МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Програмний продукт з графічним інтерфейсом на мові C# та за допомогою технології Window Forms»

Завідувач
випускаючої кафедри **Довбиш А. С.**

Керівник роботи **Колесніков В.А.**

Студента гр. ІНз-71с **Чайка Б.Ю.**

СУМИ 2021

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук
Секція комп'ютерних наук

Напрям підготовки – 6.050101 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Зав. кафедрою

_____ А.С. Довбиш
«_____»_____ 2021 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Чайка Богдан Юрійович

1 Тема роботи *Програмний додаток для підтримки роботи міжнародного аеропорту*

Керівник роботи *Колесніков В.А.*

затверджені наказом по університету від «__» травня 2021 р. №_____

2 Строк подання студентом роботи «__» червня 2021 р.

3 Вхідні дані до роботи технічне завдання на розробку програмного продукту для підтримки роботи міжнародного аеропорту; перелік типових завдань, які ставляться перед програмою

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, визначення варіантів використання програмного продукту, проектування інтерфейсу програми, розробка програмного забезпечення на мові програмування C#

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність проблеми, вимоги до проекту, структура програми, проектування програмного додатку, тестування програми.

6. Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення з предметною областю	28.01.21–05.02.21	
2	Визначення вимог до додатку	06.02.21–15.02.21	
3	Огляд аналогів	16.02.21–23.02.21	
4	Техніко-економічне дослідження	25.02.19–03.03.21	
5	Оцінка ІТ-проекту	05.03.21–11.03.21	
6	Планування структури роботи	13.03.21–17.03.21	
7	Визначення структури програми	20.03.21–29.03.21	
8	Створення прототипу програмного додатку	02.04.21–14.04.21	
9	Визначення внутрішньої структури програмного забезпечення	15.04.21–28.04.21	
10	Розробка користувацького інтерфейсу	01.05.21–10.05.21	
11	Об’єднання системи з користувацьким інтерфейсом	11.05.21–17.05.21	
12	Створення інсталяційного пакету	18.05.21–21.05.21	
13	Тестування програми	22.05.21–31.05.21	
14	Виправлення помилок	01.06.21–05.06.21	
15	Створення інструкції користувача	06.06.21–10.06.21	
16	Формування звітів	11.06.21–15.06.21	
17	Підготовка презентації проекту	16.06.21–18.06.21	
18	Захист бакалаврської роботи	19.06.21–25.06.21	

Студент _____ **Чайка Б. Ю.**

(підпис)

Керівник роботи _____ **Колесніков В.А.**

(підпис)

РЕФЕРАТ

Записка: стор. 67, рис. 18, додаток 5 , джерел 16.

Об’єкт дослідження – інфраструктурних мереж авіаліній

Мета роботи – розробити програмний додаток для робочого місця диспетчера авіаліній, для здійснення моніторингу інфраструктурної мережі

Методи дослідження – експертний аналіз аналогів, керування ризиками проекту, автоматичне тестування програмного забезпечення, робота з базою даних

Результати – програмний додаток, який представляє із себе робоче місце диспетчера-авіаліній для моніторингу та керування авіаліній

ЗМІСТ

ВСТУП

Кількість населення в світі неупинно зростає, уже зараз популяція людей досягає 7.8 мільярдів осіб [1]. Наша планета стає все більш тіснішою.

Сучасні люди мають можливість без проблем долати значні відстані, наприклад, для зміни місця проживання або подорожей до нових країн. Найшвидшим та, в деякому сенсі, найдешевшим засобом подорожування є авіалайнери, за допомогою яких можна виконати переміщення на десятки тисяч кілометрів за ліченні години, в зручних умовах.

Цивільна авіація представляє із себе надскладну інфраструктурну мережу, яка охоплює майже всі країни світу. Її головною метою – є переміщення людей та їх багажу між віддаленими містами. Так, наприклад, Boeing 747-400 [2] є одним із найпопулярніших літаків цивільного класу, він здатний перевозити до 642-х пасажирів на відстань понад 14 200 км, цього достатньо, щоб дістатися з Києву до Пекіну і повернутися назад.

Інфраструктура авіаліній є дуже розгалуженою та складною, при формуванні авіарейсів необхідно враховувати безліч факторів про літак і не тільки. Це робиться для того, щоб мінімізувати можливі ризики та аварії. Під час бакалаврської роботи буде розроблений програмний додаток, який дозволить адміністратору формувати звіти на основі наявної, в базі даних, інформації.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- проаналізувати предметну область;
- розробити структуру програмного додатку;
- виявити типові запити до бази даних;
- спроектувати програму;
- провести тестування;

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1. Актуальність проблеми

На перший погляд може здатися, що корегування польоту літака набагато легше ніж аналогічне керування автомобіля, адже в повітрі, із-за великої кількості вільного простору, транспортним засобам досить складно врізатися один в одного. Проте тут є маса нюансів.

По-перше, літак завжди розрахований для польоту на конкретну відстань. У небі немає зупинок або заправок. Пілот точно знає коли і куди він повинен посадити авіалайнер. Це означає, що в аеропорту в назначений час повинно бути місце для посадки літака, адже той не може просто чекати у повітрі.

По-друге, пасажирські авіалайнери є дуже габаритними, наприклад AirBus A380 [3], який зображений на рисунку 1.1. має довжину 73 метра. Така купа металу є дуже не поворотною не тільки на землі а й в повітрі. Тому вкрай важливо надати літаку вільне місце та достатньо часу для посадки або зльоту.

Рисунок 1.1 – Загальний вигляд AirBus A380

Слід пам’ятати, що будь-який літак дуже вразливий в момент посадки, адже часу і місця для маневрів просто немає.

Звісно, що це далеко не всі складності, які слід враховувати при корегуванні польоту, як мінімум, є ще нюанси безпосереднього управління літального засобу, але за це відповідає бортова команда разом з капітаном. У бакалаврській роботі розглядається лише питання організації інфраструктурної мережі перельотів, тобто одну із ключових задач, що виконує персонал

аеропорту. Вона полягає в узгодженні рейсів, які прямують з різних країн. Так на рисунку 1.2 показано фрагмент ГІС AirNav [4], яка представляє із себе динамічну карту авіа перельотів. За допомогою неї можна оцінити масштаб та розмір сучасної авіаційної інфроструктури.

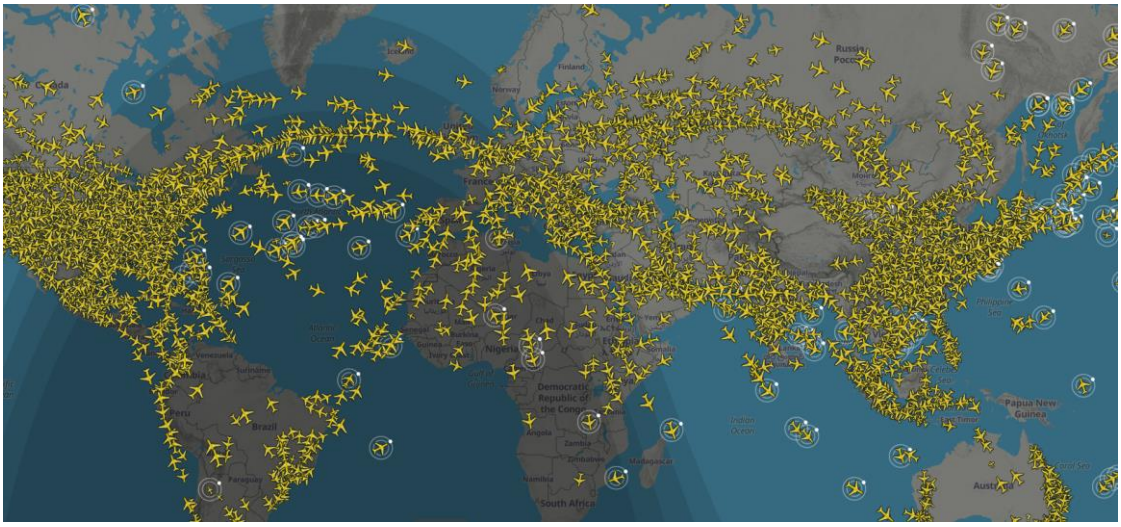


Рисунок 1.2 – Приклад візуалізації рейсів

Для грамотного керування диспетчер повинен враховувати усі літаки, які летять до його аеропорту та ті, що вже готуються до зльоту. Організувати їх час та забезпечити персонал усім необхідним.

1.2. Огляд існуючих аналогів

Щоб здійснювати моніторинг літаків у повітрі використовують радіонавігаційні супутникові системи. Більшість сучасних літальних засобів орієнтуються на GPS[5] або ГЛОНАСС[6]. Принцип дії цих систем дуже схожий і у більшості випадках вони доповнюють один одну.

Для визначення глобальних координат об’єкту на орбіті, приблизна висота 20 000 кілометрів, знаходяться 24 супутники. Вони розташовані таким чином, щоб одночасно з будь-якої точки Землі можна було б побачити, як мінімум, 4 з них а максимум 12. У кожного супутника є атомний годинник, який визначає час з точністю до наносекунд. Будь-який об’єкт, неважливо наземний чи повітряний, визначає свої координати в залежності від отриманого часу сигналу з різних супутників, схематично це показано на рисунку 1.3.

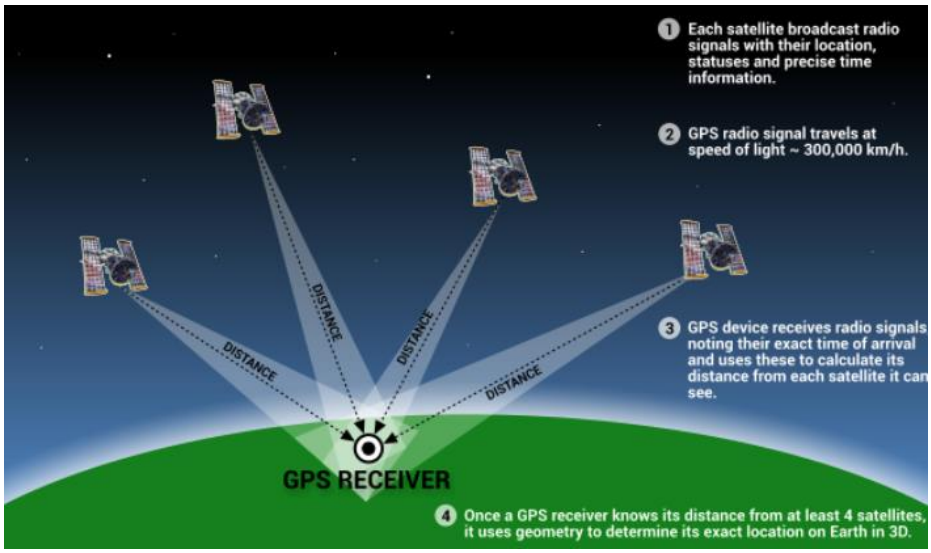


Рисунок 1.3 – Приклад роботи радіонавігаційної супутникової системи

Для того, щоб якнайкраще зрозуміти предметну область був зроблений короткий огляд існуючих, доступних програмних засобів, які використовуються для корегування зареєстрованих літаків. До таких програм відноситься уже згадана AirNav [4].

AirNav [4] – це повноцінна геоінформаційна система, яка призначена для спостереження за авіарейсами в реальному часі. Вона реалізована у вигляді веб-додатку з обмеженим безкоштовним доступом, який надає наступні можливості:

- динамічне відображення літаків на карті світу;
- можливість виокремити рейси певної країни або конкретного аеропорту;
- розгорнута характеристика літаків;
- ранжування літальних засобів за швидкістю, висотою польоту тощо;

Слід закцентувати увагу на тому, що AirNav [4] – геоінформаційна система і, скоріш за все, дані, які вона надає, є дещо надлишковими для диспетчера. Проте це один із найкращих варіантів для мандрівника, бо в такий спосіб можна отримати багато візуально зрозумілої інформації.

CrossPoint – це професійне диспетчерське програмне забезпечення, яке надає широкий спектр інструментів для роботи з моніторинговою інформацією отриманої через радіонавігаційну супутникову систему ГЛОНАСС.

Можна виділити наступні переваги цього продукту:

- робота з широким спектром задач, зокрема з моніторингу авіарейсів;
- клієнт-серверна архітектура, що дозволяє виконувати моніторинг як одній людині так і цілій команді;
- зручна інсталяція/деінсталяція програми, автоматичне оновлення через Інтернет;
- автоматичне формування звітів;

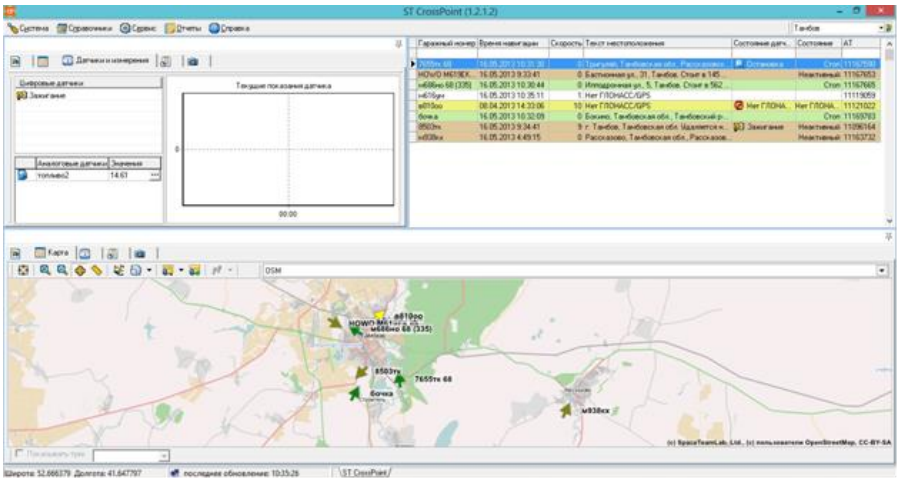


Рисунок 1.3 – Приклад інтерфейсу CrossPoint

Серед недоліків можна виділити орієнтованість на радіонавігаційну систему ГЛОНАСС, що може призвести до не точного визначення координат об’єкту, та додаткові налаштування для специфічних задач. Річ у тім, що CrossPoint призначена для моніторингу найрізноманітніших інфраструктурних мереж, тому її необхідно додатково адаптувати під конкретні потреби, що викликає ряд не зручностей.

На офіційному сайті міжнародних авіаліній України є досить зручна система моніторингу авірейсів [7]. За допомогою неї можна отримати наступну інформацію:

- заплановані на сьогодні рейси;
- доступні маршрути;
- детальну інформацію про конкретний рейс;
- збереження даних за минулі дні;

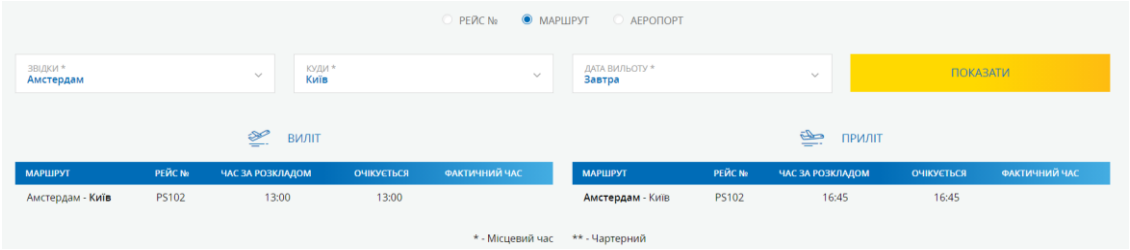


Рисунок 1.4 – Приклад інтерфейсу додатку для моніторингу міжнародних авіаліній України

Мінімалістичний і простий в розумінні інтерфейс, рисунок 1.4, є однозначною перевагою. Проте, нажаль, не надається ніякої додаткової інформації про літак а також немає можливості сформувати автоматичний звіт, на основі отриманої інформації.

Таким чином, було розглянуто ряд типових представників додатків, які використовуються для моніторингу авіаційного трафіку. Можна зробити висновок, що програмне забезпечення, яке призначене для вирішення цієї задачі, повинно вміти працювати з базою даних, яку частково заповнює людина (номера рейсів, пункт призначення тощо) іншу ж частину слід отримувати з радіонавігаційної супутникової системи (координати об’єкту, відстань до пункту призначення, висота на якій знаходиться об’єкт тощо). Сформовані дані бажано подавати у вигляді звіту.

