Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни «Об’єктно-орієнтоване програмування»

ДОВІДНИК АБІТУРІЄНТА

Керівник , професор Володимир Михайлович Бондарєв

Студент гр. ПЗПІ-19-4 Гриненко Анастасія Олексіївна

Комісія:

доцент \_\_\_\_\_\_\_\_\_\_\_\_\_ Кравець Н. С.

доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мар'їн С. О.

асистент \_\_\_\_\_\_\_\_\_\_\_ Олійник О. В.

Харків 2019

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: ***Об’єктно-орієнтоване програмування***

Спеціальність: ***121***  ***Інженерія програмного забезпечення***

Освітня програма: ***Програмна інженерія***

Курс 1 .Група *ПЗПІ-19-4* . Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

***Гриненко Анастасії Олексіївни***

(Прізвище, Ім'я, По батькові)

1 Тема проекту: *Довідник абітурієнта*

2 Термін здачі студентом закінченого проекту: ***“31” - травня - 2019 р***.

3 Вихідні дані до проекту:

*Специфікація програми, методичні вказівки до виконання курсової роботи*

4 Зміст розрахунково-пояснювальної записки:

*Вступ,* *специфікація програми, проектна специфікація, інструкція користувача, висновки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5 Перелік графічного матеріалу:

*Діаграми класів та прецедентів,* *схема об’єктної моделі, алгоритми, приклади екранних форм\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 30-03-2019 р. |
| 2 | Формулювання вимог до програми | 30-03-2019 – 01-04-2019 р. |
| 3 | Розробка підсистеми формування розкладу руху автобусів та пошуку рейсів. | 01-04-2019 – 05-04-2019 р. |
| 4 | Розробка підсистеми взаємодії з посадковою відомістю (замовлення/повернення квитків). | 05-04-2019 – 12-04-2019 р. |
| 5 | Встановлення різних рівнів доступу до функціоналу шляхом створення двох типів користувачів – диспетчера та клієнта. | 12-04-2019 – 17-04-2019 р. |
| 6 | Розробка функцій зберігання та завантаження даних | 17-04-2019 – 20-04-2019 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 20-04-2019 – 20-05-2019 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 20-05-2019 – 10-06-2019 р. |
| 9 | Захист | 10-06-2019 – 15-06-2019 р. |

Студент Гриненко Анастасія Олексіївна

Керівник Бондарєв Володимир Михайлович

(Прізвище, Ім'я, По батькові)

«30 » березня 2019 р.

**РЕФЕРАТ**

Пояснювальна записка до курсової роботи: 49 с., 15 рис., 5 додатків, 5 джерел.

ДОВІДНИК, УНІВЕРСИТЕТ, АБІТУРІЄНТ, КЛАС, МОВА ПРОГРАМУВАННЯ C#, ОБ’ЕКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, ПЛАТФОРМА .NET, ПРОГРАМА.

Метою роботи є розробка програми «Довідник абітурієнта» на засадах об'єктно-орієнтованого програмування для отримання інформації щодо закладів вищої освіти України.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2017, Windows Forms, платформи .NET Framework 4.7, мови програмування C#.

У результаті отримана програма під назвою «Довідник абітурієнта», яка дозволяє звичайному користувачу переглядати заклади вищої освіти, продивлятися інформацію про університети, додавати та видаляти університети зі Списку обраних. Авторизований користувач (адміністратор) може змінювати інформацію щодо ЗВО, видаляти вже існуючі університети та додавати нові.

**ЗМІСТ**

[ВСТУП 7](#_Toc11182230)

[1 СПЕЦИФІКАЦІЯ ПРОГРАМИ 8](#_Toc11182231)

[1.1 Діаграма прецедентів 8](#_Toc11182232)

[1.2 Бізнес-вимоги 8](#_Toc11182233)

[1.3 Вимоги користувачів 9](#_Toc11182234)

[1.4 Функціональні вимоги 9](#_Toc11182235)

[1.5 Нефункціональні вимоги 10](#_Toc11182236)

[1.6 Опис інтерфейсу користувача 11](#_Toc11182237)

[2 ПРОЕКТНА СПЕЦИФІКАЦІЯ 21](#_Toc11182238)

[2.1 Діаграма класів 21](#_Toc11182239)

[2.2 Об’єктна структура програми 21](#_Toc11182240)

[2.2.1 Клас Station 22](#_Toc11182241)

[2.2.2 Клас Route 23](#_Toc11182242)

[2.2.3 Клас Timetable 24](#_Toc11182243)

[2.2.4 Клас Ticket 25](#_Toc11182244)

[2.2.5 Клас TicketPrice 26](#_Toc11182245)

[2.2.6 Клас Password 26](#_Toc11182246)

[2.2.7 Клас PassengerList 27](#_Toc11182247)

[2.2.8 Перечислення User 28](#_Toc11182248)

[3 ІНСТРУКЦІЯ КОРИСТУВАЧА 29](#_Toc11182249)

[3.1 Встановлення, запуск та головне вікно 29](#_Toc11182250)

[3.2 Вибір маршруту та замовлення квитка 29](#_Toc11182251)

[3.3 Повернення квитка 30](#_Toc11182252)

[3.4 Авторизація та доступ диспетчера 30](#_Toc11182253)

[ДОДАТОК А Методи класу Route 34](#_Toc11182254)

[ДОДАТОК Б Методи класу Timetable 36](#_Toc11182255)

[ДОДАТОК В Методи класу TicketPrice 40](#_Toc11182256)

[ДОДАТОК Г Клас Password 42](#_Toc11182257)

[ДОДАТОК Д Методи класу PassengerList 43](#_Toc11182258)

ВСТУП

Автобусні перевезення неможливі без чітко встановленого розкладу, який дозволяє потенційному пасажиру вибрати рейс, що найбільш задовольняє його планам дістатися потрібного пункту. Задача будь-якого підприємства – переконати потенційного клієнта у тому, що саме їх послуги підходять йому найкраще. Для автовокзалу є найбільш суттєвим забезпечити легкість пасажирів у перегляді розкладів руху, пошуку маршрутів, замовленні та поверненні квитків.

