**Форма № 17**

Міністерство освіти і науки України

**Національний університет «Запорізька політехніка»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кафедра програмних засобів

(найменування кафедри)

**КУРСОВИЙ ПРОЄКТ**

**(РОБОТА)**

з дисципліни «Об’єктно-орієнтоване програмування»

(назва дисципліни)

на тему: Розробка програмного забезпечення моделювання структур графів

з використанням ООП

Студента 1 курсу КНТ-113сп групи

спеціальності 121 Інженерія   
програмного забезпечення

освітня програма (спеціалізація) інженерія програмного забезпечення

Щедровського І. А.

(прізвище та ініціали)

Керівник асистент, Короткий О. В.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   Короткий О. В.

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   Каплієнко Т.І.

(підпис) (прізвище та ініціали)

2022 рік

**Форма № 25**

Міністерство освіти і науки України

**Національний університет «Запорізька політехніка»**

( повне найменування закладу вищої освіти)

Інститут, факультет Запорізький національний університет НУ «Запорізька політехніка». Факультет комп`ютерних наук і технологій ;

Кафедра програмних засобів\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ступінь вищої освіти      бакалавр**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Спеціальність 121 Інженерія програмного забезпечення**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення**\_\_\_\_\_\_\_\_**

(назва освітньої програми (спеціалізації)

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри ПЗ, д.т.н, проф.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.О. Субботін**

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 року

З А В Д А Н Н Я

**НА КУРСОВИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА**

Щедровського Івана Андрійовича

(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Розробка програмного забезпечення моделювання структур графів з використанням ООП

керівник проєкту (роботи)\_Короткий Олександр Володимирович, асистент,

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від

2. Строк подання студентом проєкту (роботи)\_21 грудня 2022 року\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до проєкту (роботи) розробити додаток згідно теми курсової роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Огляд та аналіз сучасних методів та засобів проектування програмного забезпечення; 2. Проєктування програмного забезпечення системи; 3. Розробка програмного забезпечення системи; 4.Аналіз ефективності програмного забезпечення; 5. Розробка документів на супроводження програмного забезпечення.   
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)\_\_\_Слайди презентації

6. Консультанти розділів проєкту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада  консультанта | Підпис, дата | |
| завдання видав | прийняв  виконане завдання |
| 1-5 Основна частина | Короткий О.В., асистент |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання 06 листопада 2022 р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів курсового  проєкту (роботи) | Строк виконання етапів проєкту  ( роботи ) | Примітка |
| 1. | Аналіз індивідуального завдання. | 1 тиждень |  |
| 2. | Аналіз програмних засобів, що будуть використовуватись в роботі. | 2 тиждень |  |
| 3. | Аналіз структур даних, що необхідно використати в курсової роботі. | 3 тиждень |  |
| 4. | Затвердження завдання | 4 тиждень |  |
| 5. | Вивчення можливостей програмної реалізації структур даних та інтерфейсу користувача. | 5-9 тиждень |  |
| 6. | Аналіз вимог до апаратних засобів | 9 тиждень |  |
| 7. | Розробка програмного забезпечення | 9-13 тиждень |  |
| 8. | Проміжний контроль | 10 тиждень | Розділи 1-5 ПЗ |
| 9. | Оформлення, відповідних пунктів пояснювальної записки. | 10-14 тиждень | Розділи 1-2 ПЗ |
| 10. | Захист курсової роботи. | 15 тиждень |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_**Щедровський І.А.\_

( підпис ) (прізвище та ініціали)

**Керівник проєкту (роботи) \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_** Короткий О.В.**\_\_\_\_**

( підпис ) (прізвище та ініціали)

**Реферат**

Пояснювальна записка до курсової роботи містить 107 сторінок, 31 рисунок, 0 таблиць, 2 додатки, 8 джерел.

Пояснювальна записка складається з шести розділів.

Розділ «Огляд та аналіз сучасних методів та засобів проектування програмного забезпечення» містить в собі аналіз сучасних методів до проектування програмного забезпечення та аналіз різних сфер життя.

Розділ «Проектування програмного забезпечення системи» містить аналіз функцій системи, розробку UML діаграм використання, проєктування графічного інтерфейсу та постановку мети.

Розділ «Розробка програмного забезпечення системи» містить розробку структури програми, розробку UML діаграми класів та опис класів програмного забезпечення.

Розділ «Аналіз ефективності програмного забезпечення » містить аналіз застосунку на швидкодія та масштабованість, аналіз ефективності кожного компоненту системи, а також тестування всього програмного забезпечення.

Розділ «Розробка документів на супроводження програмного забезпечення» містить інструкцію для програміста, де є розгортка застосунку на локальній машині та на сервері, а також інструкцію користувача.

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, ГРАФИ, TYPESCRIPT, VITE, GIT, АЛГОРИТМИ, ОПТИМІЗАЦІЯ ВІЗУАЛІЗАЦІЇ ГРАФІВ

**ЗМІСТ**

[Вступ 6](#_Toc157707856)

[1 Огляд та аналіз сучасних методів та засобів проектування програмного забезпечення 7](#_Toc157707857)

[1.1 Огляд сучасного стану питання 7](#_Toc157707858)

[1.2 Основні шляхи вирішення поставленої задачі 7](#_Toc157707859)

[1.3 Формулювання мети досліджень 7](#_Toc157707860)

[2 Проектування програмного забезпечення системи 8](#_Toc157707861)

[2.1 Постановка мети 8](#_Toc157707862)

[2.2 Аналіз функцій системи 8](#_Toc157707863)

[2.3 Проєктування графічного інтерфейсу 8](#_Toc157707864)

[2.4 Розробка структури системи 8](#_Toc157707865)

[3 Розробка програмного забезпечення системи 11](#_Toc157707866)

[3.1 Розробка UML діаграми класів 11](#_Toc157707867)

[3.2 Розробка UML діаграми використання 12](#_Toc157707868)

[3.3 Опис класів програмного комплексу 12](#_Toc157707869)

[4 Аналіз ефективності програмного забезпечення 15](#_Toc157707870)

[4.1 Базовий аналіз ефективності програмного забезпечення 15](#_Toc157707871)

[4.2 Аналіз ефективності компонентів програмного забезпечення 15](#_Toc157707872)

[4.3 Тестування програмного забезпечення 15](#_Toc157707873)

[5 Розробка документів на супроводження програмного забезпечення 16](#_Toc157707874)

[5.1 Інструкція програміста 16](#_Toc157707875)

[5.2 Інструкція користувачеві 16](#_Toc157707876)

[Висновки 17](#_Toc157707877)

[Перелік джерел посилання 18](#_Toc157707878)

[ДОДАТОК А 19](#_Toc157707879)

[Додаток Б 20](#_Toc157707880)

# Вступ

У сучасному світі, де технології безперервно розвиваються, моделювання структур графів стає ключовою складовою для вирішення різноманітних завдань у галузі комп'ютерних наук. Особливо актуальною стає розробка програмного забезпечення, яке не лише ефективно відтворює графічні структури, але і використовує об'єктно-орієнтований підхід для досягнення оптимальної функціональності та надійності.