Важливо зосередитися на реалізації функціональних завдань, уникати нагромадження надлишкової інформації. Інтерфейс повинен бути простим і зрозумілим.

1.3. Постановка задачі

Головною метою бакалаврської роботи можна вважати розрозробку програмного забезпечення для моніторингу міжнародних авіаліній. Така

програма дозволить адміністратору організовувати управління авіаційних рейсів, за допомогою отримання та доповнення інформації з бази даних. Отримані результати можна буде представити у вигляді word-звітів.

У перспективі розроблену базу даних програми можна буде доповнити інформацією, яка була отримана за допомогою радіонавігаційної системи. Як один із можливих напрямків покращення програмного комплексу слід розглядати створення геоінформаційної системи моніторингу.

Базовий варіант програми повинен підтримувати наступні функції:

- можливість взаємодіяти з базою даних авіарейсів;
- автоматично формувати word-звіти;

Були виділені наступні типові запити до бази даних:

- пошук літака за його серійним номером;
- видалення/додавання нового рейсу;
- отримання списку літаків, що налітали годин більше ніж X;
- пошук літаків за маршрутом X;
- отримання списку усіх доступних літаків;

Програмний продукт буде поставлятися у вигляді інсталятору, який самостійно встановить програму та створить БД MS Access з усією необхідною документацією на робочому місці адміністратора.

Для реалізації поставленої мети в дипломному проекті необхідно виконати ряд під задач:

- провести аналіз предметної області;
- визначити варіанти використання програмного продукту;
- описати діаграми подій;
- розробити інтерфейс програмного забезпечення;
- виконати тестування програми та створити авто-тести;
- створити інсталятор для кінцевого продукту;

- написати необхідну документацію, яка буде додана до програми;

2. ОПИС МЕТОДУ ДОСЛІДЖЕННЯ

2.1 Визначення вимог

Вимоги до технології розробки:

Продукт розробляється ітеративно із врахуванням принципів та технологій уніфікованого процесу створення програмного забезпечення. До етапу здачі проекту необхідно надати мінімум один прототип, який виконує базові функції, які були вказані в постановці задачі. Програма повинна бути написана мовою C# із використанням технології Microsoft .Net.

Вимоги до інформаційної та програмної сумісності:

Для коректної роботи додатку необхідний .Net Framework 4.5 або вище та стабільне підключення до мережі інтернет.

Інтерфейс продукту повинен бути оформлений української мовою. Усі елементи інтерфейсу повинні бути попередньо узгодженні.

Додаток створений для використання на платформі Windows із встановленими Word та Access, які є частиною пакету Microsoft Office, при роздільній здатності екрану не нижче 1024x768, ОЗУ від 2 ГБ.

Вимоги до функціональних характеристик

Додаток повинен забезпечувати виконання наступних функцій:

- Запис даних у БД, а саме: номер, маршрут, ID, загальний час польотів;
- Редагування даних в базі даних;
- У кожному записі зберігається інформація:
  - номер;
  - маршрут;
  - загальний час польотів;
- Виконання типових запитів до БД:
  - пошук літака за його серійним номером;
  - видалення/додавання нового рейсу;
  - отримання списку літаків, що налітали годин більше ніж X;
  - пошук літаків за маршрутом X;

- отримання списку усіх доступних літаків;

- Експорт отриманих результатів в файл MS Word;

2.2 Визначення структури програми

Для передання внутрішньої структури програмного забезпечення була застосована модель варіантів використання.

Перш за все слід виділити акторів, таблиця 2.1.

Таблиця 2.1 – Актори, які беруть участь у роботі системи

Актори-користувачі	
Ім'я актору	**Опис**
Користувач	Будь-яка людина, яка використовує можливості програмного продукту. Взаємодія відбувається через користувацький інтерфейс.
Актори-системи	
MS Word	Отримує інформацію від користувача, формує на її основі звіт.
MS Access	Зберігає потрібну інформацію та дозволяє її використовувати.

Після попереднього аналізу вимог слід виділити варіанти використання майбутнього продукту, таблиці 2.2.

Таблиця 2.2 – Варіанти використання розробленої програми

1. Додавання інформії(Add)	Передумови: Програма повинна бути запущена. Обрана вкладка «Редагування». Процес: Editor повинен ввести корректні данні у відповідні поля на вкладці «редагування» та натиснути кнопку «добавити» Післяумови: БД була оновлена та нові данні були добавлені
2. Видалення інформації(Delete)	Передумови: Програма повинна бути запущена. Обрана вкладка «Редагування». Процес: Editor повинен обрати

	потрібній йому запис на вкладці «редагування» та натиснути кнопку «видалити». Післяумови: БД була оновлена. Запис був видалений.
3. Вивести вміст бази даних(Show All)	Передумови: Програма повинна бути запущена. Обрана вкладка «Редагування». Процес: Editor натискає кнопку «Вивести вміст бази даних». Післяумови: У відповідному вікні з'явилася інформація про всі рейси які є у БД.
4. Пошук літака за номером(Find by number)	Передумови: Програма повинна бути запущена. Обрана вкладка «Редагування». Процес: Editor вводить у відповідне поле номер рейсу та натискає кнопку «Пошук літака за номером». У відповідному вікні з'являється рейс із заданим номером. Післяумови: БД оновлена.
5. Пошук літаків, що налітали більше ніж n годин(Find by hours)	Передумови: Програма повинна бути запущена. Обрана вкладка «Вивід». Процес: Editor у відповідне поле вводить корректні данні та натискає кнопку «пошук». Якщо данні були знайдені, то у відповідному місці виведеться інформація про рейси які налітали більше ніж n годин. Післяумови: БД оновлена.
6. Пошук літаків за маршрутом(Find by route)	Передумови: Програма повинна бути запущена. Обрана вкладка «Вивід». Процес: Editor у відповідне поле вводить корректні данні та натискає кнопку «пошук». Якщо данні були знайдені, то у відповідному місці виведеться інформація про рейси які налітали більше ніж n годин. Післяумови: БД оновлена.

7. Зберегти результат(Write Data)	Передумови: Програма повинна бути запущена. Обрана вкладка «Вивід». Данні в БД повинні бути оновлені. Процес: Після знаходження потрібної інформації, Editor натискає кнопку «Зберегти результат» Післяумови: Створений файл result.docx в з'явилася потрібна інформація

У загальному вигляді описані варіанти використання (табл.2.2) можна подати як діаграму [13], яка показана на рисунку 2.1.

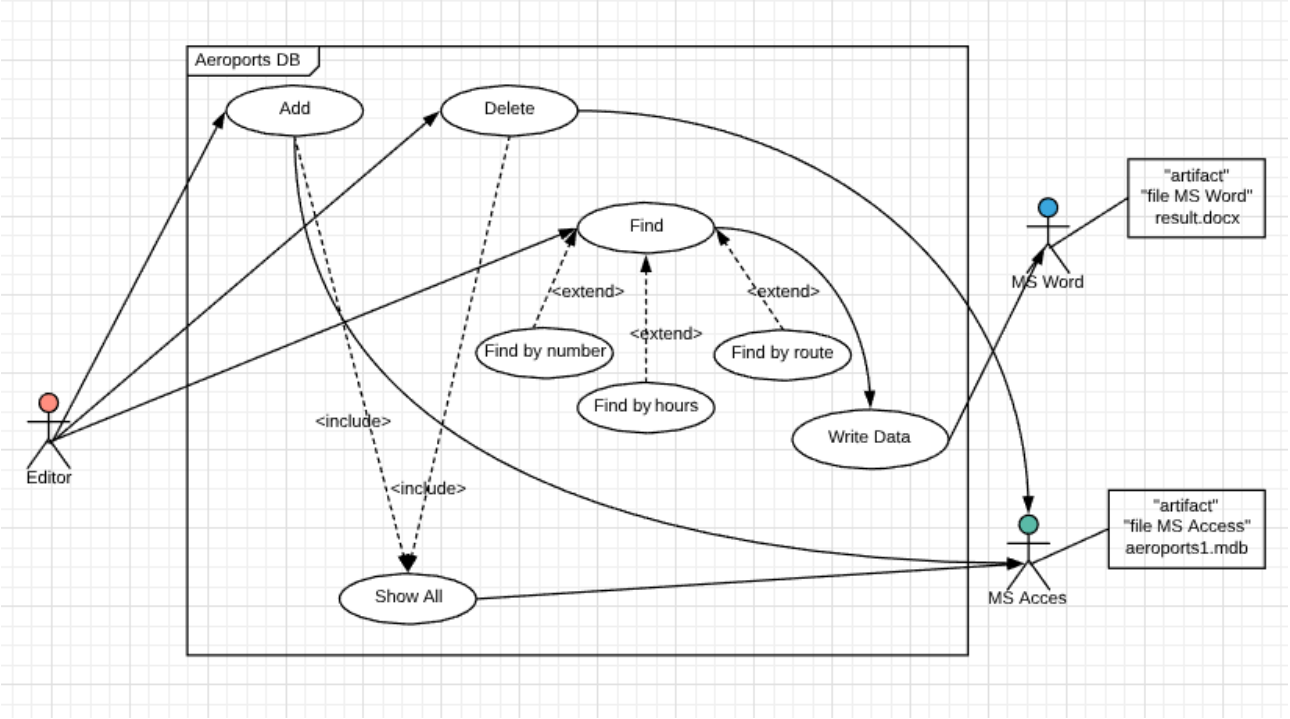


Рисунок 2.1 – Діаграма взаємодії

Для представлення безпосереднього процесу розробки була побудована нотація IDEF0. У додатку А детально описаний процес планування робіт для реалізації проекту.

Діаграми IDEF0 представляє із себе методологію функціонального моделювання для графічного представлення бізнес-процесів. Її головною особливістю є те, що в ній розглядається лише логічне відношення, тобто

послідовність робіт, без часової залежності. При цьому кожен блок, який створений у рамках цієї методології, представляє із себе «чорний ящик», що має входи, виходи та механізми управління. Деталізація виконується до рівня, який необхідний розробникам [8].

Перший блок нотації IDEF0 має назву A-0. Він дає загальне представлення процесу, який в подальшому, за необхідності, можна деталізувати. Для проекту «Програмний додаток для підтримки роботи міжнародного аеропорту» контекстна діаграма A-0 складається з даних, що описані нижче:

- Вхідні дані: досвід, інформація з зовнішніх джерел;
- Вихідні дані: готовий продукт;
- Управління: постановка задачі, модель програмного продукту (ПП), реалізація програмно забезпечення (ПЗ), тестування;
- Механізми: менеджер проекту, інженер компонентів, розробник коду, тестувальник, розробник БД, розробник користувацького інтерфейсу, розробник документації;

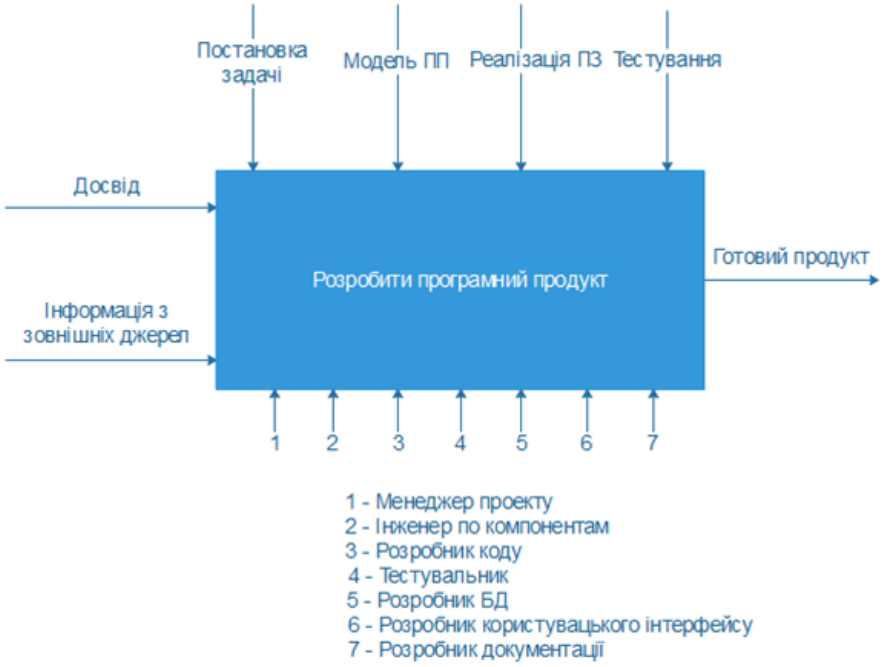


Рисунок 2.2 – Діаграма A-0

Для початку роботи, представлених, на рисунку 2.2, процесів цілком достатньо. Подальша деталізація буде здійснюватися за необхідністю.

2.3 Внутрішня структура програмного додатку

Враховуючи функціональні вимоги, які були описані в 2.1, та складені варіанти використання майбутнього продукту, розділ 2.2, було сформульовано ключові особливості інтерфейсу програми:

1) додати окреме вікно для виведення в нього інформації з бази даних;
2) за допомогою окремого поля виконувати пошук літака за його номером;
3) додати частину інтерфейсу для додавання/видалення запису з БД. При цьому необхідна наявність усіх ключових полів: номер літака, маршрут, загальний час польоту;
4) виділити місце для здійснення пошуку літака згідно типових запитів;
5) додати невелике повідомлення, яке коротко описує програму;
6) надати користувачу можливість відкрити «Інструкцію користувача» прямо із програми;

Як основна база даних була вибрана БД на основі MS Access. Такий вибір пов’язаний із тим, що більшість інших аналогів, наприклад MySql, потребують створення серверу, а це додаткові складнощі у розгортанні програми. Створена, у рамках бакалаврської роботи, програма позбута такого недоліку, вона здатна повноцінно функціонувати вже після інсталяції. У подальших ревізіях можна перейти на серверну базу даних без зміни запитів, оскільки вони, все рівно, пишуться на мові MySql.

При розробці внутрішньої структури програми було розглянуто різні схеми розміщення контенту:

- лінійна;

- ієрархічна;
- мережева;
- комбінована;

Лінійна структура компанує сторінки таки чином, щоб вони посилалися одна на одну і одночасно на головну. Такий підхід зарекомендував себе у додатках, які представляють із себе портфоліо, презентацію або будь-який інший ресурс, який створений для послідовного ознайомлення користувача з усім наявним контентом починаючи з головної сторінки. Згідно роботи [8], лінійна структура, рисунок 2.3, найкраще підходить тим, для кого не принципово отримувати трафік з пошуку.

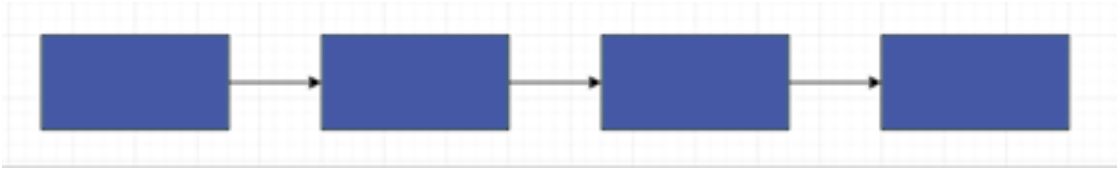


Рисунок 2.3 – Схема лінійної структури

Ієрархічна структура подібна до лінійної, з тієї відмінністю, що тут може бути застосовано і описано кілька продуктів. Найпростіший приклад такої схеми – це онлайн бібліотека, де на сторінці кожного автора можна переглянути усі його твори. При збільшенні вимог до створеного програмного застосунку, такий підхід компонування контенту може бути виправданим, але зараз це не раціонально. На рисунку 2.4 представлена схема ієрархічної структури розміщення інформації.

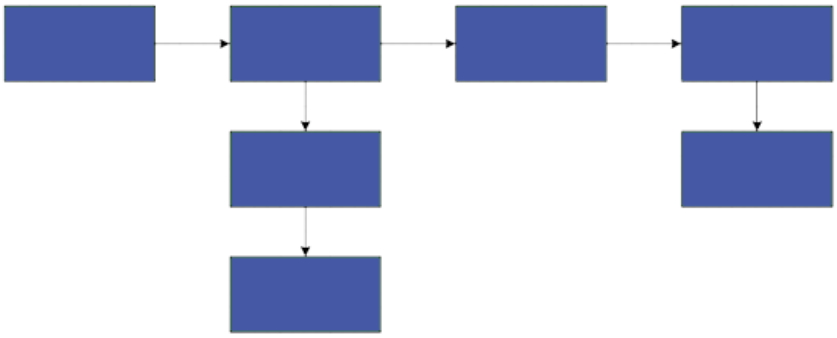


Рисунок 2.4 – Схема ієрархічної структури

Мережева структура – це підхід при якому рівнозначні між собою сторінки пов’язуються посиланнями один на одну. Цю конструкцію можна використати для подання інформації про один конкретний продукт, розміщуючи окремі записи з описом його переваг, характеристик, властивостей. Як і в лінійній структурі, усі блоки пов’язані між собою, з будь-якої частини додатку можна повернутися на головну сторінку. Приклад мережевої схеми розміщення інформації в додатку представлений на рисунку 2.5.