Розроблена програма дозволить користувачу виконувати усі згадані вище дії. Повний доступ до її функцій отримує авторизований користувач – диспетчер. Він може переглядати посадкову відомість, редагувати розклад руху автобусів, встановлювати ціни на квитки. Доступ до облікового запису диспетчера здійснюється через пароль, який може бути змінений.

Окрім розробки вище описаної програмної системи метою даної курсової роботи є освоїти мову програмування С#, навчитися об’єктно-орієнтованому програмуванню, розвити навички роботи в Windows Forms.

Задачі роботи:

* Розробка системи зчитування та запису даних;
* Створення логіки роботи системи;
* Робота з користувачем;
* Розробка логіки додавання, пошуку та видалення інформації;
* Тестування готового продукту;
* Створення стійкості програми.

1. СПЕЦИФІКАЦІЯ ПРОГРАМИ
   1. Діаграма прецедентів

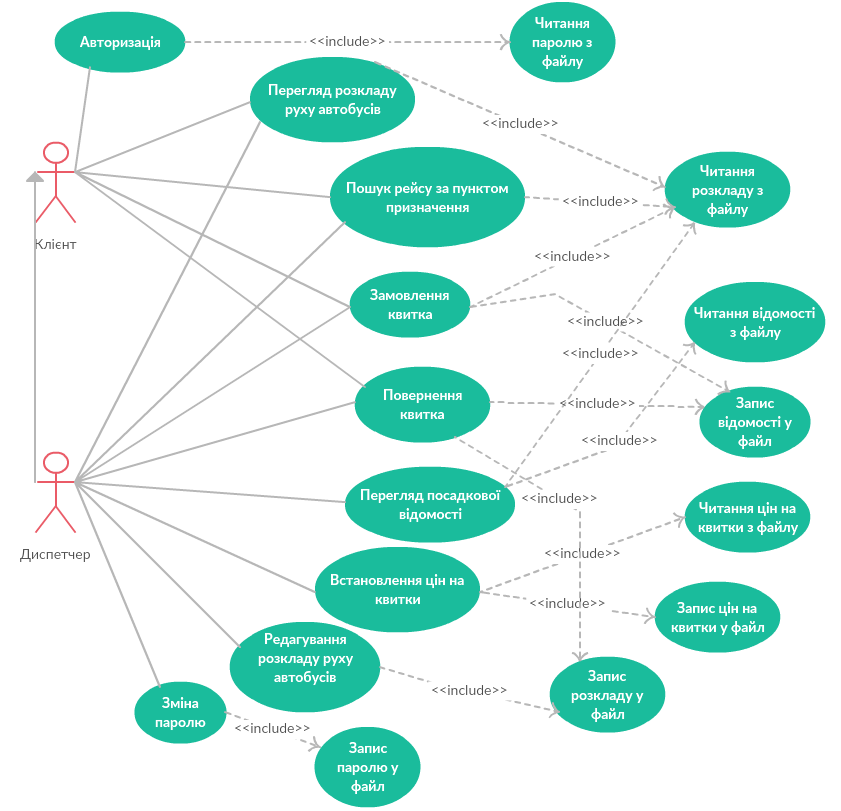


Рисунок 1.1 – Діаграма прецедентів

* 1. Бізнес-вимоги

Серед бізнес-вимог слід зазначити наступні:

* Попередній продаж та повернення квитків;
* Оформлення посадкової відомості;
* Відображення розкладу руху автобусів.
  1. Вимоги користувачів

Серед вимог користувачів слід зазначити наступні:

* Перегляд розкладу руху автобусів;
* Пошук рейсу за вказаним пунктом призначення;
* Замовлення квитка;
* Повернення квитка;
* Авторизація користувача (введення паролю);
* Перегляд посадкової відомості (диспетчер);
* Встановлення цін на квитки (диспетчер);
* Редагування розкладу руху автобусів (диспетчер);
* Зміна паролю диспетчера (диспетчер).
  1. Функціональні вимоги

Серед функціональних вимог слід зазначити наступні:

* Система має зчитувати з файла розклад руху автобусів та записувати туди зміни у ньому за необхідності;
* Система має зчитувати з файла посадкову відомість за записувати туди зміни у ній за необхідності;
* Система має зчитувати з файла список цін квитків до пунктів призначення за записувати туди зміни у ньому за необхідності;
* Система має звіряти введений пароль з паролем, зчитаним та розшифрованим з файлу;
* Система має шифрувати та зберігати змінений пароль у файл;
* Система має порівнювати введений пункт призначення з пунктами, наявними в розкладі руху автобусів та знаходити усі рейси з ним, що відправляються після нинішньої дати та мають вільні місця;
* Система має зберігати дані про квиток у посадкову відомість;
* Система має порівнювати введені дані про квиток з даними, наявними у відомості та видаляти їх звідти у випадку збігу;
* Система має порівнювати кожний запис у посадковій відомості з розкладом руху автобусів та призначати відповідний статус («Замовлено», «У дорозі», «Прибув», «Скасовано»).
  1. Нефункціональні вимоги

Серед нефункціональних вимог слід зазначити наступні:

* Легкість та простота використання;
* Легкість переміщення;
* Цілісність;
* Ефективність та стійкість до збоїв;
* Взаємодія системи і зовнішнього світу;
* Обмеження дизайну і реалізації.