Ця курсова робота спрямована на дослідження та розробку програмного забезпечення для моделювання структур графів з використанням об'єктно-орієнтованого програмування (ООП). ООП вже довго визнано як ефективний метод побудови програмних систем, але його застосування в контексті моделювання графів ще потребує ретельного вивчення та оптимізації.

Основною метою мого дослідження є створення програмного продукту, який буде не лише демонструвати високий рівень точності відображення графічних структур, але і забезпечувати зручний інтерфейс для користувача та широкі можливості взаємодії з графами. У цьому контексті я визначив ряд конкретних завдань, що включають в себе реалізацію основних операцій з графами, ефективну обробку великої кількості даних та взаємодію з іншими об'єктами програми.

Ця робота має значення для розширення можливостей в галузі моделювання структур графів, враховуючи не лише теоретичні аспекти, але і практичні можливості реалізації програмного забезпечення, що використовує ООП. Я вірю, що мій внесок допоможе вирішити виклики, пов'язані зі складністю та розміром графічних структур у сучасних інформаційних системах.

Начало формы

# Огляд та аналіз сучасних методів та засобів проектування програмного забезпечення

## Огляд сучасного стану питання

## Основні шляхи вирішення поставленої задачі

## Формулювання мети досліджень

# Проектування програмного забезпечення системи

## Постановка мети

Основною метою цієї курсової роботи є розробка програмного забезпечення для моделювання структур графів з використанням об'єктно-орієнтованого програмування (ООП). Програмний продукт повинен демонструвати високий рівень точності відображення графічних структур та забезпечувати зручний інтерфейс для користувача.

## Аналіз функцій системи

## Проєктування графічного інтерфейсу

## Розробка структури системи

Структурна модель системи, показана на рисунку 2.1, визначає основні компоненти та їх взаємозв'язки.

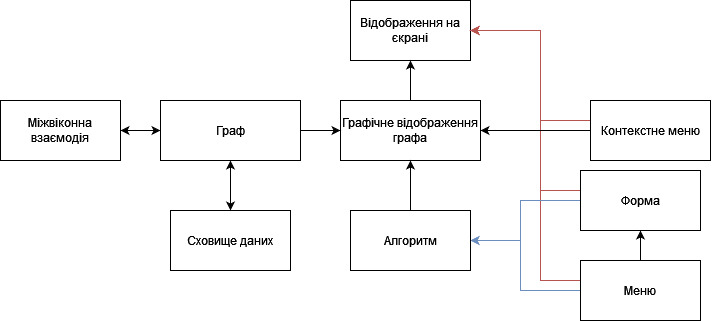


Рисунок 2.1 – Структурна модель застосунку

Система складається з основних компонентів, що взаємодіють між собою:

* відображення графа. Це основний компонент системи;
* контекстне меню. Потрібне для створення та видалення нод;
* меню. Потрібне для відкриття форми та запуску алгоритмів. Основна точка взаємодії з користувачем;
* форма для запуску алгоритму. Потрібна для запуску алгоритму з деякими вхідними даними;
* алгоритм. Алгоритм який виконується вважається окремим компонентом;
* сховище даних. Потрібне для збереження стану застосунку, такого як внутрішній зсув та пресети;
* міжвіконна взаємодія. Окремий складний компонент.

Центральним елементом є компонент відображення графа, що виступає як основний модуль системи. Він відповідає за графічне представлення структури графа на екрані, забезпечуючи зручну взаємодію з користувачем та аналіз графічної інформації.

Контекстне меню відповідає за можливість створення та видалення вузлів графа. Цей компонент дозволяє користувачеві динамічно редагувати структуру графа, адаптуючи її під конкретні вимоги та завдання.

Меню, яке є основною точкою взаємодії з користувачем, відповідає за відкриття форми та запуск алгоритмів. Це важливий компонент, який забезпечує зручний доступ до основних функцій системи.

Форма для запуску алгоритмів використовується для введення вхідних даних та запуску алгоритму. Цей компонент дозволяє користувачеві взаємодіяти з алгоритмуми, вказуючи необхідні параметри та спостерігаючи за їхнім виконанням.

Алгоритм, що виконується, розглядається як окремий компонент системи. Цей елемент відповідає за реалізацію конкретного алгоритмічного підходу та взаємодію з іншими компонентами.

Сховище даних використовується для зберігання стану застосунку, таких як внутрішній зсув та пресети. Цей компонент забезпечує постійний доступ до поточного стану системи та можливість збереження його для подальшого використання.

Міжвіконна взаємодія розглядається як окремий складний компонент, що забезпечує гармонійну взаємодію різних вікон системи та підтримує цілісність користувацького інтерфейсу.

Система використовує об'єктно-орієнтований підхід для моделювання компонентів. Наприклад, інтерфейси та класи використовуються для представлення нод, ребер, GUI, контролера та інших об'єктів системи. Це дозволяє створювати гнучкі та розширювані об'єктні структури, сприяє повторному використанню коду та підтримці чистоти дизайну.

Компоненти системи взаємодіють динамічно, реагуючи на події та зміни, що виникають у процесі взаємодії з користувачем або виконання алгоритмів. Це забезпечує живий та відзивчивий характер системи.

# Розробка програмного забезпечення системи

## Розробка UML діаграми класів

На рисунку 3.1 представлена UML діаграма класів.