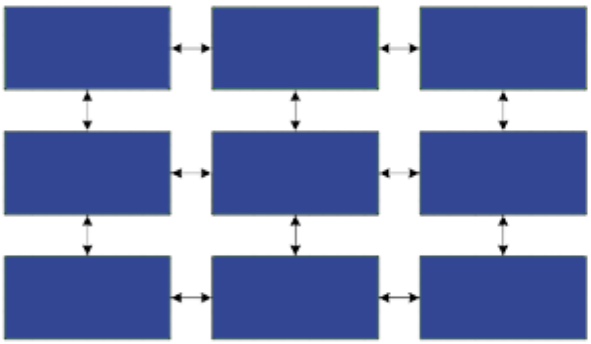


Рисунок 2.5 – Схема мережевої структури

Враховуючі усі розглянуті підходи розміщення контенту для програмного забезпечення, було прийнято рішення побудувати комбіновану структуру. Таким чином вдасться найефективніше скомпонувати наявну зараз інформацію. Для розроблюваного програмного додатку була обрана модифікована версія мережевої структури, рисунок 2.6.

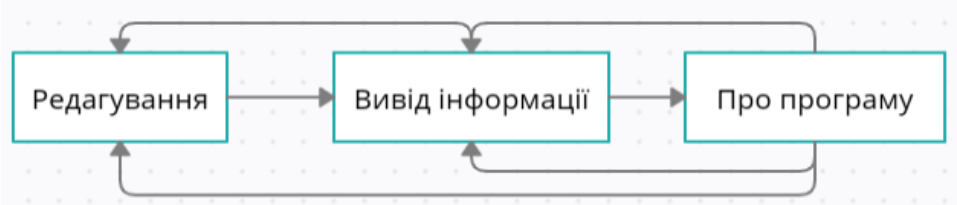


Рисунок 2.6 – Внутрішня структура програмного забезпечення

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Проектування інтерфейсу

Інтерфейс – це унікальна складова будь-якого програмного продукту. Він дозволяє користувачу взаємодіяти з кодом не втручаючись в нього. Слід пам’ятати, що усі програмні забезпечення застосовуються для спрощення деякої роботи людини, тому інтерфейс повинен бути лаконічним і зрозумілим, щоб не ускладнювати взаємодію із програмою. Існує важлива закономірність, яка дозволяє при меншій кількості зусиль ознайомитися з можливостями ПП та зрозуміти принцип його робити [9].

Було прийнято рішення почати розробку з назви програмного додатку. Згідно ресурсу [10] ім’я будь-якого комерційного продукту повинно тісно бути пов’язано або з компанією розробника або з специфікою самої програми. Наприклад, такі фірми як Adobe, Microsoft, AutoCAD, MatLAB використовують у назвах свого програмного забезпечення ім’я корпорації розробника та власну назву продукту. Програмний додаток для підтримки роботи міжнародного аеропорту було вирішено назвати «AeroportsDB».

Наступним кроком буде створення логотипу програми, який по сумісництву повинен стати ще й значком для .exe-файлу додатку. Згідно ресурсу [11] лого повинно бути простим та зрозумілим, без не доцільних деталей. Зображення має натякати на сенс та призначення ПЗ. Оскільки логотип буде використаний ще й як іконка .exe-файлу, то він не повинен містити текст, бо його все рівно не вдасться розгледіти. Найкращим варіантом для композиції буде один або декілька простих об’єктів, які розкриватимуть суть програми. В результаті було обраний наступний логотип, рисунок 3.1.

Рисунок 3.1 – Логотип розробленої програми

Згідно структури, яка була визначена у розділі 2.3, був побудований інтерфейс, який складається із трьох вкладок:

- *Редагування*. Здійснюється взаємодія з базою даних;
- *Вивід*. Виконання типових запитів до бази даних, формування звіту;
- *Про програму*. Короткий опис програмного забезпечення;

На рисунку 3.2 можна побачити загальний вигляд програми при відкритому вікні «Редагування».

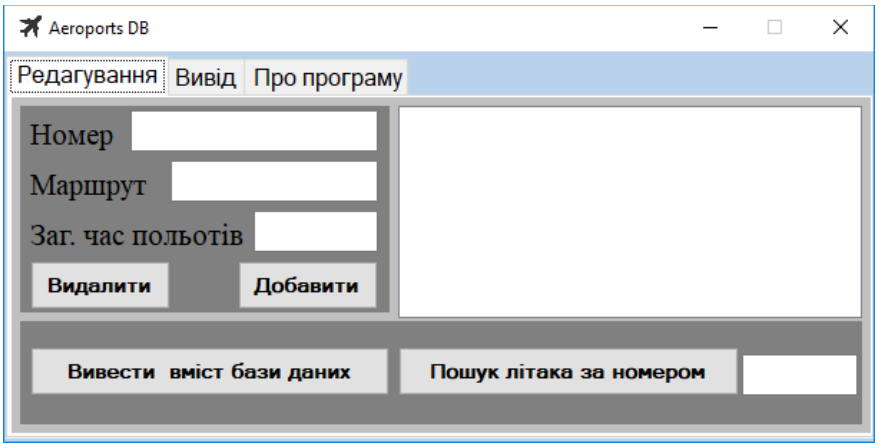


Рисунок 3.3 – Загальний вигляд програми

При натисканні на кнопку «Вивести вміст даних» користувач отримає поточний вміст бази даних, рисунок 3.4.

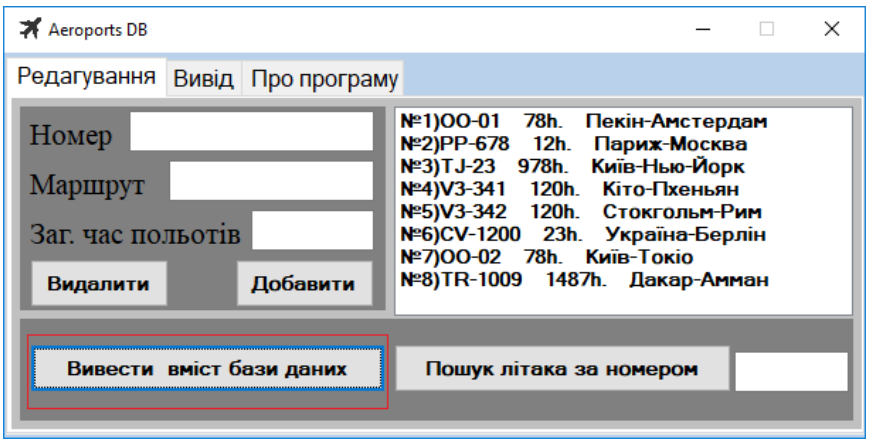


Рисунок 3.4 – Приклад виведення бази даних

Для того, щоб видалити запис із бази даних потрібно самостійно заповнити відповідні поля у правій частині вікна, рисунок 3.5, або обрати літак із списку (рис.3.4).

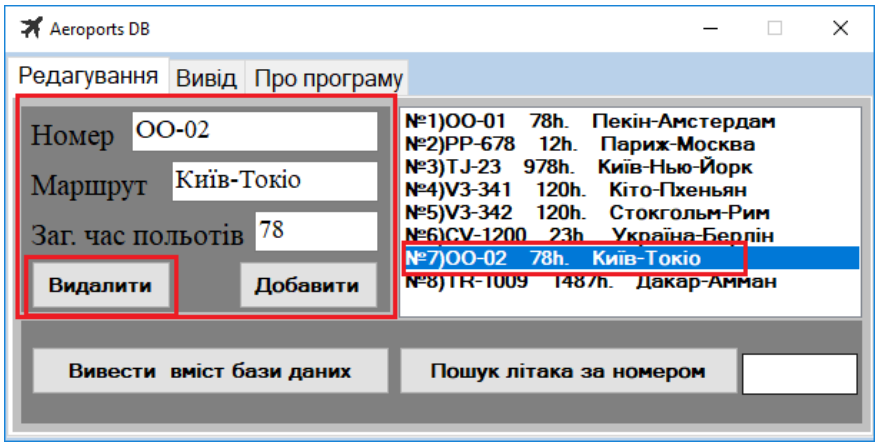


Рисунок 3.5 – Приклад видалення запису із бази даних

Для додання нового елементу слід заповнити усі поля необхідною інформацією, рисунок 3.6, та натиснути кнопку «Добавити», база даних буде оновлена.

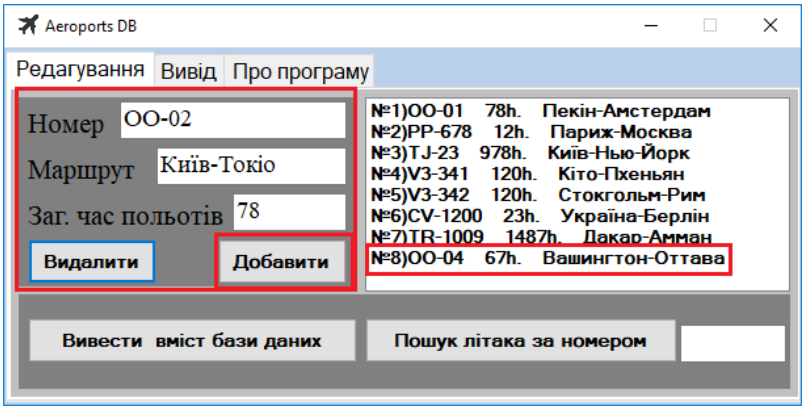


Рисунок 3.6 – Приклад додання запису в базу даних

Слід зазначити, що для додання або видалення запису із бази даних необхідно заповнити усі наявні поля: номер літака, маршрут, загальний час польотів. У базі даних не може одночасно зберігатися більше одного запису про літак, адже поле «Номер» є унікальним ключем. Це потрібно враховувати при додаванні запису.

У вікні «Вивід» користувач може виконати типові запити до бази даних, та зберегти отриманий результат у вигляді word-звіту. Для цього необхідно

заповнити відповідні поля, рисунок 3.7, та натиснути кнопку «Пошук». Таким чином буде сформований список рейсів, які будуть збережені.

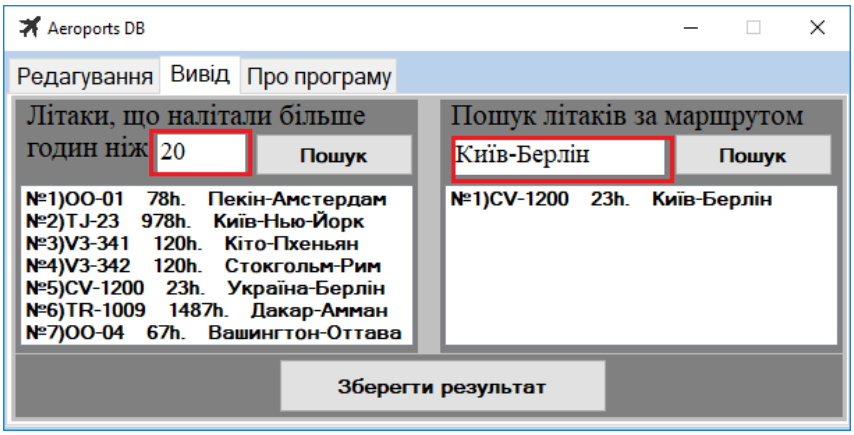


Рисунок 3.7 – Приклад формування звіту

Більш детальний опис функціоналу програми наведений в додатку Б, який представляє із себе інструкцію користувача.

3.2. Створення інсталяційного пакету

Створення інсталяційного пакету дозволить автоматично розгорнути програму на будь-якому робочому місці. Користувач не буде перейматися встановленням додаткових пакетів чи налаштувань.

Інсталятор був створений шляхом додавання додаткового проекту типу «Setup» у середовищі розробки MicrosoftVisualStudio 2019. Програмне забезпечення повинно поставлятися у вигляді файлу установки та інструкції по інсталяції, додаток В.

Після запуску інсталяційного пакету в обраній папці повинні з’явитися наступні файли, рисунок 3.8.

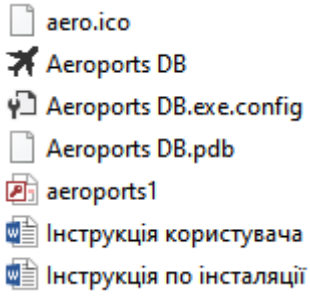


Рисунок 3.8 – Список встановлених компонентів програми

3.3. Тестування коду

При тестуванні програми були використані наступні методи:

- *Чорний ящик, ad-hoc*. Тип тестування, який здійснюється без перегляду програмного коду;
- *Функціональний тест*. Перевірка реалізації функціональних вимог у програмному продукті;
- *Юзабіліті-тестування*. Перевірка ергономічності розробленого програмного забезпечення;
- *Альфа-тестування.* Коротка перевірка розгорнутої на робочому місці системи;

Для початку слід виконати аналіз вимог. Тобто потрібно дослідити технічне завдання та уникнути суперечностей між функціями, які повинна виконувати програма. Таким чином, планується створити формалізоване бачення готового продукту, яке повинне складатися із опису дій, які буде виконувати програма і як саме. Це допоможе уникнути майбутніх непорозумінь.

Важливо звернути увагу на те, що будь-яка вимога впливає на тривалість і вартість проекту. Тому було введено умовні позначення, які визначають пріоритет:

- A - це обов’язкові умови, без них проект не можна вважати завершеним. Вони зобов'язані бути присутніми у кінцевому продукті.
- B - це бажані умови, вони, в деякому сенсі, важливі але не є обов’язковими для кінцевого продукту. Ними можна знехтувати, якщо їх реалізація потребує недопустимих витрат.
- C - це необов’язкові умови, проект без проблем може існувати без них. Такі вимоги слід реалізовувати по можливості.

Перш за все слід провести аналіз функціональних вимог, які визначають головні дії, які повинний виконувати готовий продукт. Їх перелік наведений в таблиці 3.1.

Таблиця 3.1 – Функціональні вимоги до системи

R-ID	Пріоритет	Вимога	TC-ID
R1.Функціональні вимоги			
R1.1	A	Можливість виконувати типові запити до бази даних	7
R1.2	C	Наявність можливості відкрити користувацьку інструкції з програми	14
R1.3	A	Можливість редагувати базу даних	2, 3
R1.4	B	Можливість переглядати поточний вміст бази даних	8
R1.5	A	Формувати звіт за результатами роботи типових запитів	4
R1.6	B	Пошук окремого літака за його номером	1

Наступним кроком виділимо нефункціональні вимоги, тобто ті, що описують характеристики системи і її оточення. Це те, як повинний працювати програмний продукт і які властивості мати. Список не функціональних вимог наведений в таблиці 3.2.

Таблиця 3.2 – Не функціональні вимоги до системи

R-ID	Пріоритет	Вимога	TC-ID
R2.Не функціональні вимоги			
R2.1	B	Розроблений продукт повинний містити логотип	5
R2.2	C	Програма повинна містити інформацію про автора	12
R2.3	A	Вміст бази даних або результати роботи з БД повинні відображатися в окремому вікні	8, 9, 10

R2.4	A	Програмний застосунок повинний використовувати лише одну мову	6
R2.5	A	Номер літака повинен бути унікальним ключем. У базі даних не може бути більше одного запису з ним	11, 13
R2.6	B	Для здійснення запису в базу даних повинні бути заповнені усі поля	15

Після описання вимог були сформовані тест кейси, таблиця 3.3, для тестування програмного продукту.

