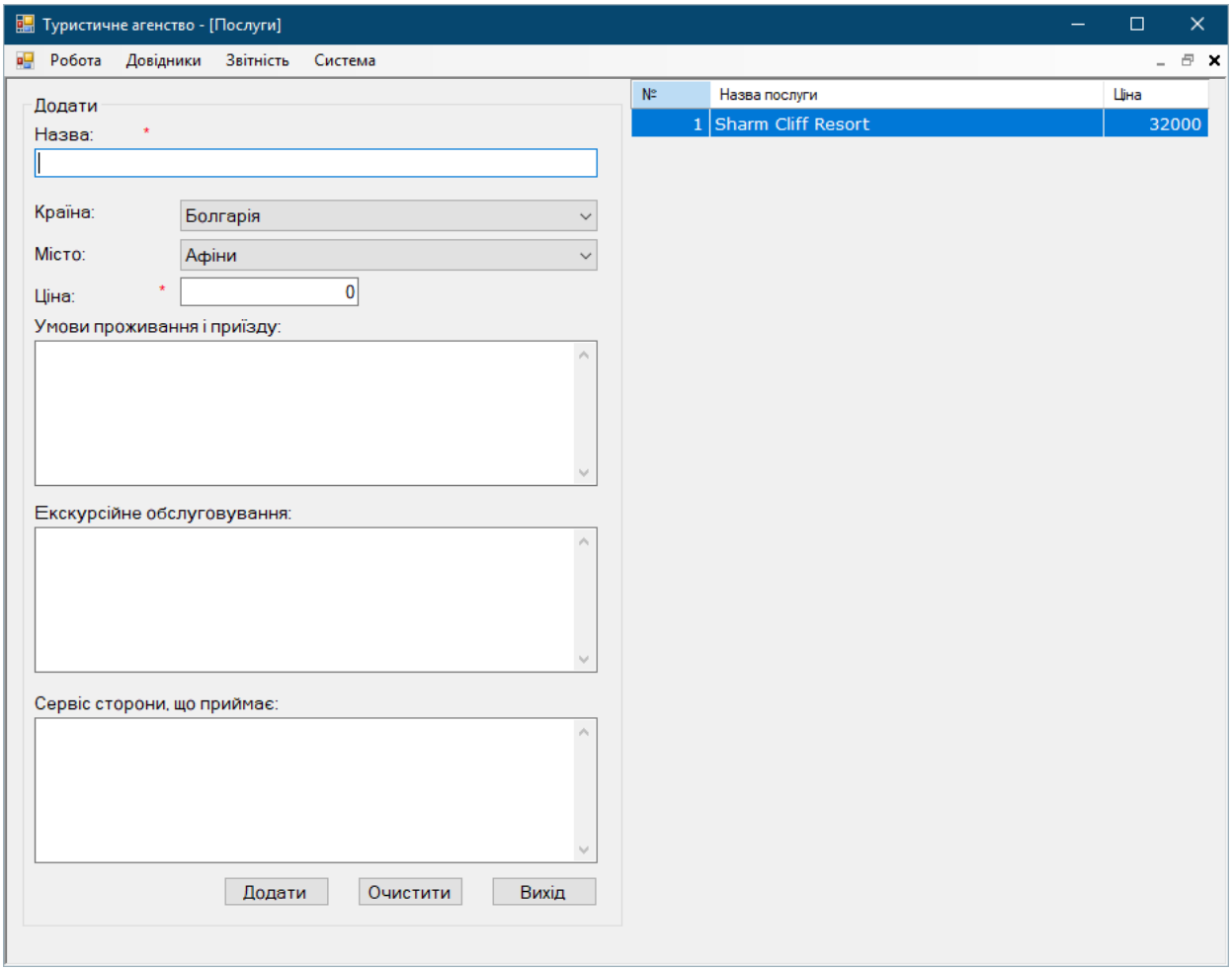


Рисунок 7– Вікно для опрацювання інформації по послугах

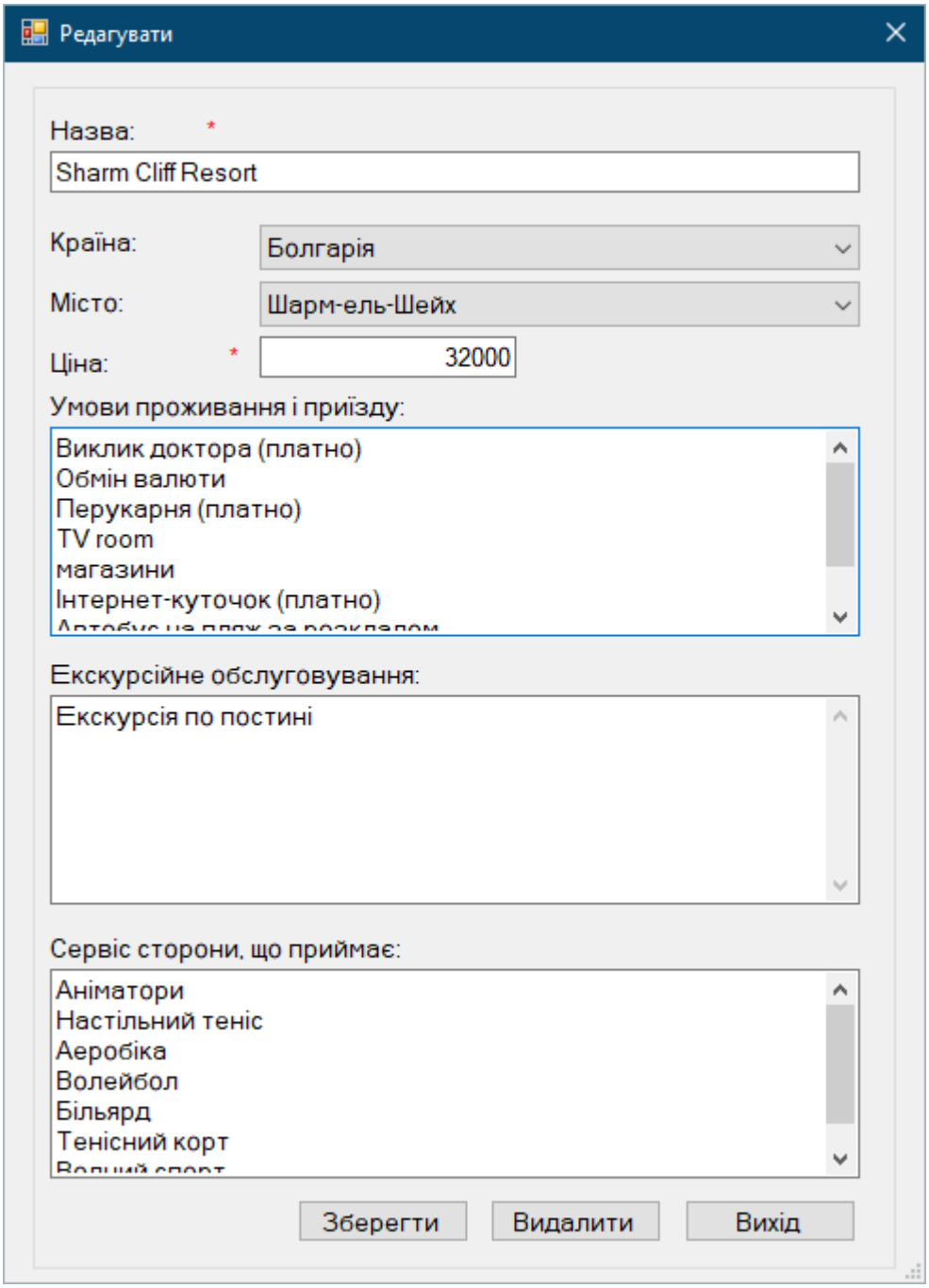


Рисунок 8– Вікно для редагування інформації по послугах

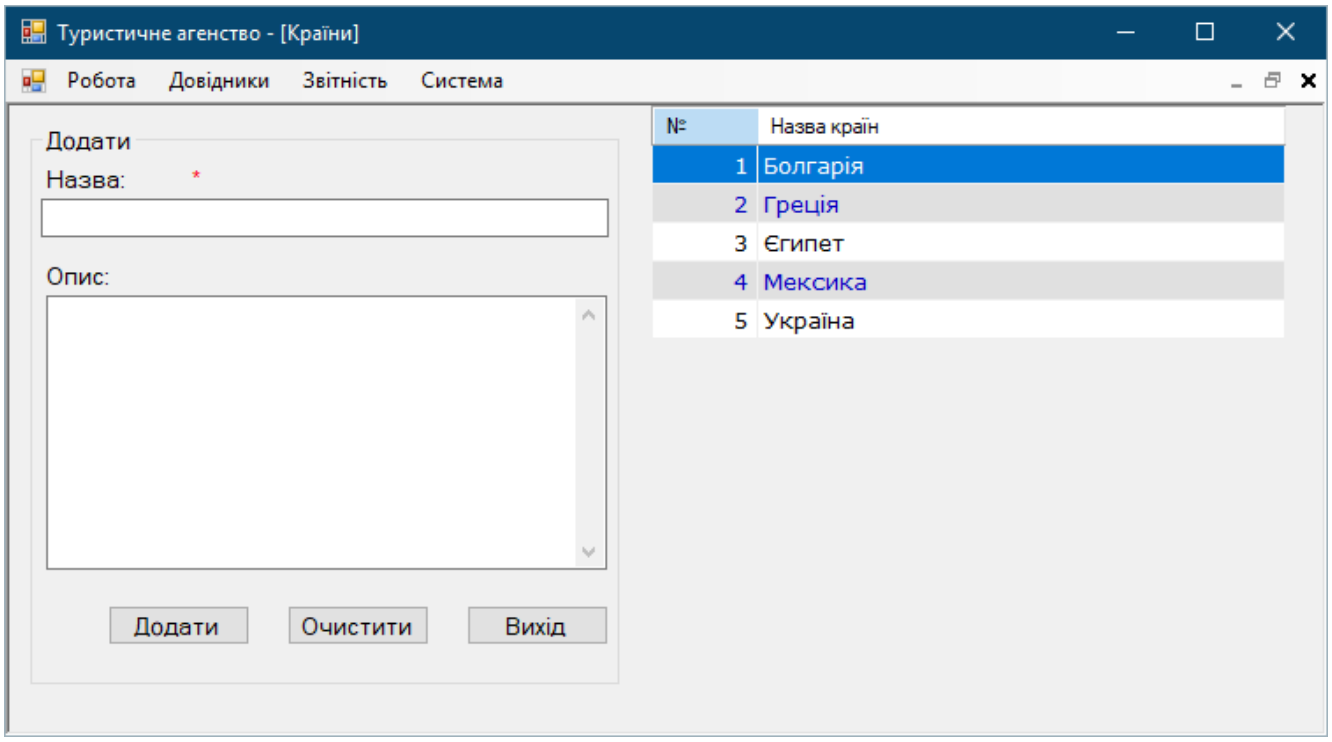


Рисунок 9– Вікно для опрацювання інформації по туристичних країнах

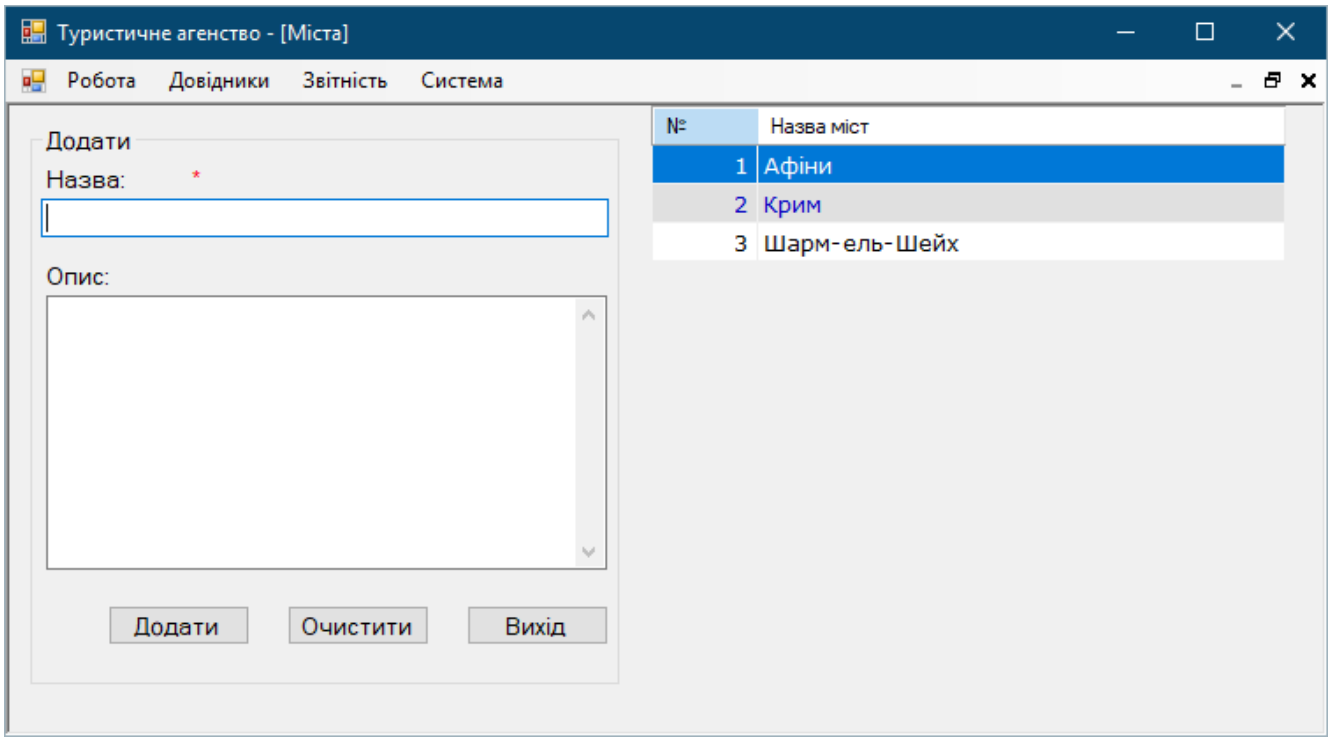


Рисунок 10– Вікно для опрацювання інформації по містах

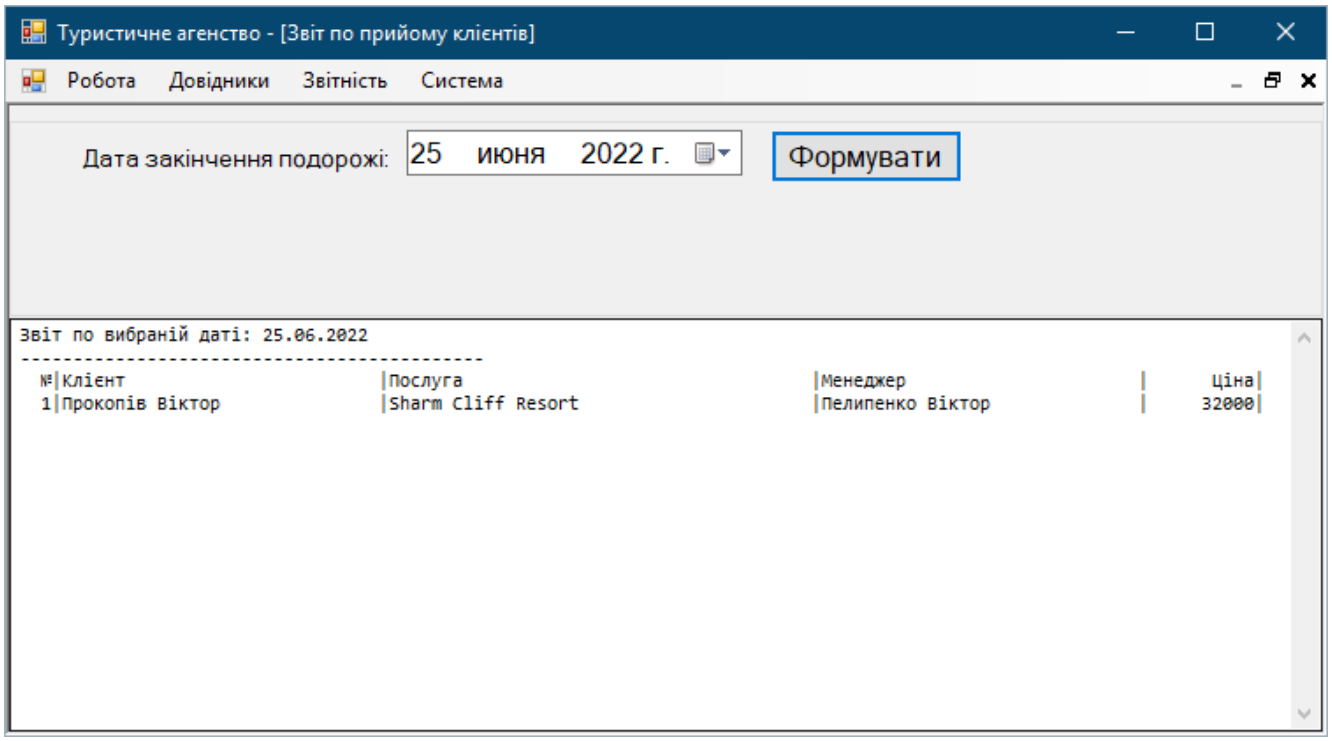


Рисунок 11– Вікно для формування звітності по клієнтах

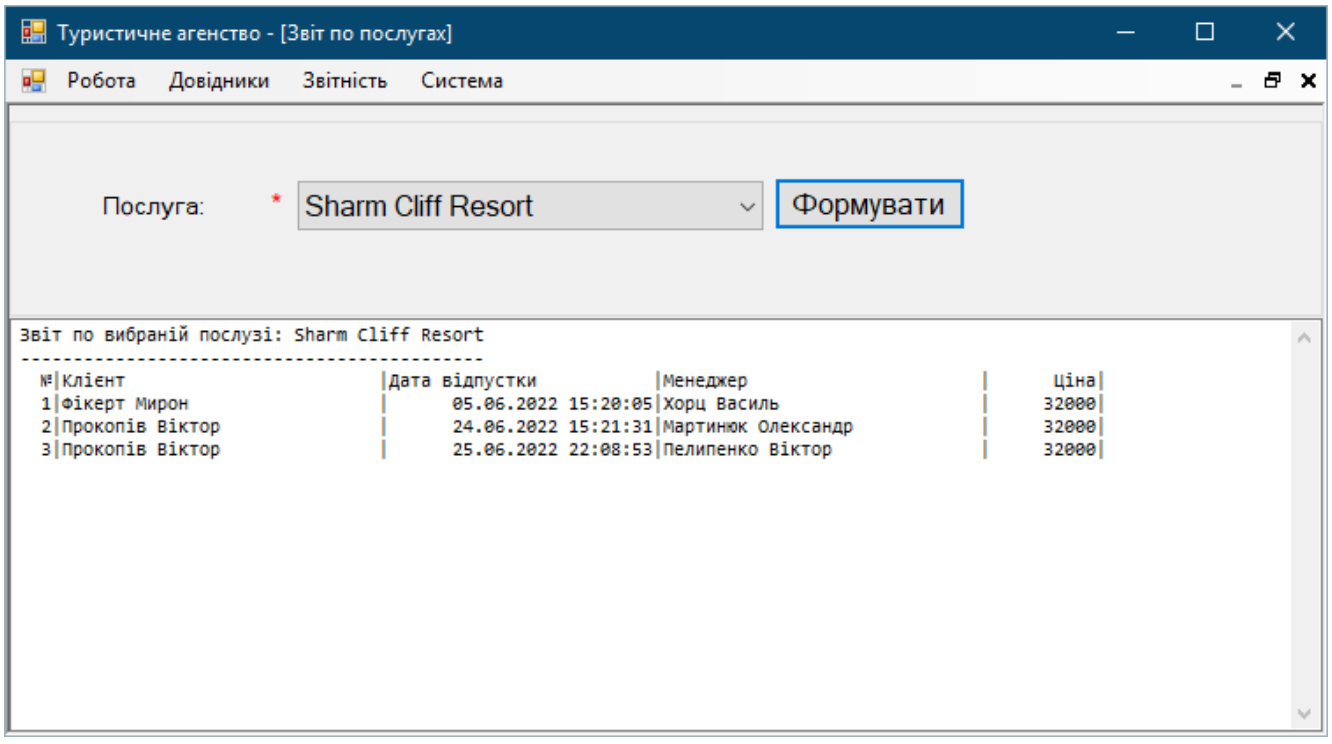


Рисунок 12– Вікно для формування звітності по послугах

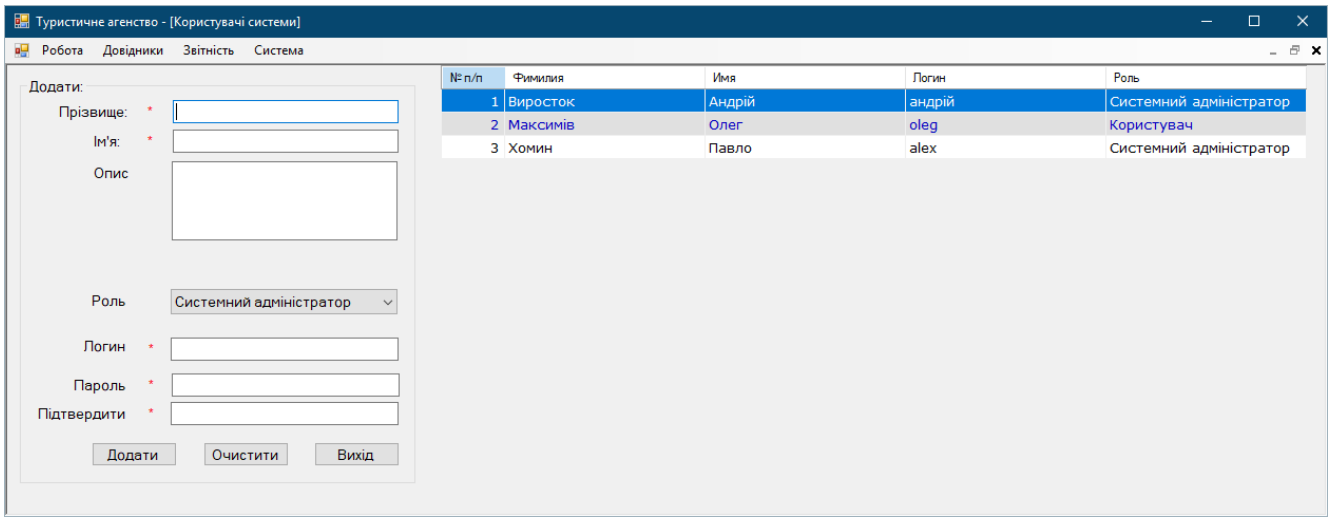


Рисунок 13– Вікно для опрацювання даних по користувачах системи

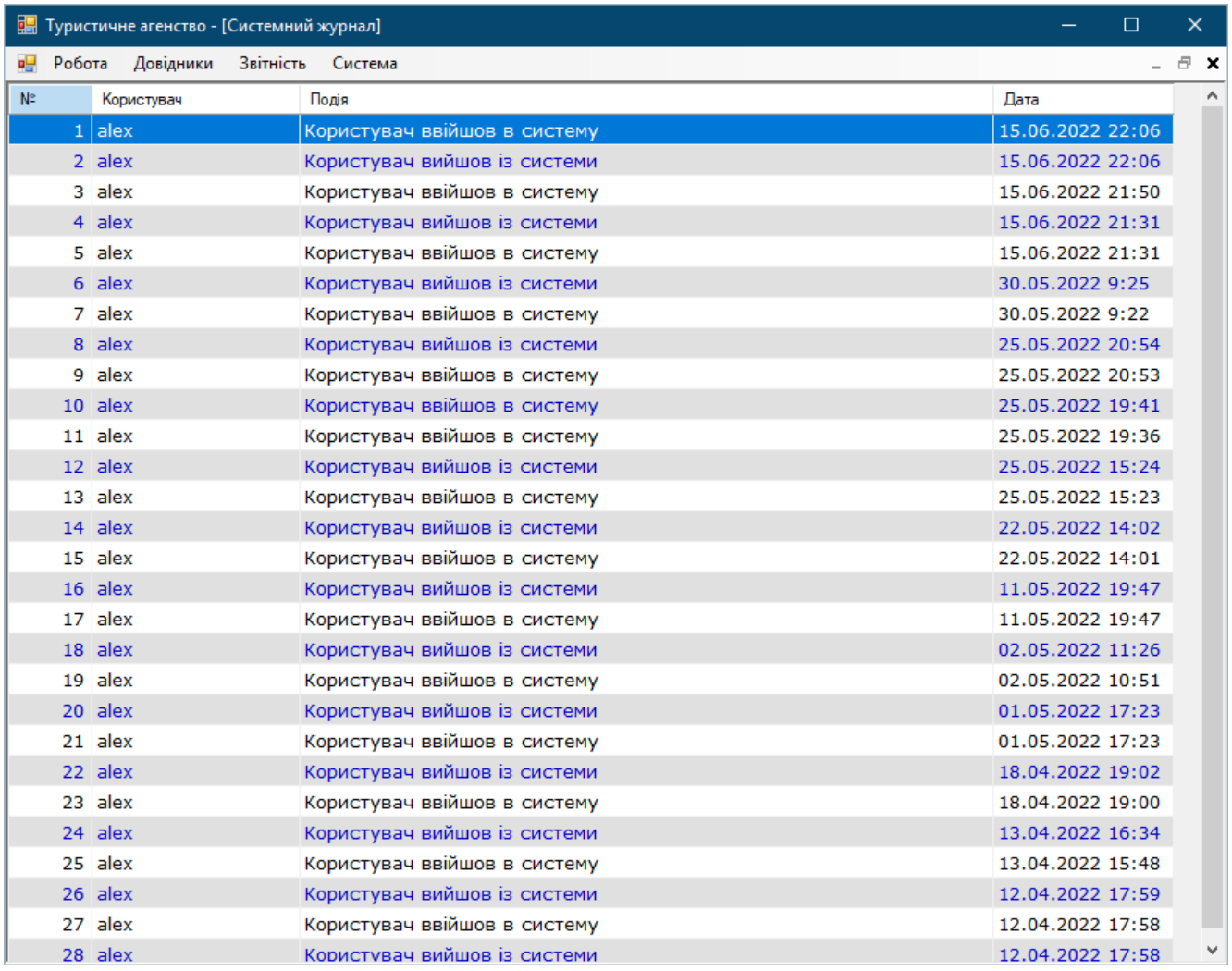


Рисунок 14– Вікно для перегляду активності користувачів програми

ДОДАТОК Б Лістинги програми

Лістинг класу «TripRaportBLL»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TravelAgenci.Providers;

namespace TravelAgenci.BLL {
  class TripRaportBLL {
    TripProvider _TripProvider = new TripProvider();
    ClientProvider _ClientProvider = new ClientProvider();
    ServiceProvider ServiceProvider = new ServiceProvider();
    ManagersProvider ManagersProvider = new ManagersProvider();

    public List<Trip> GetAllTrip() {
      List<Trip> tripList = new List<Trip>();
      List<Client> clientList = new List<Client>();
      List<Service> serviceList = new List<Service>();
      List<Managers> managersList = new List<Managers>();

      clientList = _ClientProvider.GetAllClient();