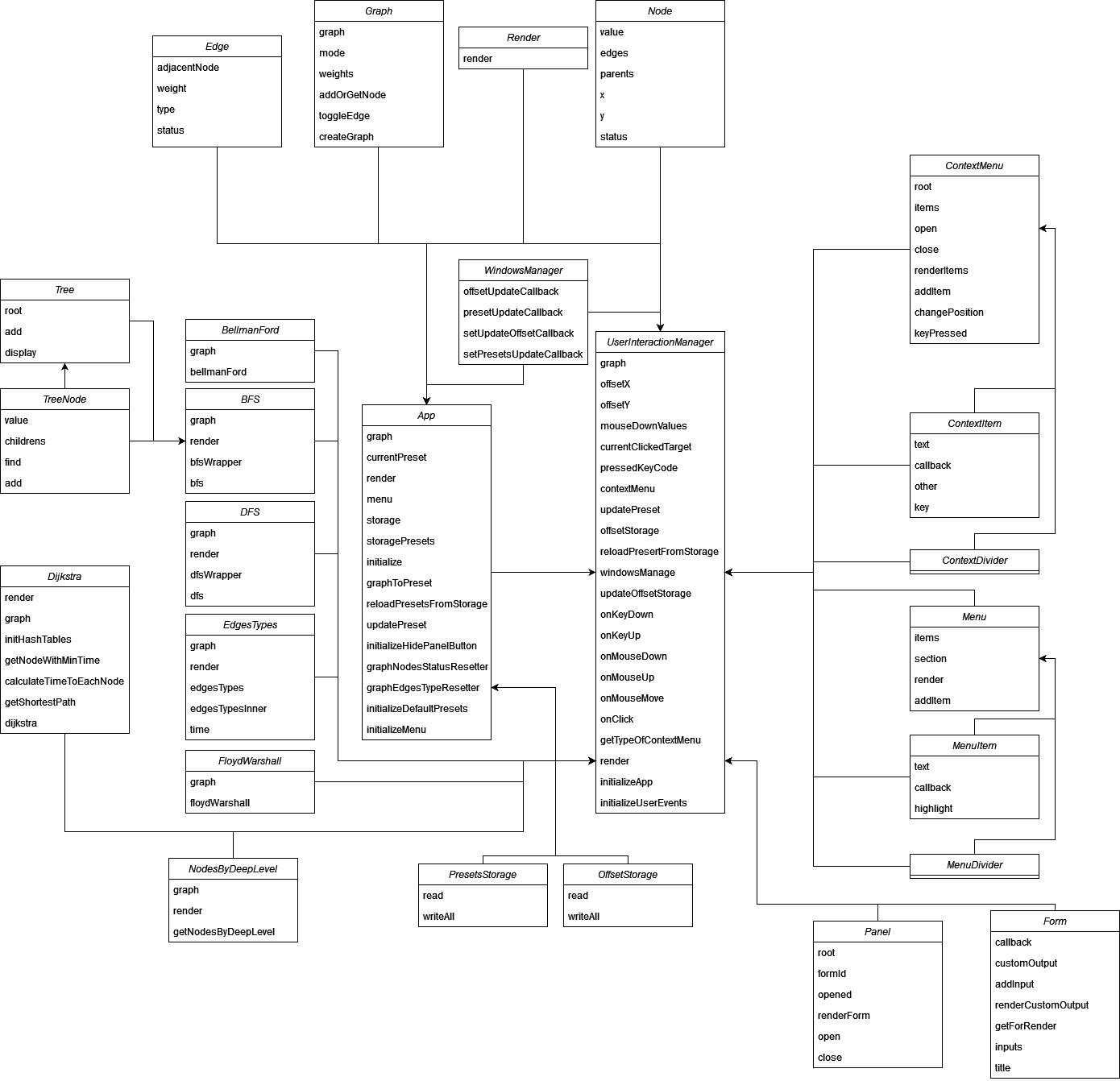


Рисунок 3.1 – UML діаграма класів

## Розробка UML діаграми використання

UML діаграма використання визначає, які функції можуть бути використані користувачами. До основних входять створення та редагування графів, взаємодія з контекстним меню та основними функціями графічного інтерфейсу.

На рисунку 3.2 показана UML діаграма використання.

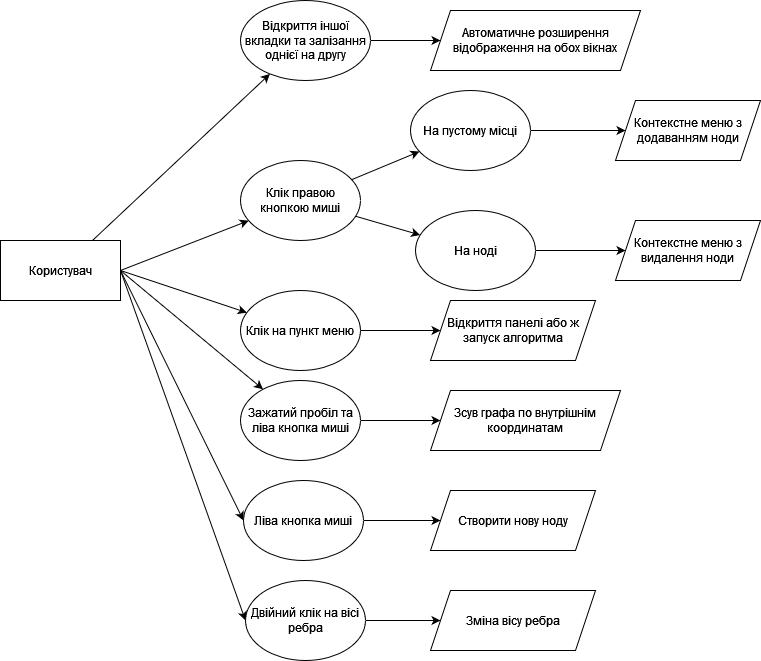


Рисунок 3.2 – UML діаграма використання

## Опис класів програмного комплексу

Опис класів починається з збереження даних в сховищах. Механізм роботи зі сховищем має в собі 1 інтерфейс та 3 класи.

Почнемо з інтерфейса InnerStorageType який описує інтерфейс зовнішнього сховища в якому можуть зберігатись дані. Система зроблена таким чином, що ми можемо перед виконання коду під’єднати якесь інше сховище через клас адаптер, паттерн проєктування адаптер.

Інтерфейс InnerStorageType має два методи:

* getItem який приймає ключ та видає результат
* setItem який приймає ключ та значення та видає результат

Головним класом який пов’язує окремі сховища данних та зовнішнє є Storage. Цей клас має наступні методи та властивості:

* key – приватне поле для ключа
* innerStorage – приватне поле для зовнішнього сховища, яке має тип InnerStorageType
* getItem() – метод для отримання даних зі сховища
* setItem(value: string) – приймає строку та встановлює її в сховище

Клас Storage потрібен, оскільки кожне окреме сховище може мати своє окреме зовнішне сховище. Тобто, у нас можуть білети зберігатись на одному сховищі з одним адаптером, а рейси – на ішному з іншим адаптером.