Таблиця 3.3 – Сценарії тестування вимог

TC-ID	Ім’я	Дії	Очікуваний результат	Статус
1	Пошук літака за його номером	1.Перейти до вкладки «Редагування» 2. У нижньому правому полі ввести номер літака 3. Натиснути кнопку «Пошук літака за номером»	Отримати запис про літак з таким номером	+
2	Редагування бази даних	1.Перейти до вкладки «Редагування» 2.Заповнити поля: номер, маршрут, загальний час польотів 3. Натиснути кнопку «добавити» або «видалити»	База даних повинна оновитися	+
3	Редагування бази даних	1.Перейти до вкладки «Редагування» 2. Натиснути кнопку «Вивести вміст бази даних» 3. Натиснути на один із виведених запитів 4. Натиснути кнопку «добавити» або «видалити»	База даних повинна оновитися	+
4	Функціонування типових запитів	1. Перейти до вкладки «Вивід» 2. Ввести кількість годин для запиту «Літаки, що налітали більше годин ніж» 3. Ввести маршрут для запиту «Пошук літаків за маршрутом» 4. Натиснути кнопку «Пошук» 5. Натиснути кнопку «Зберегти результат»	Повинний сформуватися word-звіт з результатами	+
5	Логотип	1. Переглянути усі вкладки програми	Програма повинна містити	+

			логотип	
6	Перевірка мови	1. Переглянути усі доступні вкладки	Програма повинна використовувати одну мову	+
7	Типові запити	1. Перейти до вкладки «Вивід» 2. Ввести кількість годин для запиту «Літаки, що налітали більше годин ніж» 3. Ввести маршрут для запиту «Пошук літаків за маршрутом» 4. Натиснути кнопку «Пошук»	Повинен з’явитися список з коректними результатами пошуку	+
8	Відображення бази даних	1.Перейти до вкладки «Редагування» 2. Натиснути кнопку «Вивести вміст бази даних»	Вміст бази даних повинен десь відобразитися	+
9	Відображення результату пошуку	1. Перейти до вкладки «Вивід» 2. Ввести кількість годин для запиту «Літаки, що налітали більше годин ніж» 3. Ввести маршрут для запиту «Пошук літаків за маршрутом» 4. Натиснути кнопку «Пошук»	Результат пошуку повинен десь відобразитися	+
10	Відображення результату пошуку	1.Перейти до вкладки «Редагування» 2. У нижньому правому полі ввести номер літака 3. Натиснути кнопку «Пошук літака за номером»	Результат пошуку повинен десь відобразитися	+
11	Перевірка унікальності номеру літака	1.Перейти до вкладки «Редагування» 2.Заповнити поля: маршрут, загальний час польотів. Номер літака взяти такий, що вже є в базі даних 3. Натиснути кнопку «добавити»	Повинно з’явитися повідомлення про некоректність ведених даних	+
12	Перегляд інформації про програму	1.Перейти до вкладки «Про програму»	У вкладці повинна знаходитися інформація про розробника або програму	+

13	Перевірка унікальності номеру літака	1.Перейти до вкладки «Редагування» 2. У нижньому правому полі ввести номер літака 3. Натиснути кнопку «Пошук літака за номером»	Повинно показати лише один результат	+
14	Перевірка користувацької інструкції	1.Перейти до вкладки «Про програму» 2.Натиснути кнопку «Інструкція користувача»	Повинна відкритися інструкція користувача	+
15	Заповнення усіх полів	1.Перейти до вкладки «Редагування» 2.Заповнити поля: номер, маршрут 3. Натиснути кнопку «добавити» або «видалити»	Повинно з'явитися повідомлення про некоректність ведених даних	+

Для того, щоб перевіряти коректність типових запитів автоматично, було створено юніт-тести, таблиця 3.4.

Таблиця 3.4 – Юніт-тести

№	Метод	Дії	Очікуємий результат	Статус
1	`search_hours(string hours)`	Запуск методу, для пошуку літаків, які налітали більше *200* годин.	Отримати строку: *"№4)YT-251 754h. Лусака-Пекін"*	Пройдено
2	`search_route(string route)`	Запуск методу, для пошуку літаків, які літають за маршрутом *"Стокгольм-Рим".*	Отримати строку: *"№1)V3-342 120h. Стокгольм-Рим"*	Пройдено
3	`search_number(string num)`	Запуск методу, для пошуку літака за його номером:OZ-013	Отримати строку: *"№1)OZ-013 150h. Ханой-Київ"*	Пройдено

Нижче наведений код розроблених юніт-тестів.

```
search_route ():
  [TestMethod()]
        [DeploymentItem("AeroportsDB.exe")]
        public void search_routeTest()
        {
            //act
            MainWindow_Accessor target = new MainWindow_Accessor();
            object sender = target;
            RoutedEventArgs e = null;
            target.Window_Loaded(sender, e);
```

```
            //arrange
            string route = "Стокгольм-Рим";
            string expected = "№1)V3-342    120h.    Стокгольм-Рим";

            //assert
            string actual = target.search_route(route);
            Assert.AreEqual(expected, actual);
    }

search_hours():
        [TestMethod()]
        [DeploymentItem("AeroportsDB.exe")]
        public void search_hoursTest()
        {
            //act
            MainWindow_Accessor target = new MainWindow_Accessor();
            object sender = target;
            RoutedEventArgs e = null;
            target.Window_Loaded(sender, e);

            //arrange
            string hours = "200";
            string expected = "№4)YT-251    754h.    Лусака-Пекін";

            //assert
            string actual = target.search_hours(hours);
            Assert.AreEqual(expected, actual);
     }

search_number():
          [TestMethod()]
        [DeploymentItem("AeroportsDB.exe")]
        public void search_numberTest()
        {
            MainWindow_Accessor target = new MainWindow_Accessor();
            object sender = target;
            RoutedEventArgs e = null;
            target.Window_Loaded(sender, e);

            //arrange
            string hours = "OZ-013";
            string expected = "№1)OZ-013    150h.    Ханой-Київ";

            //assert
            string actual = target.search_number(hours);
            Assert.AreEqual(expected, actual);
    }
```

3.4. Короткий опис програмної реалізації

На рисунку 3.9 можна побачити діаграму класів, які були використані при реалізації програмного додатку.

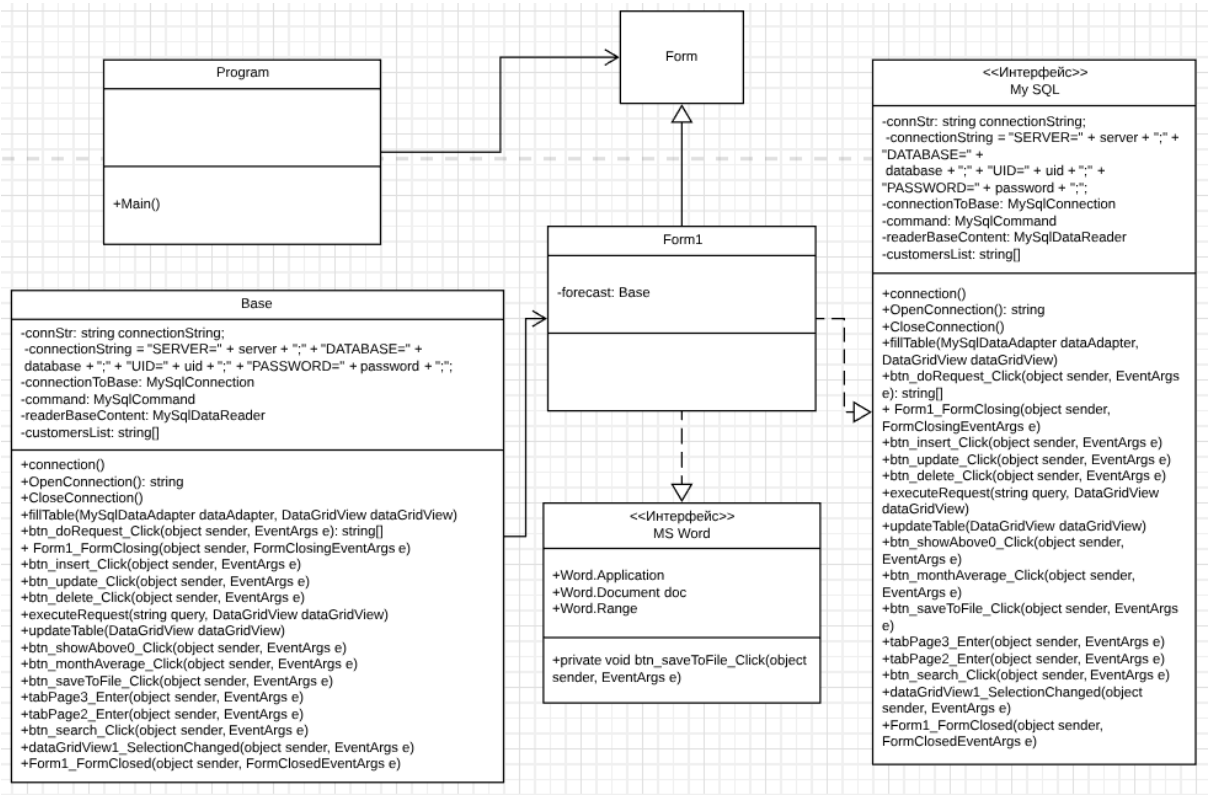


Рисунок 3.9 – Діаграма класів та інтерфейсів

Нижче наданий короткий опис процедур та функцій користувача:

public Form1() – ініціалізація компонентів форми, підключення до бази даних.

private void button2_Click(object sender, EventArgs e) – видалення інформації із бази даних.

private void button3_Click(object sender, EventArgs e) – виведення вмісту бази даних.

private void Form1_FormClosing(object sender, FormClosingEventArgs e) – закриття форми і розірвання підключення з базою даних.

private void listBox1_SelectedIndexChanged(object sender, EventArgs e) – обрання елементу із бази даних та занесення його у відповідні поля.

private void button1_Click(object sender, EventArgs e) – додання нового елементу в базу даних.

private void textBox3_KeyPress(object sender, KeyPressEventArgs e) – перевірка правильності вводу даних.

private void button4_Click(object sender, EventArgs e) – пошук літака за його номером

private void button5_Click(object sender, EventArgs e) – пошук літака, що пролітав більше ніж n годин.

private void textBox5_KeyPress(object sender, KeyPressEventArgs e) - перевірка правильності вводу даних.

private void button6_Click(object sender, EventArgs e) – пошук літаків за маршрутом.

private void button7_Click(object sender, EventArgs e) – збереження інформації в документ Word.

private void button8_Click(object sender, EventArgs e) – відкриття інструкції користувача.

У Додатку Г наведений більш детальний опис користувацьких функцій за допомогою діаграм подій [15, 16]. Повний код програми знаходиться в Додатку Д.

ВИСНОВКИ

На початку створення проекту були поставлені ряд конкретних задач, які є складовою технічного завдання, а саме:

1. Створення бази даних для внесення та можливості подальшого редагування даних про авіа-перельоти.

2. Створення прототипу програми.

3. Інтеграція бази даних у код.

4. Створення документу MSWord для виведення даних.

База даних була реалізовано локально, це було зроблено для того, щоб спростити інсталяцію програми.

Створення прототипу відбувалося за допомогою середовища MicrosoftVisualStudio 2019 з використанням мови програмування C#.

Word-звіт створюється автоматично після натискання кнопки «Зберегти результат». Для збереження обираються дані, які є результатами виконання типових запитів до БД.

По завершенню роботи над проектом було створено стабільно функціонуючу базу даних, яка не потребує наявності локального серверу на ПК. Програмний додаток виводить необхідні та коректні дані, які написані у вимогах, в документ. У користувача є можливість підключитися до своєї бази даних. Також для комфортної роботи із додатком наявна інструкція, у якій описані всі можливі дії.

СПИСОК ЛІТЕРАТУРИ

1. WorldMeter [Електронний ресурс] – https://www.worldometers.info /world-population/
2. Boeing 747-8 [Електронний ресурс] – https://www.boeing.com/ commercial/747/
3. Airbus A380 Aircraft Profile [Електронний ресурс] – https:// www.flightglobal.com/airbus-a380-aircraft-profile/66072.article
4. AirNav RadarBox [Електронний ресурс] – https://www.radarbox.com/ @21.47645,24.65467,z3
5. GPS: The Global Positioning System A global public service brought to you by the U.S. government [Електронний ресурс] – https://www.gps.gov/
6. ГЛОБАЛЬНАЯ НАВИГАЦИОННАЯ СПУТНИКОВАЯ СИСТЕМА ГЛОНАСС [Електронний ресурс] – https://www.glonass-iac.ru/guide/ gnss/glonass.php
7. Міжнародні авіалінії України [Електронний ресурс] – https:// www.flyuia.com/ua/ua/information/time-flight/arrivals-and-departures
8. Знакомство с нотацией IDEF0 и пример использования / Блог компании Trinion / Хабр [Електронний ресурс] – режим доступу: https://bit.ly/37xrnd5
9. Проектирование пользовательских интерфейсов: что это такое и зачем его заказывать — Блог Live Typing [Електронний ресурс] – режим доступу: https://bit.ly/2zwDHxR
10. How to Choose Your Brand Name in 5 Simple Steps [Електронний ресурс]– https://www.columnfivemedia.com/how-to-choose-a-brand-name
11. How to Make a Logo: Your Number One Guide [Електронний ресурс] – https://www.tailorbrands.com/logo-maker/how-to-make-a-logo?utm_source=google&utm_medium=cpc&utm_campaign=9250014894&utm_content=95502394282&utm_term=&gclid=Cj0KCQjw16KFBhCg

ARIsALB0g8Km6rDUKoJwbCIQuonAl_YRNwgmMz0jVvLFP_CSN7AmM6beJaBV-zYaAoInEALw_wcB

12. Преимущества Иерархической Структуры Работ (WBS) для менеджеров ИТ проектов / Хабр [Електронний ресурс] – режим доступу: https://bit.ly/2Y1eQM8
13. UML — диаграмма вариантов использования (use case diagram) / Хабр [Електронний ресурс] – режим доступу: https://bit.ly/30EizRc
14. Как диаграммы Ганта упрощают работу с проектами / Блог компании Hygger / Хабр [Електронний ресурс] – режим доступу: https://bit.ly/2XZIe5j
15. James Rumbaugh, Ivar Jacobson, Grady Booch (1999). *The unified modeling language reference manual* (англ.). Addison Wesley Longman Inc
16. Glossary of Key Terms at McGraw-hill.com. Retrieved 20 July 2008

ДОДАТОК А

План роботи буде складений у вигляді ієрархічної структури завдань проекту. Тобто необхідно розбити загальну задачу на окремі кроки, які потрібно виконати для її реалізації. При цьому будуть проігноровано розподіл ресурсів, часу та зв'язків між етапами проекту. Замість цього буде віддано акцент ініціалізації самих задач.

Тема бакалаврської роботи присвячена розробці програмного продукту для підтримки роботи міжнародного аеропорту. Для початку потрібно виділити основні етапи розробки:

- ***Етап ініціалізації проекту***. Збираємо і аналізуємо техніко-економічну інформацію про проект. Формуємо мету роботи і описуємо бажаний результат.
- ***Етап розроблення***. Створюємо і тестуємо програмний продукт, розробляємо користувацький інтерфейс.
- ***Етап реалізації***. Завершальна стадія розробки. Проводимо бета-тестування програми, виправляємо помилки в її роботі, формуємо звіт.

Тепер, коли етапи роботи були виокремленні, необхідно розбити їх на під задачі:

ЕТАП ІНІЦІАЛІЗАЦІЇ ПРОЕКТУ

1) ***Розробка концепції.*** Формуємо мету проекту, визначаємо його особливості.
   - **1.1)** ***Ідентифікація ідеї проекту.***
   - **1.2)** ***Деталізація мети проекту.***
   - **1.3)** ***Формування мети проекту.***
2) ***Техніко-економічне дослідження.*** Аналіз технічних, економічних, соціальних, фінансових аспектів проекту.
   - **2.1)** ***Дослідження економічної складової проект***у. Аналіз проекту з погляду його економічної вигоди та витрат.

2.1.1) ***Дослідження продукту.***

2.1.2) ***Дослідження організації.***

2.1.3) ***Дослідження ринку.***

2.1.4) ***Дослідження регіону.***

2.1.5) ***Дослідження фінансових аспектів проекту.***

2.1.6) ***Дослідження комерційних аспектів проекту.***

2.2) ***Дослідження проекту в соціальних аспектах.*** Аналіз проекту з точки зору соціальної вигоди.

2.2.1) ***Дослідження соціально-економічного аспекту.***

2.2.2) ***Дослідження соціально-інституційних аспектів.***

2.3) ***Оцінка IT проекту.*** Формування висновків щодо перспектив розроблюваного продукту.