      managersList = ManagersProvider.GetAllManagers();
      serviceList = ServiceProvider.GetAllService();

      tripList = _TripProvider.GetAllTrip();
      for (int j = 0; j < tripList.Count; j++) {
        tripList[j].ClientFIO = GetClientFIO(tripList[j].ClientId, clientList);
        tripList[j].ManagersFIO = GetManagersFIO(tripList[j].ManagersId, managersList);
        tripList[j].ServiceName = GetSevicesName(tripList[j].ServiceId, serviceList);
        tripList[j].Price = GetSevicesPrice(tripList[j].ServiceId, serviceList);
      }

      return tripList;
    }

    public List<Trip> GetAllTripByDate(DateTime EndDTP) {
      List<Trip> allList = new List<Trip>();
      List<Trip> retAllList = new List<Trip>();
      allList = GetAllTrip();
      for (int i = 0; i < allList.Count; i++) {
        if (EndDTP.ToShortDateString() == allList[i].EndDate.ToShortDateString()) {
          retAllList.Add(allList[i]);
        }
      }
      for (int i = 0; i < retAllList.Count; i++) {
        retAllList[i].Number = i + 1;
      }
      return retAllList;
    }

    public List<Trip> GetAllTripByServiceId(int ServiceId) {
      List<Trip> allList = new List<Trip>();
      List<Trip> retAllList = new List<Trip>();
      allList = GetAllTrip();
      for (int i = 0; i < allList.Count; i++) {
        if (ServiceId == allList[i].ServiceId) {
          retAllList.Add(allList[i]);
        }
      }
      for (int i = 0; i < retAllList.Count; i++) {
        retAllList[i].Number = i + 1;
```