Розглянемо класи сховищ. Перший клас сховища – FlightStorage потрібен для зберігання даних про рейси. Цей клас має наступні властивості та методи:

* storage – приватне поле яке вказує яке саме сховище ми використовуємо
* setItem(data: Flight[]) – приймає масив рейсів та записує в сховище
* getItem(): Flight[] – повертає масив рейсів зі сховища

Наступне сховище – PassengerStorage потрібен для зберігання даних про пасажирів. Цей клас має наступні властивості та методи:

* storage – приватне поле яке вказує яке саме сховище ми використовуємо
* setItem(data: Passenger[]) – приймає масив пасажирів та записує в сховище
* getItem(): Passenger[] – повертає масив пасажирів зі сховища

Останнє сховище TicketStorage потрібен для зберігання даних про білети. Цей клас має наступні властивості та методи:

* storage – приватне поле яке вказує яке саме сховище ми використовуємо
* setItem(data: Ticket[]) – приймає масив білетів та записує в сховище
* getItem(): Ticket [] – повертає масив білетів зі сховища

Розглянемо основні базові застосунку.

Клас Flight – клас, який зберігає данні про рейс та має наступні поля:

* id – Унікальний ідентифікатор
* destination – Місто, в яке прилетить літак
* startDate – Дата початку польоту
* endDate – Дата кінця польоту
* price – Ціна

Клас Passenger – клас, який відповідає за базові дані пасажира та має наступні поля:

* id – Унікальний ідентифікатор
* firstName – Ім’я пасажира
* lastName – Призвіще пасажира
* phoneNumber – Номер телефону
* passportNumber – Номер пасспорта
* email - Пошта

Клас Ticket – клас, який відповідає за базові дані квитка та має наступні поля:

* id – Унікальний ідентифікатор
* passenger – Ідентифікатор пасажира
* flight – Ідентифікатор рейсу
* position – Позиція в літаку

Головним інтерфейсом для пов’язування сховищ даних та основної логіки є StorageType. Цей інтерфейс має наступні поля:

* getItem(): T[]
* setItem(data: T[]): void

Базовий клас – один елемент. Для роботи з масивом існує спеціальний менеджер, який має імплементує інтерфейс BaseManager та має наступні властивості та методи:

* items – Поле для збереження елементів
* initializeFromStorage(storage: StorageType<any>) – Метод для інітіалізації даних зі сховища
* add(data: any) – Метод для додавання елементів
* remove(id: string) – Метод для видалення елементів
* edit(id: string, newData: T) – Метод для редагування елементів
* getBySearch(search: string) – Метод для пошуку елементів

Клас FligtManager – клас для управління рейсами, цей клас має наступні поля та методи:

* items – Поле для збереження елементів рейсів
* initializeFromStorage(storage: StorageType<Flight>) – Метод для інітіалізації даних зі сховища
* add(data: Flight) – Метод для додавання елементів
* remove(id: string) – Метод для видалення елементів
* edit(id: string, newData: Flight) – Метод для редагування елементів
* getBySearch(search: string) – Метод для пошуку елементів

**PassengerManager**

* items – Поле для збереження елементів рейсів
* initializeFromStorage(storage: StorageType<Passenger>) – Метод для інітіалізації даних зі сховища
* add(data: Passenger) – Метод для додавання елементів
* remove(id: string) – Метод для видалення елементів
* edit(id: string, newData: Passenger) – Метод для редагування елементів
* getBySearch(search: string) – Метод для пошуку елементів

**TicketManager**

* items – Поле для збереження елементів рейсів
* initializeFromStorage(storage: StorageType<Ticket>) – Метод для інітіалізації даних зі сховища
* add(data: Ticket) – Метод для додавання елементів
* remove(id: string) – Метод для видалення елементів
* edit(id: string, newData: Ticket) – Метод для редагування елементів
* getBySearch(search: string) – Метод для пошуку елементів

*<<Interface>>*  
**GUIFactory**

* - storage: StorageType<any>
* manager: BaseManager<any>
* creatOpenFormButton(): HTMLButtonElement
* getTableForRender(): HTMLTableElement
* getManager(): BaseManager<any>

**FlightFactory**

* - storage: FlightStorage
* - manager: FlightManager
* - currentTableRender: HTMLTableElement
* createOpenFormButton(): HTMLButtonElement
* - rerenderTable(): void
* - onDelete(item: string[]): void
* - onEdit(item: string[]): void
* getTableFoRender(): HTMLTableElement
* - createTable(onDelete, onEdit): HTMLTableElement
* getManager(): FlightManager

**PassengerFactory**

- storage: PassengerStorage

- manager: PassengerManager

- currentTableRender: HTMLTableElement

+ createOpenFormButton(): HTMLButtonElement

- rerenderTable(): void

- onDelete(item: string[]): void

- onEdit(item: string[]): void

+ getTableFoRender(): HTMLTableElement

- createTable(onDelete, onEdit): HTMLTableElement

+ getManager(): PassengerManager

**TicketFactory**

- storage: TicketStorage

- manager: TicketManager

- currentTableRender: HTMLTableElement

+ createOpenFormButton(): HTMLButtonElement

- rerenderTable(): void

- onDelete(item: string[]): void

- onEdit(item: string[]): void

+ getTableFoRender(): HTMLTableElement

- createTable(onDelete, onEdit): HTMLTableElement

+ getManager(): TicketManager

**Form**

+ title: string

+ onClose: () => void

+ onSubmit(formData: Record<string, string>): void

+ type: 'create' | 'update'

+ initialData: Record<string, string>

+ addInput(type: string, name: string, innerValue: string, value: string): void

+ mount(): HTMLDivElement

+ umount(): void

**Button**

+ className: string

+ textConent: string

callback: (e: MouseEvent): void

+ render(): HTMLButtonElement

**Table**

+ heads: string[]

+  items: T[][]

+ onDelete: (item: T[]) => void

+ onEdit: (item: T[]) => void

- getTableElements(item: T[]): HTMLCellElement[]

- getTableHeadElements(heads: string[]): HTMLTableCellElement[]

+ render(): HTMLTableElement

*<<Interface>>*  
**Tab**

+ title: string  
+ callback: () => void

+ active: boolean

**Tabs**

- tabs: Tab

+ addTab(title: string, callback: () => void): void

+ setActive(title: string): void

- generateButton(tabInfo: Tab): HTMLButtonElement

+ getForRender(): HTMLDivElement

**App**

+ root: Element

- factory: GUIFactory<any>

- tabs: Tabs

+ render(): void

# Аналіз ефективності програмного забезпечення

## Базовий аналіз ефективності програмного забезпечення

Основні аспекти базового аналізу ефективності програмного забезпечення:

− швидкодія;

− масштабованість;

− надійність;

− зручність використання;

Швидкодія є ключовим аспектом в розробці програмного забезпечення для автоматизації роботи диспетчера в касах аерофлоту. Для досягнення високого рівня швидкодії потрібно зосередити увагу на реалізації ефективного механізму відображення графічних даних, що забезпечить високу швидкість рендерингу та мінімізацію витрат ресурсів.