2.3.1) ***Оцінка життєздатності.***

2.3.2) ***Оцінка цінності.***

2.3.3) ***Оцінка економічної ефективності.***

2.4) ***Завершення ТЕД. Формування звіту.***

3) ***Планування структури роботи.*** Завершальним завданням етапу ініціалізації є формування структури роботи. Тобто, розбиття проекту на етапи та задачі.

ЕТАП РОЗРОБЛЕННЯ ПРОЕКТУ.

1) ***Створення прототипу програми.*** На основі наявних даних побудувати робочий прототип програми, який би міг задовольнити базові вимоги.

1.1) ***Розроблення прототипу.***

1.2) ***Перевірка ефективності прототипу.***

1.3) ***Отримання оцінки від наукового керівника.***

1.4) ***Завершення створення прототипу.***

2) ***Створення користувацького інтерфейсу.*** На основі вимог користувачів побудувати зручний інтерфейс.

2.1) ***Розробка логотипу.***

2.2) ***Створення прототипу GUI.*** Створюємо прототип користувацького інтерфейсу. Отримуємо оцінку користувачів.

2.2.1) ***Отримання вимог до інтерфейсу.***

2.2.2) ***Розробка прототипу GUI.***

2.2.3) ***Тестування прототипу користувачами.***

2.3) ***Створення GUI.*** Розробка кінцевого варіанту GUI на основі прототипу і зауважень, які були отримані під час його тестування.

2.3.1) ***Отримання додаткових вимог після використання прототипу.***

2.3.2) ***Розробка GUI***.

2.3.3) ***Тестування GUI***.

2.4) ***Формування користувацької інструкції***. Опис того як використовувати програмний продукт.

3) ***Створення програмного додатку для підтримки роботи аеропорту***. Розробка кінцевого програмного забезпечення, яке здатне працювати з базою даних і автоматично формувати звіти.

3.1) ***Розробка системи.***

3.2) ***Об'єднання системи з користувацьким інтерфейсом.***

3.3) ***Тестування програми.***

4) ***Впровадження програми.*** Виконання необхідних передумов для використання програми на робочому місці диспетчера аеропорту.

ЕТАП РЕАЛІЗАЦІЇ ПРОЕКТУ.

1) ***Бета тестування програми.*** Тестування програми з персоналом університету. Отримання відгуків й пошук можливих помилок.

2) ***Виправлення помилок.*** Аналіз та виправлення багів,

3) ***Формування звітів.*** Формування кінцевої документації, яка пов'язана з програмним продуктом.

4) ***Крайній строк закінчення реалізації.*** Дата, коли проект і документація для нього повинна бути завершена.

ДОДАТОК Б

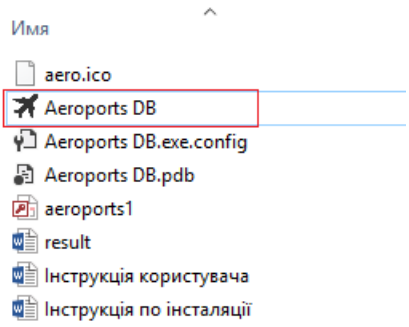


Рисунок 1

Для початку Вам слід запустити .exe-файл, рисунок 1.

Після цього Ви побачите наступне вікно, рисунок 2.

При натисканні на кнопку «Вивести вміст бази даних» Ви отримаєте поточний вміст вашої БД, рисунок 3.

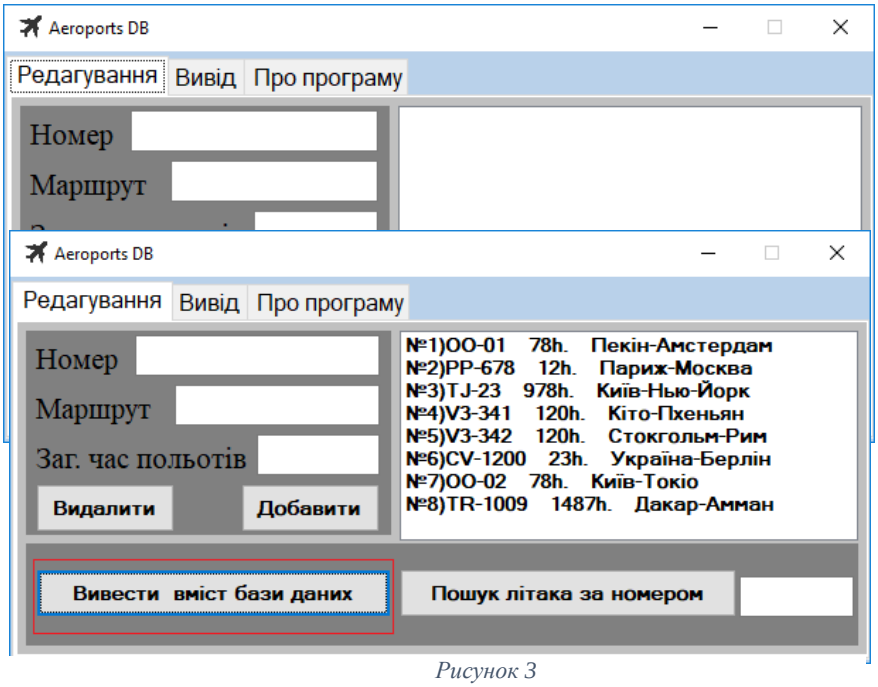
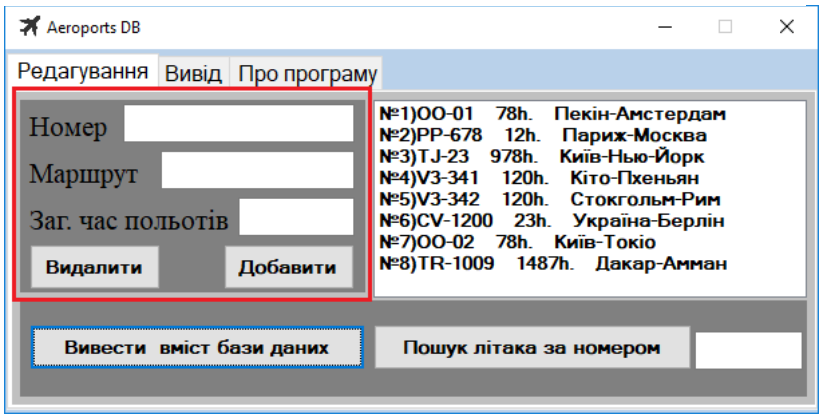


Рисунок 3

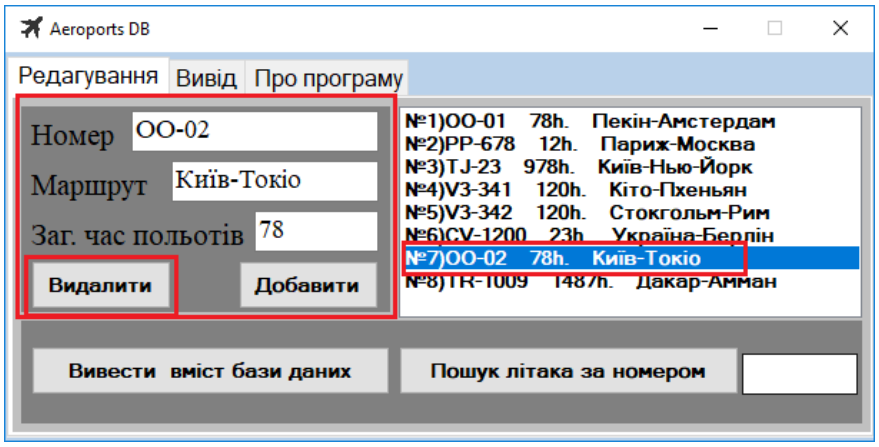
Для редагування вмісту бази даних слід використати панель, яка зображена на рисунку 4.

Рисунок 4



Для видалення інформації Ви можете самостійно заповнити панель, рисунок 4. Або обрати відповідний літак із списку, рисунок 5. Після натискання кнопки «Видалити» відповідна інформація зникне із БД.

Рисунок 5



Для додання нового елементу слід заповнити відповідні поля на панелі, рисунок 4, та натиснути кнопку «Добавити». Таким чином Ви оновите базу даних, рисунок 6.

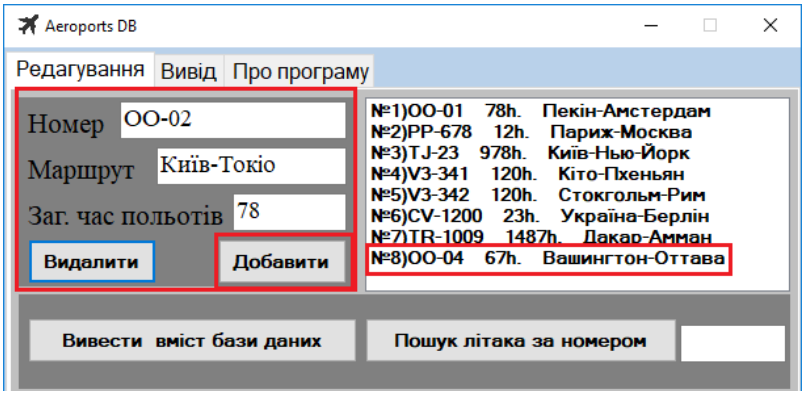
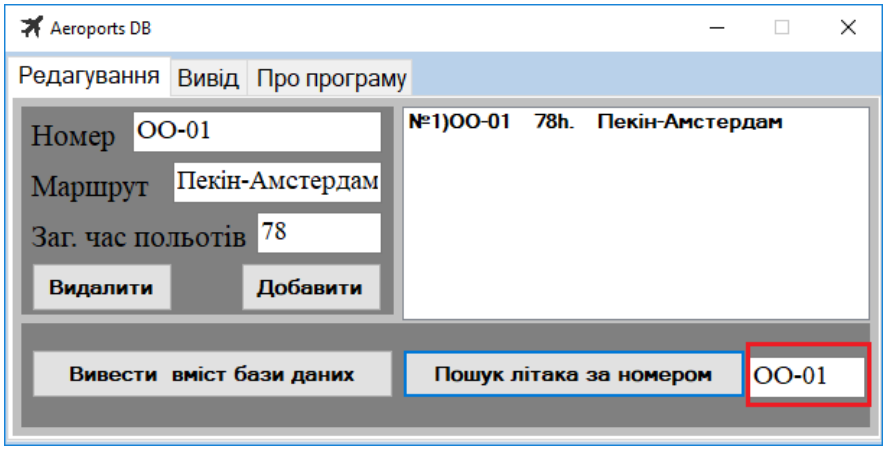


Рисунок 6

ВАЖЛИВО: для додання/видалення інформації слід заповнити всі поля на панелі, рисунок 4 а також зверніть увагу на номер літака, він не повинен повторюватися

Також у поданому вікні, рисунок 2, Ви можете знайти інформацію про конкретний літак. Для цього введіть номер шуканого транспорту у відповідне поле, рисунок 7, і натисніть кнопку «Пошук літака за номером».

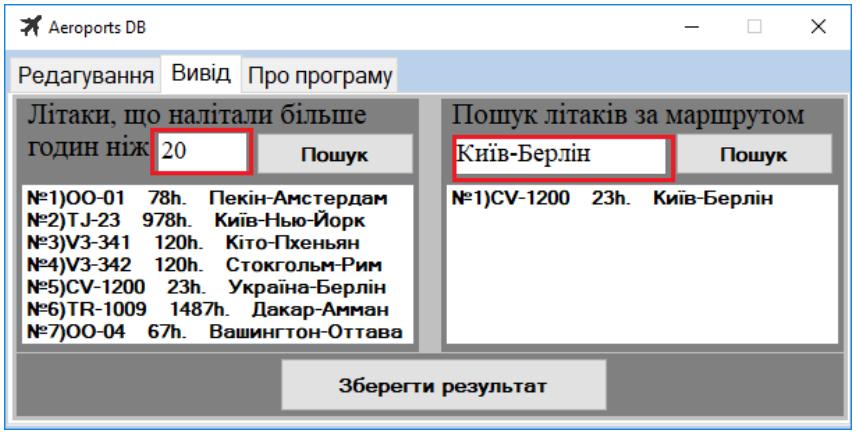
Рисунок 7



На цьому функціонал першого вікна («Редагування») закінчується. Переходимо до наступного.

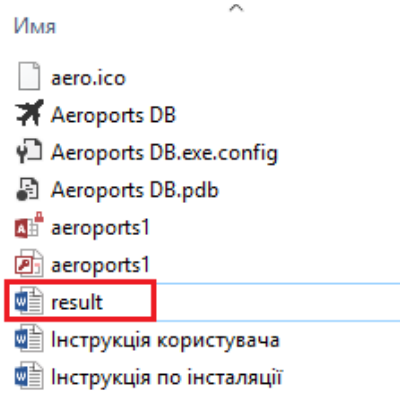
У вікні «Вивід» можна зберегти результати запитів. Для цього слід заповнити відповідні поля, рисунок 8, та натиснути кнопку «Пошук». Таким чином Ви сформуєте список із літаків, що відповідають запитам.

Рисунок 8



Наступним кроком натисніть кнопку «Зберегти результат», Ви створите Word файл з назвою *result.docx*. Цей файл буде збережений в папці з програмою та на робочому столі, рисунок 9.

Рисунок 9



Вміст збереженого файлу матиме наступний вигляд, рисунок 10.

Рисунок 10

Літаки, що налітали більше ніж 20 годин

№1)OO-01 78h. Пекін-Амстердам

№2)TJ-23 978h. Київ-Нью-Йорк

№3)V3-341 120h. Кіто-Пхеньян

№4)V3-342 120h. Стокгольм-Рим

№5)CV-1200 23h. Україна-Берлін

№6)TR-1009 1487h. Дакар-Амман

№7)OO-04 67h. Вашингтон-Оттава

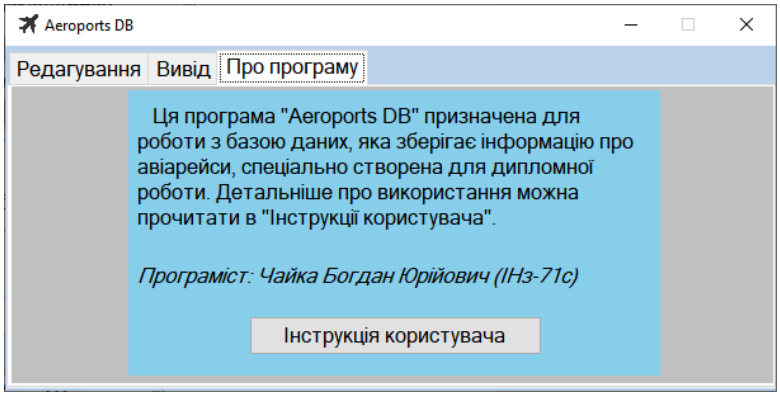
Літаки, що відповідають заданому маршруту "Київ-Берлін"

№1)CV-1200 23h. Київ-Берлін

На цьому функціонал другого вікна («Вивід») закінчується. Переходимо до наступного.

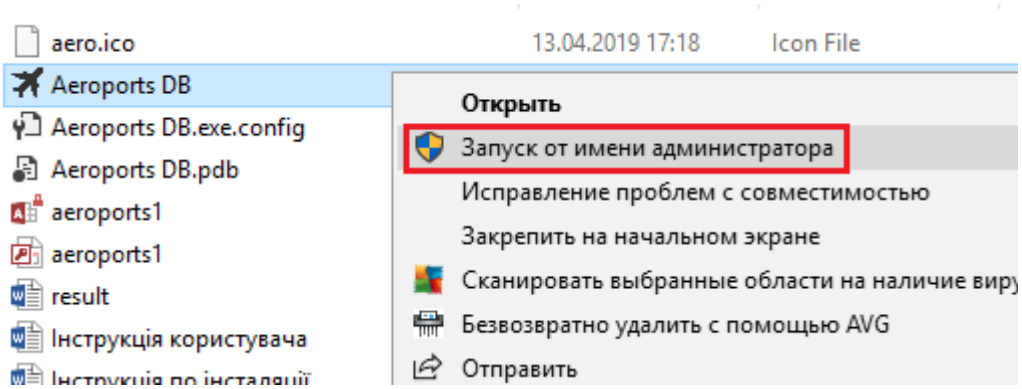
Останнє вікно «Про програму» містить загальну інформацію про цей програмний продукт, рисунок 11, а також кнопку «Інструкція користувача», при натисканні на яку Вам автоматично відкриється цей файл.