```
      }
      return retAllList;
    }

    private string GetSevicesName(int ServiceId, List<Service> ServiceList) {
      for (int i = 0; i < ServiceList.Count; i++) {
        if (ServiceId == ServiceList[i].ServiceId) {
          return ServiceList[i].ServiceName;
        }
      }
      return "";
    }

    private string GetClientFIO(int ClientId, List<Client> ClientList) {
      for (int i = 0; i < ClientList.Count; i++) {
        if (ClientId == ClientList[i].ClientId) {
          return ClientList[i].FIO;
        }
      }
      return "";
    }

    private string GetManagersFIO(int ManagersId, List<Managers> ManagersList) {
      for (int i = 0; i < ManagersList.Count; i++) {
        if (ManagersId == ManagersList[i].ManagersId) {
          return ManagersList[i].FIO;
        }
      }
      return "";
    }

    private double GetSevicesPrice(int ServiceId, List<Service> ServiceList) {
      for (int i = 0; i < ServiceList.Count; i++) {
        if (ServiceId == ServiceList[i].ServiceId) {
          return ServiceList[i].Price;
        }
      }
      return 0.0;
    }
  }
}
```

Лістинг класу « ClientForm »

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Dictionary {
  public partial class ClientForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    ClientProvider _ClientProvider = new ClientProvider();
    List<Client> _ClientList = new List<Client>();
```

```
    public ClientForm() {
      InitializeComponent();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _ClientProvider.InsertClient(LastNameTBox.Text, FirstNameTBox.Text, PhoneTBox.Text,
AddressTBox.Text, EmailTBox.Text);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

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

    private void DataLoad() {
      int firstRowIndex = 0;
      if (ClientGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = ClientGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _ClientList = _ClientProvider.GetAllClient();
        LoadDataInClientGridView(_ClientList);
        if (_selectedRowIndex == ClientGridView.Rows.Count) {
          _selectedRowIndex = ClientGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          ClientGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          ClientGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch { }
    }

    private void LoadDataInClientGridView(List<Client> ClientList) {
      ClientGridView.DataSource = null;
      ClientGridView.Columns.Clear();
      ClientGridView.AutoGenerateColumns = false;
      ClientGridView.RowHeadersVisible = false;

      ClientGridView.DataSource = ClientList;

      if (ClientList.Count > 0) {
        if (ClientList[0].Message == NamesMy.NoDataNames.NoDataInClient) {
          DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width = ClientGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
          ClientGridView.Columns.Add(messageColumn);
        } else {
          DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
          DetailIdColumn.DataPropertyName = "ClientId";
          ClientGridView.Columns.Add(DetailIdColumn);
          ClientGridView.Columns[0].Visible = false;

          DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
          numberColumn.HeaderText = "№ ";
          numberColumn.DataPropertyName = "Number";
```

```
            numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
            numberColumn.Width = NamesMy.SizeOptins.NumberSize;
            ClientGridView.Columns.Add(numberColumn);

            DataGridViewColumn LastNameColumn = new DataGridViewTextBoxColumn();
            LastNameColumn.HeaderText = "Прізвище";
            LastNameColumn.DataPropertyName = "LastName";
            LastNameColumn.Width = 200;
            ClientGridView.Columns.Add(LastNameColumn);

            DataGridViewColumn FirstNameColumn = new DataGridViewTextBoxColumn();
            FirstNameColumn.HeaderText = "Ім'я";
            FirstNameColumn.DataPropertyName = "FirstName";
            FirstNameColumn.Width = 200;
            ClientGridView.Columns.Add(FirstNameColumn);

            DataGridViewColumn PhoneColumn = new DataGridViewTextBoxColumn();
            PhoneColumn.HeaderText = "№ телефону";
            PhoneColumn.DataPropertyName = "Phone";
            PhoneColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
            PhoneColumn.Width = 150;
            ClientGridView.Columns.Add(PhoneColumn);

          }
          for (int i = 0; i < ClientGridView.Columns.Count; i++) {
            ClientGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
          }
        }
      }

      private void ClearAllControls() {
        LastNameTBox.Text = String.Empty;
        FirstNameTBox.Text = String.Empty;
        PhoneTBox.Text = String.Empty;
      }

      private bool IsDataEnteringCorrect() {
        bool isCorrect = true;
        if (_validation.IsDataEntering(LastNameTBox.Text)) {
          LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_validation.IsDataEntering(FirstNameTBox.Text)) {
          FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_validation.IsDataEntering(PhoneTBox.Text)) {
          PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        return isCorrect;
      }

      private void ClientGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
        if (e.RowIndex >= 0 && ClientGridView[0, e.RowIndex].Value.ToString() !=
_ClientList[0].Message) {
          _selectedRowIndex = e.RowIndex;
```

```
        UpdateClientForm updateClientForm = new
UpdateClientForm(Convert.ToInt32(ClientGridView[0, e.RowIndex].Value.ToString()));
        updateClientForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Лістинг класу «CountryForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Dictionary {
  public partial class CountryForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private CountryProvider _CountryProvider = new CountryProvider();
    private List<Country> _CountryList = new List<Country>();

    public CountryForm() {
      InitializeComponent();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _CountryProvider.InsertCountry(CountryNameTBox.Text, DescriptionTBox.Text);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

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

    private void DataLoad() {
      int firstRowIndex = 0;
      if (CountryGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = CountryGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _CountryList = _CountryProvider.GetAllCountry();
        LoadDataInCountryGridView(_CountryList);
        if (_selectedRowIndex == CountryGridView.Rows.Count) {
          _selectedRowIndex = CountryGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
```

```
      CountryGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
      CountryGridView.Rows[_selectedRowIndex].Selected = true;
    }
  } catch { }
}

private void LoadDataInCountryGridView(List<Country> CountryList) {
  CountryGridView.DataSource = null;
  CountryGridView.Columns.Clear();
  CountryGridView.AutoGenerateColumns = false;
  CountryGridView.RowHeadersVisible = false;

  CountryGridView.DataSource = CountryList;

  if (CountryList.Count > 0) {
    if (CountryList[0].Message == NamesMy.NoDataNames.NoDataInCountry) {
      DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
      messageColumn.DataPropertyName = "Message";
      messageColumn.Width = CountryGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
      CountryGridView.Columns.Add(messageColumn);
    } else {
      DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
      DetailIdColumn.DataPropertyName = "CountryId";
      CountryGridView.Columns.Add(DetailIdColumn);
      CountryGridView.Columns[0].Visible = false;

      DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
      numberColumn.HeaderText = "№ ";
      numberColumn.DataPropertyName = "Number";
      numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      numberColumn.Width = NamesMy.SizeOptins.NumberSize;
      CountryGridView.Columns.Add(numberColumn);

      DataGridViewColumn CountryNameColumn = new DataGridViewTextBoxColumn();
      CountryNameColumn.HeaderText = "Назва країн";
      CountryNameColumn.DataPropertyName = "CountryName";
      CountryNameColumn.Width = NamesMy.SizeOptins.NameSize;
      CountryGridView.Columns.Add(CountryNameColumn);


    }
    for (int i = 0; i < CountryGridView.Columns.Count; i++) {
      CountryGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
    }
  }
}

private void ClearAllControls() {
  CountryNameTBox.Text = String.Empty;
  DescriptionTBox.Text = String.Empty;
}

private bool IsDataEnteringCorrect() {
  bool isCorrect = true;
  if (_validation.IsDataEntering(CountryNameTBox.Text)) {
    CountryNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    CountryNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  return isCorrect;
}

private void CountryGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
```

```
      if (e.RowIndex >= 0 && CountryGridView[0, e.RowIndex].Value.ToString() !=
_CountryList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateCountryForm updateCountryForm = new
UpdateCountryForm(Convert.ToInt32(CountryGridView[0, e.RowIndex].Value.ToString()));
        updateCountryForm.ShowDialog();
        DataLoad();
      }
    }

  }
}
```

Лістинг класу «ManagersForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Dictionary {
  public partial class ManagersForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    ManagersProvider _ManagersProvider = new ManagersProvider();
    List<Managers> _ManagersList = new List<Managers>();
    public ManagersForm() {
      InitializeComponent();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _ManagersProvider.InsertManagers(LastNameTBox.Text, FirstNameTBox.Text,
PhoneTBox.Text, AddressTBox.Text, EmailTBox.Text);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();

    }

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

    private void DataLoad() {
      int firstRowIndex = 0;
      if (ManagersGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = ManagersGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _ManagersList = _ManagersProvider.GetAllManagers();
        LoadDataInManagersGridView(_ManagersList);
```

```
    if (_selectedRowIndex == ManagersGridView.Rows.Count) {
      _selectedRowIndex = ManagersGridView.Rows.Count - 1;
    }
    if (_selectedRowIndex >= 0) {
      ManagersGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
      ManagersGridView.Rows[_selectedRowIndex].Selected = true;
    }
  } catch { }
}

private void LoadDataInManagersGridView(List<Managers> ManagersList) {
  ManagersGridView.DataSource = null;
  ManagersGridView.Columns.Clear();
  ManagersGridView.AutoGenerateColumns = false;
  ManagersGridView.RowHeadersVisible = false;

  ManagersGridView.DataSource = ManagersList;

  if (ManagersList.Count > 0) {
    if (ManagersList[0].Message == NamesMy.NoDataNames.NoDataInManagers) {
      DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
      messageColumn.DataPropertyName = "Message";
      messageColumn.Width = ManagersGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
      ManagersGridView.Columns.Add(messageColumn);
    } else {
      DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
      DetailIdColumn.DataPropertyName = "ManagersId";
      ManagersGridView.Columns.Add(DetailIdColumn);
      ManagersGridView.Columns[0].Visible = false;

      DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
      numberColumn.HeaderText = "№ ";
      numberColumn.DataPropertyName = "Number";
      numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      numberColumn.Width = NamesMy.SizeOptins.NumberSize;
      ManagersGridView.Columns.Add(numberColumn);

      DataGridViewColumn LastNameColumn = new DataGridViewTextBoxColumn();
      LastNameColumn.HeaderText = "Прізвище";
      LastNameColumn.DataPropertyName = "LastName";
      LastNameColumn.Width = 200;
      ManagersGridView.Columns.Add(LastNameColumn);

      DataGridViewColumn FirstNameColumn = new DataGridViewTextBoxColumn();
      FirstNameColumn.HeaderText = "Ім'я";
      FirstNameColumn.DataPropertyName = "FirstName";
      FirstNameColumn.Width = 200;
      ManagersGridView.Columns.Add(FirstNameColumn);

      DataGridViewColumn PhoneColumn = new DataGridViewTextBoxColumn();
      PhoneColumn.HeaderText = "№ телефону";
      PhoneColumn.DataPropertyName = "Phone";
      PhoneColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      PhoneColumn.Width = 150;
      ManagersGridView.Columns.Add(PhoneColumn);

    }
    for (int i = 0; i < ManagersGridView.Columns.Count; i++) {
      ManagersGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
    }
  }
}

private void ClearAllControls() {
```

```
      LastNameTBox.Text = String.Empty;
      FirstNameTBox.Text = String.Empty;
      PhoneTBox.Text = String.Empty;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(LastNameTBox.Text)) {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(FirstNameTBox.Text)) {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PhoneTBox.Text)) {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void ManagersGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && ManagersGridView[0, e.RowIndex].Value.ToString() !=
_ManagersList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateManagersForm updateManagersForm = new
UpdateManagersForm(Convert.ToInt32(ManagersGridView[0, e.RowIndex].Value.ToString()));
        updateManagersForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Лістинг класу «ServiceForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Dictionary {
  public partial class ServiceForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private ServiceProvider _ServiceProvider = new ServiceProvider();
    private List<Service> _ServiceList = new List<Service>();
    private CountryProvider _CountryProvider = new CountryProvider();
    private List<Country> _CountryList = new List<Country>();
    private CityProvider _CityProvider = new CityProvider();
```

```
    private List<City> _CityList = new List<City>();

    public ServiceForm() {
      InitializeComponent();
      DataLoad();
      LoadAllDate();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _ServiceProvider.InsertService(Convert.ToInt32(CountryCBox.SelectedValue),
Convert.ToInt32(CityCBox.SelectedValue), LivingAndTravelConditionsTBox.Text,
          ExcursionServiceTBox.Text, HostServiceTBox.Text, Convert.ToDouble(PriceTBox.Text),
ServiceNameTBox.Text);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

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

    private void LoadAllDate() {
      _CountryList = _CountryProvider.GetAllCountry();
      CountryCBox.DataSource = _CountryList;
      CountryCBox.ValueMember = "CountryId";
      CountryCBox.DisplayMember = "CountryName";

      _CityList = _CityProvider.GetAllCity();
      CityCBox.DataSource = _CityList;
      CityCBox.ValueMember = "CityId";
      CityCBox.DisplayMember = "CityName";

    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (ServiceGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = ServiceGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _ServiceList = _ServiceProvider.GetAllService();
        LoadDataInServiceGridView(_ServiceList);
        if (_selectedRowIndex == ServiceGridView.Rows.Count) {
          _selectedRowIndex = ServiceGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          ServiceGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          ServiceGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch { }
    }

    private void LoadDataInServiceGridView(List<Service> ServiceList) {
      ServiceGridView.DataSource = null;
      ServiceGridView.Columns.Clear();
      ServiceGridView.AutoGenerateColumns = false;
      ServiceGridView.RowHeadersVisible = false;
```

```
  ServiceGridView.DataSource = ServiceList;

  if (ServiceList.Count > 0) {
    if (ServiceList[0].Message == NamesMy.NoDataNames.NoDataInService) {
      DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
      messageColumn.DataPropertyName = "Message";
      messageColumn.Width = ServiceGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
      ServiceGridView.Columns.Add(messageColumn);
    } else {
      DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
      DetailIdColumn.DataPropertyName = "ServiceId";
      ServiceGridView.Columns.Add(DetailIdColumn);
      ServiceGridView.Columns[0].Visible = false;

      DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
      numberColumn.HeaderText = "№ ";
      numberColumn.DataPropertyName = "Number";
      numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      numberColumn.Width = NamesMy.SizeOptins.NumberSize;
      ServiceGridView.Columns.Add(numberColumn);

      DataGridViewColumn ServiceNameColumn = new DataGridViewTextBoxColumn();
      ServiceNameColumn.HeaderText = "Назва послуги";
      ServiceNameColumn.DataPropertyName = "ServiceName";
      ServiceNameColumn.Width = NamesMy.SizeOptins.NameSize;
      ServiceGridView.Columns.Add(ServiceNameColumn);

      DataGridViewColumn PriceColumn = new DataGridViewTextBoxColumn();
      PriceColumn.HeaderText = "Ціна";
      PriceColumn.DataPropertyName = "Price";
      PriceColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      PriceColumn.Width = NamesMy.SizeOptins.Price;
      ServiceGridView.Columns.Add(PriceColumn);
    }
    for (int i = 0; i < ServiceGridView.Columns.Count; i++) {
      ServiceGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
    }
  }
}

private void ClearAllControls() {
  ServiceNameTBox.Text = String.Empty;
  LivingAndTravelConditionsTBox.Text = String.Empty;
  ExcursionServiceTBox.Text = String.Empty;
  HostServiceTBox.Text = String.Empty;
  PriceTBox.Text = "0";
}

private bool IsDataEnteringCorrect() {
  bool isCorrect = true;
  if (_validation.IsDataEntering(ServiceNameTBox.Text)) {
    ServiceNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    ServiceNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_validation.IsDataConvertToDouble(PriceTBox.Text)) {
    PriceValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    PriceValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  return isCorrect;
}
```