Масштабованість системи є важливим аспектом, оскільки програма повинна ефективно працювати з великими обсягами інформації. Забезпечення плавного та стабільного відображення графічних елементів є основною ціллю.

Надійність є ще одним ключовим критерієм ефективності програмного забезпечення, особливо для систем, які використовуються в диспетчерській роботі кас аерофлоту. Надійність вказує на стійкість та стабільність програми під час роботи, а також на її здатність уникати виникнення помилок чи збоїв.

Зручність використання це інша важлива характеристика ефективності програмного забезпечення, оскільки вона визначає, наскільки зручно та ергономічно користувач може взаємодіяти з системою. Потрібно враховувати якість та зручність використання графічного інтерфейсу програми, його логічність та інтуїтивність. Також легкість навчання користуванням програмою є немаловажливим.

Враховуючи ці аспекти, забезпечується не тільки висока швидкодія програмного забезпечення, але й низький поріг для використання програми. Аналіз ефективності компонентів програмного забезпечення.

## Тестування програмного забезпечення

Тестування програмного забезпечення дозволяє підтвердити функціональність та надійність розробленого програмного продукту.

Були протестовані наступні функції:

− функція відображення даних про рейс, пасажира та білет в вигляді таблиці (дивитись рисунок 4.1);

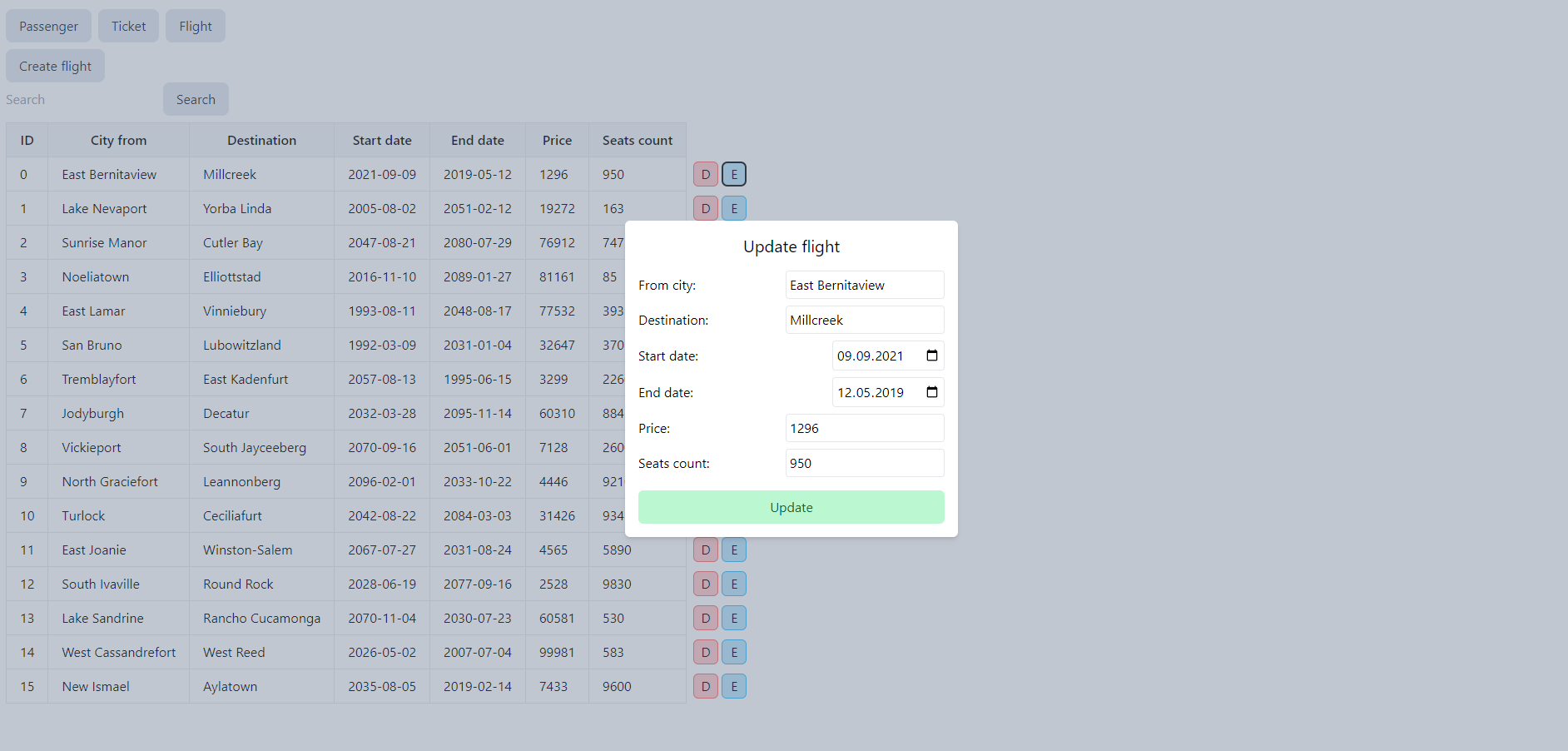
− функція зміни рейсу, пасажира та білету за допомоги форми (дивитись рисунок 4.2);

− функція створення нового рейсу, пасажира та білету через форму (дивитись рисунок 4.3);

− функція видалення рейсу, пасажира та білету (дивитись рисунок 4.4);

− функція пошуку по даним (дивитись рисунок 4.5);.

Рисунок 4.1 – Тестування відображення інформації

  
 Рисунок 4.2 – Тестування редагування інформації

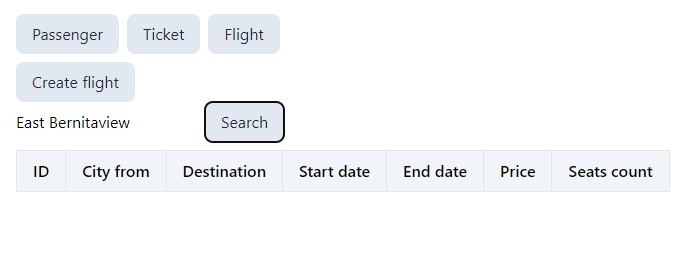


Рисунок 4.3 – Тестування видалення інформації

Після видалення відправлення, його миттєво було прибрано із бази даних. Продемонстровано за рахунок пошуку за одним із критерієв видаленого відправлення.