Рисунок 11



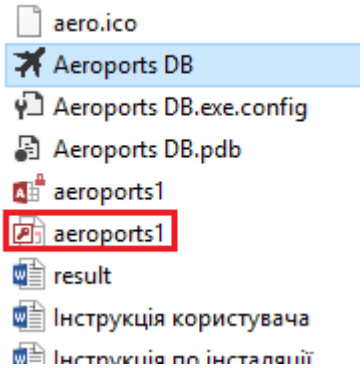
<u>ВАЖЛИВО</u>: у разі виникнення помилки під час додавання/видалення поля (Рис.5-6) спробуйте запустити програму «От имени администратора», рисунок 12.

Рисунок 12



<u>ВАЖЛИВО</u>: обов'язково перевірте чи створився, після інсталяції, файл «aeroports1.mdb», рисунок 13. <u>Не видаляйте його!</u>

Рисунок 12

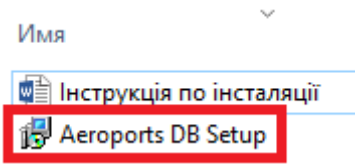


Команда розробників бажає Вам зручного та без проблемного використання програми «Aeroports DB»!

ДОДАТОК В

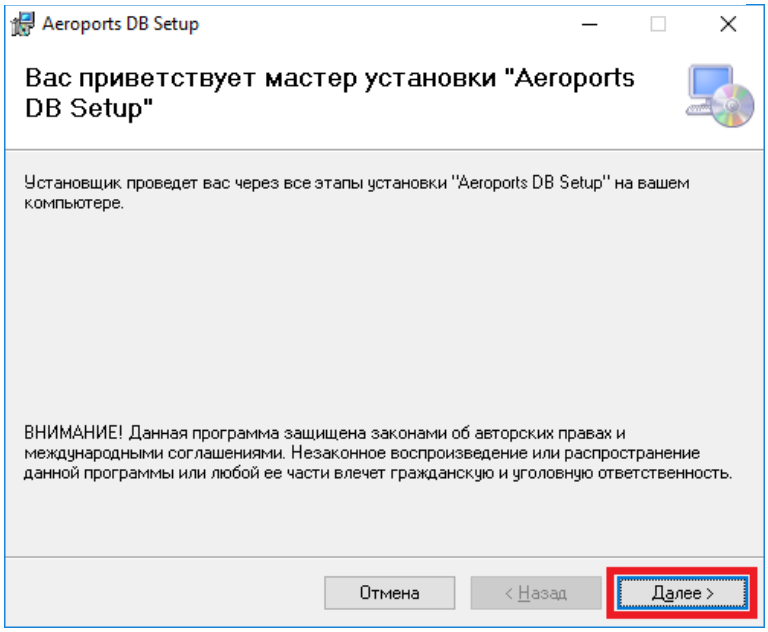
Перш ніж приступити до інсталяції Вам слід перевірити чи є на Вашому комп’ютері *MsWord* та *MsAccess,* зазвичай, вони встановлюються разом із пакетом *MsOffice.* Після цього запускайте інсталятор, рисунок 1.

Рисунок 13



Ви попадете до початкового вікна інсталятора, рисунок 2. Ознайомтесь з тим, що програма захищена законом про авторське право. Натисніть кнопку «Далее».

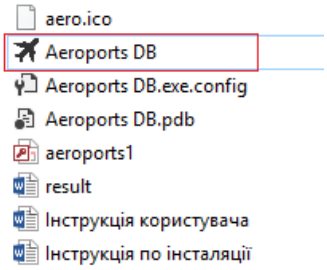
Рисунок 14



Наступним кроком оберіть шлях до місця куди буде встановлена програма. Для цього натисніть кнопку «Обзор…» та знайдіть необхідну папку, або пропишіть її шлях власноруч у відповідному полі. Також оберіть пункт «для всех» і натисніть «Далее».

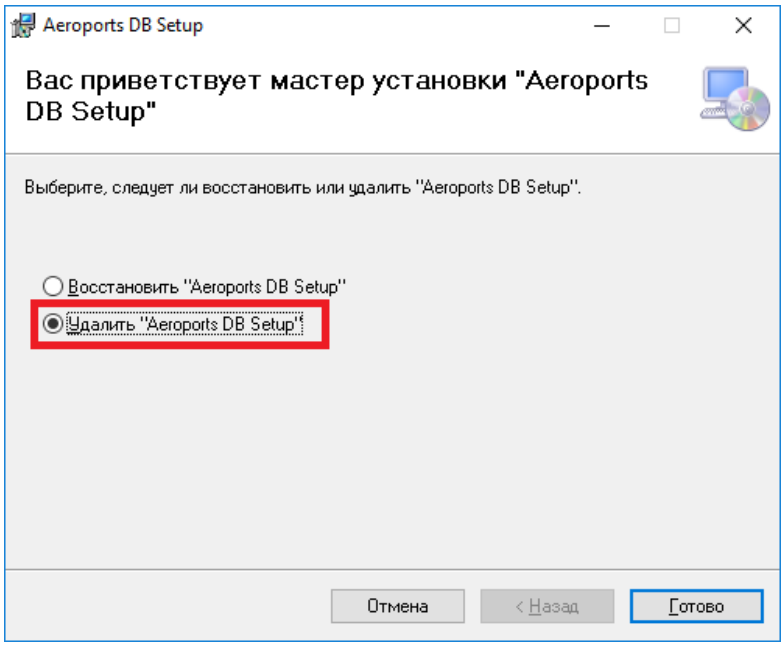
Після завершення процесу інсталяції в обраній папці з’являться наступні файли, рисунок 3.

Рисунок 3



Якщо Вам знадобиться видалити «Aeroports DB», то повторно запустіть інсталятор, рисунок 1. В з’явившомуся вікні, рисунок 4, оберіть необхідний пункт, натисніть «Готово».

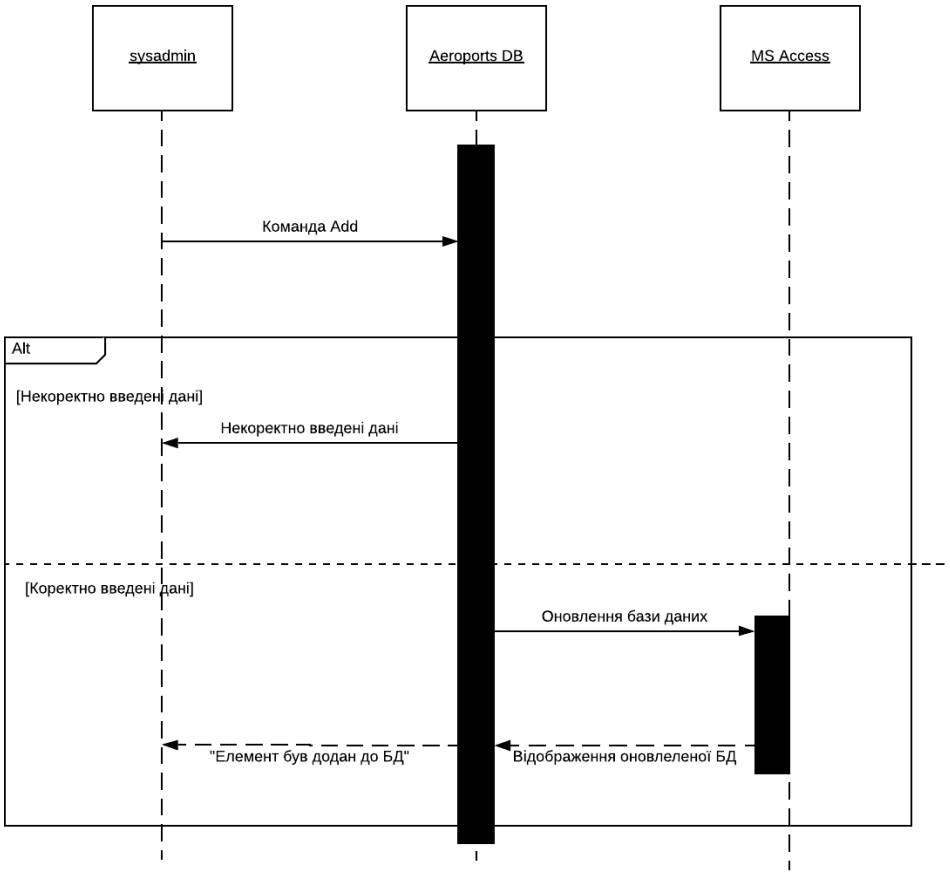
Рисунок 4

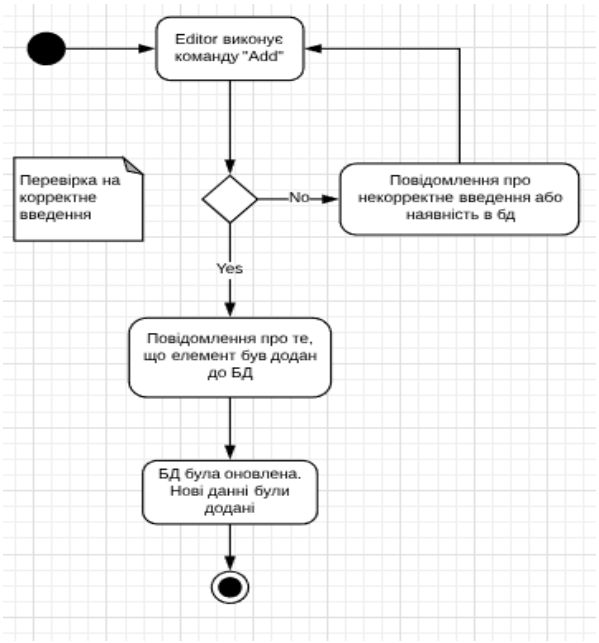


Приємного використання!

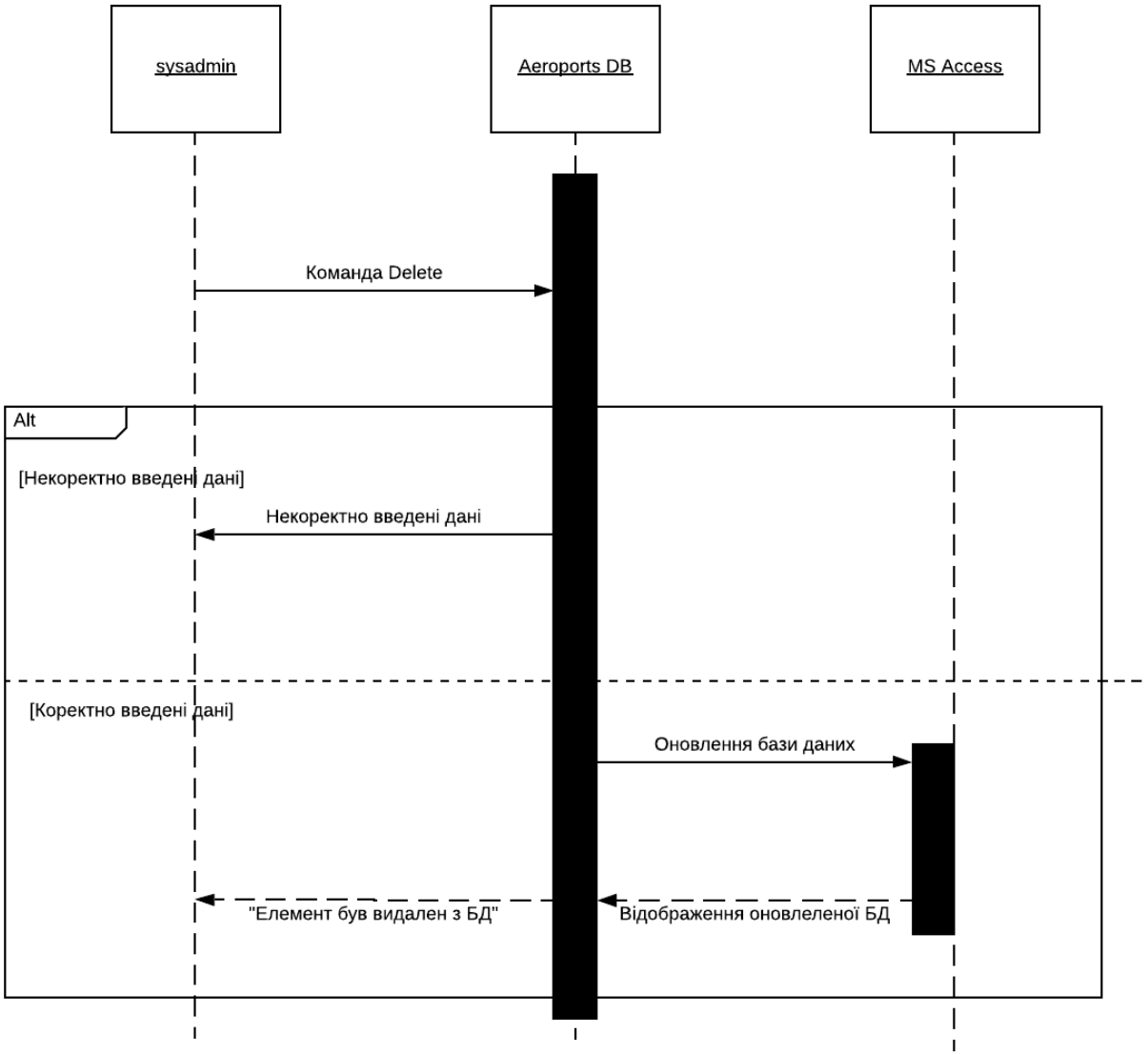
ДОДАТОК Г

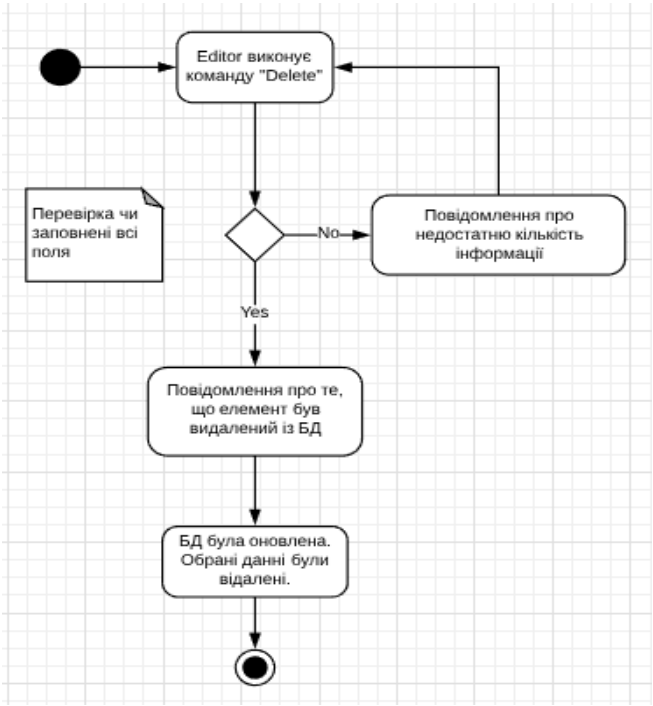
Команда add.