Серед обмежень дизайну і реалізації слід зазначити наступні:

* Використання платформи .NET Framework 4.5;
* Використання мови програмування C#;
* Використання Windows Forms.
  1. Опис інтерфейсу користувача

Після запуску програми відкривається її головне вікно (рис. 1.2), на якому відображається розклад руху автобусів та яке дає можливість відшукати потрібний рейс, повернути квиток чи авторизуватися.

Рисунок 1.2 – Головне вікно (клієнт)

У цьому вікні є такі елементи:

* Таблиця «Маршрути автобусів» – тільки для читання;
* Поле введення пункту призначення;
* Кнопка «Пошук маршруту»;
* Кнопка «Авторизація»;
* Кнопка «Повернення квитка»;
* Кнопка «Вихід».

Кнопка «Вихід» закриває програму.

Якщо користувач натисне кнопку «Авторизація», він побачить вікно введення паролю (рис. 1.3).

Рисунок 1.3 – Вікно авторизації

У цьому вікні є такі елементи:

* Поле введення паролю диспетчера;
* Кнопка «Увійти»;
* Кнопка «Продовжити як клієнт»;
* Кнопка «Назад»;

Кнопки «Назад» та «Продовжити як клієнт» закривають це вікно.

Якщо пароль, введений у поле, вірний, після натиснення кнопки «Увійти» або клавіші Enter це вікно закривається, а головне вікно модифікується і відкриває йому доступ до функцій диспетчера (рис. 1.4).

Рисунок 1.4 – Головне вікно (диспетчер)

У цьому вікні є всі елементи головного вікна клієнта (рис. 1.1), до яких додається меню з елементами:

* «Посадкова відомість» – «Відкрити»;
* «Ціни на квитки» – «Відкрити»;
* «Пароль» – «Змінити».

Після тире наведено пункти підменю.

Слід зазначити, що авторизований користувач (диспетчер) може редагувати таблицю «Маршрути автобусів».

Якщо знову натиснути кнопку «Авторизація», користувач побачить модифіковане вікно з рис. 1.3 (рис. 1.5). Тут більше немає кнопки «Увійти» та поля введення паролю. Після натиснення кнопки «Продовжити як клієнт» користувач втрачає доступ диспетчера, це вікно закривається та головне вікно набуває вигляду з рис. 1.2.

Рисунок 1.5 – Вікно авторизації після авторизації

Після натиснення у меню «Посадкова відомість» – «Відкрити» відкривається вікно посадкової відомості (рис. 1.6).

Рисунок 1.6 – Вікно посадкової відомості

У ньому є такі елементи:

* Таблиця «Посадкова відомість» (тільки для читання);
* Кнопка «Видалити скасовані»;
* Кнопка «Назад».

Кнопка «Назад» закриває це вікно.

Кнопка «Видалити скасовані» видаляє з відомості усі квитки на скасовані рейси.

Якщо у головному вікні (рис. 1.2, 1.4) натиснути на кнопку «Повернути квиток», відкривається вікно повернення квитка (рис. 1.7).

Рисунок 1.7 – Вікно повернення квитка

У цьому вікні є такі елементи:

* Поле введення імені;
* Поле введення прізвища;
* Поле введення номеру та серії документа;
* Поле введення номеру квитка, що допускає введення не більше 8 цифр;
* Кнопка «Назад»;
* Кнопка «Повернути квиток».

Кнопка «Назад» закриває дане вікно.

Кнопка «Повернути квиток» створює запит на перевірку даних, введених у поля, і, якщо вони коректні, запускає метод видалення запису (квитка) з посадкової відомості та оновляє її вікно (рис. 1.6), якщо воно відкрите.

Якщо у головному вікні (рис. 1.2, 1.4) ввести пункт призначення у поле введення пункту призначення та натиснути кнопку «Пошук маршруту», остання викличе метод, що обійде усі маршрути та знайде ті, де фігурує заданий пункт, є вільні місця і час відправлення пізніше поточного. Якщо такі будуть, відкриється вікно вибору маршруту (рис. 1.8).

Рисунок 1.8 – Вікно вибору маршруту

На ньому є такі елементи:

* Таблиця «Виберіть маршрут:» (тільки для читання);
* Кнопка «Назад»;
* Кнопка «Замовити квиток».

Кнопка «Назад» закриває це вікно.

Для вибору маршруту достатньо натиснути на будь-яку клітинку відповідного рядка таблиці «Виберіть маршрут:». Після натиснення кнопки «Замовити квиток» це вікно згортається і відкривається вікно замовлення квитка (рис. 1.9).

Рисунок 1.9 – Вікно замовлення квитка

На ньому є такі елементи:

* Таблиця вибраного квитка (тільки для читання);
* Поле введення імені;
* Поле введення прізвища;
* Поле введення номеру та серії документа;
* Поле, що відображає вибране місце;
* Кнопка «Вибрати»;
* Кнопка «Назад»;
* Кнопка «Замовити квиток».