```
    private void ServiceGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && ServiceGridView[0, e.RowIndex].Value.ToString() !=
_ServiceList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateServiceForm updateServiceForm = new
UpdateServiceForm(Convert.ToInt32(ServiceGridView[0, e.RowIndex].Value.ToString()));
        updateServiceForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Лістинг класу «UpdateClientForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Dictionary {
  public partial class UpdateClientForm : Form {
    private int _ClientId = 0;
    private Client _selectedClient = new Client();
    private ClientProvider _ClientProvider = new ClientProvider();
    private ValidationMy _validation = new ValidationMy();

    public UpdateClientForm(int ClientId) {
      InitializeComponent();
      _ClientId = ClientId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _ClientProvider.UpdateClient(LastNameTBox.Text, FirstNameTBox.Text, PhoneTBox.Text,
AddressTBox.Text, EmailTBox.Text, _ClientId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _ClientProvider.DeleteClientByClientId(_ClientId);
        this.Close();
      }

    }

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

    private void LoadAllDate() {
      _selectedClient = _ClientProvider.SelectedClientByClientId(_ClientId);
```

```
      LastNameTBox.Text = _selectedClient.LastName;
      FirstNameTBox.Text = _selectedClient.FirstName;
      PhoneTBox.Text = _selectedClient.Phone;
      AddressTBox.Text = _selectedClient.Address;
      EmailTBox.Text = _selectedClient.Email;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(LastNameTBox.Text)) {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(FirstNameTBox.Text)) {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PhoneTBox.Text)) {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
  }
}
```

Лістинг класу «UpdateCountryForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Dictionary {
  public partial class UpdateCountryForm : Form {
    private int _CountryId = 0;
    private Country _selectedCountry = new Country();
    private CountryProvider _CountryProvider = new CountryProvider();
    private ValidationMy _Validation = new ValidationMy();

    public UpdateCountryForm(int CountryId) {
      InitializeComponent();
      _CountryId = CountryId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _CountryProvider.UpdateCountry(CountryNameTBox.Text, DescriptionTBox.Text,
_CountryId);
        this.Close();
```

```
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _CountryProvider.DeleteCountryByCountryId(_CountryId);
        this.Close();
      }
    }

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

    private void LoadAllDate() {
      _selectedCountry = _CountryProvider.SelectedCountryByCountryId(_CountryId);
      CountryNameTBox.Text = _selectedCountry.CountryName;
      DescriptionTBox.Text = _selectedCountry.Description;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_Validation.IsDataEntering(CountryNameTBox.Text)) {
        CountryNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        CountryNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

  }
}
```

Лістинг класу «UpdateManagersForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Dictionary {
  public partial class UpdateManagersForm : Form {
    private int _ManagersId = 0;
    private Managers _selectedManagers = new Managers();
    private ManagersProvider _ManagersProvider = new ManagersProvider();
    private ValidationMy _validation = new ValidationMy();

    public UpdateManagersForm(int ManagersId) {
      InitializeComponent();
      _ManagersId = ManagersId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
```

```
        _ManagersProvider.UpdateManagers(LastNameTBox.Text, FirstNameTBox.Text,
PhoneTBox.Text, AddressTBox.Text, EmailTBox.Text, _ManagersId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _ManagersProvider.DeleteManagersByManagersId(_ManagersId);
        this.Close();
      }
    }

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

    private void LoadAllDate() {
      _selectedManagers = _ManagersProvider.SelectedManagersByManagersId(_ManagersId);
      LastNameTBox.Text = _selectedManagers.LastName;
      FirstNameTBox.Text = _selectedManagers.FirstName;
      PhoneTBox.Text = _selectedManagers.Phone;
      AddressTBox.Text = _selectedManagers.Address;
      EmailTBox.Text = _selectedManagers.Email;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(LastNameTBox.Text)) {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(FirstNameTBox.Text)) {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PhoneTBox.Text)) {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PhoneValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
  }
}
```

Лістинг класу «UpdateServiceForm»

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

```
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Dictionary {
  public partial class UpdateServiceForm : Form {
    private int _ServiceId = 0;
    private ValidationMy _validation = new ValidationMy();
    private ServiceProvider _ServiceProvider = new ServiceProvider();
    private List<Service> _ServiceList = new List<Service>();
    private Service _selectedService = new Service();
    private CountryProvider _CountryProvider = new CountryProvider();
    private List<Country> _CountryList = new List<Country>();
    private CityProvider _CityProvider = new CityProvider();
    private List<City> _CityList = new List<City>();

    public UpdateServiceForm(int ServiceId) {
      InitializeComponent();
      _ServiceId = ServiceId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _ServiceProvider.UpdateService(Convert.ToInt32(CountryCBox.SelectedValue),
Convert.ToInt32(CityCBox.SelectedValue), LivingAndTravelConditionsTBox.Text,
          ExcursionServiceTBox.Text, HostServiceTBox.Text, Convert.ToDouble(PriceTBox.Text),
ServiceNameTBox.Text, _ServiceId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Вы действительно хотите удалить этот элемент?", "Удалить",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _ServiceProvider.DeleteServiceByServiceId(_ServiceId);
        this.Close();
      }
    }

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

    private void LoadAllDate() {
      _CountryList = _CountryProvider.GetAllCountry();
      CountryCBox.DataSource = _CountryList;
      CountryCBox.ValueMember = "CountryId";
      CountryCBox.DisplayMember = "CountryName";

      _CityList = _CityProvider.GetAllCity();
      CityCBox.DataSource = _CityList;
      CityCBox.ValueMember = "CityId";
      CityCBox.DisplayMember = "CityName";

      _selectedService = _ServiceProvider.SelectedServiceByServiceId(_ServiceId);
      ServiceNameTBox.Text = _selectedService.ServiceName;
      CountryCBox.SelectedValue = _selectedService.CountryId;
      CityCBox.SelectedValue = _selectedService.CityId;
      PriceTBox.Text = _selectedService.Price.ToString();
      LivingAndTravelConditionsTBox.Text = _selectedService.LivingAndTravelConditions;
      ExcursionServiceTBox.Text = _selectedService.ExcursionService;
      HostServiceTBox.Text = _selectedService.HostService;
    }

    private bool IsDataEnteringCorrect() {
```

```
      bool isCorrect = true;
      if (_validation.IsDataEntering(ServiceNameTBox.Text)) {
        ServiceNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ServiceNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataConvertToDouble(PriceTBox.Text)) {
        PriceValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PriceValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

  }
}
```

Лістинг класу «UpdateCityForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Dictionary {
  public partial class UpdateCityForm : Form {
    private int _CityId = 0;
    private City _selectedCity = new City();
    private CityProvider _CityProvider = new CityProvider();
    private ValidationMy _Validation = new ValidationMy();
    public UpdateCityForm(int CityId) {
      InitializeComponent();
      _CityId = CityId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _CityProvider.UpdateCity(CityNameTBox.Text, DescriptionTBox.Text, _CityId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _CityProvider.DeleteCityByCityId(_CityId);
        this.Close();
      }
    }

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

```
    private void LoadAllDate() {
      _selectedCity = _CityProvider.SelectedCityByCityId(_CityId);
      CityNameTBox.Text = _selectedCity.CityName;
      DescriptionTBox.Text = _selectedCity.Description;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_Validation.IsDataEntering(CityNameTBox.Text)) {
        CityNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        CityNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

  }
}
```

Лістинг класу «CityForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Dictionary {
  public partial class CityForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private CityProvider _CityProvider = new CityProvider();
    private List<City> _CityList = new List<City>();
    public CityForm() {
      InitializeComponent();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _CityProvider.InsertCity(CityNameTBox.Text, DescriptionTBox.Text);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

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

    private void DataLoad() {
      int firstRowIndex = 0;
      if (CityGridView.FirstDisplayedScrollingRowIndex > 0) {
```

```
    firstRowIndex = CityGridView.FirstDisplayedScrollingRowIndex;
  }
  try {
    _CityList = _CityProvider.GetAllCity();
    LoadDataInCityGridView(_CityList);
    if (_selectedRowIndex == CityGridView.Rows.Count) {
      _selectedRowIndex = CityGridView.Rows.Count - 1;
    }
    if (_selectedRowIndex >= 0) {
      CityGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
      CityGridView.Rows[_selectedRowIndex].Selected = true;
    }
  } catch { }
}

private void LoadDataInCityGridView(List<City> CityList) {
  CityGridView.DataSource = null;
  CityGridView.Columns.Clear();
  CityGridView.AutoGenerateColumns = false;
  CityGridView.RowHeadersVisible = false;

  CityGridView.DataSource = CityList;

  if (CityList.Count > 0) {
    if (CityList[0].Message == NamesMy.NoDataNames.NoDataInCity) {
      DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
      messageColumn.DataPropertyName = "Message";
      messageColumn.Width = CityGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
      CityGridView.Columns.Add(messageColumn);
    } else {
      DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
      DetailIdColumn.DataPropertyName = "CityId";
      CityGridView.Columns.Add(DetailIdColumn);
      CityGridView.Columns[0].Visible = false;

      DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
      numberColumn.HeaderText = "№ ";
      numberColumn.DataPropertyName = "Number";
      numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      numberColumn.Width = NamesMy.SizeOptins.NumberSize;
      CityGridView.Columns.Add(numberColumn);

      DataGridViewColumn CityNameColumn = new DataGridViewTextBoxColumn();
      CityNameColumn.HeaderText = "Назва міст";
      CityNameColumn.DataPropertyName = "CityName";
      CityNameColumn.Width = NamesMy.SizeOptins.NameSize;
      CityGridView.Columns.Add(CityNameColumn);


    }
    for (int i = 0; i < CityGridView.Columns.Count; i++) {
      CityGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
    }
  }
}

private void ClearAllControls() {
  CityNameTBox.Text = String.Empty;
  DescriptionTBox.Text = String.Empty;
}

private bool IsDataEnteringCorrect() {
  bool isCorrect = true;
  if (_validation.IsDataEntering(CityNameTBox.Text)) {
```

```
        CityNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        CityNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void CityGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && CityGridView[0, e.RowIndex].Value.ToString() !=
_CityList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateCityForm updateCityForm = new UpdateCityForm(Convert.ToInt32(CityGridView[0,
e.RowIndex].Value.ToString()));
        updateCityForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Лістинг класу «RaportClientForm»

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

namespace TravelAgenci.Forms.Raport {
  public partial class RaportClientForm : Form {
    TripRaportBLL _TripRaportBLL = new TripRaportBLL();
    List<Trip> _TripList = new List<Trip>();
    public RaportClientForm() {
      InitializeComponent();
    }

    private void SearchBtn_Click(object sender, EventArgs e) {
      _TripList = _TripRaportBLL.GetAllTripByDate(EndDTP.Value);
      GetOtchet(_TripList);
    }

    public void GetOtchet(List<Trip> TripList) {
      RaportTBox.Text = "Звіт по вибраній даті: " + EndDTP.Value.ToShortDateString() + "\r\n";
      RaportTBox.Text += "-----------------------------------------\r\n";

      RaportTBox.Text += String.Format("{0,3}|{1, -30}|{2, -40}|{3, -30}|{4, 10}|\r\n", "№",
"Клієнт", "Послуга", "Менеджер", "Ціна");
      for (int i = 0; i < TripList.Count(); i++) {
          string raportString = String.Format("{0,3}|{1, -30}|{2, -40}|{3, -30}|{4, 10}|\r\n",
          TripList[i].Number,
          TripList[i].ClientFIO,
          TripList[i].ServiceName,
          TripList[i].ManagersFIO,
          TripList[i].Price);
          RaportTBox.Text += raportString;
        }
    }
```

```
  }
}
```

Лістинг класу «RaportServiseForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.BLL;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Raport {
  public partial class RaportServiseForm : Form {
    List<Trip> _TripList = new List<Trip>();
    private ServiceProvider _ServiceProvider = new ServiceProvider();
    private List<Service> _ServiceList = new List<Service>();
    TripRaportBLL _TripRaportBLL = new TripRaportBLL();
    public RaportServiseForm() {
      InitializeComponent();
      LoadAllDate();

    }

    private void SearchBtn_Click(object sender, EventArgs e) {
      _TripList =
_TripRaportBLL.GetAllTripByServiceId(Convert.ToInt32(ServiceCBox.SelectedValue));
      GetRaport(_TripList);
    }

    private void LoadAllDate() {
      _ServiceList = _ServiceProvider.GetAllService();
      ServiceCBox.DataSource = _ServiceList;
      ServiceCBox.ValueMember = "ServiceId";
      ServiceCBox.DisplayMember = "ServiceName";
    }

    public void GetRaport(List<Trip> TripList) {
      RaportTBox.Text = "Звіт по вибраній послузі: " + ServiceCBox.Text + "\r\n";
      RaportTBox.Text += "-----------------------------------------\r\n";

      RaportTBox.Text += String.Format("{0,3}|{1, -30}|{2, -25}|{3, -30}|{4, 10}|\r\n", "№",
"Клієнт", "Дата відпустки", "Менеджер", "Ціна");
      for (int i = 0; i < TripList.Count(); i++) {
        if (Convert.ToInt32(ServiceCBox.SelectedValue) == TripList[i].ServiceId) {
          string raportString = String.Format("{0,3}|{1, -30}|{2, 25}|{3, -30}|{4, 10}|\r\n",
          TripList[i].Number,
          TripList[i].ClientFIO,
          TripList[i].EndDate,
          TripList[i].ManagersFIO,
          TripList[i].Price);
          RaportTBox.Text += raportString;
        }
      }
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
```