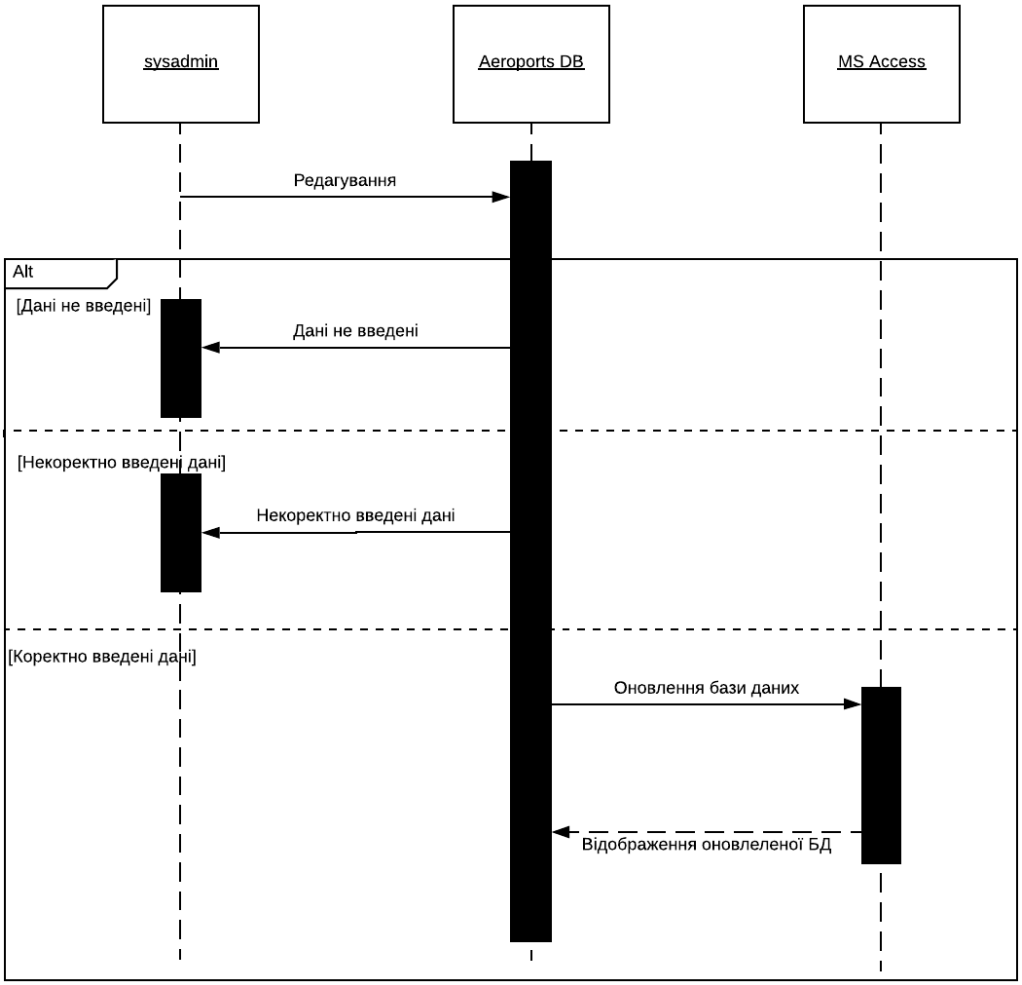


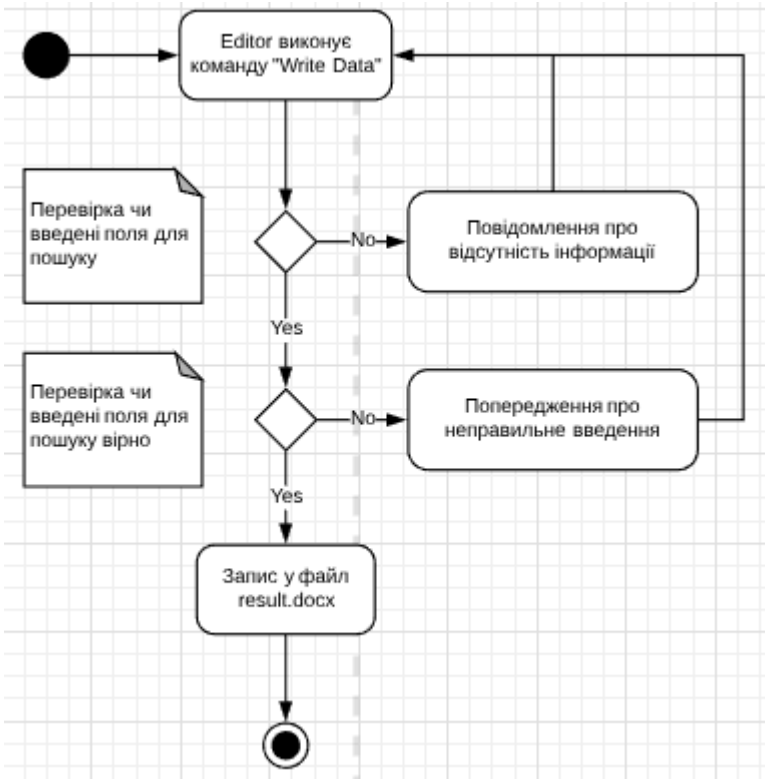
Команда Delete



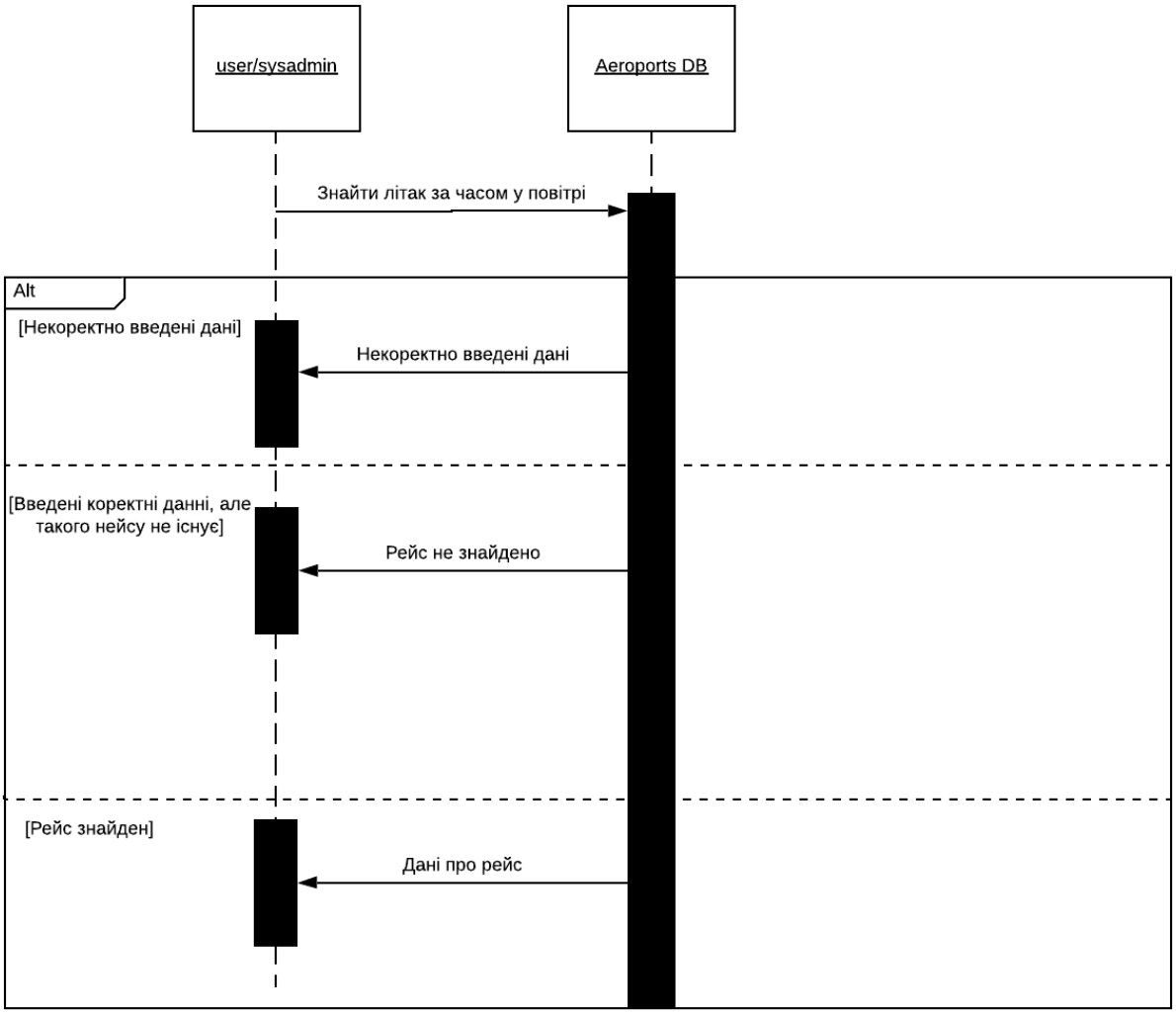


Команда Write Data

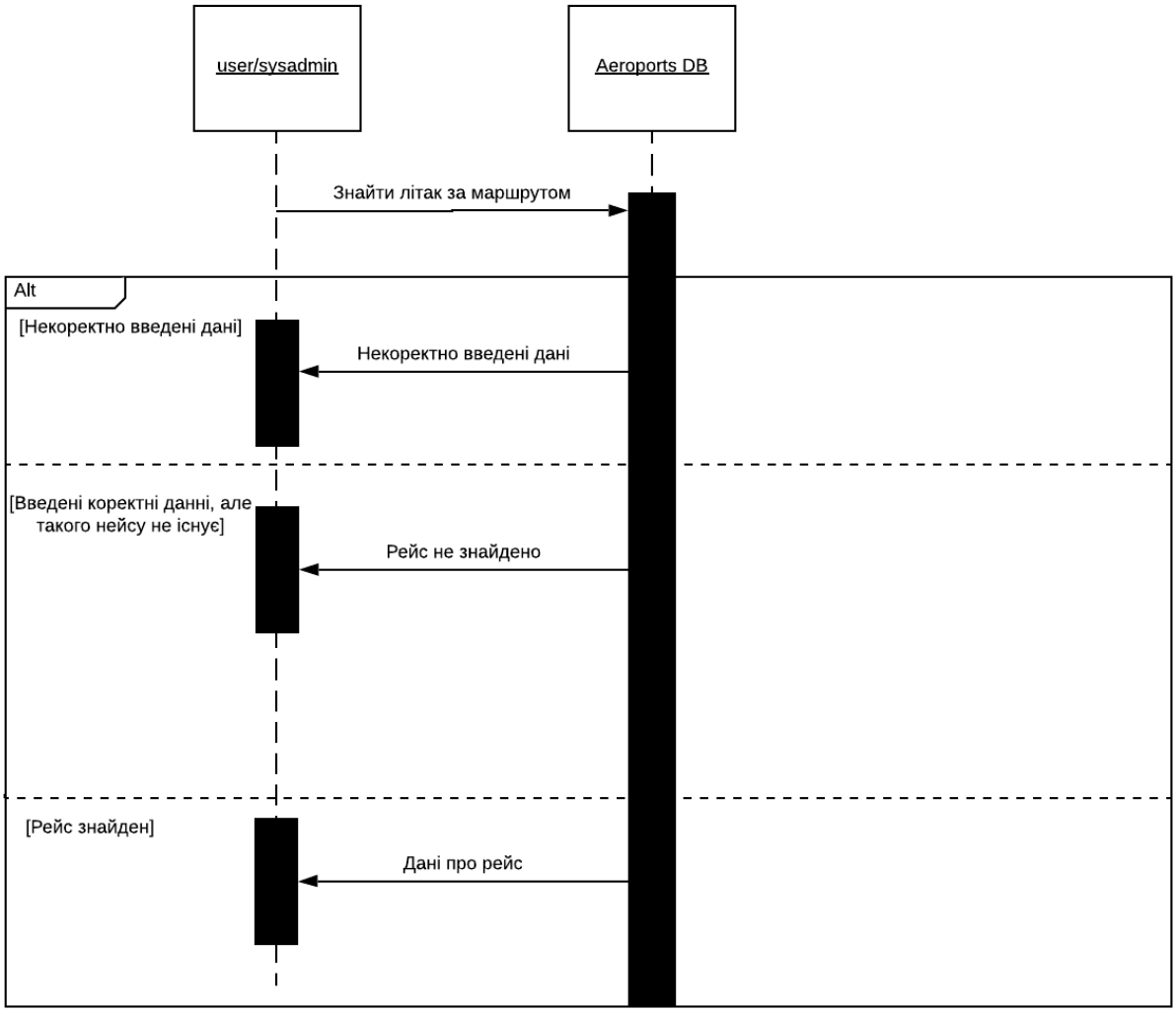


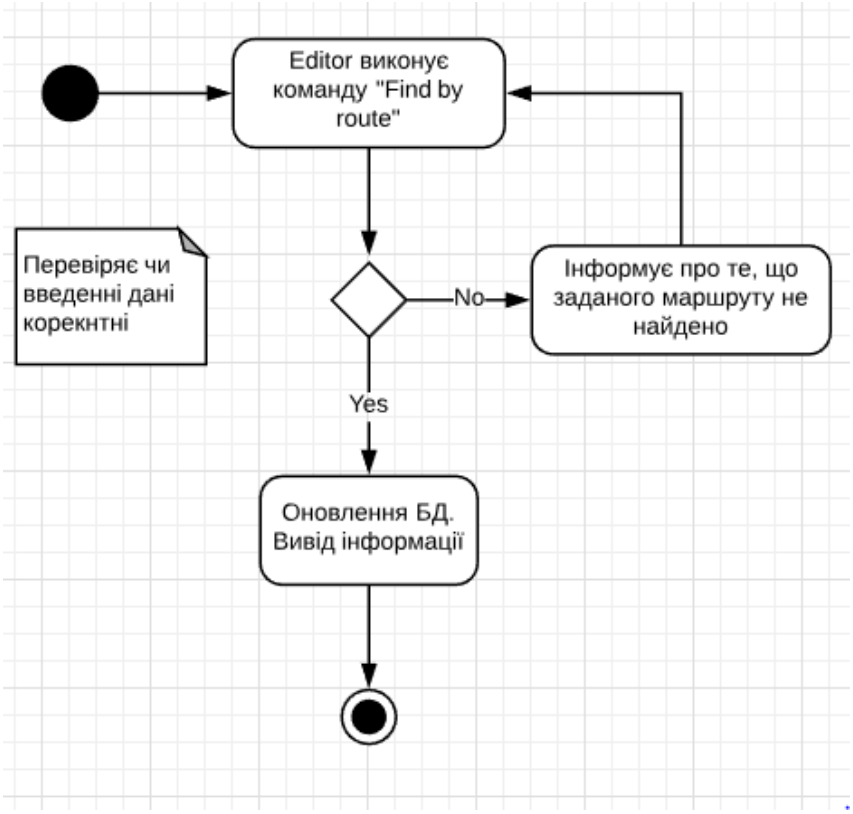


Команда Find by hours

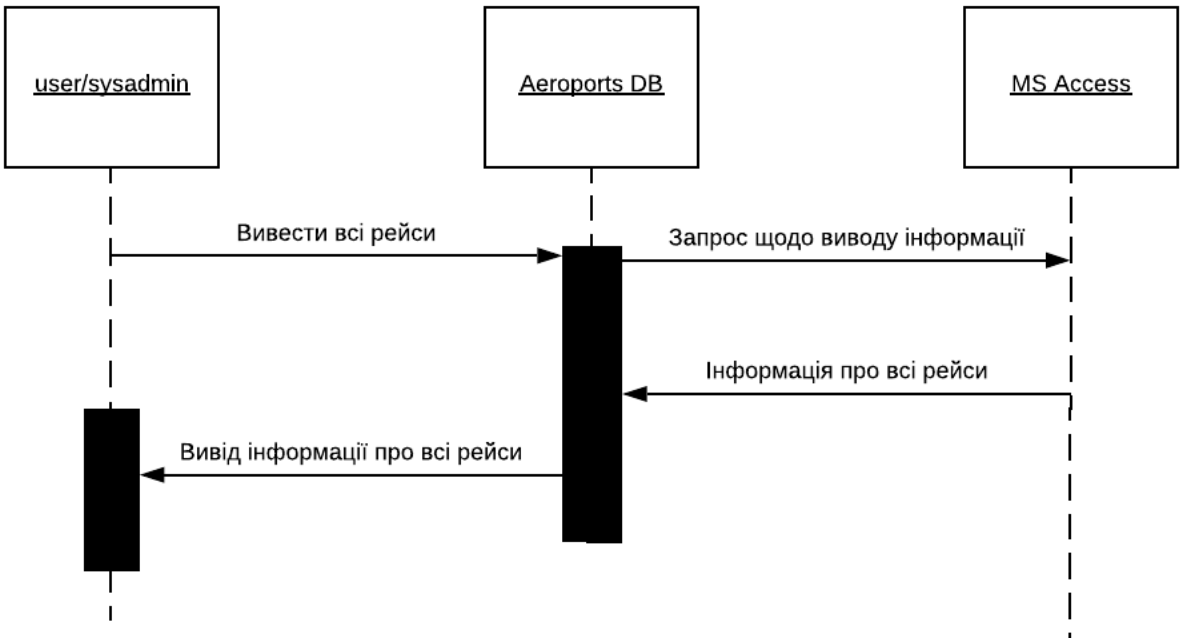


Команда Find by route

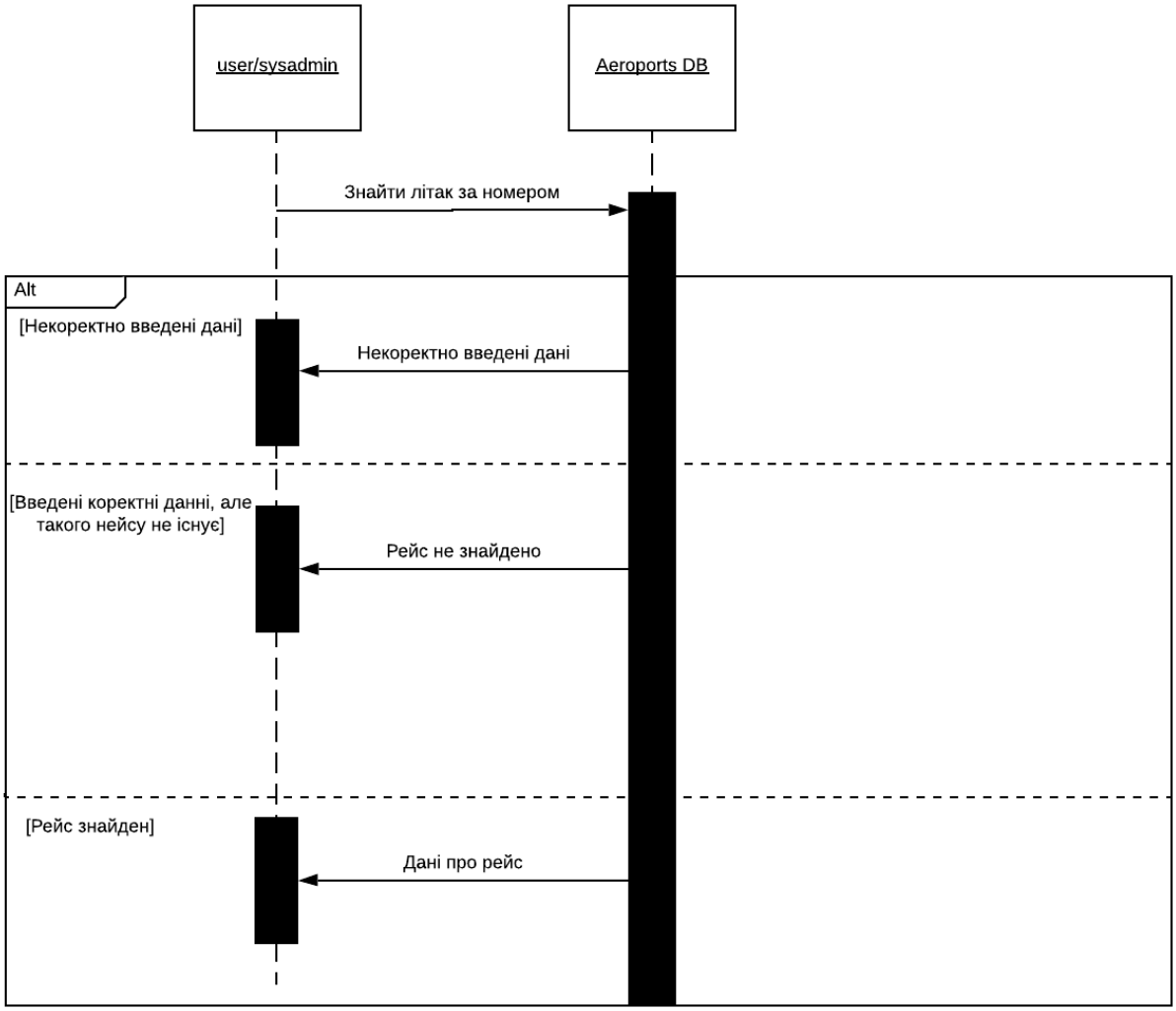


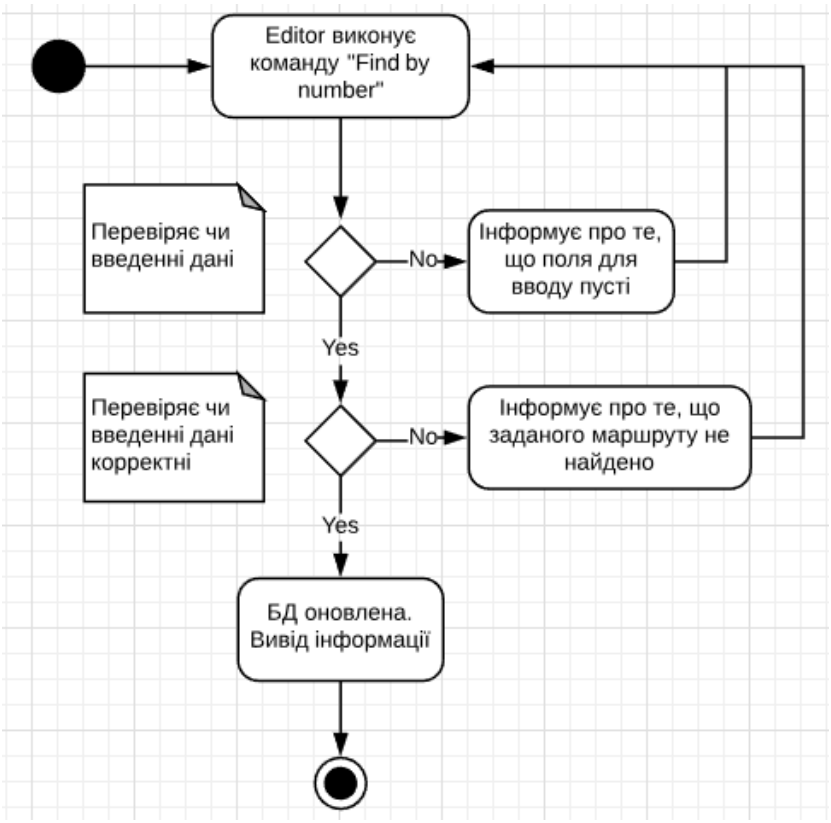


Команда Show all



Команда Find by number





ДОДАТОК Д

```
using System;
using System.Windows.Forms;
using System.Data.OleDb;
using Word = Microsoft.Office.Interop.Word;

namespace AeroportsDB
{
    public partial class Form1 : Form
    {

        public static string connectString =
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=aeroports1.mdb;";
        private OleDbConnection myConnection;
        public Form1()
        {
            InitializeComponent();

            myConnection = new OleDbConnection(connectString);
            myConnection.Open();
            label4.Text = "Літаки, що налітали більше\nгодин ніж";
            label5.Text = "Пошук літаків за маршрутом";

            FormBorderStyle = FormBorderStyle.Fixed3D;
            MaximizeBox = false;
        }

        private void button2_Click(object sender, EventArgs e)
        {

            if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text ==
"")
            {
                MessageBox.Show("Ви ввели недостатню кількість інформації про
літак", "Попередження", MessageBoxButtons.OK,MessageBoxIcon.Warning);
                return;
            }

            // текст запроса
            string query = "DELETE FROM aeroport WHERE nomer = " + '\'' +
textBox1.Text + '\'' + " and hours = " + textBox3.Text + " and route = " + '\''
+ textBox2.Text + '\'';
            try
            {
                // создаем объект OleDbCommand для выполнения запроса к БД MS
Access
                OleDbCommand command = new OleDbCommand(query, myConnection);

                // выполняем запрос к MS Access
                command.ExecuteNonQuery();
            }
            catch {
```

```
                MessageBox.Show("При видаленні поля виникла помилка. Спробуйте
запустити програму \"От имени администратора\"", "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            MessageBox.Show($"Елемент був видалений:\nНомер:
{textBox1.Text}\n3агальний час польотів: {textBox3.Text}\nМаршрут:
{textBox2.Text}",

"Повідомлення",MessageBoxButtons.OK,MessageBoxIcon.Information);
            button3_Click(sender, e);


        }

        private void button3_Click(object sender, EventArgs e)
        {
            // текст запроса
            string query = "SELECT nomer, hours , route FROM aeroport WHERE
hours > -1";
            // создаем объект OleDbCommand для выполнения запроса к БД MS
Access
            OleDbCommand command = new OleDbCommand(query, myConnection);
            // получаем объект OleDbDataReader для чтения табличного результата
запроса SELECT
            OleDbDataReader reader = command.ExecuteReader();
            // очищаем listBox1
            listBox1.Items.Clear();
            // в цикле построчно читаем ответ от БД
            int i = 0;
            while (reader.Read())
            {
                i++;
                // выводим данные столбцов текущей строки в listBox1
                listBox1.Items.Add($"№{i})" + reader[0].ToString() +
                    "    " + reader[1].ToString() + "h.    " +
reader[2].ToString());
            }
            // закрываем OleDbDataReader
            reader.Close();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            myConnection.Close();
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            string str = (string)listBox1.SelectedItem;

            if (str == null) return;

            string nomer = "";
            string hours = "";
```

```
            string way = "";

            bool num = false, time = false, route = false;

            for (int i = (str.IndexOf(')') + 1); i < str.Length; i++) {
                if (!num && !time && !route) {
                    nomer += str[i];
                    if (str[i + 1] == ' ')
                    {
                        num = true;
                        continue;
                    }
                }

                if (num && !time && !route && str[i] != ' ')
                {
                    hours += str[i];
                    if (str[i + 1] == 'h')
                    {
                        time = true;
                        continue;
                    }
                }

                if (str[i] != ' ' && num && time && str[i] != 'h' && str[i] !=
'.')
                    route = true;
                if (route) way += str[i];
            }

            textBox1.Text = nomer;
            textBox2.Text = way;
            textBox3.Text = hours;
        }

        private void button1_Click(object sender, EventArgs e)
        {

            string query = "INSERT INTO aeroport (nomer, hours, route) " +
                "VALUES ('" + textBox1.Text + '\'' + ", " + textBox3.Text + ","
+
                " \'" + textBox2.Text + "\')";
            // создаем объект OleDbCommand для выполнения запроса к БД MS
Access
            OleDbCommand command = new OleDbCommand(query, myConnection);

            // выполняем запрос к MS Access
            try
            {
                command.ExecuteNonQuery();
            }
            catch {
                MessageBox.Show("Ви ввели інформацію некоректно або літак вже є
в базі даних",
                    "Попередження", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
```

```
                return;
            }
            MessageBox.Show("Елемент доданий до бази даних", "Повідомлення",
                MessageBoxButtons.OK, MessageBoxIcon.Question);

            button3_Click(sender, e);
        }

        private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
        {
            char ch = e.KeyChar;
            if (!Char.IsDigit(ch) && e.KeyChar != (char)Keys.Back)
            {
                MessageBox.Show("Ви можете ввести лише цілі і невід'ємні
цифри",
                    "Попередження", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                e.Handled = true;
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            // текст запроса
            string route = textBox4.Text;
            if (route == "") {
                MessageBox.Show("Введіть шуканий номер літака",
                    "Повідомлення", MessageBoxButtons.OK,
MessageBoxIcon.Question);
                textBox4.Focus();
                return;
            }

            string query = "SELECT nomer, hours , route FROM aeroport WHERE