Кнопка «Назад» закриває це вікно та відкриває згорнуте раніше вікно вибору маршруту (рис. 1.8), де залишилися попереднє відібрані маршрути.

Кнопка «Замовити квиток» перевіряє, чи заповнені всі поля, і, якщо все так, викликає метод, що додає дані про цей квиток до посадкової відомості та оновлює її вікно, якщо воно відкрите; крім того, ця кнопка показує інформаційне вікно (рис. 1.10), де відображається унікальний номер квитка, за допомогою якого користувач зможе самостійно його повернути.

Рисунок 1.10 – Інформаційне вікно з унікальним номером квитка

Кнопка «Вибрати» відкриває вікно вибору місця (рис. 1.11).

Рисунок 1.11 – Вікно вибору місця

У цього вікна є такі елементи:

* Кнопка «Назад»;
* Кнопка «Підтвердити»;
* Розкриваючийся список з номерами доступних місць.

Кнопка «Назад» закриває це вікно.

Кнопка «Підтвердити» закриває це вікно і повертає у поле відображення місця вікна замовлення квитка (рис. 1.9) номер місця, яке вибрав користувач.

Якщо у головному вікні диспетчера (рис. 1.4) користувач у меню вибере «Ціни на квитки» – «Відкрити», відкриється форма встановлення цін на квитки (рис. 1.12).

Рисунок 1.12 – Вікно встановлення цін на квитки

Це вікно має такі елементи:

* Таблиця встановлення цін на квитки;
* Кнопка «Назад»;

Кнопка «Назад» закриває це вікно.

Якщо у меню вибрати «Пароль» – «Змінити», відкриється вікно зміни паролю (рис. 1.13).

Рисунок 1.13 – Вікно зміни паролю

Це вікно має такі елементи:

* Поле введення поточного паролю;
* Поле введення нового паролю;
* Поле введення для перевірки нового паролю;
* Кнопка «Назад»;
* Кнопка «Підтвердити».

Кнопка «Назад» закриває це вікно.

Кнопка «Підтвердити» перевіряє, чи всі поля заповнені, і, якщо так, порівнює поточний пароль з паролем, зчитаним з файлу, порівнює новий пароль та його перевірку, і, якщо все співпадає, записує у файл новий пароль.

1. ПРОЕКТНА СПЕЦИФІКАЦІЯ
   1. Діаграма класів

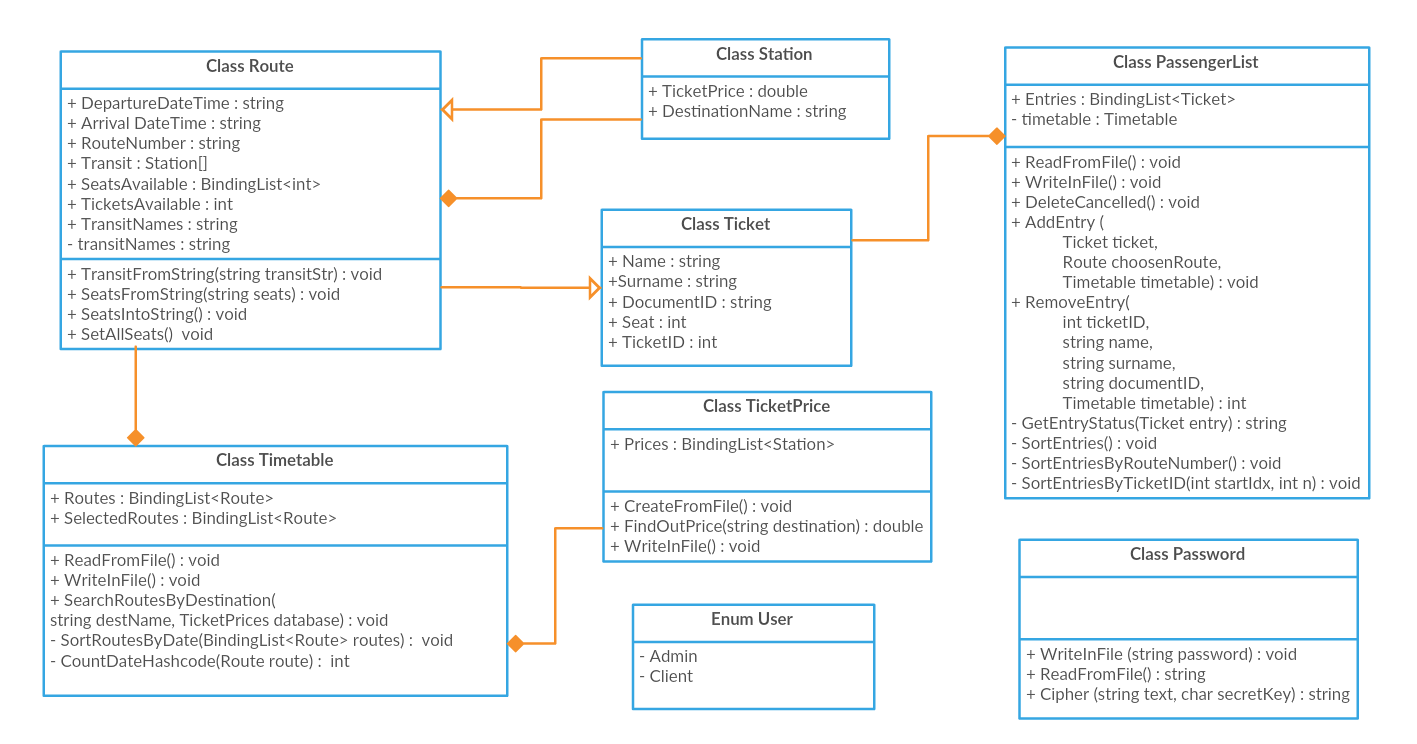


Рисунок 2.1 – Діаграма класів

* 1. Об’єктна структура програми

Програма складається з 7 класів, 1 перечислення і 9 форм. Усі форми були розглянуті у попередньому розділі.