```
      if (Convert.ToInt32(ServiceCBox.SelectedValue) > 0) {
        ServiceValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ServiceValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }

      return isCorrect;
    }
  }
}
```

Лістинг класу «LoginForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Systems {
  public partial class LoginForm : Form {
    public static Users CurrentUser = new Users();

    private UsersProvider _UserProvider = new UsersProvider();
    private ValidationMy _validation = new ValidationMy();
    private LogsProvider _LogsProvider = new LogsProvider();
    private List<Users> _UserList = new List<Users>();
    public LoginForm() {
      InitializeComponent();
      LoadAllDate();
    }

    private void SubmitBtn_Click(object sender, EventArgs e) {
      GetSubmitData();
    }

    private void DataLoad() {
      _LogsProvider.InsertLogs(CurrentUser.UsersId, "Користувач ввійшов в систему",
DateTime.Now);
      this.Visible = false;
      (new TravelAgenciMDI()).ShowDialog();
      _LogsProvider.InsertLogs(CurrentUser.UsersId, "Користувач вийшов із системи",
DateTime.Now);
      this.Close();
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(UserNameCBox.Text)) {
        UserNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UserNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(UserPasswordTbx.Text)) {
        UserPasswordValidation.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
```

```
        UserPasswordValidation.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void LoadAllDate() {
      _UserList = _UserProvider.GetAllUsersListForCBox();
      UserNameCBox.DataSource = _UserList;
      UserNameCBox.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
      UserNameCBox.AutoCompleteSource = AutoCompleteSource.ListItems;
      UserNameCBox.ValueMember = "UsersId";
      UserNameCBox.DisplayMember = "UsersName";
    }

    private void GetSubmitData() {
      try {
        if (IsDataEnteringCorrect()) {
          List<Users> listUsers = new List<Users>();
          listUsers =
_UserProvider.SelectedUsersByUsersNameAndUsersPassword(UserNameCBox.Text,
UserPasswordTbx.Text);
          if (listUsers.Count > 0) {
            CurrentUser = listUsers[0];
            DataLoad();
          } else {

MessageBox.Show(NamesMy.MessageBoxExaption.ThisUserLoginAndUserPasswordNotExistInSystem,
NamesMy.MessageBoxExaption.CaptionMessage);
          }
        }
      } catch {
        MessageBox.Show("Немає з'єднання!");
      }
    }
  }
}
```

Лістинг класу «SystemLogForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Systems {
  public partial class SystemLogForm : Form {
    private int _selectedRowIndex = 0;
    private LogsProvider _LogsProvider = new LogsProvider();
    private List<Logs> _LogsList = new List<Logs>();
    public SystemLogForm() {
      InitializeComponent();
      DataLoad();
    }

    private void DataLoad() {
      int firstRowIndex = 0;
```

```
  if (LogsGridView.FirstDisplayedScrollingRowIndex > 0) {
    firstRowIndex = LogsGridView.FirstDisplayedScrollingRowIndex;
  }
  try {
    _LogsList = _LogsProvider.GetAllLogs();
    LoadDataInLogsGridView(_LogsList);
    if (_selectedRowIndex == LogsGridView.Rows.Count) {
      _selectedRowIndex = LogsGridView.Rows.Count - 1;
    }
    if (_selectedRowIndex >= 0) {
      LogsGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
      LogsGridView.Rows[_selectedRowIndex].Selected = true;
    }
  } catch { }
}

private void LoadDataInLogsGridView(List<Logs> LogsList) {
  LogsGridView.DataSource = null;
  LogsGridView.Columns.Clear();
  LogsGridView.AutoGenerateColumns = false;
  LogsGridView.RowHeadersVisible = false;

  LogsGridView.DataSource = LogsList;

  if (LogsList.Count > 0) {
    if (LogsList[0].Message == NamesMy.NoDataNames.NoDataInLogs) {
      DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
      messageColumn.DataPropertyName = "Message";
      messageColumn.Width = LogsGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
      LogsGridView.Columns.Add(messageColumn);
    } else {
      DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
      DetailIdColumn.DataPropertyName = "LogsId";
      LogsGridView.Columns.Add(DetailIdColumn);
      LogsGridView.Columns[0].Visible = false;

      DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
      numberColumn.HeaderText = "№ ";
      numberColumn.DataPropertyName = "Number";
      numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      numberColumn.Width = NamesMy.SizeOptins.NumberSize;
      LogsGridView.Columns.Add(numberColumn);

      DataGridViewColumn UsersNameColumn = new DataGridViewTextBoxColumn();
      UsersNameColumn.HeaderText = "Користувач";
      UsersNameColumn.DataPropertyName = "UsersName";
      UsersNameColumn.Width = 150;
      LogsGridView.Columns.Add(UsersNameColumn);

      DataGridViewColumn EventNameShowColumn = new DataGridViewTextBoxColumn();
      EventNameShowColumn.HeaderText = "Подія";
      EventNameShowColumn.DataPropertyName = "EventNameShow";
      EventNameShowColumn.Width = 500;
      LogsGridView.Columns.Add(EventNameShowColumn);

      DataGridViewColumn EvendDateColumn = new DataGridViewTextBoxColumn();
      EvendDateColumn.HeaderText = "Дата";
      EvendDateColumn.DataPropertyName = "EvendDate";
      EvendDateColumn.Width = NamesMy.SizeOptins.Date;
      LogsGridView.Columns.Add(EvendDateColumn);

    }
    for (int i = 0; i < LogsGridView.Columns.Count; i++) {
      LogsGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
```

```
        }
      }
    }

  }
}
```

Лістинг класу «UpdateUsersForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Systems {
  public partial class UpdateUsersForm : Form {
    private int _UserId = 0;
    private Users _selectedUser = new Users();
    private UsersProvider _UserProvider = new UsersProvider();
    private ValidationMy _validation = new ValidationMy();
    private RoleApp _RoleApp = new RoleApp();
    private List<Role> _RoleList = new List<Role>();

    public UpdateUsersForm(int UserId) {
      InitializeComponent();
      _UserId = UserId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _UserProvider.UpdateUsers(FirstNameTBox.Text, LastNameTBox.Text, UserLoginTbx.Text,
PasswordTbx.Text,
          Convert.ToInt32(RolesCBox.SelectedValue), DescriptionTbx.Text, _UserId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _UserProvider.DeleteUsersByUsersId(_UserId);
        this.Close();
      }
    }

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

    private void LoadAllDate() {
      _RoleList = _RoleApp.GetRoleList();
      RolesCBox.DataSource = _RoleList;
      RolesCBox.ValueMember = "RoleId";
      RolesCBox.DisplayMember = "RoleName";

      _selectedUser = _UserProvider.SelectedUsersByUsersId(_UserId);
```

```
      FirstNameTBox.Text = _selectedUser.FirstName;
      LastNameTBox.Text = _selectedUser.LastName;
      UserLoginTbx.Text = _selectedUser.UsersName;
      RolesCBox.SelectedValue = _selectedUser.RoleId;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(FirstNameTBox.Text)) {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(LastNameTBox.Text)) {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsPasswordMatch(PasswordTbx.Text, RePasswordTbx.Text)) {
        PasswordAndRePasswordDontMatchLbl.Visible = false;
      } else {
        PasswordAndRePasswordDontMatchLbl.Visible = true;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(UserLoginTbx.Text)) {
        UserLoginValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UserLoginValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PasswordTbx.Text)) {
        PasswordValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PasswordValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(RePasswordTbx.Text)) {
        RePasswordValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        RePasswordValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
  }
}
```

Лістинг класу «UsersForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Systems {
```

```
  public partial class UsersForm : Form {
    private int _selectedRowIndex = 0;
    ValidationMy _validation = new ValidationMy();
    private UsersProvider _UserProvider = new UsersProvider();
    private List<Users> _UserList = new List<Users>();
    private RoleApp _RoleApp = new RoleApp();
    private List<Role> _RoleList = new List<Role>();
    public UsersForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _UserProvider.InsertUsers(FirstNameTBox.Text, LastNameTBox.Text, UserLoginTbx.Text,
PasswordTbx.Text,
          Convert.ToInt32(RolesCBox.SelectedValue), DescriptionTbx.Text);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

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

    private void DataLoad() {
      int firstRowIndex = 0;
      if (UsersGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = UsersGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _UserList = _UserProvider.GetAllUsers();
        LoadDataInKlientGridView(_UserList);
        if (_selectedRowIndex == UsersGridView.Rows.Count) {
          _selectedRowIndex = UsersGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          UsersGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          UsersGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch { }
    }

    private void LoadDataInKlientGridView(List<Users> UserList) {
      UsersGridView.DataSource = null;
      UsersGridView.Columns.Clear();
      UsersGridView.AutoGenerateColumns = false;
      UsersGridView.RowHeadersVisible = false;

      UsersGridView.DataSource = UserList;

      if (UserList.Count > 0) {
        if (UserList[0].Message == NamesMy.NoDataNames.NoDataInUsers) {
          DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width = UsersGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
          UsersGridView.Columns.Add(messageColumn);
        } else {
```

```
      DataGridViewColumn deviseIdColumn = new DataGridViewTextBoxColumn();
      deviseIdColumn.DataPropertyName = "UsersId";
      UsersGridView.Columns.Add(deviseIdColumn);
      UsersGridView.Columns[0].Visible = false;

      DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
      numberColumn.HeaderText = "№ п/п";
      numberColumn.DataPropertyName = "Number";
      numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      numberColumn.Width = NamesMy.SizeOptins.NumberSize;
      UsersGridView.Columns.Add(numberColumn);

      DataGridViewColumn lastNameColumn = new DataGridViewTextBoxColumn();
      lastNameColumn.HeaderText = "Фимилия";
      lastNameColumn.DataPropertyName = "LastName";
      lastNameColumn.Width = NamesMy.SizeOptins.Name;
      UsersGridView.Columns.Add(lastNameColumn);

      DataGridViewColumn firstNameColumn = new DataGridViewTextBoxColumn();
      firstNameColumn.HeaderText = "Имя";
      firstNameColumn.DataPropertyName = "FirstName";
      firstNameColumn.Width = NamesMy.SizeOptins.Name;
      UsersGridView.Columns.Add(firstNameColumn);

      DataGridViewColumn UserNameColumn = new DataGridViewTextBoxColumn();
      UserNameColumn.HeaderText = "Логин";
      UserNameColumn.DataPropertyName = "UsersName";
      UserNameColumn.Width = NamesMy.SizeOptins.Name;
      UsersGridView.Columns.Add(UserNameColumn);

      DataGridViewColumn roleNameColumn = new DataGridViewTextBoxColumn();
      roleNameColumn.HeaderText = "Роль";
      roleNameColumn.DataPropertyName = "RoleName";
      roleNameColumn.Width = NamesMy.SizeOptins.Name;
      UsersGridView.Columns.Add(roleNameColumn);
    }
    for (int i = 0; i < UsersGridView.Columns.Count; i++) {
      UsersGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
    }
  }
}

private void ClearAllControls() {
  FirstNameTBox.Text = String.Empty;
  LastNameTBox.Text = String.Empty;
  DescriptionTbx.Text = String.Empty;
  UserLoginTbx.Text = String.Empty;
  PasswordTbx.Text = String.Empty;
  RePasswordTbx.Text = String.Empty;
}

private bool IsDataEnteringCorrect() {
  bool isCorrect = true;
  if (_validation.IsDataEntering(FirstNameTBox.Text)) {
    FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    FirstNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  if (_validation.IsDataEntering(LastNameTBox.Text)) {
    LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    LastNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
```

```
      }
      if (_validation.IsPasswordMatch(PasswordTbx.Text, RePasswordTbx.Text)) {
        PasswordAndRePasswordDontMatchLbl.Visible = false;
        PasswordAndRePasswordDontMatchLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PasswordAndRePasswordDontMatchLbl.Visible = true;
        PasswordAndRePasswordDontMatchLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(UserLoginTbx.Text)) {
        UserLoginValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        UserLoginValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(PasswordTbx.Text)) {
        PasswordValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PasswordValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataEntering(RePasswordTbx.Text)) {
        RePasswordValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        RePasswordValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void LoadAllDate() {
      _RoleList = _RoleApp.GetRoleList();
      RolesCBox.DataSource = _RoleList;
      RolesCBox.ValueMember = "RoleId";
      RolesCBox.DisplayMember = "RoleName";
    }

    private void UsersGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && UsersGridView[0, e.RowIndex].Value.ToString() !=
_UserList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateUsersForm updateUsersForm = new UpdateUsersForm(Convert.ToInt32(UsersGridView[0,
e.RowIndex].Value.ToString()));
        updateUsersForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Лістинг класу «TripForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.BLL;
using TravelAgenci.Providers;
```

```
namespace TravelAgenci.Forms.Trips {
  public partial class TripForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _Validation = new ValidationMy();
    private TripProvider _TripProvider = new TripProvider();
    private TripRaportBLL _TripRaportBLL = new TripRaportBLL();
    private List<Trip> _TripList = new List<Trip>();
    private ClientProvider _ClientProvider = new ClientProvider();
    private List<Client> _ClientList = new List<Client>();
    private ServiceProvider _ServiceProvider = new ServiceProvider();
    private List<Service> _ServiceList = new List<Service>();
    private ManagersProvider _ManagersProvider = new ManagersProvider();
    private List<Managers> _ManagersList = new List<Managers>();
    public TripForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _TripProvider.InsertTrip(Convert.ToInt32(ManagersCBox.SelectedValue),
Convert.ToInt32(ClientCBox.SelectedValue),
          Convert.ToInt32(ServiceCBox.SelectedValue), StartDateDTP.Value, EndDTP.Value);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

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

    private void LoadAllDate() {
      _ClientList = _ClientProvider.GetAllClient();
      ClientCBox.DataSource = _ClientList;
      ClientCBox.ValueMember = "ClientId";
      ClientCBox.DisplayMember = "FIO";

      _ServiceList = _ServiceProvider.GetAllService();
      ServiceCBox.DataSource = _ServiceList;
      ServiceCBox.ValueMember = "ServiceId";
      ServiceCBox.DisplayMember = "ServiceName";

      _ManagersList = _ManagersProvider.GetAllManagers();
      ManagersCBox.DataSource = _ManagersList;
      ManagersCBox.ValueMember = "ManagersId";
      ManagersCBox.DisplayMember = "FIO";

    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (TripGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = TripGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _TripList = _TripRaportBLL.GetAllTrip();
        LoadDataInGridView(_TripList);
```

```
    if (_selectedRowIndex == TripGridView.Rows.Count) {
      _selectedRowIndex = TripGridView.Rows.Count - 1;
    }
    if (_selectedRowIndex >= 0) {
      TripGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
      TripGridView.Rows[_selectedRowIndex].Selected = true;
    }
  } catch { }
}

private void LoadDataInGridView(List<Trip> TripList) {
  TripGridView.DataSource = null;
  TripGridView.Columns.Clear();
  TripGridView.AutoGenerateColumns = false;
  TripGridView.RowHeadersVisible = false;

  TripGridView.DataSource = TripList;

  if (TripList.Count > 0) {
    if (TripList[0].Message == NamesMy.NoDataNames.NoDataInTrip) {
      DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
      messageColumn.DataPropertyName = "Message";
      messageColumn.Width = TripGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
      TripGridView.Columns.Add(messageColumn);
    } else {
      DataGridViewColumn AppointmentSheetIdColumn = new DataGridViewTextBoxColumn();
      AppointmentSheetIdColumn.DataPropertyName = "TripId";
      TripGridView.Columns.Add(AppointmentSheetIdColumn);
      TripGridView.Columns[0].Visible = false;

      DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
      numberColumn.HeaderText = "№ ";
      numberColumn.DataPropertyName = "Number";
      numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      numberColumn.Width = NamesMy.SizeOptins.NumberSize;
      TripGridView.Columns.Add(numberColumn);