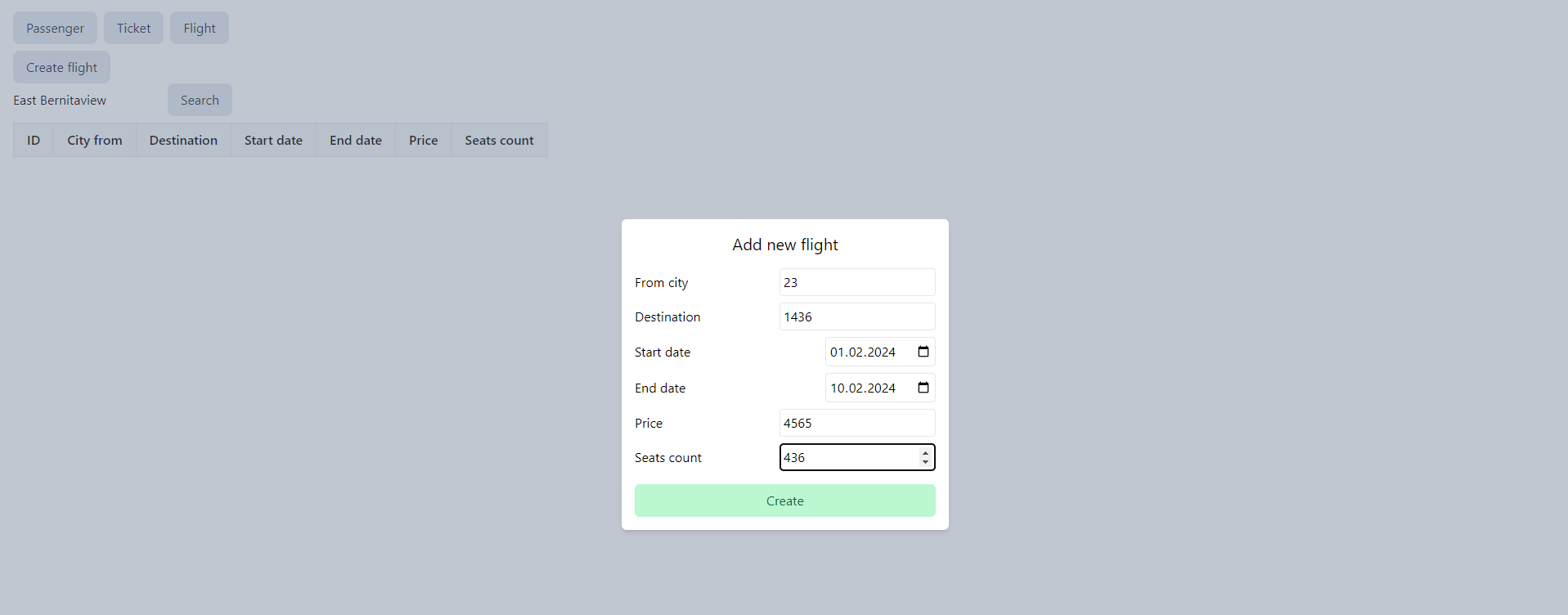
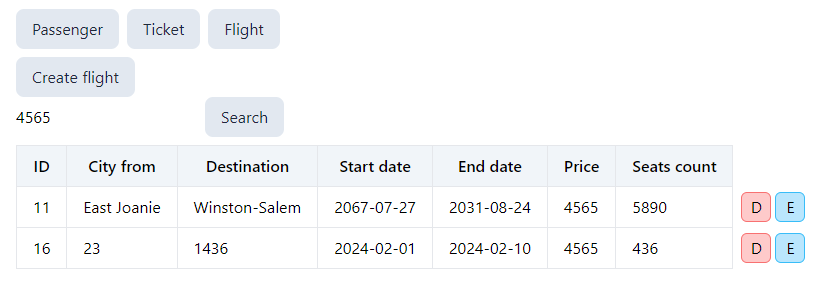


Рисунок 4.4 – Тестування додавання інформації

У тестуванні було додано дані з відповідними полями, що можна побачити на рисунку 4.5.

Рисунок 4.5 – Тестування пошуку інформації

Після проведення тестування застосунку виявлено відсутність будь-яких помилок чи несправностей в його роботі. Всі функціональні та інтерфейсні елементи працюють вірно. Тестування вказує на високу якість програмного забезпечення, адекватну відповідь на користувацькі взаємодії та відсутність технічних недоліків.

# Розробка документів на супроводження програмного забезпечення

## Інструкція програміста

Ця інструкція відповідає на запитання як розгорнути проєкт локально, як в нього вносити зміни та як тестувати застосунок.

Розглянемо як розгорнути проєкт локально.

Спочатку потрібно клонувати репозиторій застосунку на вашому робочому пристрої. Використовуйте команду git clone та вказуйте URL репозиторію. На рисунку 5.1 показаний приклад клонування репозиторію.

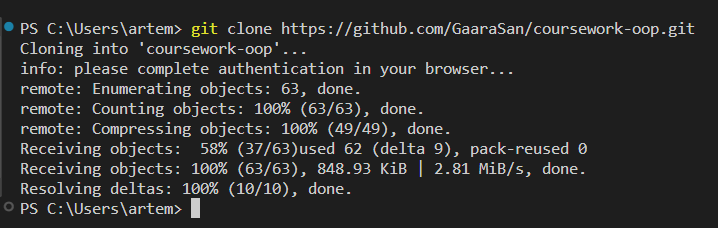


Рисунок 5.1 – Клонування репозиторію

Встановлення залежностей. Переконайтеся, що у вас встановлений пакетний менеджер pnpm. Використовуйте команду pnpm install, щоб встановити всі необхідні залежності. Це забезпечить коректну роботу застосунку. На рисунку 5.2 показаний приклад встановлення залежностей.

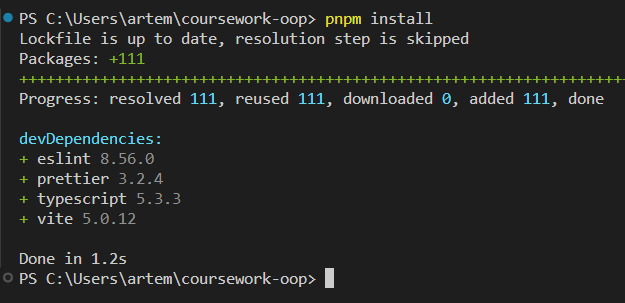


Рисунок 5.2 – Встановлення залежностей

Запуск застосунку для розробки. Використовуйте команду npm run dev, щоб запустити застосунок у режимі розробки. Це дозволить вам спостерігати за змінами в реальному часі та вносити власні виправлення. На рисунку 5.3 показаний приклад запуску застосунку.