На рисунку 2.2 наведено архітектуру проекту.

Рисунок 2.2 – Архітектура проекту

2.2.1 Клас Directory

Клас Directory описує довідник абітурієнта. Складається з двох відкритих властивостей:

//Список університетів.

public List<University> Universities { private set; get; }

//Список абітурієнтів.

public List<Applicant> Applicants { private set; get; }

Клас Directory має чотири методи (див. додаток А):

//Заповнення тестовых даних.

public void FillTestData(int n) – заповнює довідник абітурієнта тестовими даними.

public void Save() – звертається до методу, який зберігає дані в файлі .bin.

public void Load() – звертається до методу, який загружає дані з файлу.

2.2.2 Клас University

Клас описує університет з наступними відкритими властивостями:

//Адреса.

public string Adress { set; get; }

//Зображення.

public Image Image { set; get; }

//Назва.

public string Name { set; get; }

//Список спеціальностей.

public List<Spetiality> Spetialities { get; set; }

2.2.3 Клас Spetiality

Клас Spetiality описує спеціальність. Він має наступні властивості:

//Назва.

public string Name { set; get; }

//Ціна.

public decimal Price { set; get; }

//Прохідний бал минулого року.

public decimal Contest { set; get; }

//Бюджетні місця.

public decimal StateFundedPlaces { set; get; }

//Місця на денну форму.

public decimal DayPlaces { set; get; }

//Місця на заочну форму.

public decimal СorrespondencePlaces { set; get; }

//Всього місць.

public decimal Places { set; get; }

2.2.4 Клас Applicant

Клас Applicant описує абітурієнта та є спадкоємцем абстрактного класу User, у нього з’являються такі властивості:

//Ім'я пасажира.

public string Name { get; set; }

//Прізвище пасажира.

public string Password { get; set; }

2.2.5 Клас Admin

Клас Admin описує адміністратра та наслідує клас User.

Також цей клас має такі методи (див. додаток В):

* public void CreateFromFile() – зчитує список цін з файлу JSON;
* public double FindOutPrice(string destination) – знаходить станцію у списку цін і повертає ціну квитка до неї;
* public void WriteInFile() – записує список цін у файл JSON.

2.2.6 Клас Dao

Клас Dao потрібен для збереження та загрузки даних. Складається з наступних методів:

public void Save() – зберігає дані в файлі .bin.

public void Load() – загружає дані з файлу.

1. ІНСТРУКЦІЯ КОРИСТУВАЧА
   1. Встановлення, запуск та головне вікно

Для користування програмою її необхідно завантажити на ваш комп’ютер. Після цього необхідного відкрити файл за шляхом \Bus-station-tickets\Bus-station-tickets\bin\Debug\Bus-station-tickets.exe.

Після запуску програми відкривається головне вікно (рис. 1.2).

У ньому ви можете переглянути розклад маршрутів автобусів, повернути квиток та відшукати маршрут до введеного вами пункту призначення.

* 1. Вибір маршруту та замовлення квитка

Якщо введений вами пункт призначення є у розкладі і всі рейси до нього відправляються після поточної дати та часу, після натиснення кнопки «Пошук маршруту» відкриється вікно вибору маршруту (рис. 1.8).

Щоб обрати маршрут, виберіть будь-яку клітинку у таблиці, що відноситься до нього, і натисніть кнопку «Замовити квиток». Після цього відкриється вікно замовлення квитка (рис. 1.9).

Тут необхідно заповнити всі поля вашими персональними даними. Їх відповідність документу, що посвідчує особу, та його номер, мають перевірятися у касі автовокзалу перед безпосередньою оплатою та врученням квитка.

Вибір місця здійснюється у окремому вікні (рис. 1.11).

Після перевірки усіх даних натисніть кнопку «Замовити квиток».

Ви побачите інформаційне вікно (рис. 1.10), у якому буде наведено ваш унікальний номер квитка. Він вам знадобиться, якщо буде необхідно повернути квиток самостійно.

* 1. Повернення квитка

Щоб перейти до вікна повернення квитка, у головному вікні (рис. 1.2, 1.4) натисніть кнопку «Повернути квиток». Відкриється вікно повернення квитка (рис. 1.7).

Щоб повернути квиток, введіть усі персональні дані, вказані при його замовленні, та унікальний номер квитка, згенерований для вас. Після цього натисніть кнопку «Повернути квиток».

* 1. Авторизація та доступ диспетчера

Щоб авторизуватися, натисніть на кнопку «Авторизація» у головному вікні програми (рис. 1.2). Після цього відкриється вікно авторизації (рис. 1.3).

За замовчуванням пароль диспетчера – «1234». Його можна буде змінити.

Якщо пароль введено правильно, у головному вікні з’являються нові функції – можливість редагувати розклад, перегляд посадкової відомості, встановлення цін на квитки та зміна паролю (рис. 1.4).