nomer = "
                + '\'' + textBox4.Text + '\'';

            // создаем объект OleDbCommand для выполнения запроса к БД MS
Access
            OleDbCommand command = new OleDbCommand(query, myConnection);

            // получаем объект OleDbDataReader для чтения табличного результата
запроса SELECT
            OleDbDataReader reader = command.ExecuteReader();

            // очищаем listBox1
            listBox1.Items.Clear();

            // в цикле построчно читаем ответ от БД
            int i = 0;
            while (reader.Read())
            {
                i++;
                // выводим данные столбцов текущей строки в listBox1
                listBox1.Items.Add($"№{i})" + reader[0].ToString() + "    "
                    + reader[1].ToString() + "h.    " + reader[2].ToString());
```

```
            }
            if(listBox1.Items.Count == 0) MessageBox.Show("Інформацію про
заданий маршрут не вдалося знайти ",
                "Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Question);
            // закрываем OleDbDataReader
            reader.Close();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            // текст запроса
            string route = textBox5.Text;
            if (route == "")
            {
                MessageBox.Show("Введіть час, який вас цікавить",
"Повідомлення",
                    MessageBoxButtons.OK, MessageBoxIcon.Question);
                textBox5.Focus();
                return;
            }

            string query = "SELECT nomer, hours , route FROM aeroport WHERE
hours > " + textBox5.Text;

            // создаем объект OleDbCommand для выполнения запроса к БД MS
Access
            OleDbCommand command = new OleDbCommand(query, myConnection);

            // получаем объект OleDbDataReader для чтения табличного результата
запроса SELECT
            OleDbDataReader reader = command.ExecuteReader();

            // очищаем listBox1
            listBox2.Items.Clear();

            // в цикле построчно читаем ответ от БД
            int i = 0;
            while (reader.Read())
            {
                i++;
                // выводим данные столбцов текущей строки в listBox1
                listBox2.Items.Add($"№{i})" + reader[0].ToString() + "    " +
reader[1].ToString()
                    + "h.    " + reader[2].ToString());
            }
            if (listBox2.Items.Count == 0) MessageBox.Show("Інформацію про
заданий час не найдено",
                "Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Question);
            // закрываем OleDbDataReader
            reader.Close();
        }

        private void textBox5_KeyPress(object sender, KeyPressEventArgs e)
        {
            textBox3_KeyPress(sender, e);
        }
```

```
        private void button6_Click(object sender, EventArgs e)
        {
            // текст запроса
            string route = textBox6.Text;
            if (route == "")
            {
                MessageBox.Show("Введіть маршрут, який вас цікавить",
                    "Повідомлення", MessageBoxButtons.OK,
MessageBoxIcon.Question);
                textBox6.Focus();
                return;
            }

            string query = "SELECT nomer, hours , route FROM aeroport WHERE
route = "
                + '\'' + textBox6.Text + '\'';

            // создаем объект OleDbCommand для выполнения запроса к БД MS
Access
            OleDbCommand command = new OleDbCommand(query, myConnection);

            // получаем объект OleDbDataReader для чтения табличного результата
запроса SELECT
            OleDbDataReader reader = command.ExecuteReader();

            // очищаем listBox1
            listBox3.Items.Clear();

            // в цикле построчно читаем ответ от БД
            int i = 0;
            while (reader.Read())
            {
                i++;
                // выводим данные столбцов текущей строки в listBox1
                listBox3.Items.Add($"№{i})" + reader[0].ToString() + "    " +
reader[1].ToString()
                    + "h.    " + reader[2].ToString());
            }
            if (listBox3.Items.Count == 0) MessageBox.Show("Інформацію про
заданий маршрут не найдено",
                "Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Question);
            // закрываем OleDbDataReader
            reader.Close();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            if (listBox2.Items.Count == 0 && listBox3.Items.Count == 0) {
                MessageBox.Show("Інформація, яку слід зберегти відсутня",
                    "Попередження", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return;
            }

            var wordApp = new Word.Application();
```

```
            var document = new Word.Document();
            document = wordApp.Documents.Add();
            wordApp.Visible = true;
            string str = "";
            //Загаловки
            var r = document.Range();
            //Текст
            var r1 = document.Range();
            if (listBox2.Items.Count != 0)
            {
                //Загаловок
                document.Paragraphs[1].Range.Font.Name = "Times New Roman";
                document.Paragraphs[1].Range.Bold = 2;
                r = document.Paragraphs[1].Range;
                str += $"\t\t\t\tЛітаки, що налітали більше ніж {textBox5.Text}
годин \n";

                //listbox2
                for (int i = 0; i < listBox2.Items.Count; i++)
                {
                    str += listBox2.Items[i].ToString() + '\n';

                }
            }

            if (listBox3.Items.Count != 0)
            {
                //Загаловок 2
                str += $"\t\t\t\tЛітаки, що відповідають заданому маршруту
\"{textBox6.Text}\"\n";

                //listbox3
                for (int i = 0; i < listBox3.Items.Count; i++)
                {
                    str += listBox3.Items[i].ToString() + '\n';

                }
                r.Text = str;
            }

            string filePath =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
            filePath += "\\result.docx";

            try {
                document.SaveAs(filePath);
            }
            catch
            {
                document.SaveAs(@Application.StartupPath.ToString() +
"\\result.docx");
            }
            finally
            {
                document.SaveAs(@Application.StartupPath.ToString() +
"\\result.docx");
```

```
            }

            wordApp.Visible = true;
        }

        private void button8_Click(object sender, EventArgs e)
        {

System.Diagnostics.Process.Start(@Application.StartupPath.ToString()
                + "\\Інструкція користувача.docx");
        }
    }
}
```