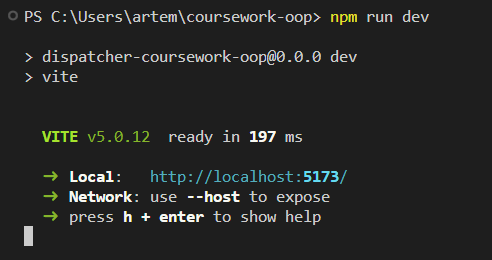


Рисунок 5.3 – Запуск застосунку

Тепер проєкт розгорнутий локально та готовий до редагування або ж використання.

## Інструкція користувачеві

Заходячи на сайт користувачу буде доступні функції для додання пасажирів, квитків та відправлень. Додання інформації виконується за допомогою спеціальної форми, яка зображена на рисунку 5.4. Після цього потрібно натиснути кнопку «Create» для підтвердження додання інформації у базу даних або можна натиснути поза межами форми для того щоб закрити її.

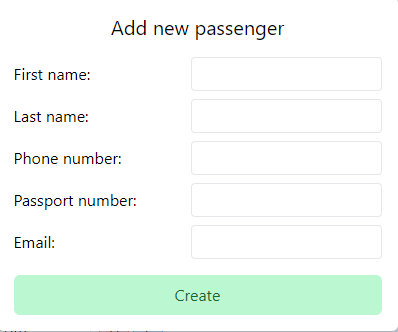


Рисунок 5.4 – Форма для додання даних

Далі користувач може переключатися між вкладками для відображення таблиць з пасажирами, заброньованими квитками та відправленнями. Також виконувати пошук по таблиці та видаляти або редагувати дані (дивитись рисунок 5.5).

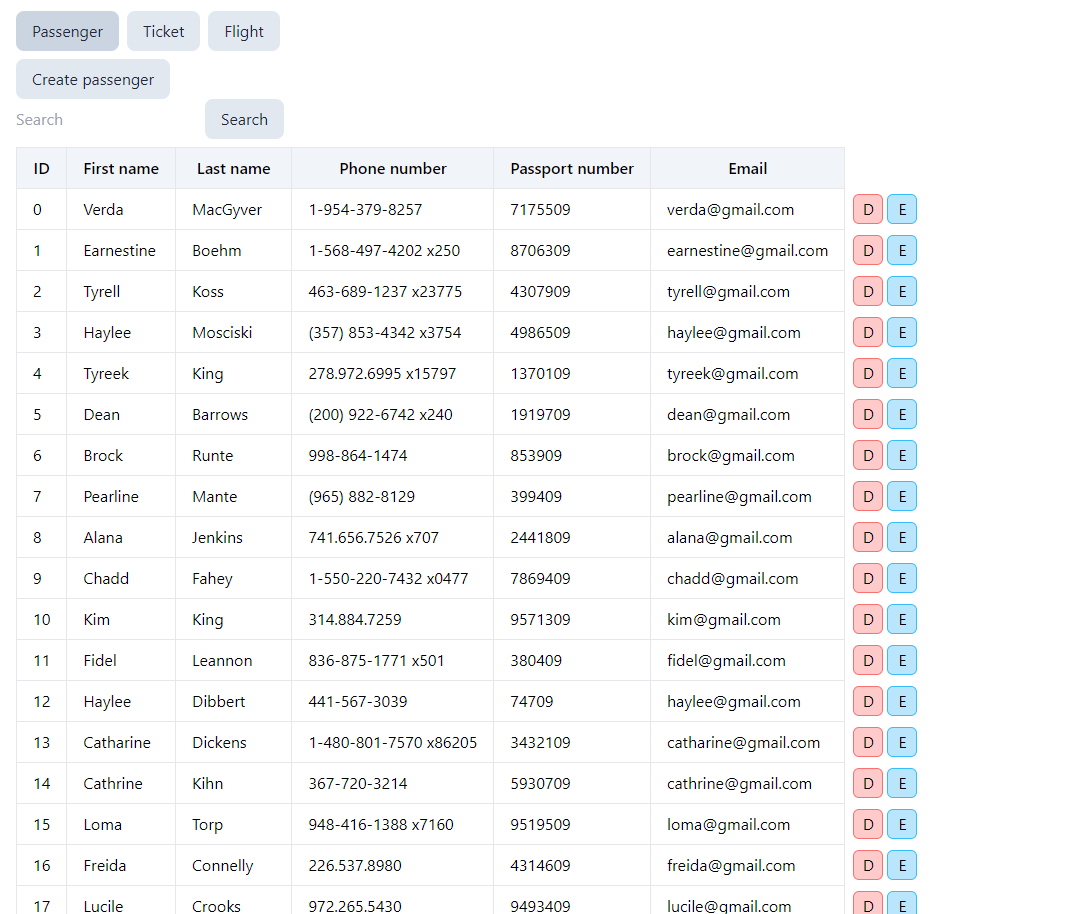


Рисунок 5.5 – Основний вид

Пошук можливий за усіма полями. Потрібно ввести у текстове поле потрібне значення та натиснути кнопку пошуку. Усі знайдені результати будуть відображені у таблиці нижче, як зображено на рисунку 5.6.

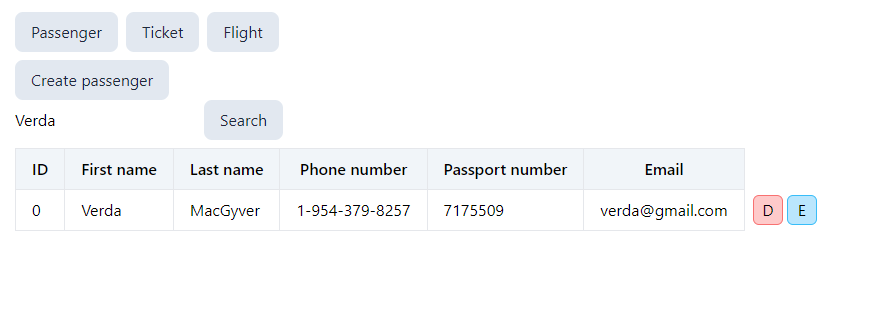


Рисунок 5.6 – Реалізація пошуку даних

Для видалення даних потрібно натиснуту відповідну кнопку «D», після чого дані будуть миттєво видалені із бази даних (дивитись рисунок 5.7).