Під час редагування таблиці розкладів слід пам’ятати про наступне:

* Номер рейсу має бути натуральним числом (> 0);
* Назва пункту призначення не може бути пустою;
* Проміжні пункти перелічуються через кому без пробілів;
* Дата і час відправлення не може бути більше або дорівнювати даті і часу прибуття;
* Кількість вільних місць встановлюється автоматично.

Під час відкриття посадкової відомості з’являється вікно, що відображає її (рис. 1.6).

Під час внесення змін у розкладі у вже існуючі маршрути, усі квитки на них набувають статусу «Скасовано». Щоб швидко видалити їх з відомості, ви можете натиснути кнопку «Видалити скасовані». Більше ніяк напряму маніпулювати посадковою відомістю ви не можете (вона оновлюється під час замовлення/повернення квитків).

Після відкриття цін на квитки з’являється вікно, у якому ви можете редагувати їх та встановлювати нові (рис. 1.12).

Назва пункту призначення не може бути пустою, а ціна менше або дорівнювати нулю.

Якщо ви вибрали зміну паролю, відкривається відповідне вікно (рис. 1.13). Якщо всі дані введені правильно, пароль змінюється.

Щоб повернутися до клієнтського доступу до функцій програми необхідно знову натиснути кнопку «Авторизація» на головному вікні (рис. 1.4) та після цього натиснути кнопку «Продовжити як клієнт» (рис. 1.5).

Щоб вийти з програми, натисніть кнопку «Вихід» у головному вікні (рис. 1.2, 1.4).

**ВИСНОВКИ**

У ході виконання курсової роботи були освоєні методи створення програмних систем на засадах об’єктно-орієнтованого програмування, набулися навички роботи з програмною системою, створення зв’язків між об’єктами.

У результаті отримана програма під назвою «Довідник абітурієнта», яка дозволяє звичайному користувачу переглядати заклади вищої освіти, продивлятися інформацію про університети, додавати та видаляти університети зі Списку обраних. Авторизований користувач (адміністратор) може змінювати інформацію щодо ЗВО, видаляти вже існуючі університети та додавати нові.

Програма написана на мові програмування C# на платформі .NET Framework 4.5 з використанням технологій Windows Forms у середі розробки Microsoft Visual Studio 2017.

Усі дані програми зберігаються локально у файла .bin. Програма може бути поліпшена застосуванням справжньої бази даних, яка синхронізується з базою даних університетів по мережі.

Галузь застосування – вища освіта.

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. Полное руководство по языку программирования С# 7.0 и платформе .NET 4.7 [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/tutorial/.
2. Бондарев В. М. Объектно-ориентированное программирование на С# [Текст] : учеб. Пособие / В. М. Бондарев. – Харьков : СМИТ, 2009. –224 с. – Библиогр. : ISBN 978-966-2028-34-8.
3. Microsoft Developer Network. Библиотека MSDN / MSDN – сеть разработчиков Microsoft [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/ru-ru/.
4. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд. [Текст]: пер. с англ./ Э. Троелсен.- М.: Вильямс, 2013 - 1312 с. – ISBN 978-5-8459-1814-7, 978-1-43-024233-8.
5. Либерти Дж. Программирование на С# [Текст] : пер. с англ. / Дж. Либерти. – 2-е изд. – СПб. ; М. : Символ-Плюс, 2003. – 688 с.

ДОДАТОК А  
Методи класу Route

//Метод, що створює масив проміжних пунктів (станцій) з рядка.

public void TransitFromString(string transitStr)

{

if (transitStr != null)

{

string[] transitStrArr = transitStr.Split(',');

Transit = new Station[transitStrArr.Length];

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

Transit[i] = new Station(transitStrArr[i]);

}

}

//Метод, що створює список вільних місць з рядка.

public void SeatsFromString(string seats)

{

SeatsAvailable = new BindingList<int>();

string[] seatsArr = seats.Split(',');

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

{

SeatsAvailable.Add(Int32.Parse(seatsArr[i]));

}

SeatsAvailable = new BindingList<int>(

SeatsAvailable.OrderBy(i => i).ToList());

}

//Метод, що записує список вільних місць у рядок.

public string SeatsIntoString()

{

string seatsStr = "";

SeatsAvailable = new BindingList<int>(

SeatsAvailable.OrderBy(i => i).ToList());

foreach (int seat in SeatsAvailable)

{

seatsStr += seat + ",";

}

return seatsStr.TrimEnd(',');

}

//Метод, що створює початковий список місць.

public void SetAllSeats()

{

SeatsAvailable = new BindingList<int>();

for (int i = 1; i <= 49; i++)

{

SeatsAvailable.Add(i);

}

}

ДОДАТОК Б  
Методи класу Timetable

//Метод, що зчитує розклад з файлу.

public void ReadFromFile()

{

Routes = new BindingList<Route>();

string[,] timetable =

JsonConvert.DeserializeObject<string[,]>

(Encoding.UTF8.GetString

(File.ReadAllBytes("Timetable.json")));

if (timetable == null)

return;

for (int i = 0; i <= timetable.GetUpperBound(0); i++)

{

Route route = new Route(

Int32.Parse(timetable[i, 0]), timetable[i, 1],

timetable[i, 2], timetable[i, 3], 0,

timetable[i, 4], timetable[i, 5]);

Routes.Add(route);

}

}

//Метод, що записує розклад у файл.

public void WriteInFile()