      DataGridViewColumn StartDateColumn = new DataGridViewTextBoxColumn();
      StartDateColumn.HeaderText = "Початок поїздки";
      StartDateColumn.DataPropertyName = "StartDate";
      StartDateColumn.Width = NamesMy.SizeOptins.Date;
      TripGridView.Columns.Add(StartDateColumn);


      DataGridViewColumn EndDateColumn = new DataGridViewTextBoxColumn();
      EndDateColumn.HeaderText = "Кінець поїздки";
      EndDateColumn.DataPropertyName = "EndDate";
      EndDateColumn.Width = NamesMy.SizeOptins.Date;
      TripGridView.Columns.Add(EndDateColumn);

      DataGridViewColumn ClientFIOColumn = new DataGridViewTextBoxColumn();
      ClientFIOColumn.HeaderText = "Клієнт";
      ClientFIOColumn.DataPropertyName = "ClientFIO";
      ClientFIOColumn.Width = 180;
      TripGridView.Columns.Add(ClientFIOColumn);

      DataGridViewColumn ManagersFIOColumn = new DataGridViewTextBoxColumn();
      ManagersFIOColumn.HeaderText = "Менеджер";
      ManagersFIOColumn.DataPropertyName = "ManagersFIO";
      ManagersFIOColumn.Width = 180;
      TripGridView.Columns.Add(ManagersFIOColumn);

      DataGridViewColumn ServiceNameColumn = new DataGridViewTextBoxColumn();
      ServiceNameColumn.HeaderText = "Послуга";
```

```
          ServiceNameColumn.DataPropertyName = "ServiceName";
          ServiceNameColumn.Width = NamesMy.SizeOptins.NameSize;
          TripGridView.Columns.Add(ServiceNameColumn);

        }
        for (int i = 0; i < TripGridView.Columns.Count; i++) {
          TripGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
        }
      }
    }

    private void ClearAllControls() {
      ClientCBox.SelectedValue = 0;
      ManagersCBox.SelectedValue = 0;
      ServiceCBox.SelectedValue = 0;
      StartDateDTP.Value = DateTime.Now;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (Convert.ToInt32(ClientCBox.SelectedValue) > 0) {
        ClientValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ClientValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (Convert.ToInt32(ManagersCBox.SelectedValue) > 0) {
        ManagersValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ManagersValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (Convert.ToInt32(ServiceCBox.SelectedValue) > 0) {
        ServiceValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ServiceValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void TripGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && TripGridView[0, e.RowIndex].Value.ToString() !=
_TripList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateTripForm updateCategoryForm = new UpdateTripForm(Convert.ToInt32(TripGridView[0,
e.RowIndex].Value.ToString()));
        updateCategoryForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Лістинг класу «UpdateTripForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Providers;

namespace TravelAgenci.Forms.Trips {
  public partial class UpdateTripForm : Form {
    private int _TripId = 0;
    private Trip _selectedTrip = new Trip();
    private TripProvider _TripProvider = new TripProvider();
    private ValidationMy _Validation = new ValidationMy();
    private ClientProvider _ClientProvider = new ClientProvider();
    private List<Client> _ClientList = new List<Client>();
    private ServiceProvider _ServiceProvider = new ServiceProvider();
    private List<Service> _ServiceList = new List<Service>();
    private ManagersProvider _ManagersProvider = new ManagersProvider();
    private List<Managers> _ManagersList = new List<Managers>();

    public UpdateTripForm(int TripId) {
      InitializeComponent();
      _TripId = TripId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _TripProvider.UpdateTrip(Convert.ToInt32(ManagersCBox.SelectedValue),
Convert.ToInt32(ClientCBox.SelectedValue),
          Convert.ToInt32(ServiceCBox.SelectedValue), StartDateDTP.Value, EndDTP.Value,
_TripId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Вы действительно хотите удалить этот элемент?", "Удалить",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _TripProvider.DeleteTripByTripId(_TripId);
        this.Close();
      }
    }

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

    private void LoadAllDate() {
      _ClientList = _ClientProvider.GetAllClient();
      ClientCBox.DataSource = _ClientList;
      ClientCBox.ValueMember = "ClientId";
      ClientCBox.DisplayMember = "FIO";

      _ServiceList = _ServiceProvider.GetAllService();
      ServiceCBox.DataSource = _ServiceList;
      ServiceCBox.ValueMember = "ServiceId";
      ServiceCBox.DisplayMember = "ServiceName";

      _ManagersList = _ManagersProvider.GetAllManagers();
      ManagersCBox.DataSource = _ManagersList;
      ManagersCBox.ValueMember = "ManagersId";
      ManagersCBox.DisplayMember = "FIO";

      _selectedTrip = _TripProvider.SelectedTripByTripId(_TripId);
      ManagersCBox.SelectedValue = _selectedTrip.ManagersId;
      ClientCBox.SelectedValue = _selectedTrip.ClientId;
```

```
      ServiceCBox.SelectedValue = _selectedTrip.ServiceId;
      StartDateDTP.Value = _selectedTrip.StartDate;
      EndDTP.Value = _selectedTrip.EndDate;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (Convert.ToInt32(ClientCBox.SelectedValue) > 0) {
        ClientValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ClientValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (Convert.ToInt32(ManagersCBox.SelectedValue) > 0) {
        ManagersValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ManagersValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (Convert.ToInt32(ServiceCBox.SelectedValue) > 0) {
        ServiceValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ServiceValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }

      return isCorrect;
    }

  }
}
```

Лістинг класу «ClientProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TravelAgenci.AppCode;

namespace TravelAgenci.Providers {
  class ClientProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];
    public void InsertClient(string LastName, string FirstName, string Phone, string Address,
string Email) {
      string SqlString = "INSERT INTO Client (LastName, FirstName, Phone, Address, " +
        "Email) Values(?, ?, ?, ?, ?)";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("LastName", LastName);
          cmd.Parameters.AddWithValue("FirstName", FirstName);
          cmd.Parameters.AddWithValue("Phone", Phone);
          cmd.Parameters.AddWithValue("Address", Address);
          cmd.Parameters.AddWithValue("Email", Email);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
```

```
  }
}

public List<Client> GetAllClient() {
  int i = 0;
  string SqlString = "SELECT ClientId, FirstName, LastName, Phone, Address, " +
    "Email FROM Client";

  List<Client> listAllClient = new List<Client>();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          Client oneClient = new Client();
          oneClient.Number = ++i;
          oneClient.ClientId = Convert.ToInt32(reader["ClientId"].ToString());
          oneClient.FirstName = reader["FirstName"].ToString();
          oneClient.LastName = reader["LastName"].ToString();
          oneClient.FIO = oneClient.LastName + " " + oneClient.FirstName;
          oneClient.Phone = reader["Phone"].ToString();
          oneClient.Address = reader["Address"].ToString();
          oneClient.Email = reader["Email"].ToString();

          listAllClient.Add(oneClient);
        }
      }
      conn.Close();
    }
  }

  if (listAllClient.Count == 0) {
    Client noClient = new Client();
    noClient.ClientId = 0;
    noClient.Message = NamesMy.NoDataNames.NoDataInClient;
    listAllClient.Add(noClient);
  }
  return listAllClient;
}

public Client SelectedClientByClientId(int ClientId) {
  string SqlString = "SELECT ClientId, FirstName, LastName, Phone, Address, " +
    "Email FROM Client Where ClientId=" + ClientId.ToString();

  Client oneClient = new Client();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          oneClient.ClientId = Convert.ToInt32(reader["ClientId"].ToString());
          oneClient.FirstName = reader["FirstName"].ToString();
          oneClient.LastName = reader["LastName"].ToString();
          oneClient.FIO = oneClient.LastName + " " + oneClient.FirstName;
          oneClient.Phone = reader["Phone"].ToString();
          oneClient.Address = reader["Address"].ToString();
          oneClient.Email = reader["Email"].ToString();
        }
      }
    }
    conn.Close();
  }
  return oneClient;
}
```

```
    public void UpdateClient(string LastName, string FirstName, string Phone, string Address,
string Email, int ClientId) {
      string SqlString = "UPDATE Client SET FirstName=?, LastName=?, Phone=?, " +
  "Address=?, Email=? WHERE ClientId=?";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("FirstName", FirstName);
          cmd.Parameters.AddWithValue("LastName", LastName);
          cmd.Parameters.AddWithValue("Phone", Phone);
          cmd.Parameters.AddWithValue("Address", Address);
          cmd.Parameters.AddWithValue("Email", Email);
          cmd.Parameters.AddWithValue("ClientId", ClientId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public void DeleteClientByClientId(int ClientId) {
      string SqlString = "DELETE FROM Client WHERE ClientId=" + ClientId.ToString();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

  }
}


public class Client {
  private int _Number;
  private int _ClientId;
  private string _FirstName;
  private string _LastName;
  private string _Phone;
  private string _Address;
  private string _Email;
  private string _FIO;
  private string _Message;

  public Client() {
    _Number = 0;
    _ClientId = 0;
    _FirstName = String.Empty;
    _LastName = String.Empty;
    _Phone = String.Empty;
    _Address = String.Empty;
    _Email = String.Empty;
    _FIO = String.Empty;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
```

```
  public int ClientId {
    set { _ClientId = value; }
    get { return _ClientId; }
  }
  public string FirstName {
    set { _FirstName = value; }
    get { return _FirstName; }
  }
  public string LastName {
    set { _LastName = value; }
    get { return _LastName; }
  }
  public string Phone {
    set { _Phone = value; }
    get { return _Phone; }
  }
  public string Address {
    set { _Address = value; }
    get { return _Address; }
  }
  public string Email {
    set { _Email = value; }
    get { return _Email; }
  }
  public string FIO {
    set { _FIO = value; }
    get { return _FIO; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг класу «CountryProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TravelAgenci.AppCode;

namespace TravelAgenci.Providers {
  class CountryProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertCountry(string CountryName, string Description) {
      string SqlString = "INSERT INTO Country (CountryName, Description) Values(?, ?)";
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("CountryName", CountryName);
          cmd.Parameters.AddWithValue("Description", Description);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }
```

```
public List<Country> GetAllCountry() {
  int i = 0;
  string SqlString = "SELECT * FROM Country ORDER BY CountryName ASC";
  List<Country> listAllCountry = new List<Country>();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          Country oneCountry = new Country();
          oneCountry.Number = ++i;
          oneCountry.CountryId = Convert.ToInt32(reader["CountryId"]);
          oneCountry.CountryName = reader["CountryName"].ToString();
          listAllCountry.Add(oneCountry);
        }
      }
      conn.Close();
    }
  }

  if (listAllCountry.Count == 0) {
    Country noCountry = new Country();
    noCountry.CountryId = 0;
    noCountry.Message = NamesMy.NoDataNames.NoDataInCountry;
    listAllCountry.Add(noCountry);
  }
  return listAllCountry;
}


public Country SelectedCountryByCountryId(int CountryId) {
  string SqlString = "SELECT * FROM Country Where CountryId=" + CountryId.ToString();

  Country oneCountry = new Country();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          oneCountry.CountryId = Convert.ToInt32(reader["CountryId"]);
          oneCountry.CountryName = reader["CountryName"].ToString();
          oneCountry.Description = reader["Description"].ToString();
        }
      }
    }
    conn.Close();
  }
  return oneCountry;
}

public void UpdateCountry(string CountryName, string Description, int CountryId) {
  string SqlString = "UPDATE Country SET CountryName=?, Description=?  WHERE CountryId=?";

  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      cmd.Parameters.AddWithValue("CountryName", CountryName);
      cmd.Parameters.AddWithValue("Description", Description);
      cmd.Parameters.AddWithValue("CountryId", CountryId);
      conn.Open();
      cmd.ExecuteNonQuery();
      conn.Close();
    }
  }
```

```
    }

    public void DeleteCountryByCountryId(int CountryId) {
      string SqlString = "DELETE FROM Country WHERE CountryId=" + CountryId.ToString();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

  }
}

public class Country {
  private int _Number;
  private int _CountryId;
  private string _CountryName;
  private string _Description;
  private string _Message;

  public Country() {
    _Number = 0;
    _CountryId = 0;
    _CountryName = String.Empty;
    _Description = String.Empty;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int CountryId {
    set { _CountryId = value; }
    get { return _CountryId; }
  }
  public string CountryName {
    set { _CountryName = value; }
    get { return _CountryName; }
  }
  public string Description {
    set { _Description = value; }
    get { return _Description; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }

}
```

Лістинг класу «LogsProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TravelAgenci.AppCode;

namespace TravelAgenci.Providers {
```

```
  class LogsProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];
    public void InsertLogs(int UsersId, string EventNameShow, DateTime EvendDate) {
      string SqlString = "INSERT INTO Logs (UsersId, EventNameShow, EvendDate) Values(?, ?,
?)";
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("UsersId", UsersId);
          cmd.Parameters.AddWithValue("EventNameShow", EventNameShow);
          cmd.Parameters.AddWithValue("EvendDate", EvendDate.ToString());
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public List<Logs> GetAllLogs() {
      int i = 0;
      string SqlString = "SELECT Logs.LogsId, Logs.UsersId, Logs.EventNameShow,
Logs.EvendDate, Users.UsersName " +
      "FROM Logs INNER JOIN Users ON Users.UsersId = Logs.UsersId  ORDER BY Logs.EvendDate
DESC";
      List<Logs> listAllLogs = new List<Logs>();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              Logs oneLogs = new Logs();
              oneLogs.Number = ++i;
              oneLogs.LogsId = Convert.ToInt32(reader["LogsId"]);
              oneLogs.UsersId = Convert.ToInt32(reader["UsersId"]);
              oneLogs.EventNameShow = reader["EventNameShow"].ToString();
              oneLogs.EvendDate = Convert.ToDateTime(reader["EvendDate"]);
              oneLogs.UsersName = reader["UsersName"].ToString();
              listAllLogs.Add(oneLogs);
            }
          }
          conn.Close();
        }
      }

      if (listAllLogs.Count == 0) {
        Logs noLogs = new Logs();
        noLogs.LogsId = 0;
        noLogs.Message = NamesMy.NoDataNames.NoDataInLogs;
        listAllLogs.Add(noLogs);
      }
      return listAllLogs;

    }

  }
}

public class Logs {
  private int _Number;
  private int _LogsId;
  private int _UsersId;
  private string _UsersName;
  private string _EventNameShow;
```

```
  private DateTime _EvendDate;
  private string _Message;

  public Logs() {
    _Number = 0;
    _LogsId = 0;
    _UsersId = 0;
    _UsersName = String.Empty;
    _EventNameShow = String.Empty;
    _EvendDate = new DateTime();
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int LogsId {
    set { _LogsId = value; }
    get { return _LogsId; }
  }
  public int UsersId {
    set { _UsersId = value; }
    get { return _UsersId; }
  }
  public string UsersName {
    set { _UsersName = value; }
    get { return _UsersName; }
  }
  public string EventNameShow {
    set { _EventNameShow = value; }
    get { return _EventNameShow; }
  }
  public DateTime EvendDate {
    set { _EvendDate = value; }
    get { return _EvendDate; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг класу «ManagersProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TravelAgenci.AppCode;

namespace TravelAgenci.Providers {
  class ManagersProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];
    public void InsertManagers(string LastName, string FirstName, string Phone, string
Address, string Email) {
      string SqlString = "INSERT INTO Managers (LastName, FirstName, Phone, Address, " +
        "Email) Values(?, ?, ?, ?, ?)";
```