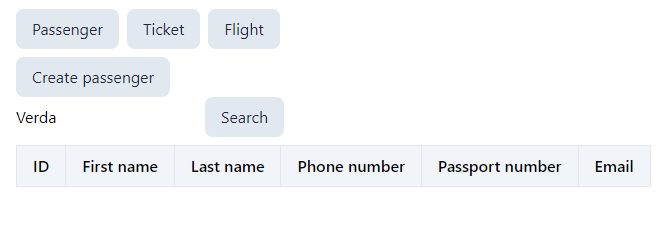


Рисунок 5.7 – Демонстрація видалення даних

Редагування даних виконується при натисканні кнопки «E». Диспетчеру потрібно буде змінити потрібні дані у спливаючому вікні. Інформація для редагування буде сама «підтянута» у поля форми, як зображено на рисунку 5.8. Після введення нових даних, потрібно натиснути кнопку «Update» для підтвердження редагування.

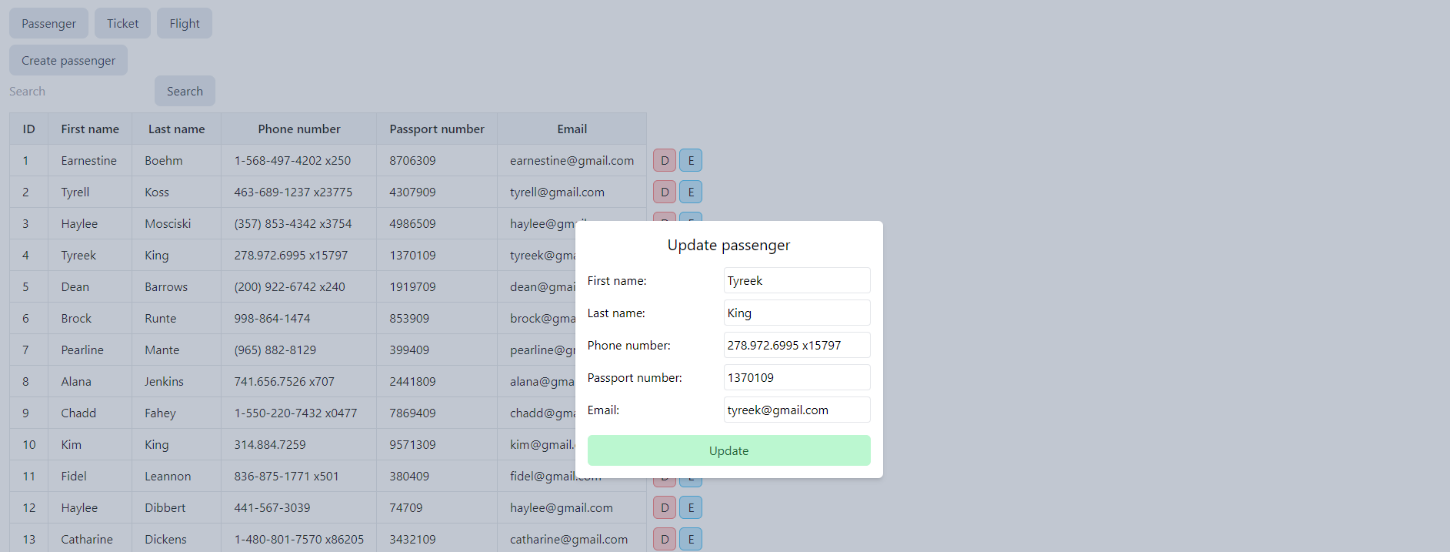


Рисунок 5.8 – Редагування даних

# Висновки

В результаті виконання цієї курсової роботи було створено сервіс, який надає можливість візуально моделювати графи та спрощує задачу візуалізації алгоритмів на графі за рахунок надання зручного та швидкого інтерфейсу.

В результаті огляду та аналізу сучасних методів та засобів проектування програмного забезпечення було встановлено, що зараз йде активний розвиток багатьох сфер життя і всі ці сфери потребують в тій чи іншій мірі роботу з графами.

Визначено мету створення програми та її основні функції.

Виконано проектування програмного забезпечення, де було проаналізовано функції системи, розроблемо діаграми використання, був спроєктований графічний інтерфейс. Визначено мету створення програми та її основні функції.

Розроблено застосунок згідно з аналізами та проєктами, розроблена структура системи, описані всі класи програмного комплексу та розроблена діаграма класів.

Проведено аналіз ефективності програмного забезпечення. Застосунок було перевірено на швидкодію та масштабованість, проаналізована ефективність кожного окремного компонента та виконано тестування.

Також розроблено методику роботи користувача з системою окремо для інших розробників, окремо для звичайних користувачів

# Перелік джерел посилання

* 1. Hatch, S.V. Computerized Engine Controls / S.V. Hatch. – Boston: Cengage Learning, 2016. – 688 p.
  2. Czichos, H. Measurement, Testing and Sensor Technology. Fundamentals and Application to Materials and Technical Systems / H. Czichos. – Berlin: Springer, 2018. – 213 p.
  3. Kaźmierczak, J. Data Processing and Reasoning in Technical Diagnostics / J. Kaźmierczak, W. Cholewa. – Warszawa: Wydawnictwa Naukowo-Techniczne, 1995. – 186 p.
  4. Diagnostics as a Reasoning Process: From Logic Structure to Software Design / [M. Cristani, F. Olivieri, C. Tomazzoli, L. Vigano, M. Zorzi] // Journal of Computing and Information Technology. – 2018. – Vol. 27 (1). –   
     P. 43-57.
  5. Wieczorek, A.N. Analysis of the Possibility of Integrating a Mining Right-Angle Planetary Gearbox with Technical Diagnostics Systems / A.N. Wieczorek // Scientific Journal of Silesian University of Technology. Series Transport. – 2016. – Vol. 93. – P. 149-163.
  6. Tso, B. Classification Methods for Remotely Sensed Data / B. Tso, P.M. Mather. – Boca Raton : CRC Press, 2016. – 352 p.
  7. Oppermann, A. Regularization in Deep Learning – L1, L2, and Dropout [Electronic resource]. – Access mode: https://www.deeplearning-academy.com/p/ai-wiki-regularization.
  8. Classic Regularization Techniques in Neural Networks [Electronic resource]. – Access mode: https://medium.com/@ODSC/classic-regularization-techniques-in-neural-networks-68bccee03764.

# ДОДАТОК А

**CODE**

# Додаток Б

Слайди презентації:

Слайд 1

Слайд 2

Слайд 3

Слайд 4

Слайд 5

Слайд 6

Слайд 7

Слайд 8

Слайд 9