{

string[,] timetable = new string[Routes.Count, 6];

int i = 0;

foreach (Route route in Routes)

{

timetable[i, 0] = route.RouteNumber.ToString();

timetable[i, 1] = route.DestinationName;

timetable[i, 2] = route.DepartureDateTime;

timetable[i, 3] = route.ArrivalDateTime;

timetable[i, 4] = route.TransitNames;

timetable[i, 5] = route.SeatsIntoString();

i++;

}

File.WriteAllText("Timetable.json",

JsonConvert.SerializeObject(timetable));

}

//Метод, що шукає рейс за пунктом призначення.

public void SearchRoutesByDestination

(string destName, TicketPrice database)

{

SelectedRoutes = new BindingList<Route>();

Route FoundRoute;

int stop;

double price = database.FindOutPrice(destName);

Route route;

DateTime currentDateTime = DateTime.Now;

DateTime departing = DateTime.MinValue;

for (int i = 0; i < Routes.Count; i++)

{

route = this[i];

if (route.Transit == null)

{

stop = 1;

}

else

{

stop = route.Transit.Length;

}

for (int j = 0; j < stop; j++)

{

departing =

DateTime.Parse(route.DepartureDateTime);

if (route.TicketsAvailable != 0 &&

departing > currentDateTime &&

(route.DestinationName == destName ||

route.Transit != null &&

route.Transit[j].DestinationName == destName))

{

FoundRoute = new Route(route.RouteNumber,

destName, route.DepartureDateTime,

route.ArrivalDateTime, price,

null, route.SeatsIntoString());

SelectedRoutes.Add(FoundRoute);

break;

}

}

SortRoutesByDate(SelectedRoutes);

}

}

private void SortRoutesByDate(BindingList<Route> routes)

{

Route swap;

int n = routes.Count;

for (int j = n; j > 1; j--)

{

for (int i = 1; i < j; i++)

{

if (CountDateHashcode(routes[i]) <

CountDateHashcode(routes[i - 1]))

{

swap = routes[i];

routes[i] = routes[i - 1];

routes[i - 1] = swap;

}

}

}

}

private int CountDateHashcode(Route route)

{

int month = Int32.Parse

(route.DepartureDateTime.Substring(9));

int day = Int32.Parse

(route.DepartureDateTime.Substring(6, 2));

int hours = Int32.Parse

(route.DepartureDateTime.Substring(0, 2));

int minutes = Int32.Parse

(route.DepartureDateTime.Substring(3, 2));

return

month \* 1000000 + day \* 10000 +

hours \* 100 + minutes \* 10;

}

ДОДАТОК В  
Методи класу TicketPrice

//Метод, що зчитує список цін з файлу.

public void CreateFromFile()

{

Prices = new BindingList<Station>();

string[,] prices =

JsonConvert.DeserializeObject<string[,]>

(Encoding.UTF8.GetString(

File.ReadAllBytes("TicketPrices.json")));

if (prices == null)

return;

for (int i = 0; i <= prices.GetUpperBound(0); i++)

{

Station entry = new Station(

prices[i, 0], Double.Parse(prices[i, 1]));

Prices.Add(entry);

}

}

//Метод, що знаходить станцію у списку цін.

//Повертає ціну квитка до станції.

//Ціна за змовчуванням: 10.

public double FindOutPrice(string destination)

{

double price = 10;

foreach (Station entry in Prices)

{

if (entry.DestinationName == destination)

{

price = entry.TicketPrice;

break;

}

}

return price;

}

//Метод, що записує список цін у файл.

public void WriteInFile()

{

string[,] prices = new string[Prices.Count, 2];

int i = 0;

foreach (Station entry in Prices)

{

prices[i, 0] = entry.DestinationName;

prices[i, 1] = entry.TicketPrice.ToString();

i++;

}

File.WriteAllText(

"TicketPrices.json",

JsonConvert.SerializeObject(prices));

}

ДОДАТОК Г  
Клас Password

public static class Password

{

//Метод, що шифрує пароль та зберігає його у файл.

public static void WriteInFile

(string password)

{

File.WriteAllText("Password.json",

JsonConvert.SerializeObject(

Cipher(password, '•')));

}

//Метод, що зчитує пароль з файлу.

public static string ReadFromFile()

{

string password =

JsonConvert.DeserializeObject<string>

(Encoding.UTF8.GetString(

File.ReadAllBytes("Password.json")));

if (password == null)

password = "";

return Cipher(password, '•');

}

private static string Cipher

(string text, char secretKey)

{

string res = "";

for (var i = 0; i < text.Length; i++)

{

res +=

((char)(text[i] ^ secretKey)).ToString();

}

return res;

}

}

ДОДАТОК Д  
Методи класу PassengerList

//Метод, що зчитує посадкову відомість з файлу.

public void ReadFromFile()

{

Entries = new BindingList<Ticket>();

string[,] entries =

JsonConvert.DeserializeObject<string[,]>

(Encoding.UTF8.GetString

(File.ReadAllBytes("PassengerList.json")));

if (entries == null)

return;

for (int i = 0; i <= entries.GetUpperBound(0); i++)

{

Ticket entry = new Ticket();

entry.RouteNumber = Int32.Parse(entries[i, 0]);

entry.DestinationName = entries[i, 1];

entry.Name = entries[i, 2];

entry.Surname = entries[i, 3];

entry.DocumentID = entries[i, 4];

entry.DepartureDateTime = entries[i, 5];

entry.ArrivalDateTime = entries[i, 6];

entry.TicketPrice = Int32.Parse(entries[i, 7]);

entry.Seat = Int32.Parse(entries[i, 8]);

entry.Status = GetEntryStatus(entry);