```
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      cmd.CommandType = CommandType.Text;
      cmd.Parameters.AddWithValue("LastName", LastName);
      cmd.Parameters.AddWithValue("FirstName", FirstName);
      cmd.Parameters.AddWithValue("Phone", Phone);
      cmd.Parameters.AddWithValue("Address", Address);
      cmd.Parameters.AddWithValue("Email", Email);
      conn.Open();
      cmd.ExecuteNonQuery();
      conn.Close();
    }
  }
}

public List<Managers> GetAllManagers() {
  int i = 0;
  string SqlString = "SELECT ManagersId, FirstName, LastName, Phone, Address, " +
    "Email FROM Managers";

  List<Managers> listAllManagers = new List<Managers>();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          Managers oneManagers = new Managers();
          oneManagers.Number = ++i;
          oneManagers.ManagersId = Convert.ToInt32(reader["ManagersId"].ToString());
          oneManagers.FirstName = reader["FirstName"].ToString();
          oneManagers.LastName = reader["LastName"].ToString();
          oneManagers.FIO = oneManagers.LastName + " " + oneManagers.FirstName;
          oneManagers.Phone = reader["Phone"].ToString();
          oneManagers.Address = reader["Address"].ToString();
          oneManagers.Email = reader["Email"].ToString();

          listAllManagers.Add(oneManagers);
        }
      }
      conn.Close();
    }
  }

  if (listAllManagers.Count == 0) {
    Managers noManagers = new Managers();
    noManagers.ManagersId = 0;
    noManagers.Message = NamesMy.NoDataNames.NoDataInManagers;
    listAllManagers.Add(noManagers);
  }
  return listAllManagers;
}

public Managers SelectedManagersByManagersId(int ManagersId) {
  string SqlString = "SELECT ManagersId, FirstName, LastName, Phone, Address, " +
    "Email FROM Managers Where ManagersId=" + ManagersId.ToString();

  Managers oneManagers = new Managers();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          oneManagers.ManagersId = Convert.ToInt32(reader["ManagersId"].ToString());
          oneManagers.FirstName = reader["FirstName"].ToString();
```

```
              oneManagers.LastName = reader["LastName"].ToString();
              oneManagers.FIO = oneManagers.LastName + " " + oneManagers.FirstName;
              oneManagers.Phone = reader["Phone"].ToString();
              oneManagers.Address = reader["Address"].ToString();
              oneManagers.Email = reader["Email"].ToString();
            }
          }
        }
        conn.Close();
      }
      return oneManagers;
    }

    public void UpdateManagers(string LastName, string FirstName, string Phone, string
Address, string Email, int ManagersId) {
      string SqlString = "UPDATE Managers SET FirstName=?, LastName=?, Phone=?, " +
  "Address=?, Email=? WHERE ManagersId=?";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("FirstName", FirstName);
          cmd.Parameters.AddWithValue("LastName", LastName);
          cmd.Parameters.AddWithValue("Phone", Phone);
          cmd.Parameters.AddWithValue("Address", Address);
          cmd.Parameters.AddWithValue("Email", Email);
          cmd.Parameters.AddWithValue("ManagersId", ManagersId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public void DeleteManagersByManagersId(int ManagersId) {
      string SqlString = "DELETE FROM Managers WHERE ManagersId=" + ManagersId.ToString();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

  }
}


public class Managers {
  private int _Number;
  private int _ManagersId;
  private string _FirstName;
  private string _LastName;
  private string _Phone;
  private string _Address;
  private string _Email;
  private string _FIO;
  private string _Message;

  public Managers() {
    _Number = 0;
    _ManagersId = 0;
    _FirstName = String.Empty;
```

```
    _LastName = String.Empty;
    _Phone = String.Empty;
    _Address = String.Empty;
    _Email = String.Empty;
    _FIO = String.Empty;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int ManagersId {
    set { _ManagersId = value; }
    get { return _ManagersId; }
  }
  public string FirstName {
    set { _FirstName = value; }
    get { return _FirstName; }
  }
  public string LastName {
    set { _LastName = value; }
    get { return _LastName; }
  }
  public string Phone {
    set { _Phone = value; }
    get { return _Phone; }
  }
  public string Address {
    set { _Address = value; }
    get { return _Address; }
  }
  public string Email {
    set { _Email = value; }
    get { return _Email; }
  }
  public string FIO {
    set { _FIO = value; }
    get { return _FIO; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг класу «ServiceProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TravelAgenci.AppCode;

namespace TravelAgenci.Providers {
  class ServiceProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];
```

```
    public void InsertService(int CountryId, int CityId, string LivingAndTravelConditions,
string ExcursionService, string HostService, double Price, string ServiceName) {
      string SqlString = "INSERT INTO Service (CountryId, CityId, LivingAndTravelConditions,
ExcursionService, HostService, Price, " +
        "ServiceName) Values(?, ?, ?, ?, ?, ?, ?)";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("CountryId", CountryId);
          cmd.Parameters.AddWithValue("CityId", CityId);
          cmd.Parameters.AddWithValue("LivingAndTravelConditions", LivingAndTravelConditions);
          cmd.Parameters.AddWithValue("ExcursionService", ExcursionService);
          cmd.Parameters.AddWithValue("HostService", HostService);
          cmd.Parameters.AddWithValue("Price", Price);
          cmd.Parameters.AddWithValue("ServiceName", ServiceName);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public List<Service> GetAllService() {
      int i = 0;
      string SqlString = "SELECT * FROM Service";

      List<Service> listAllService = new List<Service>();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              Service oneService = new Service();
              oneService.Number = ++i;
              oneService.ServiceId = Convert.ToInt32(reader["ServiceId"]);
              oneService.CountryId = Convert.ToInt32(reader["CountryId"]);
              oneService.CityId = Convert.ToInt32(reader["CityId"]);
              oneService.LivingAndTravelConditions =
reader["LivingAndTravelConditions"].ToString();
              oneService.ExcursionService = reader["ExcursionService"].ToString();
              oneService.HostService = reader["HostService"].ToString();
              oneService.Price = Convert.ToDouble(reader["Price"]);
              oneService.ServiceName = reader["ServiceName"].ToString();
              listAllService.Add(oneService);
            }
          }
          conn.Close();
        }
      }

      if (listAllService.Count == 0) {
        Service noService = new Service();
        noService.ServiceId = 0;
        noService.Message = NamesMy.NoDataNames.NoDataInService;
        listAllService.Add(noService);
      }
      return listAllService;
    }

    public Service SelectedServiceByServiceId(int ServiceId) {
      string SqlString = "SELECT * FROM Service Where ServiceId=" + ServiceId.ToString();

      Service oneService = new Service();
```

```
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              oneService.ServiceId = Convert.ToInt32(reader["ServiceId"]);
              oneService.CountryId = Convert.ToInt32(reader["CountryId"]);
              oneService.CityId = Convert.ToInt32(reader["CityId"]);
              oneService.LivingAndTravelConditions =
reader["LivingAndTravelConditions"].ToString();
              oneService.ExcursionService = reader["ExcursionService"].ToString();
              oneService.HostService = reader["HostService"].ToString();
              oneService.Price = Convert.ToDouble(reader["Price"]);
              oneService.ServiceName = reader["ServiceName"].ToString();
            }
          }
        }
        conn.Close();
      }
      return oneService;
    }

    public void UpdateService(int CountryId, int CityId, string LivingAndTravelConditions,
string ExcursionService, string HostService, double Price, string ServiceName, int ServiceId)
{
      string SqlString = "UPDATE Service SET CountryId=?, CityId=?,
LivingAndTravelConditions=?, ExcursionService=?, HostService=?, Price=?, " +
  "ServiceName=? WHERE ServiceId=?";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("CountryId", CountryId);
          cmd.Parameters.AddWithValue("CityId", CityId);
          cmd.Parameters.AddWithValue("LivingAndTravelConditions", LivingAndTravelConditions);
          cmd.Parameters.AddWithValue("ExcursionService", ExcursionService);
          cmd.Parameters.AddWithValue("HostService", HostService);
          cmd.Parameters.AddWithValue("Price", Price);
          cmd.Parameters.AddWithValue("ServiceName", ServiceName);
          cmd.Parameters.AddWithValue("ServiceId", ServiceId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public void DeleteServiceByServiceId(int ServiceId) {
      string SqlString = "DELETE FROM Service WHERE ServiceId=" + ServiceId.ToString();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }
  }
}

public class Service {
  private int _Number;
  private int _ServiceId;
  private int _CountryId;
```

```
  private int _CityId;
  private string _LivingAndTravelConditions; // условия проживания и проезда
  private string _ExcursionService; // экскурсионное обслуживание
  private string _HostService; // сервис принимающей стороны
  private double _Price; //стоимость путевки
  private string _ServiceName;
  private string _Message;

  public Service() {
    _Number = 0;
    _ServiceId = 0;
    _CountryId = 0;
    _CityId = 0;
    _LivingAndTravelConditions = String.Empty;
    _ExcursionService = String.Empty;
    _HostService = String.Empty;
    _Price = 0.0;
    _ServiceName = String.Empty;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int ServiceId {
    set { _ServiceId = value; }
    get { return _ServiceId; }
  }
  public int CountryId {
    set { _CountryId = value; }
    get { return _CountryId; }
  }
  public int CityId {
    set { _CityId = value; }
    get { return _CityId; }
  }
  public string LivingAndTravelConditions {
    set { _LivingAndTravelConditions = value; }
    get { return _LivingAndTravelConditions; }
  }
  public string ExcursionService {
    set { _ExcursionService = value; }
    get { return _ExcursionService; }
  }
  public string HostService {
    set { _HostService = value; }
    get { return _HostService; }
  }
  public double Price {
    set { _Price = value; }
    get { return _Price; }
  }
  public string ServiceName {
    set { _ServiceName = value; }
    get { return _ServiceName; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг класу «TripProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TravelAgenci.AppCode;

namespace TravelAgenci.Providers {
  class TripProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertTrip(int ManagersId, int ClientId, int ServiceId, DateTime StartDate,
DateTime EndDate) {
      string SqlString = "INSERT INTO Trip (ManagersId, ClientId, ServiceId, StartDate, " +
        "EndDate) Values(?, ?, ?, ?, ?)";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("ManagersId", ManagersId);
          cmd.Parameters.AddWithValue("ClientId", ClientId);
          cmd.Parameters.AddWithValue("ServiceId", ServiceId);
          cmd.Parameters.AddWithValue("StartDate", StartDate.ToString());
          cmd.Parameters.AddWithValue("EndDate", EndDate.ToString());
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public List<Trip> GetAllTrip() {
      ClientProvider ClientProvider = new ClientProvider();
      List<Client> ClientList = new List<Client>();
      ClientList = ClientProvider.GetAllClient();

      int i = 0;
      string SqlString = "SELECT * FROM Trip";

      List<Trip> listAllTrip = new List<Trip>();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              Trip oneTrip = new Trip();
              oneTrip.Number = ++i;
              oneTrip.TripId = Convert.ToInt32(reader["TripId"]);
              oneTrip.ManagersId = Convert.ToInt32(reader["ManagersId"]);
              oneTrip.ClientId = Convert.ToInt32(reader["ClientId"]);
              oneTrip.ServiceId = Convert.ToInt32(reader["ServiceId"]);
              oneTrip.StartDate = Convert.ToDateTime(reader["StartDate"]);
              oneTrip.EndDate = Convert.ToDateTime(reader["EndDate"]);
              listAllTrip.Add(oneTrip);
            }
          }
          conn.Close();
        }
      }
```

```
      if (listAllTrip.Count == 0) {
        Trip noTrip = new Trip();
        noTrip.TripId = 0;
        noTrip.Message = NamesMy.NoDataNames.NoDataInTrip;
        listAllTrip.Add(noTrip);
      }
      return listAllTrip;
    }

    public Trip SelectedTripByTripId(int TripId) {
      string SqlString = "SELECT TripId, FirstName, LastName, Phone, Address, " +
        "Email FROM Trip Where TripId=" + TripId.ToString();

      Trip oneTrip = new Trip();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              oneTrip.TripId = Convert.ToInt32(reader["TripId"]);
              oneTrip.ManagersId = Convert.ToInt32(reader["ManagersId"]);
              oneTrip.ClientId = Convert.ToInt32(reader["ClientId"]);
              oneTrip.ServiceId = Convert.ToInt32(reader["ServiceId"]);
              oneTrip.StartDate = Convert.ToDateTime(reader["StartDate"]);
              oneTrip.EndDate = Convert.ToDateTime(reader["EndDate"]);
            }
          }
        }
        conn.Close();
      }
      return oneTrip;
    }

    public void UpdateTrip(int ManagersId, int ClientId, int ServiceId, DateTime StartDate,
DateTime EndDate, int TripId) {
      string SqlString = "UPDATE Trip SET ManagersId=?, ClientId=?, ServiceId=?, " +
  "StartDate=?, Email=? WHERE TripId=?";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("ManagersId", ManagersId);
          cmd.Parameters.AddWithValue("ClientId", ClientId);
          cmd.Parameters.AddWithValue("ServiceId", ServiceId);
          cmd.Parameters.AddWithValue("StartDate", StartDate);
          cmd.Parameters.AddWithValue("EndDate", EndDate);
          cmd.Parameters.AddWithValue("TripId", TripId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public void DeleteTripByTripId(int TripId) {
      string SqlString = "DELETE FROM Trip WHERE TripId=" + TripId.ToString();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }
```

```
  }
}


public class Trip {
  private int _Number;
  private int _TripId;
  private int _ManagersId;
  private string _ManagersFIO;
  private int _ClientId;
  private string _ClientFIO;
  private int _ServiceId;
  private string _ServiceName;
  private DateTime _StartDate;
  private DateTime _EndDate;
  private double _Price;
  private string _Message;

  public Trip() {
    _Number = 0;
    _TripId = 0;
    _ManagersId = 0;
    _ManagersFIO = String.Empty;
    _ClientId = 0;
    _ClientFIO = String.Empty;
    _ServiceId = 0;
    _ServiceName = String.Empty;
    _StartDate = new DateTime();
    _EndDate = new DateTime();
    _Price = 0.0;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int TripId {
    set { _TripId = value; }
    get { return _TripId; }
  }
  public int ManagersId {
    set { _ManagersId = value; }
    get { return _ManagersId; }
  }
  public string ManagersFIO {
    set { _ManagersFIO = value; }
    get { return _ManagersFIO; }
  }
  public int ClientId {
    set { _ClientId = value; }
    get { return _ClientId; }
  }
  public string ClientFIO {
    set { _ClientFIO = value; }
    get { return _ClientFIO; }
  }
  public int ServiceId {
    set { _ServiceId = value; }
    get { return _ServiceId; }
  }
  public string ServiceName {
    set { _ServiceName = value; }
    get { return _ServiceName; }
```