Entries.Add(entry);

}

}

//Метод, що записує посадкову відомість у файл.

public void WriteInFile()

{

string[,] entries = new string[Entries.Count, 10];

int i = 0;

foreach (Ticket entry in Entries)

{

entries[i, 0] = entry.RouteNumber.ToString();

entries[i, 1] = entry.DestinationName;

entries[i, 2] = entry.Name;

entries[i, 3] = entry.Surname;

entries[i, 4] = entry.DocumentID;

entries[i, 5] = entry.DepartureDateTime;

entries[i, 6] = entry.ArrivalDateTime;

entries[i, 7] = entry.TicketPrice.ToString();

entries[i, 8] = entry.Seat.ToString();

entries[i, 9] = entry.Status;

i++;

}

File.WriteAllText

("PassengerList.json",

JsonConvert.SerializeObject(entries));

}

//Метод, що видаляє усі квитки на скасовані рейси.

public void DeleteCancelled()

{

for (int i = 0; i < Entries.Count; i++)

{

if (GetEntryStatus(Entries[i]) == "Скасовано")

{

Entries.RemoveAt(i);

i--;

}

}

WriteInFile();

}

//Метод, що добавляє запис у відомість.

public void AddEntry

(Ticket ticket,

Route choosenRoute,

Timetable timetable)

{

var allRoutes = timetable.Routes;

Entries.Add(ticket);

WriteInFile();

for (int i = 0; i < allRoutes.Count; i++)

{

if (allRoutes[i].RouteNumber ==

choosenRoute.RouteNumber)

{

allRoutes[i].SeatsAvailable.Remove(ticket.Seat);

timetable.WriteInFile();

}

}

}

//Метод, що видає запис з відомості.

public int RemoveEntry(

int ticketID,

string name,

string surname,

string documentID,

Timetable timetable)

{

int seat = 0;

int ticketIndex = -1;

int routeNumber = 0;

foreach (Ticket entry in Entries)

{

if (ticketID == entry.TicketID &&

name == entry.Name &&

surname == entry.Surname &&

documentID == entry.DocumentID)

{

ticketIndex = Entries.IndexOf(entry);

seat = entry.Seat;

routeNumber = entry.RouteNumber;

break;

}

}

if (ticketIndex == -1)

{

return ticketIndex;

}

foreach (Route route in timetable.Routes)

{

if (route.RouteNumber == routeNumber &&

GetEntryStatus(Entries[ticketIndex]) !=

"Скасовано")

{

route.SeatsAvailable.Add(seat);

MainForm.MainTimetable.WriteInFile();

break;

}

}

Entries.RemoveAt(ticketIndex);

WriteInFile();

return ticketIndex;

}

private string GetEntryStatus(Ticket entry)

{

bool hasFound = false;

string status = "";

DateTime currentDateTime = DateTime.Now;

DateTime arriving =

DateTime.Parse(entry.ArrivalDateTime);

DateTime departing =

DateTime.Parse(entry.DepartureDateTime);

for (int i = 0; i < timetable.Routes.Count; i++)

{

if (entry.RouteNumber ==

timetable[i].RouteNumber &&

entry.DepartureDateTime ==

timetable[i].DepartureDateTime &&

entry.ArrivalDateTime ==

timetable[i].ArrivalDateTime)

{

if (entry.DestinationName !=

timetable[i].DestinationName)

{

for (int j = 0; j < timetable[i].Transit.Length; i++)

{

if (entry.DestinationName ==

timetable[i].Transit[j].DestinationName)

{

hasFound = true;

break;

}

}

}

else

{

hasFound = true;

break;

}

}

}

switch (hasFound)

{

case true:

if (currentDateTime < departing)

{

status = "Замовлено";

}

else if (currentDateTime < arriving)

{

status = "У дорозі";

}

else

{

status = "Прибув";

}

break;

case false:

status = "Скасовано";

break;

}

return status;

}

private void SortEntries()

{

if (Entries.Count == 0) return;

SortEntriesByRouteNumber();

int routeNumber = Entries[0].RouteNumber;

int entriesCount = 0;

int startIdx = 0;

for (int i = 0; i < Entries.Count; i++)

{

if (routeNumber == Entries[i].RouteNumber)

{

entriesCount++;

}

else

{

SortEntriesByTicketID(startIdx, entriesCount);

entriesCount = 0;

startIdx = i;

routeNumber = Entries[i].RouteNumber;

}

}

}

private void SortEntriesByRouteNumber()

{

Ticket swap;

int n = Entries.Count;

for (int j = n; j > 1; j--)

{

for (int i = 1; i < j; i++)

{

if (Entries[i].RouteNumber <

Entries[i - 1].RouteNumber)

{

swap = Entries[i];

Entries[i] = Entries[i - 1];

Entries[i - 1] = swap;

}

}

}

}

private void SortEntriesByTicketID(int startIdx, int n)

{

if (n == 1) return;

Ticket swap;

for (int j = n; j > startIdx + 1; j--)

{

for (int i = startIdx + 1; i < j; i++)

{

if (Entries[i].TicketID <

Entries[i - 1].TicketID)

{

swap = Entries[i];

Entries[i] = Entries[i - 1];

Entries[i - 1] = swap;

}

}

}

}