```
  }
  public DateTime StartDate {
    set { _StartDate = value; }
    get { return _StartDate; }
  }
  public DateTime EndDate {
    set { _EndDate = value; }
    get { return _EndDate; }
  }
  public double Price {
    set { _Price = value; }
    get { return _Price; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг класу «UsersProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TravelAgenci.AppCode;

namespace TravelAgenci.Providers {
  class UsersProvider {
    private EncryptData _encryptData = new EncryptData();
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertUsers(string FirstName, string LastName, string UsersName, string
UsersPassword,
   int RoleId, string Description) {
      string SqlString = "INSERT INTO Users (FirstName, LastName, UsersName, UsersPassword, "
+
       "RoleId, Description) Values(?, ?, ?, ?, ?, ?)";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("FirstName", FirstName);
          cmd.Parameters.AddWithValue("LastName", LastName);
          cmd.Parameters.AddWithValue("UsersName", UsersName);
          cmd.Parameters.AddWithValue("UsersPassword", _encryptData.Encrypt(UsersPassword));
          cmd.Parameters.AddWithValue("RoleId", RoleId);
          cmd.Parameters.AddWithValue("Description", Description);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public List<Users> GetAllUsers() {
      int i = 0;
      string SqlString = "SELECT * FROM Users ORDER BY LastName ASC";
      List<Users> listAllUsers = new List<Users>();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
```

```
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              Users oneUsers = new Users();
              oneUsers.Number = ++i;
              oneUsers.UsersId = Convert.ToInt32(reader["UsersId"]);
              oneUsers.FirstName = reader["FirstName"].ToString();
              oneUsers.LastName = reader["LastName"].ToString();
              oneUsers.FIO = oneUsers.LastName + " " + oneUsers.FirstName;
              oneUsers.UsersName = reader["UsersName"].ToString();
              oneUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString());
              oneUsers.RoleId = Convert.ToInt32(reader["RoleId"]);
              oneUsers.RoleName = GetRoleName(oneUsers.RoleId);
              oneUsers.Description = reader["Description"].ToString();
              listAllUsers.Add(oneUsers);
            }
          }
          conn.Close();
        }
      }

      if (listAllUsers.Count == 0) {
        Users noUsers = new Users();
        noUsers.UsersId = 0;
        noUsers.Message = NamesMy.NoDataNames.NoDataInUsers;
        listAllUsers.Add(noUsers);
      }
      return listAllUsers;
    }

    private string GetRoleName(int RoleId) {
      RoleApp roleApp = new RoleApp();
      for (int i = 0; i < roleApp.GetRoleList().Count(); i++) {
        if (RoleId == roleApp.GetRoleList()[i].RoleId) {
          return roleApp.GetRoleList()[i].RoleName;
        }
      }
      return "";
    }

    public Users SelectedUsersByUsersId(int UsersId) {
      string SqlString = "SELECT * FROM Users WHERE UsersId=" + UsersId.ToString();

      Users oneUsers = new Users();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              oneUsers.UsersId = Convert.ToInt32(reader["UsersId"]);
              oneUsers.FirstName = reader["FirstName"].ToString();
              oneUsers.LastName = reader["LastName"].ToString();
              oneUsers.UsersName = reader["UsersName"].ToString();
              oneUsers.FIO = oneUsers.LastName + " " + oneUsers.FirstName;
              oneUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString());
              oneUsers.RoleId = Convert.ToInt32(reader["RoleId"]);
              oneUsers.Description = reader["Description"].ToString();
            }
          }
        }
        conn.Close();
      }
```

```
      return oneUsers;
    }

    public List<Users> GetAllUsersListForCBox() {
      string SqlString = "SELECT UsersId, UsersName, UsersPassword FROM Users ORDER BY
UsersName ASC";
      List<Users> listAllUsers = new List<Users>();

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              Users oneUsers = new Users();
              oneUsers.UsersId = Convert.ToInt32(reader["UsersId"].ToString());
              oneUsers.UsersName = reader["UsersName"].ToString();
              oneUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString());
              listAllUsers.Add(oneUsers);
            }
          }
          conn.Close();
        }
      }

      if (listAllUsers.Count == 0) {
        Users noUsers = new Users();
        noUsers.UsersId = 0;
        noUsers.Message = NamesMy.NoDataNames.NoDataInUsers;
        listAllUsers.Add(noUsers);
      }
      return listAllUsers;
    }

    public List<Users> SelectedUsersByUsersNameAndUsersPassword(string UsersName, string
UsersPassword) {
      string SqlString = "SELECT * FROM Users WHERE UsersName='" + UsersName + "' AND
UsersPassword='" + _encryptData.Encrypt(UsersPassword) + "'";
      List<Users> UsersList = new List<Users>();

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              Users oneUsers = new Users();
              oneUsers.UsersId = Convert.ToInt32(reader["UsersId"]);
              oneUsers.FirstName = reader["FirstName"].ToString();
              oneUsers.LastName = reader["LastName"].ToString();
              oneUsers.UsersName = reader["UsersName"].ToString();
              oneUsers.FIO = oneUsers.LastName + " " + oneUsers.FirstName;
              oneUsers.UsersPassword =
_encryptData.Decrypt(reader["UsersPassword"].ToString());
              oneUsers.RoleId = Convert.ToInt32(reader["RoleId"]);
              oneUsers.Description = reader["Description"].ToString();
              UsersList.Add(oneUsers);
            }
          }
        }
        conn.Close();
      }
      return UsersList;
    }
```

```
    public void UpdateUsers(string FirstName, string LastName, string UsersName, string
UsersPassword,
   int RoleId, string Description, int UsersId) {
      string SqlString = "UPDATE Users SET FirstName=?, LastName=?, " +
   "UsersName=?, UsersPassword=?, RoleId=?, Description=? WHERE UsersId=?";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("FirstName", FirstName);
          cmd.Parameters.AddWithValue("LastName", LastName);
          cmd.Parameters.AddWithValue("UsersName", UsersName);
          cmd.Parameters.AddWithValue("UsersPassword", _encryptData.Encrypt(UsersPassword));
          cmd.Parameters.AddWithValue("RoleId", RoleId);
          cmd.Parameters.AddWithValue("Description", Description);
          cmd.Parameters.AddWithValue("UsersId", UsersId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public void DeleteUsersByUsersId(int UsersId) {
      string SqlString = "DELETE FROM Users WHERE UsersId=" + UsersId.ToString();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

  }
}


public class Users {
  private int _Number;
  private int _UsersId;
  private string _FirstName;
  private string _LastName;
  private string _UsersName;
  private string _FIO;
  private string _UsersPassword;
  private int _RoleId;
  private string _RoleName;
  private string _Description;
  private string _Message;

  public Users() {
    _UsersId = 0;
    _FirstName = String.Empty;
    _LastName = String.Empty;
    _UsersName = String.Empty;
    _FIO = String.Empty;
    _UsersPassword = String.Empty;
    _RoleId = 0;
    _Description = String.Empty;
  }

  public int Number {
    set { _Number = value; }
```

```
    get { return _Number; }
  }
  public int UsersId {
    set { _UsersId = value; }
    get { return _UsersId; }
  }
  public string FirstName {
    set { _FirstName = value; }
    get { return _FirstName; }
  }
  public string LastName {
    set { _LastName = value; }
    get { return _LastName; }
  }
  public string FIO {
    set { _FIO = value; }
    get { return _FIO; }
  }
  public string UsersName {
    set { _UsersName = value; }
    get { return _UsersName; }
  }
  public string UsersPassword {
    set { _UsersPassword = value; }
    get { return _UsersPassword; }
  }
  public int RoleId {
    set { _RoleId = value; }
    get { return _RoleId; }
  }
  public string RoleName {
    set { _RoleName = value; }
    get { return _RoleName; }
  }
  public string Description {
    set { _Description = value; }
    get { return _Description; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг класу «CityProvider»

```
using System;
using System.Data;
using System.Collections.Generic;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TravelAgenci.AppCode;

namespace TravelAgenci.Providers {
  class CityProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertCity(string CityName, string Description) {
      string SqlString = "INSERT INTO City (CityName, Description) Values(?, ?)";
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
```

```
      cmd.CommandType = CommandType.Text;
      cmd.Parameters.AddWithValue("CityName", CityName);
      cmd.Parameters.AddWithValue("Description", Description);
      conn.Open();
      cmd.ExecuteNonQuery();
      conn.Close();
    }
  }
}

public List<City> GetAllCity() {
  int i = 0;
  string SqlString = "SELECT * FROM City ORDER BY CityName ASC";
  List<City> listAllCity = new List<City>();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          City oneCity = new City();
          oneCity.Number = ++i;
          oneCity.CityId = Convert.ToInt32(reader["CityId"]);
          oneCity.CityName = reader["CityName"].ToString();
          listAllCity.Add(oneCity);
        }
      }
      conn.Close();
    }
  }

  if (listAllCity.Count == 0) {
    City noCity = new City();
    noCity.CityId = 0;
    noCity.Message = NamesMy.NoDataNames.NoDataInCity;
    listAllCity.Add(noCity);
  }
  return listAllCity;
}


public City SelectedCityByCityId(int CityId) {
  string SqlString = "SELECT * FROM City Where CityId=" + CityId.ToString();

  City oneCity = new City();
  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
    using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
      conn.Open();
      using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
          oneCity.CityId = Convert.ToInt32(reader["CityId"]);
          oneCity.CityName = reader["CityName"].ToString();
          oneCity.Description = reader["Description"].ToString();
        }
      }
    }
    conn.Close();
  }
  return oneCity;
}

public void UpdateCity(string CityName, string Description, int CityId) {
  string SqlString = "UPDATE City SET CityName=?, Description=?  WHERE CityId=?";

  using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
```

```
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("CityName", CityName);
          cmd.Parameters.AddWithValue("Description", Description);
          cmd.Parameters.AddWithValue("CityId", CityId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public void DeleteCityByCityId(int CityId) {
      string SqlString = "DELETE FROM City WHERE CityId=" + CityId.ToString();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }
  }
}


public class City {
  private int _Number;
  private int _CityId;
  private string _CityName;
  private string _Description;
  private string _Message;

  public City() {
    _Number = 0;
    _CityId = 0;
    _CityName = String.Empty;
    _Description = String.Empty;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int CityId {
    set { _CityId = value; }
    get { return _CityId; }
  }
  public string CityName {
    set { _CityName = value; }
    get { return _CityName; }
  }
  public string Description {
    set { _Description = value; }
    get { return _Description; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

Лістинг класу «TravelAgenciMDI»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TravelAgenci.AppCode;
using TravelAgenci.Forms.Dictionary;
using TravelAgenci.Forms.Raport;
using TravelAgenci.Forms.Systems;
using TravelAgenci.Forms.Trips;

namespace TravelAgenci {
  public partial class TravelAgenciMDI : Form {

    public TravelAgenciMDI() {
      InitializeComponent();
    }

    public void CloseAllWindows() {
      Form[] childArray = this.MdiChildren;
      foreach (Form childForm in childArray) {
        childForm.Close();
      }
    }

    private void ПодорожіToolStripMenuItem_Click(object sender, EventArgs e) {
      CloseAllWindows();
      TripForm tripForm = new TripForm();
      tripForm.MdiParent = this;
      tripForm.WindowState = FormWindowState.Maximized;
      tripForm.Show();
    }

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

    private void КлієнтиToolStripMenuItem_Click(object sender, EventArgs e) {
      CloseAllWindows();
      ClientForm clientForm = new ClientForm();
      clientForm.MdiParent = this;
      clientForm.WindowState = FormWindowState.Maximized;
      clientForm.Show();
    }

    private void МенеджериToolStripMenuItem_Click(object sender, EventArgs e) {
      CloseAllWindows();
      ManagersForm managersForm = new ManagersForm();
      managersForm.MdiParent = this;
      managersForm.WindowState = FormWindowState.Maximized;
      managersForm.Show();
    }

    private void ПослугиToolStripMenuItem_Click(object sender, EventArgs e) {
      CloseAllWindows();
      ServiceForm serviceForm = new ServiceForm();
      serviceForm.MdiParent = this;
      serviceForm.WindowState = FormWindowState.Maximized;
      serviceForm.Show();
```

```
    }

    private void КраїниToolStripMenuItem_Click(object sender, EventArgs e) {
      CloseAllWindows();
      CountryForm countryForm = new CountryForm();
      countryForm.MdiParent = this;
      countryForm.WindowState = FormWindowState.Maximized;
      countryForm.Show();
    }

    private void МістаToolStripMenuItem_Click(object sender, EventArgs e) {
      CloseAllWindows();
      CityForm cityForm = new CityForm();
      cityForm.MdiParent = this;
      cityForm.WindowState = FormWindowState.Maximized;
      cityForm.Show();
    }

    private void КористувачіToolStripMenuItem_Click(object sender, EventArgs e) {
      if (LoginForm.CurrentUser.RoleId == 1) {
        CloseAllWindows();
        UsersForm usersForm = new UsersForm();
        usersForm.MdiParent = this;
        usersForm.WindowState = FormWindowState.Maximized;
        usersForm.Show();
      } else {
        MessageBox.Show(NamesMy.MessageBoxExaption.YouDontHavePermission);
      }
    }

    private void СистемнийЖурналToolStripMenuItem_Click(object sender, EventArgs e) {
      if (LoginForm.CurrentUser.RoleId == 1) {
        CloseAllWindows();
        SystemLogForm systemLogForm = new SystemLogForm();
        systemLogForm.MdiParent = this;
        systemLogForm.WindowState = FormWindowState.Maximized;
        systemLogForm.Show();
      } else {
        MessageBox.Show(NamesMy.MessageBoxExaption.YouDontHavePermission);
      }
    }

    private void КлієнтиToolStripMenuItem1_Click(object sender, EventArgs e) {
      CloseAllWindows();
      RaportClientForm raportClientForm = new RaportClientForm();
      raportClientForm.MdiParent = this;
      raportClientForm.WindowState = FormWindowState.Maximized;
      raportClientForm.Show();

    }

    private void ПослугиToolStripMenuItem1_Click(object sender, EventArgs e) {
      CloseAllWindows();
      RaportServiseForm raportServiseForm = new RaportServiseForm();
      raportServiseForm.MdiParent = this;
      raportServiseForm.WindowState = FormWindowState.Maximized;
      raportServiseForm.Show();
    }

    private void TravelAgenciMDI_Resize(object sender, EventArgs e) {
      this.BackgroundImage = Properties.Resources.back_gr;
    }
  }
}
```